Development of Synthetic Biology Biosensors using High-Level Modularity and Multi-Microbial Systems

Bradley Brown

Submitted for the degree of Doctor of Philosophy

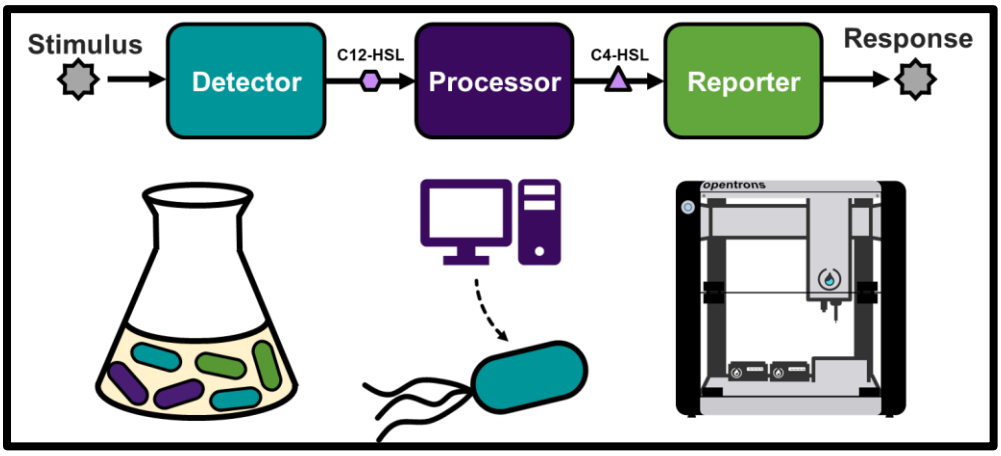
Newcastle University

School of Engineering

November 2022

Abstract

Synthetic biology devices have proven to have a wide variety of applications, especially in the field of biosensors where pollutants, biomarkers, and a range of other stimuli can be detected. However, development of biosensor devices can be difficult due to issues surrounding optimisation, and inefficient use of engineering principles such as re-usability, standardisation, and modularity. The work presented in this thesis aimed to investigate whether biosensor development could be aided by a framework which combines high-level modularity and multi-microbial systems. Previous work shows how high-level modularity could be used to develop biological devices but stop short of defining a framework which makes use of engineering principles. Building on previous work, biosensor designs were split into three module types, each of which could be implemented in separate cells and co-cultured to create the biosensor. Tools and resources were researched and developed with the aim of promoting the use of engineering principles within the framework. A pre-existing data standard was extended to allow for standard representation of multi-microbial systems. Additionally, a Python library was developed to allow for trivial and flexible generation of reproducible automation protocols for biosensor characterisation and a range of other synthetic biology workflows. Approaches for optimising biosensors developed within the modular and multi-microbial framework were investigated using computationally informed experimentation. Finally, an approach at implementing light-based intercellular communication is presented.



COVID-19 Impact Statement

This completion of this thesis was impacted by the COVID-19 pandemic in three major ways.

Firstly, due to the lockdowns and restrictions placed on working, I was not able to fully access the laboratory for the majority of my third year. I instead worked in an alternate laboratory which was limited in terms of equipment, and a lack of desk space for computer work made parallel-working difficult. These limitations meant that experiments took longer to perform than before restrictions were put in place.

Secondly, supply chain issues continued to impact me through to the end of my PhD. Most notably, 6-to-8-month lead times on 96-well plates reduced the number of plate reader experiments I was able to perform, preventing the performance of additional Design of Experiments runs, characterisation of cell ratios, and further testing of BiomationScripter functions and protocols. Additionally, I was not able to purchase PDMS for fabrication of microfluidic chips, as lead times were approximately 12 months. Therefore, silicone resin had to be used instead, which allowed for basic characterisation of the microfluidic devices but prevented testing of optical communication.

Finally, in the final 3 months of my PhD I had symptomatic COVID, during which I was unable to work at full capacity and had to isolate for 3 weeks, which delayed writing of this thesis.

Declaration

Any and all work presented in this thesis was my own unless explicitly stated otherwise. It has not been submitted in any capacity for another degree at Newcastle University or any other institution.

Bradley Brown

31st November 2022

Publications

The following work was published during the course of this PhD:

- **Multicolor Plate Reader Fluorescence Calibration:** Jacob Beal *et al.*, 2022, OUP Synthetic Biology, doi:10.1093/synbio/ysac010
- **Capturing Multicellular System Designs Using the Synthetic Biology Open Language (SBOL):** Bradley Brown, *et al*., 2020, ACS Synthetic Biology, doi:10.1021/acssynbio.0c00176
  * Pre-print available at BioRxiv: doi:10.1101/463844

Acknowledgements

There are many people I would like to thank who in some way helped in the completion of this thesis. Firstly, I would like to thank my supervisors in Newcastle, Dr Dana Ofiteru and Professor Anil Wipat, and my supervisor at NUS Singapore, Dr Chueh Loo Poh. Your guidance, suggestions, support, and overall kindness helped shape this project and made these last years more enjoyable. I would also like to thank the members of Professor Chueh Loo Poh's team for welcoming me into their group when I visited Singapore, and for the interesting discussion we had there. I would especially like to thank Jingyun for the thoughtful discussions on the optical communication aspect of the project, and in teaching me how to set up optogenetics experiments.

To my colleagues in Newcastle, whether you have been there from the start of my PhD journey or whom I have only known for a short time, thank you for all of your support and making this time a lot more fun. I'd especially like to thank Chris, Bill, David, James (fake), James (real), Wendy, Nadia, Polly, Lucy, and Valeria. I'm also grateful to the students and iGEM teams I've helped supervise or demonstrate for over the years; your enthusiasm for science helped drive me through the tough parts of the project. A big thanks goes out to friends in other groups who provided a lot of laughs. I would also like to thank my friends in Newcastle, Christian 'hwhat is machine learning' Atallah, Bill 'omg it's a Gible!' Yang, David 'without whom this thesis would have been completed a long time ago' Markham, Jasmine 'it was nice knowing you' Bird, Joshua Loh, Yanhua Zhong, Alex Laverick, and Alis Prusokas, and also to Dan 'on the z axis' Herring in Brum. To those who have joined in with the many Stellaris games we've played, I'll just say that I hope my research skills are better than my ability to win that game…

To my friends back home, especially Jake Walton, Danielle Nicholls, and Bethan Jones, I'm extremely grateful for your friendships over the many, many years we've known each other. Without you I'd be a very different person. To my family, my Mum (Jane), Dad (Gary), sister (Jess), brother (Luke), Nan (Val), and Grandad (Jim), I don't think I'll ever be able to express how much your constant support and love has helped me. Thanks also goes to my extended family, especially my Aunts and Uncles (John & Julia, Hylton & Amelia, and Robert & Hazel) for your general support and kindness. To Maisie, who was the best dog anyone could hope and sadly passed away during the writing of this thesis, I hope you're having the best time in doggy heaven. And finally, to my friend and partner Dr Jasmine E. Bird, thank you for everything. Without your support and stability, I wouldn't have been able to complete this PhD, and life would be a lot worse. I love you and am excited for the next stage of our lives.

Maisie Brown:

2011 - 2022

Table of Contents

Table of Figures

Chapter 1. Introduction and Background Information

1.1. Bioengineering

1.1.1. An overview of bioengineering

Bioengineering efforts tend to take advantage of mechanisms evolved by nature for solving a variety of problems. For example, organisms are required to detect a range of molecules in order to survive[1]. In some cases, detection of certain molecules can indicate the prey is nearby[2], [3], and in others it can indicate the direction of food[4]. The detection of metabolites is fundamental to an organism's survival, as too much or too little of primary and secondary metabolites can have serious consequences[5]. By detecting the level of these metabolites, pathways to degrade or produce that specific molecule can be regulated. It should be noted that these natural sensing mechanisms are not limited to the detection of molecules; there exist biological processes which deal with the detection of other stimuli such as temperature[6].

The detection of specific stimuli has application in many areas. For example, sensing of environmental pollutants can help guide remediation efforts[7], or the presence of certain chemicals in the blood stream of humans can indicate health problems and indicate medical interventions should occur[8]. In many cases, these molecules are difficult to detect using electronic or other manufactured sensors[9]. Instead, bioengineers look to discover the mechanisms already used in nature for detection of stimuli of interest and use those instead of non-biological sensing components. Such biological sensing devices are often termed as 'biosensors'[10].

Other areas in which biological components and systems can aid include the synthesis of organic products[11], the degradation of pollutants[12], and the production of microbial fuel cells[13]. To further illustrate the field of bioengineering, the following sub-sections focus on some examples and case studies.

1.1.2. Biosynthesis

Bioengineering can be used to enhance or replace chemical synthesis, where molecules with desired properties are created through a series of chemical reactions in a process known as total synthesis[14]. Molecules created via chemical synthesis can have wide ranging applications, including pharmaceuticals, pesticides, and food

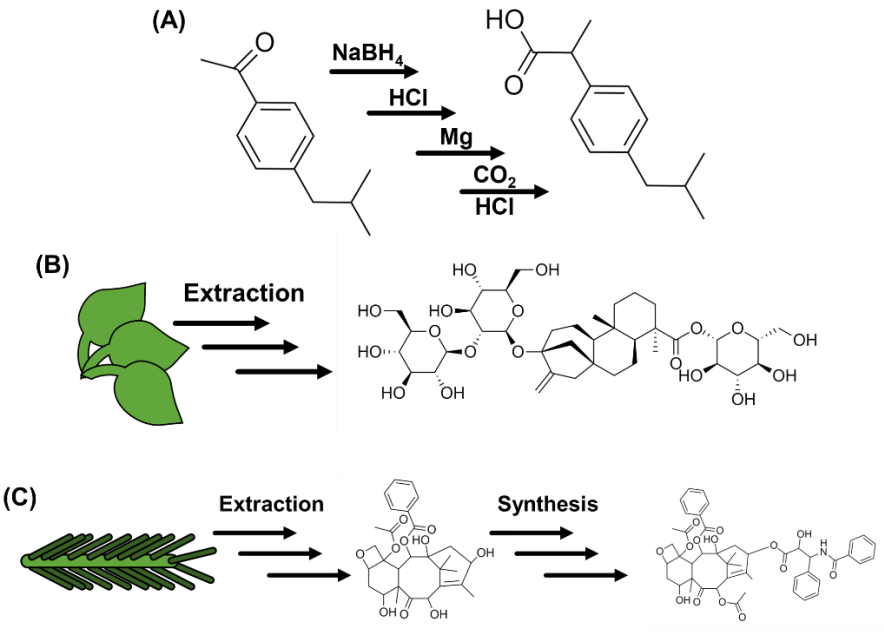


Figure 1.1. Overview of Synthesis Methods
Depicted here are three methods of synthesising molecules. All molecule images were generated using ChemDrawJs (www.chemdrawdirect.perkinelmer.cloud) by importing the SMILES structure code from PubChem. (A) An example of total synthesis, where Ibuprofen is synthesised in a series of 4 reactions from p-isobutylacetophenone. (B) An example of biosynthesis, where a steviol glycoside is extracted from the leaves of Stevia rebaudiana. (C) An example of semi-synthesis, where 10-deacetylbaccatin is extracted from Taxus baccata needles and used as a precursor in Paclitaxel chemical synthesis.

additives[15]. A simple example of total synthesis is the production of ibuprofen[16] (Figure 1.1 (A)). Unfortunately, it is not uncommon for many useful molecules and compounds to be chemically complex, and the procedures required to synthesise them from readily available precursors can be expensive, inefficient, and involve the use of harmful chemicals[17]. In some cases, organisms have evolved to produce the desired molecule, which can be extracted and purified. An example of this is the production of steviol glycosides, which are obtained from the leaves of the *Stevia rebaudiana* plant and used as sweeteners[18] (Figure 1.1 (B)). The total synthesis of these molecules is difficult due to high acid sensitivity and inefficient processes, which makes the biosynthesis of steviol glycosides in plants, followed by chemical extraction and purification techniques, a better alternative[19].

Biosynthesis and chemical synthesis can also be combined, as in the production of Paclitaxel[20] (Figure 1.1 (C)). Paclitaxel has a very complex structure, and it is not feasible to synthesise using chemical reactions from readily available precursors, however a precursor of Paclitaxel can be found in the needles of European Yew trees (*Taxus baccata*), and there is a relatively simple chemical synthesis process to convert

the precursor into Paclitaxel. In this way, the two methods of total synthesis and biosynthesis can be combined in a process known as semisynthesis.

1.1.3. Protein Engineering

Protein engineering is another example of bioengineering. Research in this area concerns itself with the modification of proteins to improve or modify the protein's function and behaviour, making it more appropriate for specific applications[21]. Examples of protein engineering can be seen in a variety of areas, including biosynthesis and semi-synthesis[22]. In these cases, the enzymes used in the synthesis pathways can be modified in a variety of ways, such as changing the enzyme's binding site to accept different substrates[23], making an enzyme more tolerable to industrial conditions[24], or attempting to increase the efficiency of the enzymes overall[25]. There are many techniques which can be employed to achieve this engineering[26]. These include rational redesign, where the sequence of the protein is modified based on existing structure-function knowledge[27], and directed evolution, where the protein of interest is randomly mutated and candidates are selected based on desired behaviour, such as the conversion of a specific molecule[28]. This process of directed evolution typically occurs in cycles, where candidates which show promise undergo further mutation and testing to refine the desired functionality.

1.1.4. Enzymatic biosensors

The development of biosensors has the most relevance within this thesis, and hence is discussed in more detail than the previous examples. As mentioned previously, nature has evolved a wide variety of mechanisms to detect the presence and absence of specific stimuli, which bioengineers can harness to develop biosensor devices. There are several formats these devices can take, but they are all designed to have the same high-level functionality: to sense a desired stimulus, or in some cases a group of stimuli, and respond in some way[29]. As well as sharing the same high-level functionality, biosensors of all types use the same abstracted mechanism of detection. The mechanism relies upon the stimulus of interest interacting with a biological element in such a way that a bio-reaction occurs. Changes caused by bio-reactions can be used downstream to generate a response[30].

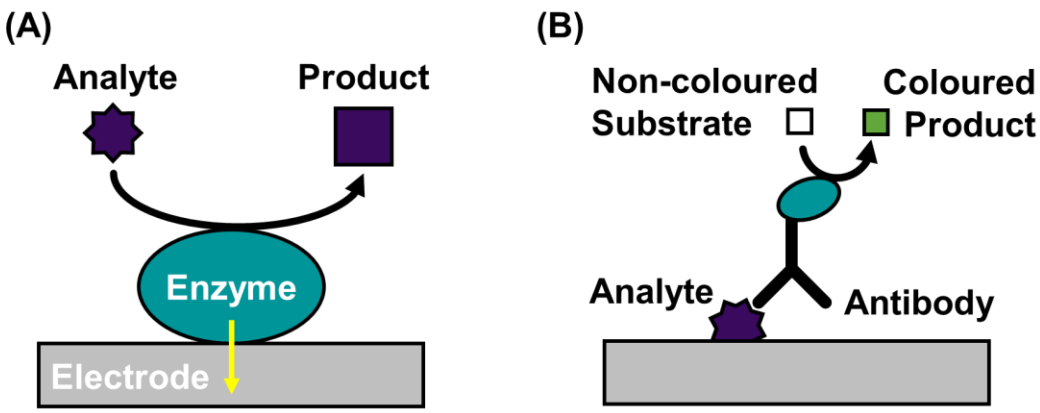


Figure 1.2. Overview of Enzymatic Biosensor Mechanisms
Two common mechanisms of enzymatic biosensing. (A) Generic schematic of an electrocatalytic biosensor. Conversion of the analyte to be detected into a different product by the enzyme generates free electrons. When the enzyme is in contact with an electrode, the electrons can flow from the enzyme and be detected electronically. (B) Schematic of direct ELISA. The analyte for detection is adsorbed to a surface and antibodies with binding capabilities to the analyte are added. After washing, antibodies bound to the analyte remain. An enzyme linked to the antibody catalyses a reaction which results in a colour change.

Early examples of biosensors generally relied upon *in vitro* biocatalysts which could biochemically react in the presence of a desired molecule[31]–[33] (Figure 1.2 (A)). As biochemical reactions can cause changes in the surrounding environment, it is possible to convert one of these changes into a desired response[33]. For example, the first generation of blood-glucose sensors relied on glucose oxidase as the biological sensing component[34]. The glucose oxidase enzyme has evolved to specifically bind glucose and convert it to gluconic acid. During this catalytic reaction, oxygen is depleted which results in a reduced concentration of dissolved oxygen. Therefore, as the concentration of glucose increases, the concentration of oxygen decreases, which means that the concentration of oxygen can be used as a proxy to the concentration of glucose. The concentration of oxygen in the biosensor device can be converted into a reading via an electrochemical oxygen sensor, which can be used as a human-readable response.

Whilst there are a number of success stories for biosensors which use enzymes to detect and report the presence of specific stimuli, the development of these sensors can be troublesome and slow. Development issues tend to stem from issues related to the enzymes themselves[35], [36]. Many enzymes in nature are not completely specific, and instead show some level of promiscuity towards their substrates[37]. This behaviour has advantages in natural systems, such as allowing a single mechanism to deal with similar situations[38]. However, for engineered biosensors, it is often desirable to

differentiate between very similar molecules or other stimuli[39]. It is also possible for no known enzymes to exist which act upon the desired stimulus, or for other issues to exist such as inactivation of the enzyme under conditions in which the biosensor needs to act, or difficulty producing the enzyme in industrially relevant quantities[40].

It is possible to use enzymes not only in stimuli detection, but also as the response section of a biosensor. For example, Enzyme-Linked Immunosorbent Assays, or ELISAs, rely on the use of antibodies to detect specific analytes, and enzymes to generate a response[41], [42] (Figure 1.2 (B)). Developed in the 1970s, ELISA assays remain a gold standard for detection in many fields[43]. There are different types of ELISA, however all types utilise antibodies as the sensing component of the sensor, and enzymes as the response component[44]. Generally, antibodies known to bind the substrate of interest are added to a sample, and several washing steps occur. During these washing steps, the antibodies are only retained if the sample was present. In some types of ELISA, the antibodies used have been modified to link an enzyme to the non-binding end, which can then perform an enzymatic reaction, usually resulting in a colour change, to indicate presence of the analyte. In other types of ELISA, secondary antibodies capable of binding either the first antibody or the substrate are instead modified to contain the enzyme responsible for generating a response. There are additional variants of ELISA which use additional antibodies; however the principle remains the same.

The primary issue during development of ELISAs for novel targets is the generation of capture antibodies able to bind specifically to the analyte of interest. The development of new antibodies, which is required when an antibody with desired binding properties does not exist, is non-trivial and can be costly and time consuming[42].

1.1.5. Nucleic acid biosensors

Another class of biosensors are those based on quantitative Polymerase Chain Reaction (qPCR). qPCR biosensors are used to detect nucleic acid molecules (both DNA and RNA) with a specific sequence, and to quantify the number of molecules encoding that sequence in a sample[45]. This functionality can help detect the presence or absence of a specific organism, such as a pathogen, by targeting DNA or RNA sequences which are specific to the species of interest[46]. It is also possible to use qPCR to measure gene expression levels by targeting the mRNA produced from the

desired gene[47]. The amount of mRNA in the sample is directly correlated to the expression levels of the gene. The idea behind qPCR is that primers can be designed to recognise and amplify a specific DNA sequence [48].

Whilst qPCR biosensors have proven incredibly useful, there are limitations. The principal limitation is in their ability to only detect nucleic acids as stimuli. It can also be difficult to develop these biosensors, as the primers must have specificity to the desired nucleic acid sequence to prevent false positives but must also have appropriate binding kinetics to the desired stimulus to ensure correct sensitivity and prevent false negatives[49].

1.1.6. Biosensor development and optimisation

As detailed above, each type of biosensor comes with its own advantages and limitations. Despite these differences, there are aspects common to any biosensor which should be addressed during the development stage, and indeed should be used to help decide the most appropriate biotechnology to use.

A biosensor's sensitivity and specificity tend to be crucial no matter the application[39]. In the case of environmental pollutant biosensors, the device should only react to the target chemical and should not respond to the presence of chemicals with similar structures which may not be toxic to the environment. In other cases, a biosensor which is too specific may be disadvantageous. This could be the case when developing a diagnostic test for a pathogen. If the test is too specific, then other strains of the pathogen which are still dangerous may be missed, resulting in false negatives. The sensitivity of the system is also important, as a sensor which is too sensitive may respond to background levels of a stimulus, which in some cases should be ignored, such as when a pollutant is harmless at lower concentrations, or the sensitivity could be too low and false negatives could be reported when the stimulus is present.

The specificity and sensitivity of a biosensor contribute to the signal-response curve[50], [51], which is a common characteristic of any biosensor (Figure 1.3). The signal-response curve describes the relationship between the input and output of a sensor and can be used to determine features of the system such as the limit of detection, the dynamic range, the operating range, and the background noise. The signal-response curve can be determined by measuring the response from the biosensor over a range

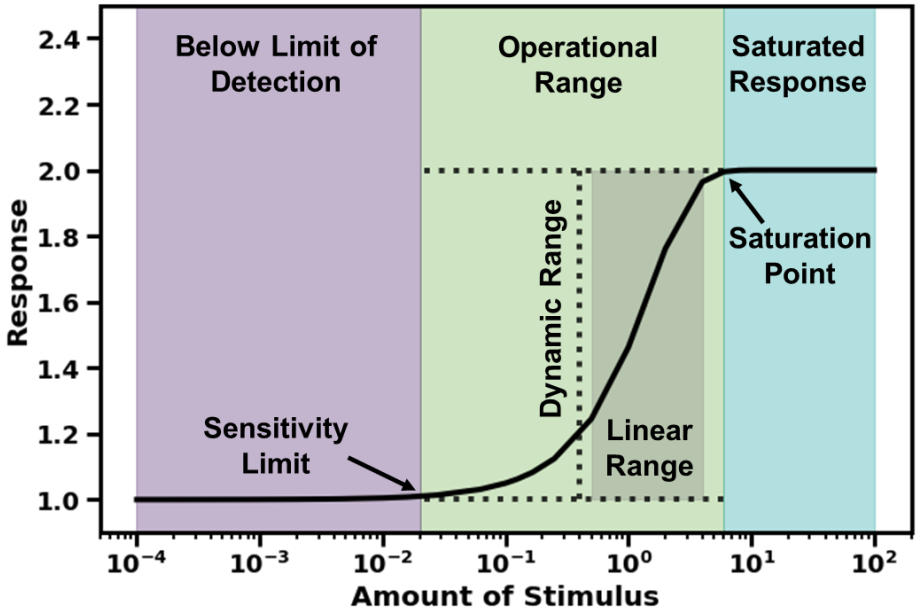


Figure 1.3. Dose-Response Curve

Annotated dose-response (or signal-response) curve. The highlighted regions illustrate stimulus ranges which are below the limit of detection, within the operational range, or which saturate the sensor's response. The grey box within the operational range identifies the linear range. The dynamic range is annotated with dotted black lines. The sensitivity limit and saturation point are labelled with arrows.

of stimuli amounts. Any response measured when no stimulus is present determines the background noise of the system. The lowest stimulus amount which produces a response above the background noise is the limit of detection (LOD), or sensitivity limit, for the biosensor. Anything below this amount cannot be distinguished from the general noise. At the other end of the scale, beyond the saturation point, the response does not continue to increase with increased amounts of the stimulus. The range between the LOD and maximal response point is defined as the operational range, where increased amounts of stimulus result in a differential response. Within the operational range is the linear range, where the biosensor's response increases proportionally with the stimulus amount. The difference between the response levels at the LOD and maximal response point is defined as the dynamic range and can be used to determine how much the response increases with each increase in concentration of the target molecule.

The methods for biosensor optimisation will be different depending on which technology is being used[52]. These methods could involve increasing the sensitivity of the detection mechanism or decreasing noise in the presence of similar targets in a sample by increasing specificity. The importance of the operating range and dynamic range are also influenced by the exact requirements of the biosensor. In some cases,

a binary ‘Yes’ or ‘No’ as to the presence of the target may be required, in which case it a large operating range may not be required. However, in other cases it may be essential to determine how much of a target is present across a range of concentrations, in which case the operating range must cover the entire range of important target amounts. The dynamic range is linked to the operating range, as it describes the difference in response between the lowest end of the operating range, and the highest. The exact response, along with how that response is measured or reported, will determine the importance and optimal values of the dynamic range. If the response is a colour change which will be observed by eye, like some types of ELISA-based biosensors, it is important to have a large dynamic range so that there is little doubt as to the intensity of a colour change.

Traditionally, the biosensor optimisation process has been somewhat limited, as it is not trivial to engineer an enzyme to tune the characteristics of an enzyme-based biosensor, or rapidly develop novel antibodies for use in ELISAs[41], [53], [54]. The rise of synthetic biology has helped to provide tools and techniques to assist with these efforts and has also introduced a whole new generation of biological devices[9]. In the next section, an overview of synthetic biology will be given, and examples will be used to discuss the impact it has had on bioengineering and to highlight the challenges which remain to be addressed.

1.2. Principles of Synthetic Biology

The rise of synthetic biology has been a gamechanger for the development of biodevices and bioengineering more generally, partly due to the adoption of engineering principles. Such principles promote the tried-and-tested methods of developing devices found in other engineering fields and has helped to provide a wealth of resources for synbio device developers[55]. Some of the most discussed engineering principles are standardisation, reproducibility, re-usability, computationally assisted modelling, modularity, and the engineering life cycle[56] (Figure 1.4).

1.2.1. Standardisation

The principles of standardisation, re-usability, and reproducibility are inherently linked. Standardisation, which refers to the development and implementation of standards, is used widely in fields such as electrical, mechanical, and software engineering, as well as architecture. The types of standards implemented vary between and within these fields, but generally any type of standard sets out to provide a common specification

for a commonly used entity or process to help ensure optimal efficiency is achieved, and that similar work completed within a field is reproducible, comparable, or re-usable, depending on the exact standard[57].

Within synthetic biology, standardisation is often associated with the physical assembly of genetic constructs, where several DNA assembly standards have been developed[58]–[60]. Assembly standards can define both a collection of genetic parts, like promoters, ribosome binding sites, and coding regions, and the methods used to assemble the individual parts into constructs. The use of assembly standards can allow researchers to rapidly assemble constructs[61]. Specific examples of DNA assembly standards and standard genetic parts are discussed in more depth in a future section.

Aside from DNA assembly, standardisation attempts have also been made for other areas, such as building computational models of biological systems[62] and calibrating experimental data to known chemical standards[63]. Another example is the Synthetic Biology Open Language (SBOL), which allows for information about synthetic biology designs and systems to be captured in a standard format[64].

1.2.2. Modularity

The principle of modularity can sometimes be conflated with that of standardised parts. However, whilst overlaps exist, there is still a distinction to be made. Modularity, as defined in other fields which make use of this concept, allows for the development of systems or devices through the combination of functional units[65]. These units, termed modules, work together to provide the overall behaviour of the system or device, but can also display independence from one another. Module independence falls into two types: functional and structural independence[66]. Most modules will display both types of independence to some degree. Functional independence, as might be inferred from the name, deals with the general characteristics of how a module confers its function. The extent to which a module can be thought of as functionally independent can be determined by considering firstly how discrete the overall function is, and secondly by how much the module relies on external elements to complete its function. Structural independence is similar to functional independence but refers instead to the physical attributes of the module. A module can be considered structurally independent if the individual units which compose the module are tightly coupled without reliance on

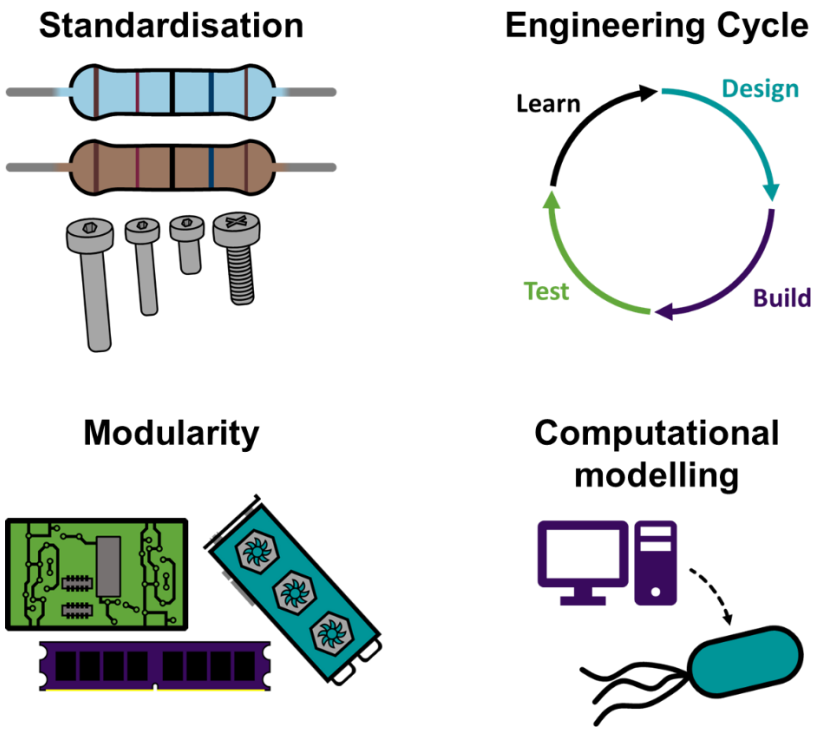


Figure 1.4. Engineering Principles in Synthetic Biology
Illustrated are engineering principles used in synthetic biology.

external elements, but also can be easily connected and disconnected from other modules in a system.

There are several advantages of using modules to develop a system or device. One of these advantages is the ability to split a design into discrete functions (each of which can be encapsulated by a module), which allows for complex systems to be developed by focusing on small sections independently and combining them once all sections are complete[65]. This approach of splitting a design into independent and discrete functional modules also has the advantage of allowing specific sections to be independently worked on by specialists in that area, without having to worry too much about the impact on the overall system. In addition, if modules are designed using standardised processes and with compatible connections, then module re-usability is possible, which helps to reduce the amount of time and resources spent on re-developing elements with identical functionalities for different systems. Finally, the separation of discrete functions into modules allows for the intricacies of how specific sections of a system works to be abstracted out into top-level functions. This abstraction allows developers to have less specialised and in-depth knowledge of all parts of a system and can instead focus more on the system as a whole and make use of modules which have been designed by specialists[67].

Modularity can be found in many areas, including software development, construction, and electronics[68]. Computer hardware is one such example of where modularisation has been successfully implemented. Typically, computers are developed using modular design, where each high-level function of a computer is developed as a discrete unit which can be connected to other modules to eventually form the entire system. Some examples of modular computing hardware are screens, microprocessors, storage devices, and motherboards. Each of these modules has a discrete function to perform: the screen should display graphical information to the user, the storage device records data which can be retrieved later, and so on. There are variations for each of these modules which can be swapped to modify the end product. For example, a portable laptop may require a small screen to keep the product small and easy to carry around, whereas a gaming computer could require a much larger, higher quality screen. Keeping with the theme of electronics, another example of a module are capacitors. Capacitors are electronic components which can store electrical charge, and have a wide variety of applications, including signal processing where the stored energy can be used to represent information. There are many different types of capacitors which act in different ways, and the most appropriate depends on the intended application.

The examples of modularity (computer hardware and electronics) given above have some definite similarities: they act as somewhat standardised discrete elements with a generalised function, and different variations of each module can be interchanged depending on the application. However, it perhaps is also not too difficult to see the differences. The functions performed by the computer components, such as ‘perform logical calculations’ or ‘display graphical information’, are more complex in nature than that of ‘store an electrical charge’. Additionally, the knowledge level required to effectively utilise these modules differs; computer components can, and often are, used by consumers with little theoretical or practical knowledge of electrical engineering to build their own computers, and in many cases are essentially ‘plug-and-play’. This contrasts with something like capacitors, where to effectively use these components in many cases requires a more in-depth working knowledge of electronics.

Based on the comparison between these examples of modules, it is possible to define two groupings of modules: high-level and low-level. The computer components mentioned here would be examples of high-level modules, as they have complex (but

still discrete) functions and require basic knowledge to use. Modules like capacitors would be examples of low-level modules, as they have much simpler and smaller functions, but tend to require a more in-depth knowledge to use to their full potential.

The hierarchy and grouping of modularity described here is a highly simplified version of the many theories of and approaches to modularisation which have been described and investigated previously[65], [66], [69]. However, the description given here is helpful in this work to differentiate between what has been achieved previously, and the approaches taken in this work. In the introductory section of chapter 3, specific examples of how modularity has been applied within synthetic biology are given.

1.2.3. Synthetic biology engineering cycle

Another widely cited principle in synthetic biology is the Design-Build-Test-Learn cycle (DBTL)[70]. This cycle is a form of the various engineering cycles and processes from other fields and industries which have been modified to better fit the engineering of biological systems[71]. The cycle is composed of four main stages which should be completed in sequence and repeated in an iterative manner, which allows for guided system development and can help ensure that appropriate information is generated to aid with optimisation in future iterations.

The first stage of the cycle is the design stage. The design stage itself can be thought of as a cycle, based heavily on the traditional engineering design process. Briefly, the design stage focuses on formulating a set of requirements that the final system/device should meet, generating a conceptual idea of the final product, creating a preliminary design, finalising the detailed design, and finally planning how the design will be built, implemented, and characterised[71]–[73]. A hallmark of synthetic biology, and another principle taken from other engineering fields, is the use of computational tools and modelling to assist with this design stage[74]. For example, CELLO is an online CAD-type tool which was developed to help design genetic circuits based on desired logic specifications[75]. Other tools, such as iBioSim[76] and Simbiotics[77] have been developed to help computationally model synthetic biology systems to help identify potential design issues, and to help predict expected behaviour prior to building and testing the system experimentally. These computational models can also help with informed design choices when several options exist, as the behaviour of each variation can be predicted prior to committing to a finalised design[78].

The build stage of the DBTL cycle is where the device or system is constructed according to plans formulated in the design stage. This very often involves the assembly of DNA encoding the genetic elements of the design[79], [80]. To aid with this stage, the DNA assembly standards mentioned previously can be utilised. Following DNA assembly is implementation of the built design. In synthetic biology research, implementation usually refers to the preparation of the built design so that it can be tested.

Implementation of a design can involve a range of steps, depending on the specifics of the system, the intended application, and the characterisation to be performed. One of the commonalities across implementation of synthetic biology designs is the involvement of a biological chassis. While there is some discourse about the term[81], here biological chassis (or just chassis) simply refers to something which is capable of using biological elements to execute an intended function. Where designs involve genetic circuits, at a minimum the chassis must be able to express the genetic elements which encode the biological system's function.

There are a wide range of possible chasses provided by nature. Whilst the large selection allows for researchers and developers to select a chassis with desirable properties, in reality only a select few organisms tend to be used[82]. This is mainly due to a combination of biological incompatibility of genetic parts, and a lack of tools and knowledge relating to non-model organisms. Biological incompatibility here refers to difference in behaviour of genetic parts in different species and strains. One example of this is codon bias, where different species produce tRNAs required for translation in different ratios[83]. This means that a codon which is commonly used in one organism may be rare in another, and so if a genetic part uses that codon often, overall expression levels are likely to be lower in one organism than the other[84]. Additionally, the presence of rare codons can exert stress on a cell through ribosome stalling, which results in fewer resources available for other processes[85]. The issue of codon bias specifically is well documented, and a range of tools exist to optimise the sequence of a genetic element towards a specific species[86]. This does mean, however, that parts may need to be re-synthesised to obtain the optimised part, which takes both time and money. Efforts have been made to try and optimise part sequences such that they can

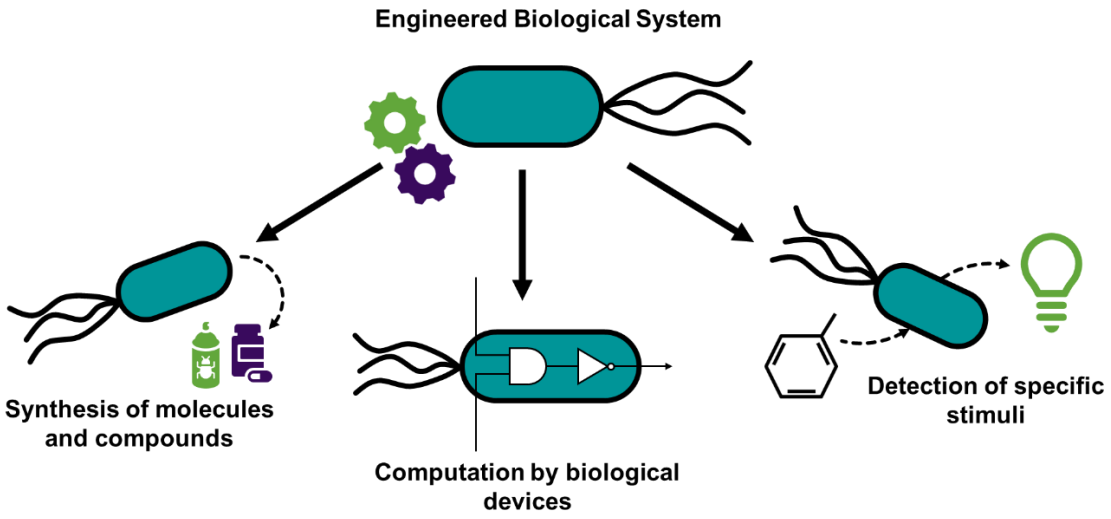


Figure 1.5. Overview of Synthetic Biology Applications
Illustrated are 3 applications of synthetic biology: the synthesis of molecules (metabolic engineering), performing logic and computation (biocomputing), and detection of stimuli (biosensors).

be used as-is in multiple species[87], however the success of such tools is yet to be determined.

Once a biological system or device has been built and implemented, its function can be validated and characterised. The exact characterisation performed will vary depending on the design specifications of that system and should be considered during the design stage[88]. Once again, researchers have developed methods, inspired by other engineering disciplines, of aiding this test stage. One example is the application of statistical Design of Experiments (DOE), which uses statistics and machine learning to help fully characterise a system in an informed and cost-effective way[89], [90].

During the final stage of the DBTL cycle, the learn stage, data collected from the test stage, along with experiences during the build stage, are used to determine whether the system or device meets the requirements set out in the design stage. If necessary, the data collected can be used to inform modifications to the design of the system in order to optimise behaviour/functionality, and the cycle is repeated in this iterative manner[91].

1.3. Synthetic Biology Applications

The availability of synthetic biology derived technologies, such as standardised biological parts, has allowed for novel systems to be developed with some level of predictability, and breakthroughs in genetic engineering techniques make it easier than

ever to modify natural systems towards specific needs. Whilst synthetic biology still has challenges ahead of it the impact it has had on bioengineering is undeniable. There have been revolutions in fields such as biocomputing, where biological logic circuits were designed and implemented to introduce computing functionality into biological systems[92], and research into the development of synthetic biology devices (synbio devices) has become increasingly popular[93]. Given in this section are examples of some devices and systems developed with the aid of synthetic biology approaches.

1.3.1. Metabolic engineering

The use of bioengineering in the production of useful molecules and compounds was discussed previously in section 1.1, where products produced naturally by organisms can be extracted and used as precursors in chemical synthesises, or natural enzymes are extracted and used *in vitro* to catalyse reactions not possible to perform by chemical means. With the application of synthetic biology, it is possible to instead engineer organisms themselves with the aim of enhancing natural production of desired molecules, or to build new biological pathways[94]. There are several approaches to this type of bioengineering, which typically fall under the branch of metabolic engineering. These include, but are not limited to, (i) engineering the host organism to knock-out or enhance genes involved in metabolism to increase flux through the desired pathway[95], (ii) expression of genes involved in biosynthesis of the metabolite of interest in a non-native host which has more desirable properties than the native host organism[96], and (iii) engineering of enzymes to increase their activity[97].

A common approach to metabolic engineering is to insert enzymes required for biosynthesis of a product into a host which is different to the organism which usually produces the molecule[94]. An advantage of this is that sometimes, the native organism is difficult to manipulate genetically, slow growing, or difficult to extract the product from[98], [99]. In these cases, moving the biosynthetic pathway to an organism which is easier to work with can be beneficial. Pathways which have been recreated in non-native organisms are often referred to as heterologous pathways.

One example where heterologous pathway recreation has been applied is in the production of artemisinic acid, which is an immediate precursor to the antimalarial drug Artemisinin[100]. Artemisinin itself is produced only in small quantities naturally, and its chemical synthesis pathway is complex and economically non-viable. It is possible to

obtain Artemisinin by chemically converting artemisinic acid, however artemisinic acid is also only obtained in low quantities naturally. Additionally, the organism which naturally produces artemisinic acid, *Artemisia annua*, is difficult to genetically engineer. Instead, researchers engineered *Saccharomyces cerevisiae* contain enzymes naturally found in *A. annua* in order to recreate the biosynthetic pathway in a host organism which could be more easily engineered. Using this approach, the heterologous biosynthetic pathway was optimised using genetic engineering approaches to produce artemisinic acid, which could then be converted into the antimalarial drug by chemical synthesis. Whilst this provided a method for more efficient production of artemisinin, scale up issues existed, and hence plate extraction of artemisinin is still common.

1.3.2. Biocomputing

One area which has been completely revolutionised by synthetic biology is that of biocomputing. This is not to say that biocomputing did not exist before the rise of synthetic biology; indeed, the concept of using biological molecules to perform computation predates synthetic biology by a few decades[101], and comparisons of the mechanisms of cells to mechanical machines dates back even further[102]. It could even be considered that efforts to harness biological systems for computation provided the initial inspiration for synthetic biology as field, given that the two papers most often referred to as initial landmark studies dealt with developing biological logic devices[103], [104].

Genetic-based biocomputing relies on genetic parts with known functionality, which are subsequently expressed by a biological chassis, such as *Escherichia coli* cells, yeast cells, or cell-free expression systems[105]. The genetic elements which compose the biocomputational functionality tend to be referred to as the genetic circuit. The mechanisms of these genetic circuits most often harness the power of natural regulatory systems found in cells, where expression of specific coding sequences (CDSs) can be controlled by certain biomolecules. For example, in one of the landmark studies mentioned above, the authors designed a genetic circuit which resulted in cells which could act as a toggle switch[104]. The abstracted function of this toggle switch was one which allowed a certain protein, in this case a green fluorescent protein (GFP), to be produced continually when a stimulus was applied (in this case the small

molecule IPTG). The production of the GFP could then be turned off by application of another small molecule (aTc).

Other examples of biological logical devices, some with much more complex mechanisms and functionality, have been developed over the past 2 decades[106]–[108]. The ability to build these devices has been heavily influenced by the wide availability of sequencing data, an increased understanding of how molecular mechanisms occur within cells, and genetic engineering/assembly techniques, as well as the presence of standardised genetic parts with characterised functions[79], [80], [105]. These parts can therefore be combined in somewhat predictable ways, although it should be noted that the issues which have plagued other areas, such as a lack of understanding as to how certain molecules and mechanisms interact, can lead to unforeseen behaviour[109]. This can result in devices with sub-optimal performance, or in some cases can be completely non-functional.

There are many other areas of research impacted by synthetic biology which have not been discussed, including drug delivery systems and DNA data storage systems[93]. Most of these other applications and areas of research suffer from issues similar to those already discussed. One such area is that of biosensors, which will be discussed in the following section.

1.4. Genetic circuit-based biosensors

Biosensors are devices which utilise biological components to detect specific stimuli. In recent decades, a new type of biosensor has become popular, thanks in part to the advent of synthetic biology technologies and approaches[110]. These types of biosensors, referred to here as genetic circuit-based biosensors, or simply genetic biosensors, are similar to the cellular biocomputational devices mentioned previously[9]. Genetic circuit-based biosensors use DNA to encode different elements of the system, and can be seen as a type of biocomputing, where logical functions are applied to generate an appropriate response to a desired stimulus (or set of stimuli). In fact, the genetic toggle switch from the previous section essentially acts as a biosensor for IPTG, as in the presence of IPTG (and absence of aTc), a response is generated (production of GFP, which produces a green, fluorescent signal).

1.4.1. Mechanisms and applications

Genetic biosensors have a wide range of applications, which is made clear by the plethora of diverse devices developed using synthetic biology[111]–[113]. Genetic biosensors have been developed to detect metabolites for aid in metabolic engineering efforts, helping to replace costly analytical techniques and provide real-time monitoring of cell cultures, and rapid identification of strains which show promising functionality during the development of these metabolic factories[114]. There are other devices with diagnostic applications, such as a paper-based test which utilises CFPS systems to detect Zika virus[115]. Devices for reporting on the presence of toxic metals or other pollutants have been developed for use in environmental monitoring and to direct remediation efforts[116].

A common biological component to be used as the sensing mechanism of a genetic biosensor are regulatory factors[117] (Figure 1.6). When a stimulus interacts with the transcription or translation factor acting as the sensing mechanism, a change in genetic regulation occurs, which results in either up or down regulation of a coding sequence (CDS). The type of CDS, or in some cases set of CDSs, determines the type of response generated. For example, a CDS which encodes a fluoresecent protein could be used to give a fluorescent signal in response to a stimulus, or a CDS for a metabolic regulatory protein could be used to control production of a specific molecule of interest. An example of a genetic biosensor is the Frm-based formaldehyde biosensor[118]. This genetic biosensor contains several elements encoded as DNA, namely the *frmR* CDS which encodes the FrmR transcription factor, the *Pfrm* promoter, and a *gfp* CDS which encodes a green fluorescent protein (GFP). Once the biosensor is activated, the *frmR* CDS is expressed which results in production of the FrmR transcription factor. Ordinarily, FrmR represses expression from the *Pfrm* promoter, which in this case is positioned upstream from the *gfp* CDS. However, formaldehyde is able to bind to FrmR, which causes a change in the conformation of the transcription factor and prevents it from repressing *Pfrm*. Therefore, in the presence of formaldehyde, FrmR no longer represses the *Pfrm* promoter, which results in expression of the *gfp* CDS, and hence GFP is produced. The GFP emits a green fluorescent signal which can be visualised or measured. The intensity of the fluorescent signal can be used as a proxy to the concentration of formaldehyde present.

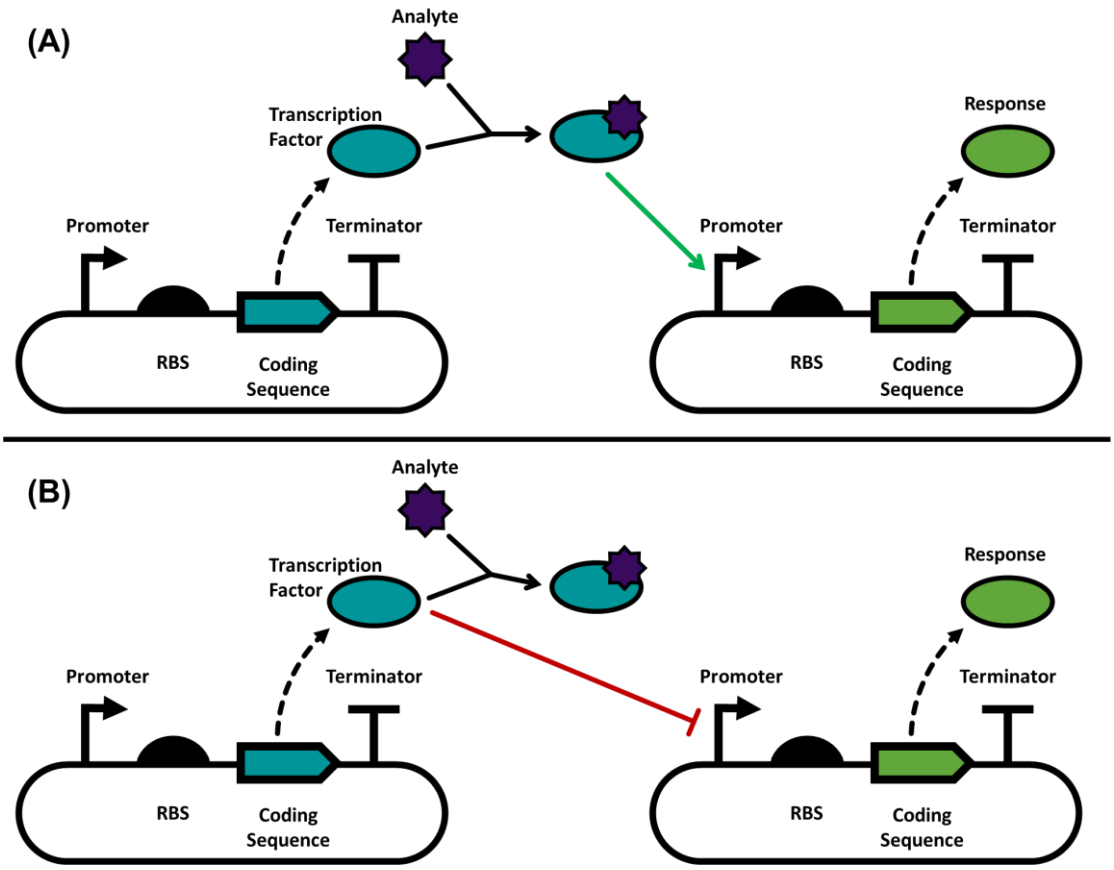


Figure 1.6. Transcription Factor-Based Biosensor Mechanisms

Two common mechanisms for transcription-factor based biosensing. In each example, the analyte acts as an inducer. (A) An example of positive induction, where the combination of a transcription factor and analyte can promote genetic expression from a specific promoter to elicit a response. (B) An example of negative induction, where a transcription factor represses expression from a promoter. When an analyte is present, the transcription factor becomes sequestered and promoter repression is relived, leading to genetic expression and a response.

Genetic biosensors cannot be implemented in the same way as other biosensors. Whilst PCR or enzymatic biosensors are implemented *in vitro*, for genetic biosensors the DNA encoding the device must be inserted into an appropriate biological chassis capable of expressing the genetic[119]. In the specific case of the formaldehyde biosensor, the DNA encoding the device is intended for transformation into *Escherichia coli*. The genetic elements of the biosensor were designed and selected in such a way that the native biological machinery present in *E. coli* is able to correctly express the formaldehyde biosensor.

While a bacterial species is used in this example, there are many examples of other species being used as a chassis for a biosensor device, including yeast[120], plant species[121], and mammalian cell lines[122]. There has also been increasing popularity in using cell-free protein synthesis (CFPS) systems as a biological chassis[123]. Put

simply, CFPS systems use biological machinery and pathways to express genetic elements, but are not, and do not contain, living cells[124]. Whilst a wide range of potential hosts exist, only a handful are used regularly[82]. This can be problematic for the development of biosensors, where mechanisms of detection for many interesting molecules are often found in non-model organisms. There is then a choice to be made between implementing the detection mechanism in a non-native host, which in some cases may be difficult, especially if the molecule of interest is toxic, or required regulatory elements perform poorly in model organisms, or trying the implement logic and response mechanisms into a species where behaviour of the parts responsible for those mechanisms is less predictable.

1.4.2. Biosensor development and challenges

When developing a genetic circuit-based biosensor, or any biosensor, the identification and engineering of the detector mechanism, which interacts with the desired stimulus to create a 'genetic signal' and generate a response, is fundamental. For some stimuli, detector mechanisms may already be known, but in other cases there may be no known mechanism of detection. Although research has been conducted into more efficient development of detection mechanisms, this work is ongoing[125], [126].

Aside from the detection mechanism, biosensors require a mechanism for generating a response in line with the design specification. For genetic biosensors, these responses tend to be the result of a change in expression of a CDS[127]. The nature of the protein encoded for by the CDS will vary based on the response, and could include fluorescent proteins, which provide a signal which can be both qualitative and quantifiable, a pigment which results in a change in colour, or an enzyme/set of enzymes required for activation of a metabolic pathway.

Genetic biosensors often make use of the standardised biological parts covered in *section 1.2.1* in their design. When the genetic biosensor's chassis is a model organism, there are a wealth of standardised parts which can be selected from to help code the desired response. If the chassis is less studied and used in synthetic biology, there are likely to be fewer choices. The exact selection of parts will determine the overall function of the biosensor. For example, a section of the biosensor may require a protein to be continually expressed, and hence require the use of a constitutive promoter. The 'strength' of the promoter (i.e. the rate of transcription of coding regions under the

control of that promoter) will affect the amount of that protein produced, and hence have an affect on the overall system[128]. This can have an impact on the signal-response curve described previously.

Development of genetic biosensors tends to hinge on the ability to optimise and tune their functionality towards the parameters set out by the design specification[117], [129]. There tend to be many trade-offs to consider, and it can take many iterations of the DBTL cycle to obtain a fit-for-purpose biosensor[52]. Whilst there are many success stories of genetic biosensors, it is still true that many genetic biosensors, and indeed a lot of synthetic biology solutions, tend to find themselves stuck in the proof-of-concept stage. This can stem from difficulties with optimisation, which could be due to the design space of a system being inaccessible, perhaps due to requirements for optimisation of aspects like enzyme kinetics or protein binding parameters[130]. These types of optimisations can be difficult as they rely on protein engineering, which as discussed earlier is still far from a simple task. It is also possible that the way in which the design has been implemented makes it time consuming and costly to generate variants for testing. This is common when designs have been built without standard DNA assembly techniques, as it becomes tough to easily swap out elements of the design, and even tougher to generate enough variants to properly explore the potential design space.

Despite difficulties associated with biosensor optimisation, there are examples of innovative approaches to improve a biosensor's functionality. One such example is that of a macrolide biosensor[131]. Macrolides represent a class of pharmaceutically relevant molecules. The initial biosensor design, which made use of transcription factor-based mechanisms, displayed low sensitivity. To improve sensitivity, an enzyme was added which modified macrolides to prevent them from diffusing out of the host cells. This allowed for detection of lower macrolide amounts by concentrating the analytes around the detection mechanisms. The optimisation approach taken here, whilst impactful and indicative of the potential bioengineering has for biosensor development, was highly specific to the type of biosensor being developed, as not all stimuli could be modified in such a way.

There has been other research into genetic biosensor optimisation approaches, such as a study which aimed to provide a more generalised framework for biosensor

optimisation[132]. This framework utilised computationally informed design in the form of protein structural modelling to modify a transcription factor for specificity towards an analyte different than its native binding partner. Potential mutants were screened rapidly *in vitro* with the aid of cell-free protein synthesis systems, and the characterisation data was used to inform future iterations of the design stage and the creation of more optimal mutants. Whilst this study showed the great potential for such an approach, it was highly reliant on high quality structural data for the transcription factor being mutated, which is not always available. Additionally, the final mutated transcription factor may require further optimisation, as the initial cell-free screening step fails to account for issues such as membrane permeability of the analyte, and potential toxicity.

In some cases, biosensor designs may be large, especially in cases which require complex, or even just moderate, signal processing or biocomputational capabilities[133]. These situations are likely to become more commonplace as synthetic biology aims to solve more complex problems and compete with other engineering fields generally. In such situations, implementation of the design into a chassis is likely to introduce host stress, which can cause reduced performance of the system as a whole[134]. In some cases, it is possible to use a chassis which is more resistant to the problematic elements. Often, however, genetic biosensors will use elements which have optimal chassis which are different from one another. This could occur if the detection mechanism has been taken from one species, such as a yeast, but the parts used for signal processing were developed for a different species, such as a gram-negative bacterium. It may be possible to address the issue of both host stress and sub-optimal chassis through the use of synthetic multi-microbial consortia. This concept is discussed in detail in the following section.

1.5. Multi-microbial systems

1.5.1. Natural microbial communities

In nature, micro-organisms are known to form microbial communities, where many different strains or species co-exist[135]. Such multi-microbial communities can differ in size, complexity, and diversity; some communities may contain variants of a single species, whilst others are composed of many diverse micro-organisms[136]. To survive in different environments, micro-organisms rely on a plethora of biological processes, including the degradation of toxic substances into less harmful products, conversion of

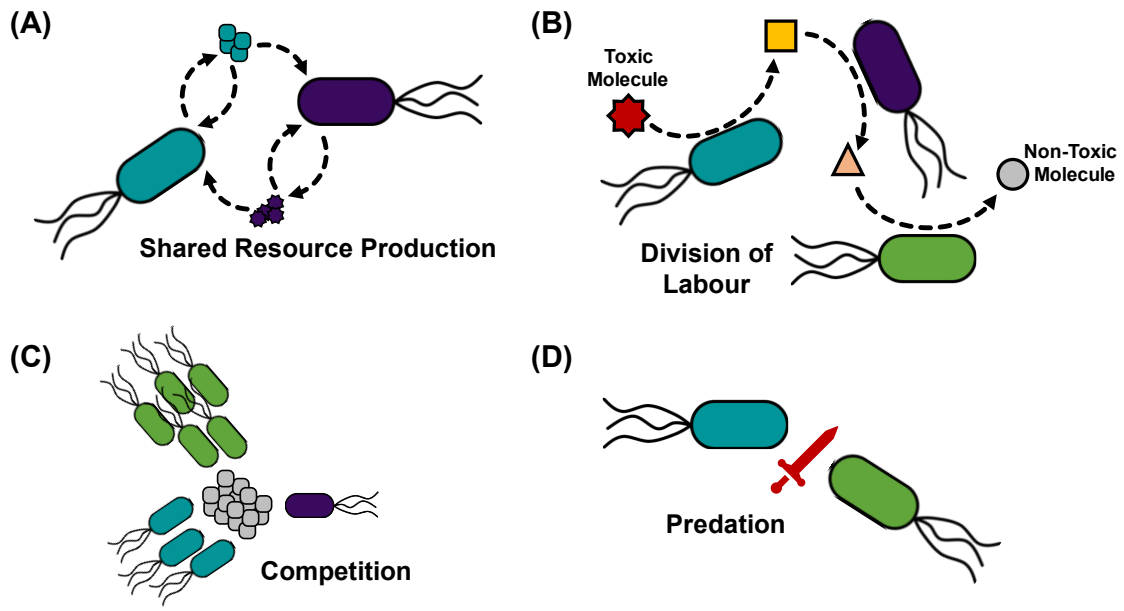


Figure 1.7. Microbial Consortia Interactions and Relationships
Illustrated are some examples of interactions which occur within microbial consortia. (A) Different species can produce resources to be shared with the wider community, and in turn gain access to resources which they do not produce. (B) Biological processes can be split across the different members of a community. Illustrated here is a hypothetical degradation pathway which converts a toxic molecule into a non-toxic product. (C) Cells within a community compete for resources, with better adapted species out growing less well adapted members. (D) Different species may directly predate on other members, which can prevent competition and provide an alternative source of resources.

resources into substrates which can be used as an energy source, and detection of threats[137]. Within a community, these processes can be divided among different populations (Figure 1.7 (A)). This 'division of labour' allows members to efficiently perform functions at which they excel, sharing the fruits of their labour with the community, whilst leaving others to deal with processes they are less adapted to, or unable to, perform[138].

It is also possible for a single process to be divided, where different populations perform separate parts of the overall process (Figure 1.7 (B)). Indeed, processes which require a large number of resources, such as some enzymatic pathways, are known to be performed through collaboration of multiple populations of a community, where each member performs part of the enzymatic process. This phenomenon can be seen in wastewater microbial communities subjected to the chemical terephthalate, where distinct populations play different roles in converting the chemical into useful products and energy for crucial cellular processes[139]. This type of division prevents individual members from having to perform the entirety of a high-burden process, which would

decrease their overall fitness by diverting resources required for other critical processes important in cell growth and survival. Instead, the burden gets split across multiple cells, ensuring that resources remain for the individual members to grow and survive, as well as to perform other functions useful to the community. It has also been shown that the division of processes such as these actually increases the overall efficiency, as each step is performed by healthier cells, which are usually better adapted to the specific part for which they are responsible[140].

Aside from co-operation, other interactions can occur within natural microbial communities, where species compete for resources through a variety of mechanisms (Figure 1.7 (C)), including the conversion of resources to substrates not usable by competitors, increased growth rates to increase the population of that species and hence use more of the available resources, or direct attack of opposing members to slow their growth rates or cause cell death[141] (Figure 1.7 (D)).

1.5.2. Intercellular signalling

Collaboration between members of a microbial consortia can be co-ordinated through the use of cell-to-cell signalling mechanisms[142]–[144]. Communication between the different populations allows for better co-operation. This is because individual members can let others know about their current situation or state. Microbial communication mechanisms tend to rely on the production of biological molecules, which can be transported out of the individual cells to the environment, where they can then be taken up by other members of the community and impact on regulation of certain processes[145]. One example of such a communication mechanism is that of acyl-homoserine lactone (AHL) based quorum sensing. AHLs are relatively simple molecules, composed of a homoserine lactone and a fatty acid acyl chain[146]–[148]. In this method of communication, AHLs, which are small secondary metabolites, are produced via enzymes called AHL synthases. As the AHLs build up in the cell, a diffusion gradient occurs, and the AHLs begin to freely diffuse across the cell membrane into the surrounding environment. As the AHLs begin to accumulate in the environment, surrounding cells take up the AHLs due to the diffusion gradient, and AHLs in the AHL producing cells accumulates further, due to the reduced diffusion gradient. Once a certain threshold of AHLs is present, genes regulated by these small molecules are turned on. This genetic regulation mechanism occurs through the use

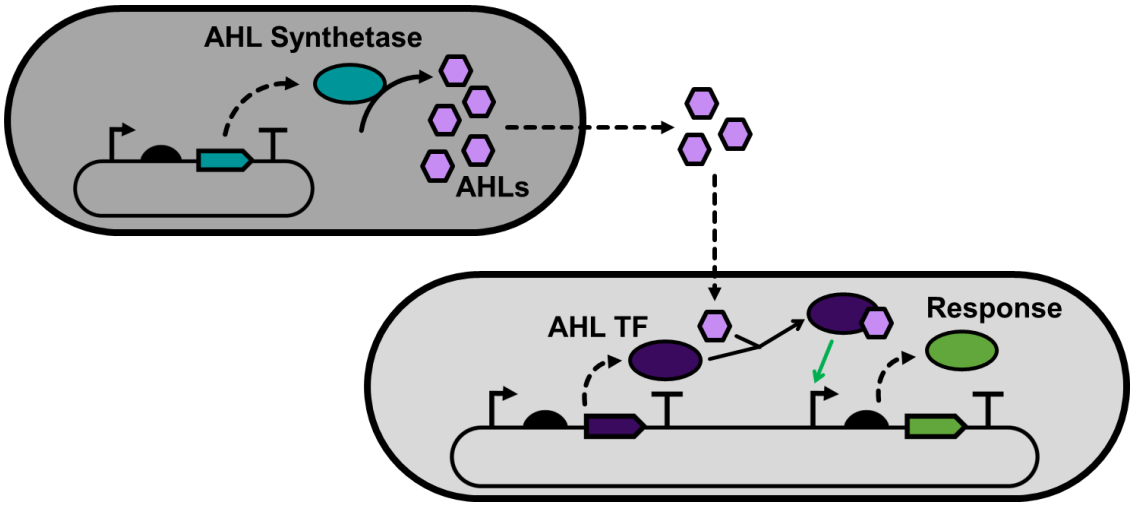


Figure 1.8. General Quorum Sensing Mechanism
Schematic depicting a generic quorum sensing mechanism. Here, one cell (top) is an acyl-homoserine lactone (AHL) producer, and the other cell (bottom) is an AHL responder. AHL molecules are produced by an AHL| synthetase enzyme. The AHLs diffuse across a concentration gradient into the extracellular environment. As the AHL accumulates, the molecules diffuse into cells with fewer AHL molecules. AHL transcription factors (TF) can be activated by AHL molecules, which in turn activation transcription from specific promoters, generating a response. In nature, individual species often have both AHL producing and responding capabilities (see main text).

of a transcription factor which promotes expression from a specific promoter, but only when bound to an AHL.

In nature, it is common to find species which both produce and respond to the same AHL[149]. By expressing AHL synthases during normal cell growth, as the cell populations grows more AHL is produced and accumulates in the environment. The increased AHL concentration can signal to members in the population that the number of cells is increasing. This mechanism can be used by cell populations to ensure that certain genes are only turned on when a specific quorum of cells are present. It is common to find pathogenic bacteria, such as *Pseudomonas aeruginosa*, which make use of this quorum sensing mechanism, where virulence factors are only produced when enough cells are present to be effective, and to prevent alerting the body's immune system early[150]. There are different types of AHLs, which mainly differ in the lengths of their acyl chain[151]. Different AHL synthases specialise in producing a specific AHL variant, and the transcription factors and promoters are similarly associated with a specific AHL. However, due to the similar structures, there is cross talk between the different AHLs, where one AHL may bind to and activate a transcription factor associated with a different AHL.

Whilst the vast majority of known communication mechanisms involve the production of small molecules, be they secondary metabolites or peptides, there is some evidence of cell-to-cell signalling which is mediated by other factors, such as light, although there is still no conclusive evidence for this[152].

1.5.3. Engineering natural communities

As evidenced by the study of natural microbial communities, the concepts of division of labour and microbial communication can have a wide range of positive impacts on the overall function and efficiency of processes necessary for survival. It is perhaps no surprise, then, that the development of multi-microbial systems composed of two or more cell types is becoming increasingly popular within synthetic biology[153]–[155]. These efforts can largely be split into two broad areas: engineering of natural communities, and the development of synthetic consortia.

There are many approaches towards the engineering of natural communities, some of which involve modifying the genetics of one or more populations, and other which focus on modifying the environment of and resources available to the members of the community. It has been shown previously that altering the feedstock available to a microbial community can influence the population composition, resulting in changes to the processes performed by that community[156]. There is also an interest in engineering the genetics of the microbial community as a whole (termed the metagenome)[157]. Aside from engineering natural communities, there is increased interest in developing completely synthetic multi-microbial systems, driven by efforts to overcome some of the issues discussed previously which plague the progression of synthetic biology.

1.5.4. Synthetic multi-microbial communities

Within synthetic biology, the limits of what can be achieved with cells in monoculture are being reached[158], [159]. This is largely due to the need for increasingly complex designs to be implemented, which can impart immense burden on any cell expressing the system. The issue of burden, as touched upon previously, is a result of resources needed for cell survival being diverted in order to express the synthetic system which has been implemented. This lack of resources can negatively impact cell growth and even lead to cell death, resulting in fewer cells available to perform the intended function. When burden becomes too high, it can impact on the system's performance, with effects ranging from less-than-optimal behaviour to a complete lack of function.

The need for complex systems also comes with other issues, such as the requirement for a large number of biological parts. In other engineering fields, such as electrical engineering, it is commonplace to re-use identical parts in a system; components such as buzzers or voltage amplifiers can be used multiple times without issue. However, in synthetic biology, re-use of identical components is a lot trickier. This is due to the high potential for cross-talk to occur, as there is very rarely any insulation between the different section of a system[160]. This means that if, for example, two or more signal amplifiers were required for a design, they must be composed of different genetic parts, as otherwise parts such as transcription factors produced by one amplifier can interact with the promoter in one of the other signal amplifiers, leading to unwanted functionality.

As alluded to, the issues described above could be alleviated through the development of multi-microbial systems composed of two or more cell types[161]. This is because of the potential to split a system's design into multiple parts, each of which is then implemented into separate chassis. As with processes being split across multiple members of a natural microbial community, this has the advantage of not only reducing the total burden experienced by any one cell, but also enables more suitable species to be selected for sections of the design which require specific chassis in order to function as intended. Additionally, re-use of identical biological parts becomes more feasible, as the separate sections of the system can be isolated within their respective chassis, meaning that the chance for cross talk to occur between elements such as transcription factors is significantly reduced.

In order to successfully implement a synthetic biological system using microbial consortia, it is necessary to engineer some method of communication between cells in the system. Usually, this involves quorum sensing, of which AHL based mechanisms tend to be the system of choice[162]. This is likely a result of the mechanism of action being well characterised, and that transport is via passive diffusion and does not require additional components, as AIP based quorum sensing does. The use of AHL based quorum sensing has been put to good use, with many examples of synthetic microbial consortia utilising them, such as a spatial patterning system composed of two strains of *E. coli* with QS-mediated communication[163]. Other studies involving quorum sensing in synthetic systems include communication between physically separated populations in a microfluidic device, which use quorum sensing to create

oscillations of GFP expression[164], and implementation of a biological system which exhibits neural-like pattern recognition[107].

Whilst synthetic consortia have shown a lot of promise, there are still challenges. These include issues with system instability, where different cell types can grow at different rates, causing the ratios of populations to change over time and the system the eventually collapse [165]. To counter the issue of co-culture instability, there have been efforts to engineer cross-feeding mechanisms into synthetic consortia[166]. Cross-feeding refers to populations of a consortia relying on each other for cell growth, as each produces a resource required by the other. In this way, the growth of the populations becomes linked, as if one population begins to outcompete the other, it will be limited by the resource provided by the population. Whilst these systems have been used successfully, they become difficult to implement and scale in complexity with increasing numbers of cell types in the system. Other problems associated with synthetic consortia, especially those involving diverse species, involve issues arising from a requirement for different conditions in order to grow and survive[167]. For example, different species can require radically different temperature to grow efficiently, and thus compromising on temperature for consortia of such cells can result in poor functionality of the system. Nevertheless, synthetic consortia could provide a solution to some of the big issues currently faced by synthetic biology.

1.6. Overview of This Work

1.6.1. A brief overview with rationale

Synthetic biology approaches are undoubtedly useful, as evidenced by the many synthetic biological systems and devices developed, and the impact they have had on bioengineering efforts across all areas. However, as can be seen from the discussion above, challenges do remain, and there is still considerable potential for the efficiency of synthetic biology approaches to be improved upon. There are many causes for these challenges, of which many overlap. However, some of the common problems found include difficulties with implementing large and complex systems, issues with re-using aspects of some designs, and challenges associated with rapidly and efficiently optimising built systems. Whilst the exact nature of these challenges differ, similar obstacles can be found in other engineering fields. One approach to tackle the problems encountered within these other fields is the use of high-level modularity, where devices and systems are designed and implemented using modules with high-

level functionality. The research presented for this thesis focused on demonstrating how the principles of high-level modularity and synthetic multi-microbial systems could be used to aid in the development of a specific type of synthetic biology device: biosensors. The following section provides a description of the project's aims, which were identified with the intention of achieving this goal.

1.6.2. Main aims

In this thesis, the overall goal was to investigate the feasibility of developing genetic biosensors with the aid of a modular, multi-microbial framework. Biosensors were selected as an application mainly due to their widespread usage within other areas of synthetic biology. To achieve this goal, the following aims were identified:

- Develop tools and methods for the design, building, and characterisation of a modular, multi-microbial biosensor.
- Design a proof-of-concept modular, multi-microbial biosensor according to a set of design principles
- Use computational simulations to guide the characterisation and optimisation of the proof-of-concept biosensor
- Characterise the proof-of-concept biosensor and its modules
- Attempt to tackle difficulties with co-culturing microbes by investigating a method of microbial communication which is not reliant on the transfer of molecules

1.6.3. Thesis structure

This thesis is split into seven main chapters. Each results chapter (3 – 7) begins with an introduction to the chapter and provides some specific background information. Results within these chapters are provided with discussion, following which the chapter is concluded with an overview of the main outcomes and discussion of next steps.

The *Introduction and Background Information* chapter provides background information and a review of previous studies within the area of synthetic biology, biosensors, and multi-microbial systems. Common challenges surrounding the development and implementation of biological systems are discussed, and current approaches to alleviate these issues are discussed, along with an explanation of their own advantages and disadvantages.

The second chapter, *Materials and Methods*, describes the methodology used to perform the experiments discussed in this thesis. An explanation of the data handling and analysis used is also given where appropriate. It should be noted that the methodologies may not be in the order in which the results they yielded are discussed in the thesis. However, where the results are discussed, references back to the appropriate sections of *Chapter 2* are given.

Chapter 3, the first results chapter, describes a potential framework for developing a modular, multi-microbial biosensor. It is also presented how the Synthetic Biology Open Language (SBOL) was extended to allow for standardised representation of multi-microbial systems, along with a set of tools created for aiding in automation of the development of biosensor modules, and synthetic biology workflows more generally.

In *Chapter 4*, the design for a proof-of-concept modular and multi-microbial genetic biosensor is given. Results are then presented from computational modelling of the biosensor modules using both deterministic and agent-based simulation. Following this, agent-based modelling is used to predict functionality of the multi-microbial biosensor, and to explore a potential avenue for optimisation.

Chapter 5 discusses characterisation of the biosensor modules, along with validation of intercellular communication between cell types expressing each module. Finally, the multi-microbial biosensor itself is tested experimentally.

Attempts to optimise the proof-of-concept biosensor through modification of cell ratios are presented in *Chapter 6*, and statistical Design of Experiments is used to determine important factors for optimisation of culturing conditions.

Informed by difficulties encountered throughout the project, *Chapter 7* deals with efforts to implement and alternative to chemical-based microbial communication. This alternative takes the form of bioluminescence-based communication, which may allow for easier implementation of multi-microbial systems by avoiding the need for co-culturing.

The *Conclusion and Future Work* chapter rounds up the main outcomes and results of this thesis, and discussion is given to the overall limitations of this work. Potential future work to address these limitations and further the project is also discussed.

The thesis finishes with *Supplementary Information* and *References* chapters. The *Supplementary Information* is split into sections according to each chapter in this thesis, and the *References* chapter provides citations for the entire thesis, arranged by order of appearance.

Chapter 2. Materials and Methods

2.1. Generic Methods

2.1.1. Bacterial cell culturing

Unless stated otherwise, Lysogeny Broth (LB) media was used for growth of cells in liquid medium. LB media was prepared by combining chemicals in required amounts (10 g/L tryptone; 10 g/L sodium chloride; 5 g/L yeast extract) and dissolving in MiliQ water. Media was sterilised by autoclaving for 20 minutes at 121^{o}C.

Table 2.1. Antibiotic Details: (*) Long term storage concentration at -20^{o}C. (†) Concentration for short-term working solution stored at 4^{o}C. (§) Final concentration used in liquid and solid culture media.

Antibiotic	Solvent	Concentrations (mg/mL)		
		Storage*	Working†	Final§
Chloramphenicol	Ethanol	100	10	0.05
Kanamycin	Water	25	10	0.05
Ampicillin	Water	50	20	0.1

In all cases, growth of cells on solid media was performed using LB agar plates. LB agar was prepared by combining chemicals in required amounts (15 g/L agar; 10 g/L tryptone; 10 g/L sodium chloride; 5 g/L yeast extract) and dissolving in MiliQ water. The mixture was sterilised by autoclaving for 20 minutes at 121^{o}C and allowed to solidify. Solid agar was heated using a microwave until molten and left to cool. The relevant antibiotic was added at the appropriate concentration (Table 2.1) and mixed. Agar was then poured into 90 mm petri dishes and left to solidify.

Antibiotics were stored in solvents as detailed by Table 2.1. Antibiotics were stored at either -20^{o}C (long term) or 4^{o}C (short term). Storage, working, and final concentrations for each antibiotic are detailed in Table 2.1.

For overnight growth of cell cultures in liquid media (referred to as ‘overnight cultures’ within this thesis), 10 mL of LB media was added to a sterile 50 mL falcon tube and inoculated with a single colony from the appropriate agar plate. If required, antibiotic from the appropriate working stock was added to the required final concentration (Table 2.1). Cultures were incubated at 37^{o}C for 16-18 hours with constant orbital shaking at 200 Revolutions per Minute (RPM), unless stated otherwise.

In all cases, the bacterial cell strain used was *Escherichia coli* DH5α originally sourced from New England Biolabs (NEB). Single colonies of *E. coli* DH5α were obtained by streaking from a glycerol stock stored at -80ºC (sourced from NEB) onto LB agar plates with no antibiotic and incubated at 37ºC for approximately 16 hours. Colonies were either used immediately, or the agar plate was stored short-term at 4ºC until needed.

2.1.2. Preparation of E. coli *competent cells*

Two transformation buffers were used for preparing competent cells: TF-1 and TF-2. TF-1 was prepared by first mixing chemicals at the required amounts (7.4 g potassium chloride; 2.95 g potassium acetate; 1.5 g calcium chloride dihydrate; 150 g glycerol) and dissolving in 950 mL MiliQ water. The buffer's pH was adjusted to ~6.4 with acetic acid and sterilised by autoclaving for 20 minutes at 121ºC. A 1 M manganese chloride tetrahydrate solution (198 g/L) was prepared in MiliQ water and sterilised by syringe filtration through a 0.22 µm filter, of which 50 mL was added to 950 mL the TF-1 buffer. TF-2 was prepared by mixing chemicals at the required amounts (0.74 g potassium chloride; 11 g calcium chloride dihydrate; 150 g glycerol) in 980 mL MiliQ water and sterilised by autoclaving for 20 minutes at 121ºC. A 0.5 M solution of MOPS buffer (104.7 g/L 3-(*N*-morpholino)propanesulfonic acid) was prepared in MiliQ water with a pH adjusted to 6.8 with potassium hydroxide, and sterilised by syringe filtration through a 0.22 µm filter. 20 mL of MOPS buffer was added to 980 mL of TF-2 buffer. Both buffers were stored at 4ºC until required.

An overnight culture of *E. coli* DH5α was prepared from a streak plate on LB agar with no antibiotic as described in sub-section 2.1.1. 40 mL of LB media was added to a 250 mL conical flask and inoculated with 400 µL of overnight culture. The culture was incubated at 37ºC with shaking at 200 RPM until an optical density measurement at 600 nm of approximately 0.5 was reached. The full culture was transferred to a 50 mL falcon tube, and cells were harvested by centrifugation at 4,500 RPM and 4ºC for 10 minutes. Supernatant was discarded. The cell pellet was re-suspended in 8 mL of ice-cold TF-1 buffer and incubated on ice for 15 minutes. Cells were harvested by centrifugation at 4,500 RPM and 4ºC for 10 minutes and supernatant discarded. The cell pellet was re-suspended in 4 mL of ice-cold TF-2 buffer and aliquoted at 50 µL into 1.5 mL tubes and stored at -80ºC until required.

2.1.3. Transformation of plasmids into **E. coli** *cells*

The appropriate number of competent *E. coli* DH5α aliquots were thawed, and between 2 and 10 µL of plasmid DNA was added per aliquot. DNA was thoroughly mixed with gentle pipetting up and down. The cells-plasmid mixtures were subjected to heat shock at 42^{o}C for 1 minute and then cooled to 4^{o}C. 250 µL of LB media was added to the aliquots, which were then incubated for 1 hour with 250 RPM shaking at 37^{o}C. All cells were dispensed onto separate LB agar plates prepared with the appropriate antibiotic, and cells were spread using autoclave sterilised glass beads. Beads were removed and cells were allowed to dry for 10 to 20 minutes, before being inverted at incubated at 37^{o}C for approximately 16 to 18 hours.

2.1.4. Purification of plasmid DNA

A single colony of cells transformed with the plasmid to be purified were used to inoculate an overnight culture with the appropriate antibiotic. Cultures were then processed using Monarch Plasmid Miniprep Kit (obtained from New England Biolabs) according to the manufacturer's protocol. Two modifications were made to this protocol: (i) overnight cultures were centrifuged at 4,500 RPM for 10 minutes instead of 13,000 RPM for 30 seconds, and (ii) nuclease-free water was during the elution step instead of elution buffer. The concentration of plasmid DNA purified was determined using a UV-Vis spectrophotometer (NanoDrop One, Thermo Scientific) according to the manufacturer's instructions. Plasmid DNA was stored at -20^{o}C for long term storage, and 4^{o}C for short term storage.

2.1.5. DNA gel electrophoresis

Gel electrophoresis was used to visualise plasmid backbones on agarose gels. Agarose gels were prepared by combining 1% w/v agarose powder with 1x Tris-acetate-EDTA (TAE) buffer and heating with stirring until all powder was dissolved. 0.1% v/v of 1% Nancy-520 (dsDNA dye supplied by Sigma-Aldrich) was added to the solution and mixed thoroughly. The molten agarose mixture was poured into a gel tray with a comb added and left to set. DNA samples were mixed with 6x purple gel loading dye (NEB) to a final volume of 10 µL and added to the agarose gel wells. 1kb and 100bp ladders (NEB) were used as markers. The gel was placed into an electrophoresis container filled with 1x TAE buffer, and a current was run across the gel for approximately 1 hour. Gels were visualised under UV light.

2.1.6. Sequence verification of plasmid DNA

The sequence of purified plasmid DNA was obtained via Sanger sequencer performed by Eurofins. Samples were prepared by combining approximately 500 ng of plasmid DNA with 2.5 µL of either 10 µM forward primer (VF2: TGCCACCTGACGTCTAAGAA) or 10 µM reverse primer (VR: ATTACCGCCTTTGAGTGAGC) and making the reaction up to 10 µL with nuclease-free water.

2.2. Manual Assembly of DNA Constructs

Constructs were designed and assembled *in silico* using Benchling prior to assembly. All assembled DNA constructs are listed by assembly method in Table 2.6 and Table 2.7, along with information such as the DNA assembly method(s) used and a link to the construct's sequence. Sanger sequencing was used to verify correct assembly and sequence (sub-section 2.1.6). For a list of all DNA constructs or parts not built in this study, see Table 2.2.

Table 2.2. DNA Obtained Outside this Study: Listed here are all DNA parts or constructs with did not originate within this study.

Name	Plasmid Backbone	Antibiotic Selection	Source
BBa_J04450	pSB1C3	Chloramphenicol	iGEM 2017 Distribution Kit
BBa_J06602	pSB1C3	Chloramphenicol	iGEM 2017 Distribution Kit
BBa_K2205010	pSB1C3	Chloramphenicol	Newcastle iGEM 2017
BBa_K2205011	pSB1C3	Chloramphenicol	Newcastle iGEM 2017
BBa_K2205014	pSB1C3	Chloramphenicol	Newcastle iGEM 2017
PBLRep-EL222	----	Kanamycin	Gift from Chueh Loo Poh[168]
PBLInd100-EL222	----	Kanamycin	Gift from Chueh Loo Poh[168]
pOdd1	pOdd1	Kanamycin	iGEM 2017 Distribution Kit
pOdd2	pOdd2	Kanamycin	iGEM 2017 Distribution Kit
pSB1C00	pSB1C00	Chloramphenicol	iGEM 2017 Distribution Kit
pSB1C3-Lux	pSB1C3	Chloramphenicol	pSB1C3-Lux was a gift from Tom Ellis (Addgene plasmid # 109383)

2.2.2. BioBrick assembly

All reagents were obtained from New England Biolabs (NEB), and all assemblies were performed at 20 µL total reaction volume in 0.2- or 0.5-mL tubes. DNA constructs assembled via BioBrick assembly are listed in Table 2.6.

For each assembly, the plasmid backbone and insert(s) were digested separately. DNA digest reactions were prepared as shown in Table 2.3. Reactions were incubated at 37ºC for 1 hour, with a subsequent heat inactivation step at 80ºC for 20 mins. Reactions were held at 4ºC until required.

Table 2.3. DNA Digest Reaction for BioBrick Assembly: (*)
Enzymes used for each reaction are shown in Table 2.6.

Reagent	Amount
Plasmid Backbone	250 ng
Enzyme 1* (10 units/µL)	1 µL
Enzyme 2* (10 units/µL)	1 µL
10x rCutSmart Buffer	2.5 µL
Nuclease-Free Water	Up to 20 µL

Unless stated otherwise, ligation reactions were prepared as shown in Table 2.4. Ligation reactions were incubated at either 16ºC for 16 hours or 25ºC for 2 hours, followed by heat inactivation at 65ºC for 10 mins. Reactions were held at 4ºC until needed. Reactions were transformed into *E. coli* DH5α as stated in sub-section 2.1.3, using approximately 5 µL of reaction mixture.

Table 2.4. Ligation Reaction for BioBrick Assembly: (*) Volume varied between reactions; aimed for ~50 ng of plasmid backbone, with BioBrick insert(s) added at an addition 3-fold. (†) DNA used for each reaction can be found in Table 2.6.

Reagent	Amount
Digested Plasmid Backbone	~ 4 µL*
Digested BioBrick Insert(s)	~ 3.5 µL*
T4 DNA Ligase (400 units/µL)	1 µL
T4 DNA Ligase Buffer (10x)	2 µL
Nuclease-Free Water	Up to 20 µL

A variant of the reactions above were used to transfer constructs between BioBrick compatible plasmids. For these reactions, all steps remained the same except EcoRI and PstI was used to digest both the plasmid backbone and BioBrick insert.

2.2.3. Gibson assembly

All reagents were obtained from New England Biolabs (NEB), and all assemblies were performed at 20 µL total reaction volume in 0.2 or 0.5 mL tubes. Gibson assembly was performed using gBlocks ordered from Integrated DNA Technologies (IDT). The gBlocks were designed by adding 20-30 bp overhangs to the desired part for assembly with homology to the plasmid backbone insert site. Assembly was verified *in silico* using the NEBuilder tool by New England Biolabs. DNA constructs assembled via Gibson assembly are listed in Table 2.7.

gBlocks from IDT were prepared by centrifuging the dried DNA for 10 to 30 seconds at 12,000 RPM, and then resuspending by vortex in 50 µL of nuclease-free water for a final concentration of 20 ng/µL. The re-suspended DNA was incubated at 50^{o}C for 20 mins and then kept at -20^{o}C until required. The plasmid backbone was linearised by restriction digestion using the same process described in sub-section 2.2.2, using enzymes mentioned in Table 2.7. Restriction sites were selected which flanked the desired insertion site.

Gibson assembly was performed using the NEBuilder HiFi DNA assembly kit supplied by New England Biolabs. Reactions were prepared as stated in Table 2.5. Reactions were incubated at 50^{o}C for 60 minutes, and then held at 4^{o}C until required. Reactions

were transformed into *E. coli* DH5α as stated in sub-section 2.1.3, using 5 µL of reaction mixture.

Table 2.5. HiFi Reaction for Gibson Assembly: (*) For all assemblies, three reactions were performed. Each reaction varied by the amount of gBlock added, which was determined as either a 1-fold, 2-fold, or 3-fold increase in pmols of gBlock relative to the number of plasmid backbone pmols.

Reagent	**Amount**
Linear Plasmid Backbone	~ 50 ng
gBlock	Various*
NEBuilder HiFi Master Mix	10 µL
Nuclease-Free Water	Up to 20 µL

Table 2.6. DNA Constructs Assembled via BioBrick Assembly: Shown here are successfully assembled DNA constructs assembled with BioBrick assembly.

Construct Name	Plasmid Backbone	Antibiotic Selection	Constitutive Parts (With Enzymes Used)	Parts Source	Link to Sequence
Connector-2 Sender + mCherry Sub-Module	pSB1AT3	Ampicillin	• **Backbone:** pSB1AT3 + BBa_J04450 (EcoRI + PstI) • **Insert 1:** pSB1C3 + BBa_J06602 (EcoRI + SpeI) • **Insert 2:** pSB1C3 + BBa_K2205011 (XbaI + PstI)	• **Backbone:** Table 2.2 • **Insert 1:** Table 2.2 • **Insert 2:** Table 2.2	benchling.com/s/seq-X7cNXRHWB0QkCkYQLfIn
Default Processor + mCherry Module	pSB1C3	Chloramphenicol	• **Backbone:** pSB1C3 + BBa_K2205010 (SpeI + PstI) • **Insert 1:** Connector-2 Sender + mCherry Sub-Module	• **Backbone:** Table 2.2 • **Insert 1:** This study (Table 2.6)	benchling.com/s/seq-GgM5DP5lxw2ixtjvjkjo

Table 2.7. DNA Constructs Assembled via Gibson Assembly: Shown here are successfully assembled DNA constructs assembled with Gibson assembly.

Construct Name	Plasmid Backbone	Antibiotic Selection	Constitutive Parts (With Enzymes Used)	Parts Source	Link to Sequence
pOdd1_AE	pOdd1	Kanamycin	• **Backbone:** pOdd1 + (J23100+B0034_EL222) (EcoRI + PstI) • **Insert:** pOdd1_AE_gBlock	• **Backbone:** This Study (Table 2.2) • **Insert:** Synthesised by IDT as linear dsDNA (this study)	benchling.com/s/seq-r8x27DijlgDaV2zqow3y?m=slm-WyLdr9bGAylIFZYWf8B7

pOdd2_BF	pOdd2	Kanamycin	• **Backbone:** pOdd1 + (J23100+B0034_EL222) (EcoRI + PstI) • **Insert:** pOdd2_BF_gBlock	• **Backbone:** This Study (Table 2.2) • **Insert:** Synthesised by IDT as linear dsDNA (this study)	benchling.com/s/seq-uy3g38t0PKDBtkRWKmuY?m=slm-ZIBdMk04ZWF7RHPVYUIV
pOdd4	-	Kanamycin	• **Backbone:** pOdd1 + (J23100+B0034_EL222) (EcoRI + PstI) • **Insert:** pOdd4_gBlock	• **Backbone:** This Study (Table 2.2) • **Insert:** Synthesised by IDT as linear dsDNA (this study)	benchling.com/s/seq-8DKZKLtQGFUbZxTrpUOn?m=slm-uzBRy5MxR35BexiCD4wj
B0040	pOdd4	Kanamycin	• **Backbone:** pOdd1 + (J23100+B0034_EL222) (EcoRI + PstI) • **Insert:** pOdd4_B0040_gBlock	• **Backbone:** This Study (Table 2.2) • **Insert:** Synthesised by IDT as linear dsDNA (this study)	benchling.com/s/seq-dgbBIl4qdOBOkbRtY4Rr?m=slm-M9llkgFuIP9w9TxSktF3

2.3. Testing BiomationScripter Templates

Described here are the methods used to test the BiomationScripter Templates presented in chapter 3.

2.3.1. EchoProto PCR Template

Eight sequence fragments were amplified from DNA templates using the Echo525 (Labcyte) and the BiomationScripter (v0.2.2) PCR Template (Table 2.13). Reactions were performed at 5 µL final volume and prepared in a 384-well, v-bottom PCR plate. Proportions of reagents per reaction are shown in Table 2.8. Nuclease-free water was transferred from a 6RES plate (Labcyte). DNA templates and primers were transferred from 384PP plates (Labcyte). All other reagents were transferred from 384LDV (Labcyte) plates. The PCR plate had a foil seal applied, was vortexed, and then centrifuged for approximately 10 seconds before being thermocycled according to Table 2.9. PCR fragments were analysed using the TapeStation 4200 (Agilent) and the D5000 Tape Kit (Agilent), according to the manufacturer's instructions. A picture of the gel was exported from the TapeStation Analysis Software (v4.1.1), and raw electrophogram data was exported as a comma separated value (CSV) file for analysis and visualisation in Python (v3.9).

Table 2.8. EchoProto PCR Reaction Composition

Reagent	Amount
NEB Q5 Reaction Buffer (5x)	1000 nL
dNTPs (10 mM)	100 nL
Forward Primer (10 µM)	250 nL
Reverse Primer (10 µM)	250 nL
Template DNA (~200 ng/uL)	1000 nL
Q5-HF DNA Polymerase	50 nL
Nuclease-Free Water	Up to 5000 nL

Table 2.9. EchoProto PCR Thermocycling Conditions

Step	Number of Cycles	Temperature (ºC)	Time (Seconds)
1	1	98	30
2	35	98	10
		60	30
		72	40
3	1	72	120
4	1	4	Inf

2.3.2. EchoProto Loop Assembly Template

Level 0 phytobricks were created by Polymerase Chain Reaction (PCR) amplification from a DNA template. Primers were designed to add 5' and 3' ends to allow for assembly into a Level 0 phytobrick acceptor plasmid (*pSB1C00*). PCRs were performed manually. Reagents and volumes used can be found in Table 2.10. All level 0 parts created, along with a list of primers used, can be found in Table 2.13. PCR fragments were analysed by gel electrophoresis (sub-section 2.1.5) to confirm correct sizing. All PCR fragments were assembled into *pSB1C00* manually via loop assembly. Reactions were prepared as described in Table 2.11, and then subjected to thermocycling as shown in Table 2.12. 5 µL of reaction mixture was transformed into *E. coli* DH5α and plated onto LB agar plates with chloramphenicol antibiotic. A number of non-red colonies (the *pSB1C00* plasmid contains a red fluorescent protein expression unit in the insertion site) per transformation were selected and subjected to plasmid purification. The sequence of level 0 parts in were confirmed via Sanger sequencing.

Table 2.10. PCR for Level 0 Part Creation: (*) For each reaction, DNA backbone and DNA parts were added in a 1:1 and a 1:2 ratio

Reagent	Amount (µL)
BsaI-HF (NEB)	0.125
T4 Ligase Buffer (NEB)	0.5
T4 Ligase	0.125
DNA Backbone (10 fmol/µL) *	0.25
DNA Parts (10 fmol/µL) *	Various *
Nuclease-Free Water	Up to 5

Four level 1 phytobrick assemblies (Table 2.14) were prepared using the Echo525 (Labcyte) and the BiomationScripter (v0.2.2) Loop assembly Template. Reactions were performed at 5 µL final volume and prepared in a 384-well, v-bottom PCR plate. Proportions of reagents per reaction are shown inTable 2.11. Nuclease-free water was transferred from a 6RES plate (Labcyte). Level 0 DNA parts, *pOdd* plasmids, and buffer were transferred from 384PP plates (Labcyte). All other reagents were transferred from 384LDV (Labcyte) plates. The PCR plate had a foil seal applied, vortexed, and was the centrifuged for approximately 10 seconds before being thermocycled according to Table 2.12. 5 µL of assembly reaction was used to transform *E. coli* DH5α cells, and colonies were selected for plasmid purification. Plasmids were subjected to PCR fragments were analysed using the TapeStation 4200 (Agilent) and the D5000 Tape Kit (Agilent), according to the manufacturer's instructions. A picture of the gel was exported from the TapeStation Analysis Software (v4.1.1), and raw electrophogram data was exported as a comma separated value (CSV) file for analysis and visualisation in Python (v3.9).

Table 2.11. Level 1 Loop Assembly Reactions

Reagent	Amount
Q5-HF Master Mix (10x)	12.5 µL
Forward Primer (10 µM)	1.25 µL
Reverse Primer (10 µM)	1.25 µL
Template DNA	~800 ng
Nuclease-Free Water	Up to 25 µL

Table 2.12. EchoProto Level 1 Loop Assembly Thermocycling Conditions

Step	Number of Cycles	Temperature (ºC)	Time (Minutes)
1	30	37	3
		16	4
2	1	50	5
3	1	80	10
4	1	4	Inf

Table 2.13. PCR Reactions: Shown here are PCR reactions performed, with DNA templates and primers used.

PCR Fragment Name	Template DNA	Forward Primer	Reverse Primer	Manual or Automated
J23100_B0034_AC	PBLInd100-EL222 (Table 2.2)	ATCAgctcttcATCGgagtctTTGACGGCTAGCTCAGTCCT	CTAGCCTAGAGAAAGAGGAGAAATACTAGaatgCGAGTgaagagcttgc	Manual
PBLInd100_B0034_AC	PBLInd100-EL222 (Table 2.2)	ATCAgctcttcATCGgagGGTAGCCTTTAGTCCATGtt	taaaTATGTCTAGAGAAAGAGGAGAAATACTAGaatgCGAGTgaagagcttgc	Manual
EL222_CE	PBLInd100-EL222 (Table 2.2)	ATCAgctcttcATCGaatgTTGGATATGGGACAAGATCG	CCGTCGAAGCCGGAATCTAAgcttCGAGTgaagagcttgc	Manual
mCherry_CE	Default Processor + mCherry Module (Table 2.6)	ATCAgctcttcATCGaATGGTGAGCAAGGGCGAGGAGG	CATGGACGAGCTGTACAAGTAAgcttCGAGTgaagagcttgc	Manual
B0015_EF	PBLRep-EL222 (Table 2.2)	ATCAgctcttcATCGgcttaggatctccaggcatcaaataaaac	tcgggtgggcctttctgcgtttatacgctCGAGTgaagagcttgc	Manual
pBLRep_B0034_AC	PBLRep-EL222 (Table 2.2)	ATCAgctcttcATCGggagTTGACAGGTAGCCTTTAGTC	TATAATTATGTCTAGAGAAAGAGGAGAAATACTAGaatgCGAGTgaagagcttgc	Automated (EchoProto)
RBS_LuxD_BE	pSB1C3-Lux (Table 2.2)	ATCAgctcttcATCGTACTagttaaaggaaattatatgaaagatg	gaaaatgaattattagaattggcttaagcttCGAGTgaagagcttgc	Automated (EchoProto)
RBS_LuxA_BE	pSB1C3-Lux (Table 2.2)	ATCAgctcttcATCGTACTataaacagaatcaccaaaaagg	gctcctttcttaaaagaacctaaataagcttCGAGTgaagagcttgc	Automated (EchoProto)
RBS_LuxG_BE	pSB1C3-Lux (Table 2.2)	ATCAgctcttcATCGTACTgtcatacaaaagaatatcaagg	cttttgcatacgtataatactaggcttCGAGTgaagagcttgc	Automated (EchoProto)
RhlR_CE	sfGFP Reporter Module (Table 2.2)	GTACgctcttcATCGaatgaggaatgacggaggctt	gcgctgggcctcatctaataaGCTTcgagtGAAGAGCgatc	Automated (EchoProto)

pRhl_AB	sfGFP Reporter Module (Table 2.2)	agtcGCTCTTCatcgGGAGtcctgtgaaatctggcagtt	tcgaattggctaaaaagtgttcTACTcgagcGAAGAGCgcgc	Automated (EchoProto)
LasI_CE	IPTG Detector + eCFP Module (from synthesis)	ATCAgctcttcATCGaATGATCGTTCAGATCGGTCG	GCGTCTGGCTGTTTCCTAATAAgcttCGAGTgaagagcttgc	Automated (EchoProto)
B0040_AF	Default Processor + mCherry Module (Table 2.6)	AGTCGCTCTTCATCGGGAGaggttctgttaagtaactgaaccca	ctcttaagaggtcactgacctaacaCGCTCGAGTGAAGAGCTTGC	Automated (EchoProto)

Table 2.14. DNA Constructs Assembled via Automated Loop Assembly: Shown here are successfully assembled DNA constructs assembled with Loop assembly using the EchoProto template.

Construct Name	Plasmid Backbone	Antibiotic Selection	DNA Parts	Link to Sequence
pOdd1: PBLInd-mCherry	pOdd1 (Table 2.2)	Kanamycin	• PBLInd100_B0034_AC (Table 2.13) • mCherry_CE (Table 2.13) • B0015_EF (Table 2.13)	benchling.com/s/seq-GZuvozOBcFWx75SNatvN?m=slm-EvXvnAGnUXFmK3migJ0K
pOdd1: J23100-mCherry	pOdd1 (Table 2.2)	Kanamycin	• J23100_B0034_AC (Table 2.13) • mCherry_CE (Table 2.13) • B0015_EF (Table 2.13)	benchling.com/s/seq-NA9sMgwo22g9MChF1Qmg?m=slm-i7QlWkXxptUJztdUqcnZ
pOdd2: PBLInd-EL222	pOdd2 (Table 2.2)	Kanamycin	• PBLInd100_B0034_AC (Table 2.13) • EL222_CE (Table 2.13) • B0015_EF (Table 2.13)	benchling.com/s/seq-5y5IyNioaWE29xJpkXwm?m=slm-1eo8OYEEB4Ry2k3BZZV7
pOdd2: J23100-EL222	pOdd2 (Table 2.2)	Kanamycin	• J23100_B0034_AC (Table 2.13) • EL222_CE (Table 2.13) • B0015_EF (Table 2.13)	benchling.com/s/seq-iiUaTlm97fh753jBtYfg?m=slm-uekx5RVwp8PVBTmyJKHf

2.4. Deterministic SBML-Based Modelling

Simulations were performed on a server with 56 Intel® Xeon® E5-2695 v3 (2.30 GHz) CPUs and 755 GB RAM. The operating system was Ubuntu 18.04. Simbiotics was run in parallel mode across all CPU modules. The Python version used was 3.8.

The deterministic SBML models were initially described using Systems Biology Markup Language (SBML; Level 2, Version 4) generated via Complex Pathway Simulator (COPASI; Version 4.36, build 260). Models were simulated using the basico python library (version 0.3.0) with the 'deterministic' simulator method. All simulation data was plotted using the matplotlib Python library (version 3.5.2)[169].

Models were initially simulated for 1440 minutes and 1440-time steps with starting entity amounts as stated in Table 2.15, Table 2.16, and Table 2.17, which acted to prime the model. The final entity amounts following initial simulation were used as starting amounts for a further simulation over 1440 minutes with 1440-time steps. For induced simulations, the starting amount of the inducer for the second simulation was modified based on the concentration of inducer as stated in the main text of Chapter 4. Reactions and parameters used for results shown in chapter 4 can be found in Table 2.18, Table 2.19, and Table 2.20. Rationale behind the model mechanisms is discussed in Chapter 4.

Table 2.15. IPTG Detector Module Deterministic Model Entity Starting Quantities

Entity Name	Description	Number
C12	C12-HSL molecule	0
C4	C4-HSL molecule	0
eCFP	Cyan fluorescent protein	0
IPTG	IPTG inducer molecule	0
LacI	LacI transcription factor	0
LacI2	LacI dimer	0
LacI2_IPTG	LacI dimer bound to 1 IPTG molecule	0
LacI2_IPTG2	LacI dimer bound to 2 IPTG molecules	0
LacI4	LacI dimer-of-dimers	0
LacI4_IPTG1	LacI dimer-of-dimers bound 1 IPTG molecules	0
LacI4_IPTG2	LacI dimer-of-dimers bound 2 IPTG molecules	0
LacI4_IPTG3	LacI dimer-of-dimers bound 3 IPTG molecules	0
LacI4_IPTG4	LacI dimer-of-dimers bound 4 IPTG molecules	0
LasI	LasI C12-HSL synthetase enzyme	0
mRNA_LacI	mRNA encoding LacI	0
mRNA_LasI_eCFP	mRNA encoding LasI and eCFP	0
PCon	DNA encoding LacI under J23100 control	200
pLac	DNA encoding LasI and eCFP under PLac control	200
pLac_LacI4	LacI dimer-of-dimers bound to PLac	0
pLac_LacI4_IPTG	LacI dimer-of-dimers with 1 IPTG bound to PLac	0

Table 2.16. Default Processor Module Deterministic Model Entity Starting Quantities

Entity Name	Description	Number
C12	C12-HSL molecule	0
C4	C4-HSL molecule	0
LasR	LasR transcription factor	0
LasR_Dim	LasR dimer	0
LasR_Dim_1_C12	LasR dimer bound to 1 C12-HSL	0
LasR_Dim_1_C4	LasR dimer bound to 1 C4-HSL	0
LasR_Dim_2_C12	LasR dimer bound to 2 C12-HSLs	0
LasR_Dim_2_C4	LasR dimer bound to 2 C4-HSLs	0
mCherry	Red fluorescent protein	0
mRNA_LasR	mRNA encoding LasR	0
mRNA_mCherry_RhlI	mRNA encoding mCherry and RhlI	0
PCon	DNA encoding LasR under J23100 control	200
PLas	DNA encoding mCherry and RhlI under PLac control	200
PLas_LasR_Dim	LasR dimer bound to PLas	0
PLas_LasR_Dim_1_C12	LasR dimer with 1 C12-HSL bound to PLas	0
PLas_LasR_Dim_1_C4	LasR dimer with 1 C4-HSL bound to PLas	0
PLas_LasR_Dim_2_C12	LasR dimer with 2 C12-HSLs bound to PLas	0
PLas_LasR_Dim_2_C4	LasR dimer with 2 C4-HSLs bound to PLas	0
RhlI	C4-HSL synthetase enzyme	0

Table 2.17. sfGFP Reporter Module Deterministic Model Entity Starting Quantities

Entity Name	Description	Number
C12	C12-HSL molecule	0
C4	C4-HSL molecule	0
mRNA_RhlR	mRNA encoding RhlR	0
mRNA_sfGFP	mRNA encoding sfGFP	0
PCon	DNA encoding RhlR under J23100 control	200
PRhl	DNA encoding sfGFP under PRhl control	200
PRhl_RhlR_Dim	RhlR dimer bound to PRhl	0
PRhl_RhlR_Dim_1_C12	RhlR dimer with 1 C12-HSL bound to PRhl	0
PRhl_RhlR_Dim_1_C4	RhlR dimer with 1 C4-HSL bound to PRhl	0
PRhl_RhlR_Dim_2_C12	RhlR dimer with 2 C12-HSLs bound to PRhl	0
PRhl_RhlR_Dim_2_C4	RhlR dimer with 2 C4-HSLs bound to PRhl	0
RhlR	RhlR transcription factor	0
RhlR_Dim	RhlR dimer	0
RhlR_Dim_1_C12	RhlR dimer bound to 1 C12-HSL	0
RhlR_Dim_1_C4	RhlR dimer bound to 1 C4-HSL	0
RhlR_Dim_2_C12	RhlR dimer bound to 2 C12-HSLs	0
RhlR_Dim_2_C4	RhlR dimer bound to 2 C4-HSLs	0
sfGFP	Green fluorescent protein	0

Table 2.18. IPTG Detector Module Deterministic Model Reactions and Parameters: For each reaction, a name, schema, parameter value, parameter unit, and parameter source is given.

Name	Reaction	Parameter Value	Parameter Unit	Source or Assumption
LacI_Dim	2 * LacI -> LacI2	6.80e-23	Litre/min	By estimation under assumption of fast kinetics, [170]
LacI_IPTG_1_Bind	LacI2 + IPTG -> LacI2_IPTG	1.61e-19	Litre/min	Based on [171]
LacI_IPTG_1_Unbind	LacI2_IPTG -> LacI2 + IPTG	0.2	1/min	Based on [171]
LacI_IPTG_2_Bind	LacI2_IPTG + IPTG -> LacI2_IPTG2	8.05e-20	Litre/min	Based on [171]
LacI_IPTG_2_Unbind	LacI2_IPTG2 -> LacI2_IPTG + IPTG	0.4	1/min	Based on [171]
LacI_Quad_Bind	2 * LacI2 -> LacI4	6.80e-23	Litre/min	By estimation under assumption of fast kinetics
LacI_Quad_Unbind	LacI4 -> 2 * LacI2	6.80e-39	1/min	By estimation under assumption of slow kinetics
LacI_Quad_IPTG_1_Bind	LacI4 + IPTG -> LacI4_IPTG1	1.61e-19	Litre/min	Assumed same kinetics as dimer binding
LacI_Quad_IPTG_1_Unbind	LacI4_IPTG1 -> LacI4 + IPTG	0.2	1/min	As above
LacI_Quad_IPTG_2_Bind	LacI4_IPTG1 + IPTG -> LacI4_IPTG2	8.05-20	Litre/min	As above
LacI_Quad_IPTG_2_Unbind	LacI4_IPTG2 -> LacI4_IPTG1 + IPTG	0.4	1/min	As above
LacI_Quad_IPTG_3_Bind	LacI4_IPTG2 + IPTG -> LacI4_IPTG3	1.61e-19	Litre/min	As above
LacI_Quad_IPTG_3_Unbind	LacI4_IPTG3 -> LacI4_IPTG2 + IPTG	0.2	1/min	As above
LacI_Quad_IPTG_4_Bind	LacI4_IPTG3 + IPTG -> LacI4_IPTG4	8.05e-20	Litre/min	As above
LacI_Quad_IPTG_4_Unbind	LacI4_IPTG4 -> LacI4_IPTG3 + IPTG	0.4	1/min	As above
pLac_Repress	LacI4 + pLac -> pLac_LacI4	4.04e-18	Litre/min	Based on [171]
pLac_Unrepress	pLac_LacI4 -> LacI4 + pLac	0.00063	1/min	Based on [171]
pLac_IPTG_Repress	pLac + LacI4_IPTG1 -> pLac_LacI4_IPTG	2.01e-20	Litre/min	Based on [171]
pLac_IPTG_Unrepress	pLac_LacI4_IPTG -> pLac + LacI4_IPTG1	0.063	1/min	Based on [171]
pLac_LacI_IPTG_1_Bind	pLac_LacI4 + IPTG -> pLac_LacI4_IPTG	3.72e-20	Litre/min	Based on [171]
pLac_LacI_IPTG_1_Unbind	pLac_LacI4_IPTG -> pLac_LacI4 + IPTG	1.0	1/min	Based on [171]

pLac_LacI_IPTG_2_Bind	pLac_LacI4_IPTG + IPTG -> pLac + LacI4_IPTG2	2.01-20	Litre/min	Based on [171]
LacI_Deg	LacI ->	1.48e-05	1/min	Based on [172]
LacI_Dim_Deg	LacI2 -> LacI	1.48e-06	1/min	Assumed slower than LacI degradation
LacI_Dim_IPTG_Deg	LacI2_IPTG -> LacI + IPTG	1.48e-06	1/min	Assumed similar to unbound LacI dimer degradation
LacI_Dim_IPTG2_Deg	LacI2_IPTG2 -> LacI + 2 * IPTG	1.48e-06	1/min	Assumed similar to unbound LacI dimer degradation
Tx_LacI	PCon -> mRNA_LacI + PCon	0.0008167	1/min	Calculated using [173]
Tl_LacI	mRNA_LacI -> mRNA_LacI + LacI	0.0007167	1/min	Calculated using [174]
Tx_pLac	pLac -> mRNA_LasI_eCFP + pLac	0.00065	1/min	Calculated using [173]
Tl_LasI	mRNA_LasI_eCFP -> mRNA_LasI_eCFP + LasI	0.0013167	1/min	Calculated using [174]
Tl_eCFP	mRNA_LasI_eCFP -> mRNA_LasI_eCFP + eCFP	0.00105	1/min	Calculated using [174]
Syn_C12	LasI -> LasI + C12	0.096	1/min	Using [175]
Deg_mRNA_LacI	mRNA_LacI ->	2.83e-05	1/min	Based on [176]
Deg_mRNA_LasI_eCFP	mRNA_LasI_eCFP ->	2.83e-05	1/min	Based on [176]
Deg_LasI	LasI ->	3.22e-06	1/min	Based on [177]
Deg_C12	C12 ->	4.72e-06	1/min	Based on [178]
Deg_eCFP	eCFP ->	3.22e-06	1/min	Based on [177]
Deg_C4	C4 ->	3.69e-6	1/min	Based on [179]

Table 2.19. Default Processor Module Deterministic Model Reactions and Parameters: For each reaction, a name, schema, parameter value, parameter unit, and parameter source are given.

Name	Reaction	Parameter Value	Parameter Unit	Source or Assumption
Tx_LasR	PCon -> PCon + mRNA_LasR	0.00127	1/min	Calculated using [173]
Tl_LasR	mRNA_LasR -> mRNA_LasR + LasR	0.0011	1/min	Calculated using [174]
Tx_PLas	PLas_LasR_Dim_2_C12 -> PLas_LasR_Dim_2_C12 + mRNA_mCherry_RhlI	0.0015	1/min	Calculated using [173]
Bg_Tx_PLas	PLas -> PLas + mRNA_mCherry_RhlI	1.05E-05	1/min	Assumed slower than above
Tl_RhlI	mRNA_mCherry_RhlI -> mRNA_mCherry_RhlI + RhlI	0.001317	1/min	Calculated using [174]
Tl_mCherry	mRNA_mCherry_RhlI -> mRNA_mCherry_RhlI + mCherry	0.00113	1/min	Calculated using [174]
Deg_mRNA_LasR	mRNA_LasR ->	2.83E-05	1/min	Based on [176]
Deg_mRNA_mCherry_RhlI	mRNA_mCherry_RhlI ->	2.83E-05	1/min	Based on [176]
Deg_LasR	LasR ->	0.889	1/min	Estimation based on assumption of fast degradation [180]
s[178]Deg_mCherry	mCherry ->	1.48E-05	1/min	By estimation based on [177]
Deg_RhlI	RhlI ->	1.48E-05	1/min	By estimation based on [177]
Deg_C12	C12 ->	4.72E-06	1/min	Based on [178]
Deg_C4	C4 ->	3.69E-06	1/min	Based on [179]
Syn_C4	RhlI -> RhlI + C4	0.046	1/min	From [181]
LasR_Dim_Bind	2 * LasR -> LasR_Dim	6.50E-10	Litre/min	Assumed fast binding [182] [183]
LasR_Dim_Unbind	LasR_Dim -> 2 * LasR	1000	1/min	Assumed fast unbinding [182] [183]
LasR_Dim_C12_1_Bind	LasR_Dim + C12 -> LasR_Dim_1_C12	1.59E-14	Litre/min	By estimation based on [178]
LasR_Dim_C12_1_Unbind	LasR_Dim_1_C12 -> LasR_Dim + C12	1.00E-09	1/min	By estimation based on [178]
LasR_Dim_C12_2_Bind	LasR_Dim_1_C12 + C12 -> LasR_Dim_2_C12	1.59E-14	Litre/min	By estimation based on [178]
LasR_Dim_C12_2_Unbind	LasR_Dim_2_C12 -> LasR_Dim_1_C12 + C12	1.00E-09	1/min	By estimation based on [178]

PLas_Activate	PLas + LasR_Dim_2_C12 -> PLas_LasR_Dim_2_C12	3.25E-16	Litre/min	By estimation based on [178]
PLas_Unactivate	PLas_LasR_Dim_2_C12 -> PLas + LasR_Dim_2_C12	0.00102	1/min	By estimation based on [178]
Bg_C12_1_PLas_Activate	PLas + LasR_Dim_1_C12 -> PLas_LasR_Dim_1_C12	1.63E-16	Litre/min	By estimation based on [178]
Bg_C12_1_PLas_Unactivate	PLas_LasR_Dim_1_C12 -> PLas + LasR_Dim_1_C12	0.00204	1/min	By estimation based on [178]
LasR_Dim_C4_1_Bind	LasR_Dim + C4 -> LasR_Dim_1_C4	7.95E-25	Litre/min	By estimation based on [178]
LasR_Dim_C4_1_Unbind	LasR_Dim_1_C4 -> LasR_Dim + C4	100	1/min	By estimation based on [178]
LasR_Dim_C4_2_Bind	LasR_Dim_1_C4 + C4 -> LasR_Dim_2_C4	7.95E-25	Litre/min	By estimation based on [178]
LasR_Dim_C4_2_Unbind	LasR_Dim_2_C4 -> LasR_Dim_1_C4 + C4	100	1/min	By estimation based on [178]
Ct_PLas_C4_1_Activate	PLas + LasR_Dim_1_C4 -> PLas_LasR_Dim_1_C4	8.13E-18	Litre/min	By estimation based on [178]
Ct_PLas_C4_1_Unactivate	PLas_LasR_Dim_1_C4 -> PLas + LasR_Dim_1_C4	0.102	1/min	By estimation based on [178]
Ct_PLas_C4_2_Activate	PLas + LasR_Dim_2_C4 -> PLas_LasR_Dim_2_C4	1.63E-17	Litre/min	By estimation based on [178]
Ct_PLas_C4_2_Unactivate	PLas_LasR_Dim_2_C4 -> PLas + LasR_Dim_2_C4	0.051	1/min	By estimation based on [178]
Tx_PLas_LasR_1_C12	PLas_LasR_Dim_1_C12 -> PLas_LasR_Dim_1_C12 + mRNA_mCherry_RhlI	0.0015	1/min	Assumed same rate as when bound to LasR_Dim_2_C12 [184]
Tx_PLas_LasR_1_C4	PLas_LasR_Dim_1_C4 -> PLas_LasR_Dim_1_C4 + mRNA_mCherry_RhlI	0.0015	1/min	Assumed same rate as when bound to LasR_Dim_2_C12 [184]
Tx_PLas_LasR_2_C4	PLas_LasR_Dim_2_C4 -> PLas_LasR_Dim_2_C4 + mRNA_mCherry_RhlI	0.0015	1/min	Assumed same rate as when bound to LasR_Dim_2_C12 [184]
Tx_PLas_LasR_2	PLas_LasR_Dim -> PLas_LasR_Dim + mRNA_mCherry_RhlI	0.0015	1/min	Assumed same rate as when bound to LasR_Dim_2_C12 [184]

Table 2.20. sfGFP Reporter Module Deterministic Model Reactions and Parameters: For each reaction, a name, schema, parameter value, parameter unit, and parameter source are given.

Name	Reaction	Parameter Value	Parameter Unit	Source or Assumption
Tx_RhlR	PCon -> PCon + mRNA_RhlR	0.00125	1/min	Calculated using [173]
Tl_RhlR	mRNA_RhlR -> mRNA_RhlR + RhlR	0.0011	1/min	Calculated using [174]
Tl_sfGFP	mRNA_sfGFP -> mRNA_sfGFP + sfGFP	0.001116	1/min	Calculated using [174]
RhlR_Dim_Bind	2 * RhlR -> RhlR_Dim	6.5e-10	Litre/min	Assumed fast binding [185]
RhlR_Dim_Unbind	RhlR_Dim -> 2 * RhlR	1000	1/min	By Estimation
RhlR_Dim_C4_1_Bind	RhlR_Dim + C4 -> RhlR_Dim_1_C4	1.59e-14	Litre/min	Assumed similar to Las mechanism
RhlR_Dim_C4_1_Unbind	RhlR_Dim_1_C4 -> RhlR_Dim + C4	1e-09	1/min	Assumed similar to Las mechanism
RhlR_Dim_C4_2_Bind	RhlR_Dim_1_C4 + C4 -> RhlR_Dim_2_C4	1.59e-14	Litre/min	Assumed similar to Las mechanism
RhlR_Dim_C4_2_Unbind	RhlR_Dim_2_C4 -> RhlR_Dim_1_C4 + C4	1e-09	1/min	Assumed similar to Las mechanism
PRhl_Activate	PRhl + RhlR_Dim_2_C4 -> PRhl_RhlR_Dim_2_C4	3.25e-16	Litre/min	By estimation based on [186] and [187]
PRhl_Unactivate	PRhl_RhlR_Dim_2_C4 -> PRhl + RhlR_Dim_2_C4	0.00102	1/min	By estimation based on [186] and [187]
Bg_C4_1_PRhl_Activate	RhlR_Dim_1_C4 + PRhl -> PRhl_RhlR_Dim_1_C4	3.25e-16	Litre/min	By estimation based on [186] and [187]
Bg_C4_1_PRhl_Unactivate	PRhl_RhlR_Dim_1_C4 -> PRhl + RhlR_Dim_1_C4	0.00102	1/min	By estimation based on [186] and [187]
RhlR_Dim_C12_1_Bind	RhlR_Dim + C12 -> RhlR_Dim_1_C12	7.95e-23	Litre/min	Assumed similar to Las mechanism (by estimation)

RhlR_Dim_C12_1_Unbind	RhlR_Dim_1_C12 -> RhlR_Dim + C12	0.1	1/min	Assumed similar to Las mechanism (by estimation)
RhlR_Dim_C12_2_Bind	RhlR_Dim_1_C12 + C12 -> RhlR_Dim_2_C12	7.95e-23	Litre/min	Assumed similar to Las mechanism (by estimation)
RhlR_Dim_C12_2_Unbind	RhlR_Dim_2_C12 -> RhlR_Dim_1_C12 + C12	0.1	1/min	Assumed similar to Las mechanism (by estimation)
Ct_PRhl_C12_1_Activate	RhlR_Dim_1_C12 + PRhl -> PRhl_RhlR_Dim_1_C12	3.25e-16	Litre/min	Assumed similar to Las mechanism (by estimation)
Ct_PRhl_C12_1_Unactivate	PRhl_RhlR_Dim_1_C12 -> RhlR_Dim_1_C12 + PRhl	0.00102	1/min	Assumed similar to Las mechanism (by estimation)
Ct_PRhl_C12_2_Activate	RhlR_Dim_2_C12 + PRhl -> PRhl_RhlR_Dim_2_C12	3.25e-16	Litre/min	Assumed similar to Las mechanism (by estimation)
Ct_PRhl_C12_2_Unactivate	PRhl_RhlR_Dim_2_C12 -> RhlR_Dim_2_C12 + PRhl	0.00102	1/min	Assumed similar to Las mechanism (by estimation)
Tx_PRhl	PRhl_RhlR_Dim_2_C4 -> PRhl_RhlR_Dim_2_C4 + mRNA_sfGFP	0.375	1/min	Calculated using [173], but assumed faster kinetics
Bg_Tx_PRhl	PRhl -> PRhl + mRNA_sfGFP	0.002625	1/min	Assumed slower than above
Tx_PRhl_RhlR_1_C4	PRhl_RhlR_Dim_1_C4 -> PRhl_RhlR_Dim_1_C4 + mRNA_sfGFP	0.28125	1/min	By estimation based on [186] and [187]
Tx_PRhl_RhlR_1_C12	PRhl_RhlR_Dim_1_C12 -> PRhl_RhlR_Dim_1_C12 + mRNA_sfGFP	0.00375	1/min	By estimation based on [186] and [187]
Tx_PRhl_RhlR_2_C12	PRhl_RhlR_Dim_2_C12 -> PRhl_RhlR_Dim_2_C12 + mRNA_sfGFP	0.01875	1/min	By estimation based on [186] and [187]
Deg_mRNA_RhlR	mRNA_RhlR ->	2.83e-05	1/min	Based on [176]
Deg_RhlR	RhlR ->	0.000889	1/min	Based on [177]
Deg_C4	C4 ->	3.69e-06	1/min	Based on [178]
Deg_C12	C12 ->	4.72e-06	1/min	Based on [179]
Deg_mRNA_sfGFP	mRNA_sfGFP ->	2.83e-05	1/min	Based on [176]
Deg_sfGFP	sfGFP ->	1.4817e-05	1/min	Based on [177]
Bg_PRhl_Activate	RhlR_Dim + PRhl -> PRhl_RhlR_Dim	3.25e-16	Litre/min	By estimation based on [186] and [187]

Bg_PRhl_Unactivate	PRhl_RhlR_Dim -> RhlR_Dim + PRhl	0.00102	1/min	By estimation based on [186] and [187]
Tx_PRhl_Rhl_2	PRhl_RhlR_Dim -> mRNA_sfGFP + PRhl_RhlR_Dim	0.0	1/min	By estimation based on [186] and [187]
C4_Bind_PRhl_RhlR	PRhl_RhlR_Dim + C4 -> PRhl_RhlR_Dim_1_C4	1.1925e-14	Litre/min	By estimation based on [186] and [187]
C4_Bind_PRhl_RhlR_1_C4	PRhl_RhlR_Dim_1_C4 + C4 -> PRhl_RhlR_Dim_2_C4	1.1925e-14	Litre/min	By estimation based on [186] and [187]
C12_Bind_PRhl_RhlR	PRhl_RhlR_Dim + C12 -> PRhl_RhlR_Dim_1_C12	5.9625e-23	Litre/min	By estimation based on [186] and [187]
C12_Bind_PRhl_RhlR_1_C12	PRhl_RhlR_Dim_1_C12 + C12 -> PRhl_RhlR_Dim_2_C12	5.9625e-23	Litre/min	By estimation based on [186] and [187]

2.5. Agent-Based Modelling

Simulations were performed on a server with 56 Intel® Xeon® E5-2695 v3 (2.30 GHz) CPUs and 755 GB RAM. The operating system was Ubuntu 18.04. Simbiotics was run in parallel mode across all CPU modules. The Python version used was 3.8, and the Java version was JDK 11.0.17.

2.5.1. General model definition and simulation settings

Simbiotics (version 1.1) was used to build and simulate the agent-based models presented in chapter 4. The simulation world was a 3-dimensional cube with a grid-depth of 3. Cells were modelled as rods (length: 1.5 μm; radius: 0.25 μm) with permeable cell membranes. Cellular behaviour was defined by SBML models described in section 2.4. The native SBML simulator was modified to use basico (see section 2.4, and simulations were performed in Python. Results for SBML simulator were imported back into Simbiotics and converted such that the results were in the same fomrat as that produced by the native SBML simulator. This allowed the SBML results from basico to be used by Simbiotics in the same way. The Simbiotics model was simulated for 1200 minutes with 1200 timesteps. The SBML models were simulated every 60 timesteps, and each model was simulated for 60 minutes with 60 timesteps. For simulations with an inducer added, the relevant inducer in the appropriate amount was applied to the centre of the system at time step 0. SBML reaction parameters were the same as stated in section 2.4. To obtain simulated single cell data, results from simulation of each SBML model was appended to a CSV file at the end of each SBML simulation step.

2.5.2. Data analysis

Where shown, background noise was calculated by first determining the mean across all uninduced simulation replicates. The noise was then calculated as 1 + standard deviation of each uninduced simulation replicate divided by the mean of all uninduced simulation replicates.

The 2^{nd} degree polynomial curves were determined using the polyfit function from the numpy (version 1.23.1) python library. The r^2 value was determined by calculating the sum of squared errors divided by the total sum of squares and subtracting this value from 1.

2.5.3. Simulating cell growth

For simulated cell growth, the Monod growth module in Simbiotics was used with parameters listed in Table 2.21. Growth was modelled as nutrient-dependent, with an entity of 'food' added to the centre of the system at time point 0. One cell type was implemented for cell growth simulation, with no SBML-derived cellular behaviour attached.

Table 2.21. Simbiotics Model Parameters: Parameters used in the Simbiotics model. All parameters were unitless.

Parameter	Value
Molecule Diffusion Coefficients	0.1
Molecule Extracellular Degradation Coefficients	0
Molecule Membrane Permeation Coefficients	0.14
Brownian Constant	2.5
Friction Constant	0
Grid Depth	3
Monod Nutrient Amount	1
Monod Maximum Rate	0.12
Monod Kinetic Rate	0.005

The cell growth experimental data was obtained by creating an overnight culture of *E. coli* DH5α in 10 mL LB media with no antibiotic. Cells were pelleted by centrifugation at 4,500 RPM for 10 minutes, and supernatant was removed. The cells were re-suspended in 10 mL of sterile MiliQ water and subsequently pelleted by centrifugation as above. Supernatant was discarded and the cell pellet was re-suspended in 10 mL of fresh LB media. OD_{600} of the cells was determined by transferring to a 1 mL cuvette and measuring absorbance with a spectrophotometer. The cells were diluted to an OD_{600} of 1.0 in fresh LB media, and 10 µL was added to four wells of a black 96-well plate with flat, clear bottomed wells. 90 µL of fresh LB was added to each well, and 100 µL of LB media was added to a further three wells. A clear, permeable seal (Breathe-Easy; Scientific Laboratory Supplies) was adhered to the top of the plate. The microplate was incubated in a microplate reader (Clariostar Plus; BMG Labtech) at 37^{o}C with shaking at 400 RPM for 20 hours. Absorbance readings at OD_{600} were taken every 30 minutes. Absorbance readings for all empty wells were averaged and subtracted from the wells containing cells and pure LB. Absorbance values from the wells containing only LB were averaged and subtracted from the absorbance values for wells containing cells. The OD readings for the cells were then converted to 'Equivalent Microsphere Particles' as described in section 2.6.

To calibrate simulated cell growth to experimental data, the initial number of cells in the experimental samples (~1e8 cells in 100 uL, where the number of cells was approximated as equivalent microsphere particles) was set as equal to the initial number of cells in the simulation.

2.6. Plate Reader Calibration

Where stated, standard curves of calibrants were used to convert plate reader data into absolute units, based on previously reported methods[63] In a black 96-well plate with flat, clear bottomed wells, 1 in 2 serial dilutions of the standard calibrants were prepared to a final volume of 100 µL across 11 wells, with a twelfth well containing no calibrant. Each serial dilution was replicated twice. The calibrants used were as follows: fluorescein (Merck) in Phosphate-Buffered Saline (PBS), sulforhodamine 101 (Merck) in PBS, cascade blue (ThermoFisher Scientific) in water, and 950 nm monodisperse silica microspheres (Nanocym) in water. The first concentration of each calibrant in the serial dilutions was 10 µM for fluorescein, 2 µM for sulforhodamine 101, 10 µM cascade blue, and $3e^9$ particles/mL for the microspheres. The serial dilutions were prepared using the BiomationScripter OTProto Standard iGEM Calibration Template. The automation protocol can be found in section 9.1. After serial dilutions were prepared, the contents were measured using the same plate reader to be used for experiments (Clariostar Plus; BMG Labtech). Fluorescence measurements were made for wells containing fluorescein, sulforhodamine 101, and cascade blue using the same excitation and emission wavelengths to be used for measurement of fluorescent proteins in experiments (470-15 / 515-20, 570-15 / 620-20, and 430-20 / 480-20 respectively). The absorbance at 600 nm was measured for wells containing microspheres. Gain and focus settings used were identical to those used for experimental measurements.

Following measurement in the plate reader, custom Python scripts were used to analyse and plot the data. Raw data values for all wells in the dilution were blanked using averaged values of the solvent-only wells (PBS for fluorescein and sulforhodamine, and water for cascade blue and the microspheres). The mean of each dilution replicate was calculated, along with standard deviation. These values were then plotted on scatter graphs of calibrant amounts vs the measure value on linear-linear and log-log plots to ensure linear correlation on both plots for the five most highly

concentration dilutions. For each calibrant, the five most highly concentrated dilutions were used to calculate a calibration factor for converting arbitrary units into absolute units using equation 2.1. The absolute units were Molecules of Equivalent Fluorescein (MEFL), Molecules of Equivalent Sulforhodamine 101 (MESR), Molecules of Equivalent Cascade Blue (MECB), and Equivalent Microsphere Particles.

$$CF_{calibrant} = \frac{1}{5}\sum_{d=1}^{5}\left(\frac{n_d}{m_d}\right) \quad \textbf{(2.1)}$$

In equation 2.1, $CF_{calibrant}$ is the calibration factor for a specific calibrant, d is the dilution index (where 1 is highest concentration), n_d is the number of particles in dilution d, and m_d is the mean, blanked measurement from the plate reader for dilution d. Results are shown in section 9.2.

2.7. Sensynova Characterisation Procedures

Unless stated otherwise, samples were added to a final volume of 100 µL in a black 96-well plate with flat, clear bottomed wells. A clear, permeable seal (Breathe-Easy; Scientific Laboratory Supplies) was adhered to the top of the plate prior to incubation in a microplate reader (Clariostar Plus; BMG Labtech) at 37°C with shaking at 400 RPM for 20 hours. Gain, focus, and wavelength settings remained the same for all experiments.

2.7.1. Plate reader data handling

All data handling and graphing was accomplished using custom Python scripts with the Matplotlib and Numpy libraries. Unless stated otherwise, all plate reader data was analysed as described here. The mean measurement value across all empty wells was calculated and subtracted from measurement values for all occupied wells. The mean value of each 'blank' well (i.e., wells with just media and non-cell additives) was calculated and subtracted from the appropriate cell-containing sample. The resulting values for each cell sample was converted from arbitrary units to absolute units as described in sub-section 2.6. The mean across cell sample replicates was calculated, along with standard error. For growth-corrected data, the calibrated fluorescence value(s) of each sample was divided by the calibrated OD_{600} value for the same sample. Fold change was calculated by dividing the non-averaged value of each sample replicate by the mean across all replicates of the relevant negative control. The relative data across each replicate was used to calculate mean fold change and standard error. Unless specified otherwise, background noise was calculated by dividing the non-

averaged value of each negative control replicate by the mean value across all negative control replicates, and then determining the range between the maximum and minimum relative replicate values.

2.7.2. Flow cytometry gating and voltage settings

For flow cytometry experiments, an Attune NxT with CytKick Max autosampler (ThermoFisher) with blue, red, yellow, and violet lasers was used. To determine voltage and gating for fluorescent and non-fluorescent cells, overnight cultures of untransformed cells and each cell type (IPTG detector, default processor, sfGFP reporter) induced with their canonical inducer (10 μM IPTG, 10 μM C12-HSL, 10 μM C4-HSL respectively) were prepared. Cells were pelleted by centrifugation at 4,500 RPM for 10 minutes and supernatant discarded. The cells were re-suspended in autoclave sterilised and syringe filtered (0.22 μm filter) PBS before being pelleted as again as described above. Cells were re-suspended in PBS and diluted 1 in 10. The final cell solutions were flowed through the flow cytometer in manual mode to determine voltage settings for differentiating each of the fluorescent cells and the non-fluorescent cells. These setting remained the same for all experiments described here. All future experiments used the flow cytometer in autosampler mode. Data was exported in FCS file format for analysis and visualisation using Python and the FlowCytometryTools (0.5.0) library.

2.7.3. Bacterial cell preparation

Unless specified otherwise, all cell cultures were prepared as stated here before being added to the microplate. *E. coli* DH5α cells were transformed with the relevant plasmid: Default Processor Module + mCherry (Table 2.6) or the processor cells, and sfGFP Reporter Module (Table 2.2) for the reporter cells. For the detector cells, the IPTG detector + eCFP module obtained with third-party synthesis by ATUM directly into the *pSB1C3* plasmid, which was then transformed into *E. coli* DH5α.

For each cell type, a single transformant colony was streaked onto a separate LB agar plate with appropriate antibiotic selection. 10 mL of LB media and chloramphenicol at the appropriate amount (Table 2.1) was added to a 50 mL falcon tube and inoculated with a single colony from the streak plate. The overnight culture was incubated at 37^{o}C for 16 to 17 hours with shaking at 200 RPM. Cells were pelleted at 4^{o}C and 4,500 RPM for 10 minutes. Supernatant was removed and cells were resuspended in 10 mL of sterile MiliQ water. Resuspended cells were pelleted as above, supernatant removed,

and resuspended in 10 mL fresh LB media. The OD_{600} of resuspended cells was measured with a UV-Vis spectrophotometer and the culture diluted to an OD_{600} of 1.0 in LB media.

2.7.4. BiomationScripter Sensynova Template

Unless specified otherwise, all experimental setup was performed using the Opentrons-2 liquid handler. The automation protocol was generated using the BiomationScripter Sensynova Template.

2.7.5. Cell growth rate experiments

Cells were prepared as described in sub-section 2.6.2. Experimental setup was performed manually, with a separate experiment for each cell type. All wells were made up to 100 µL with LB media. Untransformed cells were used as a control, where 10 µL of cells was added to 90 µL of LB media per well. For IPTG detector cells, 10 µL of cells was added to each well with 0.5 µL chloramphenicol. Plain LB media was used as a blank control. For default processor cells, 10 µL of cells were added to each well with 0.5 µL chloramphenicol and either 0, 0.5, 1, or 10 µL of DMSO (dimethyl sulfoxide; syringe filter sterilised through a 0.22 µm filter). For sfGFP reporter cells, 10 µL of cells were added to each well with 0.5 µL chloramphenicol and either 0, 0.5, 1, or 10 µL of DMSO. Cultures were incubated, OD_{600} measured, and data analysed as described previously.

2.7.6. Dose-response curve experiments

Cells were prepared as described in sub-section 2.6.2. A separate experiment was performed per cell type. Experimental setup was performed using the Opentrons-2 liquid handler. Automation protocols for characterising the detector and processor cells were generated using BiomationScripter (version 0.2) and the Sensynova Template (version 1.0). The automation protocol for reporter cells was generated using BiomationScripter (version 0.2) and the Sensynova Template (version 2.0*). Full protocols and plate maps can be found in section 9.3.

2.7.7. Cross talk experiments

Cells were prepared as described in sub-section 2.6.2. A separate experiment was performed per cell type. Experimental setup was performed using the Opentrons-2

* For monoculture experiments, version 1.0 and 2.0 of the Sensynova Template generate effectively identical liquid handling instructions and differed mainly in the format required for user input.

liquid handler. Automation protocols were generated using BiomationScripter (version 0.1) and the Sensynova Template (version 1.0). Full protocols and plate maps can be found in section 9.4.

2.7.8. Homoserine-lactone synthesis validation experiments

For agar plate experiments, reporter cells were inoculated into LB media in the absence of any inducer and incubated overnight for 16 hours. Untransformed DH5α cells and default processor cells were also inoculated in LB media in the presence and absence of 10 μM C12-HSL. 300 μL of reporter cells were plated onto six LB+CAM agar plates and left to dry at room temperature for 3 hours. and incubated for 16 hours. Cells were pelleted by centrifugation at 4500 RPM for 5 mins, and 1 mL of supernatant was transferred into 1.5 mL tubes. Supernatant was then spun for 5 mins at 12,000 RPM to remove any remaining cells. To five of the dried agar plates, 10 μL of either 1 mM C4-HSL, supernatant from untransformed cells incubated with C12-HSL, supernatant from untransformed cells alone, supernatant from processor cells incubated with C12-HSL, or supernatant from processor cells alone was spotted in the centre. To the final plate, nothing was added. The plates were left to dry at room temperature for 1 hour before inverting and incubating overnight at 37C. Plates were imaged under UV light with an ethidium bromide filter. Images were false-coloured green. Completely unaltered images can be seen in section 9.5.

The above was repeated, except uninduced processor cells were used instead of uninduced reporter cells, supernatant from IPTG detector cells either induced with 10 mM IPTG or not induced was used instead of processor cell supernatant, and C12-HSL was used as a control instead of C4-HSL.

For the plate reader experiment, samples were prepared manually. Processor cells were prepared as described in sub-section 2.6.2. Detector and untransformed cells were prepared similarly, except 10 mM of IPTG was added to each culture. The detector and untransformed cells were pelleted by centrifugation at 4500 RPM for 5 mins, and 1 mL of supernatant was transferred into 1.5 mL tubes. Supernatant was then spun for 5 mins at 12,000 RPM to remove any remaining cells. Processor cells were washed in water, resuspended in LB media, and diluted to an OD_{600} of 1.0 as described previously. 10 μL of processor cells were added to 16 wells, along with 0.5 μL chloramphenicol. To four wells, 1.0 μL of DMSO was added. To four wells, 1.0 μL

of 1 mM C12-HSL was added (final concentration of 10 µM). To four wells, 1.0 µL of supernatant from untransformed cells was added, and 1.0 µL of supernatant from the detector cells was added to the remaining four. All wells were made up to 100 µL with LB media. Cultures were incubated, OD_{600} and red fluorescence measured, and data analysed as described previously.

2.7.9. Noise propagation plate reader experiment

Cells were prepared as described in sub-section 2.6.2. A separate experiment was performed per cell type. Experimental setup was performed using the Opentrons-2 liquid handler. Automation protocols were generated using BiomationScripter (version 0.2) and the Sensynova Template (version 2.0). Full protocols and plate maps can be found in section 9.6.

2.7.10. Noise propagation flow cytometry experiment

Following approximately 20 hours of incubation in a plate reader, cell-containing samples were transferred u-bottomed 96-well plates. Samples were centrifuged for 10 minutes at 4,500 RPM, and supernatant was removed. Cells were re-suspended in 100 µL of autoclave sterilised and syringe filtered (0.22 µm filter) PBS and centrifuged again as above. Supernatant was removed and cells were re-suspended in 100 µL of PBS. Into a clean u-bottomed 96 well plate, 90 µL of PBS and 10 µL of sample was added to each well. Cells were analysed by the Attune NxT in autosampler mode, with voltages as determined previously.

2.7.11. 1:1:1 biosensor plate reader experiment

Cells were prepared as described in sub-section 2.6.2. Experimental setup was performed using the Opentrons-2 liquid handler. Co-cultures were prepared by adding 10 µL of each cell culture at a density of OD_{600} = 1.0. Automation protocols were generated using BiomationScripter (version 0.2) and the Sensynova Template (version 2.0). Full protocols and plate maps can be found in section 9.7.

2.7.12. 1:1:1 biosensor flow cytometry experiment

Following approximately 20 hours of incubation in a plate reader, a sub-set of cell-containing samples were transferred u-bottomed 96-well plates. Samples were prepared and analysed as described in sub-section 2.6.10.

2.7.13. Characterisation of cell ratios

Experimental setup followed the general protocols described in section 2.6.11. The major difference was that co-cultures were prepared by mixing different ratios of cells, rather than just 1:1:1. Ratios were calculated by volume, as each cell type had a cell density of OD_{600} = 1.0. A list of tested cell ratios and the volumes they were added at can be found in Table 2.22. Automation protocols were generated using BiomationScripter (version 0.2) and the Sensynova Template (version 2.0). Full protocols and plate maps can be found in section 9.8.

Table 2.22. Cell Ratios

Cell Type Volume (µL)		
Detectors	**Processors**	**Reporters**
2	24	2
3	7	20
5	10	15
6	13	11
7	4	19
10	10	10
11	18.5	0.5
18	2	10
19	7	4

2.7.14. Design of Experiments Main Effects Screening

Experimental setup was the same for all experiments. Experiments were performed on separate days for each cell type, and the augmented design runs were also performed on separate days. For all run conditions, uninduced and induced samples were prepared in triplicate. Various medias were prepared using proportions stated in Table 2.23. The stated chemicals were dissolved in 500 mL of MiliQ water and sterilised by autoclaving at 121^{o}C for 20 minutes.

Table 2.23. Medias for Main Effects Screening Design: Each row defines a different media. The chemicals were mixed in a duran bottle in the weights indicated, dissolved in 500 mL MiliQ water, and sterilises as described in section 2.1

Media Compositions		
Tryptone (g)	**Sodium Chloride (g)**	**Yeast Extract (g)**

0	0	0
10	10	5
10	0	0
0	10	5
5	5	2.5
0	0	5
0	10	0
10	10	0
10	0	5

Detector, processor, and reporter cells were prepared as described in section 2.6.3 however cells were incubated at temperatures according to the main effects screening designs section 9.9, and after washing in sterile water were re-suspended in media also determined by the screening designs. For all experiments, three black 96-well plate with flat, clear bottoms, cells were used, one for each incubation temperature as determined by the main effects screening design. Cells were to wells of the appropriate plate such that the final OD_{600} would match the values stated in the main effects screening design. For the detector, processor, and reporter cells, 1 mM IPTG, 10 µM C12-HSL, and 10 µM C4-HSL was added to induced samples respectively. Chloramphenicol was added to the appropriate amount (Table 2.1) and the cultures were made up to 100 µL with the appropriate media. A clear, permeable seal (Breathe-Easy; Scientific Laboratory Supplies) was adhered to the top of each plate and the cultures were incubated at the appropriate temperatures with shaking at 200 RPM. Fluorescence and absorbance readings were taken every hour as described previously but using a Clariostar (BMG Labtech) plate reader.

The fluorescence values obtained from the plate reader were corrected based on cell density, and replicate values were averaged. Fold change in fluorescence was determined by making the induced samples relative to uninduced samples. The fold change values were inputted into JMP Pro 13 (JMP Statistical Discovery LLC), and data analysed using a standard least squares effect screening model.

2.8. Optical Communication Experiments

2.8.1. Bacterial luciferase characterisation

E. coli cells were transformed with pSB1C3-Lux (Table 2.2). A single transformant colony was streaked onto a separate LB agar plate with chloramphenicol antibiotic

selection. 10 mL of LB media and the chloramphenicol at the appropriate amount (Table 2.1) was added to a 50 mL falcon tube and inoculated with a single colony from the streak plate. The overnight culture was incubated at 37ºC for 16 to 17 hours with shaking at 200 RPM. Cells were pelleted at 4ºC and 4,500 RPM for 10 minutes. Supernatant was removed and cells were resuspended in 10 mL of fresh LB media. The OD_{600} of resuspended cells was measured with a UV-Vis spectrophotometer and the culture diluted to an OD_{600} of 1.0 in LB media. Experimental setup was performed using the Opentrons-2 liquid handler (Section 9.11). Automation protocols were generated using BiomationScripter (version 0.2) and the Sensynova Template (version 2.0). Full protocols and plate maps can be found in section 9.10. The culture plate was incubated in a Clariostar Plus for 20 hours at 30ºC with shaking at 400 RPM. Luminescence was measured at 20 nm intervals between 400 nm and 600 nm everything 30 minutes. Absorbance at 600 nm was also measured everything 30 minutes.

2.8.2. Initial characterisation of the EL222 light responsive system

E. coli cells were transformed with EL222-PBLRep (Table 2.2). A single transformant colony was streaked onto a separate LB agar plate with kanamycin antibiotic selection. 10 mL of LB media and kanamycin at the appropriate amount (Table 2.1) was added to a 50 mL falcon tube and inoculated with a single colony from the streak plate. The overnight culture was incubated at 37ºC for 16 to 17 hours with shaking at 200 RPM in the presence of bright blue light. Cells were pelleted at 4ºC and 4,500 RPM for 10 minutes. Supernatant was removed and cells were resuspended in 10 mL of fresh LB media. The OD_{600} of resuspended cells was measured with a UV-Vis spectrophotometer and the culture diluted to an OD_{600} of 1.0 in LB media. To three wells of two black 96-well plates with flat, clear bottoms 89.5 µL of LB media was added along with 0.5 µL of working stock kanamycin (Table 2.1). 10 µL of *E. coli* cells were then added to each well. Clear lids were applied to each plate. One plate was wrapped in foil, whilst the other was place in the top of a pipette tip box. In the bottom of the tip box was a Raspberry Pi Zero (Pimoroni) with Unicron pHAT (Pimoroni). The Pi Zero was powered by an external battery pack. To all blue LEDs of the pHAT, 3 volts was supplied, resulting in bright blue light shining at the bottom of the 96 well plate. Both plates were incubated at 37ºC with shaking at 200 RPM for 5 hours. A Clariostar plate reader (BMG Labtech) was used to measure red fluorescence (570-15 excitation and 620-20 emission) and absorbance at 600 nm every 1 hour.

2.8.3. Sensitivity testing of the EL222 light responsive system

Cells were prepared as described in sub-section 2.7.3. To two wells of three 96-well plates, 89.5 µL of LB media was added along with 0.5 µL of working stock kanamycin (Table 2.1). 10 µL of *E. coli* cells were then added to each well. A clear, permeable seal (Breathe-Easy; Scientific Laboratory Supplies) was adhered to the top of each plate. One of the plates was wrapped in foil. The other two plates were placed on top of two separate optogenetic devices. Each optogenetic device consisted of an inverted black, 96-well plate with flat, clear bottoms, and RGB LEDs (Fedy Tech diffused 'Piranha' RGB; Adafruit 1451) placed such that the bulb was facing the clear bottom of the plate. A 3-volt CR2032 battery was used to supply 3 volts to the LED such that blue light was emitted. To one of the devices, 6 megohms of resistance was applied. To the other, no resistors were used. The 96-well plates were placed on top of a pipette tip box, such that the flat bottom was facing up. The 96-well plates containing cells were placed on top of the device, so that each well aligned. The plates were tapped to the top of the device and were wrapped in foil. All plates were incubated for 6 hours at 37^{o}C and 200 RPM, following which red fluorescence and absorbance were measured as in section 2.7.2. Images of the optogenetic devices can be seen in chapter 7.

2.8.4. Designing and modelling microfluidic chips

The microfluidic devices were designed using AutoCAD (Autodesk; 2022) and exported as DWG files. Ansys Workbench (Ansys; Student Edition; 2022 R2) was used to simulation fluid flow through the design. To save on computational resources, only on half of each design was simulated, as the designs consisted of two identical segments (Chapter 7). The DWG file was imported to ANSYS SpaceClaim (Ansys; Student Edition; 2022 R2) and converted into a 3D model. The 3D model was then imported into ANSYS Fluent (Ansys; Student Edition; 2022 R2) for fluidic simulation. The fluidic behaviours of each design variant were modelled using a 'Discrete Phase' model, where the liquid phase (i.e. the media the cells are suspended in) was modelled as water, and the discrete phase (i.e. the cells) was modelled as non-reactive spherical particles with a diameter of $5e^{-7}$ meters – approximately the same size as *E. coli* cells.

After initial exploratory simulations, a cell input flow rate of $1e^{-20}$ kg/second was used, along with a media flow rate of 0.0001 metres/second. Additionally, the following settings were used: (i) unsteady particle tracking, (ii) particles were tracked with fluid flow time step (iii) a maximum of 500 steps for particle tracking (iv) a length scale of

$1e^{-05}$ metres, (v) particles set to interact with fluid phase, (vi) simulations performed for 600-time steps, and (vii) time step size of 1 second. It should also be noted that cell growth and division has not been modelled in these simulations.

2.8.5. Fabrication of microfluidic chips

For fabrication, all design variants were positioned within a circle with a 7 inch diameter. The design was exported as a DWG (from Drawing) file and submitted to JD Photo Data (Hertfordshire; United Kingdom). JD Photo Data supplied a 7 inch chrome photomask prepared from soda lime glass, with a class 2 resolution (128k dpi) and negative, right-reading, chrome down polarity. The mask was submitted to INEX Microtechnology (Newcastle-Upon-Tyne; United Kingdom), from which an SU8 negative photoresist silicon wafer with uniform 40 µm depth features was produced.

The silicon wafer was first washed with isopropanol and acetone, and then silinised in a vacuum chamber for 30 minutes with 40 µL Tridecafluoro-1,1,2,2-tetrahydrooctyl-1-trichlorosilane. 60 mL of silicone resin (Translucent Platinum RTV-2; STARTSO WORLD) was prepared according to the manufacturer's instructions, and de-gassed under vacuum for 30 minutes. The silinised wafer was placed into a large glass petri dish lined with foil, and the silicone was carefully poured on top. The wafer and silicone were incubated at 60^{o}C for 3 hours to allow the silicone to set. Once set, the silicone was peeled away from the wafer, and scotch tape was applied to the side which was in contact with the wafer. A scalpel was used to cut a rectangle around each design, and all chips were stored until needed.

Immediately before use, a 1.2 mm biopsy punch was used to cut holes through the port sections of the microfluidic designs. A glass slide was cleaned thoroughly, and the microfluidic chip was placed onto the slide, ensuring the side which had been in contact with the wafer was face down. A plasma cleaner (PLASMAFLO PDC-FMG; Harrick Plasma) was used to plasma bond the chip to the glass slide. The glass slide with resin chip were placed into a plastic petri dish and inserted into the plasma cleaner. A vacuum was applied to the inside of the plasma chamber until a pressure of between 800 and 1,200 mtorr was achieved. The RF level of the plasma cleaner was set to high for 1 minute, before turning the RF level to off and slowly removing the vacuum from the chamber. The microfluidic chip was removed from the plasma cleaner and incubated at 40^{o}C for 20 minutes to help ensure strong bonding.

The top ends of10 µL pipette tips were cut down to size and inserted into the ports of the chips, such that the thinnest end was inside the chip. Clear HPLC tubing (1 mm) was inserted into the tops of the pipette tips. For tubing inserted into inlet ports, the top end of a 20 µL pipette tip was inserted into the other end of the tube. The thin end of the 20 µL tip was inserted into a 5 mL syringe filled with cells or media, depending on the experiment. For tubing inserted into the outlet ports, the end of the tube was taped to a waste collection tube. For all experiments, to achieve flow, syringes were placed into OEM pumps and controlled using WinPumpTerm (version 0.6 beta; NewEra).

For visualisation of microfluidic chip features, a Nikon Ti microscope with 40x lens under phase contrast was used. Images were taken using NIS Elements software.

2.8.6. Verification of cell flow

An overnight culture of *E. coli* DH5α cells was prepared as described in section 2.1.1. Cells were diluted 1 in 10 in fresh LB media and added to a 5 mL syringe. Cells were flowed through the chip as described in section 2.7.5 at various flow rates and imaged using a Nikon Ti2 microscope with 40x lens and 1.5x zoom under phase contrast, using the NIS elements software.

2.8.7. Verification of cell growth and fluorescence

An overnight culture of processor cells was prepared as described in section 2.1.1. Cells were diluted 1 in 10 in fresh LB supplemented with chloramphenicol and 10 µM C12-HSL. Cells were loaded into the chamber of a microfluidic chip with 100 µm long chamber and a shelf. Fresh LB media supplemented with chloramphenicol and 10 µM C12-HSL was flowed through the chip at a constant rate of 0.1 µL/min for 10 hours, with phase contrast and red fluorescence images taken every 30 minutes by a Nikon Ti2 microscope with 40x objective and 1.5x zoom. The microscope chamber was kept at 37^{o}C for the duration.

2.8.8. Verification of cell induction

An overnight culture of processor cells was prepared as described in section 2.1.1. Cells were diluted 1 in 10 in fresh LB supplemented with chloramphenicol only. Cells were loaded into the chamber of a microfluidic chip with 100 µm long chamber and a shelf. Fresh LB media supplemented with chloramphenicol and 10 µM C12-HSL was flowed through the chip at a constant rate of 0.1 µL/min for 10 hours, with phase

contrast and red fluorescence images taken every 30 minutes by a Nikon Ti2 microscope with 40x objective and 1.5x zoom. The microscope chamber was kept at 37ºC for the duration.

Chapter 3. Defining a Multi-Microbial Biosensor Framework using High-Level Modularity

As discussed in section 1.3.1, high level modularity has the potential to aid in the development of synthetic biological systems. In this chapter, the first section (3.1) clarifies what is meant by modular synthetic biology. Section 3.2 discusses a framework for modular synthetic biology systems, specifically focussing on genetic biosensors. To aid with implementation of the framework, section 3.3 describes a set of proposed best practices for a standardised representation of multi-microbial system designs, whilst section 3.4 presents a Python library developed to assist with building and testing modular synthetic biology systems.

3.1. Introduction

3.1.1. Modular synthetic biology

When talking about modularity in synthetic biology, a common term to come across is 'modular cloning'[59], [80], [188]. This term is often associated with modular Golden Gate assembly standards, where DNA 'parts' are viewed as standardised "discrete functional genetic elements"[189]. These genetic parts can be assembled into larger constructs using standardised cloning sites and methods, and the constructs built can be used in the development of novel synthetic biological systems and devices. This form of modular cloning, and indeed other types of DNA assembly which employ standard DNA parts such as BioBrick assembly, have been instrumental in allowing researchers to design and build DNA constructs required for the development of novel biological devices and systems[190], [191]. The use of standardised parts and assembly methods also allows for easier re-use of parts and constructs developed by other researchers. These standardised parts are a good example of low-level modularity, as described in chapter 1. In most Golden Gate based assembly standards, different 'levels' of assembly are defined. The individual genetic parts, e.g., single generic features, to be denoted as 'level 0' and act as low-level modules (Figure 3.1 (A)). The constructs assembled using these level 0 parts, usually a single genetic expression unit, are referred to as 'level 1' assemblies (Figure 3.1 (B)). The level 1 assemblies can be combined to form 'level 2' constructs (Figure 3.1 (C)). Once assembled, the constructs can be implemented to create a functioning system (Figure 3.1 (D)).

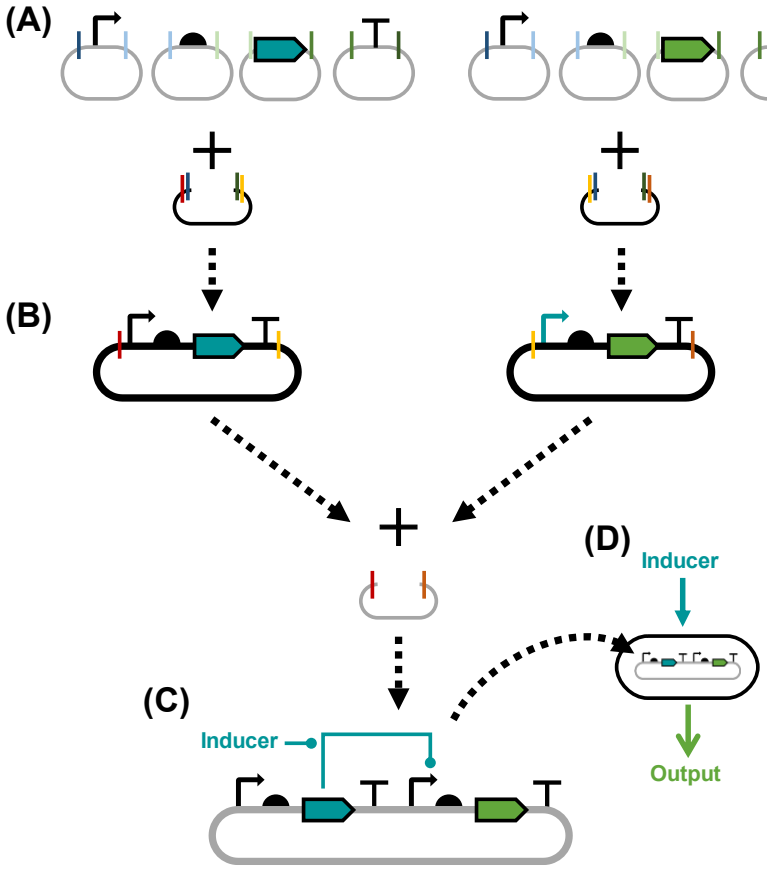


Figure 3.1. Modular Cloning in Synthetic Biology
Schematic overview of modular cloning using SBOL symbols. Compatible modular fusion sites depicted as coloured vertical lines. (A) Level 0 parts: individual genetic elements (promoters, ribosome binding sites, coding sequences, terminators) encoded on individual level 0 acceptor plasmids. Compatible restriction sites allow them to be assembled, in order, into a new level 1 assembly plasmid. (B) Level 1 assemblies: composed of the level 0 parts assembled together into a level 1 plasmid. Compatible restriction sites flanking the constructs allow them to be assembled into a level 2 assembly plasmid. (C) Level 2 assembly: composed of level 1 assemblies combined together to form a functional biological device, where the coding region from the first level 1 assembly interacts with the promoter in the second level 1 assembly. (D) The level 2 assembly can be expressed by a chassis to create a system with a higher-level function.

At first, it may seem that the level 1 and 2 constructs map well to the definition of high-level modularity discussed in chapter 1, in the same way that level 0 parts can be thought of as an example of low-level modularity. However, it is not always the case that higher order genetic constructs are high-level modules. This can be demonstrated with one the earliest examples of a synthetic biology design: the genetic toggle switch[104]. Figure 3.2(A) shows a schematic for assembly of an abstracted version of the original genetic toggle switch design using modular cloning. This design is comprised of two expression units (construct 1 and construct 2), each of which express a transcription factor which inhibits transcription from the promoter in the other expression unit.

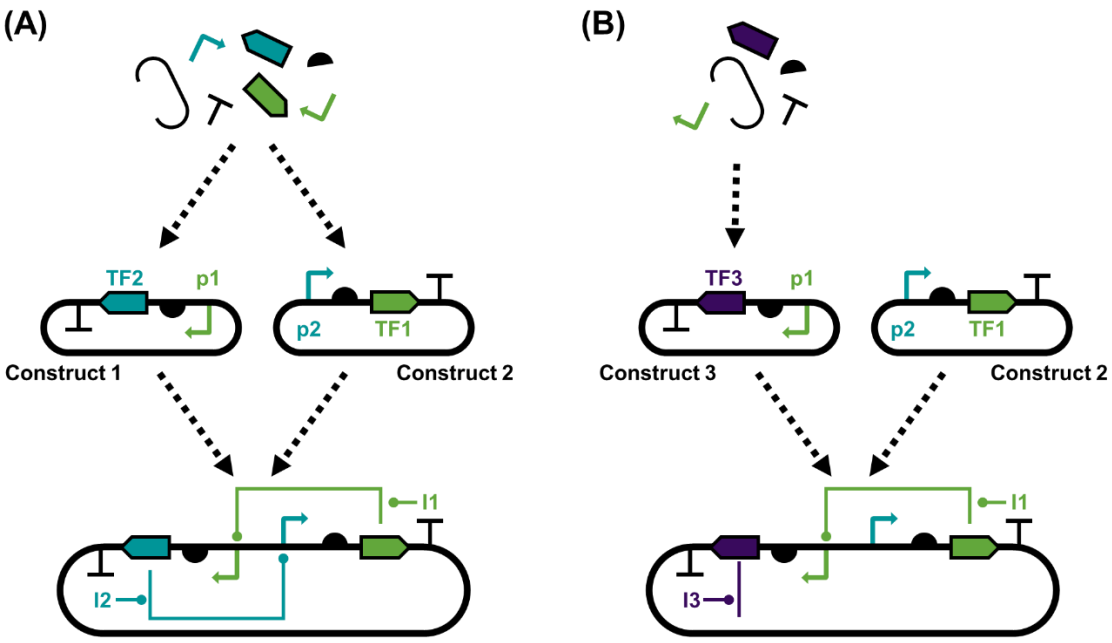


Figure 3.2. Modular Cloning of a Genetic Toggle Switch

(A) Schematic showing modular cloning for an abstracted toggle switch. The level 1 constructs (construct 1 and 2) are assembled using level 0 parts. The level 1 constructs are then combined into a level 2 assembly to form the toggle switch. In the presence of inducer 1 (I1), transcription factor 1 (TF1) is inhibited, which allows expression of transcription factor 2 (TF2) from promoter 1 (p1). Conversely, inducer 2 (I2) inhibits transcription factor 2 (TF2), which removes repression of promoter 2 (p2), which drives expression of transcription factor 1 (TF1). (B) Schematic showing the toggle switch from (A) where the level 1 assembly construct 1 has been replaced with construct 3. Construct 3 is a variant of construct 1, where TF2 is replaced with TF3. The level 2 construct assembled from construct 3 and 2 is no functional.

In the above example, it may seem that constructs 1 and 2 are acting as the high-level modules, which are assembled to form the final biological device (the genetic toggle switch). However, when considering the generalised high-level function and interchangeability of these constructs, the analogy fails, because functionality of each level 1 construct is tightly coupled. This functional dependency is illustrated in Figure 3.2 (B), where a variant of the genetic toggle switch has been designed. Here, transcription factor 2 (TF2) has been replaced with transcription factor 3 (TF3) to allow induction with inducer 3 (I3), instead of inducer 2. To do this, construct 1 has been replaced with construct 3. However, when this new construct is combined with construct 2, the genetic toggle switch no longer functions as intended, as construct 2 retains promoter 2 (p2), which is not affected by TF3. To regain functionality, construct 2 would require modification to replace p2 with a promoter repressible by TF3. The need for modification means each construct is not interchangeable, and hence does

not match the definition of high-level modularity. Additionally, it is not possible to assign high-level, abstract functionality due to the tightly coupled nature of the design[161].

Taking inspiration from other fields which have successfully utilised high-level modularisation, such as software engineering[192] and mechanics[193], it is possible to re-design the genetic toggle switch to allow for better exploitation of this concept. This approach relies on the use of standard connectors to link the modules together, and top-down design. By considering only the abstract logic employed by the genetic toggle switch, it is possible to describe the device's behaviour using two functional modules (Figure 3.3 (A)). The internal logic of these two modules consist of an inverter, an AND gate, and an OR gate. This internal logic is identical between the two functional modules, and the only differences are the inputs and outputs. Each of the modules takes its respective inducer (I2 or I1) as one of the inputs. The second input and the output have been replaced by connectors which are used to link the two modules together. At this abstract level, it is possible to see the intended function of the modules, and to predict how a system composed of these two modules would behave, such is the nature of top-down modular design. Moreover, a third module (Figure 3.3 (B)) which takes I3 as an input instead of I2 can be defined. By keeping the input and output connectors identical, module 2 and module 3 become fully interchangeable.

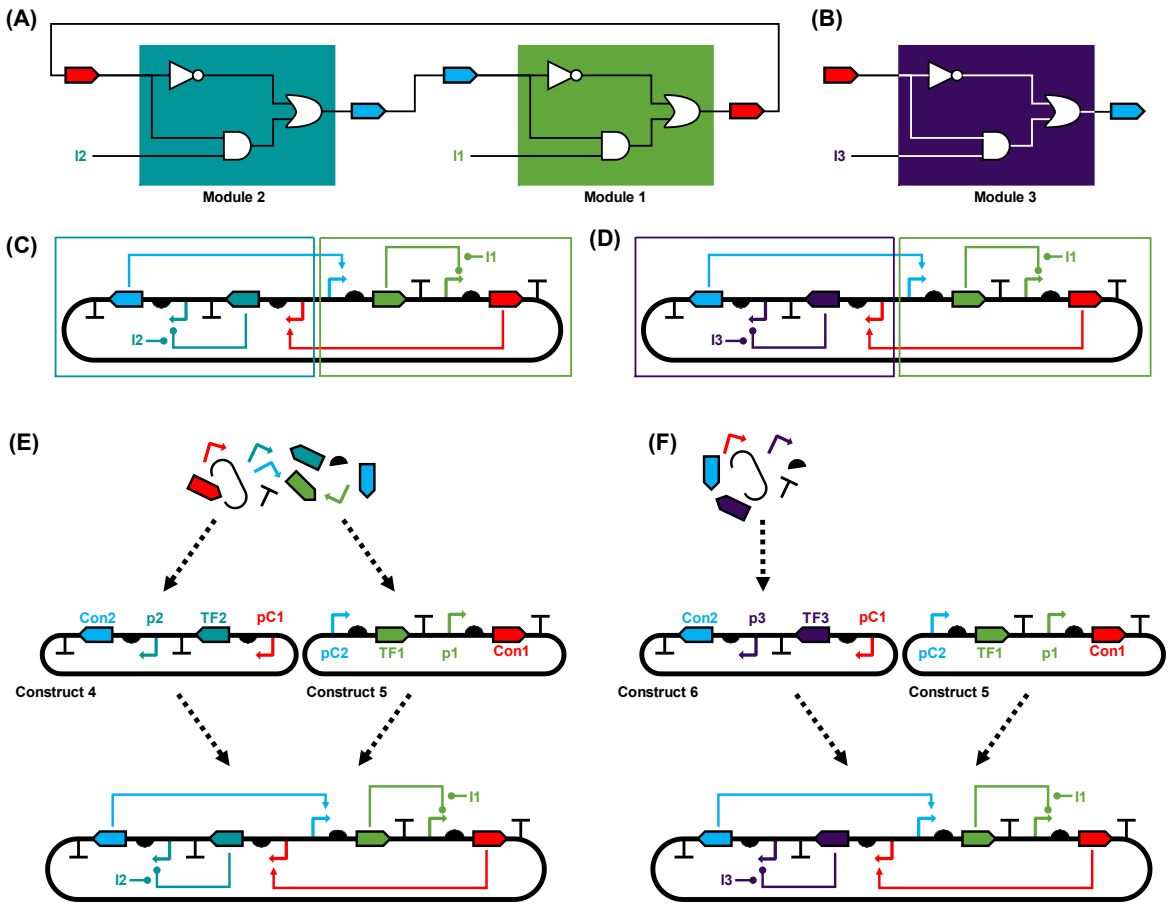


Figure 3.3. High-Level Modules for a Genetic Toggle Switch
Alternative method for modular assembly of a genetic toggle switch. (A) Two high-level modules which, when combined, show toggle switch functionality. Both modules use an inverter (▷○: switches an OFF signal to ON and vice versa), AND gate (D: only ON when both inputs are ON), and NOR gate (⌓: only ON when both inputs are OFF) to integrate signals from a connector (red and blue CDSs) and an inducer (I1 and I2). (B) A third high-level module which uses a third inducer (I3) as an input. The input and output connectors are the same as for the first module in (A). (C) Genetic implementation of the two modules in (A). (D) Genetic implementation of the module in (B) combined with the second module in (A). This is the same as the implementation in (C), except the first module has been swapped with the module in (B). (E) Modular cloning of the design in (C). the level 1 assemblies (construct 4 and 5) are built using level 0 parts. The two level 1 constructs are then combined into a level 2 assembly, which functions as a toggle switch. (F) The same as for (E), except construct 4 is replaced with construct 6. Construct 6 is based on the module design in (B).

A potential genetic implementation of modules 1 and 2 are shown in *Figure 3.3(C, E)*, and of modules 1 and 3 in *Figure 3.3(D, F),* further demonstrating the uncoupling and interchangeable nature of the modules. As a result, it would also be possible to experimentally characterise the behaviour of each module separately and use this data to better predict how a system composed of each module combination would behave, rather than having to test every combination (which can become unfeasible when many variants of a module exist). It should be noted that designing and implementing the

system in this way does come at the expense of a more complicated low-level design, but allows the high-level design to be simplified, and for better re-use of modules. There are also other issues with the design described. For example, module 3 is only interchangeable with module 2, and not module 1 (due to the connectors specified for the input/output of each module). These designs may also not be optimal for other applications, and the choice of connectors may interfere with the behaviour of other designs if those systems utilise the connectors in their internal mechanisms. Despite these issues, this concept of using high-level modules to design and implement biological devices and systems has the potential to aid synthetic biology in the same way it has other fields for the reasons discussed earlier. In the following section, some selected examples of previous implementation of such modularity are highlighted and discussed.

3.1.2. High level modularity in synthetic biology

Discussed here are some examples of where high-level modularity has previously been used to develop biological devices. These examples were highlighted as they focus on biosensors, or use biosensor-like mechanisms, which are a major focus for this thesis. Additionally, the examples were selected because the results and approaches highlighted crucial aspects, advantages, and limitations of high-level modularity in synthetic biology.

Potentially one of the earliest examples of true high-level modularity in synthetic biology is work presented by Tamsir and co-workers in 2011[194]. In this study, a series of genetic logic gates were designed and implemented in separate *E. coli* cells and 'wired' together using quorum sensing molecules. The modules were used to design higher order biocomputing functions, such as the XOR gate shown in Figure 3.4 (C). This XOR biocomputing system was implementing by spot plating the required cell types (containing logic gates with the required logic gate module) onto solid agar, spatially positioning the cell types such that they correspond with the desired logic circuit layout. These cells were spotted one layer at a time: Cell 1 was added and allowed to grow as the first layer, then Cell 2 and Cell 3 to form the second layer, and finally Cell 4 formed the third layer. These cell types appear to have only been characterised on solid media, rather than in liquid culture, and the requirement to add cells one layer at a time may present scalability issues as genetic circuits grow in complexity and size. Regardless, these modules allowed the authors to design and

implement many different biocomputing systems rapidly and, aside from the initial creation of the modules, any extra genetic engineering.

In 2019, a paper by Voyvodic and co-workers[195] described a method for expanding the 'detection space' of cell-free biosensors. This method made use of modular design, where a set of three modules were described: one which generated an output, one which could detect a common chemical using well known mechanisms, and a metabolic transducer module capable of converting a desired chemical, which perhaps does not have a known mechanism of detection, into a chemical which can be easily detected (*Figure 3.4 A*). These modules are described as 'plug-and-play', in that the modules can be assembled onto separate plasmids and mixed in cell-free systems to generate a complete biosensor. This study demonstrated the potential of high-level modularity in synthetic biology, although it should be noted that the plug-and-play nature of their output plasmid module is debatable, as it requires the output module to contain a promoter or other genetic element compatible with the sensor module. For example, in one of the use cases from the original paper, the authors design a sensor module which can detect benzoate (Figure 3.4 (B)). The sensor module works by expressing a transcription factor (BenR) which can bind benzoate. The output module contains a superfolder GFP coding sequence which is under the control of a promoter called P_{Ben}. This promoter is activated by the BenR-benzoate complex, which drives expression of the GFP. Whilst it would be possible to generate a library of output module consisting of different outputs controlled by the P_{Ben} promoter, it would not be possible to re-use these output modules with other sensor modules which do not express BenR. Nevertheless, this study presented a framework for expanding the range of chemicals which can be detected by biosensors, utilising high-level modularity which, for the most part, promotes re-use of parts with minimal modification required by other researchers.

A study by Wang *et al.* (2013) also aimed to implement high-level modularity using synthetic microbial consortia. In this work, a modular biosensor capable of detecting and responding to multiple external stimuli was designed and implemented using a series of AND gate modules, each of which was implemented in a separate cell. Signal propagation between the cell types was achieved using quorum sensing, as seen with Tamsir *et al.*'s approach. The multi-microbial biosensor developed and characterised by Wang *et al.* integrated three inputs to a single output (Figure 3.4 (D)). This biosensor

was composed of two cell types which both consist of an AND gate. The first cell type takes two stimuli as an input and produces a quorum sensing molecule as an output, and the second takes the quorum sensing molecule generated by the first cell type and the third stimulus as inputs. This second cell type was designed to produce a red fluorescence protein (RFP) as an output when both inputs were detected. This approach showed high-level modularity at two stages. The first was at the AND gate construction stage. As can be seen in *Figure 3.4 D*, both AND gates were assembled using three expression units: two which could detect the presence of the input stimuli and one which dealt with the output. The input modules were designed to express one of two proteins (HrpR or HrpS). Both proteins are required to activate expression of the promoter P_{hrpL}, which is what was included in the output expression unit to drive expression of the desired response. In this way, the construction of the AND gate modules can be considered truly modular. However, this approach is relatively limited in that only biosensors which rely on AND gate logic can be developed. It would also be difficult to easily tune the response characteristics of any biosensor by incorporating signal processing, such as signal amplification, as there is no space in the design described by the authors to add in this functionality. This is not to say that the work presented here is not useful, it should simply be noted that the approach taken is specific to a certain type of biosensor.

The fourth example of high-level modularity in synthetic biology is the study by Macia and co-workers in 2016[196], which focused on implementing spatially distributed biocomputation. Similar to the study by Tamsir *et al.* (2011), Macia and co-workers designed modular logic gates, which were implemented in yeast cells and focussed on NOT and ID gates with an inherent OR layer (Figure 3.4 (E)). This work also used multi-microbial modular logic gates to create high-order biocomputational logic circuits via phased mixing of each cell type, although the cells were combined in liquid culture using a custom built machine (Figure 3.4 (F)) rather than on agar plates. To note, the OR gate logic is not an actual module, but an inherent feature of how signals from modules in previous layers were integrated, and thus could potentially create limitations when developing systems as the OR gate cannot be swapped or easily tuned.

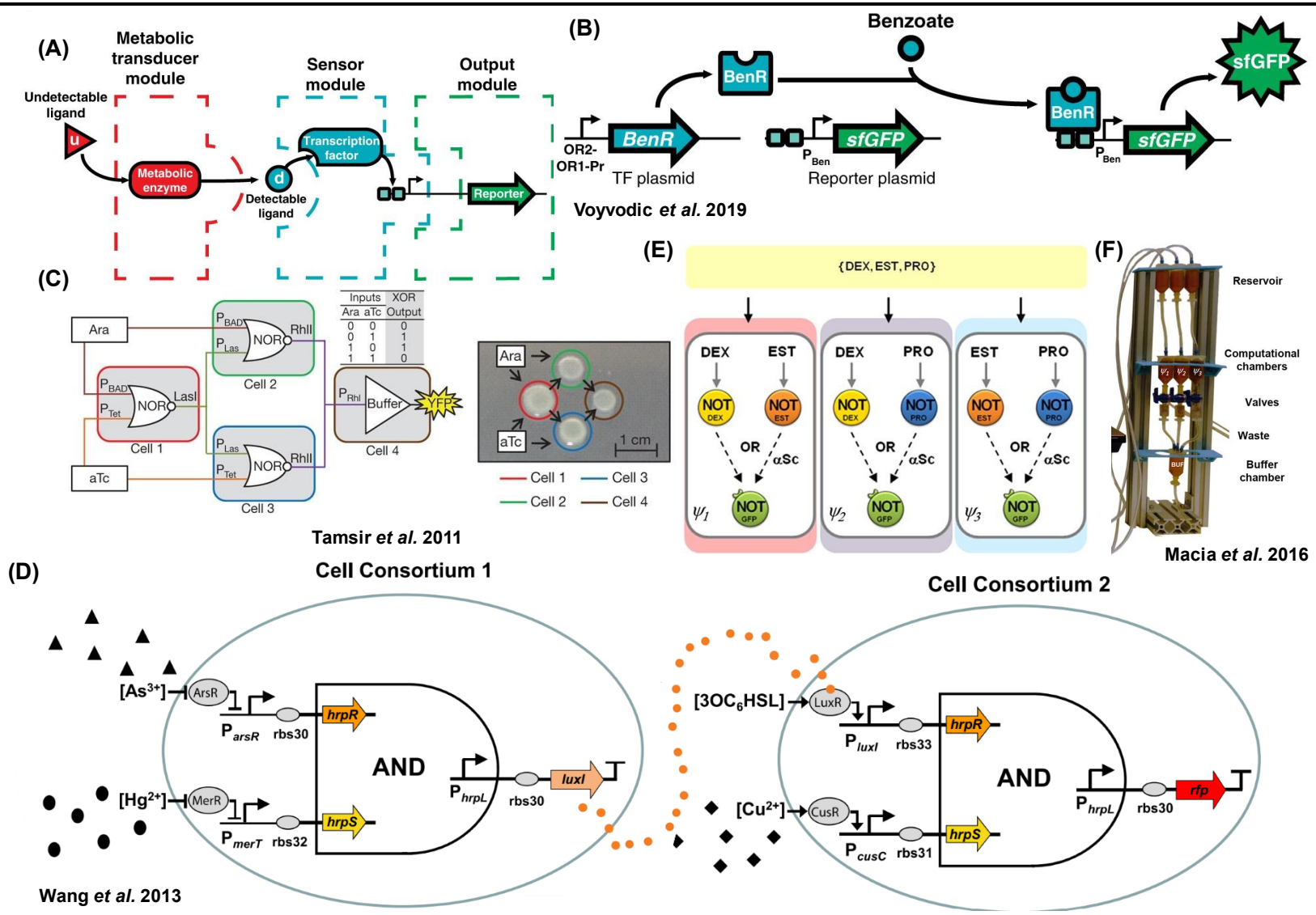

(A)
Metabolic transducer module
Sensor module
Output module
Undetectable ligand
u
Metabolic enzyme
d
Detectable ligand
Transcription factor
Reporter
(B)
Benzoate
BenR
sfGFP
OR2-OR1-Pr
TF plasmid
Reporter plasmid
Voyvodic et al. 2019
(C)
Ara
aTc
Cell 1
Cell 2
Cell 3
Cell 4
Buffer
YFP
Inputs Ara aTc
XOR Output
Tamsir et al. 2011
(E)
{DEX, EST, PRO}
DEX
EST
PRO
NOT
OR
αSc
GFP
(F)
Reservoir
Computational chambers
Valves
Waste
Buffer chamber
Macia et al. 2016
(D)
Cell Consortium 1
Cell Consortium 2
[As3+]
ArsR
hrpR
rbs30
AND
luxI
[Hg2+]
MerR
hrpS
rbs32
[3OC6HSL]
LuxR
rbs33
rbs31
[Cu2+]
CusR
rfp
Wang et al. 2013

Figure 3.4. Selected Examples of High-Level Modularity in Synthetic Biology

An overview of synthetic biology systems developed previously which make use of high-level modularity. (A) A modular framework for biosensors. Three types of modules are used: a metabolic transducer, a sensor, and an output. The metabolic transducer converts a desired molecule with no known sensing mechanism into a molecule which does have a sensing mechanism. The sensor module detects the molecule and passes a signal to the output module to generate a response. *Figure reproduced from Voyvodic et al. 2019 (figure 1) under licence agreement.* (B) Modular design for a benzoate biosensor using the framework in (A). Two modules are defined: a TF (transcription factor) plasmid which acts as the sensor module, and a reporter plasmid to act as the output module. The output module is activated in the presence of benzoate. *Figure reproduced from Voyvodic et al. 2019 (figure 2) under licence agreement.* (C) Implementation of an XOR gate using modular design. The modules are implemented into different cells, and the system is built by spotting cells onto an agar plate one layer at a time. The first layer consists of Cell1, the second layer of cell 2 and 3, and the third layer of cell 4. Signals from each cell type are passed via quorum sensing. *Figure reproduced from Tamsir et al. 2011 (Figure 3) under license agreement.* (D) Modular implementation of a three-input biosensor. Two modules are used. The first uses an AND gate to integrate signals from two stimuli. The second integrates the signal from the first module and a third stimuli, also using an AND gate. The second module produces a fluorescent reporter when activated. The two modules are implemented in separate cells, with the signal being passed from module 1 to module 2 via quorum sensing. *Figure reproduced from Wang et al. 2013 (figure 4) under license agreement.* (E) Design for a modular 'majority rule' biocomputing device, where a signal is generated when two or more of the inputs are present. Each module is implemented in a separate yeast cell. *Figure reproduced from Macia et al. 2016 (figure 3) under license agreement.* (F) Device built to implement the modular biocomputing device in (E). *Figure reproduced from Macia et al. 2016 (figure 3) under license agreement.*

3.1.3. Modularity and multi-microbial systems

Synthetic microbial consortia are commonly found in previous implementation of high-level bio-modularity, as seen in the examples above. In addition to the general advantages and applications of synthetic consortia discussed in *Chapter 1*, there are several reasons for why this might be. The first is that, at least conceptually, the creation of co-cultures to 'connect' biological high-level modules is relatively simple. The idea that cells containing different biomodules can be mixed together as seen in the Wang *et al.* 2013 study, or plated next to each other like Tamsir *et al.* in 2011, can seem analogous to some of the plug-and-play architecture seen in other engineering fields. The approach of implementing each module in a different cell type in a consortium also allows optimal organisms to be used for each aspect of a biological system, rather than relying on techniques such as codon optimisation to modify biological parts to operate in a sub-par host chassis.

Another reason why synthetic consortia are commonly used to implement high-level modularity in synthetic biology is simply because the inherent requirements of splitting a biological system across multiple cell types align closely with the requirements of high-level modularity. This is because the separate aspects of the system must be un-coupled to function independently within their host chassis, and communication must be established between the different modules. This design architecture has a lot of similarities with the modular toggle switch design shown in *Figure 3.3*, but less so with the original genetic toggle switch in *Figure 3.2*. It is therefore possible that some studies which implemented high-level modularity did so as a result of using synthetic consortia, rather than by intent. Regardless, the combination of synthetic multi-microbial consortia and high-level modularity in synthetic biology appear to have great potential and is the approach which was taken in this research.

3.2. Discussion of a high-level modular and multi-microbial framework for the development of genetic biosensor devices

The framework described here is based on work first described by the Newcastle iGEM 2017 team (2017.igem.org/Team:Newcastle), of which I was a member. The work in this thesis can be considered a direct continuation of that project.

In the previous section, it was identified that the combination of high-level modularity and multi-microbial systems could aid in the development of synthetic biology systems. In this section, it is discussed how the development of a framework could be used to explore this concept.

3.2.1. Framework requirements and application

In fields where high-level modularisation has already been implemented, the modules developed do not tend to be completely universal to everything within that area[197]–[199]. In software engineering, modular code tends to be intended for specific types of software programs[200], [201]. For example, modules used in video game development are not necessarily expected to function in or be used in code for, say, a word processor. Similarly, in computer hardware engineering, modules designed for a desktop computer may not be fully compatible with modules used in a laptop for reasons such as power requirements and size[202]. Therefore, when looking to implement high-level modularisation, one should recognise that it is unlikely that a module will be fully universal to every application within that field. Instead, it is important to consider which applications or areas within a field can benefit from the same set of modules. This contrasts some of the previous studies, such as the Tamsir *et al.* (2011) and Macia *et al*. (2016) studies discussed in *Section 3.1.1* (*Figure 3.4(B)* and *Figure 3.4(D)*) which tend to have scopes which are either much wider or narrower than those seen in other engineering fields.

In this thesis, the applicability of high-level modularity and multi-microbial systems towards to development of biological systems was explored. Specifically, the project aimed to determine whether designing and implementing systems in such a way could provide alternative, easily accessible avenues for optimisation, and promote the use of engineering principles such as standardisation and re-usability. To guide these investigations, a high-level modular and multi-microbial framework for developing synthetic biology systems was developed. Considering the points discussed above, it

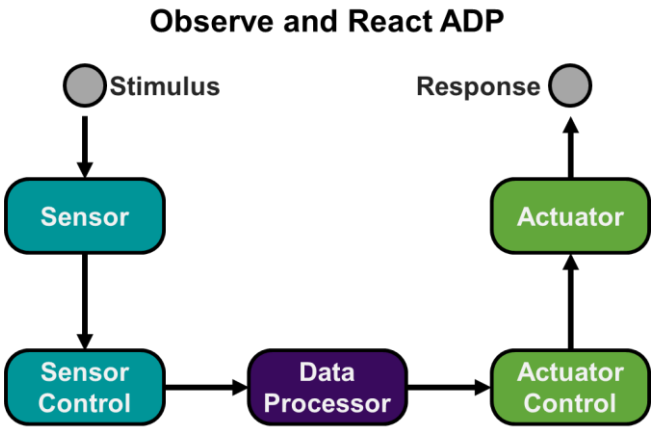


Figure 3.5. Observe and React Architectural Design Pattern
General overview of the observe and react ADP. The sensor detects a stimulus, which activates the sensor control. The sensor control informs the data processor, which processes the signal generated by the sensor control. The data processor then passes information to the actuator control, informing it as to the required response, which is then generated by the actuator.

was decided that this framework should focus on a specific area of synthetic biology, similar in scope to that seen in other fields which successfully apply high-level modularity.

There are many applications and areas of research which can be considered to fall under the general term of synthetic biology, as discussed in *Chapter 1*. One of these areas is the development of genetic biosensors, where sensing devices are built using DNA and implemented using biological chassis such as bacteria or cell-free protein synthesis systems. The general concept of genetic biosensors is ubiquitous in many synthetic biology systems, and there are a wide range of potential applications for these biological devices. As a result, it was decided that the framework's application would be towards the development of synthetic biology biosensors. An existing framework which matches this application, Sensynova, was previously proposed by the Newcastle iGEM 2017 team. As such, the framework described in this work is based on and extends the Sensynova framework.

3.2.2. General framework structure

Biosensors have many analogues in other engineering fields, such as electronic sensors in electrical engineering, and event handlers or conditional statements in software engineering and programming. In these examples, the overall functionality is to react to some external stimulus or event. In fact, this abstract behaviour is so common that it can be summarised by an often-used architectural design pattern. Architectural design patterns (ADPs) are defined as a generalised solution for a

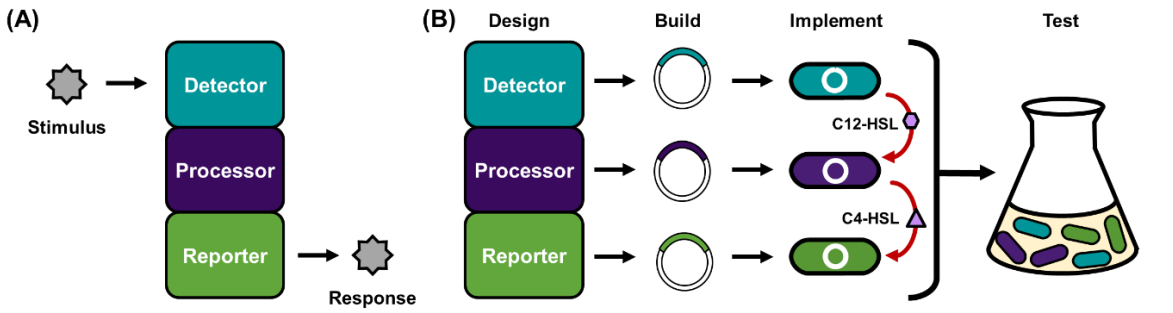


Figure 3.6. Sensynova Framework
General overview of the Sensynova framework. (A) Functional modularisation of a generic genetic biosensor, split into a detector, processor, and reporter module. This pattern mirrors the ADP shown in Figure 3.5. (B) Abstract depiction of the Sensynova framework. Biosensors are designed modularly, with each design built on a separate plasmid. Each plasmid is transformed into individual cells. Quorum sensing mechanisms built into each module's design enables uni-directional communication when in co-culture, at which point the biosensor can be tested.

common problem within a specified context, and have found popularity in software engineering and, to a lesser degree, other engineering fields and architecture. The observe and react ADP defined in software engineering can also be used to summarise the general structure of biosensors and their analogues[203] (Figure 3.5). Broadly speaking, the observe and react pattern is used when a sensor (or set of sensors) needs to be monitored and a response initiated depending on the output of the sensors. The ADP is split into three sub-processes: sensor control, data processor, and actuator control. The sensor control is responsible for collecting information from the sensor(s). The information collected is then fed to the data processor, which can analyse or modify the information in some pre-determined way. This data is then sent to the actuator control, which will generate a response depending on the information. This response could be to simply display the data, or it could generate some type of change by sending the data to some other device.

Following the observe and react ADP, synthetic biology biosensor designs can be split into three modules: a detector which reacts to the presence or absence of a stimulus, a processor which can employ some sort of logic such as amplification or tuning to a specific level, and a reporter which generates some sort of response (Figure 3.6 (A)). This follows the modules described by the original Sensynova framework, where it was shown that all biosensors designed or used by iGEM teams matches this architecture. Following on from the concept of implementing modules in different cell types which can be co-cultured to form the overall system, the detector, processor, and reporter modules described here can be implemented similarly. Based on the previous

successes within synthetic biology discussed in Chapter 1, communication between cell types expressing one of the three modules can be achieved via acyl-homoserine lactone (AHL) based quorum sensing. This communication was intended to be uni-directional, with the signal generated by the detector cells (as the result of interactions with a stimulus) passing to the processor cells, which then modifies the signal and passes it to the reporter cells, which can then generate the response. Specifically, the LasIR and RhlIR mechanisms were used to facilitate communication from detector to processor cells, and from processor to reporter cells respectively. These mechanisms were chosen as they had been previously reported to have little cross-talk[151]. Broadly, the detector module should confer C12-HSL production functionality, and the processor module should respond to C12-HSL. The processor module should also produce C4-HSL, and the reporter module should respond to C4-HSL (Figure 3.6 (B)). By defining this method of communication within the Sensynova framework, it is possible to help ensure module compatibility, where any detector, processor, or reporter cells can be combined with any other combination of detector, processor, or reporter cells.

The general framework described above is expanded on in chapter 4, where a proof-of-concept biosensor is designed and built according to the Sensynova principles. By developing a high-level modular and multi-microbial biosensor, it was possible to validate whether biosensors designed, built, and implemented according to the framework were functional. Additionally, novel approaches towards optimisation could be investigated, which may provide opportunities to ease the development process of biosensors more generally. First, however, presented in the remainder of this chapter are outcomes from research into, and development of, resources for aiding implementation of modular and multi-microbial systems. Focus was placed on how these resources could be used to promote the use of engineering principles such standardisation, reproducibility, and re-usability within the Sensynova framework.

3.3. Defining Best Practices for Standardised Representations of Multi-Microbial Systems

*This work has been published as "Capturing Multicellular System Designs Using Synthetic Biology Open Language (SBOL)" in ACS Synthetic Biology (doi.org/10.1021/acssynbio.0c00176), and also exists as a preprint on bioRxiv (*doi.org/10.1101/463844*).*

The best practices described here were submitted and accepted as SBOL Enhancement Proposal 30 (SEP 030: github.com/SynBioDex/SEPs/blob/master/sep_030.md*). This proposal was modified and included in the release of SBOL version 3.0.1 (*doi.org/10.1515/jib-2020-0017*).*

3.3.1. Overview and rationale

One of the main aims for the Sensynova framework was to aid in the development of synthetic biology biosensor designs by promoting the sharing and re-use of modules and systems. It was important, therefore, to ensure that information relating to these multi-microbial designs could be captured in a standard format that was easily shareable and allows for modules designed by different researchers to be combined into a single biosensor design. The Synthetic Biology Open Language (SBOL) is a data standard which can be used to store and share information about biological designs, along within information about how these designs have been implemented and characterised[64]. Prior to the work presented here, the SBOL data standard had only been used to capture design information about biological parts and devices, and ignored how contextual information, such as the chassis that the design is intended to be implemented in, should be represented. In addition, the SBOL standard had not been used to capture information about multi-microbial systems.

To address the limitations of SBOL stated above, a set of standard best practices were developed to aid in the design and sharing of multi-microbial designs. These best practices considered the minimal information required for others to understand, adapt, and build a multi-microbial design. Other aspects, such as machine-readability, flexibility, and intuitiveness, were also explored. The best practices described here largely fit within the SBOL version 2 specification and are therefore compatible with tools which use any release of SBOL version 2.

3.3.2. Ontologies in the SBOL data model

The SBOL data model makes use of ontologies to help describe biological entities, interactions, and functions in a standard way. An ontology can be thought of as a set

of formal descriptions for specific terms and their relationships[204]. A number of ontologies are used in SBOL to better describe entities within a system, and how those entities interact. The SBOL-OWL ontology defines relationships between classes in the SBOL data model and terms from other ontologies, allowing for recommendations of terms to be used for different aspects of a design or system[205]. For physical biological entities, terms taken from the Sequence Ontology (SO) are often used[206]. Example SO terms are 'Promoter' (SO:0000167), 'Ribosome Entry Site' (SO:0000139), and 'CDS' (SO:0000316). For interactions between entities, terms like 'Genetic Production' (SBO:0000589) from the Systems Biology Ontology (SBO) can be used to describe the role of that reaction[207]. The SBO can also be used to define functions for participants in the interaction, using terms like 'Template' (SBO:0000645) and 'Product' (SBO:0000011). Other commonly used ontologies include the Gene Ontology (GO)[208], Chemical Entities of Biological Interest (CHEBI)[209], and BioPAX[210].

3.3.3. The SBOL 2 Data Model

Explained here are relevant parts of the SBOL version 2.3.0 specification, which was the SBOL release these best practices were based on.

Two main classes are used in SBOL version 2 for representing biological designs: *ComponentDefinition* and *ModuleDefinition*. The *ComponentDefinition* class can store information about physical structures, such as DNA and proteins. Aspects of a design represented by a *ComponentDefinition* may have both a 'role' and a 'type' associated with them. The 'type' property in SBOL is used to describe the category to which a biological entity belongs and can use BioPAX terms like 'DNA' (BioPAX: 0654) or 'Protein' (BioPAX: 1208). The 'role' property is used to convey the intended or expected function of an entity, using terms like 'Promoter' (SO:0000167) or 'Transcription Factor' (GO:0003700).

The *ModuleDefinition* class is used to group together biological entities in a design, allowing for definition of functional interactions between such entities. Designs captured by a *ModuleDefinition* can range in complexity, from individual biological entities such as promoters, coding sequences, and proteins, to devices composed of multiple parts or complex systems comprising many devices, like a genetic biosensor. Unlike the *ComponentDefinition* class, instances of *ModuleDefinition* do not have a 'type' property and rely only a 'role' to represent overall functionality. For example, a

metabolic pathway might have roles of ‘metabolic process’ and ‘small molecule biosynthetic process’ from the Gene Ontology (GO), and a biosensor could have a role of ‘response to chemical’, also from the GO. In the case of devices and systems, each of the individual parts are described by separate *ComponentDefinition* class instances. The intended use and function of the part can be described within a *ModuleDefinition* class, where the part is instantiated using a *FunctionalComponent* class annotated with its intended ‘role’.

The *ModuleDefinition* class can contain interactions between biological entities in a design. One such example of an interaction could be a protein which binds to and represses a promoter. Interactions are formally captured with the use of *Interaction* and *Participation* instances. The *Interaction* class specifies the interaction type, such as genetic repression, within which instances of the *Participation* class specify interacting entities and the role played by those entities.

3.3.4. Discussion of essential information to be captured

To define a set of standard best practices for capturing designs of multi-microbial systems, it was important to consider the information needed to be captured. Essential information are attributes of a multi-microbial system which must be recorded to ensure others can understand the design’s purpose and function. Optional information about a design may provide further context but is not essential for understanding and implementation.

To determine the essential information to be captured, core components of a multi-microbial system were considered, using a basic design as an example (Figure 3.7). For this design, two populations composed the system: *E. coli* DH5α and *Bacillus subtilis* 168, which were present in proportions of 20% and 80% respectively. Information about the populations represented the first core component of the system. Thus, the SBOL standard was required to capture information the number, proportions, and taxonomic information about each population. Capturing information regarding the taxonomy of each population is required as behaviour differs between species and strains. For example, the two commonly used *Escherichia coli* strains DH5α and BL21 have distinct differences. DH5α is a cloning strain and deficient in nucleases[211], whereas BL21 is protease deficient and hence will have a lower turnover of proteins[212]. In the example system, the *E. coli* DH5α strain was modified to contain a plasmid.

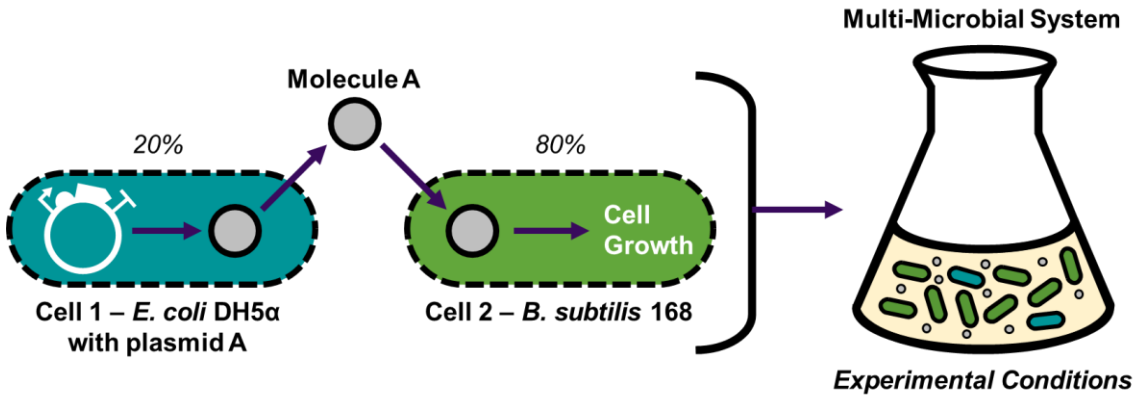


Figure 3.7. Core Components of a Multi-Microbial System
In this diagram, a design for a synthetic multi-microbial system is shown to demonstrate the core components of a multi-microbial system. This system is composed of two cell populations: cell 1 and cell 2. The cell 1 population is composed of E. coli DH5α cells transformed with plasmid A. The cell 2 population is composed of B. subtilis 168 cells. In the initial state of the system, 20% of the cells are part of the cell 1 population, and 80% are part of the cell 2 population. Plasmid A in the cell 1 population leads to the production of molecule A, which is then used by the cell 2 population to promote their own cell growth. The multi-microbial system is then implemented in a conical flask with a set of unspecified experimental conditions.

Modifications such as these, including genome editing, should be recorded, along with the functionality they are intended to impart. For example, here, plasmid A was designed to allow *E. coli* cells to produce molecule A.

Aside from information regarding each population, inter-population interactions were also identified a key feature of multi-microbial systems, as these interactions tend to guide overall functionality[213]–[215]. In the example system, molecule A produced by the *E. coli* cells enhances *B. subtilis* growth. Thus, a uni-directional interaction from cell 1 to cell 2 in the form of growth promotion existed.

In addition to the information discussed above, there were other important aspects of the design, namely information related to implementation of the system. Details such as media used to grow the cells, initial growth phase of the cell populations, culture volumes, and incubation conditions can all impact the functionality of the system, and hence should also be recorded if known[216], [217]. It should be noted that whilst the information discussed here was deemed ‘essential’, in certain cases it may not be known. It was therefore important that when multi-microbial designs are captured, it is possible to indicate when information is unknown, rather than simply missing.

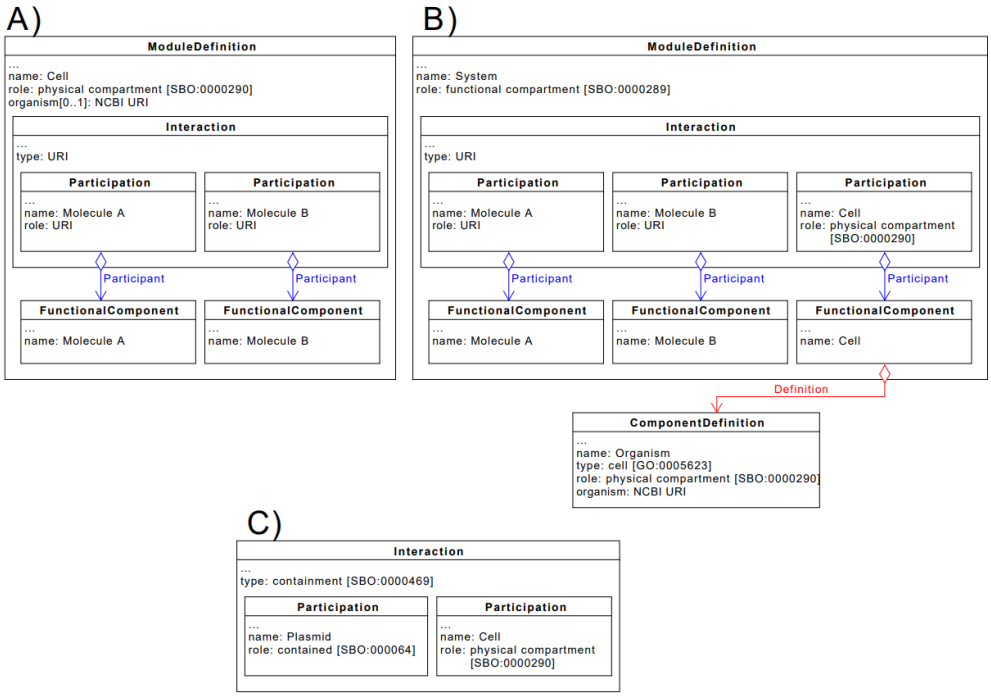


Figure 3.8. Potential Best Practices for Representing Cells using SBOL

Unified Modelling Language (UML) diagrams depicting two different approaches for representing cells in SBOL. **(A)** First approach for capturing information about cells, referred to as 'cell representation A'. A *ModuleDefinition* instance represents the cell, and taxonomic information is annotated on the *ModuleDefinition*. Interactions are defined with the *Interaction* class, and internal biological entities captured using the *FunctionalComponent* class. **(B)** Second approach, referred to as 'cell representation B'. The cell is represented as a *ComponentDefinition*, which is annotated with the cell's taxonomy. A *ModuleDefinition* is used to represent the cell system, in which the cell itself is included as a physical compartment. **(C)** Example of how the *Interaction* class could be used to explicitly capture that an entity is contained within a cell for cell representation B, rather than implicitly as in (B).

For all approaches described in this section, the essential information described above can be captured in SBOL – either explicitly or implicitly. Biological entities such as proteins and small molecules, and implementing/characterising designs, are not discussed here[218]. Instead, only aspects of multi-microbial systems discussed in this section which are not covered by the SBOL 2.3.0 specification are considered.

3.3.5. Approaches for representing cells using SBOL

The SBOL data model was not able to natively capture information about cells or other chassis, which as identified above is crucial for describing multi-microbial systems. Thus, this limitation was tackles first. Two potential approaches were detailed which could record the essential information about cells identified in the previous sub-section. The first approach is referred to as 'cell representation A', and the second as 'cell representation B'.

To illustrate both approaches, information about a cell with an interaction involving two molecules was captured (Figure 3.8). For both approaches, taxonomic information about the cells was captured through the use of an URI (Uniform Resource Identifier) for a relevant entry in the NCBI Taxonomy Database. This standardised approach allowed for easier automated retrieval of information about different organisms. In cell representation A, the cell was represented as an instance of the *ModuleDefinition* class (Figure 3.8 (A)) with a role of 'physical compartment' from the Synthetic Biology Ontology (SBO:0000290), which is defined as a "[s]pecific location of space, that can be bounded or not". To confer that the physical compartment was a cell, an 'organism' property was annotated to the *ModuleDefinition*. It should be noted that whilst the SBOL version 2.3.0 specification allowed for user-defined annotations, the 'organism' property was not defined within the specification and therefore was not standard within that version of SBOL. Taxonomic information was stored as attributes of the ModuleDefinition, and intracellular interactions were defined using *Interaction* class instance. Biological entities within the cell were represented using *FunctionalComponent* instances.

For cell representation B, an instance of the *ComponentDefinition* class represented the cell, which was annotated with information regarding taxonomy (Figure 3.8 (B)). Similar to cell representation A, the *ComponentDefinition* role was annotated as a 'physical compartment', however a type was also given of cell' from the Gene Ontology (GO:0005623) [ii]. A *ModuleDefinition* instance was used to define cell functionality, where the *ModuleDefinition* represented the cell system, and the cell itself was included as a *FunctionalComponent*. For the cells system, a role of 'functional compartment' (SBO:0000289) was used, which is defined as a "[l]ogical … subset of the event space". This essentially conveys that the cell system represents a subset of an overall system where interactions involving a cell occur. Intracellular interactions were captured similarly to cell representation A, except the cell was also involved explicitly in each interaction as a physical compartment. For cell representation B, it was possible to explicitly capture which entities resided within the cell (Figure 3.8 (C)), rather than relying on implicit representation as in Figure 3.8 (B).

[ii] It should be noted that the GO term for 'cell' (GO: 0005623) has now been made obsolete in favour of the Cell Ontology (CL) term for 'cell' (CL:0000000). However, to keep in line with the published work and accepted best practices, the GO term is used throughout this thesis.

One advantage of cell representation A is that it is conceptually simpler to understand, and the data model is smaller. However, the advantage of cell representation B is that by capturing taxonomic information in a separate *ComponentDefinition* away from the class instance representing the system, it is easier to capture how a cell has been modified. For example, with cell representation B, it is clear that Plasmid A has been added to the *E. coli* DH5α strain, and was not already present within that strain, as might be implied by approach 1. Additionally, by capturing the cell using the *ComponentDefinition* class, it is possible to instantiate the cell as a *FunctionalComponent* within the *ModuleDefinition*, and hence use it as a participant in interactions. Using this approach allows extra context to be added to specified interactions, such as the containment interaction described above to demonstrate that a plasmid is contained within the cell.

3.3.6. Approaches for representing multi-microbial systems using SBOL

In this sub-section, options for representing in SBOL multi-microbial system designs, which incorporate two or more cell types, are discussed. There are three main approaches illustrated in Figure 3.9, Figure 3.10, Figure 3.11 which capture the information in this design, with the exception of cell ratios and experimental conditions, which are addressed later.

The first approach is referred to as ‘multi-microbial representation A’. Here, the multi-microbial system is represented by a *ModuleDefinition* with a role of ‘functional compartment’ (SBO:0000289). The cell types involved in this system are captured using instances of the *Module* class, with a definition property which refers directly to the *ModuleDefinition* instance which captures information about that cell type. Figure 3.9 (A) and (B) show how this approach is compatible with both ‘cell representation A’ and ‘cell representation B’ respectively. In this approach, intercellular interactions are determined implicitly through shared interactions with identical pools of molecules or other entities. In this example, both cell 1 and cell 2 interact with molecule A; cell 1 produces the molecule and cell 2 uses it to enhance cell growth. Therefore, when both cell types and instantiated within the same multi-microbial system, it can be deduced that they exhibit an intercellular interaction via molecule A.

‘Multi-microbial representation B’ (Figure 3.10) was similar to multi-microbial representation 1 in that a *ModuleDefinition* instance with a role of ‘functional compartment’ represented the multi-microbial system. Further, *Module* instances captured cell types in the design. However, this approach required explicit definition of intercellular interactions, rather than implicit, allowing designers to highlight intended intercellular interactions within the system. This came at the cost of additional data model complexity. Multi-microbial representation B was not compatible with ‘cell representation A’. This was because the cell type must be defined as a participant in intercellular interactions. As only instances of the *FunctionalComponent* classes can be used as participants in an interaction, and as *FunctionalComponents* must refer to a *ComponentDefinition*, as ‘cell representation A’ only used *ModuleDefinition* instances to refer to a cell, it could not be used with this approach.

As with the previous two approaches, ‘multi-microbial representation C’ also uses a *ModuleDefinition* class instance with a role of ‘functional compartment’ to represent a multi-microbial system (Figure 3.11). This approach also uses both *Module* and *FunctionalComponent* classes to represent cells, except in the case when cell representation 1 is used to capture information about cells where only the *Module* class is used. Any non-cell entities are also instantiated using *FunctionalComponent* classes. The *Module* instances are defined by the *ModuleDefinition* classes used to represent cell or cell system as in multi-microbial representation 1 and 2. However, in this approach, the *Module* class instances also contain instances of the *MapsTo* class, which are used to explicitly capture links between the entities present in multiple parts of the same design. Here, a *MapsTo* class with a ’refinement’ value of ’merge’ is used to link *FunctionalComponent* classes which represent an entity in the multi-microbial system to the *FunctionalComponent* class used to represent the same cell in the lower-level cell system design.

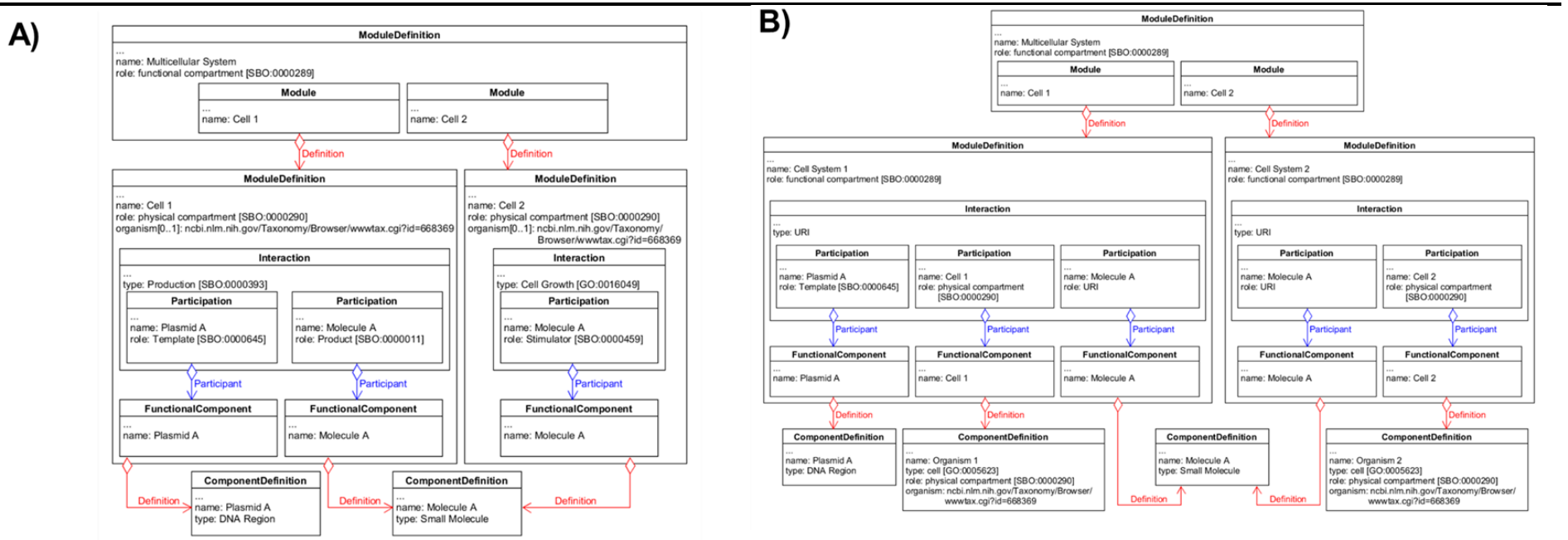


Figure 3.9. Potential Best Practice for Representing Multi-Microbial Systems using SBOL – Multi-Microbial Representation A

Both (A) and (B) are Unified Modelling Language (UML) diagrams depicting an approach for capturing information about multi-microbial systems using SBOL. The diagrams represent the same system shown in *Figure 3.7*. (A) and (B) show the same approach for capturing information about multi-microbial systems, however (A) demonstrates how this approach is compatible with cell representation A, and (B) shows compatibility with cell representation B. In this approach, the multi-microbial system is represented using an instance of the *ModuleDefinition* class, which has a role of 'functional compartment' (SBO:0000289). The cell types within this system are represented by *Module* instances, which are defined by the *ModuleDefinition* which captures information about that cell type. Intercellular interactions are captured implicitly by comparing interactions defined by the lower-level cell *ModuleDefinition* instances. Here, Cell 1 and Cell 2 interacts via Molecule A, where cell 1 produces Molecule A and cell 2 uses Molecule A to stimulate cell growth.

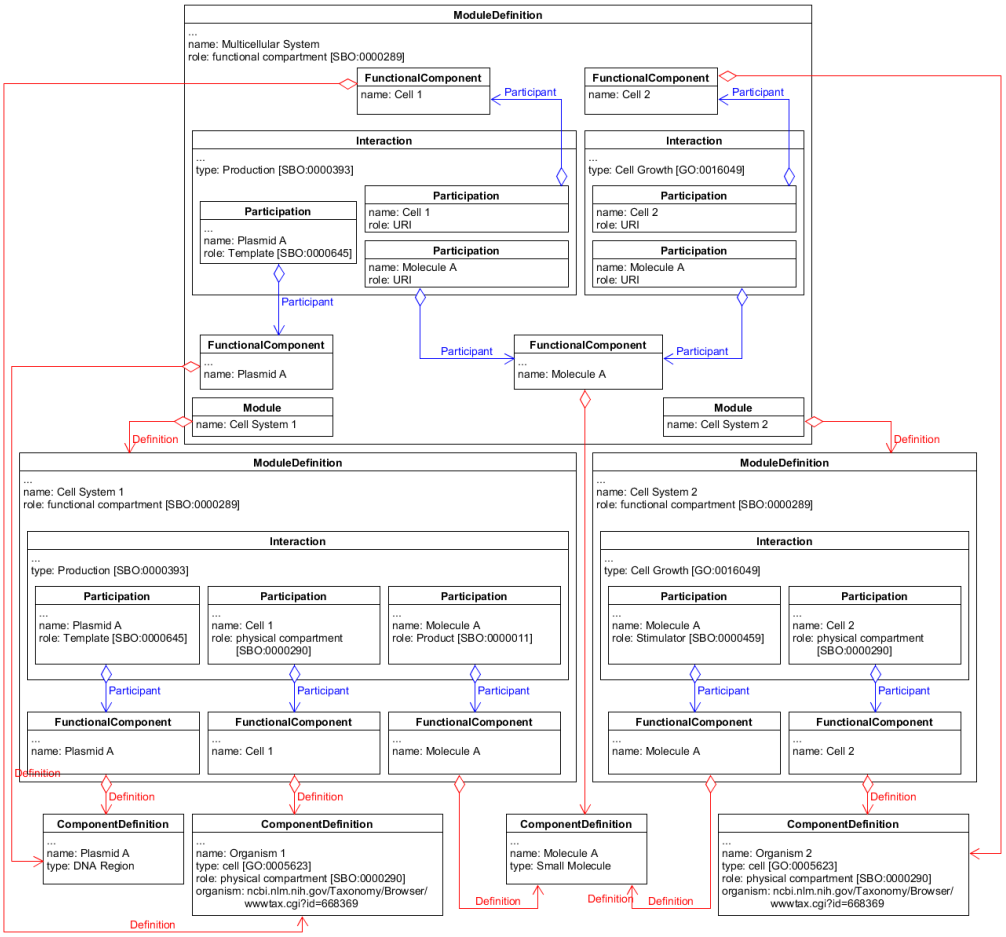


Figure 3.10. Potential Best Practice for Representing Multi-Microbial Systems using SBOL – Multi-Microbial Representation B

Unified Modelling Language (UML) diagram depicting an approach for capturing information about multi-microbial systems in SBOL. The diagram represents the same system shown in *Figure 3.7*. In this approach, the multi-microbial system is represented using an instance of the *ModuleDefinition* class, which has a role of 'functional compartment' (SBO:0000289). The cell types within this system are represented by *Module* instances, which are defined by the *ModuleDefinition* which captures information about that cell type. The cell type is also represented using a *FunctionalComponent* instance, which is defined by the *ComponentDefinition* which captures taxonomic information about that cell type. Other non-cell entities are captured only through the use of the *FunctionalComponent* class. Intercellular interactions are captured explicitly through the use of *Interaction* class instances, where the cell type related to that interaction are included as participants with a role of 'physical compartment' (SBO:0000290). Here, Cell 1 and Cell 2 interact via Molecule A, where cell 1 produces Molecule A and cell 2 uses Molecule A to stimulate cell growth.

One of the major differences between each approach presented here are how intercellular interaction are defined. Although explicit definition of interactions removes ambiguity, implicit interactions, as well requiring a similar data model, can have benefits. With implicit interactions, automated design software could combine cell designs into a multi-microbial system and automatically determine interactions

between cells. However, if designs for cells are obtained from other researchers, perhaps via a database such as SynBioHub, processes which are not important in a homogeneous design but are crucial in a multi-microbial system may be missing, and hence important interactions lost.

Aside from intercellular interactions, it was identified that proportions of different populations composing the multi-microbial system should be captured. For the example system in *Figure 3.7*, cell type 1 composed 30% of the system, whilst cell type 2 composed 70%. For all three multi-microbial representation approaches, it was possible to capture this information using an instance of the *Measure* class (Figure 3.12). The measure class could be used to annotate *Module* instances which represent cells or cell systems. The *Measure* class allows annotation of a numerical value, which here would represent the proportion of cells, and a unit of measurement. For the purposes of capturing cell proportions, it was recommended that the OM (Ontology of Units of Measure) term percentage is used. The *Measure* instance could also be annotated with a ‘type’ property using the SBO term ‘fraction of an entity pool’ (SBO:0000470), where the entity pool being referred to here is the cells.

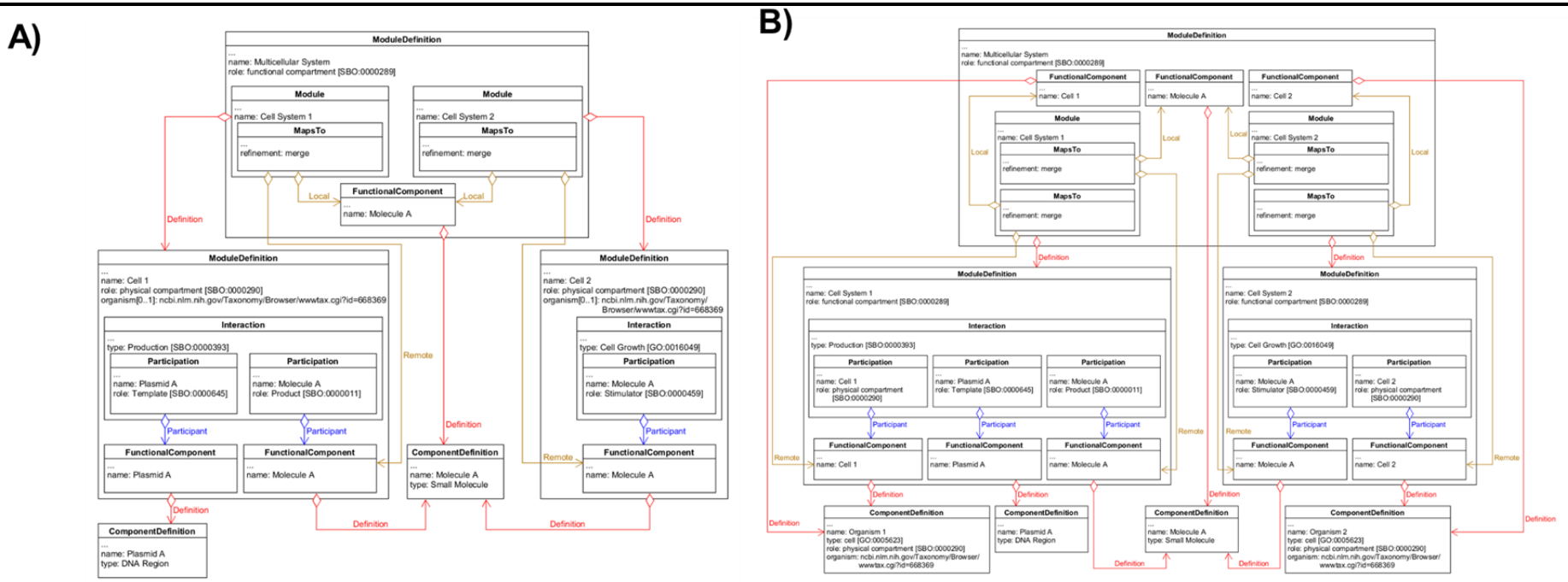


Figure 3.11. Potential Best Practice for Representing Multi-Microbial Systems using SBOL – Multi-Microbial Representation C

Unified Modelling Language (UML) diagrams depicting an approach for capturing information about multi-microbial systems using SBOL. The diagrams represent the same system shown in *Figure 3.7*. **(A)** demonstrates how this approach is compatible with cell representation A, and **(B)** shows compatibility with cell representation B. The multi-microbial system is represented using an instance of the *ModuleDefinition* class with a role of 'functional compartment' (SBO:0000289). Cell types within the system are represented by *Module* instances, which are defined by the *ModuleDefinition* representing information about that cell type. In (B), cells are also represented using a *FunctionalComponent* instance, which is defined by the *ComponentDefinition* which captures taxonomic information about that cell type. The *MapsTo* class is used to explicitly capture which entities are present in both the cells and the multi-microbial systems. This information can be used to determine intercellular interactions. Here, Cell 1 and Cell 2 interact via Molecule A, where cell 1 produces Molecule A and cell 2 uses Molecule A to stimulate cell growth.

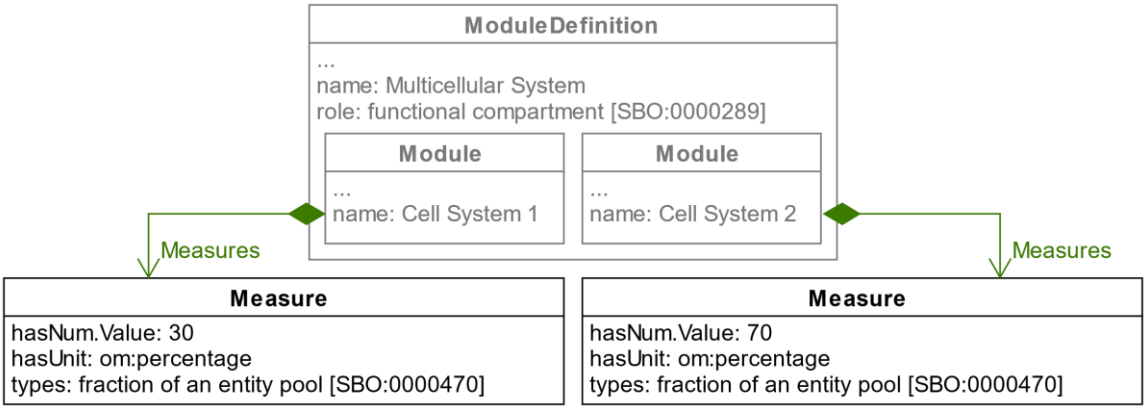


Figure 3.12. Capturing Cell Type Proportions in Multi-Microbial Systems using SBOL

This figure shows an Unified Modelling Language (UML) diagram depicting how to capture information about the proportions of each cell type present in a multi-microbial system. In this figure, information about the individual cells has been omitted for clarity, and only the class instance representing the multi-microbial system is shown. This multi-microbial system is composed of two cell types (cell 1 and cell 2). The cell 1 population composes 30% of all cells in the system, and the cell 2 population composes the other 70%. This information is conveyed through the use of the *Measure* class.

3.3.7. Accepted Best Practices

In this section, the approved best practices for capturing information about cells and multi-microbial systems is presented. These best practices were selected based on discussions with the SBOL community.

Representing cells using SBOL

The following best practices should be followed when capturing information about cells in SBOL:

- Taxonomic information about a cell type **must** be captured using an instance of the *ComponentDefinition* class. This *ComponentDefinition* instance **must** have a type property of ‘cell’ from the Gene Ontology (GO:0000290), and **must** have a role of ‘physical compartment’ from the Synthetic Biology Ontology (SBO:0000290). The *ComponentDefinition* instance **must** also have an organism property which **should** be a NCBI URI or link to another database. This property **may** also be a description of the organism if no other record exists.
- Functional information about a cell type **must** be captured using an instance of the *ModuleDefinition* class. This *ModuleDefinition* instance **must** have a role

property of ‘functional compartment’ from the Synthetic Biology Ontology (SBO:0000289). The cell type should be instantiated within this *ModuleDefinition* as a *FunctionalComponent*. The *FunctionalComponent* **must** be defined by a *ComponentDefinition* which captures taxonomic information about the cell type as described above. Other entities which interact with or within the cell **must** be instantiated as a *FunctionalComponent* which refer to suitable *ComponentDefinition* class instances. Intracellular interactions **must** be captured using instance of the *Interaction* class, where the cell type *FunctionalComponent* **must** be included as a *Participant* with a role of ‘physical compartment’ from the Synthetic Biology Ontology (SBO:0000290).

- It is recommended that entities which are contained within the cell only be specified using an *Interaction* instance with a type of ‘containment’ (SBO:0000469). The cell type **must** be included as a *Participant* with a role of ‘physical compartment’, and the entity **must** be included as a *Participant* with a role of ‘contained’ (SBO:0000064).

For acceptance by the SBOL community, the proposal was first submitted as an SBOL Enhancement Proposal (SEP)[iii] and was subsequently voted to be accepted by members of the SBOL community. The above proposal was accepted as although it was the more complex approach, it allowed more context to be added to interactions involving cells. Additionally, this approach also distinguished between the natural cell strain (represented by a *ComponentDefinition*), and the cells implemented in a system (represented by a *FunctionalComponent* within a *ModuleDefinition*), which may have been modified, such as by transformation with a plasmid. This was not possible with cell representation A, where all information about the cell was contained within one *ModuleDefinition*.

Representing multi-microbial systems using SBOL

The following best practices should be followed when capturing information about multi-microbial systems in SBOL:

- The overall multi-microbial system **must** be represented by a *ModuleDefinition* instance. This *ModuleDefinition* **must** have a role of ‘functional compartment’ from the Synthetic Biology Ontology (SBO:0000289).

[iii] https://github.com/SynBioDex/SEPs/blob/master/sep_030.md

- Each cell type **must** be instantiated using the *Module* class, which **must** be defined by the *ModuleDefinition* representing that cell system, **and** using the *FunctionalComponent* class, which **must** be defined by the *ComponentDefinition* which captures the taxonomic information for that cell. The *Module* instance **must** include a *MapsTo* instance which has a refinement property of ‘merge’. The *MapsTo* local property **must** refer to the *FunctionalComponent* which represents the cell type in the multi-microbial system *ModuleDefinition*, and the remote property **must** refer to the *FunctionalComponent* instance which represents the same cell type in the cell system *ModuleDefinition*.
- Non-cell entities **should** be included as *FunctionalComponent* instances, and a *MapsTo* instance with a refinement property of ‘merge’ **must** be used to link the *FunctionalComponent* in the multi-microbial system *ModuleDefinition* (local) and the *FunctionalComponent* representing the same entity in the cell system *ModuleDefinition* (remote).
- It is recommended that any entities which are important for intercellular interactions are included in the multi-microbial system *ModuleDefinition* using the method described above.

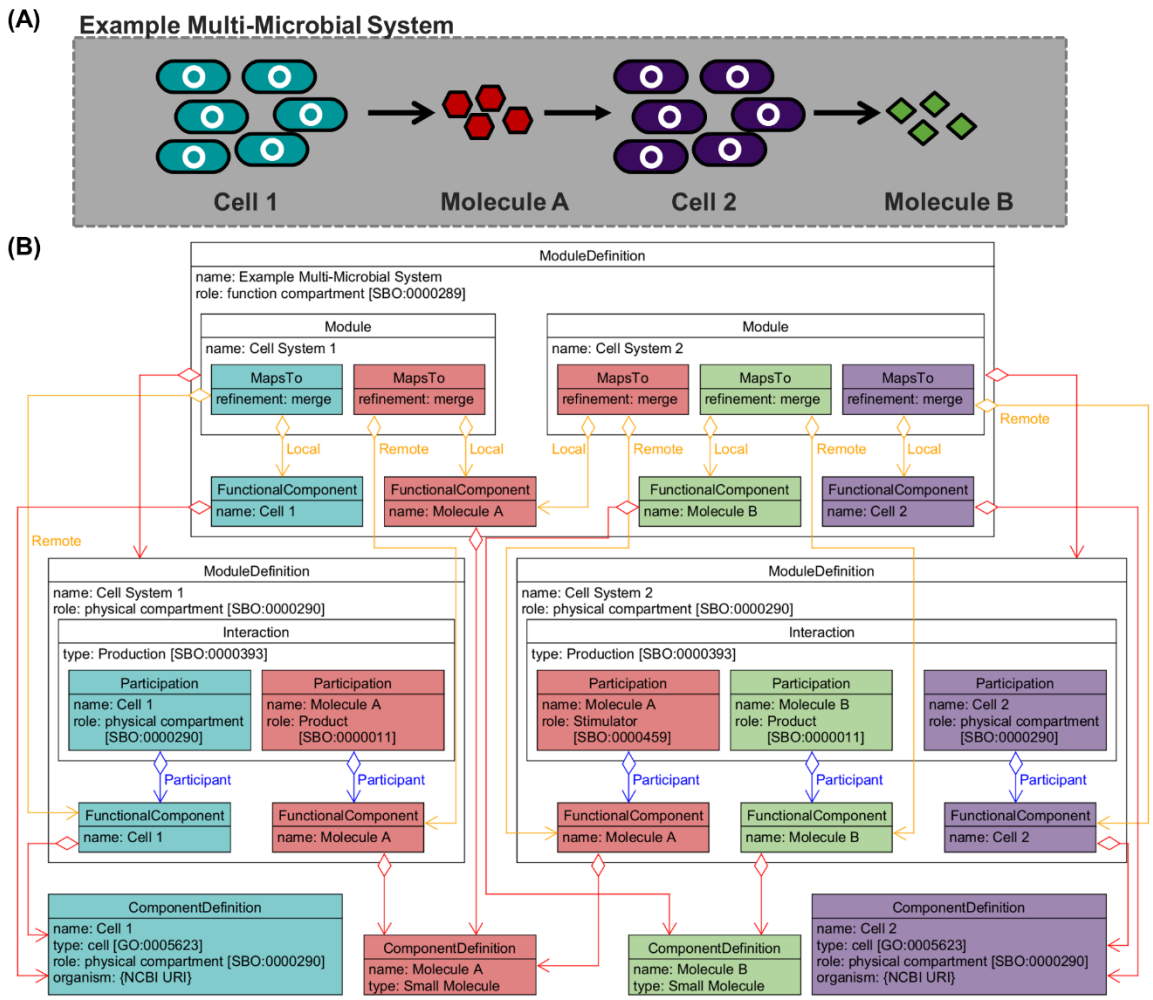


Figure 3.13. Example of Capturing a Multi-Microbial System using the Accepted Best Practices: This figure illustrates how a multi-microbial system can be captured using the accepted best practices. **(A)** Schematic of a simple multi-microbial system. This example system is composed of two cell populations: cell 1 and cell 2. Cell 1 produces molecule A, and cell 2 produces molecule B in the presence of molecule A. **(B)** UML diagram depicting how the example system in (A) can be represented using SBOL. Each cell type is captured by both a *ComponentDefinition* (CD) and *ModuleDefinition* (MD). The CD represents the physical cell and stores information about taxonomy, and the MD is used to convey information about the cell's functionality. For cell 1, an *Interaction* describes the production of molecule A. For cell 2, the *Interaction* describes how molecule B is produced when in the presence of a stimulant (molecule A). The overall multi-microbial system is captured using an MD instance, with all relevant instances (cell 1, cell 2, molecule A, and molecule B) represented by *FunctionalComponent*s. The *Module* classes are used to separate the individual entities into the cell types present.

These best practices most closely align to multi-microbial representation C described above. This approach for capturing information about multi-microbial systems was accepted by the SBOL community as, although it is more complex than multi-microbial representation A and B, it allows for intercellular interactions to be highlighted explicitly without duplicating interactions already specified in the cell system designs. Additionally, the use of *MapsTo* classes helped remove any ambiguity as to which entity pools are shared between cell types. To illustrate the accepted best practices, a simple example multi-microbial system is depicted in Figure 3.13.

3.4. Bio-Automation for Development of Modular Systems

Source code for the Python library described in this section can be found at github.com/intbio-ncl/BiomationScripterLib. Full documentation can be found at biomationscripterlib.readthedocs.io. Note that some of the text and images created for this thesis were duplicated for the documentation. The version of BiomationScripter described here is v0.2.0.

BiomationScripter was developed in collaboration with David Markham, Dr Jasmine Bird, and Dr David James Skelton. The concept for the library was my own, and more than 98% of the code committed was written by me. However, of particular note is the format of the Excel file for labware, which was designed by David Markham, along with the import functionality. Unless noted otherwise, all BMS protocols and templates presented and used throughout this thesis were largely developed and implemented by me, although some contain modifications by David Markham. Full attributions for the code can be found at github.com/intbio-ncl/BiomationScripterLib/graphs/contributors.

3.4.1. Background and rationale

Biological workflow automation by robotic systems has many advantages. One such advantage is the potential to increase reproducibility of experiments, which is well known to be essential for any scientific or engineering field[216], [219]. Bio-automation can help increase reproducibility by reducing human error, preventing deviations from protocols, and allowing better tracking of how a protocol was actually executed. Laboratory automation can also help free researchers from performing repetitive basic and tedious activities[220], in addition to allowing better exploration of large design spaces which may be otherwise unfeasible or time-consuming when performed manually[221].

The Sensynova Framework for modular and multi-microbial biosensor development has the potential to leverage bio-automation and make use of the advantages afforded by automating biological workflows. Characterising biosensor modules using automation could allow for better reproduction of results as other researchers can more easily repeat experiments exactly. Lab automation could also enable rapid testing of how different modules and biosensors respond to a variety of factors, especially in conjunction with a multifactorial Design of Experiments (DoE) approach[222]. Aside from characterisation, automation of DNA assembly workflows could also allow for easier and faster construction of module variants[59], [223], [224].

Whilst laboratory automation has seen continued increases in uptake, there are still significant barriers to entry, especially within academia[225]. One such barrier is that of cost, as many pieces of automation equipment are expensive to obtain and maintain[226]. Another barrier is the time required to learn how to program and operate various equipment, each of which tends to use proprietary software which may be vastly different from software packaged with other equipment of the same type[216], [227]–[229]. Even once the specifics of an automation equipment are understood, the conversion of manual protocols to automated workflows can be slow and tedious. Additionally, it can be difficult to re-use automation protocols for common workflows (such as DNA assembly), even when developed for the same equipment, as these protocols tend to be developed in such a way that they are highly specific[230]–[232].

There have been considerable efforts to reduce these barriers to laboratory automation. For example, companies such as Synthace offer ‘cloud laboratories’, where researchers can outsource their automation needs which can be cheaper and simpler than purchasing, maintaining, and learning the intricacies of their own automation equipment[233], [234]. These advantages are apparent when bio-automation is only intended to be used for a limited number of workflows or experiments, but costs can add up when considering automation as a mainstay of molecular and synthetic biology[235], [236]. Other efforts to increase accessibility to, and enhance the utility of, bio-automation have centred around the idea of universal programming, where many different automation equipment can be programmed using a universal interface, eliminating the need for users to learn a plethora of software[237]–[239]. These efforts range from open-source platforms which provide a novel high-level programming language from which biologists can describe an experiment and generate an appropriate protocol[230], [240], to proprietary platforms reliant on point-and-click interfaces such as Antha (antha-lang.com). Whilst these platforms have shown use, they also have significant downsides: most platforms only support a very limited number of equipment, many open-source projects have not been updated in many years, and the proprietary efforts tend to be tied towards a specific ‘cloud laboratory’, preventing easy usage by researchers with access to their own equipment[232]. Additionally, a focus on creating simple interfaces for generating automation scripts has often resulted in a lack of flexibility, constricting users to only applications considered by the developers[241]. There have been projects which focused on providing users with the ability to program equipment in common programming

languages, such as Python, which give the potential for high degrees of flexibility. However, oftentimes these types of projects, such as PyHamilton[242], focus on specific automation equipment, and tend to require a good understanding of programming to utilise.

Automation equipment tends to be costly to obtain. One notable exception is the OT2, an open-source liquid handler developed by Opentrons (New York, USA). This robot is available at a fraction of the cost of other automated liquid handling robots and can be programmed directly in Python. The open-source nature of the machine also makes it a good candidate for 'hacking', where the machine can be modified towards specific applications[223], [243]–[245]. There are, however, drawbacks associated with this liquid handler, such as reduced accuracy when compared to the higher-end liquid handlers, as well as a lack of built-in features like liquid detection and 96-channel pipetting. These limitations mean that many larger companies and biofoundries still rely on more expensive equipment, such as the Tecan Evo (Tecan, Männedorf, Switzerland), Hamilton STAR (Hamilton, Bonaduz, Switzerland), or Labcyte Echo 525 (Beckman Coultler, California, USA).

3.4.2. BiomationScripter: Overview

Presented here is a Python library, BiomationScripter, which was developed to help leverage the potential of automation in synthetic biology and address some limitations surrounding its uptake. To this end, the focus of BiomationScripter (BMS) was to help make the protocol development aspect of automating synthetic biology workflows easier and allow for better sharing and re-use of protocols. The BMS library was also used to develop tooling with the aim of introducing automation to the characterisation of modular and multi-microbial biosensors developed according to the Sensynova framework and provide the potential for automated construction of module variants.

The BMS library focused on protocol generation for two liquid handlers: OT2 and the Echo 525. The OT2 was selected due to its increasing ubiquity in academic research labs, and its relatively low barriers to access when compared to other liquid handlers. The Echo 525 provides an almost polar-opposite option as a high-end acoustic based liquid handler, capable of transferring liquid in the nanolitre range. The Echo is often used in synthetic biology for DNA assembly as its ability to handle low volumes allows for miniaturisation of reactions, resulting in the ability to assemble a vast amount of

genetic variants in parallel, as well as reducing the cost per reaction through reduced reagent usage[246]–[248]. Together, the OT2 and Echo 525 cover a wide range of use cases within synthetic biology, and the Sensynova Framework specifically. The decision was made to develop a programming library for bioautomation as scripting can provide users with a large amount of flexibility when developing automation protocols. The language of choice for BMS was Python, as this is a relatively lightweight language suitable for scripting of this nature and is already ubiquitously used within synthetic and computational biology[249].

The BMS library is split into three packages: one which contains a set of generic tools for bio-automation, one which provides specific tools for the OT2 liquid handler, and one which is specific towards the Echo 525 liquid handler. In addition to general tools included within the library, BMS enables the use of protocol 'Templates', which can be thought of as generalised implementations of a protocol which is largely performed in the same way each time, with variations between runs. For example, preparation of PCR reactions generally follows the same steps (addition of a DNA template, primers, polymerase, dNTPs, buffer, and water), however the exact DNA, primers, polymerase, and buffers used may vary, along with other user-defined parameters such as the number of reactions prepared, the reaction repeats, and the final volume of the reactions. Within BMS, protocol Templates are implemented as Python classes, and contain a `run` method which includes all of the necessary code to create the liquid handing steps based on specific user inputs. Each Template has required inputs; however, a variety of optional inputs allows advanced users more flexibility.

There are examples of previous work which have demonstrated the applicability of approaches such as the Templates described above, where automation protocols are generated with minimal user input. However, these previous efforts focused only on specific applications, and provided no support for creation of different, or even similar, protocols[223], [250]. In other cases, alternative methods of developing a range of automation protocols are provided, but with no support for simple generation of common protocols[230], [239], [242]. Here, the BMS Templates were developed and implemented in such a way that users with an intermediate knowledge of Python programming can develop their own custom Templates and make them accessible for use by others. Therefore, the BMS library and Templates not only provides tools for

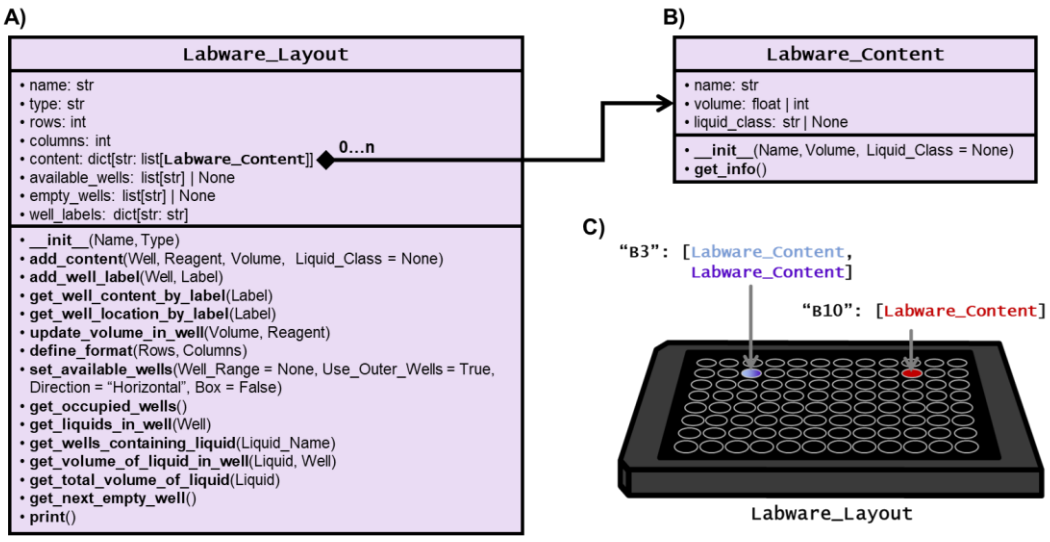


Figure 3.14. `Labware_Layout` and `Labware_Content` Classes
Architecture of labware representation in BMS. (A) The `Labware_Layout` class captures information about labware to be used in automation protocols. (B) Information about content contained within labware is captured using the `Labware_Content` class. (C) Illustration of labware representation in BMS.

rapid generation of automation protocols for synthetic biology workflows but also presents a basis for future expansion and use across a range of applications.

3.4.3. BMS equipment-agnostic tools: Labware

The first BMS package contains a set of equipment-agnostic tools which are applicable to protocol generation for a wide range of automation equipment. For liquid handling protocols, the presence of labware such as multi-well plates or tube racks is universal. These labware contain liquids, such as reagents, buffers, or cells. For example, DNA assembly workflows generally require DNA parts, plasmid backbones, water, and reagents such as enzymes and ligases. In this example, the source material (that is the DNA parts, plasmids, water, and reagents) could be stored in tubes held by a tube rack. In this case, the tube rack is the labware. The protocol would also require destination labware for assembly reactions to be prepared, such as a multi-well plate.

Within BMS, labware are represented using the `Labware_Layout` class, which acts as a labware 'blueprint'. The `Labware_Layout` class stores basic information about physical attributes of the labware (the labware type and number of rows and columns), along with a labware name. In addition to physical properties, `Labware_Layout` instances can track a labware's state. Wells (or slots) available for use can be specified, along with empty wells. Well content, in terms of liquid, volume, and optionally a liquid class (i.e. the properties of the liquid), can also be specified using the `Labware_Content` class. Further context for each well's content can be given the in

the form of a well label. For example, wells containing DNA assembly reactions may be labelled as such.

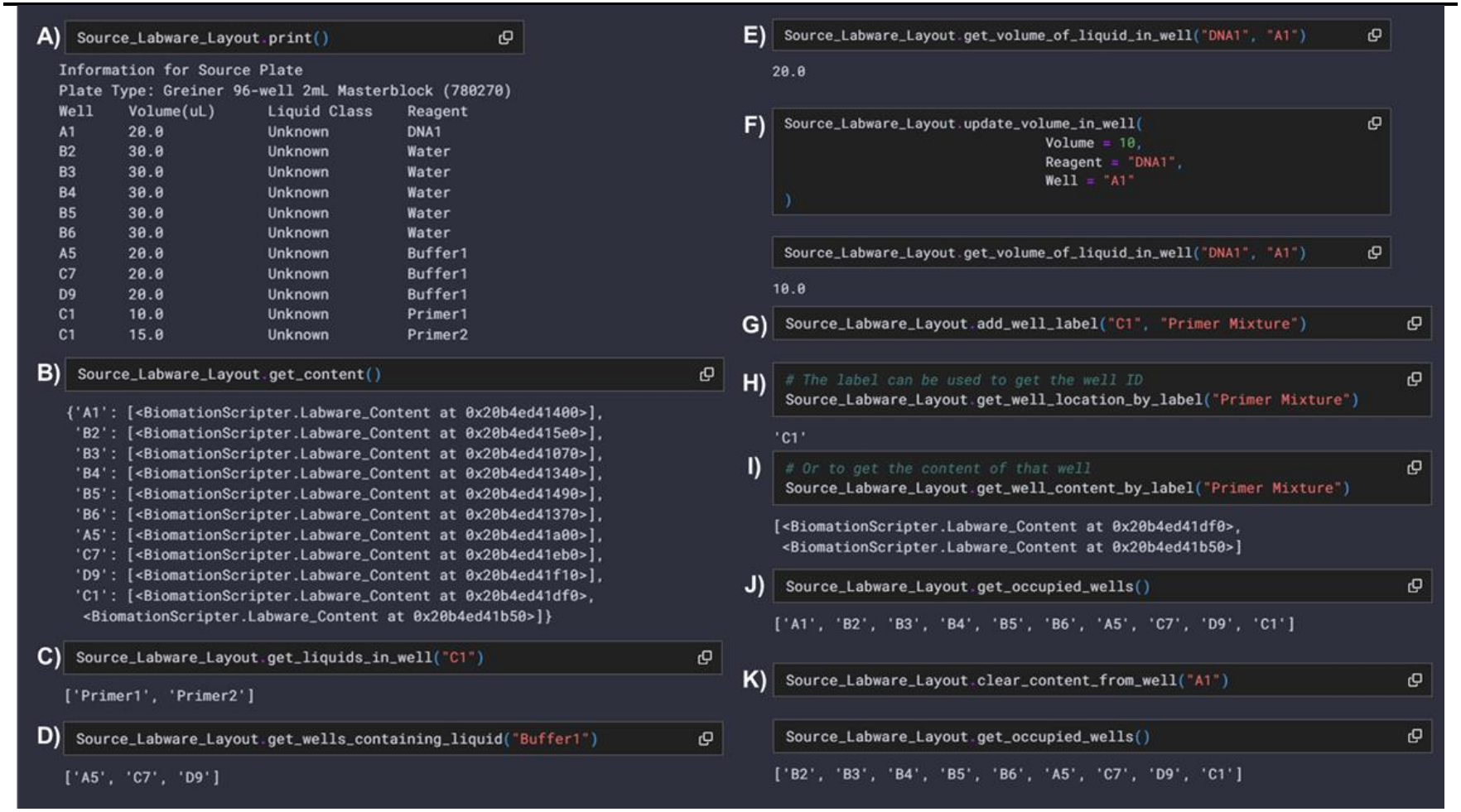


Figure 3.15. `Labware_Layout` Usage: Example usage of the `Labware_Layout` class

(A) Content in the `Labware_Layout` class can be displayed to OUT using `print` method. (B) A list of `Labware_Content` objects contained within the labware can be returned using the `get_content` method. (C) Liquids within a specific well can be retrieved using `get_liquids_in_well`. (D) Wells which contain a specific liquid can be retrieved using `get_wells_containing_liquid`. (E) The volume of a specific liquid in a well can be determined using `get_volume_of_liquid_in_well`. (F) The volume of a liquid in a well can

be changed using `update_volume_in_well`. (G) Wells can be labelled using the `add_well_label` method. (H) Wells can be retrieved via their label with `get_well_location_by_label`. (I) The content of a well can also be retrieved using a label with `get_well_content_by_label`. (J) Wells which are occupied can be determined using the `get_occupied_wells` method. (K) All content in a specific well can be removed `clear_content_from_well`.

A)

Plate Summary | Well lookup

	A	B
1	Plate Name	Example DNA Stocks
2	Plate Type	384PP
3	Total Wells	384
4	Rows	16
5	Columns	24
6	Minimum working volume	12
7	Maximum working volume	50
8	Description	All DNA in this plate are at concentrations of 10 fmol/uL

B)

Plate Summary | Well lookup

	Well	Row	Column	Name	Volume (uL) - Initial	Concentration (ng/uL)	Concentration (uM)	Volume (uL) - Current	Calibration Type	Notes
2	A1	A	1	pOdd1	50				AQ_BP	
3	A2	A	2	J23100	50				AQ_BP	
4	A3	A	3	B0034	50				AQ_BP	
5	A4	A	4	GFP	50				AQ_BP	
6	A5	A	5	mCherry	50				AQ_BP	
7	A6	A	6	B0015	50				AQ_BP	
8	A7	A	7							
9	A8	A	8							
10	A9	A	9							
11	A10	A	10							
12	A11	A	11							
13	A12	A	12							

Figure 3.16. Example of a Standard Format Labware File

Example of an Excel file for capturing information about a piece of labware. (A) Metadata relating to the labware. (B) Content of the labware.

The `Labware_Layout` class provides a variety of methods for easy retrieval and modification of the labware's state. Figure 3.15 shows examples for some of the key `Labware_Layout` methods. Other methods, such as `get_total_volume_of_liquid`, which returns the total amount of a certain liquid within a labware, also exist. The `Labware_Layout` methods were developed to help users easily query and modify the state of specific labware.

In a number of cases, labware will exist beyond the scope of a single protocol or workflow. An example of this is with DNA storage plates, where standard DNA parts and plasmids are stored within a plate which is re-used in many protocols. In these cases, it is necessary to have the option of long-term storage of a `Labware_Layout` object. Within BMS, this takes the form of a standard format Excel file which can be imported to create a `Labware_Layout` object. The Excel file contains two sheets. The first sheet is named 'Plate Summary' and contains the name and type of the labware, along with the total number of wells and number of rows and columns. There are also options to provide maximum and minimum well volumes and a short description of the labware. The second sheet, 'Well lookup', is where labware content is stored. It should be noted that the mapping of the Excel sheet to a `Labware_Layout` class is not exact. The only information imported to BMS are the labware's name and type, the number

of rows and columns, and the name, volume, and liquid class of any content. The decision was made to exclude other information from import as it is generally not needed in order to run a protocol and creating a full data model for all protocol metadata is outside the scope of BMS. The ability to import a labware layout from a file has the advantage of not only providing a method of using labware which exists outside the lifetime of a single protocol, but also allows users with less programming experience to more easily define their labware.

3.4.4. BMS equipment-agnostic tools: Common features

There tend to be similar requirements for many liquid handling protocols, and the purpose of the generic BMS functions are to aid implementing these requirements without having to re-develop methods for achieving them. For example, many protocols will require source material to be provided as aliquots, as the volume of liquid required is too large for compatible labware. In these cases, BMS can automatically calculate the number of aliquots required and split the required volume, considering dead volumes (Figure 3.17 (A)). The number of aliquots is determined via Equation (4.1)a.i.1.a.i. BMS can also calculate the total number of a specific labware needed for a protocol based on the number of wells or slots required (Figure 3.17 (B)).

$$Number\ of\ Aliquots = \left\lceil \frac{Total\ Volume\ of\ Source\ Material\ Required}{Aliquot\ Volume - Dead\ Volume} \right\rceil \quad \text{i.}$$

Another common function provided by BMS is automatic generation of transfer events for individual liquids (Figure 3.17 C). This function produces lists which specify the transfer volumes and destination location of a named liquid. This information can then be used by the more specific functions in OTProto and EchoProto to generate liquid handling instructions for the desired automation equipment.

Mastermixes are commonly used in molecular and synthetic biology to assist with the preparation of reactions or experiments which share a proportion of their composition, and perhaps differ in only a few components. In these cases, mastermixes can be prepared which contain shared components, and which can be divided across different reactions or experiments. This gives several advantages, including consistency in preparation, as slight pipetting inaccuracies when adding the individual components will be the same across all reactions, and saving time during setup as individual

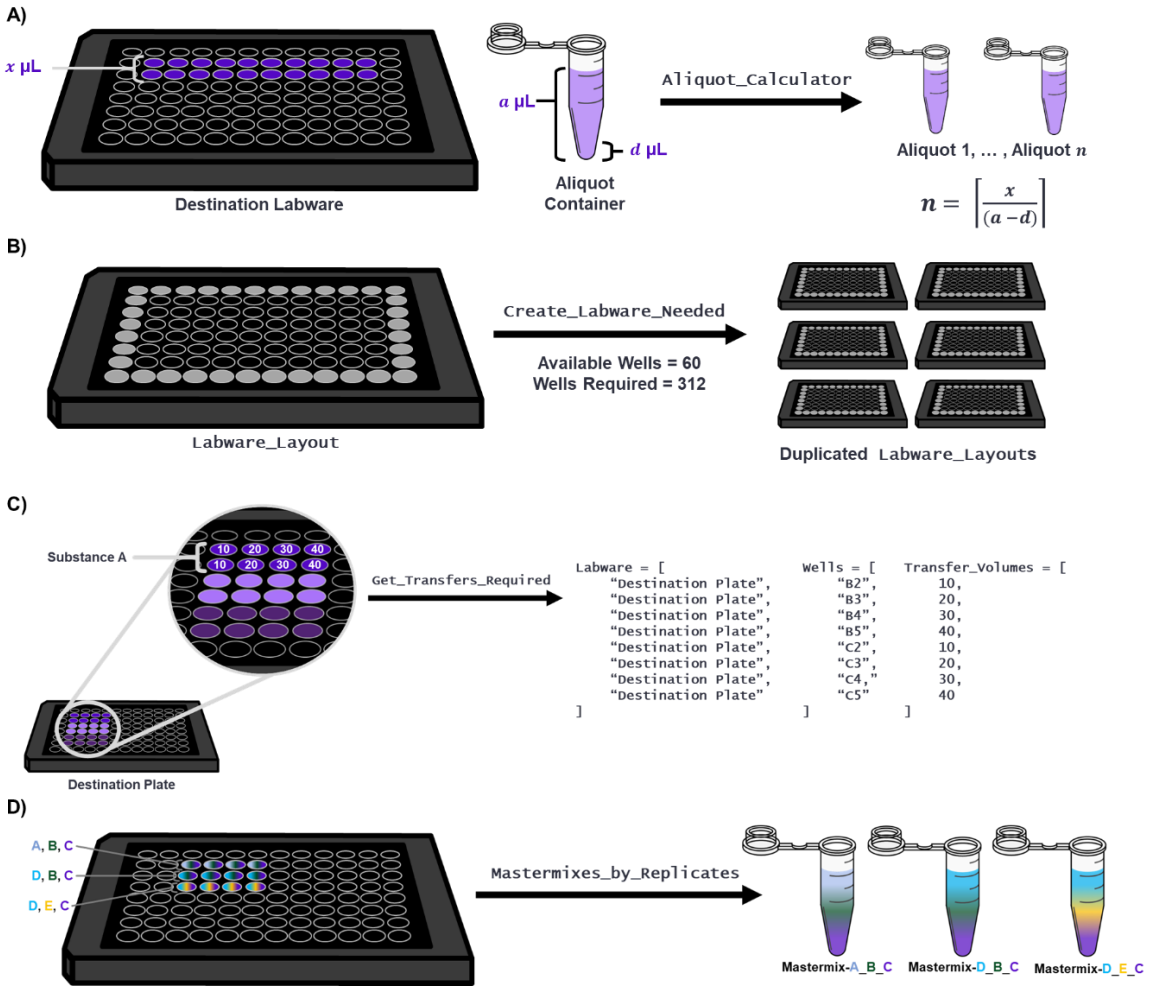


Figure 3.17. General Purpose BMS Functions
Illustrations depicting some of the BMS functions. (A) `Aliquot_Calculator` is used to determine source material aliquots are required by an automation protocol, based on the amount of volume required. (B) The required number of `Labware_Layout` objects for an automation protocol can be generated based on a template layout object, the number of available wells per labware, and the number of wells required. (C) Information regarding transfer actions for a specific liquid into a destination labware can be automatically generated using `Get_Transfers_Required`. The destination labware, well, and transfer volume are generated, which can be used by other functions. (D) Mastermixes can be automatically generated using `Mastermixes_By_Replicates`. Within a list of destination labware, wells with identical proportions of the same source materials are grouped, and mastermixes for those wells are generated and added to a `Labware_Layout` object. The destination labware is then updated to indicate which wells should be supplied by which mastermixes.

components can be added to the mastermix in one step, rather than being added separately to all experiments. Additionally, preparing mastermixes can help prevent liquids being transferred at low volumes, which can lead to inaccuracies depending on the equipment used. These advantages are true of both manual and automated setup.

BMS provides a function to generate mastermixes based on replicates found in destination labware. Users can specify the maximum volume of mastermix aliquots, the number of extra reactions mastermixes should be prepared for (which can help

account for dead volumes and under-pipetting accuracies), and a minimum transfer volume. The function then automatically identifies identical reactions or experiments within the destination labware and populates a mastermix labware layout object with mastermixes containing the required components (Figure 3.17 D). The destination layout objects are also updated such that the contents of the wells now refer to the newly created mastermixes, rather than individual components. The mastermix layouts contain the individual components as content, and the mastermix name is stored as the label for the well or slot it occupies.

The functions and classes mentioned thus far cannot be used to directly generate liquid handling instructions. Instead, there are two separate modules within BMS which aim to assist with the generation of instructions for the OT2 liquid handling robot, and the Echo 525 acoustic-based liquid hander. These are described in the following sub-sections.

3.4.5. BiomationScripter: OTProto

The OT2 robot includes two pipette slots which can be occupied by a range of different pipettes. The pipettes aspirate and dispense liquids via a piston-based mechanism, which is controlled by a stepper motor. Opentrons provides a Python API to control OT2 liquid handlers, as well as simulating protocols prior to execution by the robot. The native Opentrons API mainly focuses on providing an interface for controlling the core OT2 functionalities, such as aspirating, dispensing, mixing, and controlling the hardware modules. There is, however, a noticeable lack of so-called wrapper functions, which can be used to provide more complex and intelligent tools. Wrapper functions can be used to handle the common logic and calculations required of many liquid handling protocols, allowing users to focus on protocol-specific features, and reducing the requirement for repeated development of code with identical functionalities. By providing these wrapper functions, OTProto allows users to develop protocols with more complex logic without needing to understand the underlying Opentrons API in great depth, and also helps disincentivise hard-coding of run-specific information in such a way which makes modification of a protocol by other users difficult. This is because the wrapper functions can easily cope with changing parameters which can differ between runs (such as labware types, sample numbers, and number of tip boxes required) without needing to re-write large sections of the code.

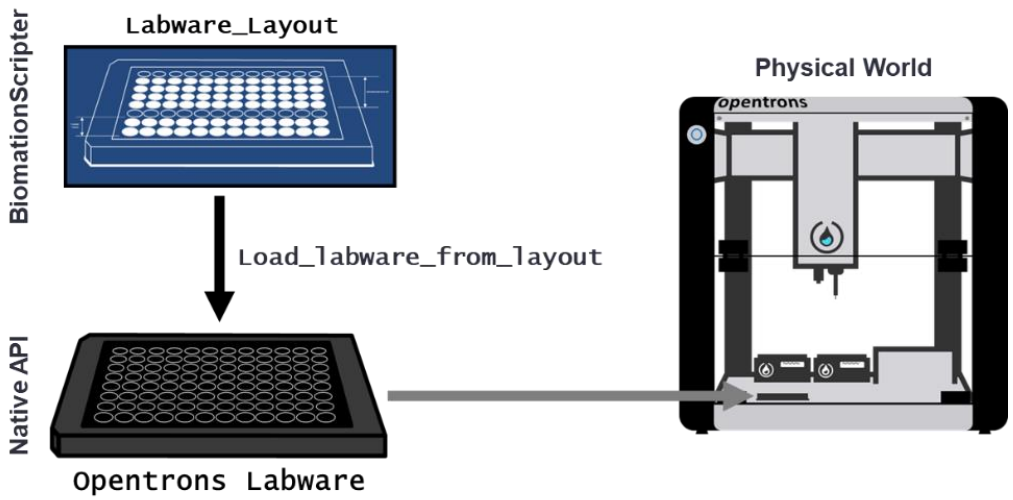


Figure 3.18. Labware Representations within OTProto
Relationship between labware in BMS, the native Opentrons API, and the physical world. The `Labware_Layout` class in BMS can be used to capture the state of a labware in terms of current content. The Labware class in the native Opentrons API represents the physical attributes of the labware and is used to inform OT2 robots as to the XYZ co-ordinates of individual wells and other positions in the physical world.

When using the native Opentrons API, labware is split into two types: default labware which is included within the Opentrons labware library, and custom labware which is not pre-defined. The Opentrons API uses a `Labware` class to inform the robot about the specifics of labware loaded to the deck and are instantiated using a specific labware API name. The API name provides a reference to an entry in the default labware library, or to the definition file for the desired custom labware. The loading of custom labware when using the native API can differ depending on how the protocol will be executed and/or simulated. Additionally, in some cases the way in which default and custom labware are loaded may also differ These differences can cause extra complexities during development of a protocol, may require modification of a protocol between simulation and execution, and can reduce the flexibility of a protocol in terms of easily replacing labware from run-to-run. OTProto provides the `load_labware` function to handle these issues, which allows users to load labware in the same way each time.

OTProto also has the `calculate_and_load_labware` function, which will determine the amount of `labware` objects of a specified type required based on the number of wells needed by the protocol. There are also the `get_labware_format` and `get_labware_well_capacity` functions which allow users to get more easily (i) the number of rows and columns of a labware and (ii) the well capacity of a labware. These

functions allow for the development of more flexible protocols, where information about the labware (such as how much liquid can be stored in a labware) do not need to be hard-coded.

The representation of labware by the native API differs to BMS, which uses the `Labware_Layout` class. However, within the OTProto module, these two classes can act in synergy. The native Opentrons `Labware` class captures physical information required by the OT2 robot to convert well positions into XYZ co-ordinates, whereas the BMS `Labware_Layout` class can be used to capture the state of the labware, storing information about the labware's content or intended content (Figure 3.18). Using the OTProto `load_labware_from_layout` function, it is possible to create Opentrons labware objects using a BMS layout object. The main requirement here is that the `Labware_Layout`'s type should be an Opentrons labware API name.

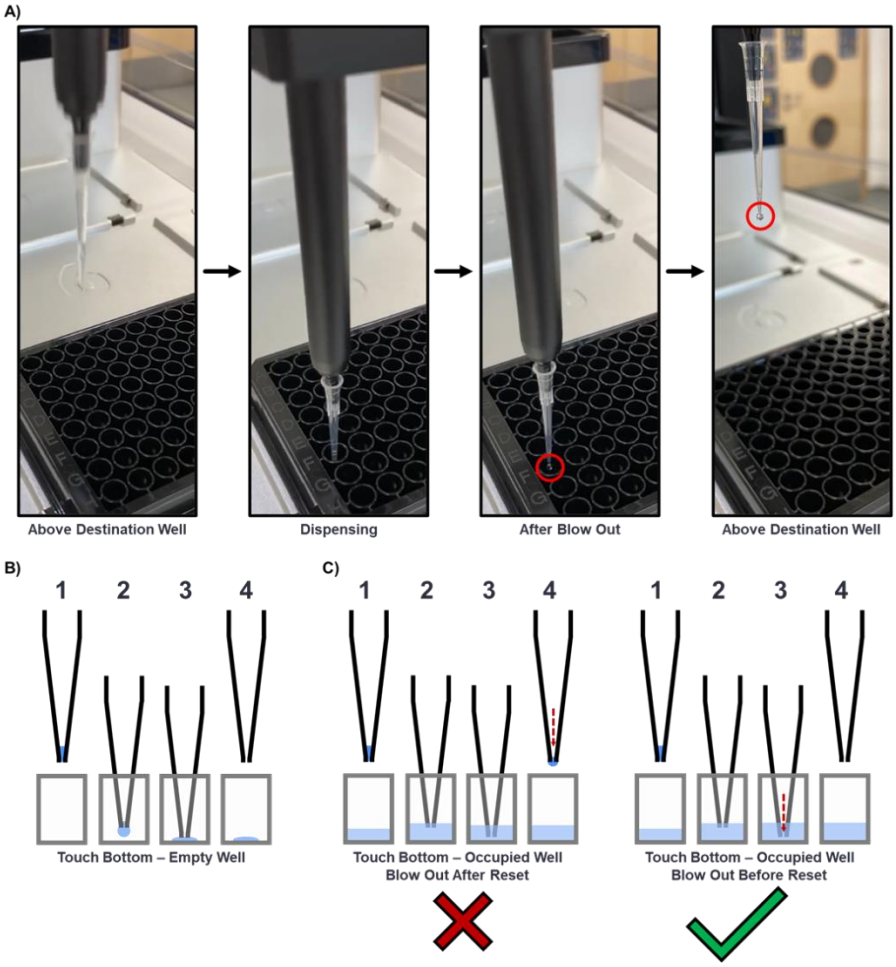


Figure 3.19. Opentrons Liquid Transfer Behaviour
Overview of the OT2 robot transferring low volumes of liquid. (A) Depiction of liquid droplets remaining on the pipette tip after dispensing and blow out when transferring low volumes of liquid into an empty well. (B) Method employed by BMS to ensure droplets are removed from the tip. After dispensing, the tip moves to touch the bottom of the well which causes the droplet to be removed. (C) Options for when blow out should occur. If blow out occurs after the tip position has reset to above the well, then a droplet can still form if liquid is already in the well. This is because a negative pressure differential can cause liquid to be slightly sucked up into the tip. If the blow out occurs whilst the tip is at the bottom of the well, it is less likely that liquid will remain in the tip.

In terms of generating liquid handling commands, OTProto includes two main functions: `transfer_liquids` and `dispense_from_aliquots`. The `transfer_liquids` function generates OT2 commands from a list of transfer volumes (specified in µL), a list of source locations, and a list of destination locations, such that each index within these lists describes a single transfer event. Unlike the transfer functions provided in the native API, the OTProto function will automatically select the most appropriate pipette to use for each transfer based on those which are loaded. There are also a range of optional arguments which can be supplied to modify the way in which liquid is transferred to help optimise the transfer events for specific protocols. These are all

possible to perform using just the native API, however OTProto wraps them together for easier implementation. The `new_tip` argument allows users to specify whether or not a new tip should be used for each transfer event specified. Beyond this, there are arguments to define mixing events (repeated aspirate and dispense actions) for the source and/or destination locations, and arguments to modify the speed at which liquid is aspirated, dispensed, or mixed. The ability to control these speeds can be important for liquids which are either significantly more or less viscous than water. There is also the option to define a ‘blow out’ action, which pushes extra air through the pipette tip to ensure all liquid is dispensed. This blow out can be performed into the source location, destination location, or the integrated trash. During initial testing of liquid transfer on the Opentrons, it was found that even with blow out actions, liquid droplets can still remain on the end of the pipette tip, particularly when dispensing low volumes of liquid into empty labware (Figure 3.19 A). To help tackle this issue, an option was added to the `transfer_liquids` function to move the pipette to the bottom of the well after dispensing. This allows any droplet on the pipette tip to be removed (Figure 3.19 B). In the case where a well in occupied, moving the tip to the bottom of the well can cause liquid to remain within the bottom of the tip. To prevent this, the blow out action is performed before removing the tip from the liquid, to ensure all liquid is expelled (Figure 3.19 C). These actions are optional and thus can be modified by users to ensure accurate pipetting based on the specific application.

The `dispense_from_aliquots` function acts similarly to `transfer_liquids`, and all of the optional liquid handling parameter arguments are still valid. However, `dispense_from_aliquots` allows users to present a list of aliquots of a specific liquid instead of a list of source wells. Based on the volume of each aliquot and the transfer volumes specified, the function using volume tracking to select the source location to aspirate from for each transfer event, ensuring that aspiration from empty aliquots does not occur. This helps make protocols more flexible and makes the use of aliquots simpler for protocol developers.

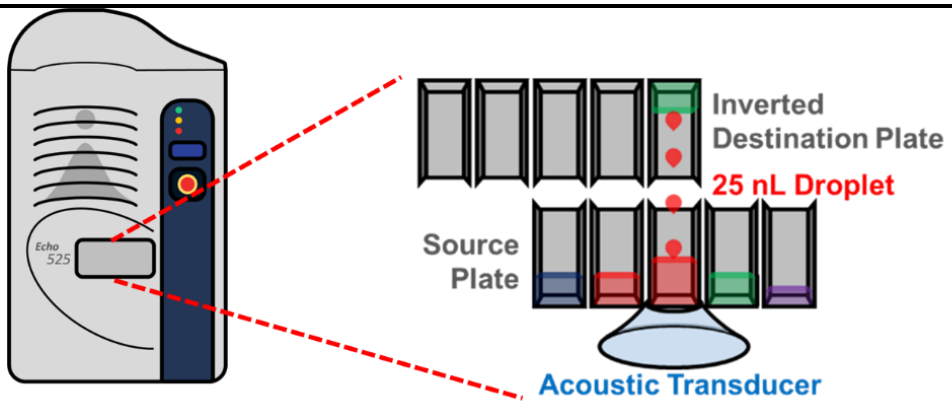


Figure 3.20. Echo 525 Liquid Transfer Mechanism
Schematic showing how liquid is transferred by the Echo 525. The acoustic transducer positions below the source well and emits an acoustic wave. The wave causes droplets to be expelled from the top of the liquid in the source well into the well of an inverted destination plate. The droplets formed are 25 nL in volume. Multiple droplets can be expelled to reach the required transfer volume.

Aside from these functions, OTProto includes a number of less complex functions, such as determining the number of tip boxes required for a protocol, getting the next empty slot on the OT2 deck, and associating tip boxes with pipettes. These functions are simpler to implement within a protocol than using the native API, requiring far less lines of code and less understanding of how the API works.

3.4.6. BiomationScripter: EchoProto

The Echo 525 is a liquid handler which is capable of transferring low volumes of liquid in the range of nanolitres. The robot functions via the use of acoustic sound waves to eject droplets of liquid, which are 25 nL in volume, from an acoustically validated source plate into wells of a destination plate, which is held inverted above the source plate (Figure 3.20). This has the advantage of allowing molecule biology reactions to be miniaturised which reduces the overall cost per reaction and has even been shown to increase efficiency in some cases. There is also the advantage of the transfers being pipette-less, which means that running costs can be reduced as tips do not need to be purchased. However, the source plates are very specific and must have been validated acoustically to ensure that the droplets generated by the acoustic waves are as close to 25 nL as possible.

The generation of Echo protocols can be somewhat tedious, especially for large or complex experiments. This is because main method of creating the protocol is to use one of the Echo's proprietary software, which are point-and-click interfaces. The other method of creating an Echo protocol is the use of 'picklists', which are CSV files where

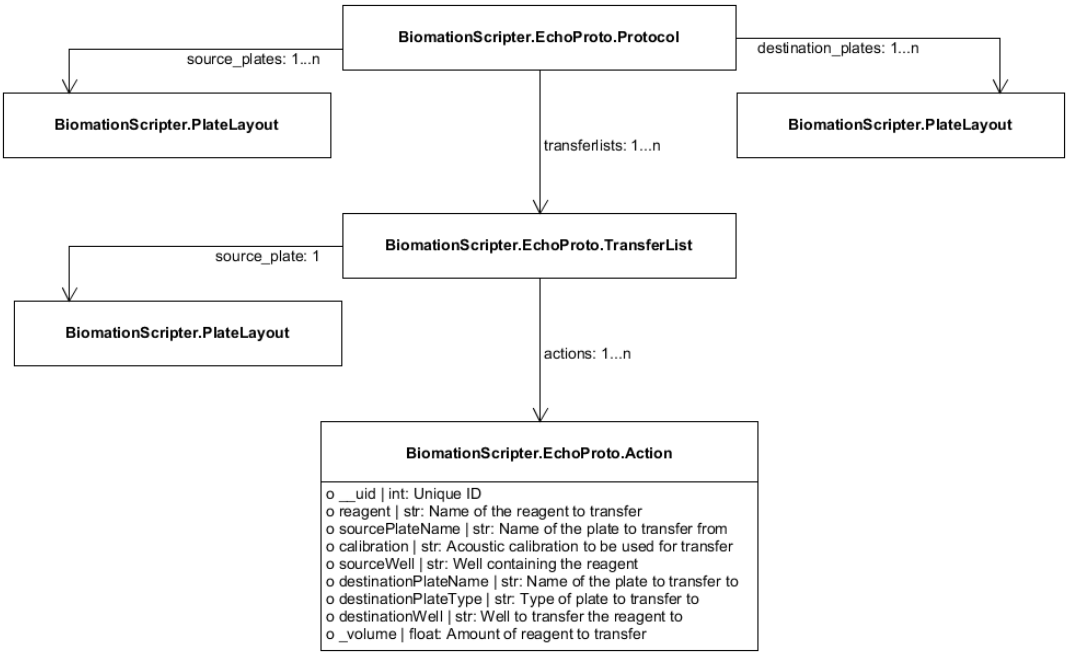


Figure 3.21. EchoProto Architecture
Architecture of the EchoProto module. All information for an EchoProto protocol is captured by the `Protocol` class. Aside from metadata such as the protocol name, the available source plates are stored using the `Labware_Layout` class from BMS. Destination plates containing the desired content after the protocol has completed are also captured using the `Labware_Layout` class. Information regarding transfer events from each source plate are captured using the `TransferList` class, and information about the transfers themselves is captured using the `Action` class.

each line defines a transfer event. These two methods can be somewhat slow when writing large protocols and are not particularly easy to quickly modify to adjust aspects of an experiment such as sample types, number of reactions, or reaction proportions. EchoProto aims to solve these issues by providing a Python interface for the creation of the Echo picklists, allowing for protocols to be scripted, and thus enable a level of automation in protocol development.

Implementation of the EchoProto module in BMS differs from OTProto; OTProto extends a pre-existing API and allows for direct control of the bio-automation robot, whereas EchoProto has no pre-existing, open-source API to be based on. Instead, EchoProto provides a method of generating CSV picklist files using Python scripts, which are then converted into liquid handling instructions by the Echo's proprietary software.

The architecture of EchoProto revolves around three core classes: `EchoProto.Protocol`, `EchoProto.TransferList`, and `EchoProto.Action` (Figure 3.21). The `Protocol` class holds all information for the entire protocol, including any metadata. The `TransferList` class is used to capture all transfer events from a single

source plate. The individual transfer actions themselves are captured by the `Action` class. It should be noted that transfer actions are split by source plate due to a quirk of the Echo's proprietary software; when generating liquid handling instructions from CSV picklist files, only one type of source plate can be specified at a time. Like OTProto, EchoProto also makes use of BMS's `Labware_Layout` class. Here, source plates are defined using the layout class, where the type must be one of the three source plate types accepted by BMS (384PP, 384LDV, or 6Res), and the content is populated with the available reagents. These source `Labware_Layout` objects can be defined either within the code or imported from a Labware Layout Excel file. The destination plates are defined in a similarly, however the content should specify the desired final state once all liquid has been transferred, and the range of plate types is much larger. Once defined, these `Labware_Layout` objects are stored within a `Protocol` object as either a source or destination plate. The `Generate_Actions` function can then be used to automatically generate the transfer events required to prepare the destination plates specified by the user. The function can handle aliquots of the same reagent across a source plate and will use volume tracking to ensure that once one source well is depleted, the next well containing the required reagent is used. In the case of a lack of source material, the user will be prompted, and informed which reagents are lacking, and told how much extra is required. The user will also be alerted to any wells which are below the working range for that plate and hence cannot be used for the protocol. Once the actions are generated, the `Write_Picklists` function is used to generate the CSV picklist files. The user has the option of where these files should be saved. There is also the option to group transfers from source plates of the same type into a single picklist, or to have a separate file for every plate. As mentioned previously, source plates of different types cannot be grouped into a single file. Once the files are generated, they can be imported to the Echo and the protocols can be performed.

3.4.7. OTProto Templates

OTProto Templates for different types of protocols or experiments are created by extending the `OTProto_Template` superclass. This superclass contains a number of methods and attributes which help keep some level of standardisation across all OTProto Templates and helps users with developing the Templates. For example, the number and type of tips required throughout the protocol can be tracked using the `calculate_and_add_tips` method, and the appropriate number of tip boxes can be added and associated with the correct pipette using the `add_tip_boxes_to_pipettes`

method. Runtime prompts alerting users as to the number of tips and boxes can also be generated by using the `tip_rack_prompt` method. The general purpose BMS and OTProto tools described previously can be used here to help easily implement flexible code for generating liquid handling instructions.

An OTProto template is created by making a `Template` class within a python file (named for the protocol), which extends the `OTProto_Template` superclass. The `__init__` method of this new class should be used to collect the user-defined information required by the protocol and store them as attributes for later use. The bulk of the code for generating the liquid handling instructions should be contained within a `run` method, which can then be called by users at runtime. To illustrate the process further, a simple example for mixing different coloured liquids on the OT2 is described below. Code for this example can be found in the documentation (biomationscripterlib.readthedocs.io/en/latest/example_code/OTProto/OTProto_Template-Superclass). and a flowchart describing the general steps taken by the Template (Figure 3.22) are also presented.

The requirements for the example OTProto Template are as follows: (i) to take a list of coloured solutions, (ii) to prepare 2-colour mixtures in equal amounts in a destination labware specified by the user, (iii) to allows the user to define the final volume of the mixtures, and (iv) to allow the user to define if the mixtures are permutations or combinations. The Template class for this protocol is defined by extending the `OTProto_Template` superclass. The required arguments are then defined in the `__init__` argument and stored as attributes. Keyword arguments required by the superclass, namely the protocol's name, metadata, starting tip positions, and custom labware directory location, are passed to the superclass. Next, a `run` method is defined. Within this run function, the 2-colour mixtures are defined and stored in a list. A `Labware_Layout` object is then created for the destination labware, and the colour mixtures are added to it as content. Next, a `Labware_Layout` object is defined for the source labware, and the number of aliquots of each colour are determined. These aliquots are then added to the source layout. Opentrons `Labware` object are then generated from these layout objects and loaded to the deck. The transfer events required are determined using `get_transfers_required`, and this information is used to determine how many tips and tip boxes are required, which are then loaded to the deck. Finally, the `dispense_from_aliquots` function is used to generate the liquid handling

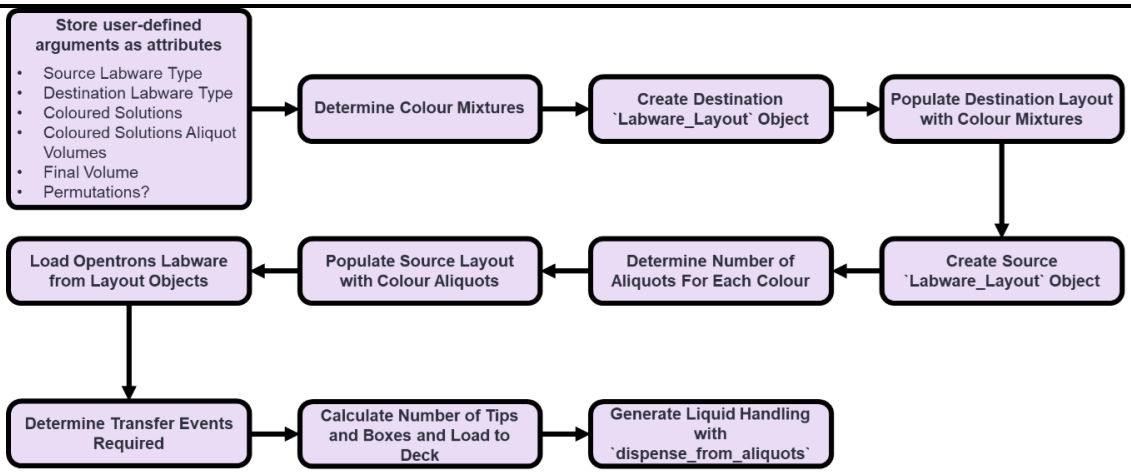


Figure 3.22. OTProto Template Example Flowchart
General steps performed by the colour mixing OTProto Template to generate liquid handling instructions.

commands. This Template can then be defined by a user, and a liquid handling protocol for the OT2 will be generated based on the arguments supplied, with minimal effort from the user.

OTProto has several Templates which have been developed and tested. These include a heat shock transformation Template, a Template for spot plating cells onto an agar plate, and a Template which prepares a microplate for calibrating a plate reader according to the iGEM standardisation procedure[63]. For the heat shock transformation template, the general functionality involves firstly dispensing competent cells into the transformation labware, which is located on either the Opentrons temperature module or the Opentrons thermocycler module. The DNA to transform with is then added to the cells and mixing occurs by pipetting up and down. The transformations then undergo the heat shock step, are cooled down to 4°C, and the user is prompted to supply the transformation media to the OT2 deck. The media is then dispensed into the transformation plate, which can then be placed into a shaking incubator for the growth stage of the transformation. Users are able to modify the transformation protocol by specifying parameters such as the labware types to use, the volume of competent cells, DNA, and media per transformation, the number of replicates for each transformation, the heat shock temperature and time, and whether or not the competent cells source labware should be placed onto a temperature module to keep them cool. By default, the transformation labware is kept at 4°C until the heat shock step.

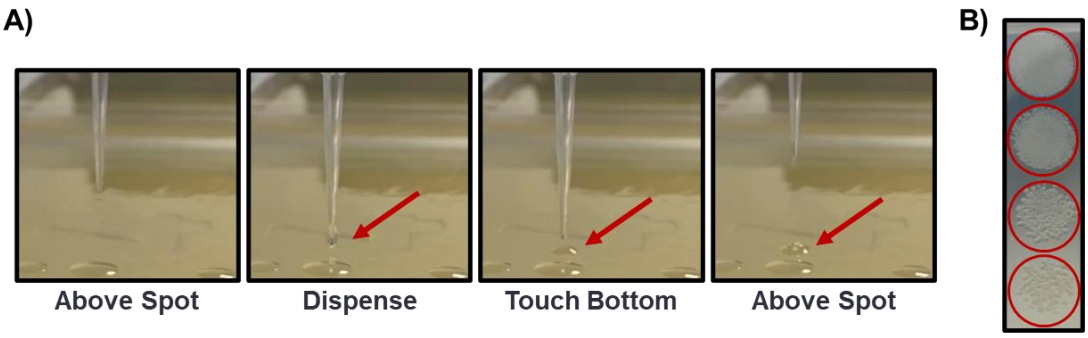


Figure 3.23. OTProto Spot Plating

Spot plating methods employed by the OTProto spot plating Template. (A) After dispensing cells, the pipette tip moves down slightly into the agar plate below to ensure no droplets are left on the tip. (B) Serial dilutions of the cells are performed to ensure that single colonies are obtained. The highlighted circles show the results of increasing dilutions of a cello culture after incubation overnight at 37ºC.

The spot plating Template is intended to add cells to an agar plate by spotting small volumes of liquid onto the plate. This can be used to automate the plating stage of transformations. The Template also includes a dilution stage, where the cells can be diluted to different concentrations before plating. This can be useful to help ensure that single colonies grow on the agar plate, rather than a circular lawn. Initial testing of this Template showed that when small volumes (< 5 µL) of cells are used to create spots, the liquid may remain on the pipette tip as a droplet, similar to the observation presented in Figure 3.19. This droplet then has the potential to fall off as the pipette moves across the deck, risking cross-contamination. These small volumes are very common when using the spot plating Template, as larger volumes tend to create spots with large diameters on the plate and can cause multiple drops to merge together. To prevent this, the `move_after_dispense` argument of the OTProto transfer function was used to move the pipette down and slightly into the agar after dispensing, ensuring that droplet is always removed (Figure 3.23). This does have the disadvantage of occasionally piercing the agar slightly, however the depth to which it pierces does not appear to cause any negative effects and is required to ensure spotting is routinely successful. Users are able to customise specific instances of the spot plating Template by specifying parameters such as the cell dilutions, the spotting volumes (which can be a list to spots of many sizes for each source cell culture), the repeats per source cell culture, and the types of labware used. Users can also specify whether the protocol should be paused after the dilution step and before the spotting step, allowing the agar plates to only be supplied when needed and help prevent unnecessary contamination from being exposed to the environment.

The iGEM standard plate reader calibration protocol can be used by researchers to convert arbitrary units acquired from a plate reader (fluorescence intensity and optical density) into absolute units[63]. The advantage of using absolute units rather than arbitrary (or relative) units is that data reported using such units can be compared directly, no matter what type of plate reader was used to collect the data. Data reported using arbitrary units cannot be used in this way as the values obtained differ between types of plate readers, and even plate readers of the same type. The calibration protocol works by calibrating arbitrary fluorescence values against a standard curve of a calibrant which has similar fluorescent properties to the fluorescent protein being measured. Similarly optical density, which is commonly used as a proxy for number of cells in a sample, can be calibrated against a standard curve of microspheres which have similar dimensions to the cells being measured. The plate reader calibration Template allows users to easily set up a calibration plate, which contains serial dilutions of the calibrants in a microplate. The calibration plate can then be measured using the described plate reader, and the data can be used to help convert units from experimental data collected using that plate reader into absolute units. The calibration Template accepts arguments from the user such as the calibrant types, stock concentrations, and initial concentration at the 1 in 1 position of the serial dilutions. The volume per well and number of repeats, as well as the solvents in which the calibrants should be diluted can also be specified. The types of labware to use at all stages are also customisable. Finally, the Template allows users full control over the liquid handling parameters, such as controlling the speed at which different liquids are aspirated and dispense, how the dilutions are mixed, and the option to enable actions like touching the tip to the sides of the wells after aspiration and dispensing, blowing out air from the tip after dispensing, and moving the tip to the bottom of the well after dispensing. Therefore, users can optimise their protocol and ensure the highest accuracy possible. For all of these options, the Template has default values, allowing users to ignore this extra complexity if desired. Results from using this Template are presented in section 2.6.

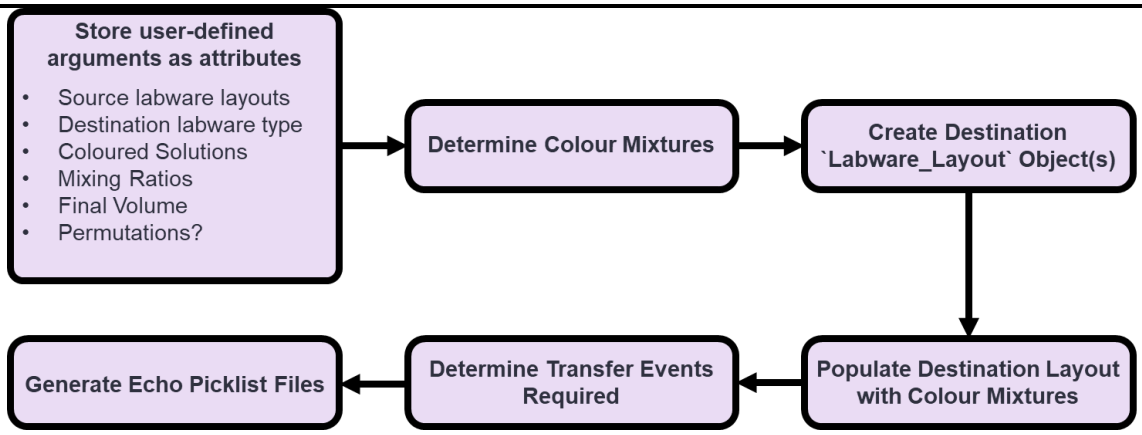


Figure 3.24. EchoProto Template Example Flowchart
General steps performed by the colour mixing EchoProto Template to generate picklist files.

3.4.8. EchoProto Templates

Like OTProto, EchoProto enables the development of protocol Templates, but for the Echo liquid handler, and they extend the `EchoProto_Template` superclass. There are three methods associated with the EchoProto superclass to help developers create Templates: `add_source_layout` and `add_destination_layout` help to track the source and destination plates respectively, and `create_picklists` can be used to automatically generate the picklist files based on the source and destination layouts added. Again, like OTProto, developers should use the `__init__` method of their Template class to collect the required user inputs and store them as attributes, and a `run` method should be added which contains the code for populating the labware layout objects and generating the picklists. An example EchoProto Template is shown in Figure 3.24 and can be found in the documentation (biomationscripterlib.readthedocs.io/en/latest/example_code/EchoProto/EchoProto-EchoProto_Template-Superclass), which can be used to generate 2-colour mixtures, and functions similarly to the example shown for OTProto.

Aside from the example Template mentioned above, two other Templates were developed for the Echo 525.The first Template can be used to help prepare PCR reactions. Users are able to supply a list of source plates as labware layout objects, which may either be generating using code or imported from a labware layout Excel file. The destination plate to use is supplied as an unpopulated labware layout object, allowing users to define the wells available for use (supporting re-use of partially used plates). PCR reactions are defined by the user as a list of tuples in the format `(DNA, Primer1, Primer2)`. Specifics for preparing the PCR reactions, such as the type of

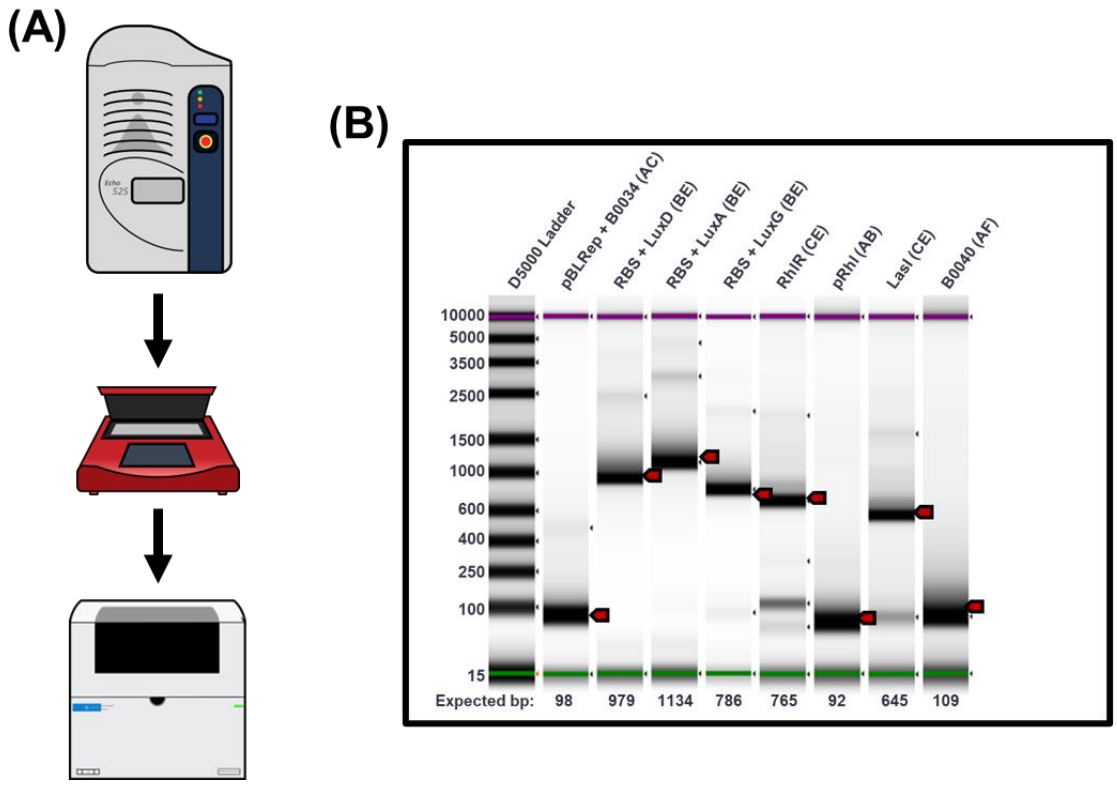


Figure 3.25. EchoProto PCR Template Results

Results from testing the EchoProto loop assembly Template. (A) Workflow employed to test the assembly of level 1 constructs using the loop assembly Template. See section 2.3.1 for full methods. Briefly, (B) DNA gel obtained by the Tapestation 4200 visualising PCR products generated by the Echo 525 using the EchoProto PCR Template. Red arrows show the approximate expected fragment sizes.

polymerase and buffer to use and the amount of DNA to add per reaction, can also be defined by the user, along with the number of reaction repeats. It is also possible to specify that a mastermix should be used to prepare the reactions, rather than adding each reagent (polymerase, buffer, and dNTPs) separately. When not using a mastermix, the proportions at which each reagent is added to the reactions is pre-defined for a 5 µL reaction, which are then scaled based on the final reaction volume supplied. For advanced users, it is possible to modify the reagent proportions by changing the relevant attributes of the Template, before calling the `run` method to create the picklists. When the `run` method is called, the Template first checks that all of the source material required to prepare the PCR reactions are present in the source plates supplied. If not, the user is alerted to any issues and prompted on how to resolve the problem. Otherwise, the Template creates the number of destination plates required to contain the PCR reactions, and then populates the with the correct source material. From this, the transfer events and picklists can be generated as described previously.

To test the PCR Template, eight PCR reactions were performed (Figure 3.25). The low number of reactions allowed downstream processes required to verify the PCR reactions to be performed manually (Figure 3.25 (A)). Performing these verification steps manually allowed for just the PCR Template to be validated, rather than simultaneously testing additional automation protocols in the same workflow. The PCR reactions aimed to amplify up genetic elements from various plasmids and add specific flanking regions (sub-section 2.3.1). These PCR reactions were verified via capillary electrophoresis using the TapeStation 4200, as shown in Figure 3.25 (B). For all PCR reactions, the major band was at roughly the expected size, suggesting that automation of the PCR reactions was successful.

The other EchoProto Template developed was for the preparation of DNA assemblies using the Loop method. Loop DNA assembly is able to assemble DNA parts which adhere to the Phytobrick standard. The Template is able to assemble parts at any level (including the creation of level 0 parts), as users are able to specify which enzyme (typically BsaI or SapI) should be used. The user can also define the type of buffer to be used, the number of DNA assembly repeats to prepare, the final volume of each reaction, the initial concentration of the DNA parts (in fmol/µL), and the backbone to part ratios to use. The ratio of backbone to part can be supplied as a list, in which case each assembly will be prepared using all ratios. Assemblies are specified using the BMS `Assembly` class. An assembly object is instantiated by supplying a name for the assembly, the DNA backbone to use, and a list of DNA parts to add. The source plates and destination plate must also be supplied in the same way as described for the PCR template, and the picklists are generated in a similar way.

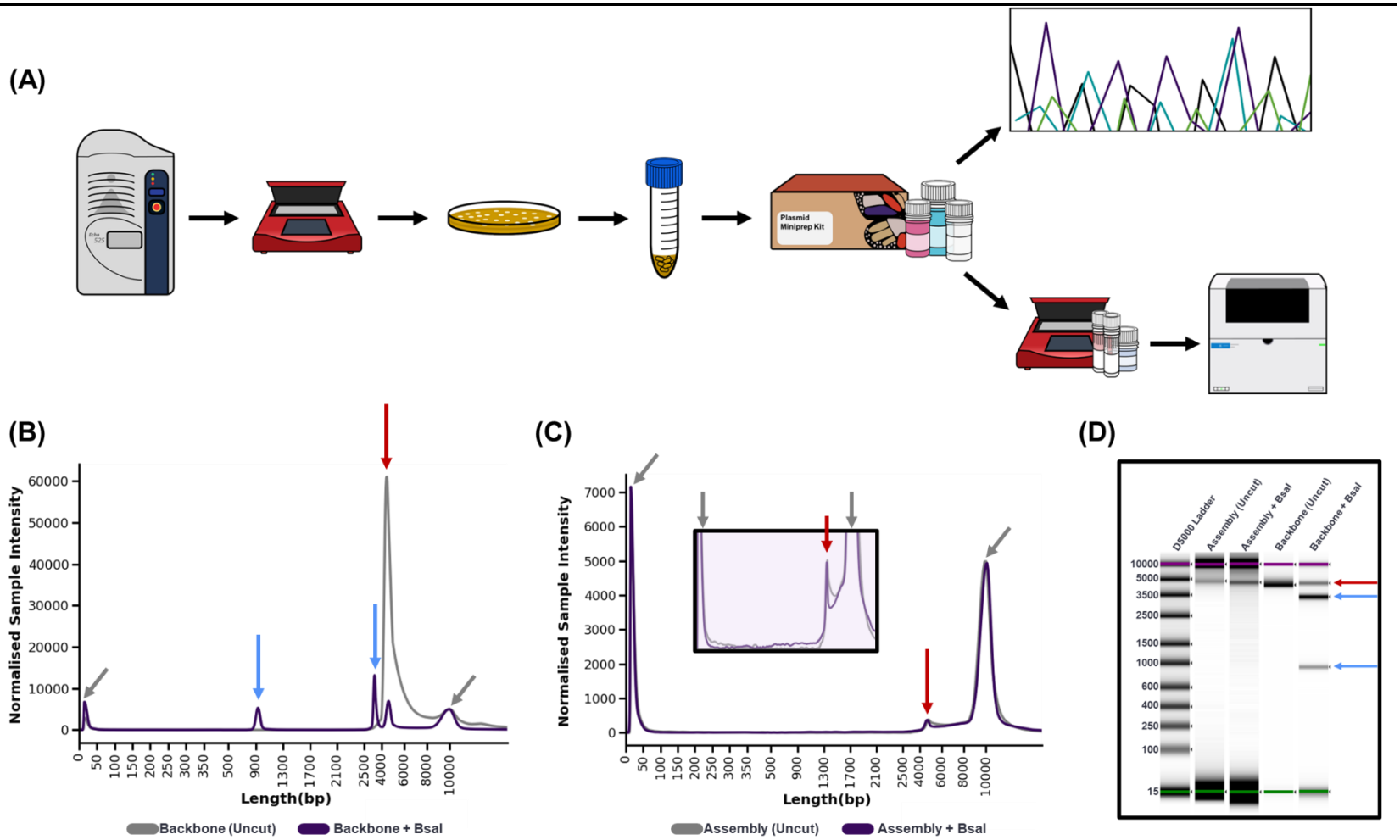


Figure 3.26. EchoProto Loop Assembly Template Results

Results from testing the EchoProto loop assembly Template. (A) Workflow employed to test the assembly of level 1 constructs using the loop assembly Template. See section 2.3.2 for full methods. Briefly, the Echo 525 was used to prepare the assembly reactions, which were then incubated in a thermocycler, transformed into *E. coli* cells, and grown in liquid culture. Plasmids were extracted and screened via confirmation digest with band visualisation by the Tapestation 4200 before confirming

assembly by Sanger sequencing. (B) Electrophogram showing DNA bands from the undigested backbone and the backbone digested with BsaI. The grey arrows show sample markers at 15 bp and 10000 bp. The red arrow shows the undigested backbone. The light blue arrows show the two linear fragments generated by digestion with BsaI, as expected (C) Electrophogram showing DNA bands from one of the undigested level 1 assemblies, and the digested sample. Grey arrows shown the markers, and the red arrow shows the undigested backbone. No linear fragments were observed, as expected. The inset shows a close up of the lower section of the electrophogram. (D) DNA gel obtained by the Tapestation, which was used to generate the electrophograms shown in (B) and (C). The red arrows shows the size for the assembly backbone, and the light blue arrows show sizes for the linear fragments after digestion.

To test the Loop assembly Template, four Level 1 expression units consisting of a promoter, RBS, CDS, and terminator were assembled into either pOdd1 or pOdd2 backbones. To confirm the assemblies, first a confirmation digest with BsaI was performed. During the Level 1 assembly, the two BsaI sites in the plasmid backbones should be removed. Therefore, the assemblies should not be cut when incubated with BsaI, but the backbone should form two linear fragments. The confirmation digests were visualised via capillary electrophoresis performed by the Tapestation 4200. In addition to the BsaI digests, all assemblies were confirmed as correct via Sanger sequencing. Once again, all downstream processing was performed manually (Figure 3.26 (A)). Figure 3.26 (B) shows results for one of the backbones, where the two linear fragments (light blue arrows) can be seen, along with some of the undigested backbone (red arrow) due to the digest reaction not reaching completion. Conversely, the assembly shown in Figure 3.26 (C) showed only one band: that of the undigested backbone. Only one of the four assemblies is shown here for clarity.

3.4.9. BMS Templates for testing Sensynova modules and biosensors

Characterisation of modules and biosensors developed in accordance with the Sensynova framework stands to benefit from bio-automation. To aid in this endeavour, the BMS library was used to develop a characterisation template for Sensynova-compatible biosensors and modules. This template aimed to allow for trivial generation of automation protocols to test a Sensynova module or a multi-microbial biosensor and allow for standardisation in testing procedures. The OT2 was selected for this task as it allowed for a much larger degree of flexibility compared to the Echo 525, allowing for more complex protocols to be performed in a walk-away manner. However, the DNA assembly EchoProto could have use within the Sensynova framework by allowing for automated assembly of module variants, although this was not explored here.

The Sensynova OTProto Template was implemented similarly to the OTProto Templates already described. Additionally, a new Python class, `Cell_Type`, was defined for use in the Template. This cell type class requires instantiation by users, and requires information such as a name, antibiotic selection (if any), media for the cells to be cultured in, and the stock cell density of the culture supplied to the automation equipment. Cell types can be passed to the Sensynova Template as either positive controls, negative controls, or experimental samples, which allowed for easy

understanding of the intended role for each cell type in the experiment. Users can also define other global properties of the experiment, such as the different source materials (antibiotics, media, inducers, etc.) to be used during the experiment along with the volumes of aliquots and their stock concentrations, the types of labware to use for the source materials and the destination labware, and the final volume for each experimental.

Aside from the initialisation and run methods seen previously with OTProto Templates, the Sensynova Template contains a number of additional methods which can be used to define specific experiments. The `add_sample` method is used to define individual experimental samples within the experiment. The types of cells to use, along with their volume per cell, are used to define an individual sample. Information such as the type of media and antibiotic to use are retrieved from the `Cell_Type` objects. Users are also required to specify how many repeats of each experimental sample should be added to the destination labware. If required, the type of inducer to use can also be given, along with the final concentration at which it should be added. For common types of characterisations associated with the Sensynova Framework, such as dose-response or cross-talk experiments, methods are provided which act as wrapper functions to create the required samples, without requiring users to specify them individually. However, the `add_sample` method can still be used to provide flexibility in the types of experiments performed. There are also methods to add control samples, which simply adds the positive and negative cell types specified during Template creation. The controls can be added with or without inducers, and the number of repeats can also be specified independently. Finally, the `add_inducer_controls` method automatically creates controls for all samples containing an inducer, where the inducer is replaced with the solvent in which it is dissolved.

Aside from defining the experimental samples, other methods can be used to modify the experimental setup. The use of mastermixes can be used to help decrease variation across a protocol by ensuring that the source material for replicates is pre-mixed and then dispensed into the required wells, rather than adding the source material to each repeat well separately. Mastermixes can also be used to help increase pipetting accuracy by ensuring that liquid is not transferred at lower volumes. Mastermix generation uses the ‘mastermixes_by_replicates’ function described previously.

It is also possible to ensure samples with different concentrations of an inducer are added at the same volume. This can be particularly important when an inducer is dissolved in a solvent which may have an impact on cell behaviour, such as toxicity. The `normalise_inducer_volumes` method can be used to prepare uninform or non-uniform serial dilutions of an inducer stock, allowing the same volume of inducer to be added to the samples, no matter the final concentration. The volume at which the inducer will be added to each sample is user-defined.

As with all OTProto Templates, the `run` method is used to generate the liquid handling instructions based on the specified information. Error checking is used throughout the Template to help prevent oversights or mistakes. This error checking goes beyond that which is included with the native Opentrons API, as BMS is able to track information about the experiment, rather than just checking for physical impossibilities (like trying to load two labware onto the same deck slot). For example, the Template will give warnings if a source material has been defined but never used, and errors are raised when cells with different antibiotic selection markers or media requirements are co-cultured.

The Template described here was used in chapters 5 and 6 to characterise Sensynova modules as well as modular and multi-microbial biosensors. Although the Sensynova Template was developed specifically for this purpose, it was found to have a large degree of flexibility and could be used to characterise different synthetic biology systems. An example of this can be seen in chapter 7, where a bioluminescent construct was characterised using the Sensynova Template. These use cases helped validate the applicability of the BMS library to a variety of applications within synthetic biology by providing a flexible language with which to develop automation protocols and Templates.

3.5. Conclusions, Limitations, and Next Steps

This chapter began by exploring how the concept of high-level modularity could aid in the development of synthetic biology systems. It was also identified that multi-microbial systems provided an ideal method for implementing modular designs. Further, a framework for developing genetic biosensors, termed the Sensynova framework, was presented which leveraged modularity and multi-microbial systems. Based on work described by the Newcastle iGEM 2017 team, this framework also builds upon previous research discussed in section 3.1[120], [194]–[196]. These previous efforts showed not only the potential for high-level modular synthetic biology, but also how multi-microbial systems could be used in conjunction with such modular approaches. However, for reasons explained previously, research up until now tended to (i) focus on either very specific applications, (ii) require complex and custom equipment, or (iii) not provide guidelines for module development or implementation. By taking examples from other fields, the Sensynova framework differed from previous work by focusing on a wide-reaching but specific type of biological system, genetic biosensors, and providing standard guidelines for module development to ensure flexibility and re-usability of modules between various projects.

The Sensynova framework presented here extends the version documented by the Newcastle iGEM 2017 team in several ways. Discussed in this chapter was the development of resources for use within the Sensynova framework. Firstly, the Synthetic Biology Open Language (SBOL) was extended to allow for representation of cells and other chassis, and a set of best practices were proposed to allow for representation of multi-microbial systems. This extension provided a method of representing and sharing Sensynova biosensor designs using a standard format. Although methods for representing such systems existed before the work presented here[251], they did not have the uptake or reach of the SBOL data standard in the field of synthetic biology, as evidenced by the variety of software tools which are SBOL compatible[76], [252]–[261]. A limitation of the work provided here is that there currently do not exist tools which can leverage the newly-added capability to develop and share multi-microbial designs. Therefore, future work should focus on developing tooling towards this application, similar to the pre-existing tools for genetic designs[76], [262]. Without this tooling, the ability to capture multi-microbial systems in SBOL remains limited to data scientists and programmers familiar with the low-level SBOL libraries.

Nevertheless, the proposals discussed here, which were subsequently accepted by the SBOL community, allowed for a greater range of systems to be captured by SBOL than was previously possible.

In addition to extension of the SBOL data model, a Python library, BiomationScripter (BMS) was developed to help automate characterisation of Sensynova modules and biosensors. As discussed previously, other research in the area of bio-automation protocol generation provided only an alternate method of programming protocols and did not have the capacity for BMS Template-like functionality[230], [239], or only developed a method of automatically generating protocols for very specific workflows[223], [250]. Thus, the BMS library is novel in combining these two approaches to not only develop a flexible method for intermediate Python programmers to script complex protocols, but to provide protocol Templates for rapidly and trivial generation of protocols for synthetic biology procedures and allow for further Template development by users. Next steps for development of BiomationScripter should focus on optimisation of the developed Templates in terms of liquid transfer parameters and reaction component proportions, where applicable. Such optimisation would be possible due to the in-built ability to easily pass optional parameters to BMS Templates. Such optimisation data would allow users to make informed choices when selecting these optional parameters, help increase successful implementation of automation workflows, and potentially aid with user uptake due to confidence that the developed Templates are high-quality.

Chapter 4. Design and Computational Modelling of a Modular and Multi-Microbial Biosensor

In chapter 3, a modular and multi-microbial framework for assisting biosensor development and optimisation was described. In order to investigate this framework's potential, a proof-of-concept biosensor was developed. This chapter details the initial design stage of the biosensor. The biosensor's specification and modular designs are presented in section 4.2, along with a brief overview of how each module was built and implemented. To help inform experimental characterisation, computational modelling was first performed. Each module was modelled deterministically (section 4.3), and also modelled using an agent-based approach (section 4.4). The agent-based models were combined to simulate the biosensor co-culture. Section 4.5 gives an overview of the insights gained from the simulation results which were used to inform the experimental characterisation and validation presented in chapter 5.

4.1. Introduction

4.1.1. Computational modelling of biological systems

Synthetic biology devices and systems can be simulated using computational models to assist with (i) the system's design[76], (ii) experimentally testing the system[263], (iii) performing optimisation[89], [264], or (iv) a mixture of all three[265]. Modelling a system can allow for a more informed approach towards development by providing insight to a system's behaviour and identifying fundamental issues with the design prior to expending time and resources building and testing the system. Simulating a system can also be used to investigate targets for optimisation by investigating aspects which have the most impact on the system's desired function.

Biological systems can be described using a series of stoichiometric equations, which detail interactions and reactions occurring within that system. Stoichiometric equations simply formalise entities which are involved in some reaction. There are a variety of approaches and algorithms which may be used to simulate models described by such stoichiometry[266]. One such approach, often referred to as deterministic simulation, in which ordinary differential equations (ODEs) are generated based on the stoichiometric reactions and their kinetic. The ODEs can be used to calculate, according to the law of mass action, how each entity varies over time as reactions

progress[267], [268]. Thus, by using the amount of each entity at the current time point as inputs for the ODEs, the state of the system can be determined after a set time period.

Modelling processes such as transcription, translation, molecular binding, and enzymatic reactions allows deterministic models to predict how a system might react to the introduction or removal of specific biological or chemical entities (such as molecules used as inducers, or DNA encoding specific proteins)[269]. These models have also been employed to help inform on the design of genetic constructs used in synthetic biology systems by predicting how the behaviour of the system changes with increased or decreased transcription and translation (which can represent different strengths of promoters and ribosome binding sites), or other processes like protein degradation which can be modified via the genetic design, such as through the addition or removal of degradation tags[270].

Whilst a deterministic view of biological systems can be helpful, it is not able to meaningfully represent biological noise[271]. Random noise is a constant within biology, and in some cases is even integral to the correct functioning of a biological mechanism[272]. Therefore, it can be necessary to include this feature. An example of this is in gene expression, where the binding of transcription factors to regions within a promoter is subject to random chance[273], [274]. To account for random noise, stochastic models may be used rather than deterministic[272]. Stochastic models can be built in a similar way to deterministic models (by formalising the system via a series of reactions), however a different set of methods and algorithms are required for simulation[275]. These methods make use of statistical probability to introduce randomness into the model, and hence each run of the simulation will yield different results each time. Therefore, unlike deterministic models which will always give the same set of results when all inputs and parameters are the same, it is usually necessary to run stochastic simulations multiple times to determine the general pattern of behaviour for a given set of conditions. Additionally, stochastic simulations can be more computationally complex and expensive than deterministic models[276], [277]. The advantages of stochastic simulations come from the ability to account for noise, which also allows for them to more accurately represent systems containing entities present in low quantities. This is because when low numbers of an entity are present, randomness involved in certain processes becomes more important. For example, the binding of two molecules requires them to first come into contact with one another – a

situation which becomes more subject to random probability when the molecules are present in low numbers within a large space[278]. Therefore, despite their computational complexity, stochastic models are becoming more common within this field.

4.1.2. Systems Biology Markup Language (SBML)

To help capture information required for modelling of a biological system, a standard data format termed SBML (Systems Biology Markup Language) can be used[279]. SBML formalises information about a system in a machine-readable format and is commonly encoded using extensible markup language (XML). SBML can be used to represent a biological model by capturing information about reactions, such as the species involved and their roles, the mathematical equations which describes the reaction, and the values of any parameters used, as well as the species present in the system, their starting amounts, and the units for any values. Other information such as the presence of compartments can also be captured. There are no specifications relating to how a system should be simulated, allowing for applicability to a wide range of model types and simulators.

There are a variety of software tools and programming libraries which have been developed with SBML compatibility in mind, allowing users to easily define a biological system for modelling, and perform simulations of that system. For example, libSBML is a C++ library with APIs (Application Programming Interfaces) for other languages (such as Python and Java) which can be used to create and edit models encoded in SBML[280]. Similarly, the LibSBMLSim library can be used to simulate SBML models[281]. COPASI (a COmplex PAthway SImulator) is an SBML-compatible software tool which provides a user interface for creating, editing, and simulating biological models using a variety of deterministic and stochastic simulators[282].

4.1.3. Modelling microbial communities

For modelling microbial communities, agent-based modelling, sometimes referred to as individual-based models, are an appropriate option as each microbe in the system can be represented as its own agent[283]–[285]. Each agent can then be modelled separately within the overall simulation, rather than assuming homogeneity across all microbes. This is important when it comes to studying microbial communities as it is well known that heterogeneity, where microbes exhibit different behaviours, is common[286], [287]. As discussed in section 1.2.6, heterogeneity can even be observed in microbes of the same type, depending on the interactions occurring with other

nearby members and the state of the local environment[288]. Considering the immediate surroundings of each individual microbes is also crucial when modelling communication between individuals, as this communication most often relies on the diffusion of small chemicals, and hence the exposure of each microbe to the signalling chemical depends on their position within the diffusion gradient[289]. ABMs are also able to simulate mechanical interactions and forces which are important factors when modelling communities with distinct structures such as biofilms[290]. In these types of systems, affects like fluid flow and shear force have an impact on the community's morphology. The use of agent-based modelling also enables graphical simulations in 2- or 3-dimensional space, where the general positions of microbes and other entities, such as extracellular chemicals or proteins, within the system can be predicted. This allows for predictions relating to the shape and structure of communities which do not exist in a homogenous mixture[291].

There are a number of ABM simulators and software tools which have been developed for modelling microbial communities. One such example is gro, which is a specification language for defining and simulating a multi-microbial community[291]. The gro tool allows for specification of different cell types along with information about their growth rates. The internal reactions occurring within the cells can also be defined using rule-based modelling, and cell-to-cell signalling can be modelling by defining signal emission and reception for each cell type. The downside of gro is that systems can only be modelled in 2-dimensions, which means that systems which exist in 3-dimensions, such as those grown in liquid culture, cannot be accurately modelled. The NUFEB simulator is another example of an ABM tool, which allows for modelling of communities in 3-dimensions[292]. There is also extensive support for modelling nutrient-limited growth and essential features of biofilm formation, such as Extracellular Polymeric Substances (EPS), however there is no option to model other cellular mechanisms such as the expression of genes or cell-to-cell signalling.

Simbiotics is a platform implemented in Java which allows for simulation of microbial communities in either 2- or 3-dimensions[77]. There is support for a range of cellular processes, including growth and transport of chemicals across membranes. There is also the option to include more complex internal cellular mechanisms such as the expression of genetic circuits using SBML models, which can be attached to the different cell types in the system and solved individually at each time step. These SBML

models can be simulated using solvers provided with the libsbmlsim library, which provides a variety of methods for solving ODEs. The ability to implement cellular behaviour using SBML files enables a modular approach to development of a model for microbial communities, as cell types can be easily swapped by changing the SBML file provided.

4.2. Design, Assembly, and Implementation of a Proof-Of-Concept Biosensor

4.2.1. Biosensor specification

To determine the feasibility of developing genetic biosensors using the Sensynova framework, a proof-of-concept biosensor was taken as a case study. The aims of this feasibility study were to **(i)** demonstrate how a biosensor can be split into three functional modules Sensynova framework's design principles, **(ii)** determine whether these modules can be combined via co-culturing to form a biosensor with desired functionality, and **(iii)** explore potential avenues for optimisation. For the purposes of testing the framework, it was decided that this biosensor should be relatively simple in terms of the response characteristics and should be constructed of well characterised parts. This allowed more time and resources to be directed towards testing and development of the framework, rather than development of a complex and novel biological device. To this end, the following design parameters were used to define the biosensor specification:

- **i.** The biosensor should respond to the presence of the small molecule IPTG (Isopropyl β-D-thiogalactoside)
- **ii.** The biosensor's response should generally scale with the concentration of IPTG present
- **iii.** The response should be well defined, easily detectable, and able to generate both quantitative and qualitative data
- **iv.** The biosensor should make use of well-defined genetic parts

Using these design parameters, abstract functionalities for the three module types defined by the Sensynova framework (detector, processor, and reporter) were determined. Broadly, these abstract functionalities were as follows. The Detector module should convert the presence of IPTG into a genetic signal. This module was termed the IPTG Detector Module. As no specific response characteristic was specified by the design parameters, the Processor module needed to simply pass the signal from the Detector module to the Reporter module. This was termed the Default Processor Module. Finally, for the response, production of a green fluorescent protein (GFP) was chosen. This is because fluorescent proteins can provide quantitative data by using a fluorometer to measure the fluorescent intensity of a sample, and many GFPs are also visible to the naked eye or under ultraviolet (UV) light, producing a green colour.

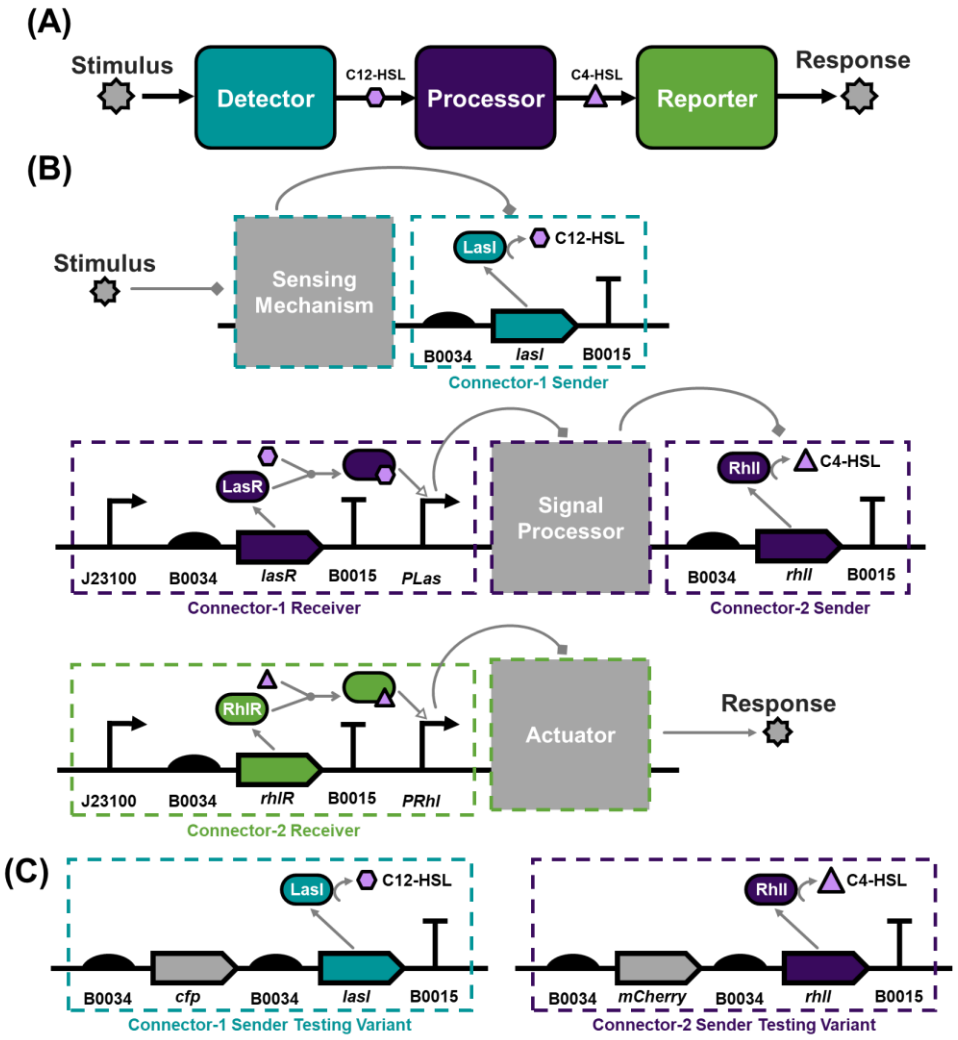


Figure 4.1. Module Interface Designs

Genetic diagrams depicting interfaces between the three module types. **(A)** Abstract overview of signal propagation. C12-HSL carries the signal from the detector to the processor, and C4-HSL from the processor to reporter. **(B)** Top: detector module interface. A sensing mechanism regulates the Connector-1 sender sub-module, which is responsible for expression of *lasI*, and hence production of C12-HSL. Middle: processor module interfaces. Connector-1 receiver sub-module activates the signal processor in the presence of C12-HSL. The signal processor regulates the Connector-2 sender, which encodes *rhlI*. Bottom: reporter module interface. The Connector-2 receiver sub-module activates an actuator in the presence of C4-HSL. **(C)** Alternative interface sub-modules for the connector-1 sender and connector-2 sender. These variants co-express a fluorescent protein (eCFP for connector-1 sender and mCherry for connector-2 sender) to allow for easier measurement of activation.

Aside from these abstract functionalities, each module also needed to be compatible with the standard interfaces described in Chapter 3, namely that the Detector module should produce quorum sensing molecule C12-HSL, the Processor module should respond to the presence of C12-HSL and produce the molecule C4-HSL, and the Reporter module should respond to C4-HSL. As discussed previously, this functionality allows for uni-directional cell-to-cell communication and the propagation of a signal from the Detector module to the Processor module to the Reporter module.

Finally, it was decided that the biosensor modules should be implemented within *Escherichia coli* cells. This was because the use of well-defined parts was specified, and many of the most commonly used and characterised parts were designed for and tested in *E. coli* cells[82].

4.2.2. Considerations for designing the high-level modules

Here, the genetics for the three biosensor modules defined above are presented. For all constructs described here, genetic parts known to work in *E. coli* cells were used. A full list of all part names, sources, and sequences can be found in the materials and methods section. It should be noted that whilst a high-level modular framework such as the one described in this work can eventually promote top-down design, initial development of the individual modules must necessarily be bottom-up. This is because there are currently no suitable higher-level modules to use, and therefore individual genetic parts must be used instead. It is only once compatible modules have been developed that top-down design is possible.

As described previously, the interfaces between each module in the Sensynova framework were implemented as quorum sensing mechanisms (Figure 4.1 (A)). Each interface (detector-processor and processor-reporter) was split into two sub-modules: a sender and a receiver (Figure 4.1 (B)). The first interface (detector-processor) was facilitated by LasIR quorum sensing, where the small diffusible molecule C12-HSL was sent from the detector module and received by the processor module. The second interface (processor-reporter) used the RhlIR mechanism, where the quorum sensing molecule C4-HSL was sent from the processor and received by the reporter. The genetic designs for these senders and receivers are shown in Figure 4.3. These sender and receiver designs were taken from those presented in the Newcastle iGEM 2017 project, which ensured compatibility.

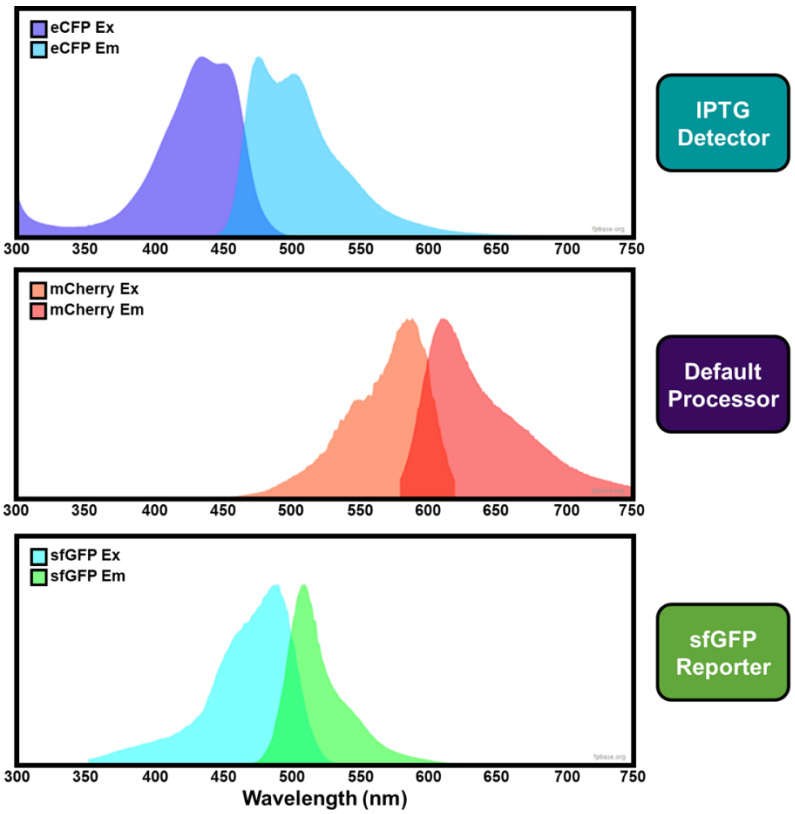


Figure 4.2. Fluorescent Protein Markers' Spectral Profiles
Spectral profiles for eCFP, mCherry, and sfGFP. Spectra were obtained from FPBase[293].

As one of the aims for this work was to characterise all three biosensor modules independently, it was important to ensure that the modules were designed so as to facilitate appropriate measurement of functionality. A common method of measuring a device or system's behaviour within synthetic biology is to use fluorescent markers, which are co-expressed along with aspects of the system which require measurement. Fluorescent proteins tend to be used as their presence can be easily quantified using a range of laboratory equipment, such as fluorescent microplate readers. Additionally, there are a large range of fluorescent proteins which have been validated within organisms commonly used in synthetic biology research[294]. For these reasons, it was decided that fluorescent proteins would be used as markers to measure module activity. For the reporter module, no modifications to the design were required as the biosensor specification already requires the output of this module to be a fluorescent signal. For the detector and processor modules, the sender sub-modules were modified to contain a fluorescent protein coding sequence (with ribosome binding site) immediately upstream from the AHL synthetase coding sequence (*lasI* for the detector module and *rhlI* for the processor module). This positioning meant that the fluorescent proteins markers should be co-transcribed with the AHL synthetases, which in turn should only

be expressed when the modules have been activated. Therefore, presence of the fluorescent proteins, and hence a fluorescent signal, could be used to determine activation levels for the modules.

The fluorescent protein markers have applicability not only in characterising the modules individually, but also in measuring activity of each module within a co-culture. It was therefore important to consider the excitation and emission spectra of the fluorescent proteins to ensure that minimal overlap exists. If significant overlap were to exist within the emission and/or excitation profiles of the fluorescent proteins, it would be difficult to differentiate from which modules the signal was generated. It was decided that cyan, red, and green fluorescent proteins would be used, as these colours are separated across the colour spectrum. The specific fluorescent proteins chosen were eCFP (enhanced cyan fluorescent protein), mCherry (a red/pink fluorescent protein), and sfGFP (superfolder green fluorescent protein). The spectral profiles for these proteins can be seen in Figure 4.2. Whilst the mCherry emission and excitation spectra have no overlap with eCFP or sfGFP, the two other proteins do have some overlap. This overlap, however, could be mitigated through careful selection of the wavelengths chosen for measurement. Here, eCFP expression was measured using an emission wavelength of 480 nm and sfGFP expression with an emission of 515 nm, both of which have little overlap with the emission profile of the other fluorescent protein. Therefore, it could be assumed that the majority of each fluorescent signal measured came from the expected protein.

It was decided to use expression of the sfGFP protein as the output of the reporter module, as the protein exhibits high stability, and a large dynamic range and signal-to-noise ratio[295]. Therefore, use of sfGFP as a reporter should allow for more sensitivity when measuring functionality of the multi-microbial biosensor. It was then decided that the processor module should use mCherry as the fluorescent marker, and the detector module should use eCFP.

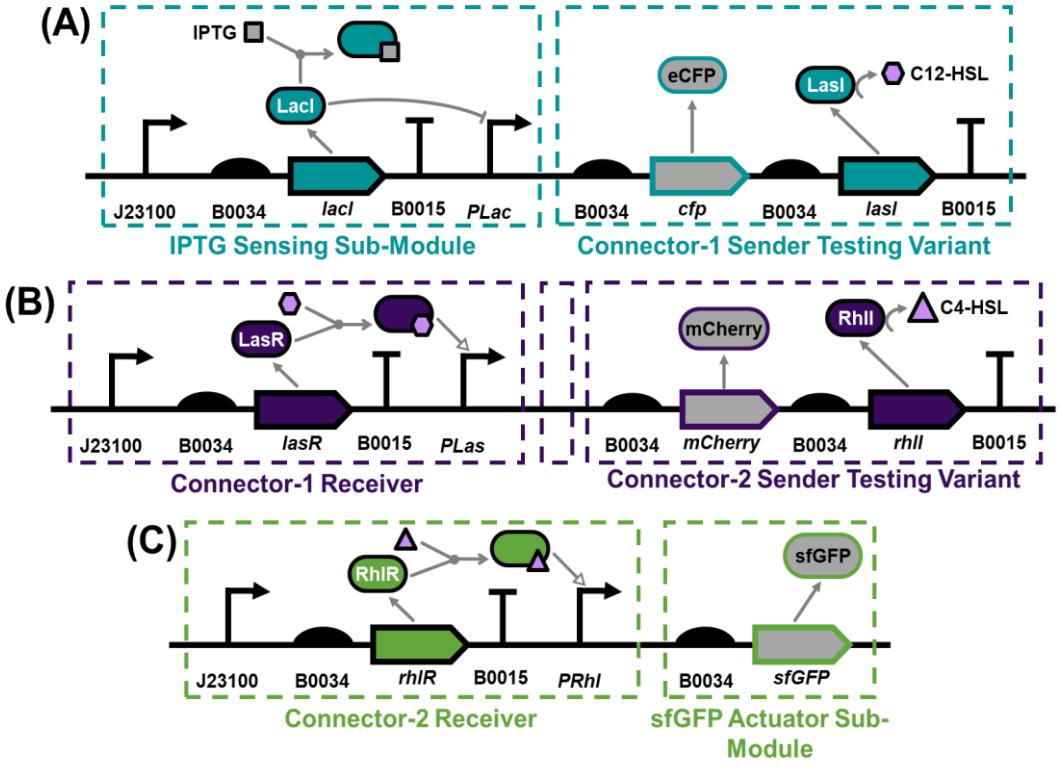


Figure 4.3. Biosensor Module Designs
Schematics depicting genetic designs for the three proof-of-concept biosensor modules. Abstracted functionality and interactions are shown. **(A)** Genetic design for the IPTG detector module. Consists of the IPTG sensing sub-module and Connector-1 sender testing variant. **(B)** Genetic design for the default processor module. Consists of the Connector-1 receiver sub-module and the Connector-2 sender testing variant. No processing sub-module is included in this design (shown as an empty box between the receiver and sender sub-modules). **(C)** Genetic design for the sfGFP reporter module. Consists of the Connector-2 receiver and sfGFP actuator sub-modules.

4.2.3. IPTG detector module design

An IPTG sensing sub-module was designed to be used in the IPTG detector module. The sensing sub-module design utilised the well-known and well-utilised Lac system to convert the presence of IPTG into a genetic signal[104], [296]–[298]. In the Lac system, a transcription factor (LacI) represses the *PLac* promoter. However, IPTG can bind and sequester LacI from the promoter. Therefore, in the absence of IPTG expression from *PLac* is repressed, and in the presence of IPTG expression is permitted. The IPTG detector module design was completed by adding the connector-1 sender sub-module upstream of the IPTG sensing component. This design meant that *PLac* was positioned upstream of the *lasI* coding sequence, and hence allowed for regulation of LasI production (and consequently C12-HSL synthesis) by IPTG. The IPTG sensing sub-module was designed to be assembled immediately upstream of the connector-1 testing variant sub-module, as shown in Figure 4.3 (A).

4.2.4. Default processor module design

The default processor module was intended to have no prescribed function other than accepting the signal from the detector and passing it to the reporter. Therefore, the default processor module's design consisted simply of the connector-1 receiver sub-module upstream of the connector-2 sender testing variant sub-module (Figure 4.3B). For processor variants, a design could be added between these two sub-modules with functionality such as signal amplification.

4.2.5. sfGFP reporter module design

For the reporter module, an actuator sub-module capable of generating the desired signal (in this case fluorescence) was designed. Here, this actuator sub-module consisted of a coding sequence for superfolder GFP (sfGFP) flanked by a ribosome binding site and terminator. This sfGFP actuator sub-module was then combined with the connector-2 receiver to form the complete sfGFP reporter module (Figure 4.3C).

4.2.6. Overview of assembly strategies

The genetic designs for the processor module shown in Figure 4.3 was assembled in stages using Biobrick assembly, as described in section 2.2[299].. The reporter module construct was previously constructed and required no further modification (section 2.2, Table 2.2). Initially, Biobrick assembly was also to build the IPTG detector module, however the final stage of assembling the IPTG sensor sub-module and connector-1 sender testing variant sub-module repeatedly failed to yield plasmids containing the correct construct. Instead, the IPTG detector module was obtained via third-party synthesis (by ATUM) directly into the *pSB1C3* plasmid (section 2.2).

Once each module had been obtained and sequence verified, the plasmids were transformed into *E. coli* DH5α cells. Following transformation, the cells were termed IPTG detector cells, default processor cells, or sfGFP reporter cells, depending on the type of module each cell type contained.

4.3. Deterministic modelling of biosensor modules

Prior to experimental characterisation of the modules described above, and the multi-microbial biosensor as a whole, computational modelling was used to help predict behaviour and guide the experiments performed. To achieve this, an SBML model was developed for each module (IPTG detector, default processor, and sfGFP reporter) Initially, the models were simulated using stochastic algorithms, namely the Gillespie stochastic algorithm[300]. However, the stochastic simulations showed a large degree of numerical instability and often resulted in internal time step limit errors. These issues were likely caused by the large number of some entities which accumulated during simulation, which are not handled well by stochastic algorithms. Subsequently, the SBML models were simulated deterministically.

4.3.1. General model assumptions

For each biosensor module, a deterministic model was created in SBML using COPASI[282] (version 4.36, build 260), and simulated using the basico[301] python library (version 0.3.0). The models were defined by a series of reactions which assumed mass action rate kinetics and simulated deterministically. There were a few universal assumptions made for all models. The first was that the required cellular resources for the modelled processes, such as ribosomes, polymerases, and AHL synthetase substrates, were present in excess. This allowed for processes like transcription and translation to be abstracted to a single reaction, without modelling the presence of cellular machinery, nucleotides, amino acids, or other entities. The assumption of resource abundance, whilst not necessarily true, was made as it was more likely in the biosensor module systems that other entities, such as the inducers/signalling molecules, would be the limiting factors, and allowed for a simplified model which required less computation resources to simulate. Another assumption made was that all DNA elements comprising the biosensor modules were present at an initial starting amount of 200. This number was selected based on the reported copy number of pSB1C3 (100-300), which was the plasmid used to contain the genetic system[302], [303]. The final assumption made here was that of homogeneity across the system, where each cell of the same type had exposure to identical conditions as other cells and behaved in the same way. As discussed in section 4.1, this is not always necessarily true, however the assumption allowed for an initial insight into the functionality of the biosensor modules. The models could then be used to build an agent-based model to better account for heterogeneity in the system, as will be presented in section 4.4.

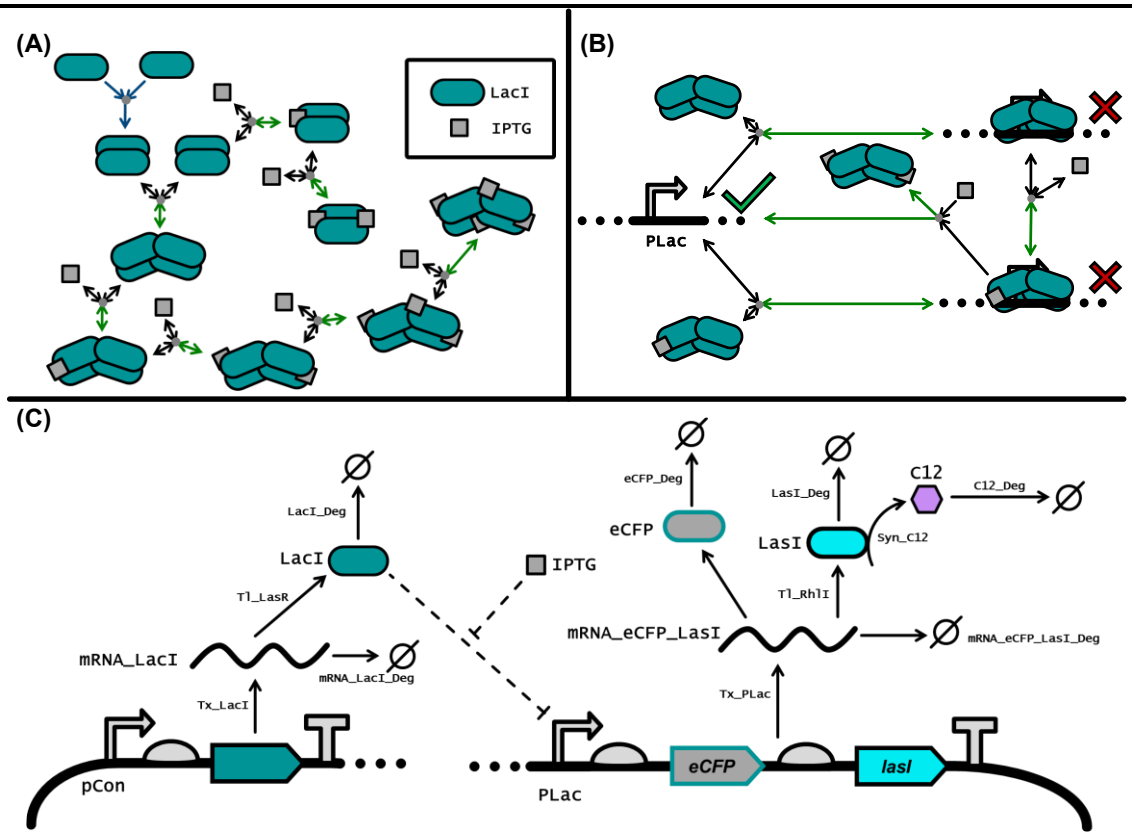


Figure 4.4. IPTG Detector Module Schematics
Depictions of reactions and interactions in the IPTG detector module SBML model. **(A)** LacI complex formation and IPTG binding. Grey circles (●) are used to indicate association/dissociation reactions. Green double-sided arrows (↔) show products of association, black double-sided arrows (↔) show products of dissociation, and blue arrows (→) show binding reactions assumed to not dissociate. **(B)** LacI interaction with *PLac* promoter. Red crosses (✗) indicate blocked transcription. Green ticks (✓) indicate permitted transcription. **(C)** Schematic depicting the IPTG detector module SBML model. Entity names are shown in larger font size, and reaction names in smaller font size. Black arrows (→) show irreversible reactions and capped arrows (⊣) show repressive interactions. Dashed black lines represent abstracted interactions shown in (A) and (B). The empty set symbol (∅) is used as the product of degradation reactions. A comprehensive list of reactions and their parameters can be found in section 2.4.

The parameters for the deterministic models were obtained through a mixture of literature and estimation based on previously reported behaviour of similar systems. The specifics for each model are described in the following sub-sections, along with discussion of simulation results.

4.3.2. IPTG Detector Module

For the Lac repressor system employed by the IPTG detector module, there is a general consensus regarding its mechanism. It is thought that the LacI transcription factor forms a dimer of dimers, which permits binding to operator sequences in the *PLac* promoter region[304], [305]. Binding of LacI prevents polymerases from accessing the *PLac* promoter, and hence prevents transcriptional initiation[306]. For the IPTG

detector module model presented here, the formation of the dimer of dimers was modelled as a two-stage process, where each dimer first formed separately, before binding together to form the two-dimer complex. A common assumption for the formation of the first dimers is that this process is very energetically favourable, and thus does not tend to be rate limiting[170]. Additionally, the reverse reaction of the dimer into two separate proteins is generally assumed to happen so infrequently that it is frequently removed from models[171]. This approach was also taken here. The formation of the dimer-dimer complex is not always considered when modelling the LacI system, and as LacI is usually observed already in its tetrameric state, there were no readily available rates to use from previous studies for this reaction[307]. In the IPTG detector module model, the assumption was made that dimer-dimer binding was as energetically favourable as the formation of LacI dimers, although the collapse of the complex into two separate dimers was included, but at a slow rate.

The small molecule IPTG is known to bind to LacI and sequester it from binding to the DNA operator[308]. For the purposes of this model, it was assumed that a LacI complex with one molecule of IPTG would have reduced binding to the DNA operon, but any extra would prevent binding completely. This assumption was made based on the mechanism of LacI binding, where each dimer in the complex is thought to bind to a separate region in the *PLac* promoter[309]. Thus, whilst one molecule of IPTG might allow for the dimer without any IPTG to still bind, and in the process aid binding of the other dimer due to close proximity with the DNA, two bound molecules of IPTG would make the binding kinetics unlikely enough as to be ignored.

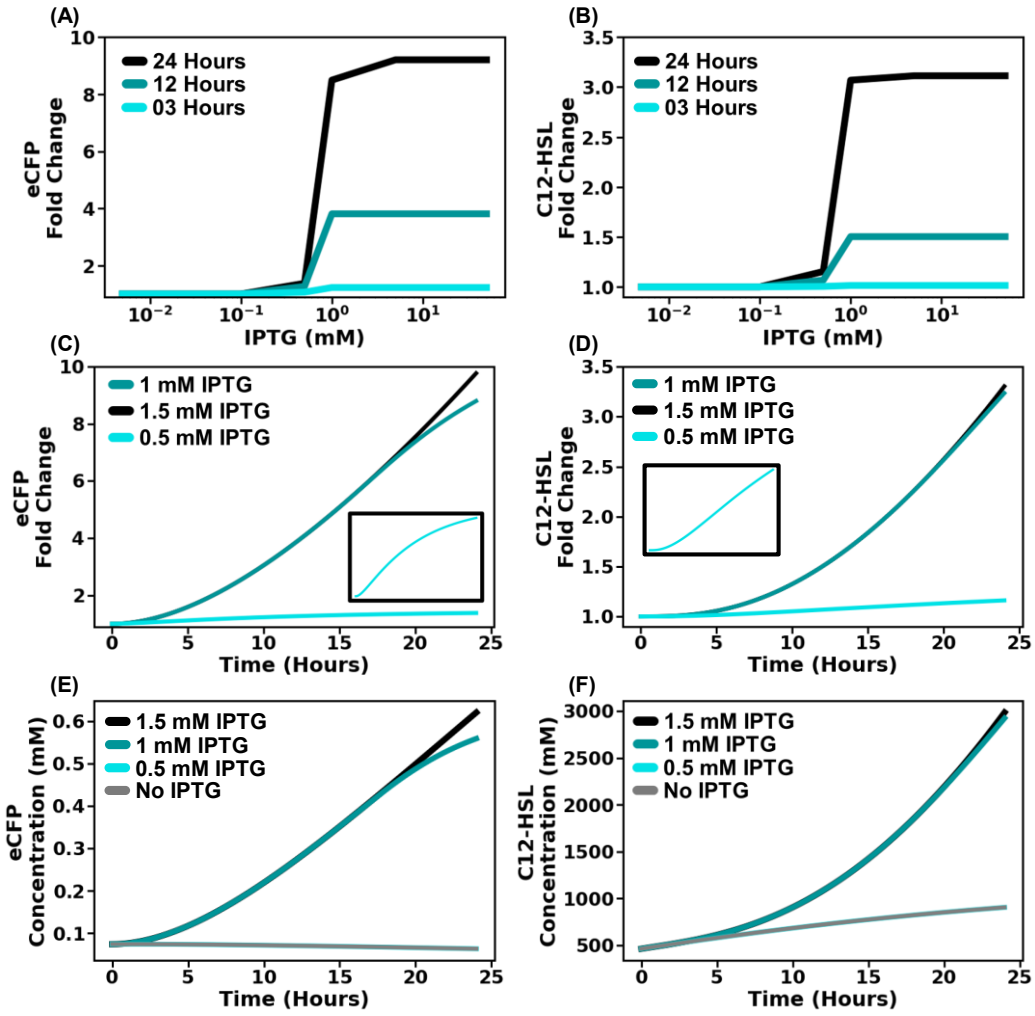


Figure 4.5. Deterministically Simulated IPTG Detector Module Behaviour
Results from deterministic simulation of the IPTG detector module SBML model depicted in Figure 4.4. Full details regarding simulation can be found in section 2.4. **(A-B)** Dose-response curves with concentrations of IPTG between 0.005 and 50 mM IPTG. Responses were measured as fold change in either eCFP (A) or C12-HSL (B) production compared to no IPTG addition after 3, 12, and 24 hours. **(C-D)** Time course curves over 24 hours for the IPTG detector module induced with either 0.5, 1.0, or 1.5 mM IPTG. The dependent variable was fold change in eCFP (C) or C12-HSL (D) production compared to the system with no IPTG added. Insets show curves for induction with 0.5 mM at increased resolution. **(E-F)** Time course curves over 24 hours when induced with 1.5, 1.0, 0.5, or 0.0 mM IPTG. Dependent variable was concentration of eCFP (E) or C12-HSL (F) in mM.

It is thought that binding of IPTG to LacI-DNA complexes can occur to cause de-repression of the *PLac* promoter[310]. In this model, one molecule of IPTG can bind to a LacI-*PLac* complex, but if another IPTG molecule binds the complex then *PLac* is freed and an unbound LacI dimer of dimers with two IPTG molecules bound forms. The LacI binding mechanisms used in this model are illustrated in Figure 4.4 (A-B). Figure 4.4 (C) illustrates the remainder of the IPTG detector module model with the LacI binding kinetics abstracted for clarity. The genetic expression of the relevant proteins (LacI, LasI, and eCFP) are modelled, along with the synthesis of C12-HSL by

LasI, and degradation of all entities other than the plasmid DNA molecules, which are assumed to remain consistent throughout.

The IPTG detector module model was first simulated for 24 hours with all starting entities at an amount of 0, except for the DNA components which began at 200. This allowed the system to stabilise and provide initial starting amounts for the other entities. Experimentally, this step correlated with culturing of the cell types in liquid media overnight prior to performing experiments. Following initialisation, the model was simulated for a further 24 hours in the presence of a range of IPTG concentrations, including no IPTG. The concentrations used were based on data from previous studies involving similar systems[311]–[313].

Simulation results were used to predict the detector's dose-response curve, which allowed an informed choice of inducer concentrations to use in experiments. Dose-response curves for fold changes in eCFP and C12-HSL production are shown in Figure 4.5 (A-B). eCFP and C12-HSL were chosen as responses for the curves as the former could be measured experimentally via fluorescence intensity (and hence allowed for comparison of simulated and experimental data), whilst the latter was essential for propagation of the signal to processor cells. It was also important to determine the relationship between the two entities to ensure that eCFP measurement was indicative of C12-HSL production. The simulations suggested that whilst the dose-response curves for both entities display almost identical shapes, the fold change magnitude of C12-HSL is roughly a third of that seen with eCFP production. This can also be seen in the time course plots (Figure 4.5 (C-D)), where once again the time course curve shapes are similar, but differences are less pronounced with C12-HSL. The predicted fold change in production of C12-HSL was lower than that of eCFP, however the model also suggested that the overall concentration of C12-HSL present in the system would be ~5000 fold higher (Figure 4.5 (E-F)). These values should be treated with caution, as the model had not yet been experimentally validated, and the concentration of C12-HSL predicted (up to 3 mol/L^{-1}) seemed exceptionally high. Nevertheless, the prediction of a higher C12-HSL concentration than eCFP was not unexpected, as multiple AHL molecules could be produced by a single enzyme (LasI), which was present in similar quantities to that of eCFP. Overall, these results suggested that the IPTG detector module should function as expected, and also that measurement of eCFP could be used as a proxy for module activity.

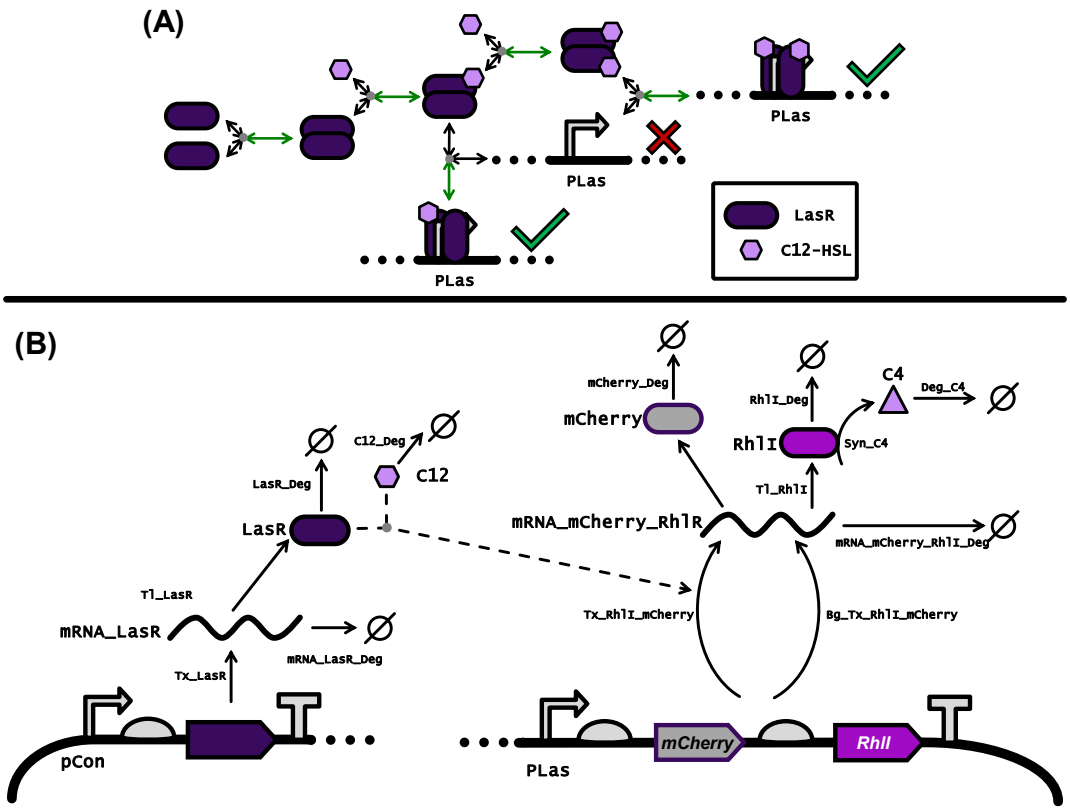


Figure 4.6. Default Processor Model Schematics

Depictions of reactions and interactions in the default processor module SBML model. **(A)** LasR complex formation, C12-HSL binding, and interaction with *PLas* promoter. Grey circles (●) are used to indicate association/dissociation reactions. Green double-sided arrows (↔) show products of association, black double-sided arrows (↔) show products of dissociation. Red crosses (✗) indicate unfavourable transcription. Green ticks (✓) indicate favourable transcription. **(B)** Schematic depicting the default processor module SBML model. Entity names are shown in larger font size, and reaction names in smaller font size. Black arrows (→) show irreversible reactions. Dashed black lines represent abstracted interactions shown in (A). The empty set symbol (∅) is used as the product of degradation reactions. A comprehensive list of reactions and their parameters can be found in section 2.4.

4.3.3. Default Processor Module

The default processor module was designed to respond to the presence of C12-HSL (the molecule produced by detector cells), and express RhlI to catalyse production of C4-HSL. To detect C12-HSL, the LasR transcription factor was used. Similar to the LacI mechanism, LasR forms a dimer, however it is not thought to form a dimer of dimers[314], [315]. C12-HSL can bind dimeric LasR with a stoichiometry of one AHL molecule per protein, which enables the LasR complex to bind a specific region of the *PLas* promoter. When bound, unlike LacI which blocks polymerase binding, LasR recruits RNA polymerase to the promoter site and hence drives transcription of downstream coding regions. In the default processor module, *rhlI* was placed under the control of *PLas*, and therefore RhlI production could be induced by the presence

of LasR and C12-HSL. In addition to the RhlI coding region, *mCherry* was also included downstream of *PLas*, allowing for co-expression of the two proteins. As mentioned previously, this allowed for characterisation of the module system via detection of red fluorescence.

Like the IPTG detector module model, the default processor module was modelled as a set of mass action reactions and simulated deterministically. Based on previous studies, a number of assumptions were made regarding the LasR mechanism. Firstly, it was assumed that LasR dimerisation occurs at a relatively high rate, but that dissociation of the dimer also occurs quickly unless an AHL molecule is bound to the dimer[182]. It was also assumed that degradation of LasR monomers is high[180], but LasR dimers are protected from proteases to such a high degree that their degradation could be ignored. These assumptions were based on an experimental study[316] which found that in the absence of C12-HSL, LasR is unstable and difficult to extract from cells in high quantities. When C12-HSL was added, however, LasR showed increased stability and extraction of the protein was far more successful. These observations lead the authors to conclude that a feature of the LasR mechanism is that C12-HSL not only allows LasR to bind to DNA, but also increases the amount of LasR present by preventing degradation. Another assumption made was that LasR dimers with only one C12-HSL molecule bound would still be able to bind DNA, although at a much slower rate due to reduced conformational change, but LasR dimers with no ligand bound would show no DNA binding activity. This assumption was based on studies where ligand-free LasR did not appear to bind DNA at all, but that dimeric LasR complexed with non-canonical ligands still showed some DNA binding activity, despite non-optimal conditions[184], [316].

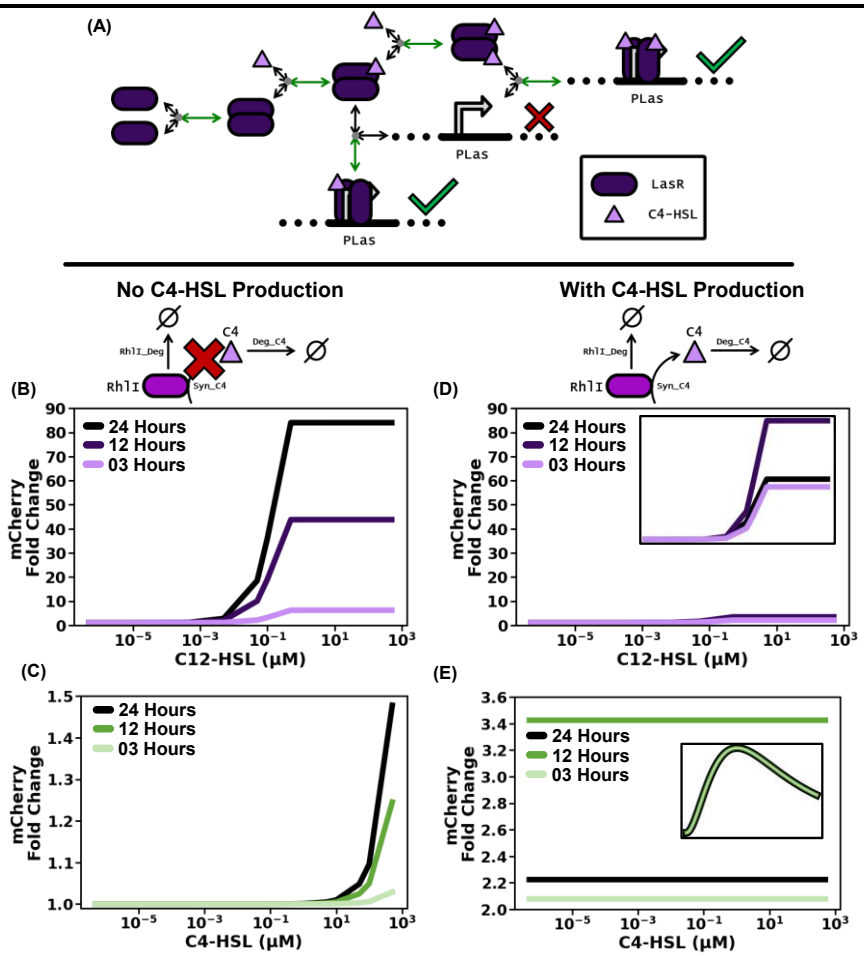


Figure 4.7. Canonical and Crosstalk Response Predictions for the Default Processor Module

Results from deterministic simulation of the default processor module SBML model depicted in Figure 4.6. Full details regarding simulation can be found in section 2.4. (A) Schematic depicting C4-HSL crosstalk interactions. Symbols are the same as seen in Figure 4.6 (A). (B-C) Results for the system with no C4-HSL synthesis activity. (D-E) Results for the system with C4-HSL synthesis activity. (B, D) Dose-response curves with concentrations of the canonical inducer (C12-HSL) between 10-6 and 50 μM. Responses were measured as fold change in either mCherry production compared to no C12-HSL addition after 3, 12, and 24 hours. (C, E) Same as (B, D) but with C4-HSL as inducer. The inset in (E) shows time course of mCherry fold change over 24 hours.

An important aspect of the default processor system was crosstalk, where molecules other than C12-HSL could bind and activate LasR. Crosstalk behaviour has been well documented for LasR, where a range of ligands have shown to have binding capacity, including C4-HSL[151], [175], [180]. Because C4-HSL was to be present in the default processor system as a product of RhlI, interactions between LasR and C4-HSL were included in the model. In line with previous studies, the binding of C4-HSL to LasR was modelled as occurring at a slower rate than C12-HSL binding, and the complex as being less stable. However, it was assumed that once LasR was bound to the *PLas* promoter, no matter the type or number of ligands bound, the rate of transcription would

be the same. The basis for this assumption was that polymerase recruitment activity would not be affected by the type of ligand, and that observations of reduced genetic expression when inducing with molecules other than C12-HSL were a result of reduced binding to the LasR transcription factor. A simplified schematic of the system modelled is shown in Figure 4.6.

Initial parameters for the modelled reactions were obtained through a mixture of literature, assumptions, and estimations. Initial parameters were subsequently modified to ensure behaviour approximated that found in similar systems. As other systems using LasR reported in literature tend to not involve C4-HSL production by RhlI, initial simulations excluded catalysis of the AHL to allow for better comparison of functionality. Results for these simulations can be seen in Figure 4.7 (B) and (C), where (B) shows simulations in which C12-HSL was used as the inducer, and (C) shows crosstalk simulations with C4-HSL as the inducer. In these simulations production of mCherry is shown. The dose-response curves presented largely matched with experimental observations presented previously[151], [315]; namely that activation with C12-HSL was apparent at concentrations above approximately 0.1 μM, and that a maximum fold change of around 5 is seen after 3 hours of incubation. The initial simulations shown here also match crosstalk experiments which suggested much lower activation when using C4-HSL as the inducer, with a fold change of less than 2.

Once simulations had been performed without C4-HSL production, the RhlI synthesis reaction was added using a rate found in literature[181]. Dose-response curves for these simulations are shown in Figure 4.7 (D) and (E). Once again, the response shown is mCherry production. As with the IPTG detector module, production of the AHL synthetase and AHL (RhlI and C4-HSL here) followed similar patterns to mCherry production. As might be expected, responsiveness to C4-HSL as an inducer was completely diminished when RhlI activity was restored, due to high background levels of the AHL. The C12-HSL dose-response curve also displayed expected behaviour; maximal activity was reduced due to the C4-HSL background resulting in higher basal activity. It was also observed that over time, fold change expression of mCherry reduced, with higher levels at 12 hours post induction compared to 24 hours (Figure 4.8 (A)). The system without C4-HSL production showed more typical behaviour, where mCherry production increased gradually before plateauing (Figure 4.8 (C)). The behaviour observed when C4-HSL production is enabled appears to be a result of a

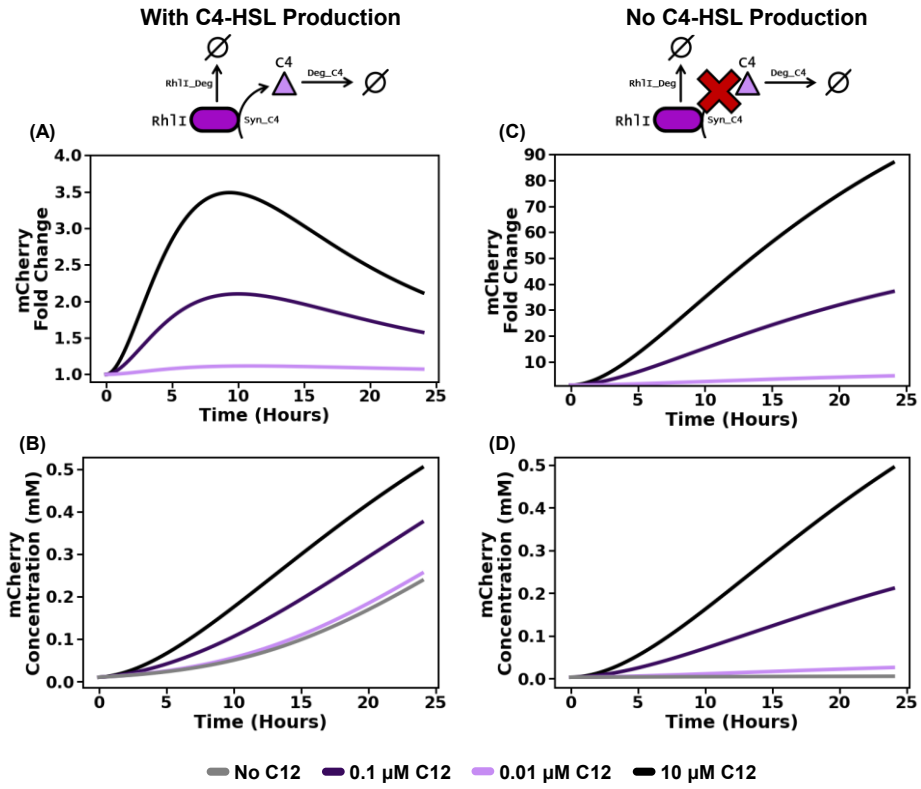


Figure 4.8. Simulated Default Processor Module Feedback Loop
Simulated results demonstrating how the default processor module's feedback loop impacts response characteristics. (A-B) Results for the processor module system with C4-HSL synthesis activity. (C-D) Results for the system with no C4-HSL synthesis activity. (A, C) Time course curves over 24 hours for systems induced with 0.1, 0.01, or 10 µM C12-HSL. Fold change in mCherry production compared to the uninduced system was used as the dependent variable. (B, D) Same as (A, C), but results for the uninduced system are also shown, and the dependent variable was mCherry concentration in mM.

positive feedback loop occurring in systems with no inducer, where background levels of C4-HSL could activate production of RhlI, which lead to production of more C4-HSL. Thus, over time, the levels of C4-HSL increased in inactivated systems which manifested as a reduction in relative activity. Background activation in systems with C4-HSL production can be more clearly observed in time course plots of mCherry production, where noticeable increases in mCherry over time occur even with no or little inducer (Figure 4.8 (B)), compared to the system without AHL production where mCherry production remained low throughout the simulation when no inducer was present Figure 4.8 (D).

The default processor module model suggested that although activation by C12-HSL may be lower than observed in similar systems which do not include production C4-HSL, functionality should be retained. The model also predicted that the fold change of mCherry, RhlI, and C4-HSL levels compared to uninduced samples are likely to

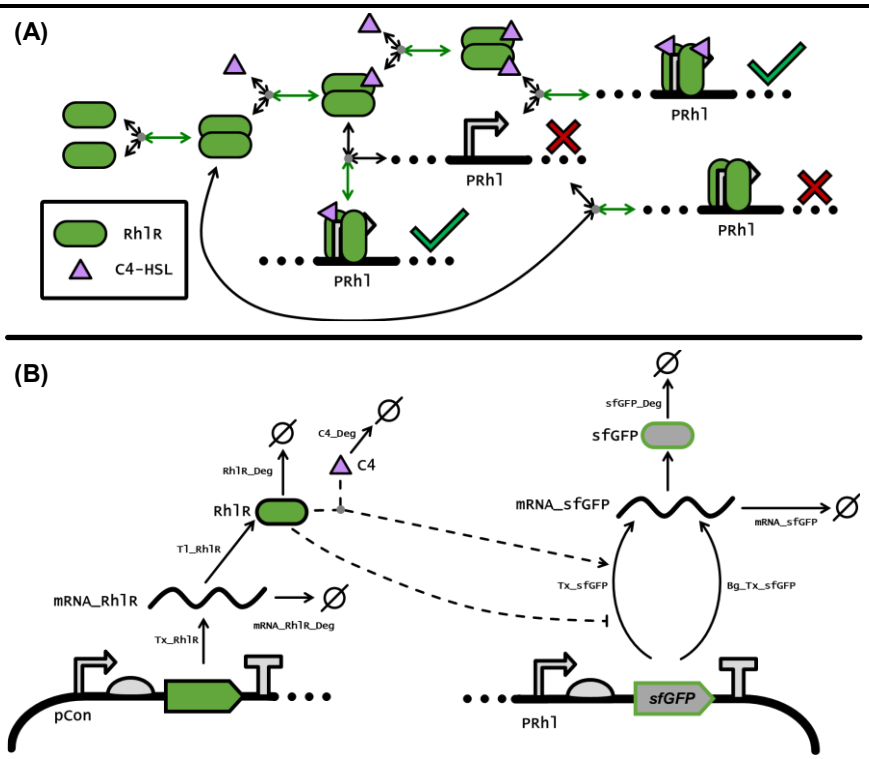


Figure 4.9. sfGFP Reporter Module Model Schematics

Depictions of reactions and interactions in the sfGFP reporter module SBML model. **(A)** RhlR complex formation, C4-HSL binding, and interaction with *PRhl* promoter. Grey circles (●) are used to indicate association/dissociation reactions. Green double-sided arrows (↔) show products of association, black double-sided arrows (↔) show products of dissociation. Red crosses (✗) indicate unfavourable transcription. Green ticks (✓) indicate favourable transcription. **(B)** Schematic depicting the sfGFP Reporter module SBML model. Entity names are shown in larger font size, and reaction names in smaller font size. Black arrows (→) show irreversible reactions and capped arrows (⊣) show repressive interactions. Dashed black lines represent abstracted interactions shown in (A). The empty set symbol (∅) is used as the product of degradation reactions. A comprehensive list of reactions and their parameters can be found in section 2.4.

peak after roughly 10 hours of growth post-induction, before slowly dropping, likely due to a positive-feedback loop.

4.3.4. sfGFP Reporter Module

The sfGFP reporter module was designed for induction by C4-HSL. This functionality was achieved by employing the RhlR transcription factor. The mechanism of RhlR is thought to be similar to that of LasR, where dimeric RhlR forms and binds its ligand (C4-HSL) with a stoichiometry of one molecule per protein[185]. The RhlR dimer complex can also bind DNA, but in this case recognises a region within the *PRhl* promoter, and recruits RNA polymerase to drive transcription of downstream coding regions. In the case of the sfGFP reporter module, *PRhl* was designed to control expression of sfGFP. Unlike LasR, there is evidence that ligand-free, dimeric RhlR can repress expression from *PRhl*[187]. However, many studies have also reported

activation of RhlR as a transcriptional activator with non-canonical ligands, including C12-HSL[151], [186] (Figure 4.10 (A)). This suggests that binding of RhlR to the promoter region is independent of ligand binding, but the ligands are required for polymerase recruitment. It has also been observed that genetic expression from *PRhl* is higher than that from *PLas*, and that fold changes in expression levels between induced and uninduced systems also tend to be higher[151]. Aside from these differences, the binding mechanics are thought to be similar.

As with the previous two models, the sfGFP reporter model was also modelled as a set of mass action reactions. Due to the similarities between LasR and RhlR, many of the parameters used in the default processor module model were used as the basis for the sfGFP reporter module model. As before, these parameters were modified to ensure simulation results were in general agreement with similar systems in previous studies. One notable difference was the transcription reaction from *PRhl* bound to ligand-free RhlR. In the reporter model, this reaction had a rate of 0 min^{-1}, which reflected RhlR's ability to act as a repressor in the absence of any ligands. Additionally, the binding of dimeric RhlR in all states were set to the same rate, as it has been reported that the DNA binding functionality of RhlR is unaffected by the presence or type of ligands. However, unlike the processor model where transcription from *PRhl* occurred at the same rate once LasR was bound, in this reporter model the rate of transcription was dependent on the ligands bound, as a result of the apparent impact ligand binding to RhlR has on polymerase recruitment. The inclusion of ligand-free, dimeric RhlR to DNA required a set of extra reactions to be defined; binding of C4-HSL or C12-HSL to RhlR already bound to *PRhl*. For these reactions, it was assumed that AHL binding would occur at a slower rate than RhlR not bound to DNA, as the free transcription factor would not experience affects such as steric hinderance.

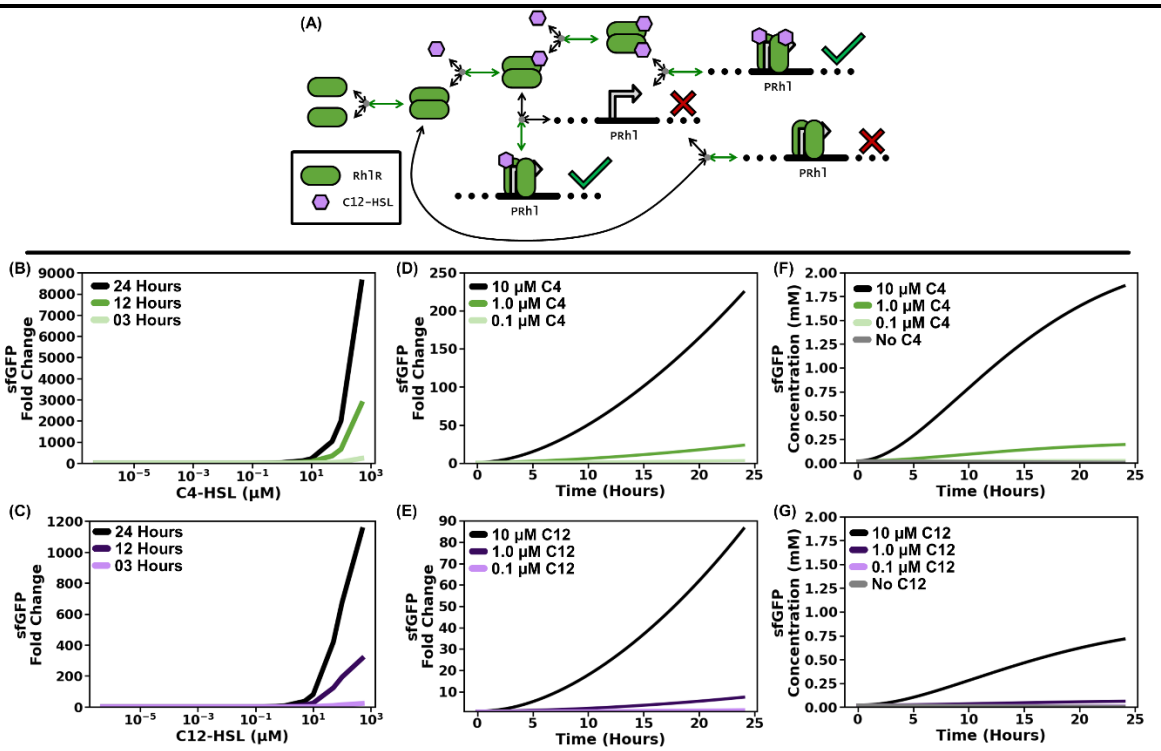


Figure 4.10. Simulated sfGFP Processor Module Behaviour

Results from deterministic simulation of the sfGFP Reporter module SBML model depicted in Figure 4.9. Full details regarding simulation can be found in section 2.4. **(A)** Schematic depicting C12-HSL crosstalk interactions. Symbols are the same as seen in Figure 4.9 (A). **(B, D, F)** Results from systems induced with the canonical C4-HSL inducer. **(C, E, G)** Results from systems induced with the non-specific C12-HSL inducer. **(B-C)** Dose-response curves with concentrations of inducer between 10-6 and 50 μM IPTG. Responses were measured as fold change in sfGFP production compared to no inducer addition after 3, 12, and 24 hours. **(D-E)** Time course curves over 24 hours when induced with either 10, 1.0, or 0.1 μM of relevant AHL. The dependent variable was fold change in sfGFP production compared to the system with no AHL added. **(F-G)** Time course curves over 24 hours when induced with 10, 1.0, 0.1, or 0.0 μM AHL. Dependent variable was concentration of sfGFP in mM.

A simulated dose-response curve for the sfGFP reporter module with the canonical C4-HSL inducer is shown Figure 4.10 (B). Unlike the IPTG detector and default processor modules, the range of inducers tested were not predicted to saturate the signal. This behaviour was in accordance with experimental data previously reported[151]. Additionally, the fold change in activation relative to no inducer was predicted to be much larger than that seen for the detector and processor. This was likely due to the inclusion of ligand-free RhlR as a repressor, which resulted in reduced background expression from *PRhl* when no inducers were present. The dose-response curve shown in Figure 4.10 (C) shows responsiveness to increasing concentrations of C12-HSL. Whilst not present in the reporter module, in a co-culture the IPTG detector module would be expected to produce C12-HSL, and hence modelling the crosstalk was important. The dose-response curve shows predicted that the reporter module

would begin responding at similar levels of both C4- and C12-HSL, but the fold change in sfGFP expression would be much lower for C4-HSL. Over time, for both C4- and C12-HSL as an inducer, the fold change (Figure 4.10 (D-E)) and total concentration ((Figure 4.10 (F-G)) for sfGFP shows a similar pattern to activation of the IPTG detector module, rather than the behaviour predicted for the processor module. This was as expected due to the lack of a potential feedback loop, which was present in the processor module system.

4.4. Agent-Based Modelling of a Modular and Multi-Microbial Biosensor

The models and simulations presented in section 4.3 provided a basis for predicting behaviour of each module. In this section, a model for predicting behaviour of the multi-microbial biosensor is presented. For this model, it was not appropriate to assume homogeneity, due to the importance of heterogeneity in multi-microbial systems, as discussed previously. Therefore, an agent-based model approach was used (see sub-section 4.1.3). The agent-based biosensor model was required to simulate the behaviour of each cell type in the system (detector, processor, and reporter) and model interactions between the cells, specifically intercellular communication via production and diffusion of small molecules. To assist with building and simulating the biosensor model, Simbiotics was selected as the modelling and simulation platform[77]. Simbiotics was selected as it was developed with agent-based modelling of multi-microbial systems in mind and has the unique feature of allowing complex cellular behaviour to be defined via SBML models. This allowed for the models shown in section 4.3 to be directly integrated as modular behaviour. Simbiotics is also capable of modelling systems in 3-dimensional space, which was essential for predicting the biosensor's behaviour in liquid culture and provides simulators for diffusion of chemicals extracellularly and across cell membranes. The ability to model diffusion across the cell wall meant that quorum sensing-based communication could be simulated.

In Simbiotics, the native SBML simulator module uses libsbmlsim, which is a java package capable of simulating models captured in SBML. However, testing of this module found that the simulator methods employed by the java package were not suitable for simulation of the biosensor models. Whilst the integrated SBML solver had been demonstrated previously to function for relatively simple models [317], here it was found to have issues when simulating more complex models with a larger number of entities and interactions. Specifically, issues seemed to stem from 'stiff' equations in the models, where interacting entities can be present in quantities magnitudes apart, and difficulties in identifying appropriate time steps can lead to numerical instabilities within simulations[318]. To tackle this problem, the SBML simulator module was changed to use basico instead[301], which allowed for the same deterministic method described in section 4.3 to be used for solving cell behaviour. Although the cell behaviour models were simulated deterministically, the rest of the model, including

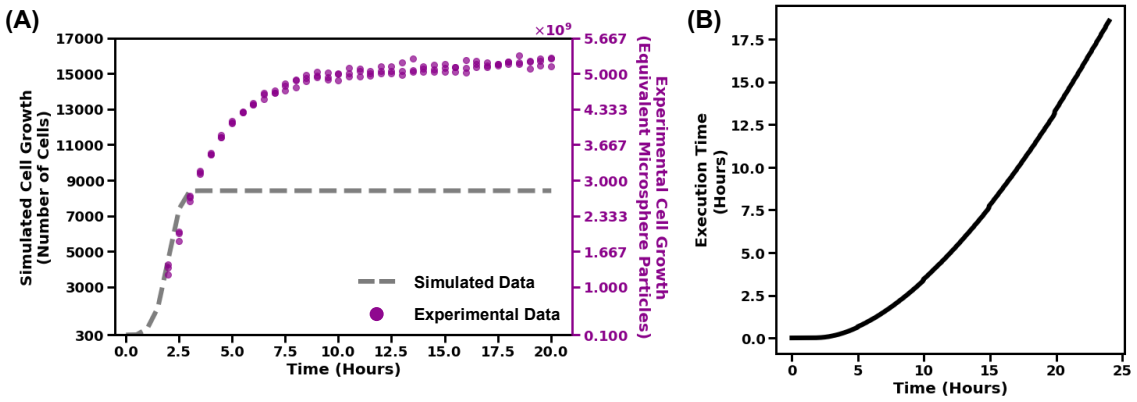


Figure 4.11. Simulated Cell Growth Curve

Results from simulating cell growth using Simbiotics. **(A)** Growth curves obtained over 20 hours from either agent-based simulation or experimentally (section 2.5.3). Simulated data shows number of cells (agents) over time (left Y-axis). Experimental data shows equivalent microsphere particles over time (right Y-axis). The Y-axis values for both curves were calibrated against one another as described in section 2.5.3. **(B)** Graph showing simulated time against cumulative execution time. Simulation details, including computational specifications, can be found in section 2.5.

diffusion of chemicals and movement of cells, were simulated stochastically using the built-in Simbiotics methods.

4.4.1. General Model Assumptions

Agent-based modelling can require far more computational resource than other types of models, and the resources and time required tends to scale with the number of agents present[319]. In biological systems such as the ones under study here, the number of cells can easily reach 10^7 to 10^9 [63]; agent-based models typically cannot easily scale to these numbers when integrating complex modelling for each agent. Therefore, the decision was made to only model a subsection of the total system and assume that this local section would be representative of the global system[285]. For the simulations presented here, the initial number of cells was set to 300. For the cells themselves, Simbiotics defines two morphologies: spherical or rod shaped. As the proof-of-concept biosensor was implemented in *Escherichia coli*, cells were modelled as rod-shaped with a length of 1.5 µm and a diameter of 0.5 µm. For each cell type, behaviour was defined by the corresponding SBML model developed previously. As with the simulations presented in section 4.3, the initial starting amount for each entity was determined using the results from simulating each SBML model for 24 hours. As before, basico was used to perform the simulations prior to running the Simbiotics model.

When defining extracellular chemicals (in this case IPTG, C12-HSL, and C4-HSL) in Simbiotics, three coefficients can be set: extracellular diffusion, diffusion across cell membranes, and degradation. Although degradation of the AHLs was included for reactions within the cells, it was assumed that once in the media all chemicals would remain stable for the duration of the experiment. For initial simulations, the diffusion coefficients for all three chemicals were assumed to be similar, and thus were set to the same coefficients. During experimentation, it was planned that samples would be continually shaken, and thus the diffusion coefficients were set relatively high to account for this constant motion, along with a global Brownian motion coefficient of 2.5. Addition of either IPTG, C12-HSL, or C4-HSL as an inducer was incorporated into the model as a pipetting event, where the appropriate number of molecules were added to the centre point of the space at time point 0. For systems with no inducer added, no pipetting event was scheduled.

To determine the size of the system subsection to be simulated, the average volume of liquid (in µL) per cell was calculated using equations ii, iii, and iv:

$$[Cells_{exp}] = \frac{Cells_{exp}}{Volume_{exp}} \quad \text{i.}$$

$$Volume_{sim} = \frac{Cells_{sim}}{[Cells_{exp}]} \quad \text{i.}$$

$$World_{edge} = \sqrt[3]{(Volume_{sim} \times 1e^{9})} \quad \text{/.}$$

For the above equations, $Cells_{exp}$ and $Cells_{sim}$ are the initial number of cells in the experimental and simulated systems respectively, $Volume_{exp}$ and $Volume_{sim}$ are the total volumes (in µL) of the experimental and simulated systems respectively, and $World_{edge}$ is the length (in µm) of each edge of the simulated subsystem boundary.

4.4.2. Simulating cell growth

Simbiotics allows for nutrient-dependent growth to be modelled, where the rate at which cells in the system grow and divide is dependent on resource availability. The modules and biosensor were intended to be characterised in batch culture, rather than in a continuous flow or chemostat environment. Therefore, if cell growth was to be modelled, nutrient availability would likely be a limiting factor over time as resources would not be replenished. The growth rate of each agent, or cell, in the system was determined using Monod kinetics, where a maximum possible growth rate is moderated based on the availability of a specific nutrient or set of nutrients. For the

simulations here, a single abstracted nutrient (simply termed ‘food’) was used to represent all resources required for cell growth. This was because the exact resources required were unknown. To determine Monod growth rate parameters, different values were tested until a growth curve approximating experimental data was found. The experimental growth curve was obtained by inoculating 100 µL of Lysogeny Broth (LB) media in a standard 96 well, flatbottomed microplate with untransformed *E. coli* DH5α at a starting optical density of 0.1 when measured at 600 nm. The cultures were then incubated at 37^{o}C with shaking at 300 RPM for 20 hours. Optical density measurements were converted to an approximate number of cells according to standard calibration protocols (see sub-section 5.1.1 and section 2.5.3 for more detail). This approach made the initial assumption that all three cell types would grow at identical rates, and at the same rate as untransformed cells, which was unlikely to be true due to the different stresses placed upon each cell by each plasmid and construct. Additionally, the detector, processor, and reporter cell types would be grown in the presence of antibiotics to exert a selection pressure for retention of the plasmids, which tends to reduce growth rates further. Nevertheless, this provided an initial basis for indicative predictions of behaviour.

The simulated growth curve compared to experimental data can be seen in Figure 4.11 (A). Simulations were performed using a single cell type with no behaviour (i.e., no SBML model) to save on computational resources and simulation time. It was possible to exclude the SBML-based behaviour as it had no bearing on how the cells grew *in silico* (although as mentioned previously, in reality there would be an effect). However, it was found that even with a small number of starting cells (300) and an underestimate of cell growth, total execution time for 24 hours of simulated growth was more than 18 hours (Figure 4.11 (B)) when simulations were performed using hardware described in section 2.5. It became apparent that experimental characterisation, where 60 samples could be tested in parallel over 24 hours, would be less time consuming than computational simulation, given the hardware specifications used. Therefore, it was decided that cell growth would be abstracted from the model. This abstraction placed introduced limitations, as factors like increased cell density and competition for resources between the different cell types in co-culture could not be accounted for. However, it was thought to still be possible to make informed decisions based on indicative results from the scaled back models.

4.4.3. Agent-based modelling of each cell type in monoculture

Before simulating the multi-microbial biosensor in co-culture, each cell type was simulated using the Simbiotics platform in a monoculture. Agent-based modelling of each cell type allowed for comparisons to be made between the pure-deterministic models presented in the previous section and the agent-based model. As each cell in the system was simulated separately, it was possible to obtain simulated single cell data, allowing for an insight as to the possible heterogeneity between cells of the same type.

As before, a range of inducer concentrations were tested for each cell type in order to obtain a dose-response curve. In the agent-based model, stochasticity has an impact, and thus simulations were performed in replicates of at least three to help determine run-to-run variation. The simulated dose-response curve for the IPTG detector module can be seen in Figure 4.12 (A). Compared to the curve shown in Figure 4.5 (A), it can be seen that the agent-based model predicted a smaller dynamic range for the IPTG detector module, with a maximal fold change in eCFP production of approximately 4.0 after 24 hours, compared to an increase of 9.5 times background predicted by the purely deterministic model. However, both models predict similar levels of sensitivity, with a noise threshold of between 0.1 and 0.5 mM IPTG. Both models also agreed that after 24 hours of growth, the system would not have reached a steady state in terms of eCFP production (Figure 4.12 (B) and Figure 4.5 (C)) when induced with higher concentrations of IPTG. It was possible that the smaller dynamic range prediction by the Simbiotics model was a result of the incorporation of IPTG diffusion, both in the extracellular environment and across the cell membrane. Cells in the agent-based model would have had a longer period of delay before exposure to inducer molecules, compared to the purely deterministic simulations where cells had access to IPTG from time point 0. Therefore, as eCFP production was predicted to still be increasing after 24 hours, the state of the system when simulated by Simbiotics would be delayed, and hence the fold change at this time point would be lower.

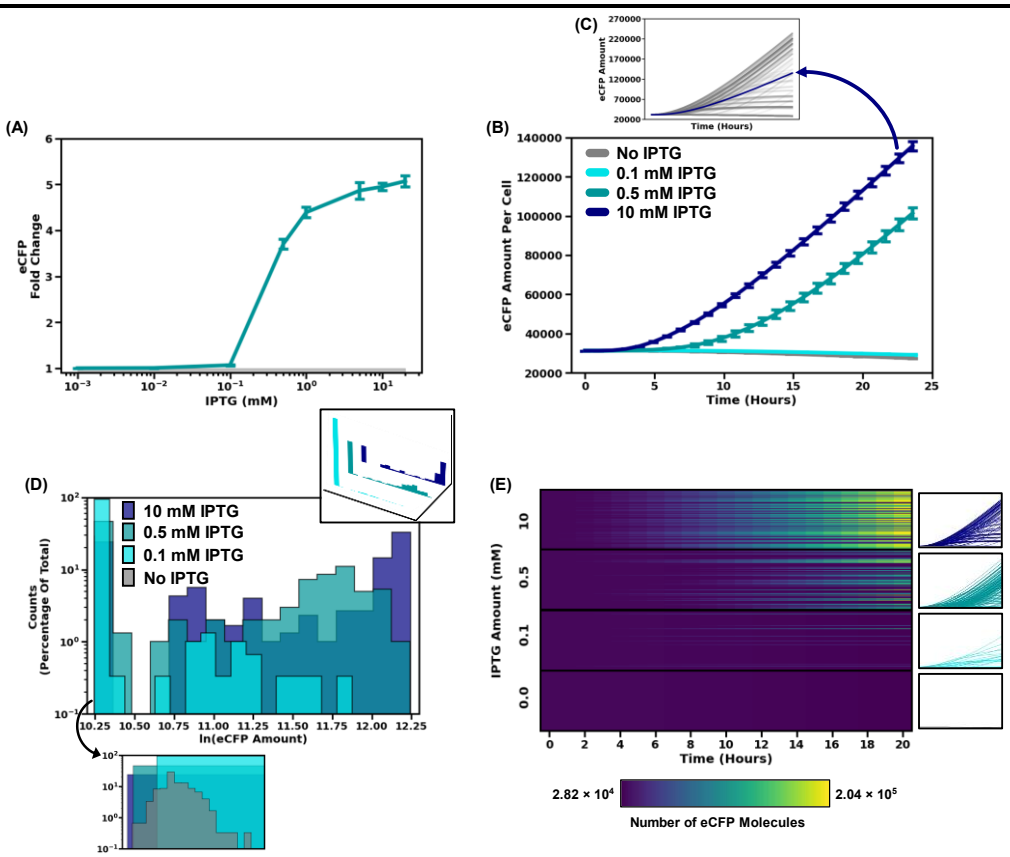


Figure 4.12. Agent-Based Modelling Results for the IPTG Detector Module
Results from agent-based simulation of the IPTG detector module in monoculture. All simulations were performed with 300 cells and according to the methods in section 2.5.1. **(A)** Dose-response curve with different concentrations of IPTG as an inducer, where fold change in eCFP production after 24 hours was measured as the response. The grey box shows background noise, calculated as stated in section 2.5.2. **(B)** Time course curve for production of eCFP proteins over 24 hours. Results are shown as a bulk measurement of all cells, and a number-per-cell was calculated by averaging the total value across all cells in the system. **(C)** Time course of eCFP production per cell when induced with 20 mM IPTG. The navy line shows bulk measurement, where the number of eCFP molecules per cell was calculated as an average from the total value. The grey lines show single cell data for eCFP production. Time from 0 to 24 hours is along the x-axis. **(D)** Histogram depicting simulated single cell data from systems 20 hours post induction with either 0, 0.1, 0.5, or 10 mM IPTG. Data was grouped into 17 bins. The box below the graph shows histogram zoomed in for 0 mM. Top right inset shows the same data as main graph for 0.1, 0.5, and 10 mM IPTG on a 3D plot, where the x-axis is the number of eCFP molecules, and the y-axis is the counts on a linear scale. On the z-axis is the amount of IPTG used for induction. **(E)** Lasagna plots[320] depicting single cell data for systems induced with either 0, 0.1, 0.5, or 10 mM IPTG. Each horizontal line represents a single cell over 20 hours. Colouring is based on the number of eCFP molecules per cell. Time course plots are shown for each system to the right of the lasagna plot. The y-axis is number of eCFP molecules.

The plot in Figure 4.12 (B) shows the number of eCFP molecules per cell produced over time when induced by different concentrations of IPTG, along with error bars to indicate potential stochastic variation across repeats. This data can be thought of as ‘bulk’ measurements, where the total amount of an entity is measured for the system as a whole, and values per cell can be obtained by averaging the value across the

number of cells present. Although bulk measurements are a common way to measure the overall behaviour of a system[294], it does not give any insight as to variation across cells within a system. As Simbiotics simulates each cell separately, it was possible to export the results obtained for each cell to compare differences in state compared to other cells. Figure 4.12 (C) shows a time course of eCFP production for all cells in a single system when induced with 10 mM IPTG. It was observed that the activity of each cell can vary drastically. As the cell behaviours were simulated using a deterministic solver, the majority of variation was thought to come from proximity of each cell to different amounts of IPTG, which was the major impactor on cell behaviour. In Figure 4.12 (D), a histogram shows the spread of eCFP production across cells in a single system when induced with 4 different IPTG concentrations, and when uninduced, after 20 hours of growth. Although the general pattern of eCFP production seen with bulk measurements (higher amounts of IPTG correlates to more eCFP), there is a noticeable overlap of data. For example, it can be seen that some cells in the system induced with 0.1 and 0.5 mM of IPTG show higher eCFP production than some cells in the system induced with 20 mM IPTG. Again, this is consistent with the thought that the behaviour of each cell was influenced by the local concentration of inducer, as higher eCFP production would be expected if cells were disproportionately exposed to different concentrations of IPTG. If this was the case, it would be expected that cells in simulations with not IPTG added would show no variation. Indeed, the spread of data is far smaller than that seen in systems with IPTG, although there are some differences across the cells (Figure 4.12 (D) bottom left). These differences appear to be an artifact of numerical errors in the deterministic solver due to the stiff equations, which can occasionally cause entities to spontaneously appear in very low quantities or be calculated as a negative value (which is biologically impossible). To help account for this known behaviour, after each cell was solved deterministically, the results were automatically checked for these spontaneous appearances of entities at very low quantities or negative values and corrected to a value of 0. However, in between these checks, the erroneous values could have had a small impact on other entities in the system, and this is what causes slight variations in what should be consistent solutions to deterministic simulations.

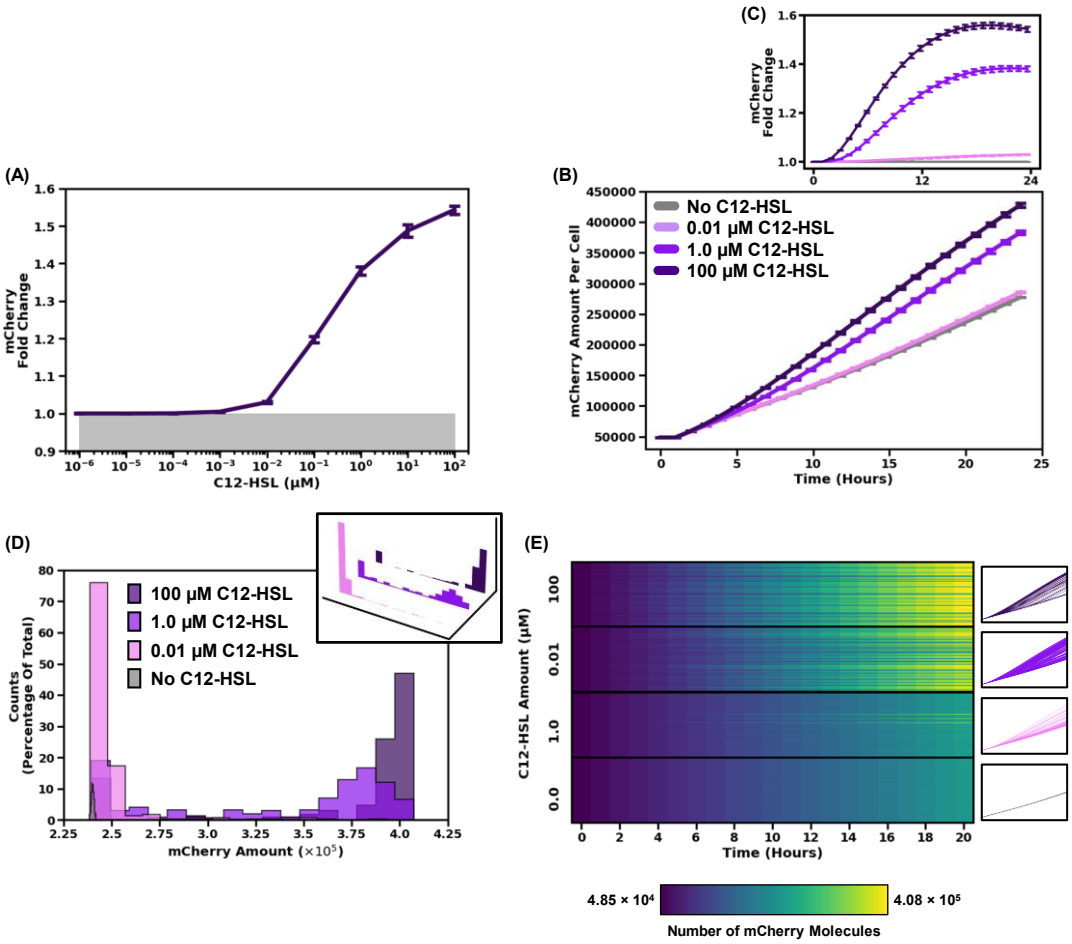


Figure 4.13. Agent-Based Modelling Results for the Default Processor Module
Results from agent-based simulation of the default processor module in monoculture. All simulations were performed with 300 cells and according to the methods in section 2.5.1. **(A)** Dose-response curve with different concentrations of C12-HSL as an inducer, where fold change in mCherry production after 24 hours was measured as the response. The grey box shows background noise, calculated as stated in section 2.5.2. **(B)** Time course curve for production of mCherry proteins over 24 hours. Results are shown as a bulk measurement of all cells, and a number-per-cell was calculated by averaging the total value across all cells in the system. **(C)** Time course curve of fold change in mCherry production per cell over 24 hours relative to uninduced cells. **(D)** Histogram depicting simulated single cell data from systems 20 hours post induction with either 0, 0.01, 1.0, or 100 μM C12-HSL. Data was grouped into 17 bins. Top right inset shows the same data as main graph for 0.01, 1.0, and 100 μM IPTG on a 3D plot, where the x-axis is the number of mCherry molecules, and the y-axis is the counts on a linear scale. **(E)** Lasagna plots depicting single cell data for systems induced with either 0, 0.01, 1.0, or 100 μM C12-HSL. Each horizontal line represents a single cell over 20 hours. Colouring is based on the number of mCherry molecules per cell. Time course plots are shown for each system to the right of the lasagna plot. The y-axis is number of mCherry molecules.

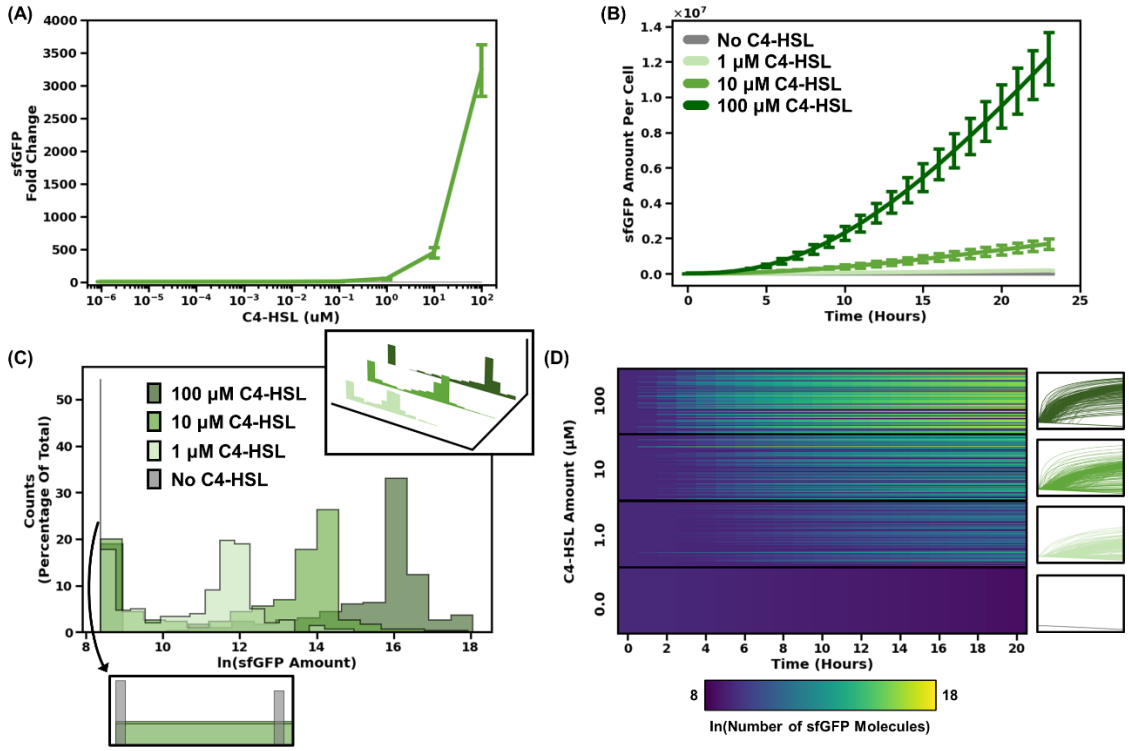


Figure 4.14. Agent-Based Modelling Results for the sfGFP Reporter Module
Results from agent-based simulation of the sfGFP Reporter module in monoculture. All simulations were performed with 300 cells and according to the methods in section 2.5.1. **(A)** Dose-response curve with different concentrations of C4-HSL as an inducer, where fold change in sfGFP production after 24 hours was measured as the response. The grey box shows background noise, calculated as stated in section 2.5.2. **(B)** Time course curve for production of sfGFP proteins over 24 hours. Results are shown as a bulk measurement of all cells, and a number-per-cell was calculated by averaging the total value across all cells in the system. **(C)** Histogram depicting simulated single cell data from systems 20 hours post induction with either 0, 1, 10, or 100 µM C4-HSL. Data was grouped into 17 bins. Box below the graph shows histogram zoomed in for 0 µM. Top right inset shows the same data as main graph for 1, 10, or 100 µM C4-HSL on a 3D plot, where the x-axis is the number of eCFP molecules, and the y-axis is the counts on a linear scale. (E) Lasagna plots depicting single cell data for systems induced with either 0, 1, 10, or 100 µM C4-HSL. Each horizontal line represents a single cell over 20 hours. Colouring is based on the natural log of the number of sfGFP molecules per cell. Time course plots are shown for each system to the right of the lasagna plot. The y-axis is the natural log of the number of eCFP molecules.

The simulation results also suggested that heterogeneity may be higher when inducer concentrations within the linear range of the dose-response curve (~0.1 to ~ 10 mM for the detector cells), than when concentrations below the detection threshold or above the saturation point were used. For IPTG concentrations above the saturation point, this likely reflected the fact that although individual cells may be exposed to different amounts of inducer, this could not be observed in the response due to saturation occurring within most cells. For IPTG concentration below the detection threshold, lower heterogeneity may have occurred due a majority of cells being exposed to levels of IPTG below the amount required to show a dramatic difference in response.

To investigate how heterogeneity was predicted to change over time, values for the amount of eCFP production over 20 hours for four concentrations of IPTG (0, 0.1, 0.5, and 10 mM) were visualised as a Lasagna plot (Figure 4.12 (E)). This shows that the general trend seen with bulk measurements remains, where the amount of eCFP produced increased over time, except in the uninduced system where eCFP numbers remained stable. However, it can also be seen that cells within the same system did not all activate synchronously, and eCFP production began at different points in time. There was also agreement with the conclusion drawn from Figure 4.12 (D) that inducing the detector cells with IPTG at concentrations within the linear range of the dose-response curve resulted in higher heterogeneity than when using IPTG concentrations outside of this range.

Simulations performed for the IPTG detector module were repeated for the default processor module. Compared to the purely deterministic processor model, the agent-based model predicted a similar level of sensitivity (approximately 0.01 μM) and fold change in mCherry production after 24 hours, with a 2.2-fold increase for the deterministic model compared to 1.5-fold for the agent-based model (Figure 4.13 (A) and Figure 4.7 (D)). Both models predicted a higher level of background noise than for the IPTG detector module, with mCherry accumulating over time in the absence of any inducer, albeit at a slower rate than induced systems (Figure 4.13 (B) and Figure 4.8 (B)). The deterministic model also predicted that a maximum fold change would be observed approximately 8 hours post induction, at which point the relative signal was predicted to decrease. Whilst the agent-based model simulations showed some evidence of this behaviour (Figure 4.13 (C)), it was far less pronounced, and the decrease in signal was predicted to occur after 20 hours. As with the IPTG detector module models, this may have been due to delays associated with cells becoming exposed to the inducer. Additionally, the maximum fold change predicted by the agent-based model was approximately 2.2 times lower than predicted by the deterministic model.

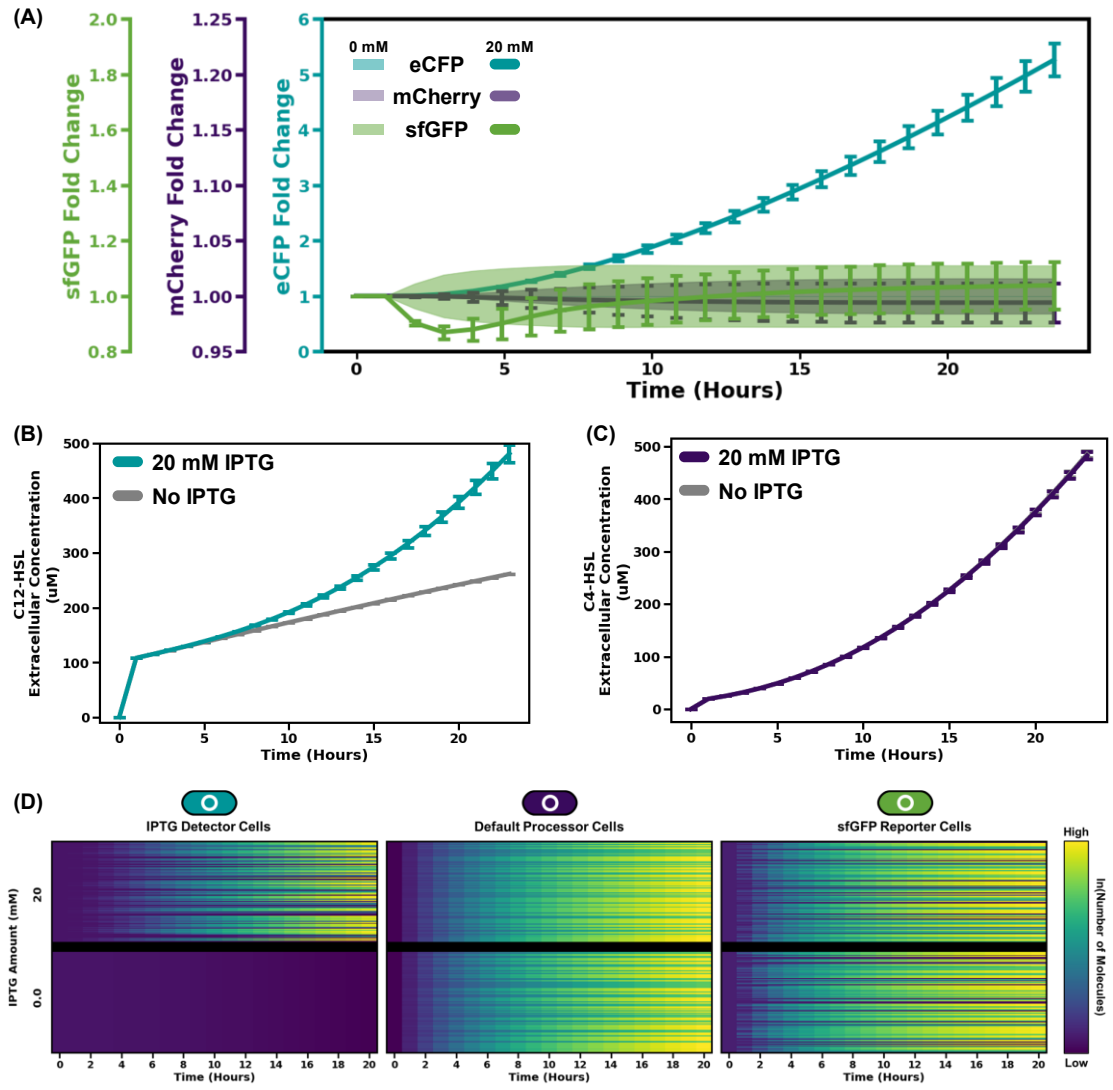


Figure 4.15. Simulation Results for Multi-Microbial Biosensor with a 1:1:1 Cell Ratio
Simulated behaviour of the proof-of-concept modular and multi-microbial IPTG biosensor. All three cell types were added in equal amounts (100 of each). The system was simulated in the presence of either 0 or 20 mM IPTG over 24 hours. Error bars show standard deviation of 4 replicates centred on the mean value. **(A)** Fold change in eCFP, mCherry, and sfGFP production relative to the uninduced system. Fold change values for each fluorescent protein were plotted on Y-axes with different scales. **(B)** Time course curve over 24 hours for the extracellular concentration (in μM) of C12-HSL in the presence of 0 mM and 20 mM IPTG. **(C)** Same as (B) but for extracellular C4-HSL. **(D)** Lasagna plots showing either eCFP (left), mCherry (middle), and sfGFP (right) production over time when induced with 0 mM or 20 mM IPTG. Each horizontal line represents a single cell over time.

For the default processor module, heterogeneity within the system was observed, as shown in Figure 4.13 (D). Similar to observations with the IPTG detector module, there appeared to be a larger cell-to-cell variation in activity when the inducer was added at a concentration within the linear dose-response range (1.0 μM) compared to concentrations near the limit of detection (0.01 μM) or above the saturation point (100 μM). Whilst both the detector and processor modules showed increasing heterogeneity as time progressed (Figure 4.13 (E)), the uninduced processor system showed more

heterogeneity than the uninduced detector system. Whilst the uninduced processor module's variation was still lower than observed with the induced systems, and some of this variation would have been due to numerical artifacts, the positive feedback loop present in the processor module would likely have caused the majority of variation. The feedback loop causes low-level induction by C4-HSL produced by the processor itself, leading to a system which was not truly uninduced.

For agent-based simulation of the sfGFP reporter module, the general conclusions drawn from the dose-response curve (Figure 4.14 (A)) were much the same as those for the detector and processor modules; compared to the purely deterministic model the predicted sensitivity was similar, but the maximum fold change was lower. There was again evidence of delayed response, as evidenced by the lack of a signal plateau in time course curves for sfGFP production when simulated as an agent-based model (Figure 4.14 (B)) compared to when simulated purely deterministically (Figure 4.10 (F)). As with the detector and processor modules, heterogeneity was also observed for the sfGFP reporter module (Figure 4.14 (C-D)). For the reporter system, heterogeneity appeared to increase with higher concentrations of C4-HSL added as an inducer, which was in line with previous observations and conclusions as the saturation point was not reached even at 100 μM of C4-HSL. Cell-to-cell variation for the uninduced system was similar to that observed with the IPTG detector module, where differences in sfGFP quantities were almost identical across all 300 cells in the system, and the only differences were likely due to the numerical instabilities arising during deterministic simulation. It should be noted, however, that the heterogeneity within the sfGFP reporter systems was much larger than that seen with the other two modules. This was likely due to the much larger fold changes in activity observed with increasing concentrations of inducer seen in Figure 4.14 (A), which resulted in more pronounced differences between cells when exposed to different amounts of inducer.

Overall, the agent-based models simulated using Simbiotics showed general agreement with the deterministic-only models presented in section 4.3. The main difference observed between the two model types was a slower response time when using agent-based simulation, which was thought to be due to delays in exposure of cells, and the entities within each cell. The agent-based models also predicted lower maximal fold changes in activity when compared to uninduced systems, which may have been related to the delayed response times. The agent-based models also

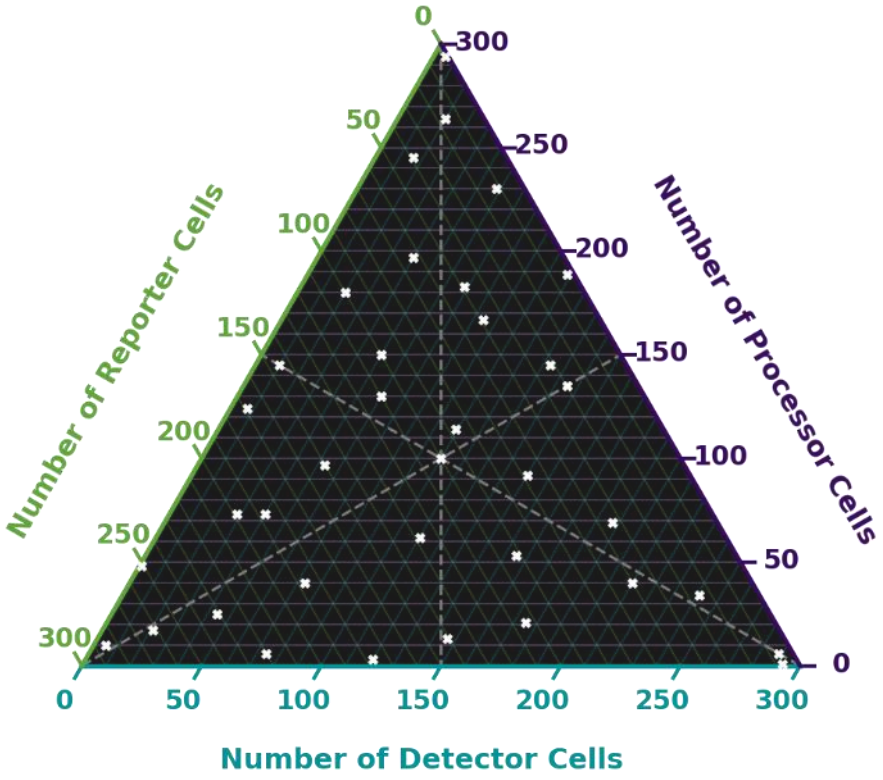


Figure 4.16. Visual Representation of the Cell Ratio Design Space
The cell ratios used in each simulation were visualised on a ternary plot, where each axis represented the number of detector, processor, and reporter cells. The cell ratios used are shown as white crosses. The dashed grey lines show stoichiometric boundaries, where the amounts of two cell types are present in equal amounts.

allowed for prediction of variation across cells within the same system. It was observed that the potential for heterogeneity in terms of behaviour could be significant, even within monoculture.

4.4.4. Simulating the multi-microbial biosensor

Simbiotics was used to predict behaviour of the multi-microbial biosensor by modelling systems with all three module types. As with the mono-culture simulations, systems were limited to 300 cells. Initially, a multi-microbial system consisting of each cell type present in equal amounts (i.e. 100 of each cell type) was simulated. The agent-based model predicted that the detector cells would exhibit behaviour similar to that seen in homogenous culture, but that the processor and reporter cells would not produce a signal above the background noise (Figure 4.15 (A)). The amount of C12-HSL produced by uninduced detector cells was above the predicted saturation limit for processor cells (Figure 4.15 (B)), and hence additional production of C12-HSL by the induced detector cells would not have caused a difference in activation of the processor cells. This meant that there was no fold change in amount of C4-HSL produced by the processor cells when co-cultured with induced or uninduced detector cells (Figure 4.15

(C)). As the levels of C4-HSL were predicted to be the same in both induced and uninduced samples, the reporter cells were also predicted to show no difference in terms of response (sfGFP production). This pattern could also be seen at the single-cell level, where differences were only apparent in the detector cells (Figure 4.15 (D)).

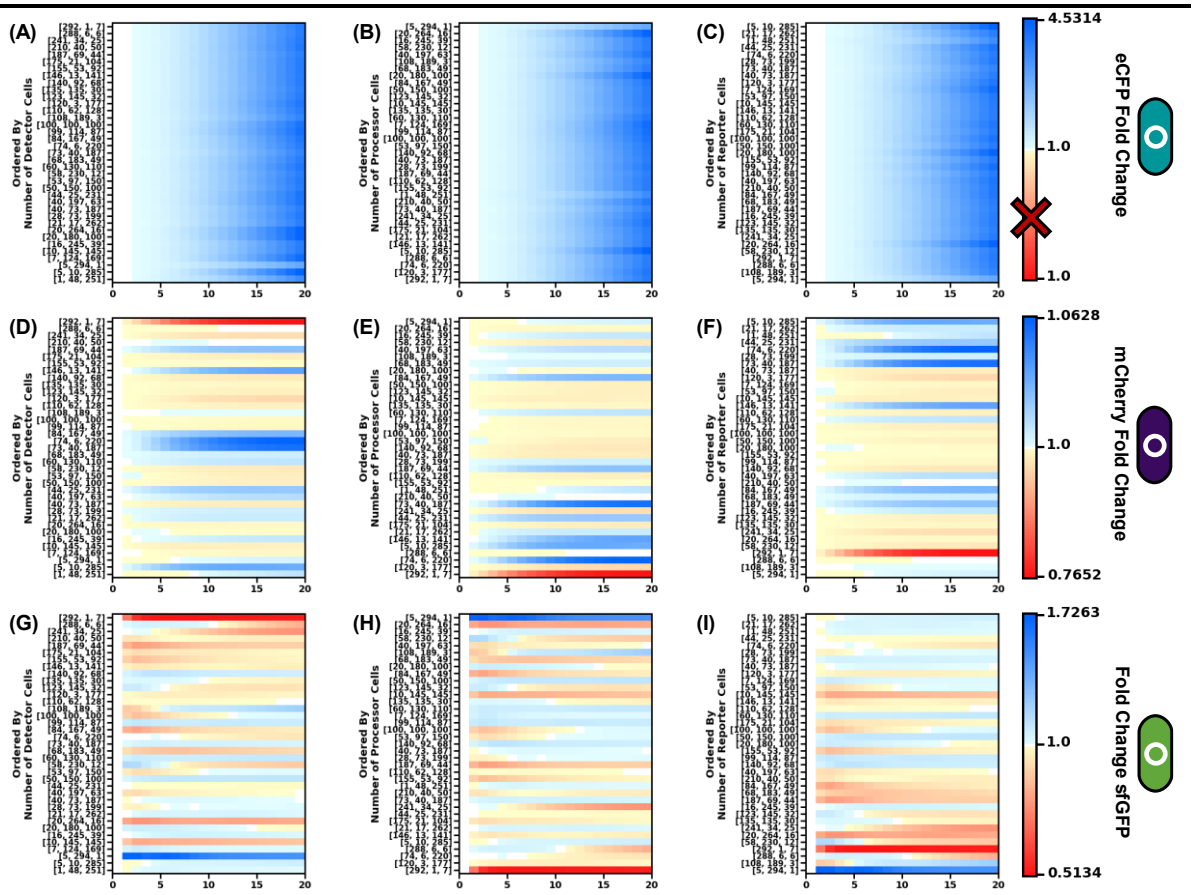


Figure 4.17. Impact of Cell Ratios on Module Behaviour

Fold change in fluorescent protein production for systems induced with 20 mM IPTG compared to uninduced systems. Each horizontal line of the lasagna plots shows the average fold change (of 4 replicates) for a system with a specific cell ratio over 20 hours. Fold change is visualised using an asymmetrical colour scale. The colour scale is centred on 1.0 (white colour), and ranges from the minimum value (red) to the maximum value (blue). A red cross (✖) indicates that no values fell within that range of the colour scale. Fold change values are for eCFP production by detector cells **(A-C)**, mCherry production by processor cells **(D-F)**, or sfGFP production by reporter cells **(G-I)**. Systems are ordered by increasing number of detector cells **(A, D, G)**, processor cells **(B, E, H)** or reporter cells **(C, F, I)**.

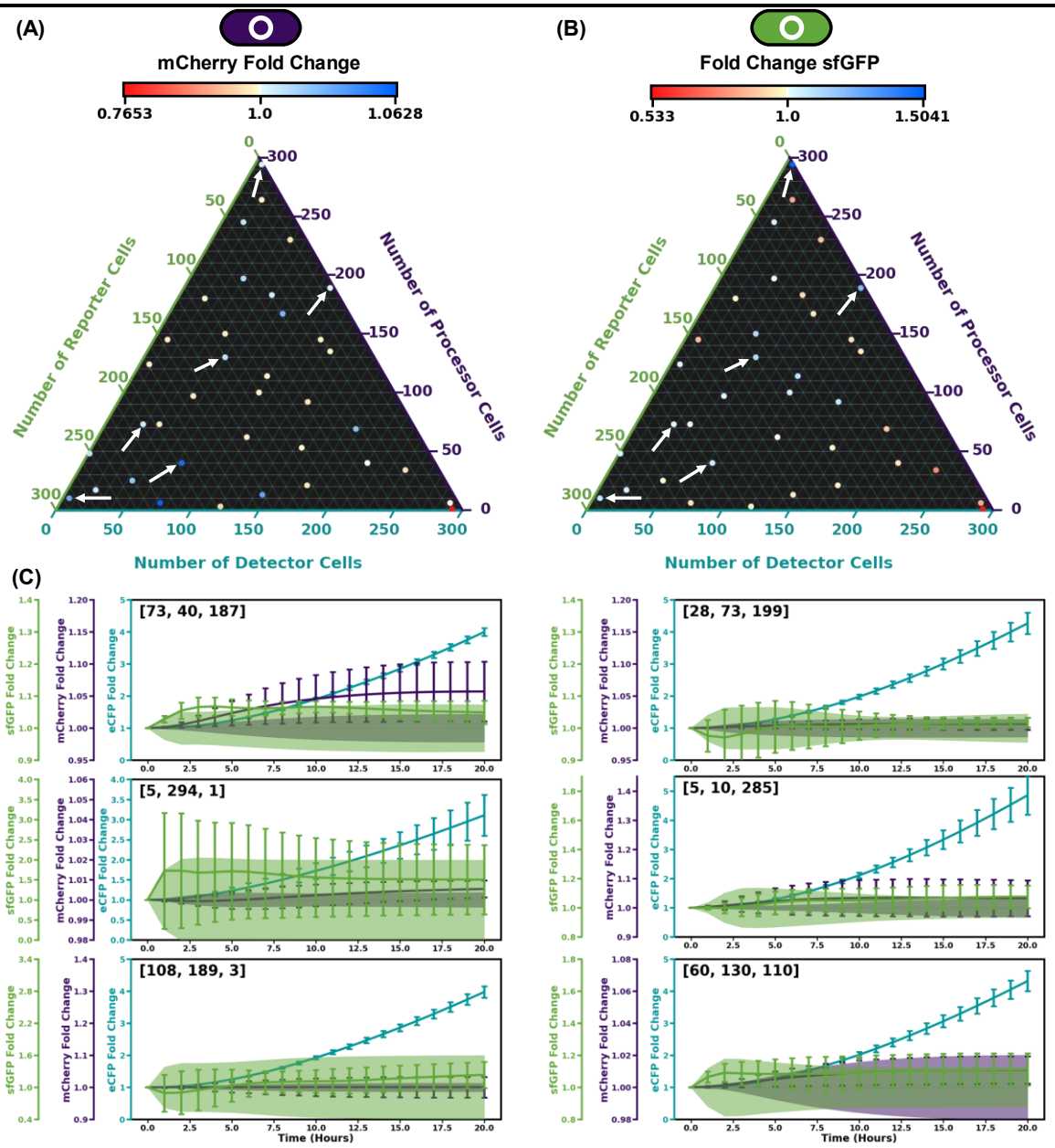


Figure 4.18. Identifying Optimal Cell Ratios

Potentially functional biosensor systems were identified using ternary plots. **(A-B)** Ternary plots showing the cell ratios used in simulations. Dots are coloured according to fold change in either mCherry (A) or sfGFP (B) production for systems induced with 20 mM IPTG relative to uninduced systems. An asymmetrical colour scale was used centred on 1.0 (white colour), ranging from the minimum value (red) to the maximum value (blue). White arrows show a selection of cell ratios which gave a positive fold change in both mCherry and sfGFP. **(C)** Fold change in eCFP, mCherry, and sfGFP production for systems with cell ratios indicated by white arrows in (A) and (B). Fold change values for each fluorescent protein were plotted on Y-axes with different scales. Error bars show standard deviation of 4 replicates centred on the mean value.

Synthetic multi-microbial communities have an easily accessible design space which is absent from systems composed of only one cell type. This design space is that of

cell ratios, where the initial proportion of each cell type in the system can be modified. By modifying the ratios of each cell type in the modular, multi-microbial biosensor may have allowed for optimisation of the system, resulting in a functional biosensor, rather than the non-functional system predicted by the agent-based model. To this end, the model presented above was simulated a number of times with different proportions of each cell type. The cell ratios were chosen to ensure the design space was explored fully, as visualised in Figure 4.16.

Results from simulations using each cell ratio are shown in Figure 4.17, where the systems were ordered by number of detector, processor, and reporter cells present. Activity was determined as the fold change in eCFP, mCherry, and sfGFP for systems induced with 20 mM of IPTG relative to uninduced systems, and visualised using an asymmetrical, 2-tone colour scale centred on a fold change of 1.0. The simulation results suggested that there was little impact of cell ratios on the behaviour of detector cells (Figure 4.17 (A-C)), but there appeared to be an effect on the activity of the processor and reporter cells (Figure 4.17 (D-I)). The detector cells were likely not impacted as their functionality was not impacted by the other cell types in the system, whereas activation of the processor and reporter cells was dependent on the amount of quorum sensing molecules produced by the other cells. Whilst the activity of processor and reporter cells appeared to be impacted by the cell ratios, there was not a noticeable correlation between the amount of each cell type and fold change in fluorescent protein production. To help visualise the data across the entire design space, the fold change values at 20 hours post induction were graphed on ternary plots, with each axis representing the amount of detector, processor, or reporter cells in the system (Figure 4.18 (A-B)). Although a noticeable trend remained absent, the ternary plots allowed for easier identification of cell ratios predicted to have a positive fold change for both mCherry and sfGFP production. Six of these cell ratios were selected, and fold change over time was plotted for eCFP (detector cells), mCherry (processor cells), and sfGFP (reporter cells) (Figure 4.18 (C)). The time course graphs showed that although there was an average increase in fluorescence protein production predicted, there was sufficient variation between repeated simulations of the same system that the fold change increases were not significant.

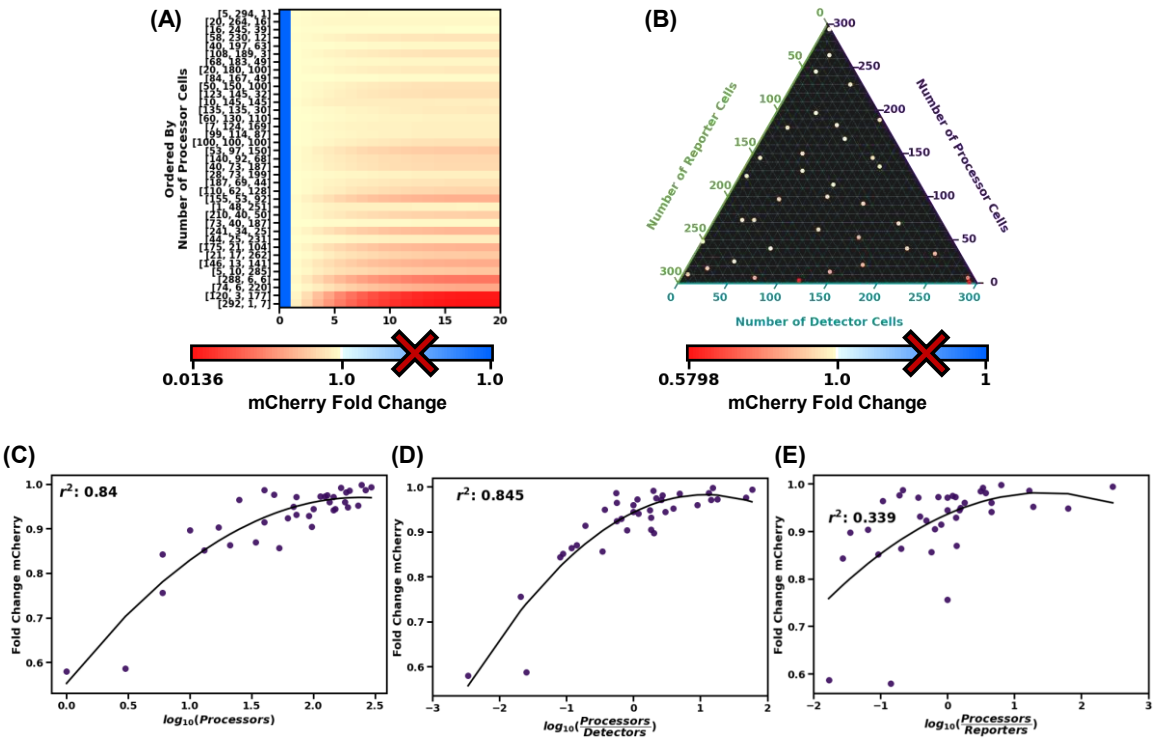


Figure 4.19. Processor Cells Activity Above Noise

Impact of cell ratios on the fold change activity for processor cells. Here, fold change was calculated at each time point using the maximum value from the uninduced systems, and the minimum value from the induced systems. For (A-B), fold change is visualised using an asymmetrical colour scale. The colour scale is centred on 1.0 (white colour), and ranges from the minimum value (red) to the maximum value (blue). A red cross (✕) indicates that no values fell within that range of the colour scale. (A) Lasagna plot showing mCherry fold change over time. Systems were ordered by the number of processor cells. (B) Ternary plot showing mCherry fold change 20 hours post induction. (C-E) Scatter plots visualising the relationship between the number of processor cells and fold change in mCherry production after 20 hours. The dependent variable was number of processor cells for (C), relative number of processors compared to the number of detectors for (D), and the number of processor cells relative to the number of reporter cells for (E). The curve of best fit was calculated as a 2nd degree polynomial curve as described in section 2.5.2. The r^2 for each curve is shown on the plots.

To help better determine module activity outside of the noise, fold change was re-calculated as before, but this time using the repeat with the highest and lowest value for the control (uninduced) and sample (induced) respectively. For both the processor and reporter cells, a positive fold change increase was not predicted for any cell ratio, however a correlation between number of cells and module activity was identified. The correlation for mCherry production by the processor cells is shown in Figure 4.19. For both the lasagna and ternary plots (Figure 4.19 (A-B)), it can be seen that smaller fold changes tended to result from systems with fewer processor cells, whereas systems with a higher number of processors exhibited fold changes closer to 1. Figure 4.19 (C) shows this trend with a 2nd degree polynomial line fitted to the data. The correlation indicates that whist there does appear to be a correlation between number of

processors and mCherry fold change, the correlation is only moderate with an r^2 of 0.84. To determine whether the number of detector and reporter cells had a confounding effect on the behaviour of the processor cells, scatter plots with fitted lines were also generated for the number of processors relative to the number of detectors and reporters against fold change in mCherry (Figure 4.19 (D-E)). This showed that the agent-based model predicted a positive correlation between the relative number of processors compared to detectors, although again the strength of this correlation was not strong (r^2 = 0.845). There was, however, no noteworthy correlation between the number of processor relative to the number of reporters and the fold change in mCherry. These observations were likely due to the unidirectional communication from detector to processor cells, which was designed to have a direct impact on the processor cells' activity, whereas the reporter cells could not directly impact the processors.

For the reporter cells, a similar trend to that observed with the processor cells was observed (Figure 4.20), whereby more reporter cells seemed to be correlated with a fold change in sfGFP closer to 1. However, this trend was less obvious from the lasagna and ternary plots (Figure 4.20 (A-B)), and the 2nd degree polynomial line of best fit for number of reporter cells against sfGFP fold change was weaker than that observed for processor cells vs mCherry fold change, with an r^2 of 0.809 (Figure 4.20 (C)). Additionally, the correlation for relative number of reporter cells (compared to the number of detector or processor cells) against sfGFP fold change was found to be weak to non-existent (Figure 4.20 (E)), which may be indicative of the fact that the reporter cells' behaviour was dependent on not only the processor cells, which directly influence the reporters' activity, but also on the detector cells which had both indirect (via activation of the processor cells) and direct (via quorum sensing cross-talk) interactions with the reporters.

The results presented throughout this sub-section indicated that there may be the potential to influence behaviour of the multi-microbial biosensor system via the design space of cell ratios. However, the agent-based model also predicted that the majority of any signal would be lost to background noise.

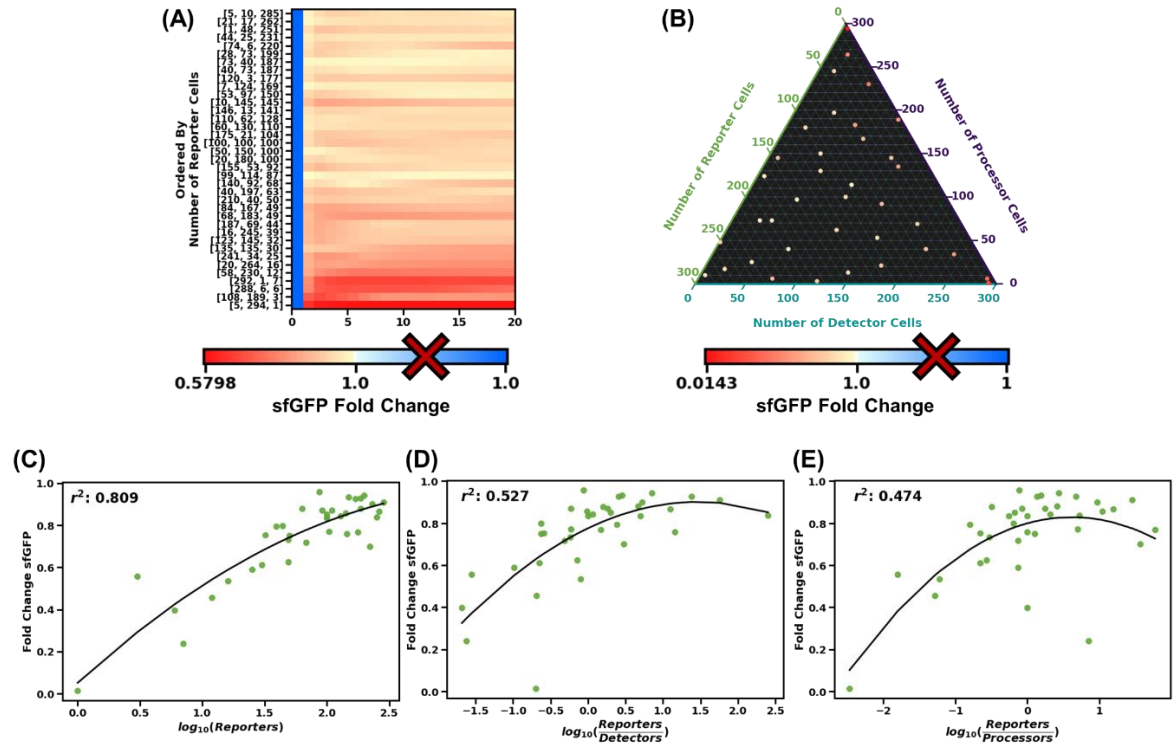


Figure 4.20. Reporter Cells Activity Above Noise

Impact of cell ratios on the fold change activity for reporter cells. Here, fold change was calculated at each time point using the maximum value from the uninduced systems, and the minimum value from the induced systems. For (A-B), fold change is visualised using an asymmetrical colour scale. The colour scale is centred on 1.0 (white colour), and ranges from the minimum value (red) to the maximum value (blue). A red cross (✖) indicates that no values fell within that range of the colour scale. **(A)** Lasagna plot showing sfGFP fold change over time. Systems were ordered by the number of reporter cells. **(B)** Ternary plot showing sfGFP fold change 20 hours post induction. **(C-E)** Scatter plots visualising the relationship between the number of reporter cells and fold change in sfGFP production after 20 hours. The dependent variable was number of reporter cells for (C), relative number of reporters compared to the number of detectors for (D), and the number of reporter cells relative to the number of processor cells for (E). The curve of best fit was calculated as a 2nd degree polynomial curve as described in section 2.5.2. The r^2 for each curve is shown on the plots.

4.5. Conclusions and Next Steps

In this chapter, modular designs for a proof-of-concept multi-microbial biosensor were presented and discussed, including engineering intercellular communication interfaces between each module. Results from computational modelling of this biosensor system indicated that individually, each module should function as expected, and discrepancies between purely deterministic and agent-based modelling of the modules were highlighted. Specifically, the agent-based model predicted slower response times to induction, which were thought to arise from inclusion of inducer diffusion both throughout the extracellular environment, and into/out of the cells. The agent-based model was also used to gain insight into potential heterogeneity between cells in monoculture. The simulation results suggested that greater heterogeneity would be apparent when systems were induced with concentrations above the sensitivity threshold, but below the saturation point, although variation between cells was observed for systems containing any amount of inducer.

Parameters used by the models were acquired through either literature or estimation, and although this is common practice within the field[77], [171], [321], the quantitative simulation results may have inaccuracies. Therefore, in order to gain more accurate predictions about the biosensor system and modules, experimental parameterisation should be completed[322]. This could include a sensitivity analysis of each parameter to determine which have the most impact on the simulation results, and hence identify the most important parameters to have accurate values[323]. Experimental parameterisation could then be accomplished with approaches including genetic algorithms, where parameter values which lead to simulation results matching most closely to experimental data could be determined[324]. The use of experimentally determined parameter values would then lead to more accurate simulation data, and hence more useful insights.

Regardless of the limitations above, results generated by simulation of the agent-based, multi-microbial biosensor model provided useful insights into functionality of the system. Simulation results predicted that whilst each module should function individually, when mixed in a co-culture the biosensor would be non-functional. This appeared to be due to background synthesis of AHL quorum sensing molecules by the detector and processor cells in the absence of any inducer. High levels of background

AHL synthesis could result in saturation of downstream module responses, and hence inducing the biosensor would have no impact. In this way, background noise could propagate through the system in the form of AHL accumulation.

The above predictions prompted exploration of the design space of cell ratios as a method of system optimisation. It was hypothesised that modifying the proportions of cells in the co-culture may allow for reduced AHL accumulation and background induction by either decreasing the amount of AHL synthesisers, increasing the number of AHL receivers to prevent saturation, or a combination of both. Whilst no functional system was predicted for any cell ratio simulated, it was found that the relative amounts of each cell had an impact on biosensor behaviour, which provided guidance for experimental optimisation of the system.

Chapter 5. Development and Validation of a Modular and Multi-Microbial Biosensor

Chapter 4 presented the design for a proof-of-concept biosensor, along with results of computational modelling to help predict behaviour of each module and the biosensor as a whole. In this chapter, results from experimental characterisation are presented and discussed. Section 5.2 focuses on characterisation of each biosensor module's behaviour in response to their canonical inducer, as well as determining potential cross talk between modules and validation that the detector and processor modules confer AHL production capabilities. Section 5.3 presents results from co-culture experiments, where propagation of noise through the system was first investigated, based on results gathered from the agent-based model, before testing the proof-of-concept modular, multi-microbial biosensor as a whole. The final section (5.4) concludes the outcomes and findings from this chapter and discusses next steps for optimisation of the proof-of-concept biosensor, which are explored further in chapter 6.

5.1. Introduction

5.1.1. Standard calibration of plate reader data

The characterisation of many synthetic biology devices and systems tend to rely heavily on data collected by a microplate reader. This is partially due to synthetic biology devices often including fluorescent proteins as markers for determining behaviour of a system by either acting as a final output of the system, or to indicate expression of another element in the system via co-transcription or -translation[294]. Using a plate reader to measure fluorescent intensity of samples allows for determination of expression, as higher expression leads to more fluorescent protein and thus an increase in fluorescent signal. For systems involving cells, measurement of optical density at 600 nm is often used as a proxy for the density of cells in a sample and to determine cellular growth, as more cells results in increased light scattering.

The units in which plate readers report data are arbitrary and not directly comparable between different instruments, leading to difficulties to reproducibility[325] These comparability issues can be mitigated to some extent by making the data relative to a control sample, however this has been shown to be sub-optimal as the quality and reproducibility of such data is highly dependent on the quality of the controls used[326].

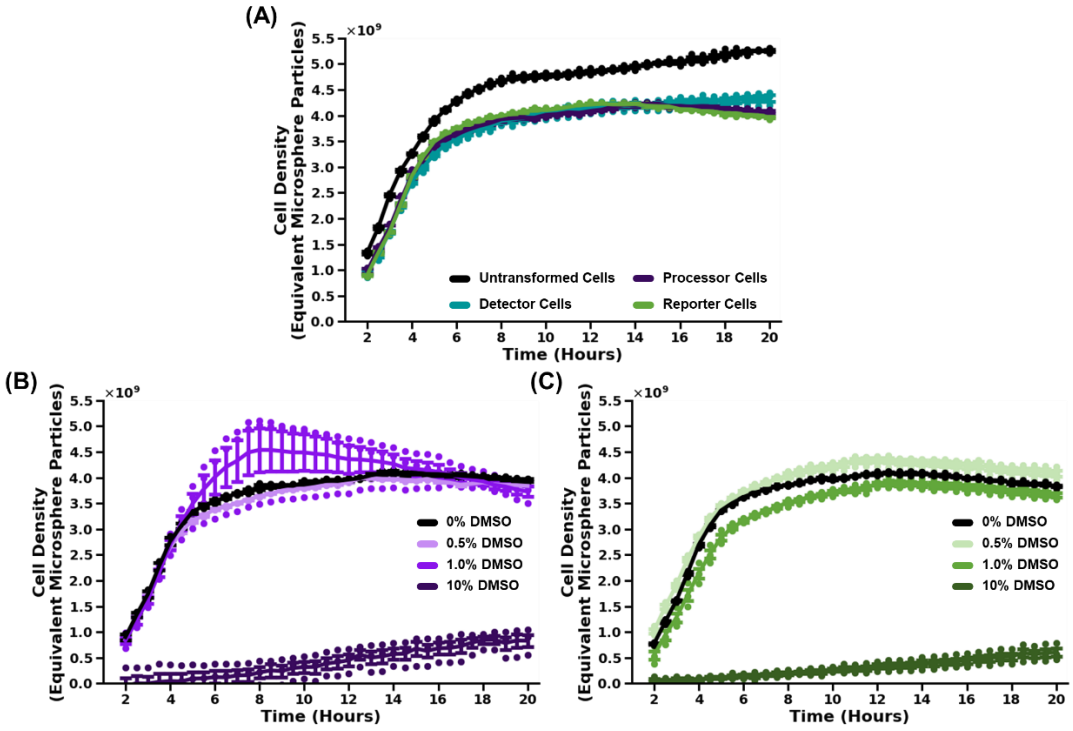


Figure 5.1. Cell Module Growth Rates

Growth rates of *E. coli* DH5α cells expressing one of the three biosensor modules. Error bars are +/- standard error of replicates centred on the mean. Individual points represent each replicate. (A) Growth curves for uninduced IPTG detector cells, default processor cells, and sfGFP reporter cells. Untransformed *E. coli* cells were included as a control. (B-C) Growth rates of default processor cells (B) and sfGFP reporter cells (C) in the presence of DMSO. Cells without DMSO as presented in (A) are shown as a control.

Additionally, the most appropriate type of controls can vary between experiments. Instead, it is possible to use standard calibrants to calibrate arbitrary units reported by a plate reader to absolute units, which can be directly compared and allows for much easier detection of failed or deviant samples and controls. There are currently standard calibrants defined for fluorescent proteins in the blue, green, and red sections of the spectrum[63]. These calibrants are cascade blue for blue fluorescent proteins, sulforhodamine-101 for red fluorescent proteins, and fluoresceine for green fluorescent proteins. By preparing serial dilutions of these calibrants and measuring fluorescence using the same settings to be used for the experimental samples, a standard curve can be determined and used to calculate a conversion factor for arbitrary units to absolute units. The absolute units are termed ‘Molecules of Equivalent Cascade Blue’ (MECB), ‘Molecules of Equivalent sulforhodamine-101’ (MESR), and ‘Molecules of Equivalent Fluorescein’ (MEFL), depending on the calibrant used. Similarly, microspheres with a radius approximately the same as the cells being measured can be used to calibrate OD600 readings, allowing for the number of cells to be reported as ‘number of equivalent microsphere particles’.

5.1.2. Flow cytometry for multi-microbial cultures

Whilst plate readers can provide bulk measurements for a sample, it is not possible to determine heterogeneity within the sample and distinguish between different cell types within a mixed-microbial system. For many experiments these bulk measurements which assume homogeneity are sufficient to determine overall behaviour of a system, however for other systems where heterogeneity needs to be measured, single-cell measurement techniques are required[327]. Flow cytometry is one such technique, where cells are flowed through a set of lasers one at a time, allowing for attributes including fluorescence of individual cells to be measured apart from other cells in the sample. This allows for different populations of cells, which exhibit different levels of fluorescence, fluorescence at different wavelengths, and different shapes, to be determined. Flow cytometry has been successfully applied to the analysis of multi-microbial communities previously, including determining the microbial composition of a natural community by the shapes of cells present[328]. There have also been studies which have shown heterogeneity in monocultures using flow cytometry, highlighting the importance of singe cell analysis[329]. However, flow cytometry is a destructive method of analysis and more expensive compared to plate readers. Therefore, plate readers are still useful for making many measurements of a system over a time course.

5.2. Initial Biosensor Module Characterisation

The behaviour of the IPTG detector, default processor, and sfGFP modules was determined by measuring fold change in expression of a fluorescent marker of induced samples relative to uninduced samples. For all experiments, the detector, processor, and reporter cell types refer to *Escherichia coli* DH5α cells transformed with the IPTG detector with CFP (cyan fluorescent protein), default processor with mCherry, or sfGFP (superfolder green fluorescent protein) reporter modules respectively. Data collected in the first 2 hours of incubation were excluded as it was routinely found that measurements were at or below the limit of accurate detection for the equipment used. In all cases, raw data was converted to absolute units and processed according to the methods presented in section 2.7.1.

5.2.1. Characterising cell growth rates

Prior to validation of module functionality, the growth rates of each cell type were measured and compared to untransformed *E. coli* DH5α cells. Each cell type was incubated in LB media overnight before being sub-cultured into fresh LB media in a 96 well microplate. Cell cultures were shake incubated at 37ºC for 20 hours in a plate reader, and absorbance readings were taken periodically to measure cell density. The complete experimental procedure is detailed in section 2.7.5.

All cells were observed to reach exponential growth phase within 2 hours (Figure 3.7 (A)), however, all three module types were found to have a slower growth rate than untransformed cells. This finding was expected, as the detector, processor, and reporter cells were likely to have a higher burden and experience increased stress compared to untransformed cells due to the inclusion of high copy number plasmids encoding additional proteins for expression. The expression of these additional proteins would have diverted resources away from normal cellular processes and hence reduce the rate at which cells were growing and dividing. Although the three cell types contained plasmids encoding different proteins, when uninduced the growth rates were found to be identical. This indicated that when in co-culture, the different cell types may also grow at similar rates which may reduce the chances of any cell type becoming outcompeted due to domination over resources by a faster growing cell type.

Both the processor and reporter cells were designed for induction by acyl-homoserine lactones (AHLs). Due to the low solubility of AHLs in water, dimethyl sulfoxide (DMSO) is commonly used as a solvent. However, it is known that DMSO can be toxic to *E. coli* cells at high amounts. Therefore, it was important to determine the impact DMSO would have on cell growth and determine the maximum percentage of DMSO which could be used. Figure 3.7 (B-C) shows the impact of adding 0.5, 1, and 10% of DMSO to processor and reporter cells. It was found that 10% DMSO severely impacted cell growth, whilst 0.5 and 1% had little overall effect. Therefore, a maximum of 1% DMSO was used for all following experiments.

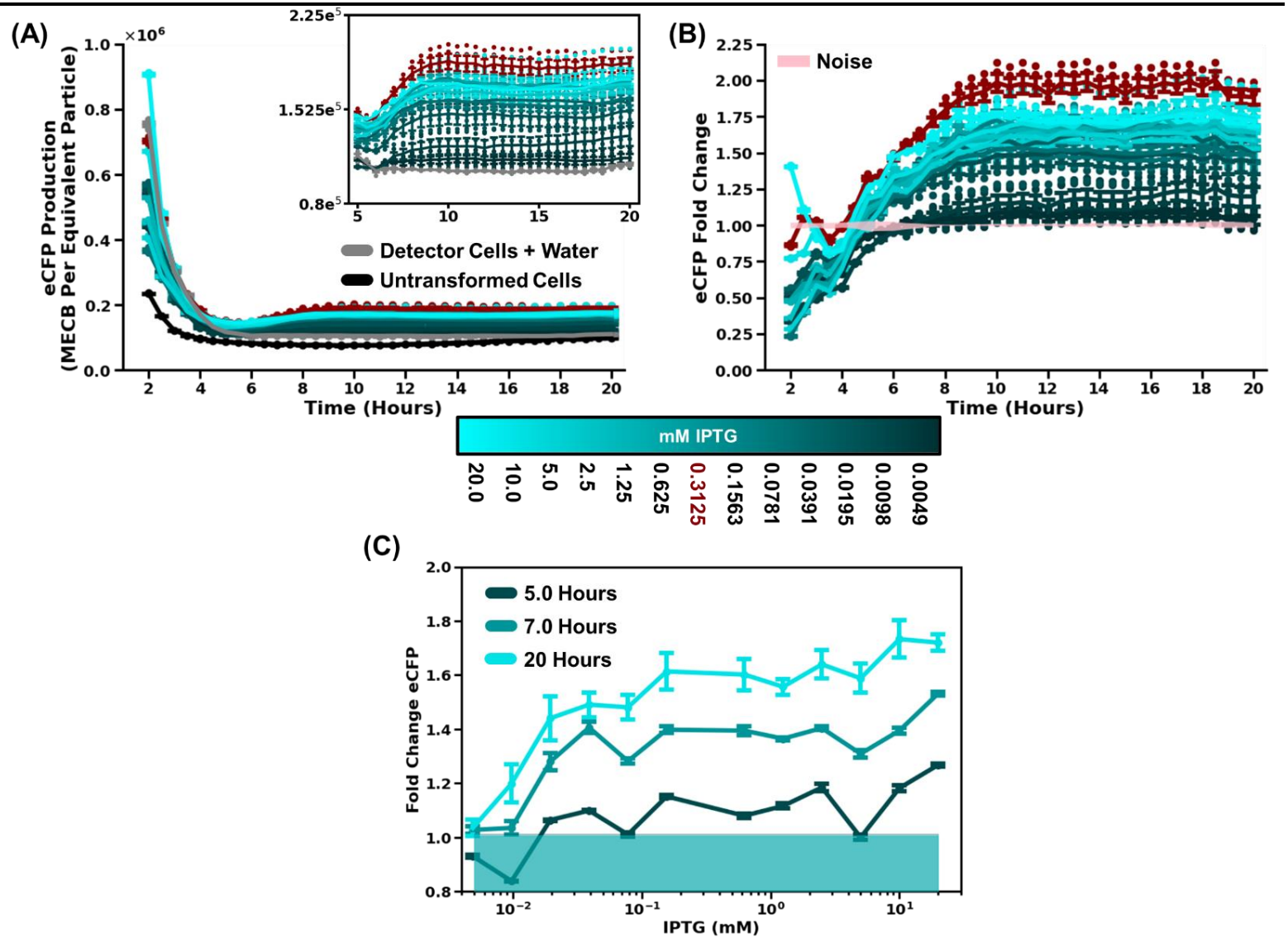

(A)
eCFP Production (MECB Per Equivalent Particle)
×10^6
Time (Hours)
Detector Cells + Water
Untransformed Cells
(B)
eCFP Fold Change
Noise
Time (Hours)
mM IPTG
20.0
10.0
5.0
2.5
1.25
0.625
0.3125
0.1563
0.0781
0.0391
0.0195
0.0098
0.0049
(C)
Fold Change eCFP
5.0 Hours
7.0 Hours
20 Hours
IPTG (mM)

Figure 5.2. IPTG Detector Module Dose-Response Characterisation

Behaviour of the IPTG detector cells when induced with IPTG. Error bars show +/- standard error centred on the mean of 3 or 4 replicates. The IPTG concentration shown in dark red indicates an outlier in the data. (A) The time course curve shows average eCFP fluorescence per cell (reported as molecules of equivalent cascade blue (MECB) per equivalent microsphere particles) over time. Autofluorescence of untransformed cells and uninduced IPTG detector cells were used as controls. Inset shows a zoomed-in portion of the plot. (B) Time course curve of fold change in eCFP fluorescence over 20 hours for induced detector cells induced with IPTG relative to uninduced cells. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (C) Dose-response curve for the detector cells at 5, 7, and 20 hours post induction. Coloured boxes show the sensitivity limit at each time point, which was calculated as a fold change of 1.0 plus standard deviation of the negative control.

5.2.2. IPTG detector module: dose-response behaviour

Following initial characterisation of each cell type's growth rate, the behaviour of each module when induced by their canonical inducer was determined. To perform initial validation of each module type, the cell types were prepared separately before using a liquid handler to sub-culture into a 96-well microplate in the presence and absence of the relevant inducer. The concentrations used for each inducer were determined based on both previously reported experiments for similar systems[151], [330], and results from both the deterministic and agent-based models presented in chapter 4. The microplate was then incubated with shaking at 37ºC, and fluorescence and absorbance readings were taken periodically to measure fold change in fluorescent marker production and cell growth. The complete experimental procedure is detailed section 2.7.6.

For the IPTG detector cells, an increase in cyan fluorescence was observed when induced with IPTG compared to uninduced cells (Figure 5.2). This indicated expected behaviour, where the presence of IPTG allowed for un-repression of the *PLac* promoter and increased expression of eCFP. As LasI was designed to be co-expressed with eCFP, it was therefore assumed that LasI expression, and thus C12-HSL synthesis, increased similarly. However, in the first 5 hours of measurement, no difference could be observed in fluorescence of induced and un-induced cells (Figure 5.2 (A)). Additionally, a decrease over time in fluorescence per cell was observed. These results indicated that eCFP production in the first 5 hours was low, and fluorescence was below the limit of detection. The decrease in fluorescence per cell also indicated that initially, cell growth was faster than eCFP production. This conclusion is in accordance with the growth curves shown in Figure 3.7 (A), as the detector cells remained in exponential growth until around 4-6 hours, after which time the increase in number of cells plateaued.

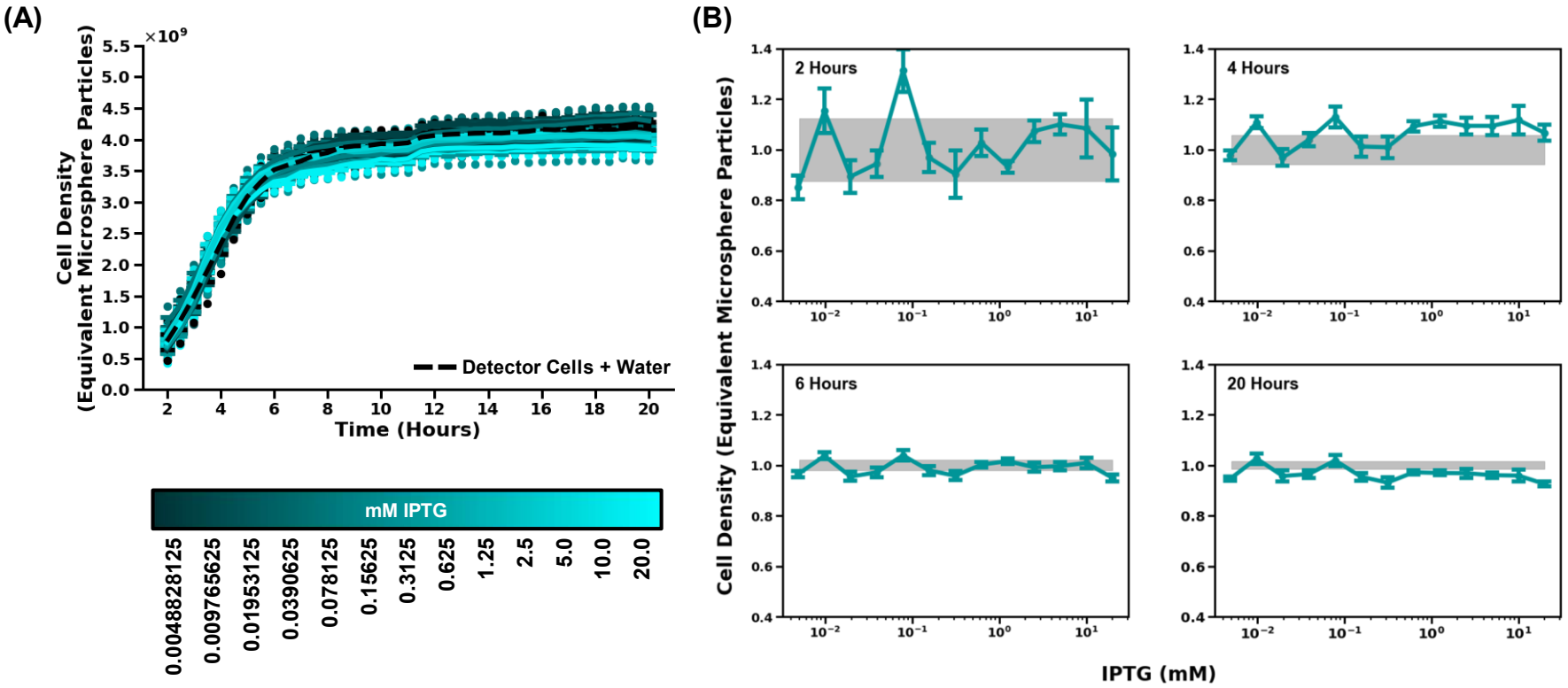


Figure 5.3. Impact of Induction on IPTG Detector Cell Growth

Data showing the impact induction at different levels has on cell growth. Error bars show standard error of 3 to 4 replicates centred on the mean. Data points show results of individual replicates. (A) Time course curve over 20 hours. Coloured lines show samples grown in the presence of an inducer. Dashed black lines show cells grown in the presence of water only. (B) Dose-response curves for detector cells, where the response was measured as cell density relative to uninduced control cells at 2, 4, 6, and 20 hours post induction. Grey boxes show +/- standard deviation of the uninduced control cells.

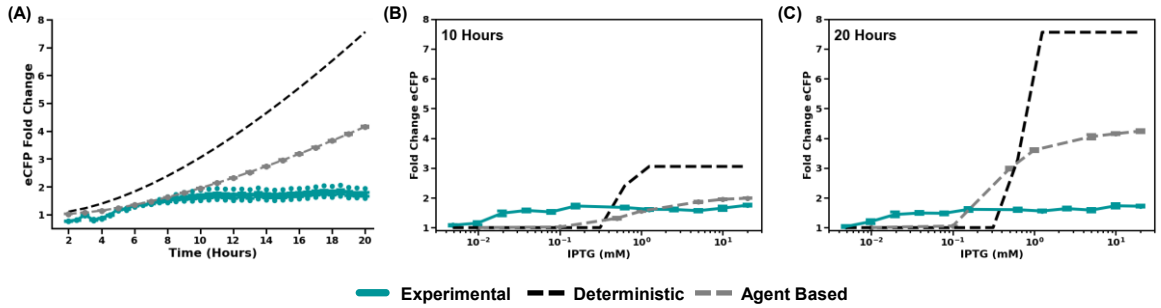


Figure 5.4. Simulated vs. Experimental Data for IPTG Detector Module
Comparison of data collected experimentally, by deterministic simulation (section 4.3), and by agent-based simulation (section 4.4). (A) Time course curve of fold change in eCFP when detector cells were induced with 10 mM IPTG. (B-C) Dose-response curve of IPTG concentration against eCFP fold change at 10 hours (B) and 20 hours (C) post induction.

Generally, an increase in concentration of IPTG correlated with an increase in eCFP production over time (Figure 5.2 (A-C)). An exception to this was 0.3125 mM IPTG, which showed fluorescence values outside of the established trend (Figure 5.2 (A-B) and section 9.10). This was assumed to be an outlier caused by either inaccurate pipetting or insufficient mixing by the automated liquid handler. Aside from this erroneous result, a clear dose-response curve had emerged by 7 hours post induction, with a sloping region (i.e., the range of concentrations at which signal correlated with dosage) between approximately 0.01 and 0.04 mM IPTG (Figure 5.2 (C)). The limit of sensitivity was lowered to 0.005 mM IPTG by 20 hours post induction, with the saturation point also decreasing to approximately 0.02 mM. From this data, it can be seen that the operational range of the IPTG detector cells was small, with high sensitivity relative to the concentrations tested. However, the dynamic range was also found to be low, with a maximum fold change in fluorescence of approximately 1.7 after 20 hours of growth.

The growth rate of IPTG detector cells in the presence of increasing IPTG concentrations were analysed to determine the impact that induction had on the overall growth rates. Although there appeared to be a slightly lower number of cells at the end of the experiment for cultures exposed to the higher concentrations of IPTG (Figure 5.3 (A)), no significant trend was identified (Figure 5.3 (B)).

The fold change in eCFP production (measured as a change in fluorescence experimentally) was compared to simulated data collected from the deterministic and agent-based models presented in chapter 4 (Figure 5.4). It was observed that both model types generally over-estimated the fold change in eCFP production. Additionally,

whilst both model types predicted that after 20 hours the fold change signal would continue to increase, experimentally the signal plateaued after approximately 7 to 8 hours (Figure 5.4 (A)). Both models predicted that the IPTG detector module would be less sensitive than observed experimentally (Figure 5.4 (B-C)).

5.2.3. Default processor module: dose-response behaviour

The experiment described in the previous sub-section was repeated for the default processor cells. mCherry fluorescence per cell increased over time, and induction with higher concentrations of C12-HSL appeared to result in higher mCherry production (Figure 5.5 (A)). Additionally, the change in mCherry production appeared to occur in three stages. Firstly, there was an increase in fluorescence in the first 4 hours, which is more obvious in cultures inducted with C12-HSL above 2 μM. After this initial increase, fluorescence remained relatively stable for an additional 8 to 10 hours, after which mCherry production appeared to begin increasing rapidly, where once again cultures induced with higher concentrations of C12-HSL exhibited higher rates of production. The uninduced sample also exhibited this behaviour; based on results from the default processor module in chapter 4, this could be due to the positive feedback loop in the processor's design, where leaky expression of *rhlI* lead to accumulation of C4-HSL and additional activation of *PLas*. This increase in fluorescence exhibited by uninduced cells seemed to be delayed compared to cells induced with higher concentrations of C12-HSL. The delayed accumulation of mCherry by uninduced processor cells compared to cells induced with concentrations of C12-HSL above 2 μM results in the fold-change signal forming a 3rd degree polynomial (Figure 5.5 (B)). When these higher concentrations of C12-HSL were used for induction, a peak in signal at around 8 to 10 hours occurred, after which a decrease in fold change was observed. This indicated that the default processor module may not be suitable for use in systems required to function for longer than 10 hours as background expression can cause increased noise and reduced signal.

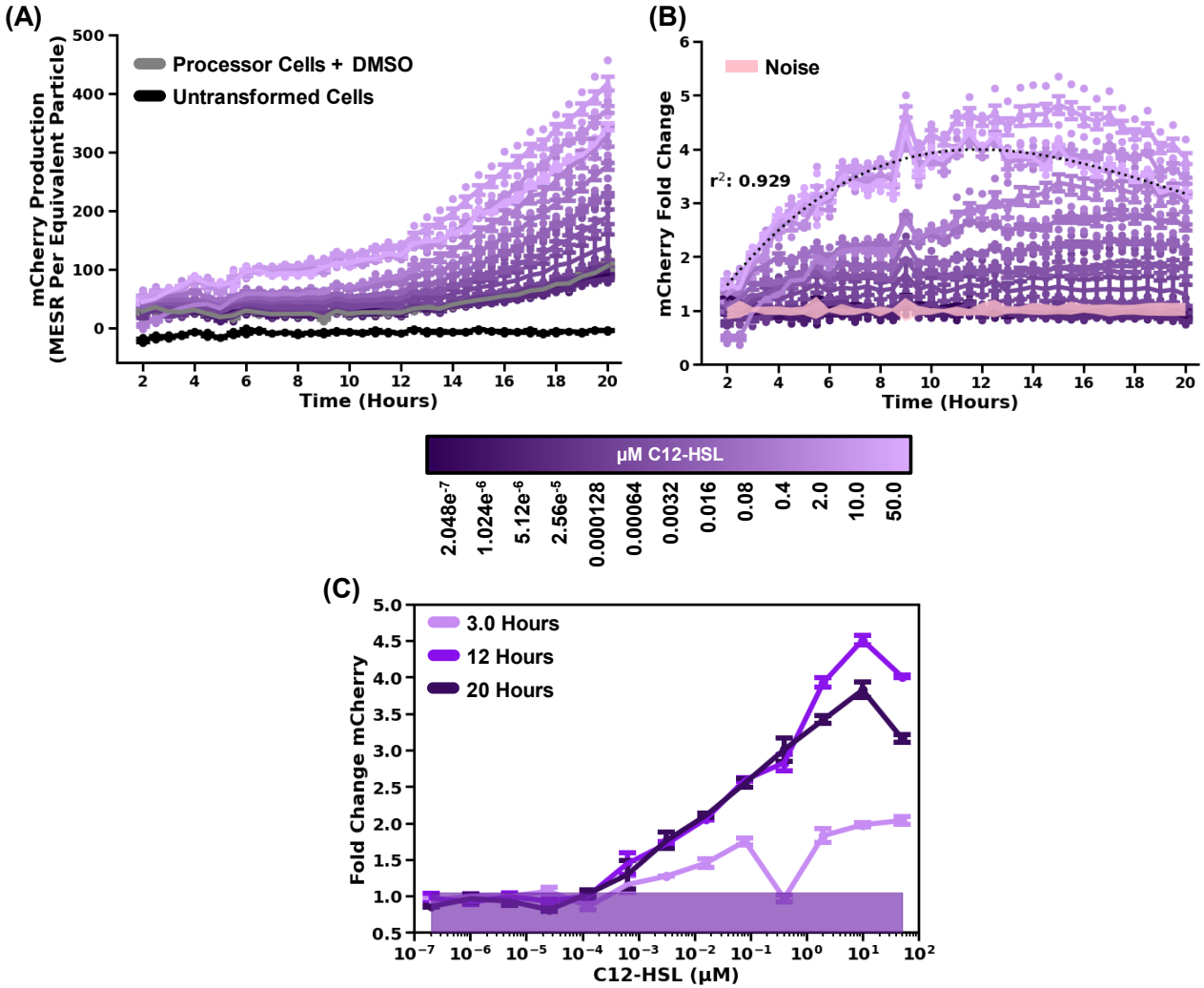


Figure 5.5. Default Processor Module Dose-Response Characterisation

Behaviour of the default processor cells when induced with C12-HSL. Error bars show +/- standard error centred on the mean of 3 or 4 replicates. (A) The time course curve shows average mCherry fluorescence per cell (reported as molecules of equivalent

sulforhodamine-101 (MESR) per equivalent microsphere particles) over time. Autofluorescence of untransformed cells and uninduced default processor cells were used as controls. (B) Time course curve of fold change in mCherry fluorescence over 20 hours for processor cells induced with C12-HSL relative to uninduced cells. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. The dotted black line is a 3rd degree polynomial curve fitted to data collected from cells induced with 50 mM C12-HSL, with an r^2 of 0.929. (C) Dose-response curve for the processor cells at 5, 10, and 20 hours post induction. Coloured boxes show the sensitivity limit at each time point, which was calculated as a fold change of 1.0 plus standard deviation of the negative control.

The sensitivity of default processor cells was determined from dose-response curves (Figure 5.5 (C)). For all curves after 5 hours post induction, the limit of sensitivity was approximately $10e^{-4}$ μM C12-HSL. At 5 hours post induction, a dynamic range of almost 3-fold was observed, with a saturation point at 10 μM C12-HSL. At 10 and 20 hours post induction, the maximal dynamic range was observed to be 4.6- and 3.75-fold respectively. When measuring fluorescence 20 hours post induction, the signal saturation point was reached after induction with approximately 0.2 μM C12-HSL, whereas the sloping region continued to 10 μM for measurements taken at 10 hours. This was likely due to the decrease in signal relative to uninduced cells after 10 hours observed in Figure 5.5 (D).

For measurements taken at both 10 and 20 hours, induction with 20 μM C12-HSL showed a notably lower fold change in fluorescence compared to 10 μM C12-HSL. The reason for this was identified by analysis of growth rates for processor cells in the presence of different C12-HSL concentrations (Figure 5.6). It was found that induction of cells with concentrations of C12-HSL above 0.02 μM resulted in reduced cell density, and a slower rate of growth, with the most dramatic impact observed for induction with 20 μM C12-HSL. As time progressed, the difference in cell density between all samples decreased as stationary phase was reached. The AHL inducers were prepared by serial dilution, which meant the total volume of DMSO added was identical for all samples. Therefore, the decrease in growth rate could not be due to DMSO toxicity, as all samples should have been affected by the presence of DMSO in the same way. Thus, there was a potential that decreased cell growth was due to increased production of mCherry and RhlI resulting in increased burden and stress on the cells. Such burden has been well documented and is known to negatively impact a system's functionality [331], [332].

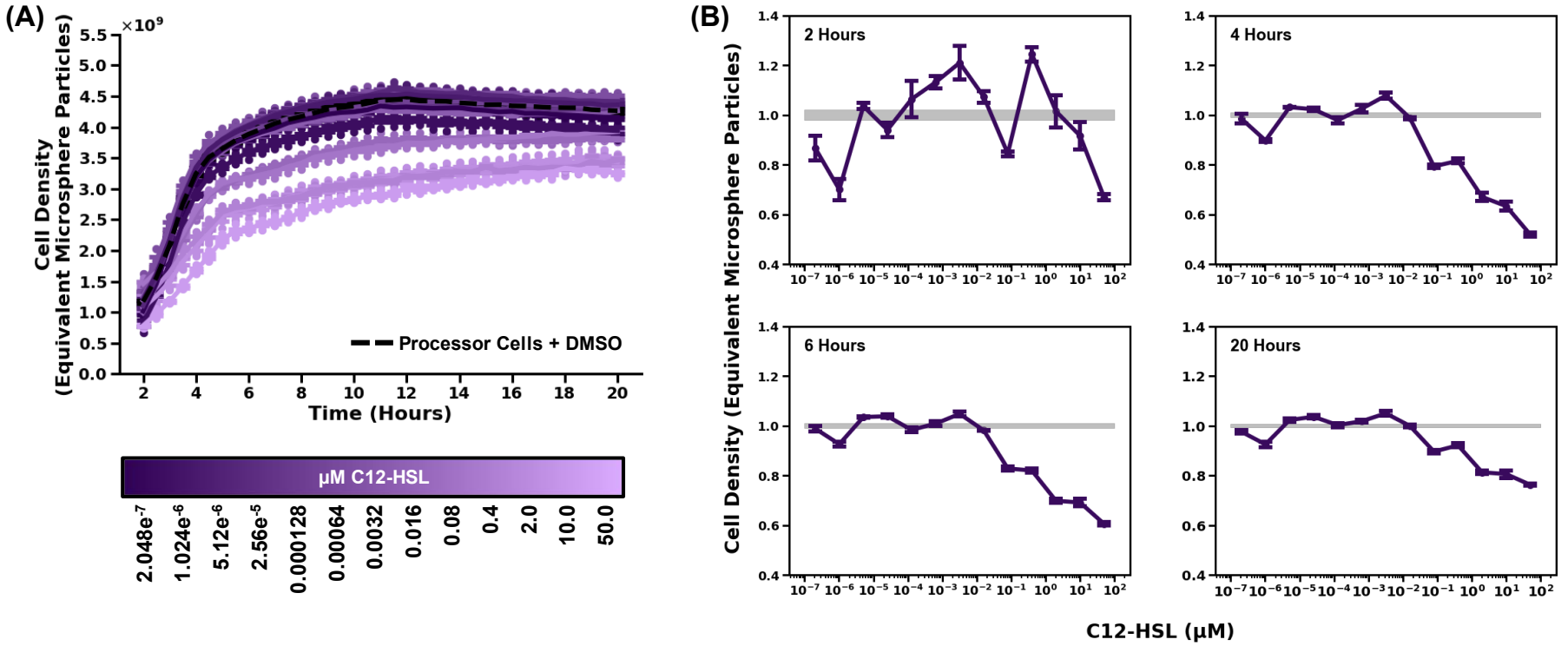


Figure 5.6. Impact of Induction on Default Processor Cell Growth

Data showing the impact induction at different levels has on cell growth. Error bars show standard error of at least 3 replicates centred on the mean. Data points show results of individual replicates. (A) Time course curve over 20 hours. Coloured lines show samples grown in the presence of C12-HSL. The dashed black line show cells grown in the presence of DMSO only. (B) Dose-response curves for processor cells, where the response was measured as cell density relative to uninduced control cells at 2, 4, 6, and 20 hours post induction. Grey boxes show +/- standard deviation of the uninduced control cells.

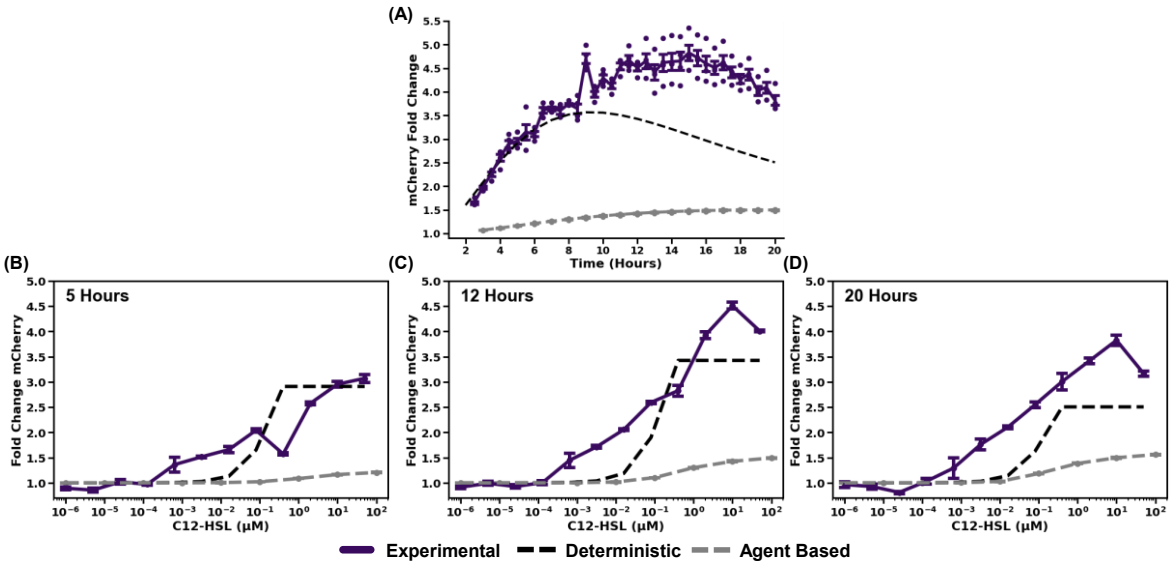


Figure 5.7. Simulated vs. Experimental Data for Default Processor Module
Comparison of data collected experimentally, by deterministic simulation (section 4.3), and by agent-based simulation (section 4.4). (A) Time course curve of fold change in mCherry when processor cells were induced with 10 μM C12-HSL. (B-D) Dose-response curve of C12-HSL concentration against mCherry fold change at 5 hours (B), 10 hours (C), and 20 hours (D) post induction.

Predictions made by the processor models in chapter 4 were compared to the experimental characterisation data to determine simulation accuracy (Figure 5.7). Whilst both model types correctly predicted that fold change in mCherry production would peak before decreasing (Figure 5.7 (A)), it was found that the agent-based model underestimated the observed fold change across the range of concentrations of C12-HSL tested (Figure 5.7(B-D)). Additionally, the agent-based model predicted that the processor module would be less sensitive than was observed experimentally. Whilst the deterministic processor model also predicted a lower sensitivity than was observed, the simulated fold change values were much closer to the observed values. The deterministic model also predicted that the peak and subsequent decrease in signal would occur earlier than was observed. Nevertheless, overall, the deterministic model appeared to be better at predicting behaviour of the processor module than the agent-based approach. This may have been due to the parameter used in the agent-based model for diffusion of C12-HSL into/out of the cells, as this parameter was largely based on assumptions and therefore may have been an underestimate. A lower diffusion parameter would have meant decreased amounts of C12-HSL passing into the cells, and hence reduced activation of LasR leading to the smaller fold change in mCherry production predicted by the model.

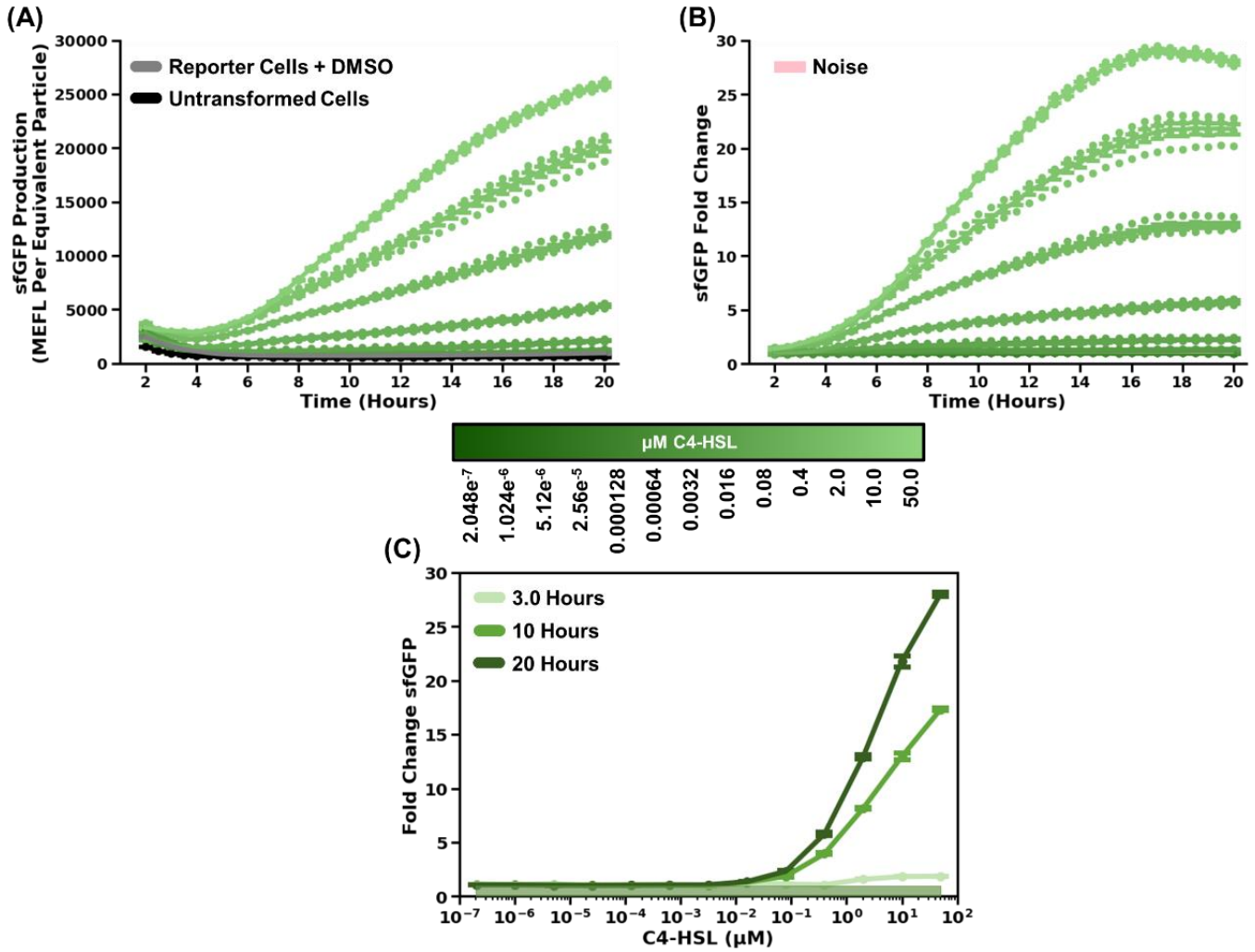


Figure 5.8. sfGFP Reporter Module Dose-Response Characterisation

Behaviour of the sfGFP reporter cells when induced with C4-HSL. Error bars show +/- standard error centred on the mean of 3 or 4 replicates. (A) The time course curve shows average sfGFP fluorescence per cell (reported as molecules of equivalent fluorescein (MEFL)

per equivalent microsphere particles) over time. Autofluorescence of untransformed cells and uninduced sfGFP reporter cells were used as controls. (B) Time course curve of fold change in sfGFP fluorescence over 20 hours for processor cells induced with C4-HSL relative to uninduced cells. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (C) Dose-response curve for the reporter cells at 3, 10, and 20 hours post induction. Coloured boxes show the sensitivity limit at each time point, which was calculated as a fold change of 1.0 plus standard deviation of the negative control.

5.2.4. sfGFP reporter module: dose-response behaviour

Behaviour of the sfGFP reporter cells was determined experimentally by measuring fluorescence and optical density over time of cultures induced with differing concentrations of C4-HSL, and in much the same way as the detector and processor cells were characterised. When induced with a sufficient concentration of C4-HSL, the reporter cells displayed increased green fluorescence, as expected Figure 5.8 (A-B). When higher concentrations of C4-HSL were used, the fold change signal was found to plateau after 16 to 18 hours; for lower concentrations of C4-HSL (below 0.4 µM), this plateau was observed earlier.

The maximal fold change observed experimentally with reporter cells was higher than seen with detector and processor cells, which was likely due to the high expression and low leakiness exhibited by the *PRhl* promoter, as discussed in chapter 4. It was also found that even 20 hours post induction with 50 µM C4-HSL, the signal saturation limit had not been reached (Figure 5.8 (C)). For the experimental setup used here, it was not feasible to add more than 50 µM C4-HSL due to solubility of the AHL in DMSO, and the toxicity of DMSO to *E. coli* cells. At 3 hours post induction, it can be seen that the system signal was saturated at a C4-HSL concentration of approximately 2 µM, however the dynamic range was much smaller than when measured at later time points (~2-fold at 3 hours compared to ~28-fold at 20 hours).

Unlike the processor cells, higher concentrations of AHL used for induction did not appear to cause any drop in signal. This indicated that induction of reporter cells with high amounts of C4-HSL did not impact cell growth in the same way that high induction of the processor cells did. Results shown in Figure 5.9 confirm this, where no significant reduction in cell density or growth rate was observed with increasing C4-HSL concentrations.

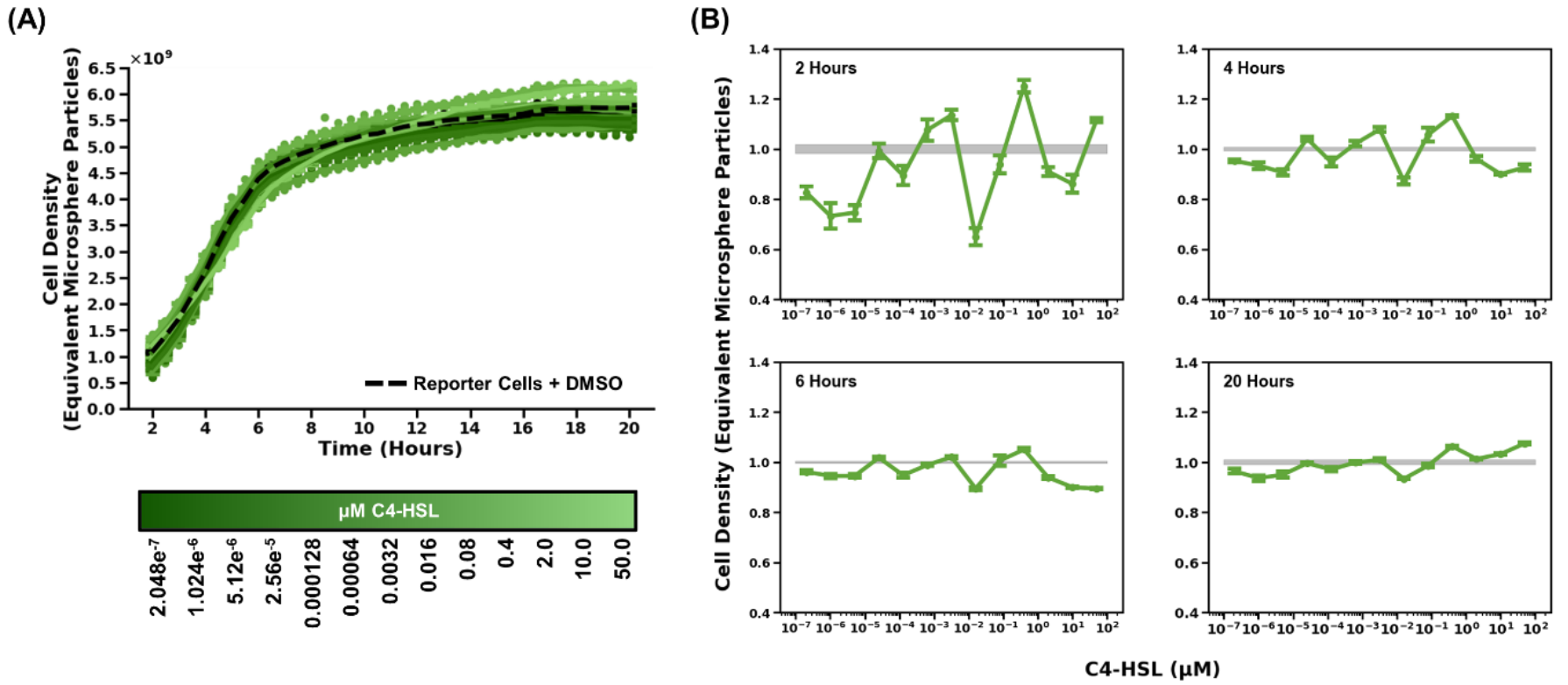


Figure 5.9. Impact of Induction on sfGFP Reporter Cell Growth

Data showing the impact induction at different levels has on cell growth. Error bars show standard error of 3 to 4 replicates centred on the mean. (A) Time course curve over 20 hours. Coloured lines show samples grown in the presence of C12-HSL. The dashed black line show cells grown in the presence of DMSO only. Data points show results of individual replicates. (B) Dose-response curves for processor cells, where the response was measured as cell density relative to uninduced control cells at 2, 4, 6, and 20 hours post induction. Grey boxes show +/- standard deviation of the uninduced control cells.

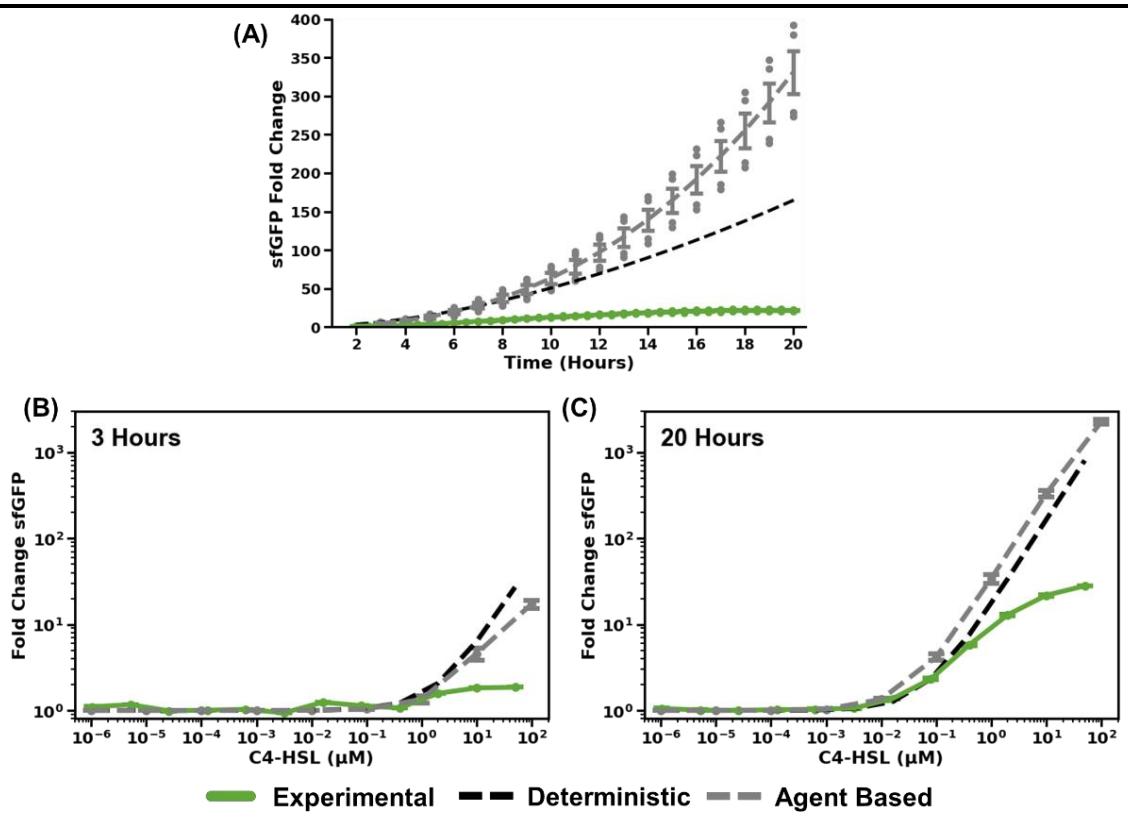


Figure 5.10. Simulated vs. Experimental Data for sfGFP Reporter Module
Comparison of data collected experimentally, by deterministic simulation (section 4.3), and by agent-based simulation (section 4.4). (A) Time course curve of fold change in sfGFP when reporter cells were induced with 10 μM C4-HSL. (B-C) Dose-response curve of C4-HSL concentration against mCherry fold change on a $\log_{10}$ scale at 3 hours (B) and 20 hours (C) post induction.

Both model types presented in chapter 4 predicted that the sfGFP reporter module would show the highest levels of fold change in fluorescent protein production. Whilst this prediction was accurate, the deterministic and agent-based reporter modules predicted a far higher maximal fold change than was observed experimentally (approximately 1000- to 2000- fold compared to below than 30-fold) (Figure 5.10). Additionally, it was observed that sfGFP fold change had stopped increasing by 20 hours post induction, whilst the models predicted that the signal would continue to increase. However, despite the large differences in predicted and observed fold change values, the dose response curves predicted by both models showed similarities to the observed curve, with almost identical sensitivity limits at different time points (Figure 5.10 (B-C)), and the lack of a definitive saturation point 20 hours post induction (Figure 5.10 (C) and Figure 5.8 (C)), although there it appears that the saturation point may be approaching at 50 μM C4-HSL induction, as can be seen more easily when fold change was plotted on a log scale as shown in Figure 5.10 (C). Additionally, a saturation point was observed experimentally at 3 hours post induction, which was absent in either of the models. This suggests that whilst the models were

accurate in their prediction of sensitivity, parameters related to the kinetics of sfGFP production over time were inaccurate, especially with regard to the overall fold change compared to uninduced samples. The higher fold change values were likely due to either an underestimation of background noise exhibited by the *PRhl* promoter in the reporter module design, or an overestimation of expression levels from activated *PRhl*.

5.2.5. Cross talk between module inducers

Whilst C12-HSL and C4-HSL were selected as the intercellular chemical communication molecules due to reports of high orthogonality, cross talk has been previously reported as discussed in chapter 4. Additionally, although IPTG is not known to interact in any meaningful way with any of the genetic elements or proteins used for the processor or reporter modules, it was important to ensure that IPTG does not have any impact on the signal or behaviour of the processor and reporter cells. This would ensure that any signal produced by the processor or reporter cells when in co-culture with detector cells in the presence of IPTG was due to cellular communication, and not direct interaction with IPTG. Therefore, cross talk characterisation was performed for all three modules with IPTG, C12-HSL, and C4-HSL to (i) determine the level of communication between the detector and reporter cells, (ii) help further characterise the self-induction behaviour thought to occur in the processor cells, and (iii) validate that IPTG does not impact behaviour of the processor and reporter cells.

When IPTG was used as the inducer, only the detector cells were found to be activated (Figure 5.11 (A, D G)). This confirmed that IPTG could not directly cause induction of the processor or reporter cells, and hence any activation observed during co-culture experiments was likely to be the result of uni-directional communication from the detector to processor to reporter cells. C12-HSL caused activation of the processor cells as seen previously (Figure 5.11 (E)), and also caused very low-level activation of the reporter cells, although the fold change magnitude did not appear to be dose dependent and hence may have been within natural noise and variation (Figure 5.11 (H)). Unexpectedly, C12-HSL was observed to cause activation of the detector cells in a dose-dependent manner, with 10 µM C12-HSL resulting in almost 1.1-fold increase at 20 hours (Figure 5.11 (B)). However, as these values were very low, it was once again thought to be within variation between samples.

When C4-HSL was used to induce the detector cells, very low-level activation of the detector cells was also observed (Figure 5.11 (C)). The reporter cells induced with C4-HSL showed behaviour in-line with results from the previous sub-section (Figure 5.11 (I)), however the processor cells showed no activation (Figure 5.11 (F)). From the deterministic processor model in chapter 4, it was predicted that although non-specific activation of LasR with C4-HSL could occur, no signal would be observed due to the presence of background levels of C4-HSL produced by RhlI. Therefore, this observation was in line with previous predictions.

The growth rates of all three cell types when grown in the presence of each inducer were visualised to determine whether there were any growth rate effects (Figure 5.12). It was found that for almost all cases, no impact on cell growth was observed. The only exception was for default processor cells in the presence of C12-HSL (Figure 5.12 (B)). This was in-line with the results discussed previously, where the growth rate of processor cells was found to be negatively correlated with induction by C12-HSL at increasing concentrations. Observations that neither the detector nor reporter cells were negatively impacted by the presence of C12-HSL, and that the processor's growth remained unaffected by the addition of IPTG and C4-HSL, lent credibility to the hypothesis that induction of processor cells with C12-HSL was placing burden onto the cells. If the decreased growth rate was due to another factor, such as C12-HSL toxicity, this behaviour would be expected in cells other than the processors, like the IPTG detector and sfGFP reporter cells.

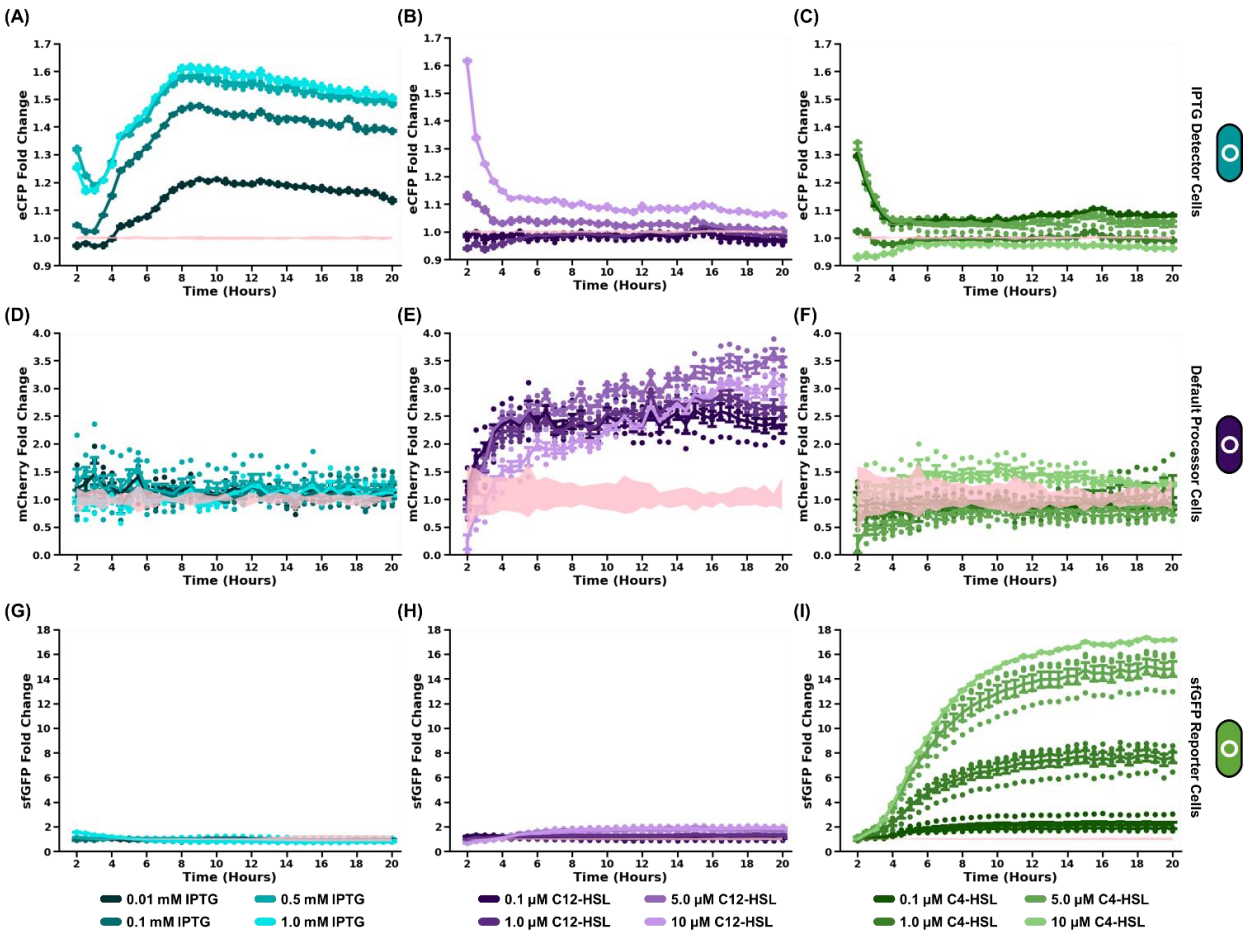


Figure 5.11. Characterising Module Cross Talk

Behaviour of the IPTG detector, default processor, and sfGFP reporter modules was characterised when induced with IPTG, C12-HSL, and C4-HSL. Autofluorescence of untransformed cells and background fluorescence of uninduced cells were used as controls.

Uninduced cells were grown in the presence of water (the solvent used for IPTG) or DMSO (the solvent used for AHLs). The plots show time course curve for fold change in fluorescence of induced cells relative to non-induced cells. Error bars show +/- standard error centred on the mean of 3 or 4 replicates, except for reporter cells induced with 10 μM C4-HSL, which had 2 replicates. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells.

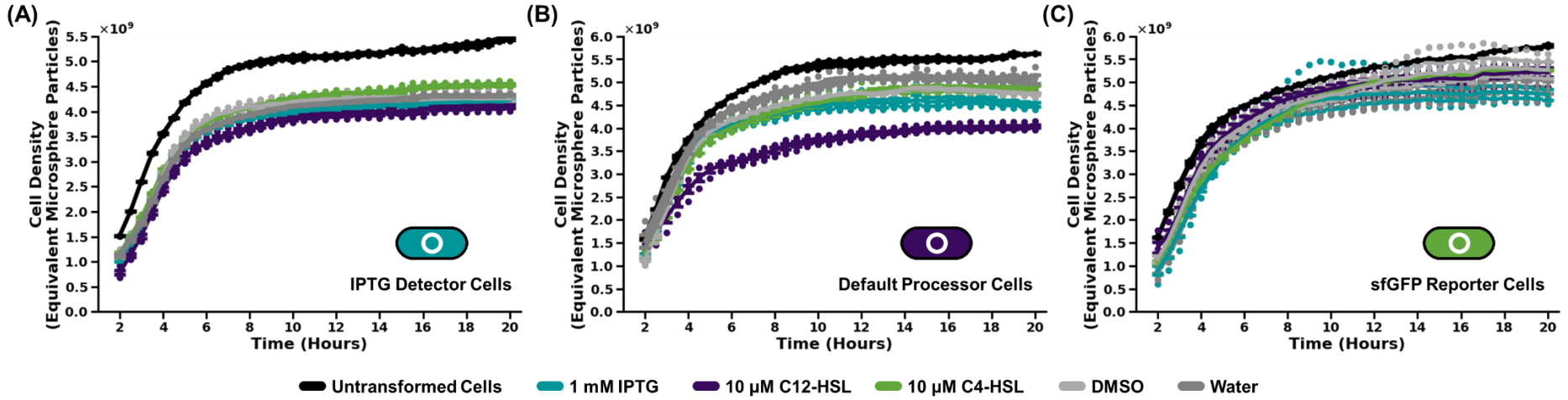


Figure 5.12. Impact of Inducers on Cell Growth

The impact of all three inducers, IPTG (1 mM), C12-HSL (10 µM), and C4-HSL (10 µM) on the growth rates of (A) IPTG detector, (B) default processor, and (C) sfGFP reporter cells. The growth rates of untransformed cells and each cell type in the presence of water and DMSO are shown as controls. Error bars show +/- standard error centred on the mean of at least 3.

5.3. Initial Multi-Microbial Biosensor Characterisation

5.3.1. Validating quorum sensing based intercellular communication

In section 5.2, the responsiveness of each module to the presence of an inducer was described. Module response was measured as a fold change in fluorescence, which indicated expression of a fluorescent protein. For the detector and processor modules, the fluorescent protein was co-transcribed along with the AHL synthetase (LasI for the detector module and RhlI for the processor module), and so the detection of a fluorescent signal suggested that the AHL synthetase was also being expressed. However, it was possible that the AHL synthetase was not present due to issues with translation, or that the enzyme was non-functional and hence unable to synthesise the AHL molecule required for cell-to-cell communication. Therefore, it was necessary to confirm AHL production by the detector and processor cells. As the processor and reporter modules had been confirmed to respond in the presence of C12-HSL and C4-HSL respectively, it was possible to utilise these modules as biosensors for AHL detection.

The confirmation of production of AHLs by the detector and processor cells was initially planned to be visual. To achieve this, uninduced processor and reporter cells were spread onto LB agar plates. To the processor cell plates, 10 µL of supernatant from detector and untransformed cells induced with IPTG or water were spotted onto the centre of the plates. For the reporter cell plates, supernatant from processor and untransformed cells induced with C4-HSL or DMSO were used. In both cases, the untransformed cells were used as a negative control which should not have produced any AHL. The agar plates were then incubated for 18 hours at 37^{o}C. The reporter cell plates were visualised under ultraviolet light and imaged with a UV filter in order to detect green fluorescence. It was expected that plates with supernatant from the processor cells and the positive control plates with C4-HSL added should show green fluorescence spreading from the centre, but the untransformed cells should show no fluorescence. This expectation was met (Figure 5.13 (A)), validating that the processor cells had synthesised C4-HSL which diffused into the extracellular environment.

The processor cell plates were visualised under blue/green light and imaged using a filter for detection of red fluorescence. With the processor cell plates, no difference in red fluorescence was observed across all plates, and no ‘halo’ could be seen

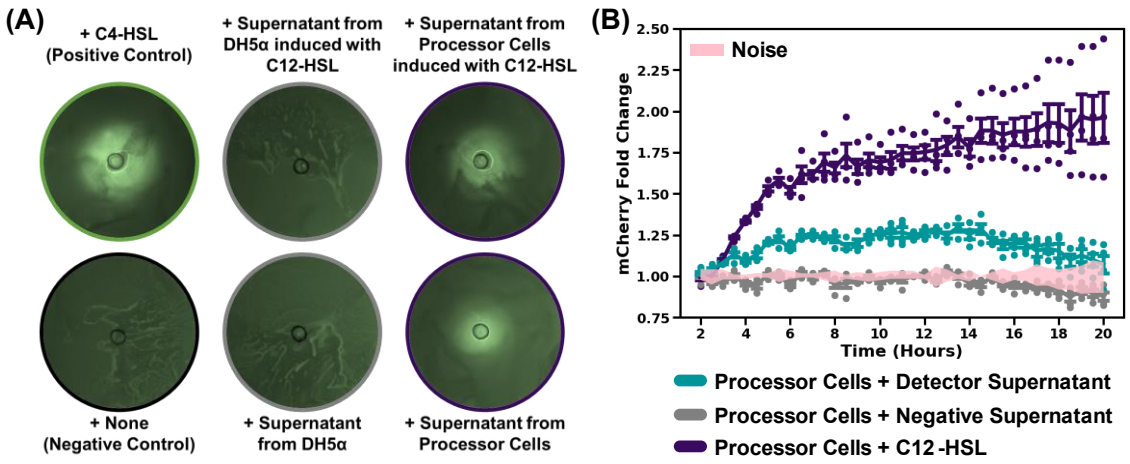


Figure 5.13. Production of AHLs by Detector and Processor Cells

Detector and processor cells were tested for production of AHLs. (A) C4-HSL production by processor cells. Uninduced sfGFP reporter cells were spread onto LB agar plates, and 10 µL of supernatant from untransformed or default processor cells either induced with C12-HSL or uninduced were spotted onto the plate's centre. Plates with C4-HSL and nothing added were used as positive and negative controls. Following incubation overnight at 37°C, the plates were visualised under UV light. Bright green halos appeared in the presence of supernatant from processor cells, but not untransformed cells. Images were false-coloured green, originals can be found in section 9.5. (B) C12-HSL production by detector cells. Uninduced processor cells were added to a 96 well microplate and induced with supernatant from detector and untransformed cells previously induced with IPTG. Processor cells were also induced with 10 µM C12-HSL and DMSO as positive and negative controls. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells.

expanding from the centre of any plates, including the positive control plate with C12-HSL added (images not shown). It was possible that no difference in fluorescence could be seen due to processor cells exhibiting lower magnitude fold change values compared to reporter cells, as seen in section 5.2. Therefore, a different approach was taken to determine if detector cells were synthesising C12-HSL. Uninduced processor cells were added to LB media in wells of a 96-well plate, to which 1 µL of either DMSO, C12-HSL, or supernatant from detector or untransformed cells induced previously induced with IPTG was added (section 2.7.8). The processor cells were then incubated with shaking at 37°C for 20 hours, during which red fluorescence and optical density measurements were taken. From this experiment, it was observed that processor cells grown in the presence of detector cells induced with IPTG showed increased red fluorescence compared to cells grown with either C12-HSL or supernatant from untransformed cells (Figure 5.13 (B)). The results therefore suggested that the detector cells were producing C12-HSL, which accumulated in the media.

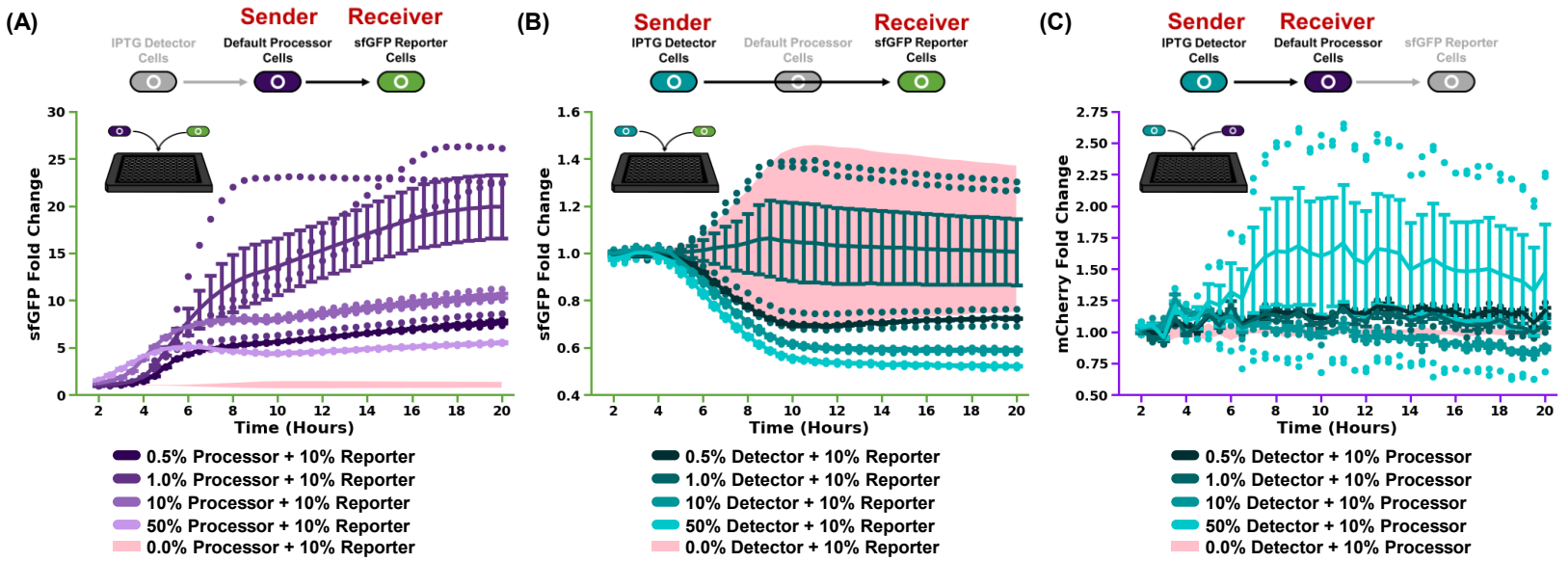


Figure 5.14. Propagation of Noise During Intercellular Communication

All module types were co-cultured in 2-population combinations. The sender cells (detector and processor) were not induced to determine background activation of receiver cells (processor and reporter). Error bars show +/- standard error centred on the mean of 3 or 4 replicates. Fold change was calculated as fluorescence of co-cultures relative to fluorescence from the receiver cells in monoculture. Receiver background noise (pink area) was calculated as the maximum and minimum calibrated fluorescence values from receiver cells in monoculture at each time point, relative to the mean fluorescence for all uninduced cells. (A) Processor and reporter cell co-cultures. (B) Detector and reporter cell co-cultures. (C) Detector and processor cell co-cultures.

5.3.2. Measuring noise propagation

The agent-based model presented in chapter 4 predicted that propagation of background noise may result in a non-functional biosensor. In biological systems, noise can propagate as a result of leaky expression. In the case of the biosensor system, background production of the AHL synthetases can lead to accumulation of AHLs, which are able to activate downstream modules. To determine the accuracy of this prediction, the ability for uninduced detector cells to activate processor cells, and uninduced processor cells to activate reporter cells, was investigated. In a 96 well plate, detector cells diluted to an OD_{600} of 1.0 were added to processor and reporter cells in at percentages of 0, 0.5, 1.0, 10, and 50 % v/v (volume of cells to final culture volume). Similarly, processor cells diluted to an OD_{600} of 1.0 were added to reporter cells in the percentages. For all co-cultures, the receiver cells (reporters or processors) were added at 1% v/v. The co-cultures were incubated in a 96-well plate at 37^{o}C with shaking for 20 hours. Fluorescence and absorbance measurements were made at regular intervals. It should be noted that in previous experiments, fluorescence readings were corrected based on cell density to get an average fluorescence per cell. For the co-cultures, this was not possible as there was no method to determine how many of each cell type was present at each time step. Therefore, the fluorescence values presented here were for the entire culture.

When co-cultured with processor cells, sfGFP reporter cells showed increased fluorescence compared to uninduced monoculture (Figure 5.14 (A)). This indicated that background production of C4-HSL by the processor cells was sufficient to activate the reporter cells. Within the first 5 hours of culture, fluorescence fold change appeared to be correlated with the number of processor cells present, and an ‘S’-like dose-response curve could be identified (Figure 5.15 (A)). This was not unexpected as more processor cells should have resulted in higher amounts of C4-HSL present, and hence increased sfGFP expression as seen previously. However, as time progressed, this correlation was lost. This may have been due to changes in the proportions of processor and reporter cells present if one cell type was able to outcompete the other. If this were the case then it was possible that as the stationary phase of growth approached (which was observed to occur between 5 and 6 hours in previous experiments), the proportion of processor and reporter cells became similar across all samples, no matter the starting amounts. Another possibility was that beyond 5 hours

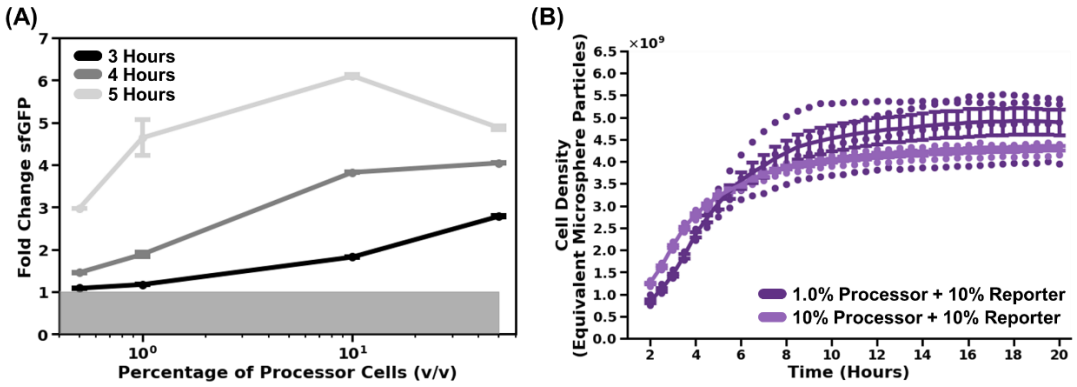


Figure 5.15. Visualising Behavioural Features of Processor and Reporter Co-Cultures

The plots here visualise features of interest from the processor and reporter co-culture experiment. (A) Dose-response curve of processor cell percentage (v/v for volume of processor cells to final culture volume) and fold change in sfGFP fluorescence by the reporter cells in co-culture, relative to uninduced reporter cells in monoculture. Curves are shown for 3 time points. Coloured box shows limit of detection, calculated as 1 + standard deviation of uninduced reporter cells in monoculture. Error bars show standard error of 3 or 4 replicates centred on the mean value. (B) Growth curves for two processor-reporter co-cultures. Error bars show standard error of 3 or 4 replicates centred on the mean value. Individual points show cell density measurements for each replicate.

background production of C4-HSL by processor cells in all samples had reached levels which were saturating the reporter cells' response. There was also the possibility that the initial dose-response observed in the first 5 hours was an artifact from experimental setup, whereby C4-HSL was present in the media of the processor cells added to the initial cultures, and hence the addition of increasing volumes of processor cells unintentionally introduced different concentrations of AHL, which caused the initial response seen. Measures were taken to avoid this possibility, namely removing the liquid media used to culture the cells in overnight and then washing the cells in sterile water to remove any excess media, and hence any AHL present. However, it still remained a possibility that a non-significant amount of AHL was retained.

Another feature of note in the time course data was that the sample containing 1% v/v of processor cells showed a large degree of variation, with all 4 repeats showing distinct time course curves. Each repeat was prepared in bulk before aliquoting into separate wells of the microplate, which should have removed the chance of each replicate containing different amounts of each cell type. However, as the cell growth curves also show similar variation (Figure 5.15 (B)), and this behaviour was not identified in other samples, error during experimental setup (such as improper mixing by the liquid handling robot of the bulk culture) seemed likely.

The presence of detector cells did not appear to result in any increased sfGFP expression by reporter cells (Figure 5.14 (B)), as expected based on results from crosstalk experiments (Figure 5.11 (bottom middle)), where it was concluded that C12-HSL had no noticeable impact on reporter cell behaviour. However, increased percentages of detector cells did seem to cause a decrease in sfGFP production by reporter cells after 5 hours of co-culture. This may have been due to detector cells managing to outcompete reporter cells over time, and hence less reporter cells were present than in cultures with less detector cells.

When detector and processor cells were co-cultured, in the majority of samples no increase in processor cell fluorescence was observed Figure 5.14 (C). As it was previously confirmed that detector cells were capable of producing C12-HSL, and processor cells were induced by the AHL, it was suspected that the uninduced detector cells simply produced C12-HSL below the processor cells' limit of sensitivity. This would have been in opposition to the agent-based model predictions in chapter 4. Another possibility was that one of the cell types were outcompeted significantly by the other during co-culture. As it was found that the default processor cells appeared to have decreased growth rates when induced, it was possible that the presence of C12-HSL produced by the detector cells resulted in slower growing processor cells, allowing for the amount of detector cells to increase. This would have led to even higher amounts of AHL, repressing processor growth further and so-on, such that not enough processor cells were present to generate a detectable signal. From the plate reader data, it was not possible to determine if this was the case, as growth curves of each cell type could not be obtained. It should be noted that 2 of the 50% detector + 10% processor co-cultures did show increased red fluorescence over time. However, as with the high-variation culture identified in the processor-reporter co-cultures, it was possible that this was simply a result of experimental setup, where more processor cells were present in these replicates than the remaining two. This possibility is supported by lower fluorescence in these other two replicates, suggesting an un-equal distribution of processor cells and poor mixing prior to replicate aliquoting by the liquid handling robot.

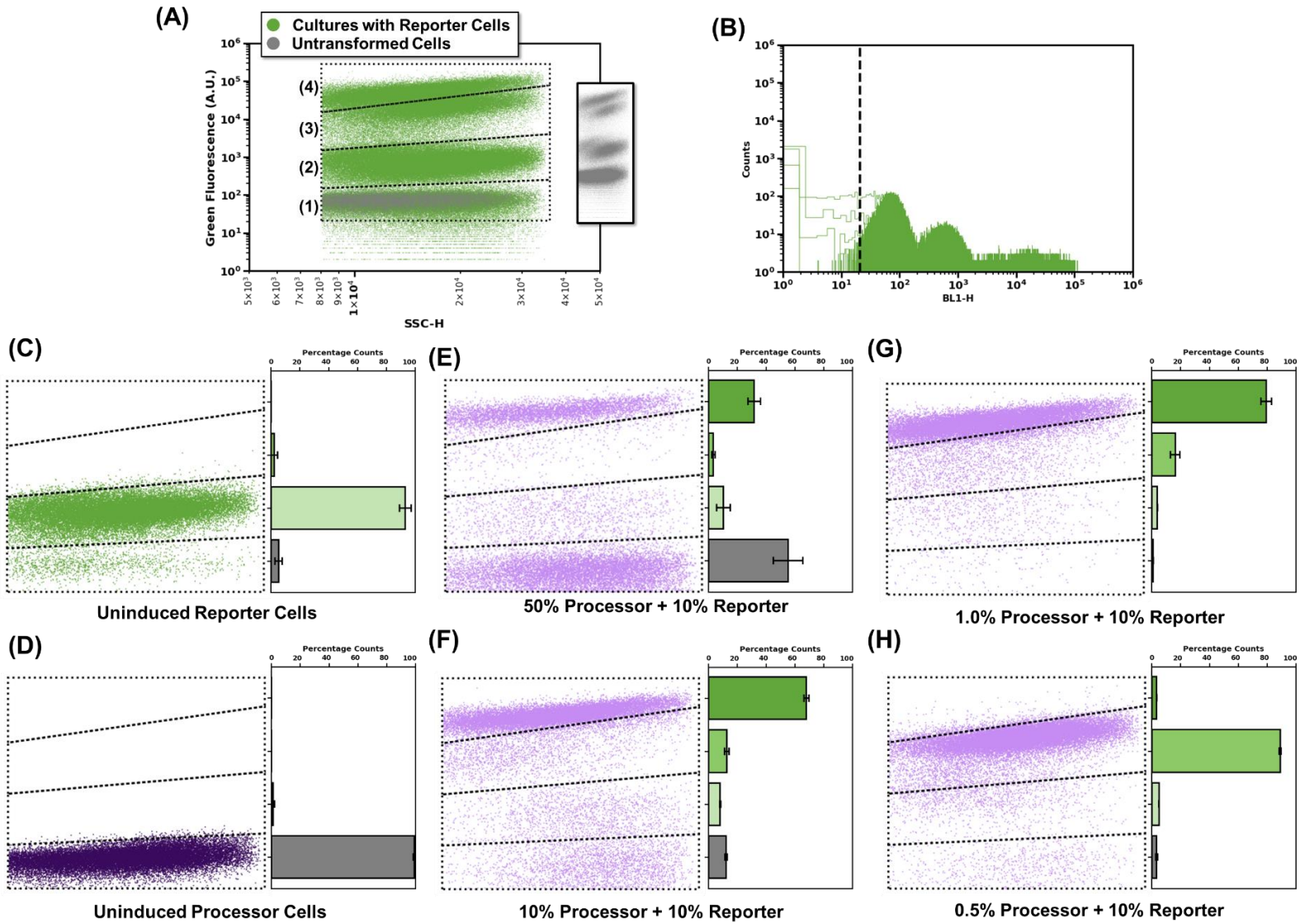

(A)
Cultures with Reporter Cells
Untransformed Cells
Green Fluorescence (A.U.)
SSC-H
(1)
(2)
(3)
(4)
(B)
Counts
BL1-H
(C)
Uninduced Reporter Cells
(D)
Uninduced Processor Cells
(E)
50% Processor + 10% Reporter
(F)
10% Processor + 10% Reporter
(G)
1.0% Processor + 10% Reporter
(H)
0.5% Processor + 10% Reporter
Percentage Counts

Figure 5.16. Single Cell Analysis of Processor to Reporter Noise Propagation

Analysis of processor-reporter co-cultures from the experiment shown in Figure 5.14 via flow cytometry after ~20 hours incubation. (A) 2D-scatter plot of side scatter height (SSC-H) vs green fluorescence. Green points show all reporter-containing cultures. Grey dots show monocultures of untransformed cells. Data was segmented into four sections based on fluorescence. (A1) Non fluorescent cells. (A2) Cells expressing sfGFP at background levels. (A3) Cells expressing sfGFP at moderate levels. (A4) Cells expressing sfGFP at high levels. Inset shows the same data as main plot at a higher resolution to better visualise the four populations. (B) Histogram of green fluorescence vs cell count. For all reporter-containing samples. Dashed black line shows cut-off point for analysis. (C-H) Left plots are 2D-scatter plots of SSC-H vs green fluorescence on the same scale as shown in (A). Right plots are bar plots showing the percentage of all cells in each of the four segments displayed in (A). (C-D) show uninduced monocultures of reporter and processor cells respectively. (E-H) shows co-cultures of processor and reporter cells added at the stated percentages. Percentages refer to volume-per-volume percentage of cell to final culture volume. Bar height is the mean of three or four replicates. Error bars show +/- standard deviation across replicates.

As bulk measurements of the co-cultures collected using a plate reader were unable to explain fully the behaviour observed by co-culturing the cell types, flow cytometry was used to perform single cell measurement. This allowed for each cell type to be separated based on fluorescence in an attempt to determine how the proportion of cell populations changed during the experiment. Samples from the plate reader experiment above were collected and prepared prior to analysis using a flow cytometer. Initial gating and voltage setup had been performed prior to experimentation using controls, as described in section 2.7.2.

All samples containing reporter cells were initially visualised on a scatter plot of side scatter height (SSC-H) vs green fluorescence, from which four distinct populations were observed (Figure 5.16 (A)). Overlaying data from monocultures of untransformed cells, one of these populations could be identified as cells which exhibit no green fluorescence above cellular autofluorescence (Figure 5.16 (A1)). The remaining populations (Figure 5.16 (A2-4)) had increasing levels of green fluorescence, and so were described as following: (A2) cells expressing sfGFP at background levels, (A3) cell expressing sfGFP at moderate levels, and (A4) cells strongly expressing sfGFP. A cut-off point was also defined based on the distribution of fluorescence levels, below which entities were assumed to be debris rather than cells (Figure 5.16 (B)).

For monocultures containing uninduced sfGFP reporter cells, the majority of cells (more than 90%) could be seen to have background levels of sfGFP expression, whilst a minority (approximately 0 to 5%) were found to have moderate levels of fluorescence. The remainder (approximately 5 to 10%) appeared to show no fluorescence (Figure 5.16 (C)). There are a number of possible reasons for this observed heterogeneity, including non-fluorescing cells caused by factors like loss of plasmid over time, and differences in cell cycle state [333], [334]. For uninduced default processor cells, as expected, no fluorescent cells were observed (Figure 5.16 (D)). When the processor cells were added to reporter cell cultures, all samples exhibited increased fluorescence (Figure 5.16 (E-H)), which was in-line with the plate reader data (Figure 5.14 (A)). It was also observed that when 1.0% v/v or above of processor cells were added, the majority of reporter cells were in a highly fluorescent state (Figure 5.16 (E-G)). However, when only 0.5% v/v of processor cells were added, most reporter cells were in a state of moderate sfGFP expression (Figure 5.16 (H)). This indicated that it may have been possible to create a processor-reporter co-culture with low enough

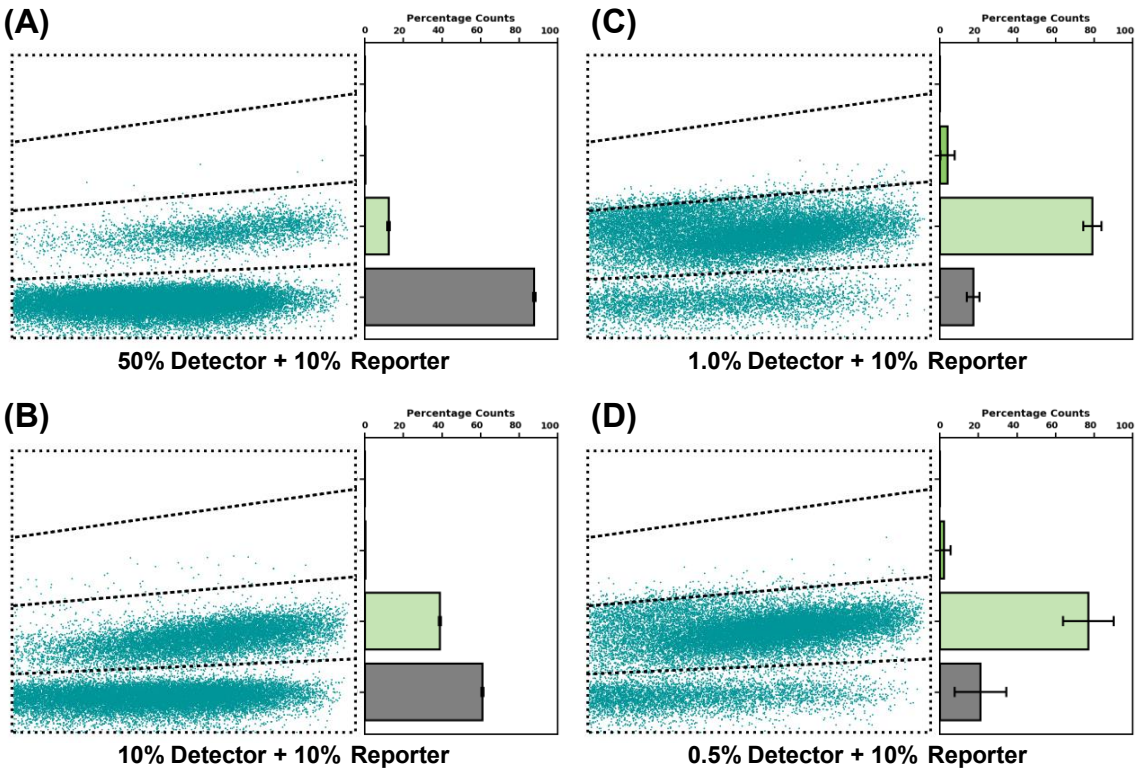


Figure 5.17. Single Cell Analysis of Detector to Reporter Noise Propagation: Analysis of detector-reporter co-cultures from the experiment shown in Figure 5.14 via flow cytometry after ~20 hours incubation. (A-D) Left plots are 2D-scatter plots of SSC-H vs green fluorescence on the same scale as shown in (Figure 5.16 (A)). Right plots are bar plots showing the percentage of all cells in each of the four segments displayed in (Figure 5.16 (A)). All plots show co-cultures of detector and processor cells added at the stated percentages. Percentages refer to volume-per-volume percentage of cell to final culture volume. Bar height is the mean of three or four replicates. Error bars show +/- standard deviation across replicates.

background activation of reporters by the processor to establish a functional system. Although, as even a 5:100 processor to reporter cell ratio showed significant increase in background activation compared to reporters in monoculture, the agent-based model's prediction that the multi-microbial biosensor could have high noise seems to be somewhat accurate.

For the processor-reporter co-cultures, it was possible to approximate the proportion of each cell type present at the point of measurement (i.e., after incubation for 20 hours, as described above), as only the reporter cells showed fluorescence of background levels or above. This method has some inaccuracy, as it could not be determined how many reporter cells were completely non-fluorescent, but it could be used as an indicator. The single cell results shown in Figure 5.16 (E-H) suggest that the reporter cells may be outcompeting the processor cells over time, resulting in an unstable culture. When processor and reporter cells were added in initial ratios of 5:1, it was

found that after 20 hours of incubation, the final approximate cell ratio was approximately 1.2:1 (Figure 5.16 (E)), representing a relative increase of reporter cells compared the processor cells. Similarly, when the initial ratio of processors to reporters was 1:1, a final ratio of 0.14:1 was measured (Figure 5.16 (F)), and when a ratio of 0.05:1 was added, the final ratio was 0.03:1 (Figure 5.16 (H)). This supported the thought that the collapse of a correlation between number of processors and reporter activation over time could have been partially caused by unstable cell populations.

The plate reader data suggested that the presence of uninduced IPTG detector cells had no influence over reporter cell activation (Figure 5.17). The single cell data supported this conclusion, as the distribution of reporter cells with regards to green fluorescence was almost identical to that of uninduced reporter cells in monoculture. Furthermore, when the detector and reporter cells were added in similar proportions (5:1 and 1:1), the final ratios detected appeared to be very similar (6:1 and 1.5:1 respectively). However, when the reporter cells were present in higher amounts (0.1:1 and 0.05:1), the relative number of detector cells seemed to slightly increase, such that the final ratios in both cases were approximately 0.25:1 in both cases. However, the variation between replicates observed for these co-cultures was also markedly increased, and hence were within error. Overall, it appeared that the detector and reporter cells established a stable co-culture over time.

The detector-processor co-cultures were analysed similarly to the reporter containing cultures, except red fluorescence was measured instead of green. For the detector-processor cultures, three populations were identified: non-fluorescent, background fluorescent, and moderately fluorescent (Figure 5.18(A)). For the uninduced processor cells in monoculture, the majority of cells were found to have background levels of fluorescence (Figure 5.18(C)).

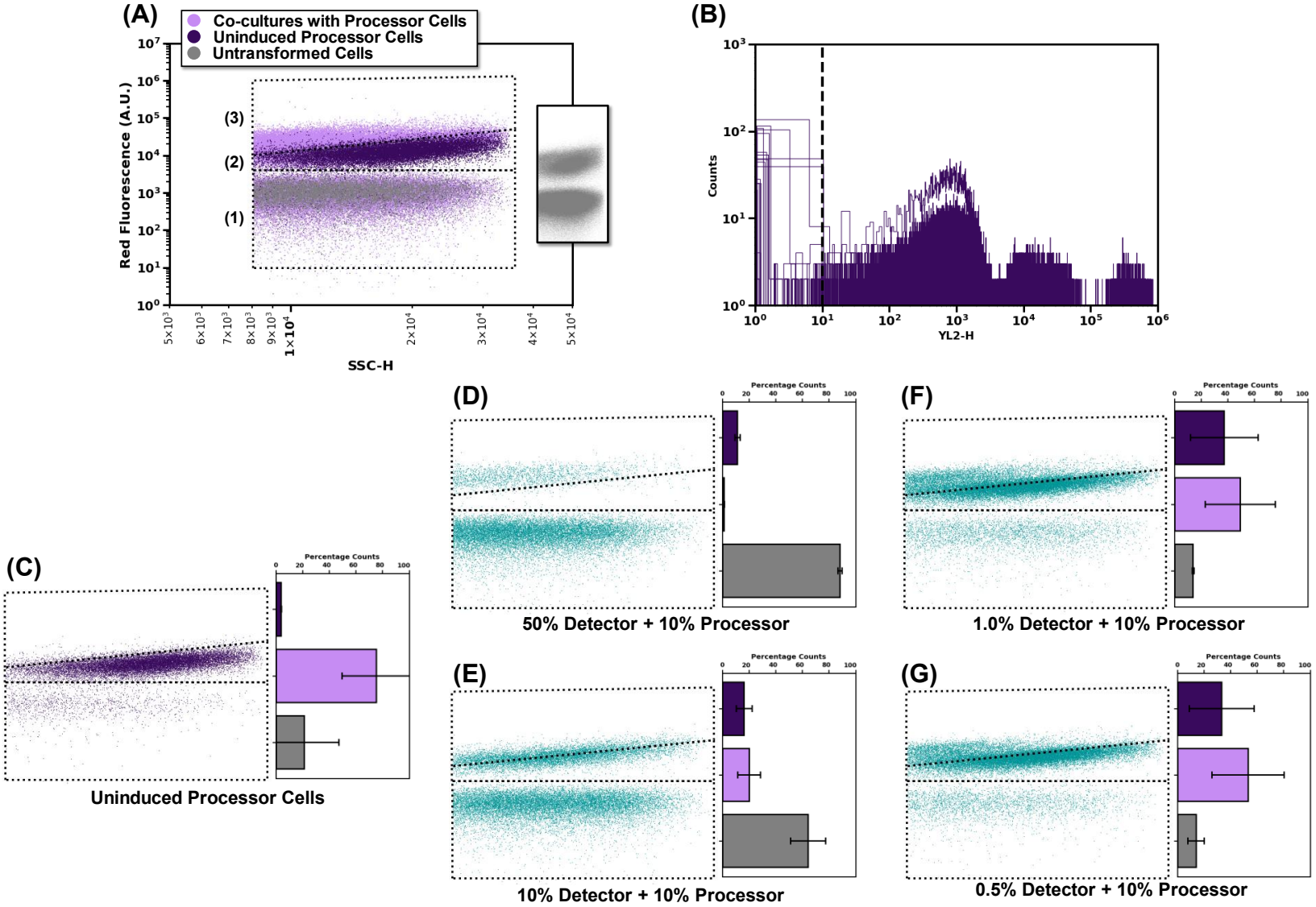

(A)
Co-cultures with Processor Cells
Uninduced Processor Cells
Untransformed Cells
Red Fluorescence (A.U.)
SSC-H
(1)
(2)
(3)
(B)
Counts
YL2-H
(C)
Uninduced Processor Cells
(D)
50% Detector + 10% Processor
(E)
10% Detector + 10% Processor
(F)
1.0% Detector + 10% Processor
(G)
0.5% Detector + 10% Processor
Percentage Counts

Figure 5.18. Single Cell Analysis of Detector to Processor Noise Propagation

Analysis of detector-processor co-cultures from the experiment shown in Figure 5.14 via flow cytometry after ~20 hours incubation. (A) 2D-scatter plot of side scatter height (SSC-H) vs red fluorescence. Light purple points show all detector-processor cultures, and dark purple points are uninduced processor monocultures. Grey dots show monocultures of untransformed cells. Data was segmented into three sections based on fluorescence. (A1) Non fluorescent cells. (A2) Cells expressing mCherry at background levels. (A3) Cells expressing mCherry at moderate levels. Inset shows the same data as main plot at a higher resolution to better visualise the populations. (B) Histogram of red fluorescence vs cell count for all detector-processor co-culture samples. Dashed black line shows cut-off point for analysis. (C-G) Left plots are 2D-scatter plots of SSC-H vs red fluorescence on the same scale as shown in (A). Right plots are bar plots showing the percentage of all cells in each of the three segments displayed in (A). (C) Uninduced monocultures of processor cells. (D-G) Co-cultures of detector and processor cells added at the stated percentages. Percentages refer to volume-per-volume percentage of cell to final culture volume. Bar height is the mean of three or four replicates. Error bars show +/- standard deviation across replicates.

In the bulk plate reader data, the detector-processor co-cultures with 50% v/v detector cells and 10% v/v processor cells showed large variation between replicates. It was thought that this variation was due to poor mixing by the liquid handler during experimental setup, in which case it was expected that in the single cell measurements, all processor cells would show background levels of red fluorescence. This was because the increased fluorescence would have resulted from more processor cells than intended, rather than increased activity of each cell. However, the single cell data showed processor cells with increased fluorescence in all replicates (Figure 5.18(D)). Additionally, the variation between samples was far lower for single cell measurements. Therefore, it was possible that the variation may have been due to a biological factor, but the factor's identity was unclear. For the remaining detector-processor co-cultures, increased red fluorescence of processor cells was also observed, with approximately half of all fluorescent processor cells exhibiting a signal above background levels (Figure 5.18(E-G)). This suggested that background expression of processor cells by detector cells was occurring, although large variation across repeats of some co-cultures made it difficult to determine definitively. Additionally, this conclusion was in contradiction to the bulk plate reader data, although as large degree of heterogeneity was detected, with processor cells being split almost in half between two states, it was possible that any signal was undetectable in bulk measurements.

The amount of fluorescent and non-fluorescent cells in each co-culture were compared to determine how the proportion of each cell type had changed during the course of experimentation. Whilst it seemed that there may have been some outgrowing of the processor cells by the detector cells, this was within error, and largely the two cell types were observed to grow in a stable manner during the 20-hour experiment. Therefore, one of the potential causes for low background activation of processor by the detectors, was proven to be incorrect. Instead, it was more likely that contrary to the prediction made by the agent-based model, in their uninduced state the detector cells only caused a low-level activation of processor cells.

Overall, the experiments presented here indicate that the greatest source of noise in the multi-microbial biosensor was likely to arise from background activation of the reporter cells by the processor cells. This background activation was observed to occur after approximately 5 hours, which suggested that the potential positive feedback loop in the processor module was the cause, as it was shown in sub-section 5.2.4 that there

was a delay before background mCherry expression began increasing. Regardless, the background activation of the sfGFP reporter cells appeared to be below the saturation limit, and thus processor cells induced by detector cells should have been able to activate higher sfGFP expression by the reporters.

5.3.3. Validation of the modular, multi-microbial biosensor

To determine if the proof-of-concept biosensor was functional, co-cultures of all three cell types were prepared as described in section 2.7.11, such that each cell type was added in equal amounts, as determined by culture optical density at a wavelength of 600 nm. Each co-culture was induced with one of a range of IPTG concentrations, including no IPTG. Monocultures of each cell type were also prepared, for which half were induced with their canonical inducer, and half were left uninduced. Samples were incubated at 37ºC for 20 hours with periodic fluorescence and absorbance measurements. As with the co-cultures in sub-section 5.3.2, fluorescence values are presented for the culture as a whole, and not correct for number of cells present.

Fluorescence measurements in the cyan wavelength were plotted over time to determine activation of IPTG detector cells (Figure 5.19 (A-B)). It was observed that detector cells in all co-cultures induced with IPTG showed increased eCFP fluorescence compared to the uninduced co-cultures. These results indicated that the detector cells could be induced by IPTG in co-culture, and activation levels appeared similar to that observed in monoculture up until 6 hours of incubation time.

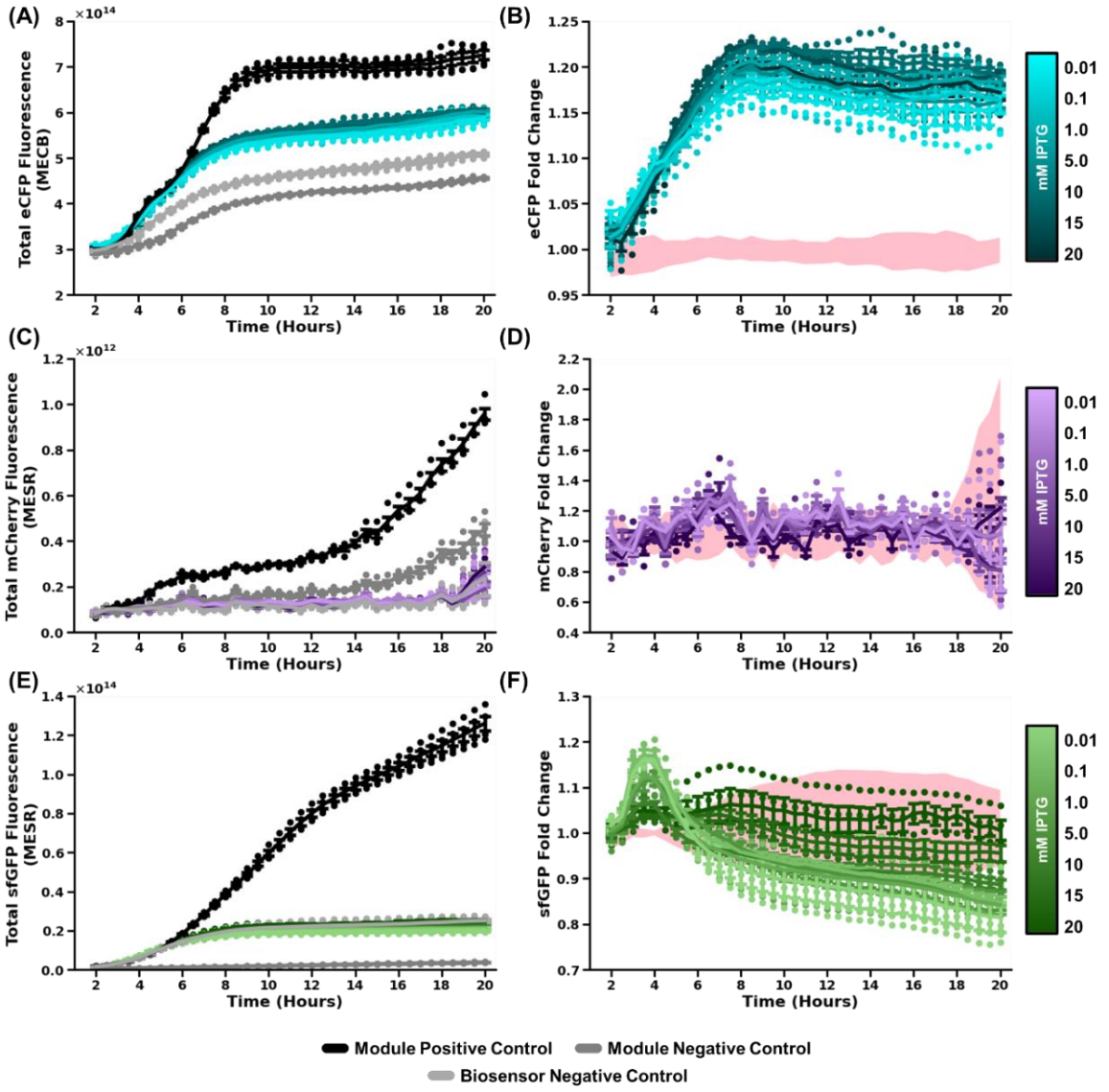


Figure 5.19. Characterisation of a 1:1:1 Cell Ratio Multi-Microbial Biosensor
Detector, processor, and reporter cells were added at an initial ratio of 1:1:1. Error bars show +/- standard error centred on the mean of 3 or 4 replicates. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (A, C, E) Time course curves of non-growth corrected eCFP (A), mCherry (C), and sfGFP (E) fluorescence for the multi-microbial biosensor system induced with a range of IPTG concentrations. The module positive controls were detector (A), processor (C), or reporter (E) cells in monoculture induced with 1 mM IPTG, 10 µM C12-HSL, or 10 µM C4-HSL respectively. The module negative controls were uninduced detector (A), processor (C), or reporter (E) cells in monoculture. The biosensor negative control was the multi-microbial system culture with water instead of IPTG. (B, D, F) Time course curves of fold change in eCFP (B), mCherry (D), and sfGFP (F) fluorescence of the multi-microbial biosensor system induced with a range of IPTG concentrations, relative to the biosensor negative control.

Measurements of red fluorescence were used to determine activity of the default processor cells over time (Figure 5.19 (C-D)). From this data, it appeared that the processor cells in co-cultures induced with IPTG showed no mCherry fluorescence

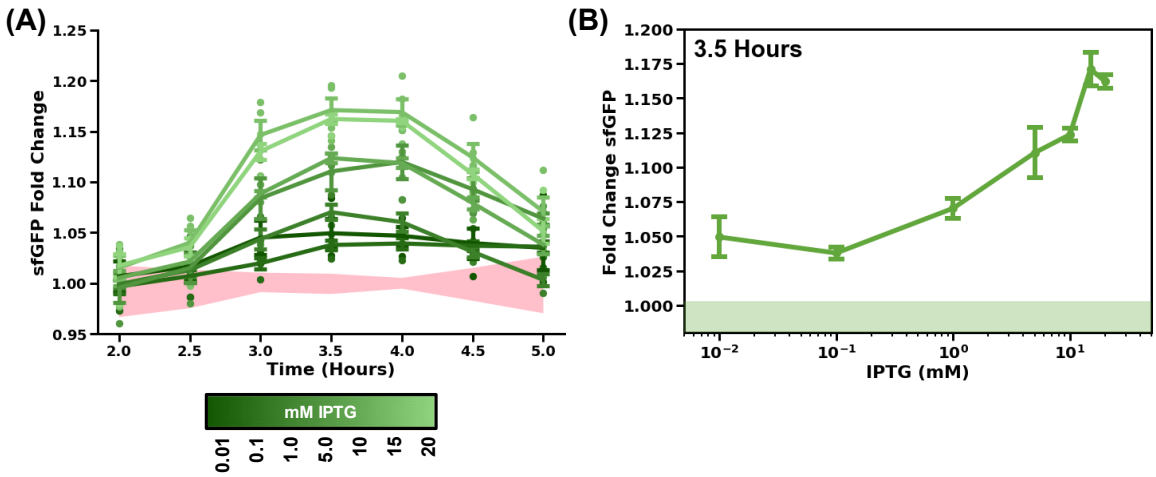


Figure 5.20. sfGFP Reporter Cell Response in Early Biosensor Culture
Data for reporter cells presented in Figure 5.19 (F) was plotted for the first 5 hours of culturing time to better visualise response. Error bars show +/- standard error centred on the mean of 3 or 4 replicates. (A) Time course curve of sfGFP fold change between 2 and 5 hours. Background noise was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (B) Dose-response curve of IPTG concentration vs sfGFP fold change after 3.5 hours of culture time. Coloured box shows limit of detection, calculated as 1 + standard deviation of uninduced co-culture after 3.5 hours of culture time.

above that of uninduced co-cultures. The agent-based model predicted that no fold change in mCherry production would be observed in biosensor co-cultures induced with IPTG compared to non-induced cultures. This was because of the high background activation of processor cells by the detector cells. However, as revealed by the noise propagation experiments, processor cells were only induced at a low-to-moderate level by the presence of uninduced detector cells. Additionally, the total fluorescence for biosensor co-cultures was below that of the uninduced processor cells in monoculture (Figure 5.19 (C)). Therefore, it was concluded that the reason for lack of processor activity was due to either insufficient production of C12-HSL by the induced detector cells, or out-growth of processor cells by the detector and reporter cells. Another possibility was that the processor cells were activated, but at a level too low to detect with plate reader measurements, as indicated by the detector to processor noise propagation experiments.

To determine whether the reporter cells were more highly induced in biosensor co-cultures induced with IPTG compared to uninduced biosensor systems, fluorescence in the green wavelength were made (Figure 5.19 (E-F)). Despite an apparent lack of processor cell activation, there was a small but significant fold change in sfGFP fluorescence for biosensor systems induced with IPTG compared to uninduced

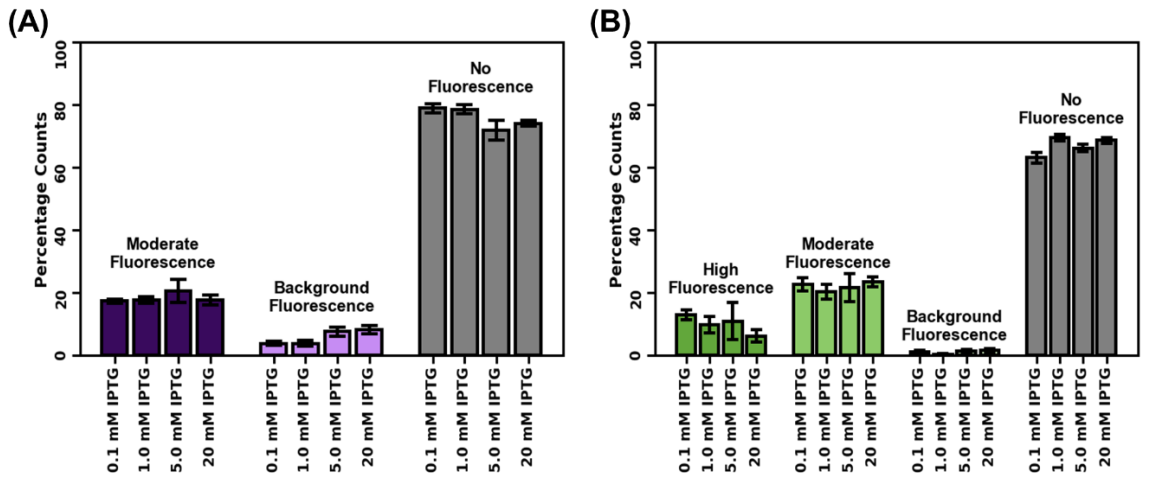


Figure 5.21. Single Cell Analysis of Biosensor Culture

The bar plots show single cell analysis for biosensor cultures after 20 hours of growth. **(A)** Bar plot showing the percentage of cells displaying no, background, and moderate red fluorescence, using thresholds visualised in Figure 5.18. **(B)** Bar plot showing the percentage of cells displaying no, background, moderate, and high green fluorescence, using thresholds visualised in Figure 5.16. For both bar plots, bar height shows mean percentage counts across 3 replicates, and error bars show standard error.

systems within the first four hours of culturing (Figure 5.20 (A)). This response was found to exhibit dose-dependent characteristics, although variation between replicates made it difficult to establish a definitive relationship between IPTG concentration and fold change in sfGFP (Figure 5.20 (B)). As it was found in sub-section 5.2.6 that IPTG could not be used to induce reporter cells, the fold change in fluorescence must have occurred through production of communication from either the detector or processor cells.

Past 4 hours of culture time, no fold change in sfGFP fluorescence above background noise was observed. This suggested that after this point, the co-culture became unstable as outgrowth between the cell types occurred, or background production of AHLs became too high, or a combination of both. To help determine the final proportion of cell types, a selection of the cultures was processed and analysed by flow cytometry as described in section 2.7.12 (Figure 5.21). From this, it was found that across all samples, approximately 30 to 35% of all cells exhibited green fluorescence, and hence were assumed to be reporter cells, and approximately 20 to 30% were found to have red fluorescence, and hence assumed to be processors. This suggested that after 20 hours of culture time, all cells remained at similar proportions to that at which they were added, although the processor cells appeared to have a slight decrease relative to the other cell types. It appeared that there were fewer processor cells in co-cultured induced with a lower concentration of IPTG, however the difference was small and not

definitive. The clear presence of both processor and reporter cells, along with evidence of continued eCFP production by detector cells as shown by the plate reader data, indicated that loss of signal by the reporter cells over time was not due to competition between cell types in co-culture. Therefore, it was likely that loss of signal was due to an increase in noise over time, where continued background production of C12-HSL and C4-HSL saturated the processor and reporter cells' response.

5.4. Conclusions and Next Steps

All module types could be activated by their canonical inducers, and a range of inducer concentrations was found for each in which response (measured as fold change in fluorescence) was correlated with amount of inducer. However, it was found that the detector and processor modules exhibited a lower maximal fold change above background noise than the reporter module. It was also found that IPTG could only activate the detector cells, and only a low level of cross talk was identified for the AHLs. In addition to responsiveness of the modules to induction, the ability for detector and processor module cells to produce C12-HSL and C4-HSL respectively was confirmed.

It was identified that induction of the default processor cells leads to decreased growth rates when compared to uninduced processor cells. It was not entirely clear why the processor module appeared to impart a higher burden than the detector or reporter modules, however there are a number of possible reasons. Firstly, the processor module may have caused a higher burden than the reporter module because the processor module required an additional protein to be expressed; the reporter module encoded RhlR and sfGFP, whereas the processor module encoded LasR, RhlI, and mCherry. However, this would not explain why the detector module would not also impart similar levels of burden onto host cells, as it also contained coding regions for the same number of proteins with similar sizes (LacI, LasI, and eCFP). Secondly, it was possible that *PLas* allowed for a higher level of expression than *PLac* or *PRhl*, and hence required host cells to produce more proteins. This reason was supported by a higher fold change in mCherry observed when inducing processor cells compared to the fold change in eCFP seen from detector cells, although the reporter cells exhibited an even high fold change in fluorescence. Thirdly, the processor module may have resulted in a higher burden on cells due to the positive feedback loop, which would likely have caused higher background production of proteins encoded by the processor module. Therefore, the processor cells may have already been in a state of low-level stress, which began to impact cell growth more heavily following induction. Rather than just one of the reasons being true, it was likely that all had some impact. However, as different fluorescent markers were used for all three module types, it was not possible to directly compare levels of expression to determine the true reason.

The agent-based model presented in chapter 4 predicted a high level of noise, where background production of AHLs could result in unwanted activation of the processor cells by the detector cells, and activation of reporter cells by the processor cells. Therefore, the potential for noise to propagate through the multi-microbial system via background synthesis of AHL communication molecules was tested experimentally. It was found that uninduced detector cells could activate the processor cells, and uninduced processor cells could activate reporter cells. Therefore, the potential for noise to propagate from the detector cells to the processor cells, and onto the reporter cells, for validated.

Co-culturing of the detector, processor, and reporter cells was found to result in a functioning biosensor, however loss-of-signal occurred 4 hours post induction. This signal loss was hypothesised to be due to an increase in background noise as a result of noise propagating through the system.

It has been previously shown how positive feedback loops in genetic circuits based on quorum sensing can reduce responsiveness to induction[315]. This supports the idea that over time the processor module may become unresponsive to the detectors and simply ‘leak’ a false signal to the reporter, hence destroying the biosensor’s activity. However, in contradiction to the results shown here, many previously reported synthetic microbial systems employing sequential, uni-directional cell-to-cell signal propagation did not display a loss of behaviour over time, with no evidence of noise propagation[107], [120], [335], [336]. The main differences between these previous systems and the biosensor system presented here is that three cell types are used rather than two. The inclusion of three cell types in a uni-directional fashion required one cell (the processor) to have both sending and receiving capabilities, which leads to issues with cross-talk and adds in an additional layer for the signal to pass through. Cells which display sending and receiving capabilities are largely missing from previously recorded systems, except in cases where bi-directional feedback is utilised to help with stability and robustness.

The experimental results shown here had some similarities to the simulation results presented in chapter 4, namely that background noise had the potential to decrease functionality of the system. The model also suggested that it may have been possible to optimise the biosensor through the design space of cell ratios. Chapter 6 presents

approaches taken towards optimisation of the biosensor, including exploration of cell ratios.

Chapter 6. Modular and Multi-Microbial Biosensor Optimisation

In chapter 5, it was found that whilst the modular, multi-microbial biosensor was functional, the maximal response was low, and the signal was unstable and decreased to background levels over time. The agent-based model presented in chapter 4 indicated that modification of cell ratios may have presented an easily accessible design space for optimisation of the biosensor's response. In section 6.2, the impact of cell ratios on biosensor behaviour is explored in the context of experimental data. Section 6.3 focuses on an alternative method of optimisation, making use of statistical Design Of Experiments to investigate the impact of environmental factors on each biosensor module. Finally, section 6.4 concludes the findings from this chapter, and discusses future work.

6.1. Introduction

6.1.1. Statistical Design Of Experiments

Optimisation of systems and processes can be performed by first measuring the impact different factors have on desired response characteristics, and then using this information to help determine which factor values could lead to optimal results. Within synthetic biology and biotechnology, the optimisation process is often performed by changing factors individually whilst keeping all other conditions the same[337]. This One Factor At a Time (OFAT) methodology is intuitive and allows for a degree of certainty as to whether a factor is impacting upon a system, however it does not allow for efficient exploration of the entire design space, and can lead to local optima being found, rather than the true global optima[338] (Figure 6.1). An alternative to the OFAT approach is to employ a statistical Design Of Experiments (DOE) approach. DOE refers to a collection of statistical tools and methodologies which allow for a rational and unbiased exploration of multi-factorial design spaces[89] (Figure 6.1). With DOE, experimental 'runs', where each run consists of a set of factor values to be tested, are determined using statistical algorithms. The experimental runs generated aim to yield the most amount of information in the least number of experiments possible, typically leading to faster, cheaper, and more efficient experimentation than with an OFAT approach[337]. However, the number of runs required can still be large when many factors require investigation. Once the experimental runs have been performed, the data can be fed back to a DOE statistical model, which helps determine potentially

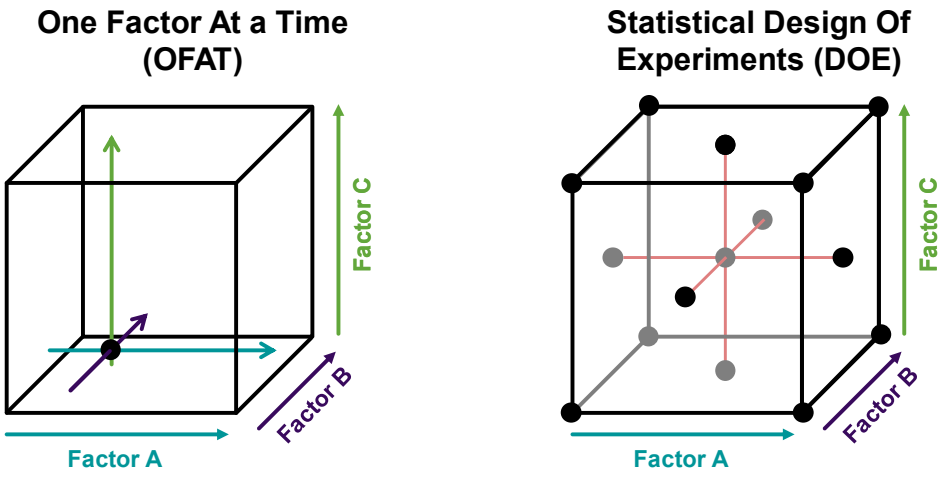


Figure 6.1. OFAT vs. DOE
Schematics representing One Factor At a Time (OFAT) and statistical Design Of Experiments (DOE) methodologies for a hypothetical system with 3 factors. With OFAT, the factor under investigation is modified whilst keeping values for the other 2 factors the same. With DOE, factor values are chosen to allow for representative sampling of the design space.

optimal factor values, and can also measure the relationship between factors which may have dependencies or interactions[89].

There exist several categories of DOE, each of which have advantages and disadvantages based on the desired information about a system or process to be determined[89]. Broadly, the types of DOE employed can be described by two groups: screening designs and optimisation designs[339]. Screening designs focus on measuring the impact different factors have on specific responses characteristics of a system, whilst optimisation designs can be used to map a system's response surface with an aim of finding optimal responses. Typically, when working with novel systems or processes, screening designs are employed initially to determine which factors may have the greatest impact on the system's response characteristic of interest[340]. Once important factors with the greatest impact have been determined, optimisation designs can be used to find optimal values for these factors. Optimisation designs are not typically used in the first instance as they require more experimental runs to be performed than screening designs, especially when a large number of potential factors require investigation[89]. Therefore, the less expensive screening designs can be used to determine only the most important factors, allowing fewer factors to be investigated in optimisation designs, and hence reducing the number of runs required.

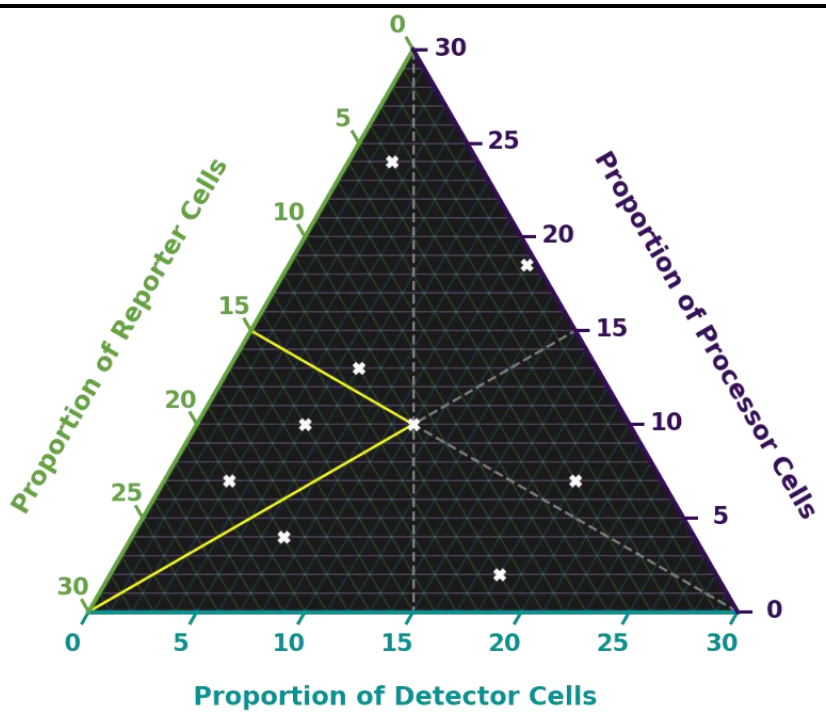


Figure 6.2. Selected Cell Ratios

Ternary plot displaying the cell ratios selected for experimental testing. The axis values represent the proportion of each cell type in the system, calculated as the volume (in µL) of each cell type (at a cell density of OD_{600} = 1.0) in a culture volume of 100 µL. The total volume of all cells was 30 µL. The white crosses represent cell ratios tested experimentally, and dashed grey lines show stoichiometric boundaries, where the amounts of two cell types are equal. The solid yellow lines bound the section predicted by the agent-based model to contain more optimal cell ratios (detectors < processors < reporters).

6.2. Experimental Exploration of Cell Ratio Design Space

Although the agent-based model in chapter 4 predicted that the multi-microbial biosensor would not show any activity, it suggested that modifying the ratio of cell types could impact upon the biosensor's behaviour. Additionally, as seen in chapter 5, the biosensor with a 1:1:1 cell ratio exhibited functionality within the first few hours of co-culture, where the biosensor responded to IPTG in a dose-responsive manner. However, 5 hours after induction, the signal could no longer be detected. Therefore, it was possible that although the model was not quantitatively accurate with its predictions, qualitatively it may have yielded useful insight into which cell ratios could display more optimal biosensor functionality.

6.2.1. Selecting cell ratios

In chapter 4, the design space of cell ratios was visualised using a ternary plot. The design space could be split into 6 sections based on the stoichiometric boundaries between each cell type, with the 1:1:1 ratio sitting at the centre of all sections. The *in silico* results generated by the agent-based model indicated that a larger signal-to-

noise response could be generated by the biosensor when more reporter cells were present than detector or processor cells (Figure 4.20), and when the number of processor cells were present in higher amounts than the detector cells (Figure 4.19). Cell ratios which matched these conditions fell within the section highlighted in Figure 6.2. To fully explore the design space, at least one ratio from within each section was selected which corresponded to a similar cell ratio tested by the model. It was ensured that an additional ratio was tested from the section predicted by the model to incorporate potentially better performing biosensor systems (Figure 6.2).

6.2.2. Experimental characterisation of cell ratios

To test the selected cell ratios, cell cultures were incubated in a plate reader at 37^{o}C for 20 hours, with fluorescence and cell density measurements taken every 30 minutes. For all systems, the calibrated green fluorescence (Figure 6.5), calibrated cyan fluorescence (Figure 6.3), calibrated red fluorescence (Figure 6.4), and total cell growth (Figure 6.6) is reported. For these experiments, the fluorescence values were not corrected for cell density as it was not possible to determine the proportion of each cell type in the system (section 2.7.13).

Table 6.1: Summary of Multi-Microbial Biosensor Behaviour by Cell Ratio

Cell Ratio	Observations		
	Detector Cells	**Processor Cells**	**Reporter Cells**
2:24:4	Small fold change in fluorescence detected after 12 hours	Fold change detected after 14 hours	Initial peak in fluorescence fold change at 3 hours, drop in signal below noise at 8 hours, fold change detected again at 12 hours
3:7:20	Small fold change in fluorescence detected after 7 hours	Fold change detected after 18 hours	Initial peak in fluorescence fold change at 3 hours, drop in signal at 7 hours, fold change peaked again at 15 hours
5:10:15	No fold change in fluorescence detected	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 6 hours
6:13:11	No fold change in fluorescence detected	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 4 hours

7:4:19	Small fold change in fluorescence detected after 6 hours	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 4 hours
10:10:10	Fold change in fluorescence detected after 2 hours	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 6 hours
11:18.5:0.5	Large fold change in fluorescence detected after 7 hours	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 10 hours
18:2:10	Large fold change in fluorescence detected after 7 hours	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 16 hours, although only a small fold change was observed from 6 hours
19:7:4	Large fold change in fluorescence detected after 7 hours	No fold change in fluorescence detected	Peak in fluorescence fold change at 3 hours, loss-of-signal at 4 hours

For all cell ratios tested excepting two (5:10:15 and 6:13:11), a fold increase in cyan fluorescence was observed (Figure 6.3). This indicated successful induction of the IPTG detector cells. Additionally, it was found, as expected, that a larger number of detector cells corresponded with a larger fold change in cyan fluorescence. The four systems containing the highest proportions of detector cells (10:10:10, 11:18.5:0.5, 18:2:10, and 19:7:4) displayed an increase in cyan fluorescence at approximately the same time as a fold increase in green fluorescence was measured (Figure 6.5). This behaviour indicated activation of the reporter cells by signal propagation through the detector and processor cells. However, for the remaining systems, although activation of the detector cells could not be measured, a response by the reporter cells was observed. It was therefore possible that amplification of the signal was occurring, allowing a small, undetectable response by the detector cells to result in a larger response by the reporter cells. For the majority of systems, no fold change in red fluorescence by the processor cells could be detected (Figure 6.4), for reasons which will be expanded on in due course. A summary of all biosensor behaviours is detailed in table 6.1.

In this experiment, analysis of green fluorescence data showed that once again, the 1:1:1 cell ratio biosensor system exhibited initial functionality, but eventually the signal

became lost to the background noise (Figure 6.5 (A)). Indeed, the majority of cell ratios tested showed similar behaviour, with the reporter cells showing increased green fluorescence when the co-culture was induced with IPTG, followed by a decrease over time. Some of the ratios, however, showed a second peak in signal after this initial decrease, and a further sub-set of ratios had another subsequent decrease after the secondary peak, displaying an oscillation-like behaviour. The exact pattern of this oscillation-like signal differs between systems, where in some cases the signal never dropped below the noise threshold, whilst others regained signal after temporarily losing it. In some cases, although there was a noticeable second peak pattern, the signal could not be distinguished from the noise (as in 7:4:19 and 10:10:10).

In chapter 5, it was discussed that the eventual loss-of-signal by the 1:1:1 biosensor system may have been due to co-culture instability, resulting in some cell types outcompeting others. However, the single cell data at 20 hours post induction indicated that all three cells were present in similar proportions, and thus it was thought that the accumulation of noise in the form of leaky quorum sensing molecule production was to blame for the loss-of-signal. From the data presented here, it is posited that perhaps a combination of both factors (accumulation of noise and unstable co-culture) contributed to the observed behaviour.

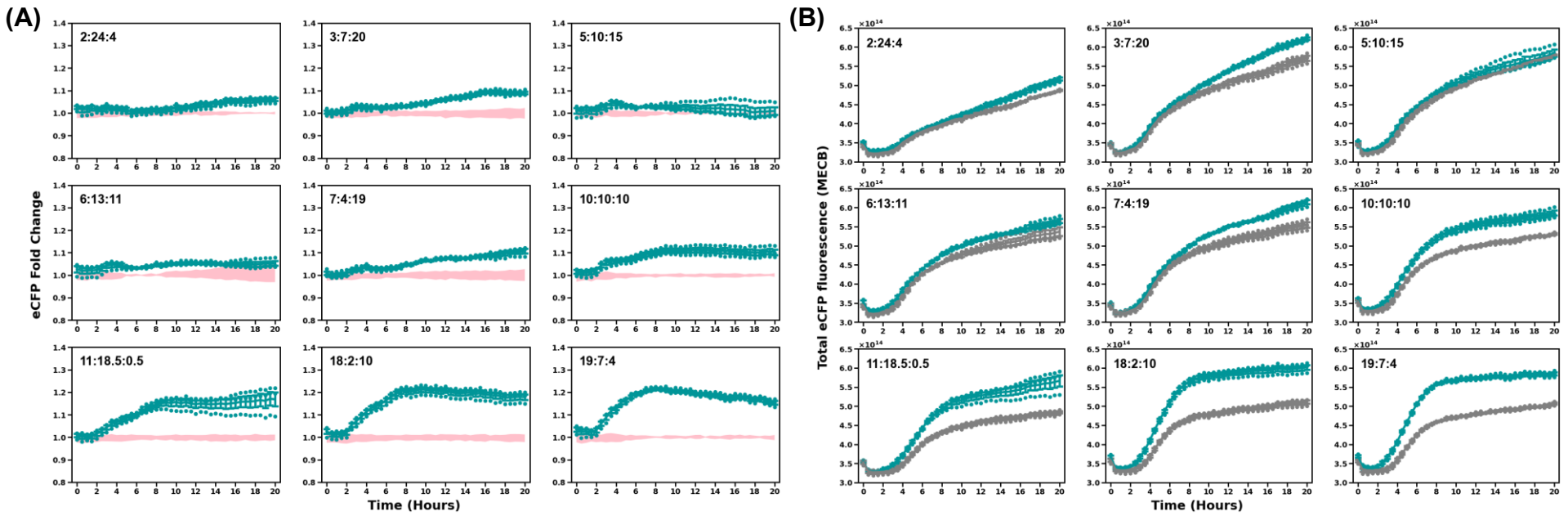


Figure 6.3. Detector Cell Behaviour in Co-Culture at Various Cell Ratios

All plots show the change in cyan fluorescence of IPTG detector cells in the multi-microbial biosensor co-culture when induced with 20 mM IPTG compared to the uninduced system. Error bars show +/- standard error centred on the mean of 3 replicates, and individual points show data for each replicate. The cell ratios used are displayed on each plot. (A) Fold change in cyan fluorescence of IPTG detector cells in co-culture, calculated as the change in fluorescence of co-cultures induced with 20 mM IPTG relative to uninduced co-cultures. The pink areas represent background noise, which was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (B) Total calibrated cyan fluorescence (in molecules of equivalent cascade blue) for induced (cyan) and uninduced (grey) systems.

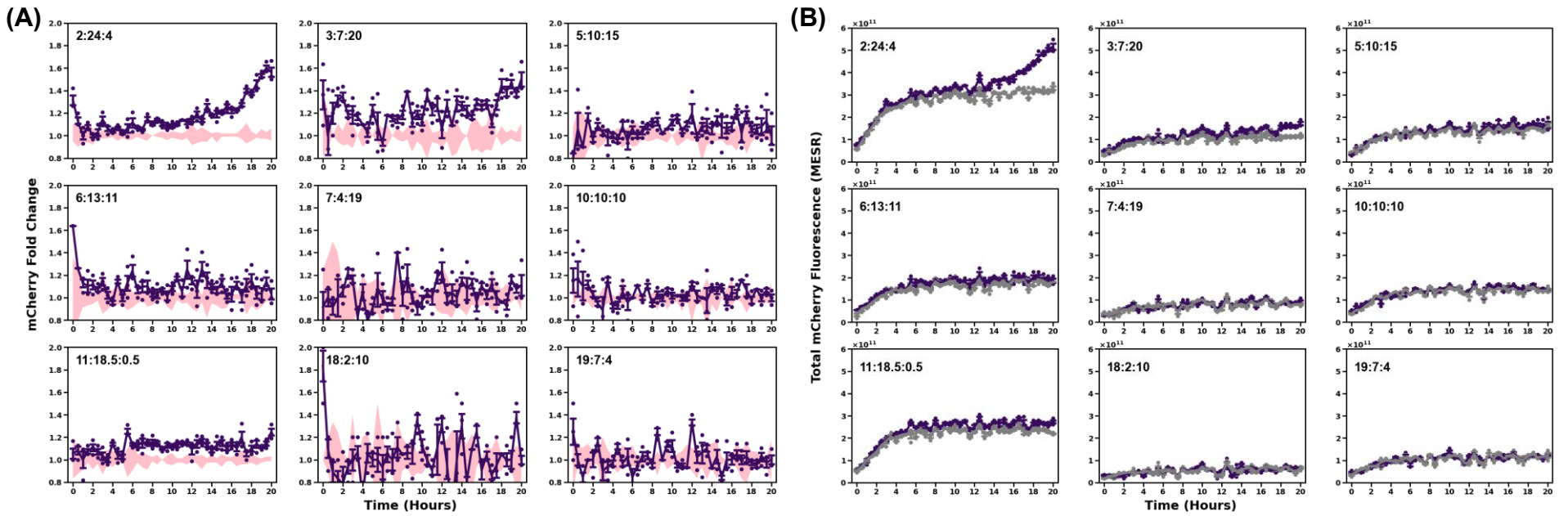


Figure 6.4. Processor Cell Behaviour in Co-Culture at Various Cell Ratios

All plots show the change in red fluorescence of default processor cells in the multi-microbial biosensor co-culture when induced with 20 mM IPTG compared to the uninduced system. Error bars show +/- standard error centred on the mean of 3 replicates, and individual points show data for each replicate. The cell ratios used are displayed on each plot. (A) Fold change in red fluorescence of default processor cells in co-culture, calculated as the change in fluorescence of co-cultures induced with 20 mM IPTG relative to uninduced co-cultures. The pink areas represent background noise, which was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (B) Total calibrated cyan fluorescence (in molecules of equivalent sulforhodamine 101) for induced (purple) and uninduced (grey) systems.

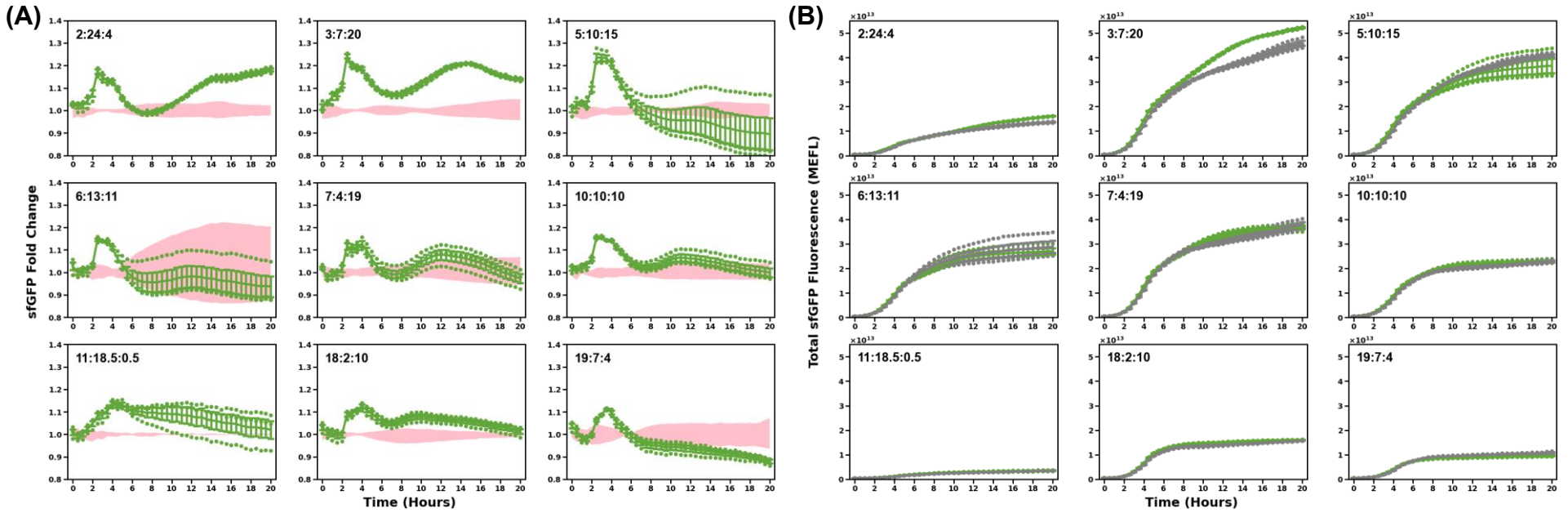


Figure 6.5. Reporter Cell Behaviour in Co-Culture at Various Cell Ratios

All plots show the change in green fluorescence of sfGFP reporter cells in the multi-microbial biosensor co-culture when induced with 20 mM IPTG compared to the uninduced system. Error bars show +/- standard error centred on the mean of 3 replicates, and individual points show data for each replicate. The cell ratios used are displayed on each plot. (A) Fold change in green fluorescence of sfGFP reporter cells in co-culture, calculated as the change in fluorescence of co-cultures induced with 20 mM IPTG relative to uninduced co-cultures. The pink areas represent background noise, which was calculated as the maximum and minimum calibrated fluorescence values from uninduced cells at each time point, relative to the mean fluorescence for all uninduced cells. (B) Total calibrated green fluorescence (in molecules of equivalent fluorescein) for induced (green) and uninduced (grey) systems.

The unstable signal observed across the biosensor co-cultures was due to non-regular increases in fluorescence over time of both the uninduced and induced systems, rather than an actual decrease in fluorescence (see the non-relative fluorescence data in Figure 6.5 (B)). Additionally, as would be expected, the total fluorescence of systems with fewer reporter cells added at the start of the experiment is lower than that of systems with more reporter cells. The overall growth curves for each system are similar, although some systems displayed decreased cell densities in the induced systems compared to the non-induced systems (Figure 6.6). In chapter 5, it was found that only the processor cells had decreased growth rates when induced, and so it was thought that the decreased growth in the co-cultures was due to induction of the processor cells by the detector cells. This thought was supported by observations that the decreased growth of induced systems was more prevalent when a higher ratio of processor cells was used. A noticeable outlier to this observation was the 2:24:4 cell ratio system, which despite having the largest proportion of processor cells displayed almost no decrease in cell growth. This could be explained by the lack of detector cells, however, as fewer detector cells would have meant less C12-HSL present in the system to induct the processor cells.

If it were the case that the detector cells were causing a decrease in processor cell growth, this could explain the potential unstable cell type proportions in co-culture over time, and the fluctuations in reporter cell fluorescence. In this scenario, near the beginning of the experiment when the cells are in lag to early exponential phase and before the detector cells have been induced to produce C12-HSL, the default processor cells could grow at a similar rate to the other cell types. As the detector cells begin producing C12-HSL, the default processor cells would then be induced to produce C4-HSL, which could activate the reporter cells and lead to the first peak in signal observed for all systems. However, as the default processor cells become induced, their growth rate would drop and the amount of C4-HSL produced would be expected to fall to levels similar to that seen in the uninduced systems. This decrease in C4-HSL production would result in the initial loss-of-signal observed. However, this behaviour alone does not explain the second increase in signal observed for some cell ratios. Instead, this may be explained by the positive feedback loop inherent in the default processor cells' design.

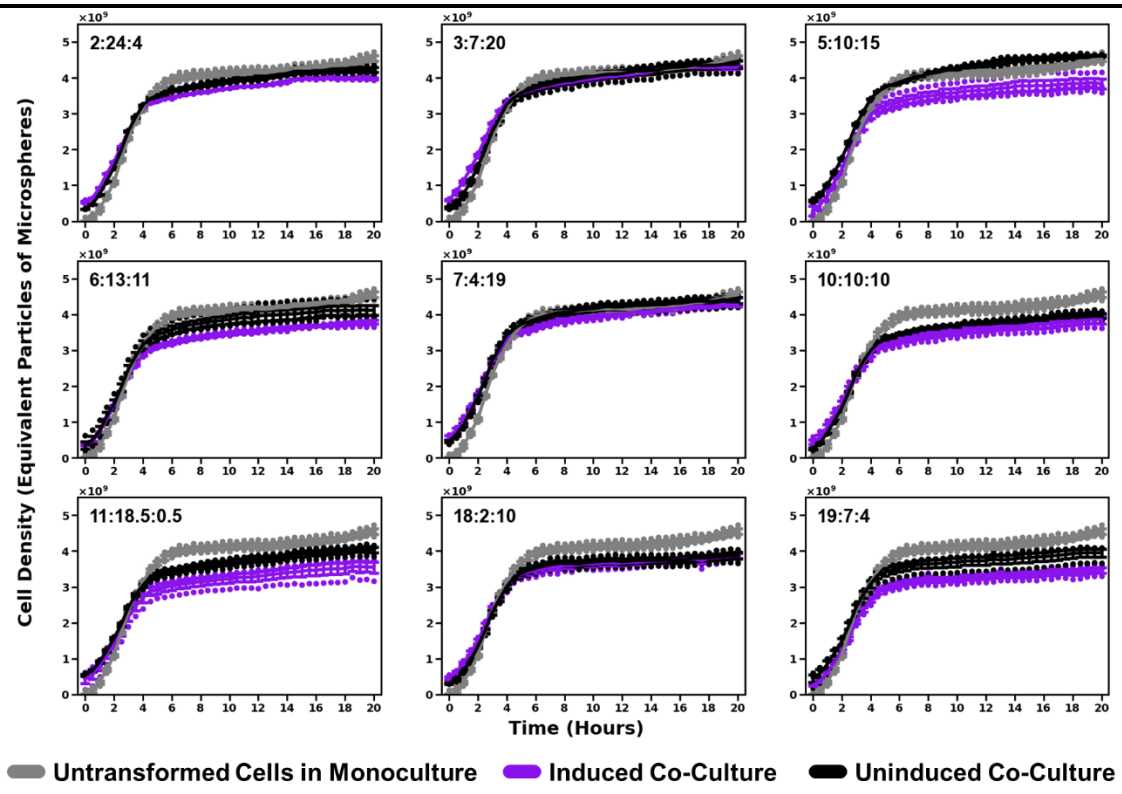


Figure 6.6. Cell Growth in Multi-Microbial Biosensor Systems
Time course plots of calibrated cell density over 20 hours. Black lines show uninduced systems for each cell ratio, and purple lines show systems induced with 20 mM IPTG. Untransformed *E. coli* DH5α cells in monoculture were used as a control (black lines). Error bars show +/- standard error centred on the mean of 3 replicates, and individual points show data for each replicate. The cell ratios used are displayed on each plot.

In monoculture, the default processor cells were shown to have similar behaviour to that observed with the multi-microbial biosensor, where an initial increase in signal was followed by a subsequent decrease. This was concluded to be due to a positive feedback loop, where the C4-HSL produced by the processor cells, in close proximity to the LasR transcription factor, could induce further expression of RhlI which was responsible for C4-HSL synthesis. When the processor cells were induced, the feedback loop appeared to occur at an earlier time point than the uninduced system. In the biosensor co-culture, after around 6 hours it appeared that the cells reached stationary phase. At this point, it would be expected that the proportions of cells in the system would stabilise. Over time, the C4-HSL produced by the default processor cells could continue accumulating due to the feedback loop, leading to further induction of the reporter cells. Indeed, this appears to happen in some systems, most noticeably in the cultures with cell ratios of 2:24:4, 3:7:20, and 7:4:19 (Figure 6.5). The feedback loop could also explain the second decrease in signal observed, as it appears the

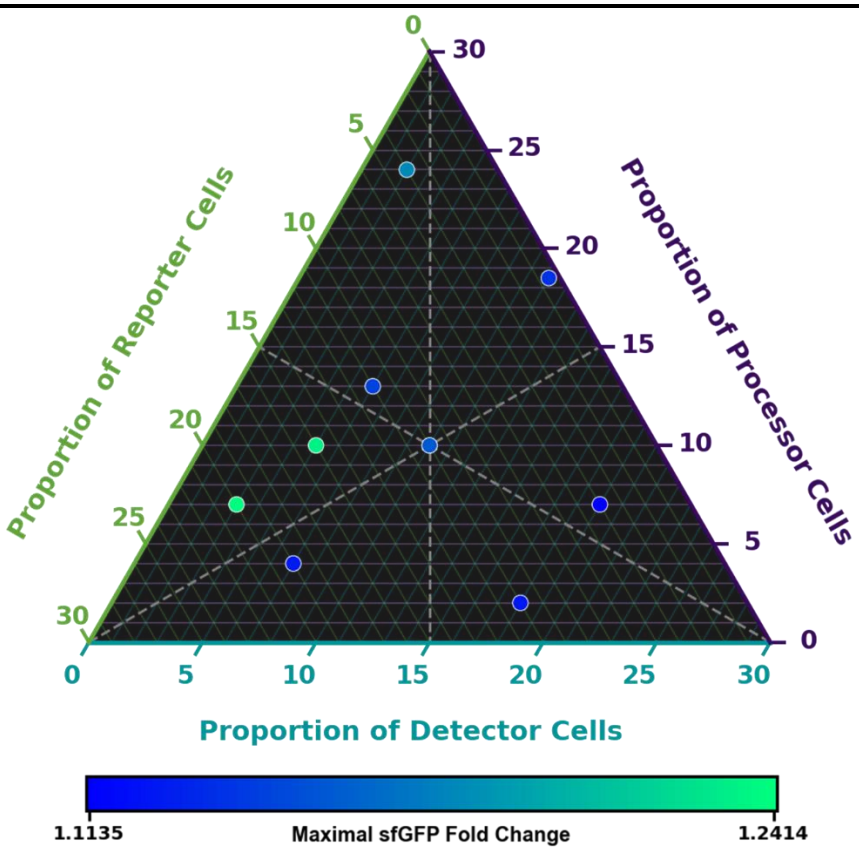


Figure 6.7. Impact of Cell Ratios on Maximal Biosensor Response
Ternary plot displaying the maximal biosensor response, measured as the maximal fold change in green fluorescence for induced vs non-induced systems, for each cell ratio. Each dot represents a characterised cell ratio and is coloured based on the maximal biosensor response. The axis values represent the proportion of each cell type in the system, calculated as the volume (in µL) of each cell type (at a cell density of OD_{600} = 1.0) in a culture volume of 100 µL. The total volume of all cells was 30 µL. White crosses represent the cell ratios tested experimentally. The dashed grey lines show stoichiometric boundaries, where the amounts of two cell types are present in equal amounts.

uninduced systems show a similar increase in green fluorescence but lagging behind the induced system, mirroring the behaviour observed for the processor cells alone.

For two of the systems, an eventual steady signal appeared to be reached. These systems had cell ratios of 2:24:4 and 3:7:20 (Figure 6.5 (A)). Although for the majority of systems fluorescence by the default processor cells could not be detected, potentially due to the aforementioned out competition of processor cells by the detectors and reporters, the 2:24:4 and 3:7:20 systems did eventually exhibit a measurable signal (Figure 6.4). These two systems had the lowest detector-to-processor cell ratio compared to all other systems tested, which may have allowed for a high enough proportion of processor cells to remain in the system that as the feedback loop allowed for accumulation of both C4-HSL and mCherry, a signal could be observed. Additionally, the lower number of detector cells would have resulted in a

lower background activation of the processor cells, which was found to occur even with uninduced detector cells in chapter 5. The lower background activation could have then contributed to the more stable signal observed for these systems.

6.2.3. Determination of optimal cell ratios

To help determine the optimal ratios for biosensor activity, the maximal fold change in green fluorescence for all systems tested were visualised on a ternary plot (Figure 6.7). From this visualisation, two systems exhibited the maximum activity: 3:7:20 and 5:10:15. Both of these systems populated the same section of the cell ratio design space. In this section of the design space, the number of detectors is always lower than the number of processors, and the number of processors is always lower than the number of reporter (detectors < processors < reporters). Interestingly, this was the same section of the design space tentatively predicted by the agent-based model to represent the best chance of creating a functional multi-microbial biosensor. This therefore indicated the usefulness of the agent-based model as a qualitative predictor of optimal cell ratios, even if the values predicted were inaccurate.

The identified design space section (detectors < processor < reporters) representing the most optimal cell ratios was in keeping with the analysis provided in the previous sub-section, where it was thought that a lower detector-to-processor cell ratio would result in a more functional system. However, the 2:24:4 system identified to have the most stable signal was not included in this design space, although it did exhibit the highest maximal activity outside of the 3:7:20 and 5:10:15 systems. This was likely due to the much lower proportion of reporter cells in the 2:24:4 system, as less reporter cells would have led to lower fluorescence intensity. Therefore, it appears that the most optimal cell ratio for a system may be dependent on the biosensor's application, where it should be decided if a high fold change or more stable signal is required. However, to determine this relationship, further experimentation with a larger number of cell ratios should be performed.

6.3. Statistical Design of Experiments Driven Characterisation

The design space of cell ratios was, as discussed in the previous section, a promising approach for optimisation of modular and multi-microbial biosensors. Another potential avenue for optimising the biosensor system was to focus on external factors which could impact upon the cells which compose the biosensor. It is well known that cellular synthetic biology systems are impacted by factors such as the composition of media used for culturing and incubation temperature[52], [219], [341], [342]. Thus, it stood to reason that optimisation of factors such as these may have allowed for enhanced biosensor functionality. As mentioned in sub-section 6.1.2, statistical Design of Experiments can be used to screen for factors which have a large impact on response characteristics of a system, and to determine factor values which yield optimal responses. A common design type for determining factors which have an impact on a specific system or process is the main effects screening design[343]. These main effects designs are able to measure the relative impact of various factors on a user-specified characteristic of a system (such as the dynamic range of a biosensor) in a low number of experiments. The major drawback of such designs is that interactions between factors cannot be easily determined without additional experimental runs [89].

For the Sensynova framework, it was decided that each module type should be characterised independently, as these results would provide context-free information about optimal conditions for each module, which could then be used to help inform theoretical future systems which make use of the modules developed here in conjunction with other modules. For each module, the JMP software was used to generate a custom, D-optimal, main-effects screening design. Main-effects screening designs can be used to gain insight as to which factors are having the greatest impact upon the system in a low number of experimental runs. There were six factors investigated in total. The first three factors were the temperature at which the cells were incubated during measurement (incubation temperature), the temperature at which cells were grown in overnight cultures prior to inoculation in a 96-well plate (overnight culture temperature), and the initial cell density (cell starting OD_{600}). The remaining three factors described the composition of LB media used for culturing, specifically the proportions of tryptone, NaCl, and yeast extract. For each factor, upper and lower limits were set. All limits were the same across the three cell types, except for the IPTG detector cells' overnight culture temperature factor. For the default

processor and sfGFP reporter cells, the overnight culture temperature limits were 25^{o}C and 37^{o}C. In initial experiments, it was found that the IPTG detector cells grew poorly in overnight cultures at 25^{o}C, and the final cell density was routinely too low for experimental setup. Therefore, limits of 30^{o}C and 37^{o}C were used instead. The final designs are listed in the supplementary material.

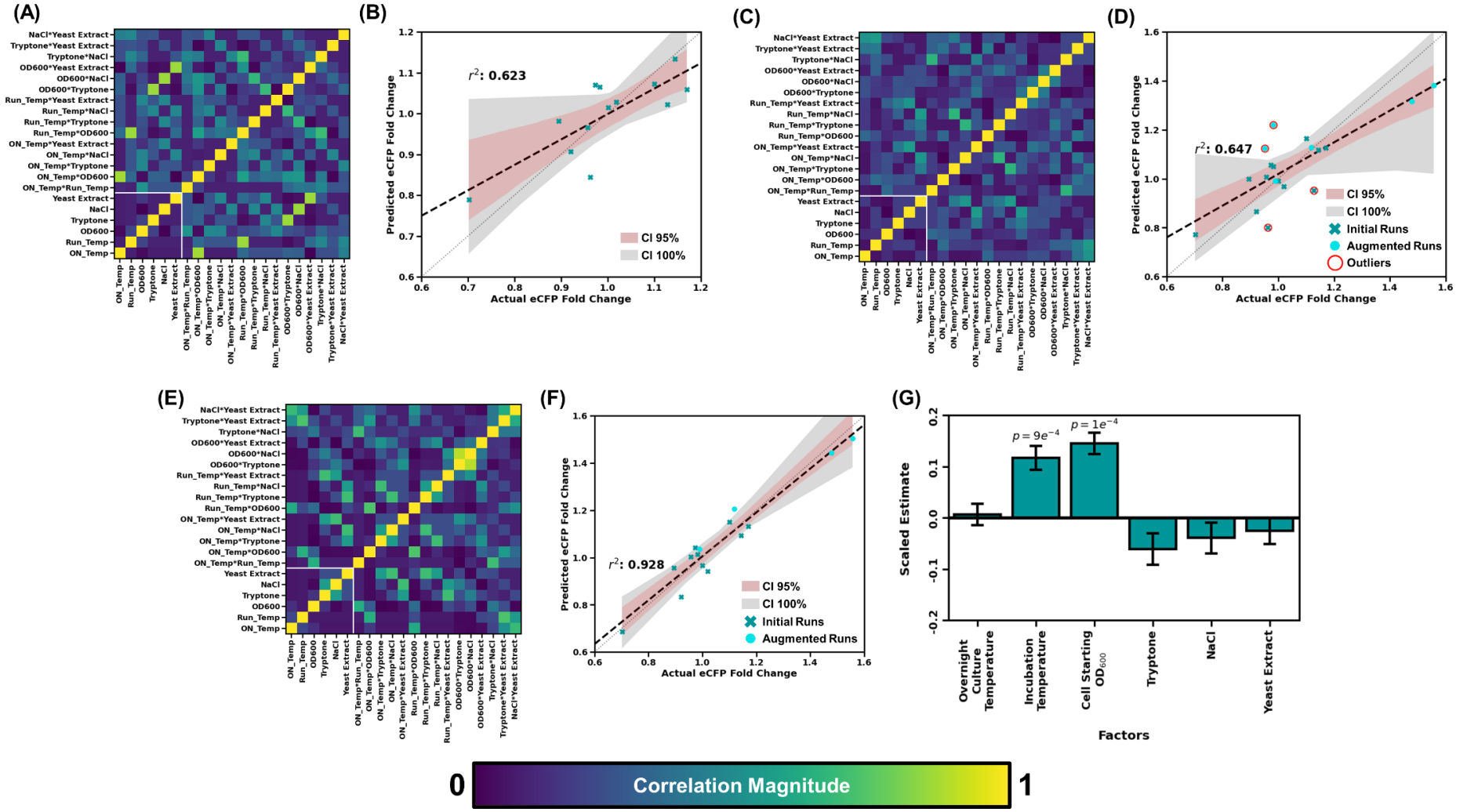


Figure 6.8. Main Effects Screening for IPTG Detector Cells

Analysis of DOE screening design, model, and results for IPTG detector cells. (A-B) Initial main effects screening design. (C-D) Augmented main effects screening design. (E-F) Main effects screening design without anomalous runs. (A, C, E) Colourmap on correlations for main

effects (bounded by white box) and interaction effects. Each cell represents the absolute correlation magnitude between two effects or interactions. (B, D, F) Scatter plot of actual eCFP fold change responses as measured experimentally vs fold change responses predicted by the main effects, standard least squares statistical model. Crosses show data points from runs determined by the initial screening design, and dots represent runs determined by the design augmentation stage. Red circles highlight runs with results thought to be anomalous. The dashed black line shows the line of best fit through the data points (the r^2 value shows the variation for this fit), and the dotted grey line shows line with slope of 1. Confidence levels are displayed at the 95% (pink shading) and 100% (grey shading) levels. (G) Bar chart showing scaled estimates of main factor effects on fold change in eCFP. The *p* value is shown for factors with effects significant at the 1% level, as calculated by a t-test. Error bars show the standard error of scaled estimates.

6.3.2. IPTG Detector Cell Factor Screening

For the IPTG detector cells, 13 experimental runs were performed according to the initial design and as described in section 2.7.14. For each experimental run, the fold change in eCFP fluorescence of induced samples compared to uninduced samples after 6 hours was calculated and used as the system response characteristic. When the samples were measured in the 96-well plate by the plate reader, the initial cell densities were measured as slightly different to the density aimed for and determined by the DOE design. This was due to slight pipetting inaccuracies and imprecise measurements by the spectrophotometer. The DOE design was modified so that the initial cell density factor had values corresponding to those measured by the plate reader rather than the density aimed for, allowing for more accurate interpretation of results. Changing these values impacted on the DOE design, resulting in higher correlation scores between the factors, and a lower D efficiency score (60.2 compared to the original 89.5). The higher correlation and lower D efficiency resulted in lower confidence in separating factor effects, however from the correlation colourmap, it could be seen that the main effects correlations remained relatively low. The JMP software was used to model main factor effects on the fold change response using a standard least squares effect screening model. It was found that the model performed poorly, where the predicted response values did not correlate well with the observed response values. This poor performance was potentially due to sampling bias introduced by the change in initial cell starting densities used compared to those suggested by the design. To help increase the design's power, the augment feature of JMP was used to generate an additional 6 experimental runs. With the additional runs, the main effect factor correlations were lowered (although remained higher than the initial design), and the D efficiency was increased to 90.2 (which was higher than the initial design). These runs were performed in an identical manner to the initial experiment, and the fold change responses were added to the DOE model. Whilst the model appeared to perform slightly better, with the line of best fit through the actual vs predicted values having a slope closer to 1, four experimental runs were identified which had much higher residuals than the other samples. These experimental runs were thought to be potentially anomalous results, and so were removed from the DOE model. Exclusion of these results once again increased the factor correlation effects, particularly for the media composition factors, and decreased the D efficiency score to 80.9. However, the main effects screening model showed far better performance, with

predicted values correlating closely to the actual values. Regarding factor effects on fold change in eCFP, the model indicated that two of the factors, incubation temperature and the initial cell starting densities, were important. The remaining factors did not appear to have any impact on functionality of the IPTG detector cells. Further testing would be required to confirm these results; however, it appears that it would be possible to optimise the detector cells by changing their external conditions.

6.3.3. Default Processor Cell Factor Screening

Factor effects for the default processor cells were initially estimated by performing 12 experimental runs as defined by the initial main effects screening design, and as described in section 2.7.14. The response for the processor cells was defined as the fold change in mCherry fluorescence of induced cells relative to uninduced cells. As with the detector cell experiment, some of the initial cell density measurements varied from the density values which were aimed for. These values were used in the screening design instead, which resulted in higher correlation magnitudes between factors, especially between the initial cell density and the overnight culture incubation temperature. Therefore, the design was augmented with an additional 6 experimental runs, which helped decrease the main factor correlations. However, the standard least squares effect screening model could be seen to perform poorly, with line of best fit through actual mCherry fold change values vs values predicted by the model having a slope far from 1, and large confidence intervals. It was observed that 3 of the experimental runs did not fit well with the other results, and thus were thought to be anomalous. These runs were removed, and the design was re-analysed. It could be seen that the correlations between main effects once again increased, however they remained below 0.5. The statistical model also appeared to perform better, with the actual vs precited response values showing moderate correlation, with a slope close to 1. From this model, it was estimated that the incubation temperature was the only factor to impact upon the default processor cells' response. However, due to relative high correlation of incubation temperature with the other factors, particularly the overnight culture temperature and initial cell densities, it was possible that confounding effects were impacting on the data. Therefore, to better estimate these effects, additional runs would need to be performed.

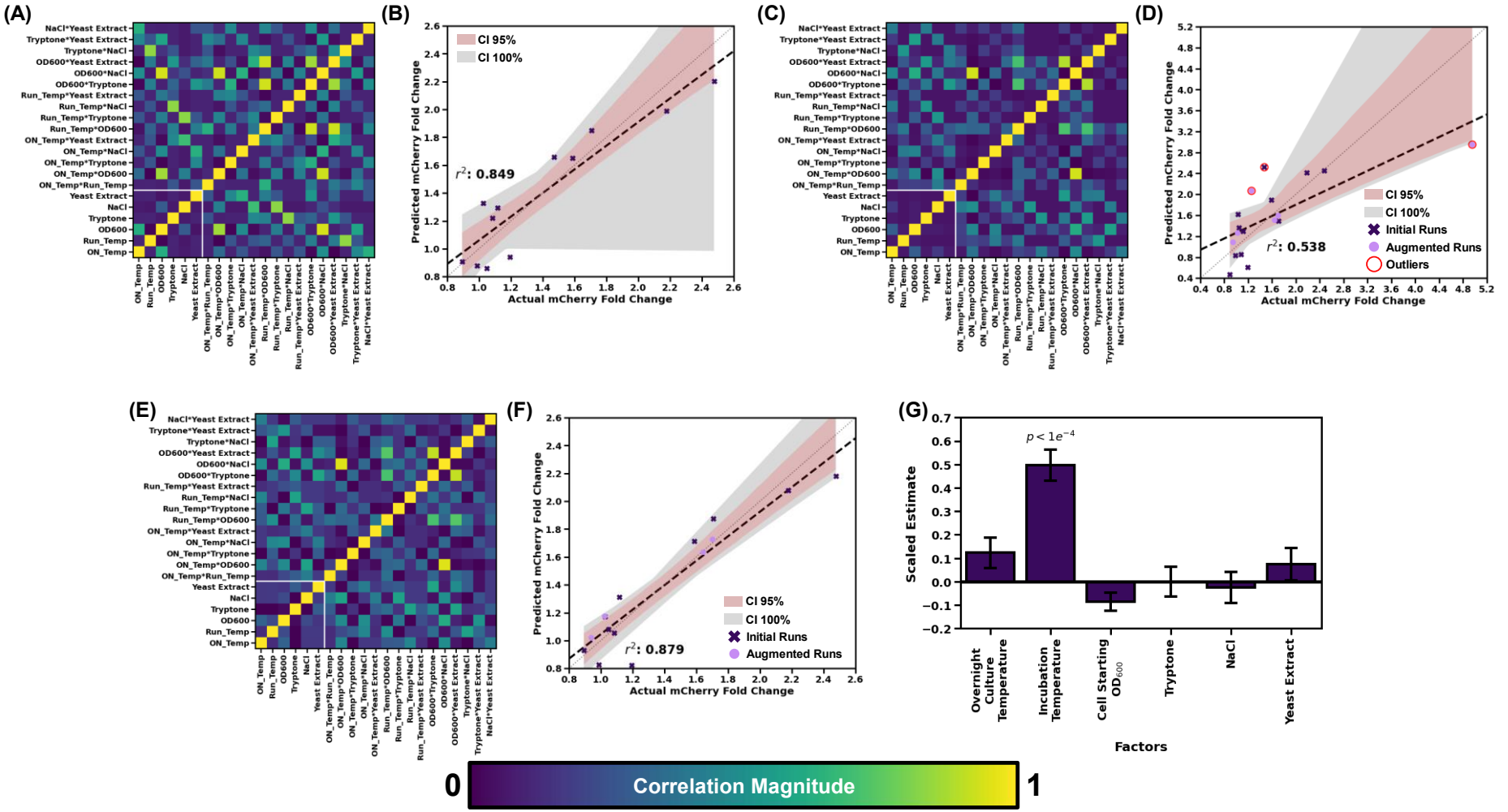


Figure 6.9. Main Effects Screening for Default Processor Cells

Analysis of DOE screening design, model, and results for default processor cells. (A-B) Initial main effects screening design. (C-D) Augmented main effects screening design. (E-F) Main effects screening design without anomalous runs. (A, C, E) Colourmap on correlations for main effects (bounded by white box) and interaction effects. Each cell represents the absolute correlation magnitude between two effects or

interactions. (B, D, F) Scatter plot of actual mCherry fold change responses as measured experimentally vs fold change responses predicted by the main effects, standard least squares statistical model. Crosses show data points from runs determined by the initial screening design, and dots represent runs determined by the design augmentation stage. Red circles highlight runs with results thought to be anomalous. The dashed black line shows the line of best fit through the data points (the r^2 value shows the variation for this fit), and the dotted grey line shows line with slope of 1. Confidence levels are displayed at the 95% (pink shading) and 100% (grey shading) levels. (G) Bar chart showing scaled estimates of main factor effects on fold change in mCherry. The *p* value is shown for factors with effects significant at the 1% level, as calculated by a t-test. Error bars show the standard error of scaled estimates.

6.3.4. sfGFP Reporter Cell Factor Screening

A main effects screening design generated for the default processor cells was used for characterisation of the sfGFP reporter cells, where fold change in sfGFP was used as the response. Experimental runs were performed as described in section 2.7.14. Once again, the guide cell densities determined by the design differed from those measured experimentally, and so the design was modified to incorporate this. Whilst the standard least squares main effects model generated from this data showed good correlation of actual vs predicted responses, where the line of best fit showed a slope close to 1, the colourmap of factor correlations showed high magnitudes for the initial cell density and overnight culture temperature factors. Thus, the design was augmented by 6 additional experimental runs. The augmentation helped decrease the main effects factor correlation magnitudes, however the statistical model exhibited poorer performance. Three experimental runs showed potentially anomalous results, and so were removed from analysis. The resulting design showed higher correlations between the main factors than the fully augmented design, but values were still overall lower than the initial design. Additionally, the statistical model showed good performance with small confidence intervals. From this model, it was predicted that as with the default processor cells, the only factor to influence sfGFP fold change was incubation temperature.

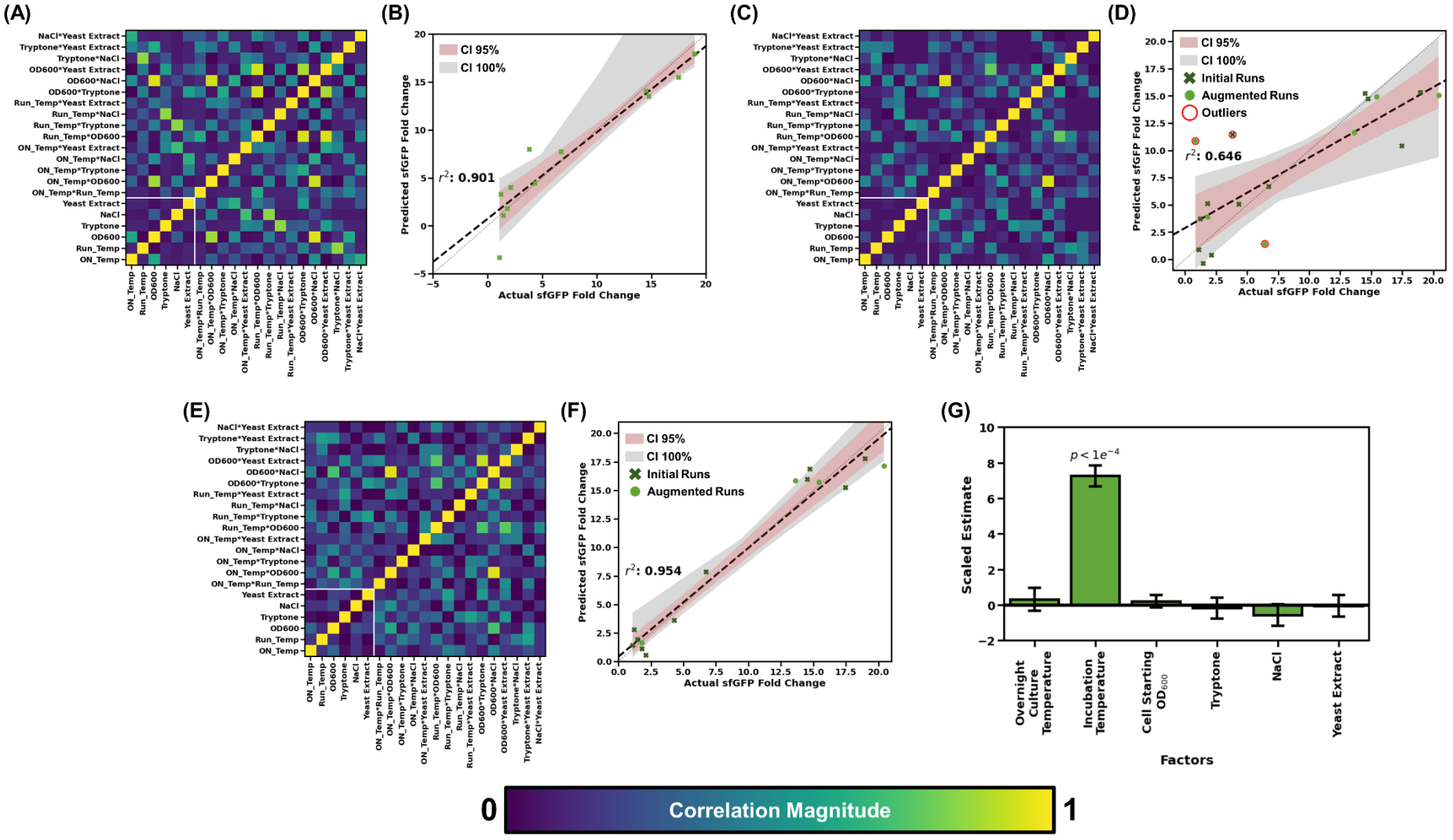

(A)
NaCl*Yeast Extract
Tryptone*Yeast Extract
Tryptone*NaCl
OD600*Yeast Extract
OD600*NaCl
OD600*Tryptone
Run_Temp*Yeast Extract
Run_Temp*NaCl
Run_Temp*Tryptone
Run_Temp*OD600
ON_Temp*Yeast Extract
ON_Temp*NaCl
ON_Temp*Tryptone
ON_Temp*OD600
ON_Temp*Run_Temp
Yeast Extract
NaCl
Tryptone
OD600
Run_Temp
ON_Temp
(B)
CI 95%
CI 100%
r^2: 0.901
Predicted sfGFP Fold Change
Actual sfGFP Fold Change
(C)
(D)
CI 95%
CI 100%
Initial Runs
Augmented Runs
Outliers
r^2: 0.646
Predicted sfGFP Fold Change
Actual sfGFP Fold Change
(E)
(F)
CI 95%
CI 100%
Initial Runs
Augmented Runs
r^2: 0.954
Predicted sfGFP Fold Change
Actual sfGFP Fold Change
(G)
$p < 1e^{-4}$
Scaled Estimate
Overnight Culture Temperature
Incubation Temperature
Cell Starting OD_{600}
Tryptone
NaCl
Yeast Extract
Factors
0
Correlation Magnitude
1

Figure 6.10. Main Effects Screening for sfGFP Reporter Cells

Analysis of DOE screening design, model, and results for sfGFP reporter cells. (A-B) Initial main effects screening design. (C-D) Augmented main effects screening design. (E-F) Main effects screening design without anomalous runs. (A, C, E) Colourmap on correlations for main effects (bounded by white box) and interaction effects. Each cell represents the absolute correlation magnitude between two effects or interactions. (B, D, F) Scatter plot of actual sfGFP fold change responses as measured experimentally vs fold change responses predicted by the main effects, standard least squares statistical model. Crosses show data points from runs determined by the initial screening design, and dots represent runs determined by the design augmentation stage. Red circles highlight runs with results thought to be anomalous. The dashed black line shows the line of best fit through the data points (the r^2 value shows the variation for this fit), and the dotted grey line shows line with slope of 1. Confidence levels are displayed at the 95% (pink shading) and 100% (grey shading) levels. (G) Bar chart showing scaled estimates of main factor effects on fold change in sfGFP. The *p* value is shown for factors with effects significant at the 1% level, as calculated by a t-test. Error bars show the standard error of scaled estimates.

6.4. Conclusions and Future Work

In this chapter, it was investigated how modifying the ratios in which each cell type was initially added impacted the modular and multi-microbial biosensor's behaviour. It is typical for studies focusing on the engineering of multi-microbial systems give some thought towards the proportions in which different cell types or species are added, as it is well documented that population changes within microbial communities can have large effects [344]–[348]. However, usually such thoughts extend only to documenting the proportions each population was added at and ensuring populations remain at equal amounts during an experiment or process[349], [350]. Studies which have considered cell ratios as an avenue for optimisation have mainly done so for natural communities, rather than *de novo* synthetic biological systems[351], [352]. In recent years, as multi-microbial systems have become ever more popular in synthetic biology, the potential of cell ratios as a design space has become increasingly apparent, especially within the field of metabolic engineering. For example, it has been shown how mixing cell-free extracts containing enzymes involved in mevalonate synthesis could optimise yield [353], and a recent study (Liu and co-workers 2022) demonstrated how different *E. coli* cells expressing sections of a pathway could be co-cultured in different proportions to optimise biosynthetic efficiency [354].

In this chapter, it was found that cell ratios could be used as a design space to optimise the functionality of a multi-microbial biosensor, mainly by impacting on the dynamic range and signal stability over time. This finding was similar to how previous studies have shown the impact of population proportions on natural communities, and how ratios can be used to improve the yield of biosynthetic pathways.

The observation of an unstable response was thought to be due to changes in the proportions of each cell types over time as a result of competition for resources. Therefore, changes in the initial cell type ratios could have impacted upon whether or not a specific cell type became outcompeted. Additionally, based on information from chapter 5 indicating that induced processor cells displayed slower cell growth than uninduced cells, it was concluded that the amount of detector cells present, which are able to induce processor cells, could have directly impacted on the ability for the processors to grow. It was indeed observed that only with a sufficiently small number of detector cells was a response able to be measured from the processors (2:24:2 and

3:7:20). The lack of a measurable processor response with higher numbers of detector cells may have been due to the amount of processor cells being too low.

The different cell ratios may have also impacted on the propagation of noise through the biosensor system. Noise propagation could occur by background synthesis of AHLs by the detector and processor cells, which could accumulate over time and result in activation of downstream cell types even in uninduced systems. Reduction of background activation could also explain the lack of fold change in red fluorescence by the processor cells when in the presence of a large number of detectors, as background accumulation of C12-HSL could have caused saturation of the processor's response. Therefore, a fold change could only be seen in systems with detector cells below a certain threshold, where background C12-HSL synthesis did not saturate the processor. It was also seen that response stability, where a fold change in green fluorescence by the reporter cells, was only observed for the 2:24:2 and 3:7:20 cell ratios (which were the systems with the lowest proportion of detector cells). It was therefore possible that the eventual loss-of-signal by the other systems with higher numbers of detector cells was due to saturation of the processor cells after approximately 6 to 10 hours of growth, leading to similar levels of C4-HSL being produced, and hence non-differential activation of the reporter cells in both induced and uninduced systems. To determine if either, or both, of these scenarios were the case, it would be necessary to measure growth of each cell type in co-culture over time, and to determine the concentration of AHLs in each system at various time points. By measuring the change in cell type proportions, which could be achieved through single cell analysis using techniques such as flow cytometry or microfluidics, it could be determined whether increasing amounts of detector cells did inhibit processor cell growth. Measuring the amount of each AHL in the system would identify whether the AHL concentrations were sufficient to saturate either the processor or reporter cells' responses.

Visualisation of each cell ratio tested on a ternary plot allow for identification of the optimal cell ratio design space. Using maximal reporter cell response, it was found that the two best performing biosensors had ratios where the number of detectors was less than the number of processors which was less than the number of reporters, which was in-line with preliminary predictions made by the agent-based model. However, it was possible that this was a local optimum, as the third best performing system had a

cell ratio in a different section of the design space. Therefore, further testing with additional cell ratios perhaps employing a Design Of Experiments approach, would be required to better characterise the cell ratio design space. Additionally, better parameterisation of the agent-based model, making use of sensitivity analysis to identify the most crucial parameters and experimental parameterisation to determine parameter values which fits simulation data to the experimental data, would allow for further *in silico* exploration of the design space.

Following on from exploration of the cell ratio design space, it was investigated whether it was possible to optimise each biosensor module individually by modifying the conditions in which they were cultured. To achieve this characterisation, statistical Design Of Experiments (DOE) was used to screening for factors which impacted upon the behaviour of each cell type module. It was found in all cases that the temperature at which the cells were incubated at had the largest impact. Additionally, for the IPTG detector cells, it was found that the cell density at which the cells were initially added also had an impact. Although the incubation temperature impacting upon each cell type's ability to respond to induction was expected, as *E. coli* grows optimally at 37^{o}C, the observation that initial cell density only impacted the detector cells was unexpected. Additionally, it was unexpected that the media composition would have little-to-no effect on the cells, as previous studies employing Design of Experiments have found culture medium to impact on the behaviour of a variety of species [355], [356]. To investigate these factors further, it would be necessary to perform additional screening experiments which could explore interactions between each factor, rather than simply the main effects as seen here, and to conduct surface-response characterisation to find optimal values for the impacting factors.

The overall success of the Sensynova system could have been improved in several ways. Firstly, screening of quorum sensing channels for use in 3-cell unidirectional communication would have allowed for selection of mechanisms which reduced the amplification of noise at the processor cell level. Similar screening has been successfully implemented previously, and this historical data was indeed used to select the quorum sensing mechanisms used here[151]. However, these previous studies focus more on responsiveness to outside induction, rather than use within sequential, uni-directional communication. Secondly, the identification of more appropriate computational modelling/simulation which can scale with the complexity of multi-

microbial systems would have allowed for faster design iterations, and the exploration of a larger number of parameters. The hybrid agent-based and deterministic SBML-based modelling, whilst potentially highly representative of multi-microbial systems, is far too computationally expensive to allow for these investigations. Therefore, more traditional modelling approaches, such as deterministic and non-agent-based modelling, may be more appropriate[214], [357]. Finally, the optimisation of each individual module at a genetic level, rather than relying primarily upon external conditions and modulation of cell ratios when in co-culture, would likely have helped optimise the final biosensor's activity, as issues such as high background noise could have been reduced. It is therefore suggested that any future work look to use a combination of genetic and non-genetic interventions to optimise multi-microbial biosensors based on the Sensynova framework.

Chapter 7. Optical Communication as an Alternative to Quorum Sensing

One challenge of synthetic multi-microbial systems is difficulties associated with co-culturing different cell types. Populations can compete for resources leading to unstable systems and changes in proportions throughout the lifetime of the system. Different cell types, species, and strains can require different environmental factors to operate optimally, which is not possible to establish in co-culture. The propagation of noise, where chemical-based communication molecules can accumulate over time. In this chapter, investigations into an alternative, light-based intercellular communication mechanism are presented. In section 7.1, optogenetics and bioluminescence are explained. In section 7.2, results validating the behaviour of light senders (bioluminescent) and light receivers (optogenetic) are presented, whilst section 7.3 explains a microfluidic approach to experimentally testing optical communication. Finally, section 7.4 concludes findings from this chapter and discusses future next steps.

7.1. Introduction

7.1.1. Bacterial optogenetics

In nature, there are many examples of bacterial systems which are regulated by light. A classical example is that of bacteriorhodopsin, which is a bacterial proton pump driven by light[358]. Another example is the protein YtvA, which is a *Bacillus subtilis* receptor that responds to the presence of blue light, and which is involved in stress response signalling mechanisms. EL222 is a protein found in *Erythrobacter litoralis* which, upon dimerization, can bind DNA[359], [360]. The dimerization of EL222 only occurs in the presence of blue light, allowing for light regulated control of genetic expression.

Optogenetics, a field of light-regulated genetic expression, has been founded based on natural light-responsive mechanisms. Whilst optogenetics has traditionally been associated with neuroscience, it has become increasingly popular in synthetic biology[361]. The ability to use light as a method of controlling genetic expression, rather than the more commonly used chemical method, has several advantages. One such advantage is that light can be applied to a system transiently, which allows for dynamic

switching between activation and deactivation of genetic expression[360]. Such dynamic regulation is far more difficult with chemical analytes, as they cannot be easily removed from the system once added, and many commonly used chemicals, such as IPTG, have long half-lives. Another advantage is that spatial regulation, where different sections of a system require independent regulation, can be achieved more easily with light, as chemicals may diffuse through a system and 'bleed' into unwanted areas, whereas light has a higher spatial resolution[362], [363]. Systems which involve patterning have benefited greatly from optogenetics for this reason. Disadvantages of optogenetics include accidental background activation, where exposure to ambient light may inadvertently influence the system, and the requirement for electronics and specialist setup, which contrasts chemical induction where the chemical need simply be added to the system.

7.1.2. Bioluminescence

Bioluminescence is found abundantly in nature, especially in marine environments where animals use light to confuse predators or hunt for prey. Although many bacteria have been discovered which exhibit bioluminescence, the exact reasons for why bacteria display luminescence are not clear[364]. Bacterial luminescence tends to be conferred via an operon, the most common of which is the luciferase, or Lux, operon[365]. This operon encodes a set of enzymes which form three complexes: (i) a fatty acid reductase composed of LuxC, LuxD, and LuxE, (ii) the luciferase complex composed of LuxA and LuxB, and (ii) a flavin reductase which exists as a LuxG homodimer[366]. The fatty acid reductase converts long chain aldehydes produced by fatty acid metabolism into aldehydes. The LuxAB luciferase uses reduced flavin mononucleotide ($FMNH_2$) to oxygenate the aldehyde metabolites, the reaction for which generates light. The flavin reductase regenerates $FMNH_2$ by reduction of FMN.

Within synthetic biology, bioluminescence has been used as a reporter. Compared to the more commonly used fluorescent reporters, bioluminescence can have a greater dynamic range as background luminescence of cells is far lower than background fluorescence[367]. However, the vast majority of natural bioluminescence emits in the blue or green wavelengths, which may not be suitable for more complex systems. These systems require multiple reporters that must be distinguishable from each other[364]. Additionally, expression of bioluminescence can place a larger burden on

cells, as it tends to require expression of multiple enzymes, compared to fluorescence which can be achieved with a single protein.

7.1.3. Light-based intercellular communication

Engineering communication between cells in a synthetic biology system has almost universally focused on chemical-based signalling, such as the quorum sensing mechanisms discussed throughout this thesis[368], [369]. Whilst there has been much success with this approach, chemical-based signalling suffers from many of the issues of chemical-based regulation described in sub-section 6.1.1[370]. Additionally, chemical-based signalling requires the cells to be co-cultured in the same media and environment, which can prove problematic when working with different strains or species which differ in terms of their optimal conditions. Other issues can also arise from co-culturing, such as competition for resources causing the proportions of each population to change over time, and difficulties in distinguishing each population separately.

As optogenetics provided an alternative to chemical-based genetic regulation, so might light-based communication provide an alternative to chemical intercellular communication. In such a system, the 'senders' in the system would generate light via bioluminescence, and the 'receivers' could respond to the bioluminescence through the use of light-responsive transcription factors, such as the EL222 system described previously. Optical communication such as this could benefit not only from the advantages afforded to optogenetics, such as a more dynamic signalling system, but may also allow for the development of multi-microbial systems which do not rely on co-culturing. Additionally, different populations could be cultured separately in optically clear containers, through which the light could travel. The potential for physically-separated co-cultures would allow for optimisation of each population's environment, and also help prevent interpopulation competition. Although optical communication has been suggested previously in literature[371], and a few projects have attempted to engineer such light-based communication[372], [373], rigorous scientific experimentation still appears to be lacking. Recent work has suggested the potential for this approach in artificial cell communities[370].

7.2. Validation of Light Sender and Light Receivers

7.2.1. Optical communication system

To determine the feasibility of a light-based intercellular communication system, two cell types were defined: light senders and light receivers. The light senders were selected to generate blue bioluminescent light in response to an inducer, and the light receivers were selected to respond to blue light. The light senders consisted of *E. coli* DH5α cells expressing the Lux operon under the control of a *PBad* promoter, which was arabinose inducible. The light receivers expressed mCherry under the control of *PBLRep*. The *PBLRep* promoter is capable of constitutive expression, however when in its dimerised form, the EL222 protein can bind to the promoter sequence and block transcription[360]. Thus, the light receivers could only produce mCherry in the absence of blue light. As the bacterial luciferase has been previously reported to generate blue light, and the *PBLRep*-EL222 has been shown to respond to blue light, it was expected that the light senders and receivers should engage in unidirectional communication.

7.2.2. Luciferase operon characterisation

The bioluminescence of light senders over time with differing levels of induction was characterised to determine conditions for maximal light intensity. Light sender cells were prepared and characterised as described in section 2.8.1. It was found that luminescence was maximal between 4 and 6 hours post induction, after which the signal decayed to background levels (Figure 7.1 (A)). This demonstrated the transient nature of bioluminescence as both a reporter and a potential intercellular communication channel. It was also found that the light sender cells could be induced in a dose-dependent manner between approximately 0.04 μM and 0.625 μM of arabinose (Figure 7.1 (B)). When fewer than 0.04 μM of arabinose was added, no signal could be seen above background levels, and arabinose added above 0.625 μM had no impact on luminescence. The bacterial luciferase was observed to generate light across a broad range of wavelengths, although maximal intensity was between 440 nm and 520 nm. This range of wavelengths was similar to the wavelengths of light previously reported to activate EL222[374]. Thus, it was concluded that the Lux operon could allow for cells to emit light in the correct range of wavelengths to activate the EL222 light receiver system.

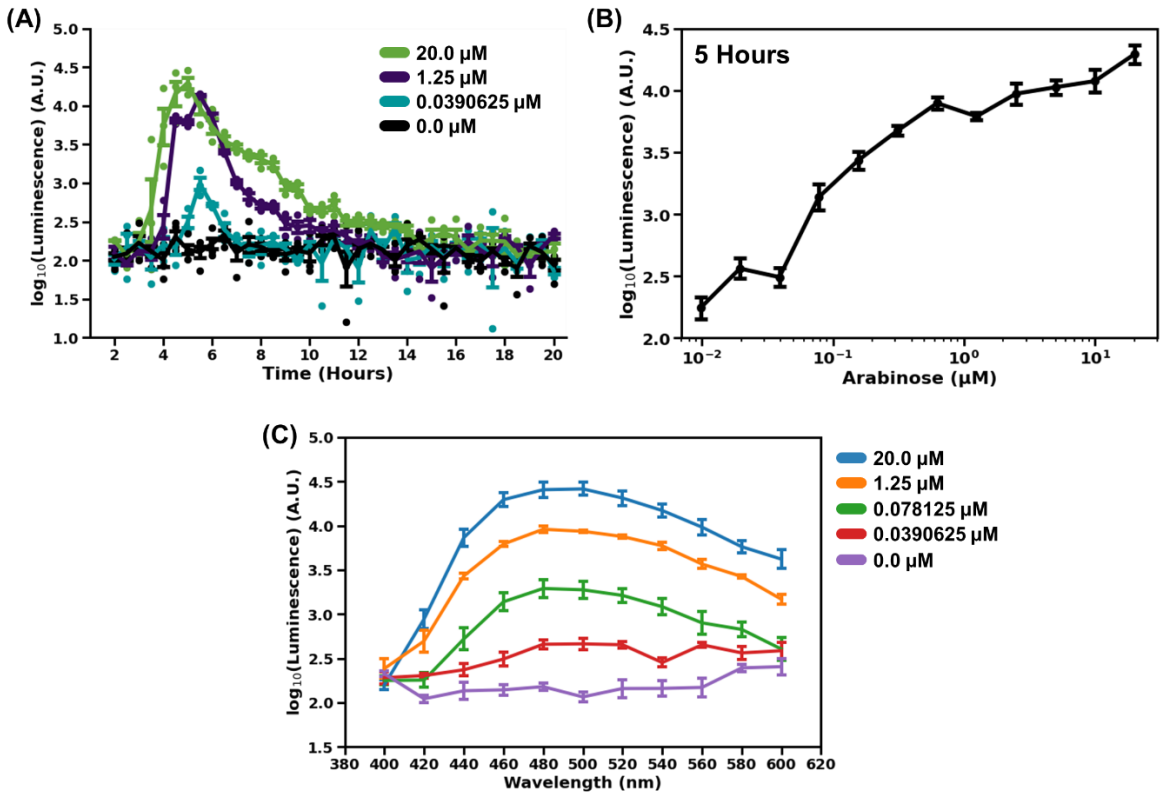


Figure 7.1. Light Sender Cell Characterisation
Characterisation of *E. coli* cells expressing the Lux operon. Error bars shows standard error of 3 or 4 replicates centred on the mean luminescence value. (A) Time course curve of $\log_{10}$ luminescence at 460 nm when induced with 20, 1.25, 0.04, or 0 µM of arabinose. Dots show individual luminescence values for each replicate. (B) Dose-response curve of arabinose concentration vs $\log_{10}$ luminescence at 460 nm, 5 hours post induction. (C) Spectral scan of cell cultures between 400 nm and 600 nm in 20 nm intervals, 5 hours post induction. Data shown for cells induced with 20, 1.25, 0.08, 0.04, or 0 µM of arabinose.

7.2.3. Comparison of bioluminescence to electronic light

Whilst it was demonstrated that bacterial luciferase could emit light in the appropriate wavelength range to activate EL222, it was not clear if the intensity of light emitted would be sufficient to cause a measurable response by the light receiver cells. To help determine whether the light sender cells would emit light at a great enough intensity, the bioluminescence was compared to electronic light from a Light Emitting Diode (LED). To this end, a light calibration plate device was prepared (Figure 7.2(A)). This device consisted of a black 96-well plate with clear, flat bottoms, a blue LED (Fedy Tech diffused ‘Piranha’ RGB; Adafruit 1451), and a 3-volt CR2032 battery. The LED and battery were wired together, with space for resistors to be added in series. The number of resistors was used to modulate the LED’s brightness, which allowed for better approximation of bioluminescence. Luminescence from the light calibration plate device was measured using the same plate reader and settings used for characterisation of the light sender cells. The LED brightness was measured when

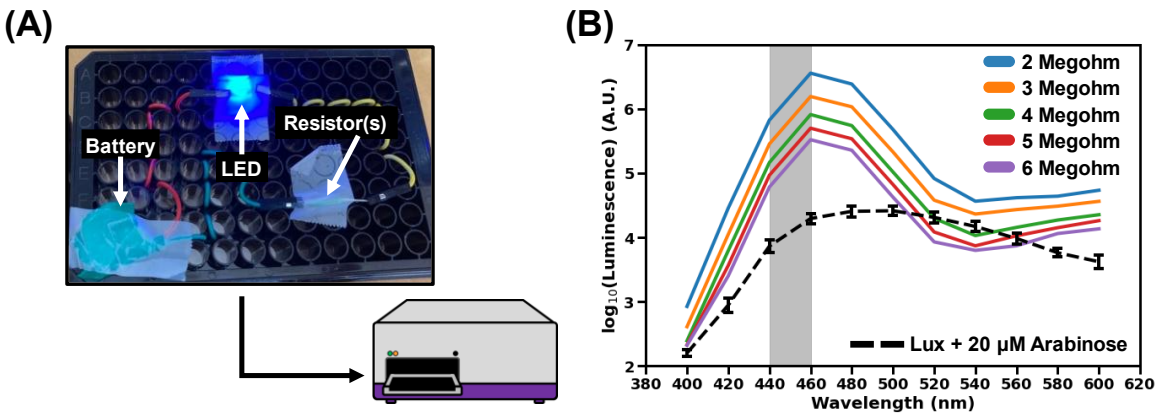


Figure 7.2. Comparing Bioluminescence to Electronic Light

Plate reader luminescence readings were calibrated to a blue LED. (A) Picture showing setup. An LED (Fedy Tech diffused 'Piranha' RGB; Adafruit 1451) was placed facing down into a 96-well plate. A 3-volt CR2032 battery was adhered to the plate, along with space for 1 megohm resistors connected in series. The plate was placed into a plate reader and luminescence readings were taken using the bottom luminometer. (B) Spectral scan of LED luminescence between 400 nm and 600 nm. Solid lines show luminescence of the LED with different numbers of resistors added. The dashed line shows bioluminescence of the light sender cells with 20 µM of arabinose added (Figure 7.1). The grey box shows the wavelengths at which EL222 is maximally activated[374].

different numbers of 1 megohm resistors (between 0 and 6) were added in series. When 0 or 1 resistor was added, the LED's brightness saturated the plate reader's detector, and hence this data was excluded. For the remaining data, it could be seen that the LED emitted light maximally between 440 and 480 nm (Figure 7.2(B)). The spectral properties of bacterial luciferase and the LED were similar, although the light sender cells showed luminescence across a broader spectrum. Additionally, the brightest light sender cultures showed luminescence approximately 10 times lower than the dimmest LED setup measured. It therefore needed to be determined whether the EL222 optogenetic system could be impacted by light at this low level.

7.2.4. Validation of EL222 optogenetic construct

To initially validate responsiveness of cells expressing the EL222 optogenetic construct to blue light, cells were grown in a 96 well plate exposed to either bright blue light or kept in complete darkness (Figure 7.3 (A)). The *PBLRep* promoter allows translation of downstream coding regions when EL222 is inactive (not exposed to light) but is blocked by EL222 in its active form (exposed to blue light). It was seen that over a period of 5 hours, cells exposed to blue light had red fluorescence repressed compared to cells kept in the dark, as was expected.

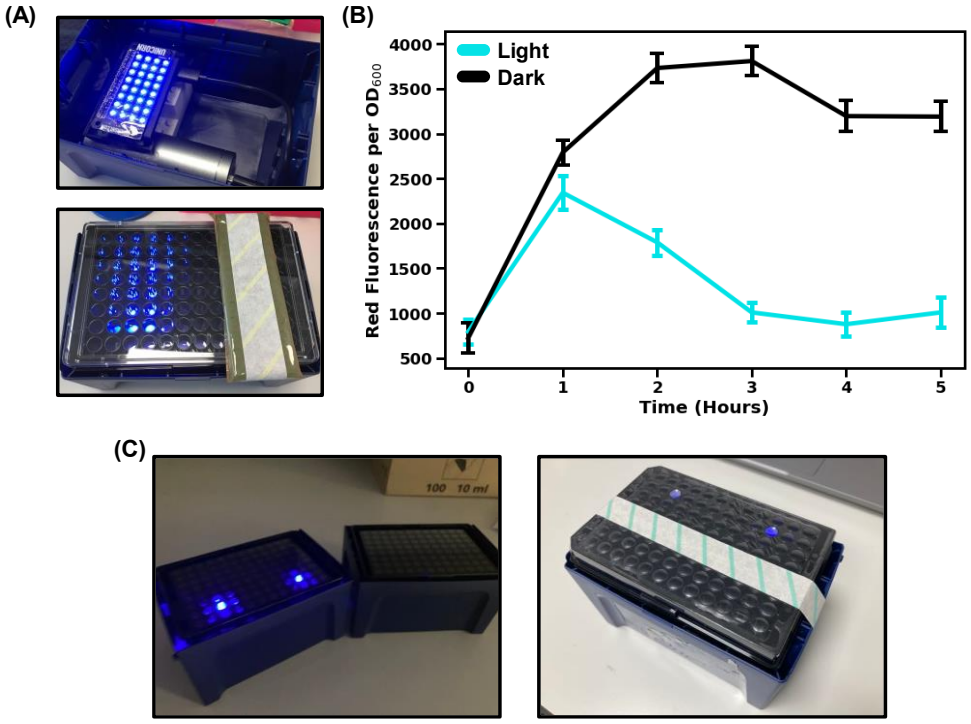


Figure 7.3. Validation of Light Receiver Cells' Response to Blue Light

The light receiver cells were cultured in the presence and absence of blue light. (A) Optogenetic setup. A Unicorn pHAT (Pimoroni) was connected to a Raspberry Pi Zero (Pimoroni), and 3 volts was supplied to all blue LEDs. The light emitting device was placed into a box, with the 96 well plate containing cell cultures on top. The box was covered in foil and experimentation was performed (section 2.8.2) (B) Time course curve of red fluorescence per cell density over 5 hours. Error bars show standard deviation across three replicates, centred on the mean. (C) Alternative optogenetic setup allowing for characterisation with calibrated LEDs. The LED circuits were powered individually as described in Figure 7.2 (A). For the setup in the first image on the left, no resistors were used. In the setup on the right in the first image, 6 megohms of resistance was applied to the LED circuits. The second image shows the setup with a 96-well culture plate placed on top.

An alternative optogenetic setup was used to determine whether the light receivers could respond to light at an intensity approaching that of the light sender cells. In this experiment, each well containing light receiver cells were exposed to either no light, or blue light from a single LED with 0 or 6 megohms of resistance. Plates were then covered in foil to prevent exposure to ambient light. Cells were prepared and characterised as described in section 2.8.3. Unlike in the previous experiment, there was no difference in fluorescence between the cells exposed to bright blue light, and the cells kept in total darkness (data not shown). The reason for this was not clear, as cells were prepared in the same ways, although it was likely related to the change in experimental setup. Regardless, the initial experiment demonstrated that the light receiver cells could respond to blue light, although the system appeared not to be robust.

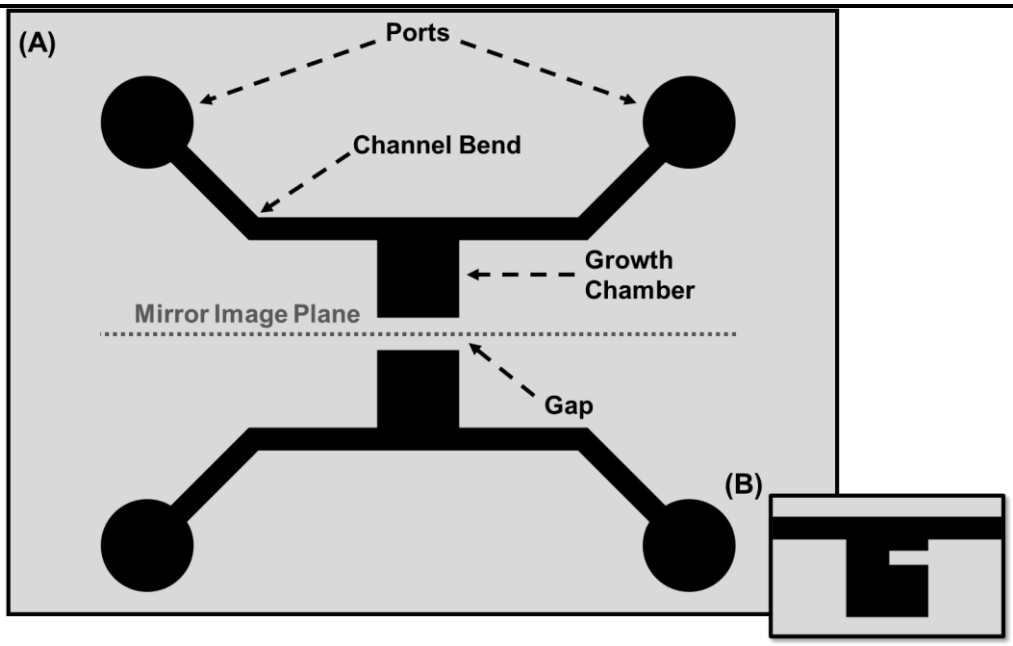


Figure 7.4. Optical Communication Microfluidic Chip Design
Basic design for the microfluidic chip. (A) Full device, consisting of two unconnected sections mirror imaged about the plane indicated by the dotted grey line. The lengths of the growth chambers were 100, 120, or 150 µm long, and 100 µm wide. The channels were 10 µm wide. The ports were 2 mm in diameter. The distance between each growth chamber was 10, 50, or 100 µm. The distance between each port and the channel bend and the distance between the channel bend and opening of the growth chamber were both 2 mm. The depth across the entire chip was 40 µm. (B) Growth chamber variant with a 'shelf'. The shelf was 10 µm wide and reached to the mid-point of the chamber (50 µm). The distance between the top of the shelf and growth chamber opening was 25 µm.

7.3. Microfluidic-Based Optical Communication Validation

7.3.1. Microfluidic chip design

A microfluidic chip was designed with the aim of validating optical communication between the light senders and receivers. The microfluidic chip was required to allow for each cell type to accumulate and grow within separate chambers but be positioned such that light could pass between the two chambers. The microfluidic device was based upon a previously reported design [375], where cells flow through a channel which has a chamber extending from one edge of the channel, as illustrated in Figure 7.4 (A). The optical communication microfluidic chip was designed to incorporate two channels of 10 µm width and 40 µm depth, with growth chambers extending from the centre of each. The channels were positioned as illustrated in Figure 7.4 (A), which allowed for the edge of each chamber to be facing one another, and hence light could pass between each chamber. The two channels each had an input and output port, where cells and fresh media could be flowed in one end of the chamber, and waste could be

collected from the other end. This allowed for continuous flow to be established, which can promote cell growth in microfluidic devices[376].

The microfluidic chips were designed using Autocad as described in section 2.8.4. Initially, three variants of the microfluidic chips were designed by modifying the length of the growth chamber, such that each chamber was 100 µm in width, and either 100 µm, 120 µm, or 150 µm in length. These variants were created as it has been shown previously that different chamber sizes can impact on trapping of cells. Prior to fabrication of the microfluidic chips, fluidic modelling was performed using ANSYS workbench (section 2.8.4) to help identify and problems relating to the designs. It was found that whilst cells did enter the growth chambers (Figure 7.5 (A-C)), there was the potential for significant escape to occur. To address this issue, another set of variants were designed. These variants incorporated a 'shelf' 50 µm long and 10 µm wide, 25 µm from the growth chamber opening (Figure 7.4 (B)). This feature was added with the aim of stopping cells from simply flowing out of the chamber and into the channel. Fluidic modelling indicated that the shelf feature did help prevent cell escape (Figure 7.5 (D-E)). Additionally, it was found that a slow vortex formed in the 100 µm long chamber, assisting with continual mixing of cells and nutrients from the media to assist with growth.

For all designs described above, a final three variants were generated for each by varying the distance between each growth chamber, such that the chambers were placed either 10 µm, 50 µm, or 100 µm apart (48 variants in total). The varied distances were incorporated to allow for characterisation of the relationship between light travel distance, and response by the light receivers.

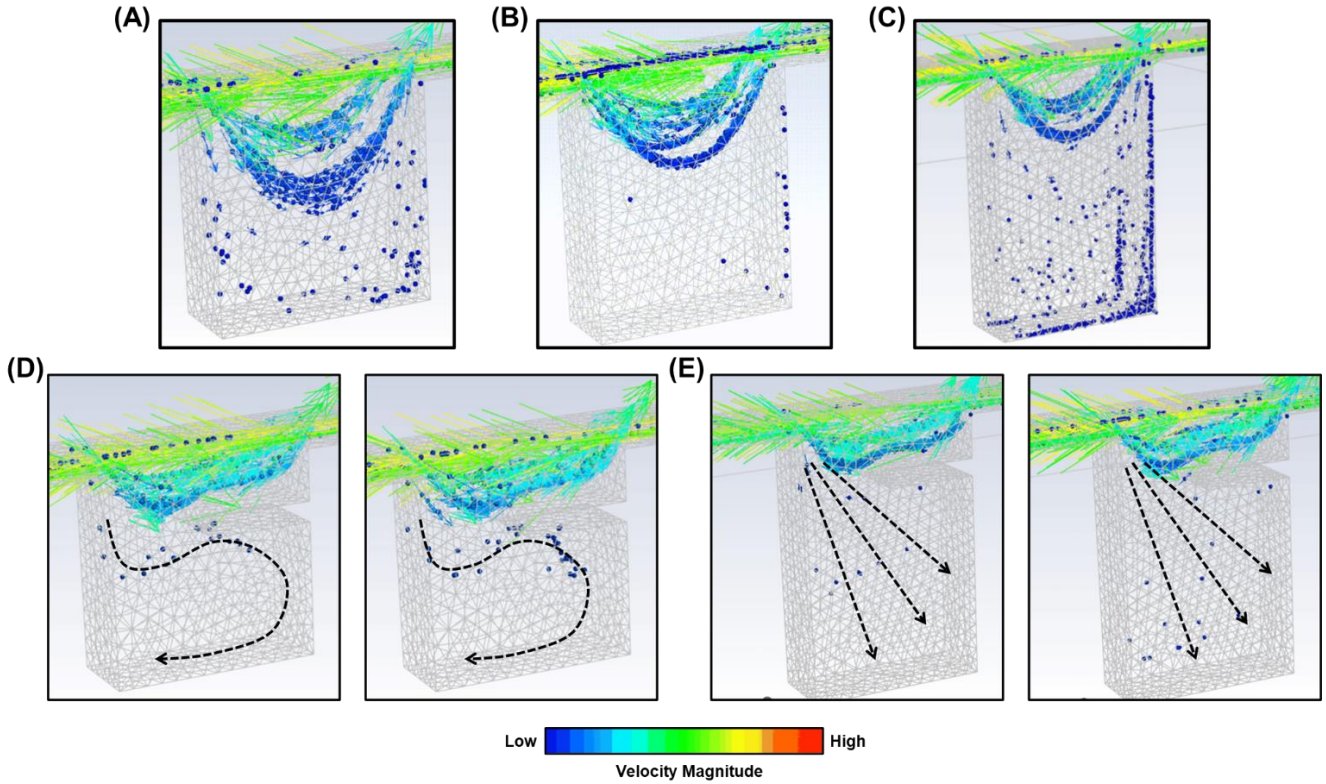


Figure 7.5. Fluidic Simulation Results

Images captured from fluidic simulation of microfluidic chip designs using ANSYS as described in section 2.8.4. For all images, direction of flow was from left to right. Spheres represent cells and were coloured according to velocity. Solid arrows show the cell's direction of travel, and the size and colour of each arrow were determined by the cell's velocity. (A-C) Images showing the state of 100 (A), 120 (B), and 150 (C) µm long growth chambers after 10 minutes of simulation time. (D-E) Images showing the state of 100 (D) and 150 (E) µm long growth chambers with shelf after approximately 5 (left) and 10 (right) minutes of simulation time. Dashed black arrows show general flow patterns within each chamber.

7.3.2. Microfluidic chip fabrication and cell culturing

The microfluidic designs were fabricated as described in section 2.8.5. It should be noted that initially, PDMS was selected to fabricate the microfluidic chips from. However, due to supply chain issues, PDMS could not be acquired (see COVID impact statement). As an alternative, silicone resin was used to allow for initial testing of cell culturing within the microfluidic chips (Figure 7.6 (A)). However, silicone is far less optically clear than PDMS, and hence was not suitable for testing optical communication. To begin testing of cell culturing, the fabricated chips were first imaged under a microscope to check for damaged features. It could be seen that all features had good resolution and were as designed (Figure 7.6 (B)).

The fabricated microfluidic chips were tested to ensure their viability for cell growth. Initially, the chips were tested by flowing *E. coli* DH5α cell culture through the channels to ensure cells could be trapped in the chambers. It was found that the designs containing longer growth chambers (120 and 150 μm) were prone to air bubbles becoming trapped at the end of the chamber (Figure 7.6 (C)). Therefore, only the designs with 100 μm long growth chambers were used for subsequent experiments. Additionally, it was observed that the 100 μm long growth chambers with shelf appeared to allow for better retention of cells, however it appeared that performance increased when liquid flow was in the opposite direction to that tested during fluidic simulation. Therefore, subsequent experiments made use of the 100 μm long growth chamber with shelf designs, with liquid flow in the direction of right to left with respect to Figure 7.4 (B) and Figure 7.5 (D).

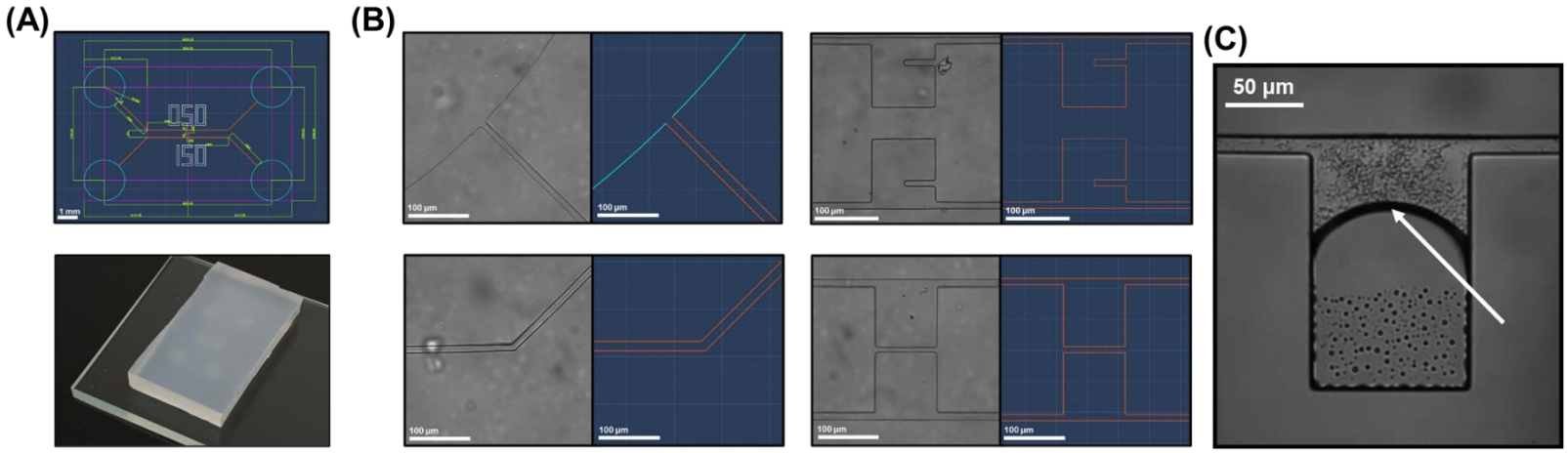


Figure 7.6. Validation of Microfluidic Chip Fabrication

Initial validation of fabricated microfluidic chip feature accuracy and functionality. (A) Images showing the design for a single microfluidic device in Autocad (top) and the fabricated device (bottom). (B) Phase contrast images of microfluidic chip features using a Nikon Ti microscope with a 40x objective with 1.5x zoom (left). Images on the right show the corresponding feature design in Autocad. (C) Phase contrast image (40x objective with 1.5x zoom) showing *E. coli* DH5α cells in a microfluidic chip with 150 µm growth chamber. White arrow indicates liquid-air interface.

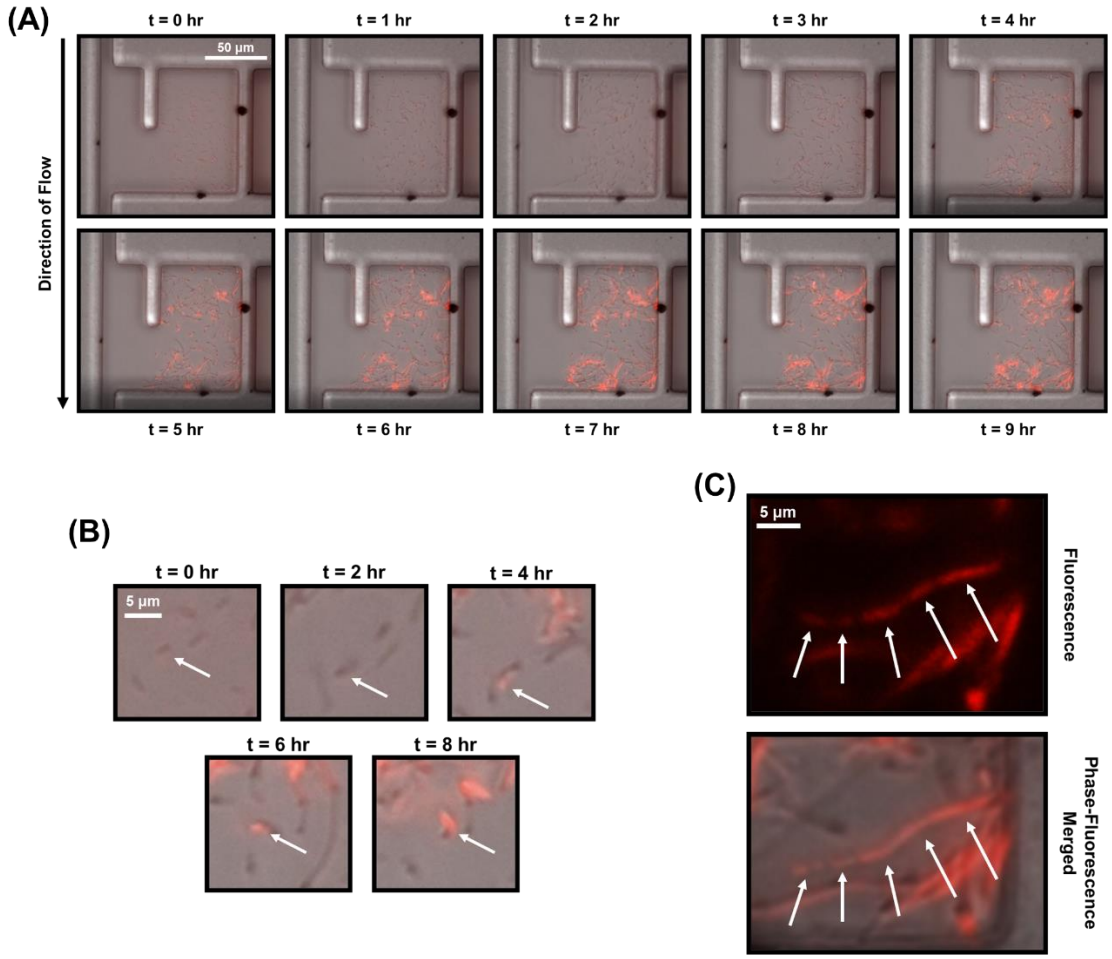


Figure 7.7. Cell Growth and Red Fluorescence within Microfluidic Device
Default processor cells from the Sensynova framework were induced to produce mCherry and trapped in a microfluidic growth chamber. (A) Merged phase contrast and red fluorescence images taken by a Nikon Ti2 microscope with 40x objective and 1.5x zoom. Images were taken over a 10-hour time course. (B) Tracking of an individual cell throughout the images shown in (A). (C) Red fluorescence (top) and merged phase contrast and red fluorescence (bottom) images after 8 hours of growth showing an *E. coli* cell displaying failed division and cytoplasmic condensation. White arrows indicate segments of segregated cytoplasm.

As the light receiver cells were engineered to express mCherry in the absence of blue light, it was therefore necessary to ensure red fluorescence could be detected in the microfluidic device. To this end, the default processor cells developed for the Sensynova framework were cultured overnight, mixed with 10 µM C12-HSL to induce mCherry production, and loaded into a microfluidic device (2.8.7). Fresh LB media supplemented with 10 µM C12-HSL and chloramphenicol was flowed at a constant rate (approximately 0.1 µL/min) through the chip for 10 hours, with phase contrast and red fluorescence images taken every 30 minutes by a Nikon Ti2 microscope with 40x objective and 1.5x zoom (section 2.8.7). The microscope chamber was kept at 37ºC for the duration of the experiment. As expected, red fluorescent cells were observed,

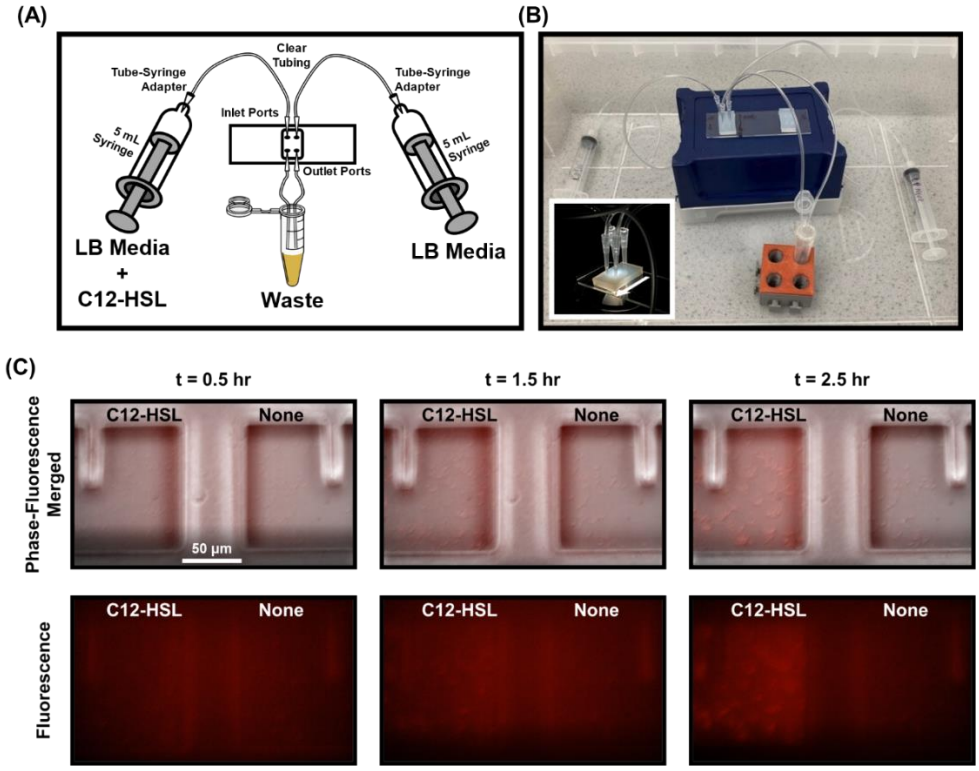


Figure 7.8. Induction of Fluorescence in Microfluidic Device

Induction of default processor cells in the optical communication microfluidic device. (A-B) Schematic (A) and picture (B) of experimental setup following loading of default processor cells into the growth chambers. The tube-syringe adapters were 200 µL pipette tips cut to size. The inlet and outlet ports were 10 µL pipette tips cut to size. The syringes were controlled by automatic pumps. Inset in (B) shows microfluidic device positioned in the microscope. White arrow indicates flow direction. (C) Time lapse images of red fluorescence (bottom) and merged phase contrast and red fluorescence (top) taken by a Nikon Ti2 with a 40x object and 1.5x zoom.

with increasing fluorescence observed over time (Figure 7.7 (A-B)). However, some *E. coli* cells appeared to fail to replicate correctly, and instead formed elongated chains with evidence of cytoplasmic condensation[377] (Figure 7.7 (C)). As mCherry accumulated in the cytoplasm, it was possible to visualise individual cytoplasm sections. This behaviour of both disrupted cell division and cytoplasmic condensation have been shown to occur in the presence of DMSO and chloramphenicol previously[377], [378]. Therefore, as chloramphenicol was used as the selection pressure to retain the default processor module, and DMSO was used as the solvent for C12-HSL, it was likely that these additives were the cause. Although cell division was somewhat impacted, cell activity was apparent due to the increase in red fluorescence over time.

Although it was seen that the default processor cells produced mCherry over time, it was not clear whether this behaviour was due to induction with C12-HSL or the result of background expression. This uncertainty stemmed from observations of high

background mCherry production over time discussed in chapter 5. It was important to establish whether cells could be induced within the microfluidic device, as the light sender cells required induction by arabinose to produce bioluminescence. Therefore, a second experiment was conducted in which default processor cells were grown overnight but were not mixed with C12-HSL. Instead, cells were loaded into both chambers of a single fluidic device, and LB media with only chloramphenicol added was flowed through one section, whilst LB media with chloramphenicol and 10 μM C12-HSL was flowed through the other (Figure 7.8 (A-B)). It was observed that cells in the chamber with C12-HSL in the media displayed greater fluorescence than those in the chamber with no C12-HSL, suggesting that induction of cells within the microfluidic device was possible (Figure 7.8 (C)).

7.4. Conclusions and Future Work

The use of multi-microbial systems in synthetic biology relies heavily upon intercellular communication mechanisms, the vast majority of which make use of diffusible chemical molecules [175], [379]–[381], or at least require cells to occupy the same physical space [382]. The Sensynova framework presented in this thesis make use of such chemical-based communication in the form of quorum sensing mechanisms. In this chapter, efforts were made to investigate a light-based intercellular communication mechanism which would not require co-culturing of cells.

Two cell types, a light sender and a light receiver, were identified to help validate optical communication. The light sender consisted of an inducible bacterial luciferase pathway which could generate blue light. The light receiver used the EL222 light-activated transcription factor which could prevent transcription of a fluorescent protein when active. As the EL222 protein responded to blue light, it was thought that the bacterial luciferase could activate EL222.

To validate appropriateness of the bacterial luciferase as a light sender, it was necessary to ensure that light emitted was in the range of wavelengths known to activate the EL222 protein. In accordance with previous studies [366], it was found that bacterial luciferase emitted light maximally in the range of 440 nm to 520 nm. This range corresponded with wavelengths known to activate EL222 [374], indicating the potential for bacterial luciferase to act as a light sender.

Whilst bacterial luciferase had been shown to emit light in appropriate wavelengths, it was not clear whether the intensity of light was high enough to activate EL222. It was therefore necessary to calibrate the brightness of the luciferase against a known light source. This light source could then be used to characterise the EL222 light receivers and determine their limit of sensitivity. Whilst chemical standards exist for calibrating fluorescence [63], appropriate luminescence standards have not yet been validated. Instead, a custom calibration plate was built, which allowed for calibration of bioluminescence against electronic light in a plate reader. Intensity of the electronic light was modulated via resistance in the circuit, and raw luminescence was compared to that of the luciferase. Whilst the luciferase was found to emit light below the measured electronic light intensities, it was possible to obtain a base sensitivity

threshold for the EL222 system. Although initial characterisation demonstrated responsiveness of the EL222 system to light, the optogenetic setup used to measure sensitivity failed to generate a response, even when high intensity light was used. Therefore, future work should aim to optimise this setup to ensure robustness, and hence provide a method for establishing the minimum bioluminescent intensity required to activate EL222-based light receivers.

If future work indicated that the bacterial luciferase tested in this project was too dim, a brighter variant of the Lux operon should be investigated, such as the iLux operon[383]. Alternatively, BRET (Bioluminescent Resonance Energy Transfer) could be employed to increase the light intensity emitted[384]. BRET would occur by co-expressing a bright fluorescent protein with emission at the wavelength required for activation of the EL222 system, and excitation within the range of wavelengths emitted by the luciferase.

To enable implementation and characterisation of an optical communication mechanism, and microfluidic device was developed. This novel device would allow for physically separated culturing of light senders and receivers, leveraging the ability to establish multi-microbial systems without the need for co-culturing. The microfluidic device was shown to be suitable for growth of bacterial cells over time, and it was successfully demonstrated how cells could be induced within the device. However, the microfluidic chips had to be fabricated from silicone resin rather than PDMS due to a lack of availability. PDMS was initially selected as it is optically clear, however the chemical could not be obtained. Therefore, the culturing experiments had to use microfluidic devices fabricated from the far less transparent silicone resin. Thus, it was not possible to use the microfluidic device to validate the potential for optical communication. Future work should aim to fabricate the designs from PDMS.

To summarise, initial characterisation of a potential light sender and light receiver was performed, with the aim of establishing suitability for implementing optical communication. Additionally, a microfluidic device was developed to allow for future validation of light-based intercellular communication.

Chapter 8. Conclusions and Future Work

8.1. Summary of Research Objective

The research objective of this thesis was to demonstrate how the principles of high-level modularity and synthetic multi-microbial systems could be used to aid in the development of a specific type of synthetic biology device: biosensors. In chapter 1, bioengineering and synthetic biology were introduced. The strengths and challenges associated with synthetic biology approaches towards the development of biological systems and devices were discussed, using examples of previous studies. It was identified that although synthetic biology has shown much promise, the development and optimisation of systems is difficult, and implementation of engineering principles can be inefficient. A particular area of interest was highlighted: genetic biosensors. These types of biosensors represent a commonly developed biological system and have wide-reaching applications, although their development was found to suffer similarly to other synthetic biology systems. In this thesis, it was investigated how high-level modularity and multi-microbial systems could be used to provide alternative approaches towards biosensor development and optimisation. Specifically, focus was given to identifying novel and easily accessible design spaces and developing a framework to promote more efficient use of engineering principles in development. Throughout the thesis, the findings, outcomes, limitations, and potential future avenues were discussed at the end of each chapter. In this chapter, the impact of the project as a whole is considered, and the major outcomes highlighted.

8.2. Summary of Previous Work

As discussed in section 3.1, there is a large potential for the combination of high-level modularity and multi-microbial systems to aid with the development of biological devices. However, previous studies which have used modular multi-microbial systems had various limitations, such as the requirement for complex hardware and very specific applications (section 3.1.2). More broadly, however, these attempts focused only on presenting the functionality of synthetic biology systems implemented using their approaches, rather than developing a framework and associated tools. Moreover, many examples did not consider engineering principles other than those inherently linked to modularity, including computationally informed experimentation and optimisation (section 3.1.2). The work in this thesis has considered the benefits, disadvantages, and challenges of modular and multi-microbial systems which require

uni-directional signal propagation. An overview of the insights and conclusions gained from work detailed in this thesis is given throughout the remainder of this chapter.

8.3. Tools for Enhancing a High-Level Modular and Multi-Microbial Framework

In this thesis, the focus was not simply on determining whether a genetic biosensor could be developed using the Sensynova framework. Instead, focus was also placed on investigating how such an approach could be enhanced using a range of tools and resources to promote the use of engineering principles.

In section 3.3.1, it was identified that the Synthetic Biology Open Language (SBOL), which is commonly used to develop and share synthetic biology designs, could not be used to represent multi-microbial systems. As discussed in section 1.2, re-usability and reproducibility are key principles not only in modularity, but also synthetic biology more widely. The ability to share information easily and accurately about a design and its intended function are key to such principles. To aid in this endeavour, a proposal of how the SBOL data model could be extended to capture information about multi-microbial systems, along with a set of best practices for representing such information was researched and presented (section 3.3). Also proposed were methods for storing information about cells and other chassis in SBOL, as this was not currently possible. These proposals were accepted by the SBOL community, and as such, multi-microbial systems and contextual information surrounding cells and chassis can now be represented in SBOL (section 3.3.7). Acceptance of the proposals was the first major outcome of this project, as it provided a method for users to capture information not only about Sensynova compatible modules and biosensors, but other modular and multi-microbial approaches more widely.

In section 3.4.1, it was discussed how automation could aid high-throughput development of synthetic biology systems and allow for easier sharing and reproduction of experiments. However, it was also found that the development of automation protocols can be difficult, and often pre-existing protocols cannot be easily adapted (section 3.4.1). Further, it was discussed how although previous efforts to aid in automation protocol generation have been made, these tools are either poorly documented or only provide an alternative language with which to write protocols, with no support for trivial protocol generation (section 3.4.1). Some tools were identified which allowed for easy generation of automation protocols, however they were towards

highly specific applications. Presented in section 3.4 was the second major outcome of this project: a Python library (BiomationScripter) which provided not only tools to aid in writing automation protocols for a wide range of workflows, but also Templates which could quickly generate protocols for common applications based on a few user inputs. Additionally, optional parameters allowed for flexibility and the potential for optimisation of protocols generated from Templates, and support was provided to allow users to create custom Templates. Documentation with walkthrough examples were made available to aid with uptake of BiomationScripter. Within the context of the Sensynova framework, a BiomationScripter-enabled Template for characterisation of modules and multi-microbial biosensors was developed and used throughout the project, allowing for simpler replication of experiments, and standardised generation of characterisation protocols. This represented a major difference from previously described implementations of high-level modular and multi-microbial synthetic biology, which did not make use of flexible automation.

The use of computational simulations has become popular in synthetic biology, as it can aid guided design and experimentation (section 4.1). In chapter 4, a proof-of-concept biosensor and its constituent modules were modelled and simulated using both deterministic and agent-based approaches. The Simbiotics platform was used to model the multi-microbial biosensor as it allowed for different cell types to have behaviour defined in a modular fashion, using the Systems Biology Markup Language (SBML) standard. In section 4.4, it was discussed how the standard SBML solver implemented within Simbiotics was not appropriate for simulation of complex designs, such as the ones developed here. Thus, a different method of simulation SBML models was implemented, adapting Simbiotics towards the needs of the Sensynova framework. Through simulating the modules and biosensor it was found that noise could easily propagate through the system, but an easily accessible design space, cell ratios, could be used to optimise the biosensor's functionality. As was discussed in section 6.4, the cell ratios design space has been shown previously to have importance but has been underutilised.

Whilst the Simbiotics platform allowed for some basic *in silico* interrogation of the Sensynova concept, it was found that the computational expense of simulating the system was high. This expense was found to be a major limitation as it was not feasible to rapidly explore large areas of the design space, and approaches such as sensitivity

analysis and parameter scanning were severely hindered. Therefore, a key finding from this section of work was that although the hybrid deterministic and agent-based modelling approach employed can be highly representative of biological systems, approaches which are less computationally expensive are vital in aiding the development of synthetic multi-microbial consortia.

8.4. Validating a Proof-Of-Concept Sensynova Biosensor

Through the use of the tools developed in this thesis, a proof-of-concept modular and multi-microbial biosensor was developed. Initially, the biosensor modules were characterised separately before combining into a co-culture. Section 5.2 described how information from computational models was used to help validate module functionality, and how BiomationScripter trivially generated protocols for automating experimental setups. Further, all data presented in chapter 5 was calibrated according to a standard protocol, which would allow for data collected by different researchers using different equipment to be compared more accurately. This was the first time that a modular and multi-microbial approach made use of standardised and automated characterisation protocols and calibrated data to help ensure reproducibility and re-usability, in conjunction with computationally informed experimentation. Future work should aim to conduct interlaboratory studies to determine the extent to which these approaches aided with increasing reproducibility and easing implementation.

From the experimental results, it was found that each module displayed different time-course fold-change behaviour when comparing induced and uninduced cells. Whilst the detector and reporter modules appeared to show increased fold change over time until a plateau was reached, the processor module instead showed that after an initial increase, fold change then began to decrease. This behaviour was found to be due to background activation of uninduced processor cells over time. Based on the design of the processor module and the potential for cross-talk between the two quorum sensing mechanisms used in the Sensynova platform, it was hypothesised that a positive feedback loop was amplifying leaky expression. This finding highlights the importance of selecting the correct communication channels when designing synthetic microbial consortia.

In section 5.2.3, the impact of noise propagation was characterised using BiomationScripter-generated protocols. Whilst the agent-based model indicated that

noise propagation may be an essential component of multi-microbial systems which rely on signal transfer through intercellular communication (section 4.4.4), this has not been thoroughly explored previously. Here, it was found that activation of cell types through background production of intercellular communication molecules was significant, and hence should be addressed when considering optimisation of multi-microbial systems. Indeed, an initial implementation of the multi-microbial biosensor, where all cell types were added in a 1:1:1 ratio, showed that whilst the system was functional, as time progressed background noise increased, and the signal become indifferentiable from the noise. This was thought to be due to the positive feedback loop exhibited by the processor cells, where background expression of the C4-HSL molecule saturated the response of the downstream reporter cells. The behaviour documented in this thesis demonstrates how noise can propagate through uni-directional cell-to-cell communication channels via feedback loops to saturate the response of downstream populations. Therefore, a key finding from this thesis is that if synthetic consortia are to become increasingly complex as the field of biocomputing advances, it is important that interventions to prevent noise amplification are developed. Such interventions could take a number of forms, including the development or discovery of fully orthogonal communication pathways, or methods to increase the transience of communication 'messengers' (such as quorum sensing molecules) to prevent background accumulation.

The impact of cell ratios on biosensor functionality was investigated experimentally in section 6.2, using predictions made by the agent-based model and making use of BiomationScripter to automate the testing process. It was found that cell ratios did indeed have a significant impact on response characteristics of the proof-of-concept multi-microbial biosensor. Additionally, results indicated that the design space section predicted by the agent-based model to contain the most optimal cell ratios was accurate, although further testing with a greater number of ratios would be required to confirm this finding. Nevertheless, these results indicated the usefulness of agent-based modelling in the optimisation of multi-microbial biosensors, and further development of the model to include more accurate parameters may allow for more precise predictions. Aside from the cell ratios design space, a computationally driven, multifactorial approach was taken to determine factors which most heavily impact on the functionality of individual modules (section 6.3). Results of the initial screening design indicated which factors should be investigated further in future experiments.

The success of using cell ratios as a design space to modulate the behaviour of a synthetic consortium has implications for the development of future synthetic biology systems and devices. The findings in this thesis makes clear that cell population composition is important even in multi-microbial systems which have not been designed to be impacted by changing cell ratios, such as previously described predator-prey systems[385]. Further, this work has shown that stable cell populations present in equal ratios is not always the most optimal configuration for synthetic consortia, and thus efforts should be made to properly consider this aspect of multi-microbial systems developed in the future.

8.5. Optical Intercellular Communication

In chapter 7, the limitations of chemical-based intercellular communication, such as the requirement for co-culturing, was discussed. A potential alternative communication method based on light was identified as an approach to address these concerns by allowing cells to grow separated by a physical barrier, but still maintain communication (section 7.1.3). As was also discussed in section 7.1.3, optical communication has been suggested previously and identified as a crucial technology in the development of multi-microbial communities but has had limited success in implementation.

In this project, strides were made towards implementing a method of optical communication. Mechanisms for light sender and receiver cells were identified and attempts were made to validate their feasibility (section 7.2). Additionally, a microfluidic device was designed, modelled, and fabricated which could allow for characterisation of optical communication. Whilst this device could not be fully tested due to difficulties in sourcing PDMS (see the COVID impact statement), the ability to grow and induce bacterial cells within the device was demonstrated.

Within this thesis, a method of calibrating bioluminescence to electrically generated light using a microplate reader was demonstrated. This method can be used to rapidly determine if a physically separated bioluminescent cell population would be able to control an optogenetic cell population. It is hoped that the efforts made here will aid in future work to develop a light-based cell-to-cell communication mechanism.

8.6. Future Work

The work presented in this thesis laid the foundations for a high-level modular and multi-microbial framework and demonstrated how it could aid development of a proof-of-concept biosensor. Whilst the approaches and principles researched here were shown to have promise, there remains future work which may be conducted based on the presented findings. Whilst further work has been discussed in each chapter for individual parts of the project, listed here are more general areas which require further investigation to help direct future efforts in this field.

Firstly, it is recommended that the interoperability of high-level modules designed and implemented according to the framework should be investigated. To achieve this, module variants could be rapidly assembled using automation protocols generated by BiomationScripter. The construction of a multitude of biosensors could then be created through co-culture of module variants to demonstrate not only how variants for each module type (detector, processor, reporter) can be easily interchanged, but the impact on biosensor response characteristics could be measured and reported on. These endeavours could be assisted by agent-based modelling, where deterministic models for each module variant are created and simulated using the Simbiotics platform to predict biosensor behaviours.

Secondly, as mentioned previously, conduction of interlaboratory studies would aid in determining how the availability of resources developed in this project, such as SBOL representation of multi-microbial systems and BiomationScripter generated characterisation protocols, impact on reproducibility of Sensynova biosensors. These studies could also help determine further barriers to re-usability and reproducibility and guide future efforts.

Thirdly, although the computational models presented here provided invaluable insight concerning optimisation of multi-microbial systems, the use of experimentally derived parameters would allow for more accurate predictions. It is also recommended that alternative agent-based modelling software which allows for less computationally intensive simulation of complex, SBML-defined multi-microbial systems be identified or developed. Faster simulation times would allow for better exploration of design spaces and variants *in silico* and could allow for cell growth to be added to the model.

Finally, limitations of multi-microbial systems were found to largely stem from issues related to co-culturing of different cell types. These findings prompted investigation of optical intercellular communication. Further efforts in this area could aim to use the approaches presented in chapter 7 for the purposes of identifying appropriate mechanisms for light production and reception. Additionally, as the microfluidic devices developed in this project showed promise, future work could use these designs to aid in validating optical communication mechanisms.

8.7. Conclusions

In this project, it was shown how a biosensor design could be split into three functional modules. It was then demonstrated that each biosensor module acted as a functional, high-level module by displaying its own functionality. Further, each module was implemented into different cells, and a biosensor system was assembled via co-culturing of each modular cell type. Tools and resources were developed and their application in aiding the implementation of a modular and multi-microbial biosensor demonstrated. By building on previous approaches at implementing high-level modularity and multi-microbial systems in synthetic biology, the work presented in this thesis has provided extra insight and tools to aid in the development of biological systems, and more efficient use of engineering principles.

Chapter 9. Supplementary Information

9.1. BiomationScripter Plate Calibrant Protocol

```
def run(protocol):
    Custom_Labware_Dir = "C:/Users/bradl/OneDrive - Newcastle
University/Nextcloud/Private/Automation/Opentrons_Labware_Definitions"
    Starting_20uL_Tip = "A1"
    Starting_300uL_Tip = "A1"
    Calibrants = [
        "Fluorescein",
        "Sulforhodamine 101",
        "Cascade Blue",
        "Microspheres"
    ]
    Calibrants_Stock_Concs = [
        10,
        2,
        10,
        3e9,
    ]
    Calibrants_Initial_Concs = [
        10,
        2,
        10,
        3e9
    ]
    Calibrants_Solvents = [
        "PBS",
        "PBS",
        "Water",
        "Water",
    ]
    Calibrant_Aliquot_Volumes = 500
    Solvent_Aliquot_Volumes = 5000
    Volume_Per_Well = 100
    Repeats = 2
    Calibrant_Labware_Type = "3dprinted_24_tuberack_1500ul"
    Solvent_Labware_Type = "3dprinted_15_tuberack_15000ul"
    Destination_Labware_Type = "greiner655087_96_wellplate_340ul"
    Trash_Labware_Type = "axygen_1_reservoir_90ml"
    Solvent_Mix_Before = None
    Solvent_Mix_After = None
    Solvent_Source_Touch_Tip = True
    Solvent_Destination_Touch_Tip = True
    Solvent_Move_After_Dispense = "well_bottom"
    Solvent_Blowout = "destination well"
    First_Dilution_Mix_Before = (10, "transfer_volume")
    First_Dilution_Mix_After = (10, "transfer_volume")
    First_Dilution_Source_Touch_Tip = True
    First_Dilution_Destination_Touch_Tip = True
    First_Dilution_Move_After_Dispense = False
    First_Dilution_Blowout = "destination well"
    Dilution_Mix_Before = (10, "transfer_volume")
```

```
    Dilution_Mix_After = (10, "transfer_volume")
    Dilution_Source_Touch_Tip = True
    Dilution_Destination_Touch_Tip = True
    Dilution_Move_After_Dispense = False
    Dilution_Blowout = "destination well"
    Mix_Speed_Multipler = 2
    Aspirate_Speed_Multipler = 1
    Dispense_Speed_Multipler = 1
    Blowout_Speed_Multiplier = 1
    Dead_Volume_Proportion = 0.95
    Calibration_Protocol = Templates.Standard_iGEM_Calibration(
        Calibrants = Calibrants,
        Calibrants_Stock_Concs = Calibrants_Stock_Concs,
        Calibrants_Initial_Concs = Calibrants_Initial_Concs,
        Calibrants_Solvents = Calibrants_Solvents,
        Calibrant_Aliquot_Volumes = Calibrant_Aliquot_Volumes,
        Solvent_Aliquot_Volumes = Solvent_Aliquot_Volumes,
        Volume_Per_Well = Volume_Per_Well,
        Repeats = Repeats,
        Calibrant_Labware_Type = Calibrant_Labware_Type,
        Solvent_Labware_Type = Solvent_Labware_Type,
        Destination_Labware_Type = Destination_Labware_Type,
        Trash_Labware_Type = Trash_Labware_Type,
        Solvent_Mix_Before = Solvent_Mix_Before,
        Solvent_Mix_After = Solvent_Mix_After,
        Solvent_Source_Touch_Tip = Solvent_Source_Touch_Tip,
        Solvent_Destination_Touch_Tip = Solvent_Destination_Touch_Tip,
        Solvent_Move_After_Dispense = Solvent_Move_After_Dispense,
        Solvent_Blowout = Solvent_Blowout,
        First_Dilution_Mix_Before = First_Dilution_Mix_Before,
        First_Dilution_Mix_After = First_Dilution_Mix_After,
        First_Dilution_Source_Touch_Tip = First_Dilution_Source_Touch_Tip,
        First_Dilution_Destination_Touch_Tip =
First_Dilution_Destination_Touch_Tip,
        First_Dilution_Move_After_Dispense =
First_Dilution_Move_After_Dispense,
        First_Dilution_Blowout = First_Dilution_Blowout,
        Dilution_Mix_Before = Dilution_Mix_Before,
        Dilution_Mix_After = Dilution_Mix_After,
        Dilution_Source_Touch_Tip = Dilution_Source_Touch_Tip,
        Dilution_Destination_Touch_Tip = Dilution_Destination_Touch_Tip,
        Dilution_Move_After_Dispense = Dilution_Move_After_Dispense,
        Dilution_Blowout = Dilution_Blowout,
        Mix_Speed_Multipler = Mix_Speed_Multipler,
        Aspirate_Speed_Multipler = Aspirate_Speed_Multipler,
        Dispense_Speed_Multipler = Dispense_Speed_Multipler,
        Blowout_Speed_Multiplier = Blowout_Speed_Multiplier,
        Dead_Volume_Proportion = Dead_Volume_Proportion,
        Protocol=protocol,
        Name=metadata["protocolName"],
        Metadata=metadata,
        Starting_20uL_Tip=Starting_20uL_Tip,
        Starting_300uL_Tip=Starting_300uL_Tip,
    )
    Calibration_Protocol.custom_labware_dir = Custom_Labware_Dir
```

```
Calibration_Protocol.run()
```

9.2. Plate Reader Calibrant Standard Curves

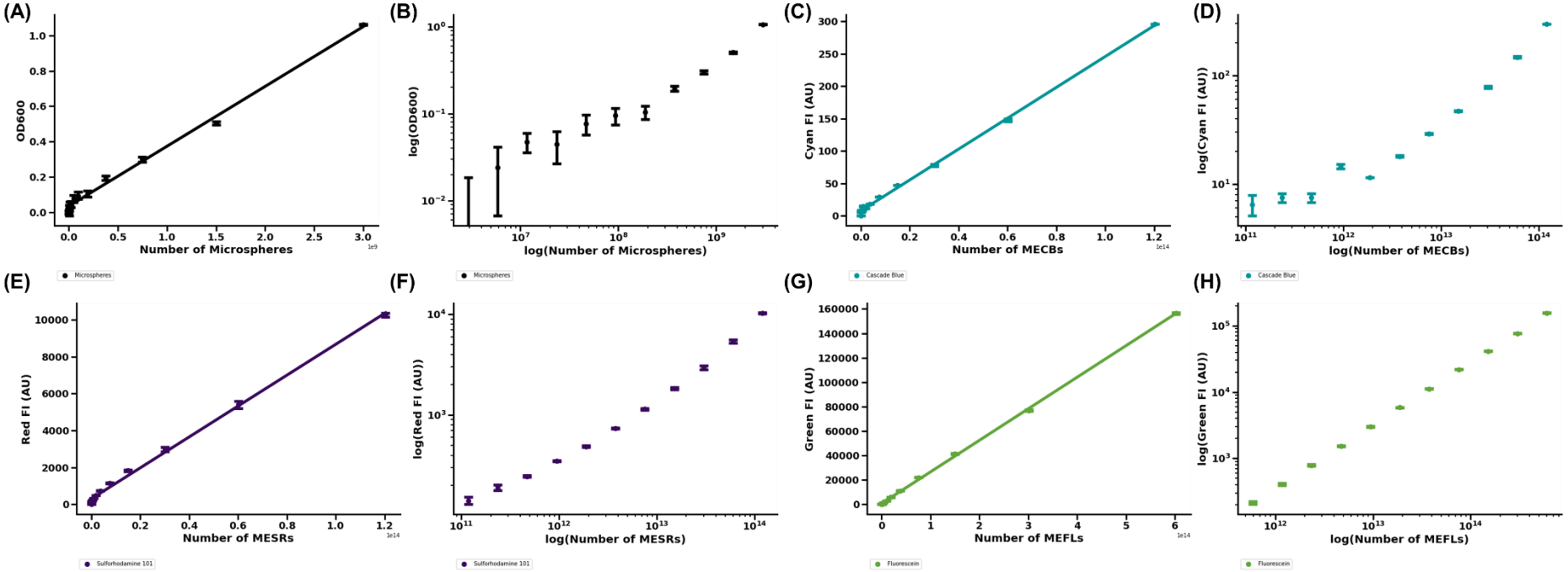


Figure 9.1. Standard Curves for Plate Reader Calibration: Shown are linear (A, C, E, H) and log-log (B, D, F, H) standard curves for plate reader calibrations. In each graph, number of calibrant molecules is plotted again raw plate reader data. In all cases, the 5 most concentrated dilutions were used to calculate calibration factors.

9.3. Dose Response Curve Plate Maps and Automation Protocols

Protocols can be found on GitHub: https://github.com/Brad0440/PhD-Thesis-Data

9.3.1. IPTG Detector Module Plate Map

Well	Content	Well	Content
B2	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep0	E2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.3125)-Rep2
B3	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep1	E3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.3125)-Rep3
B4	DH5alpha-ON(37.0)OD(0.1)-None-Rep0	E4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.15625)-Rep0
B5	DH5alpha-ON(37.0)OD(0.1)-None-Rep1	E5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.15625)-Rep1
B6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(20.0)-Rep0	E6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.15625)-Rep2
B7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(20.0)-Rep1	E7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.15625)-Rep3
B8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(20.0)-Rep2	E8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.078125)-Rep0
B9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(20.0)-Rep3	E9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.078125)-Rep1
B10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(10.0)-Rep0	E10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.078125)-Rep2
B11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(10.0)-Rep1	E11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.078125)-Rep3
C2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(10.0)-Rep2	F2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0390625)-Rep0
C3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(10.0)-Rep3	F3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0390625)-Rep1
C4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(5.0)-Rep0	F4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0390625)-Rep2
C5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(5.0)-Rep1	F5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0390625)-Rep3
C6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(5.0)-Rep2	F6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01953125)-Rep0

C7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(5.0)-Rep3	F7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01953125)-Rep1
C8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(2.5)-Rep0	F8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01953125)-Rep2
C9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(2.5)-Rep1	F9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01953125)-Rep3
C10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(2.5)-Rep2	F10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.009765625)-Rep0
C11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(2.5)-Rep3	F11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.009765625)-Rep1
D2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1.25)-Rep0	G2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.009765625)-Rep2
D3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1.25)-Rep1	G3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.009765625)-Rep3
D4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1.25)-Rep2	G4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0048828125)-Rep0
D5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1.25)-Rep3	G5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0048828125)-Rep1
D6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.625)-Rep0	G6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0048828125)-Rep2
D7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.625)-Rep1	G7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.0048828125)-Rep3
D8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.625)-Rep2	G8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep0
D9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.625)-Rep3	G9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep1
D10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.3125)-Rep0	G10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep2
D11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.3125)-Rep1	G11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep3

9.3.2. Default Processor Module Plate Map

Well	Content	Well	Content
B02	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None	E02	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.003200000000000001)

B03	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None	E03	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.003200000000000001)
B04	DH5alpha-ON(37.0)OD(0.1)-None	E04	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.0006400000000000003)
B05	DH5alpha-ON(37.0)OD(0.1)-None	E05	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.0006400000000000003)
B06	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(50.0)	E06	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.0006400000000000003)
B07	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(50.0)	E07	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.0006400000000000003)
B08	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(50.0)	E08	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.00012800000000000008)
B09	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(50.0)	E09	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.00012800000000000008)
B10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10.0)	E10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.00012800000000000008)
B11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10.0)	E11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.00012800000000000008)
C02	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10.0)	F02	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.5600000000000012e-05)
C03	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10.0)	F03	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.5600000000000012e-05)
C04	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0000000000000004)	F04	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.5600000000000012e-05)
C05	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0000000000000004)	F05	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.5600000000000012e-05)
C06	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0000000000000004)	F06	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5.120000000000003e-06)
C07	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0000000000000004)	F07	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5.120000000000003e-06)
C08	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.4000000000000001)	F08	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5.120000000000003e-06)
C09	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.4000000000000001)	F09	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5.120000000000003e-06)

C10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.4000000000000001)	F10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1.0240000000000007e-06)
C11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.4000000000000001)	F11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1.0240000000000007e-06)
D02	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.08000000000000002)	G02	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1.0240000000000007e-06)
D03	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.08000000000000002)	G03	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1.0240000000000007e-06)
D04	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.08000000000000002)	G04	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0480000000000011e-07)
D05	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.08000000000000002)	G05	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0480000000000011e-07)
D06	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.016000000000000004)	G06	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0480000000000011e-07)
D07	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.016000000000000004)	G07	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(2.0480000000000011e-07)
D08	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.016000000000000004)	G08	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(0.5uL)
D09	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.016000000000000004)	G09	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(0.5uL)
D10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.003200000000000001)	G10	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(0.5uL)
D11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.003200000000000001)	G11	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(0.5uL)

9.3.3. sfGFP Reporter Module Plate Map

Well	Content	Well	Content
B2	['J23100-B0034-mCherry-B0015-Temp(37)-AB(KAN 1)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 0	E2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.0032)-LB-Rep 2
B3	['J23100-B0034-mCherry-B0015-Temp(37)-AB(KAN 1)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 1	E3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.0032)-LB-Rep 3
B4	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 0	E4	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.00064)-LB-Rep 0

B5	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 1	E5	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.00064)-LB-Rep 1
B6	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(50.0)-LB-Rep 0	E6	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.00064)-LB-Rep 2
B7	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(50.0)-LB-Rep 1	E7	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.00064)-LB-Rep 3
B8	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(50.0)-LB-Rep 2	E8	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.000128)-LB-Rep 0
B9	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(50.0)-LB-Rep 3	E9	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.000128)-LB-Rep 1
B10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10.0)-LB-Rep 0	E10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.000128)-LB-Rep 2
B11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10.0)-LB-Rep 1	E11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.000128)-LB-Rep 3
C2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10.0)-LB-Rep 2	F2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.56e-05)-LB-Rep 0
C3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10.0)-LB-Rep 3	F3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.56e-05)-LB-Rep 1
C4	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.0)-LB-Rep 0	F4	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.56e-05)-LB-Rep 2
C5	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.0)-LB-Rep 1	F5	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.56e-05)-LB-Rep 3
C6	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.0)-LB-Rep 2	F6	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(5.12e-06)-LB-Rep 0
C7	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.0)-LB-Rep 3	F7	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(5.12e-06)-LB-Rep 1
C8	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.4)-LB-Rep 0	F8	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(5.12e-06)-LB-Rep 2
C9	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.4)-LB-Rep 1	F9	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(5.12e-06)-LB-Rep 3
C10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.4)-LB-Rep 2	F10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(1.024e-06)-LB-Rep 0
C11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.4)-LB-Rep 3	F11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(1.024e-06)-LB-Rep 1

D2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.08)-LB-Rep 0	G2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(1.024e-06)-LB-Rep 2
D3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.08)-LB-Rep 1	G3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(1.024e-06)-LB-Rep 3
D4	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.08)-LB-Rep 2	G4	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.048e-07)-LB-Rep 0
D5	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.08)-LB-Rep 3	G5	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.048e-07)-LB-Rep 1
D6	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.016)-LB-Rep 0	G6	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.048e-07)-LB-Rep 2
D7	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.016)-LB-Rep 1	G7	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(2.048e-07)-LB-Rep 3
D8	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.016)-LB-Rep 2	G8	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(0.5 uL)-LB-Rep 0
D9	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.016)-LB-Rep 3	G9	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(0.5 uL)-LB-Rep 1
D10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.0032)-LB-Rep 0	G10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(0.5 uL)-LB-Rep 2
D11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(0.0032)-LB-Rep 1	G11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(0.5 uL)-LB-Rep 3

9.4. Cross Talk Plate Maps and Automation Protocols

Protocols can be found on GitHub: https://github.com/Brad0440/PhD-Thesis-Data

9.4.1. IPTG Detector Module Plate Map

Well	Content	Well	Content
B2	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep0	E2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(1)-Rep2
B3	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep1	E3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(1)-Rep3
B4	DH5alpha-ON(37.0)OD(0.1)-None-Rep0	E4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep0
B5	DH5alpha-ON(37.0)OD(0.1)-None-Rep1	E5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep1

B6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1)-Rep0	E6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep2
B7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1)-Rep1	E7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep3
B8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1)-Rep2	E8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(10)-Rep0
B9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(1)-Rep3	E9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(10)-Rep1
B10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.5)-Rep0	E10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(10)-Rep2
B11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.5)-Rep1	E11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(10)-Rep3
C2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.5)-Rep2	F2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(5)-Rep0
C3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.5)-Rep3	F3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(5)-Rep1
C4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.1)-Rep0	F4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(5)-Rep2
C5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.1)-Rep1	F5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(5)-Rep3
C6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.1)-Rep2	F6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(1)-Rep0
C7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.1)-Rep3	F7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(1)-Rep1
C8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01)-Rep0	F8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(1)-Rep2
C9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01)-Rep1	F9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(1)-Rep3
C10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01)-Rep2	F10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep0
C11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-IPTG(0.01)-Rep3	F11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep1
D2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(10)-Rep0	G2	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep2
D3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(10)-Rep1	G3	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep3
D4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(10)-Rep2	G4	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep0
D5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(10)-Rep3	G5	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep1

D6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(5)-Rep0	G6	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep2
D7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(5)-Rep1	G7	IPTG Detector + eCFP-ON(37.0)OD(0.1)-Water(1.0uL)-Rep3
D8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(5)-Rep2	G8	IPTG Detector + eCFP-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep0
D9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(5)-Rep3	G9	IPTG Detector + eCFP-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep1
D10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(1)-Rep0	G10	IPTG Detector + eCFP-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep2
D11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-C12-HSL(1)-Rep1	G11	IPTG Detector + eCFP-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep3

9.4.2. Default Processor Module Plate Map

Well	Content	Well	Content
B2	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep0	E2	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.1)-Rep2
B3	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep1	E3	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.1)-Rep3
B4	DH5alpha-ON(37.0)OD(0.1)-None-Rep0	E4	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.01)-Rep0
B5	DH5alpha-ON(37.0)OD(0.1)-None-Rep1	E5	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.01)-Rep1
B6	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10)-Rep0	E6	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.01)-Rep2
B7	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10)-Rep1	E7	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.01)-Rep3
B8	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10)-Rep2	E8	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(10)-Rep0
B9	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(10)-Rep3	E9	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(10)-Rep1
B10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5)-Rep0	E10	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(10)-Rep2
B11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5)-Rep1	E11	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(10)-Rep3
C2	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5)-Rep2	F2	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(5)-Rep0
C3	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(5)-Rep3	F3	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(5)-Rep1

C4	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1)-Rep0	F4	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(5)-Rep2
C5	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1)-Rep1	F5	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(5)-Rep3
C6	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1)-Rep2	F6	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(1)-Rep0
C7	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(1)-Rep3	F7	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(1)-Rep1
C8	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep0	F8	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(1)-Rep2
C9	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep1	F9	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(1)-Rep3
C10	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep2	F10	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep0
C11	Default Processor + mCherry-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep3	F11	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep1
D2	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(1)-Rep0	G2	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep2
D3	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(1)-Rep1	G3	Default Processor + mCherry-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep3
D4	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(1)-Rep2	G4	Default Processor + mCherry-ON(37.0)OD(0.1)-Water(1.0uL)-Rep0
D5	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(1)-Rep3	G5	Default Processor + mCherry-ON(37.0)OD(0.1)-Water(1.0uL)-Rep1
D6	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.5)-Rep0	G6	Default Processor + mCherry-ON(37.0)OD(0.1)-Water(1.0uL)-Rep2
D7	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.5)-Rep1	G7	Default Processor + mCherry-ON(37.0)OD(0.1)-Water(1.0uL)-Rep3
D8	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.5)-Rep2	G8	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep0
D9	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.5)-Rep3	G9	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep1
D10	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.1)-Rep0	G10	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep2
D11	Default Processor + mCherry-ON(37.0)OD(0.1)-IPTG(0.1)-Rep1	G11	Default Processor + mCherry-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep3

9.4.3. sfGFP Reporter Module Plate Map

Well	Content	Well	Content
B2	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep0	E2	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.1)-Rep2
B3	J23100-B0034-mCherry-B0015-ON(37.0)OD(0.1)-None-Rep1	E3	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.1)-Rep3
B4	DH5alpha-ON(37.0)OD(0.1)-None-Rep0	E4	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.01)-Rep0
B5	DH5alpha-ON(37.0)OD(0.1)-None-Rep1	E5	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.01)-Rep1
B6	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(10)-Rep0	E6	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.01)-Rep2
B7	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(10)-Rep1	E7	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.01)-Rep3
B8	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(10)-Rep2	E8	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(10)-Rep0
B9	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(10)-Rep3	E9	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(10)-Rep1
B10	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(5)-Rep0	E10	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(10)-Rep2
B11	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(5)-Rep1	E11	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(10)-Rep3
C2	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(5)-Rep2	F2	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(5)-Rep0
C3	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(5)-Rep3	F3	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(5)-Rep1
C4	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(1)-Rep0	F4	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(5)-Rep2
C5	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(1)-Rep1	F5	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(5)-Rep3
C6	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(1)-Rep2	F6	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(1)-Rep0
C7	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(1)-Rep3	F7	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(1)-Rep1
C8	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep0	F8	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(1)-Rep2
C9	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep1	F9	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(1)-Rep3
C10	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep2	F10	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep0
C11	sfGFP Reporter-ON(37.0)OD(0.1)-C4-HSL(0.1)-Rep3	F11	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep1
D2	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(1)-Rep0	G2	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep2
D3	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(1)-Rep1	G3	sfGFP Reporter-ON(37.0)OD(0.1)-C12-HSL(0.1)-Rep3
D4	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(1)-Rep2	G4	sfGFP Reporter-ON(37.0)OD(0.1)-Water(1.0uL)-Rep0
D5	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(1)-Rep3	G5	sfGFP Reporter-ON(37.0)OD(0.1)-Water(1.0uL)-Rep1

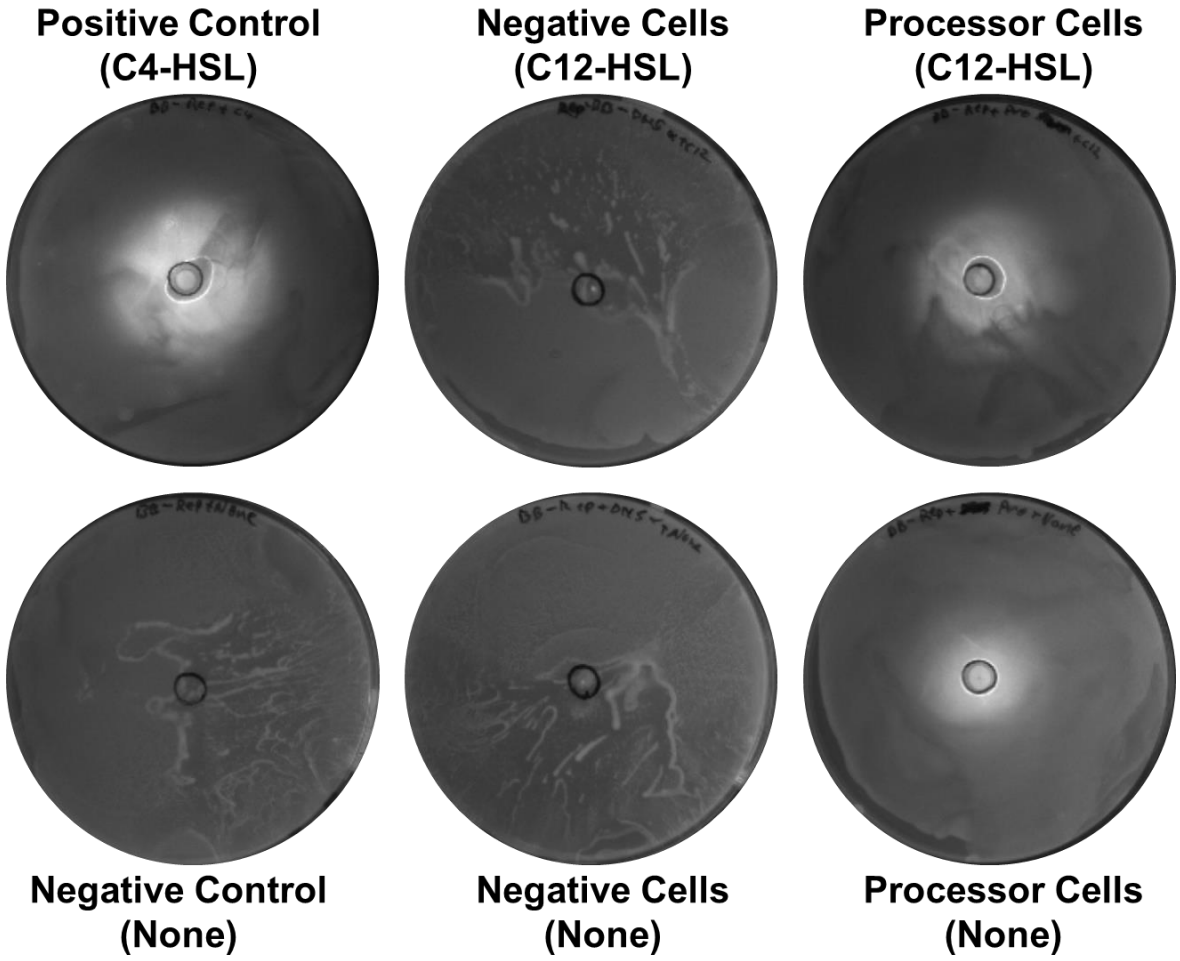


Figure 9.2. Unprocessed images of those shown in figure 5.13

D6	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.5)-Rep0	G6	sfGFP Reporter-ON(37.0)OD(0.1)-Water(1.0uL)-Rep2
D7	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.5)-Rep1	G7	sfGFP Reporter-ON(37.0)OD(0.1)-Water(1.0uL)-Rep3
D8	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.5)-Rep2	G8	sfGFP Reporter-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep0
D9	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.5)-Rep3	G9	sfGFP Reporter-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep1
D10	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.1)-Rep0	G10	sfGFP Reporter-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep2
D11	sfGFP Reporter-ON(37.0)OD(0.1)-IPTG(0.1)-Rep1	G11	sfGFP Reporter-ON(37.0)OD(0.1)-DMSO(1.0uL)-Rep3

9.5. Homoserine-lactone synthesis validation agar plates

9.6. Noise Propagation Plate Maps and Automation Protocols

Protocols can be found on GitHub: https://github.com/Brad0440/PhD-Thesis-Data

Well	Content	Well	Content
B2	['J23100-B0034-mCherry-B0015-Temp(37)-AB(KAN 1)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 0	E2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
B3	['J23100-B0034-mCherry-B0015-Temp(37)-AB(KAN 1)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 1	E3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
B4	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 0	E4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
B5	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 1	E5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
B6	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	E6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
B7	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	E7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
B8	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2	E8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
B9	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3	E9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
B10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	E10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'Default Processor + mCherry-

			Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
B11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	E11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
C2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2	F2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
C3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3	F3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
C4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	F4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
C5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	F5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
C6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2	F6	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
C7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3	F7	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
C8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	F8	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
C9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	F9	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-

			Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
C10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2	F10	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
C11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3	F11	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
D2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	G2	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
D3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	G3	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
D4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2	G4	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
D5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3	G5	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
D6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	G6	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
D7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	G7	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3
D8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM	G8	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-

	1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2		Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0
D9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(1)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3	G9	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1
D10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 0	G10	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 2
D11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 1	G11	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(50)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-None(0 uL)-LB-Rep 3

9.7. Initial Biosensor Testing Plate Maps and Automation Protocols

Protocols can be found on GitHub: https://github.com/Brad0440/PhD-Thesis-Data

Well	Content	Well	Content
B2	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 0	E2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.01)-LB-Rep 2
B3	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 1	E3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.01)-LB-Rep 3
B4	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 2	E4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 0
B5	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 3	E5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 1
B6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-	E6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-

	Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 0		Media(LB)Vol(10)']-IPTG(1)-LB-Rep 2
B7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 1	E7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 3
B8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 2	E8	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C12-HSL(10)-LB-Rep 0
B9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 3	E9	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C12-HSL(10)-LB-Rep 1
B10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(15)-LB-Rep 0	E10	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C12-HSL(10)-LB-Rep 2
B11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(15)-LB-Rep 1	E11	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C12-HSL(10)-LB-Rep 3
C2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(15)-LB-Rep 2	F2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10)-LB-Rep 0

C3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(15)-LB-Rep 3	F3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10)-LB-Rep 1
C4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(10)-LB-Rep 0	F4	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10)-LB-Rep 2
C5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(10)-LB-Rep 1	F5	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-C4-HSL(10)-LB-Rep 3
C6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(10)-LB-Rep 2	F6	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 0
C7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(10)-LB-Rep 3	F7	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 1
C8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(5)-LB-Rep 0	F8	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 2
C9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-	F9	['Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 3

	Media(LB)Vol(10)']-IPTG(5)-LB-Rep 1		
C10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(5)-LB-Rep 2	F10	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 0
C11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(5)-LB-Rep 3	F11	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 1
D2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 0	G2	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 2
D3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 1	G3	['sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-DMSO(1.0 uL)-LB-Rep 3
D4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 2	G4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 0
D5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(1)-LB-Rep 3	G5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 1
D6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-	G6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default Processor + mCherry-Temp(37)-

	Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.1)-LB-Rep 0		AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 2
D7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.1)-LB-Rep 1	G7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 3
D8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.1)-LB-Rep 2	G8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 0
D9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.1)-LB-Rep 3	G9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 1
D10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.01)-LB-Rep 0	G10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 2
D11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(0.01)-LB-Rep 1	G11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 3

9.8. Cell Ratio Testing Plate Maps and Automation Protocols

Protocols can be found on GitHub: https://github.com/Brad0440/PhD-Thesis-Data

Well	Content	Well	Content
B2	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 0	E2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default

			Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 0
B3	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 1	E3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 1
B4	['DH5alpha-Temp(37)-AB(None None)-Media(LB)Vol(10.0)']-None(0 uL)-LB-Rep 2	E4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)']-Water(1.0 uL)-LB-Rep 2
B5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 0	E5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(3.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(20.0)']-Water(1.0 uL)-LB-Rep 0
B6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 1	E6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(3.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(20.0)']-Water(1.0 uL)-LB-Rep 1
B7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 2	E7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(3.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(20.0)']-Water(1.0 uL)-LB-Rep 2
B8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(3)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(20)']-IPTG(20)-LB-Rep 0	E8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(5.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(15.0)']-Water(1.0 uL)-LB-Rep 0

B9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(3)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(20)']-IPTG(20)-LB-Rep 1	E9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(5.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(15.0)']-Water(1.0 uL)-LB-Rep 1
B10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(3)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(20)']-IPTG(20)-LB-Rep 2	E10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(5.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(15.0)']-Water(1.0 uL)-LB-Rep 2
B11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(15)']-IPTG(20)-LB-Rep 0	E11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(6.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(13.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(11.0)']-Water(1.0 uL)-LB-Rep 0
C2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(15)']-IPTG(20)-LB-Rep 1	F2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(6.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(13.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(11.0)']-Water(1.0 uL)-LB-Rep 1
C3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(5)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(15)']-IPTG(20)-LB-Rep 2	F3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(6.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(13.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(11.0)']-Water(1.0 uL)-LB-Rep 2
C4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(6)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(13)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(11)']-IPTG(20)-LB-Rep 0	F4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(2.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(24.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4.0)']-Water(1.0 uL)-LB-Rep 0
C5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(6)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(13)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-	F5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(2.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(24.0)', 'sfGFP Reporter-Temp(37)-

	Media(LB)Vol(11)']-IPTG(20)-LB-Rep 1		AB(CAM 1)-Media(LB)Vol(4.0)']-Water(1.0 uL)-LB-Rep 1
C6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(6)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(13)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(11)']-IPTG(20)-LB-Rep 2	F6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(2.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(24.0)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4.0)']-Water(1.0 uL)-LB-Rep 2
C7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(2)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(24)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)']-IPTG(20)-LB-Rep 0	F7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(11.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(18.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)']-Water(1.0 uL)-LB-Rep 0
C8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(2)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(24)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)']-IPTG(20)-LB-Rep 1	F8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(11.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(18.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)']-Water(1.0 uL)-LB-Rep 1
C9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(2)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(24)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)']-IPTG(20)-LB-Rep 2	F9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(11.0)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(18.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)']-Water(1.0 uL)-LB-Rep 2
C10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(11)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(18.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)']-IPTG(20)-LB-Rep 0		
C11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(11)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(18.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)']-IPTG(20)-LB-Rep 1		
D2	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(11)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-		

	Media(LB)Vol(18.5)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(0.5)']-IPTG(20)-LB-Rep 2	
D3	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(19)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)']-IPTG(20)-LB-Rep 0	
D4	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(19)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)']-IPTG(20)-LB-Rep 1	
D5	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(19)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)']-IPTG(20)-LB-Rep 2	
D6	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(18)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(2)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 0	
D7	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(18)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(2)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 1	
D8	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(18)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(2)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(10)']-IPTG(20)-LB-Rep 2	
D9	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)',	

	'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(19)']-IPTG(20)-LB-Rep 0	
D10	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(19)']-IPTG(20)-LB-Rep 1	
D11	['IPTG Detector + eCFP-Temp(37)-AB(CAM 1)-Media(LB)Vol(7)', 'Default Processor + mCherry-Temp(37)-AB(CAM 1)-Media(LB)Vol(4)', 'sfGFP Reporter-Temp(37)-AB(CAM 1)-Media(LB)Vol(19)']-IPTG(20)-LB-Rep 2	

9.9. Main Effects Screening Designs

Shown below are the main effects screening designs for each module type. Runs highlighted in blue represent the augmented runs. Temperatures are in ^{o}C and media compositions in g/L. Runs were performed as described in section 2.7.14.

9.9.1. IPTG Detector Module

	ON_Temp	Run_Temp	OD600	Tryptone	NaCl	Yeast Extract
1	37	25	0.324	0	0	0
2	30	37	0.37	20	20	10
3	30	37	0.169	20	0	0
4	37	37	0.392	0	20	10
5	30	30	0.311	10	10	5
6	30	25	0.179	0	0	10
7	30	25	0.367	0	20	0
8	37	25	0.139	20	20	0
9	30	25	0.336	20	0	10
10	37	37	0.338	20	0	0
11	30	37	0.184	0	20	0
12	37	25	0.153	20	20	10
13	37	30	0.259	10	10	5
14	30	25	0.955	0	20	0
15	37	37	1.036	20	0	0
16	37	37	1.184	0	20	10
17	37	25	0.949	0	0	0
18	30	25	0.788	20	0	10
19	30	37	0.797	20	20	10

9.9.2. Default Processor Module

	ON_Temp	Run_Temp	OD600	Tryptone	NaCl	Yeast Extract
1	25	25	0.149	0	0	0
2	25	37	0.154	20	0	10
3	37	25	0.611	0	0	0
4	25	25	0.368	20	20	0
5	25	37	0.15	0	20	0
6	37	37	0.499	20	0	0
7	37	37	1.864	20	0	0
8	37	37	0.485	0	20	10
9	25	25	0.147	20	0	10
10	37	25	2.024	0	0	10
11	37	25	0.551	20	20	10
12	30	30	0.848	10	10	5
13	37	25	0.559	20	20	0
14	25	37	1.265	0	20	10
15	37	37	0.458	20	0	10
16	37	25	0.537	0	20	10
17	37	37	0.469	0	0	0
18	25	37	1.042	20	20	0

9.9.3. sfGFP Reporter Module

	ON_Temp	Run_Temp	OD600	Tryptone	NaCl	Yeast Extract
1	25	25	0.143	0	0	0
2	25	37	0.154	20	0	10
3	37	25	0.537	0	0	0
4	25	25	0.409	20	20	0
5	25	37	0.138	0	20	0
6	37	37	0.504	20	0	0
7	37	37	1.928	20	0	0
8	37	37	0.475	0	20	10
9	25	25	0.145	20	0	10
10	37	25	2.002	0	0	10
11	37	25	0.486	20	20	10
12	30	30	0.841	10	10	5
13	25	37	1.247	0	20	10
14	37	37	0.493	0	0	10
15	37	37	0.464	20	20	0
16	25	37	1.269	0	20	0
17	25	25	1.111	20	20	10
18	37	25	0.511	0	20	0

9.10. IPTG Dose Response Outliers

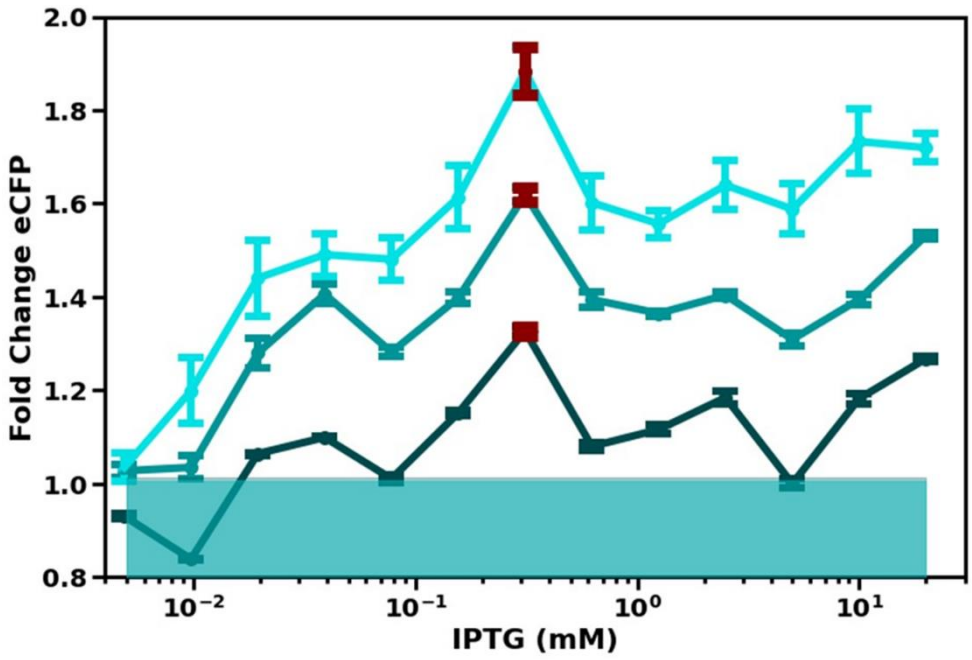


Figure 9.3. Shown here is the dose response curve from figure 5.2 (C). Red points show the excluded outlier data.

9.11. Luciferase Characterisation Plate Map and Automation Protocol

Protocols can be found on GitHub: https://github.com/Brad0440/PhD-Thesis-Data

Well	Content	Well	Content
B02	pBad Lux-ON(37.0)OD(0.1)-None	E02	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.625)
B03	pBad Lux-ON(37.0)OD(0.1)-None	E03	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.625)
B04	pBad Lux-ON(37.0)OD(0.1)-None	E04	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.3125)
B05	pBad Lux-ON(37.0)OD(0.1)-None	E05	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.3125)
B06	DH5alpha-ON(37.0)OD(0.1)-None	E06	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.3125)
B07	DH5alpha-ON(37.0)OD(0.1)-None	E07	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.3125)
B08	DH5alpha-ON(37.0)OD(0.1)-None	E08	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.15625)
B09	DH5alpha-ON(37.0)OD(0.1)-None	E09	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.15625)
B10	pBad Lux-ON(37.0)OD(0.1)-Arabinose(20)	E10	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.15625)
B11	pBad Lux-ON(37.0)OD(0.1)-Arabinose(20)	E11	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.15625)
C02	pBad Lux-ON(37.0)OD(0.1)-Arabinose(20)	F02	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.078125)
C03	pBad Lux-ON(37.0)OD(0.1)-Arabinose(20)	F03	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.078125)
C04	pBad Lux-ON(37.0)OD(0.1)-Arabinose(10)	F04	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.078125)
C05	pBad Lux-ON(37.0)OD(0.1)-Arabinose(10)	F05	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.078125)
C06	pBad Lux-ON(37.0)OD(0.1)-Arabinose(10)	F06	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.0390625)
C07	pBad Lux-ON(37.0)OD(0.1)-Arabinose(10)	F07	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.0390625)
C08	pBad Lux-ON(37.0)OD(0.1)-Arabinose(5)	F08	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.0390625)
C09	pBad Lux-ON(37.0)OD(0.1)-Arabinose(5)	F09	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.0390625)
C10	pBad Lux-ON(37.0)OD(0.1)-Arabinose(5)	F10	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.01953125)
C11	pBad Lux-ON(37.0)OD(0.1)-Arabinose(5)	F11	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.01953125)
D02	pBad Lux-ON(37.0)OD(0.1)-Arabinose(2.5)	G02	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.01953125)
D03	pBad Lux-ON(37.0)OD(0.1)-Arabinose(2.5)	G03	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.01953125)
D04	pBad Lux-ON(37.0)OD(0.1)-Arabinose(2.5)	G04	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.009765625)

D05	pBad Lux-ON(37.0)OD(0.1)-Arabinose(2.5)	G05	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.009765625)
D06	pBad Lux-ON(37.0)OD(0.1)-Arabinose(1.25)	G06	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.009765625)
D07	pBad Lux-ON(37.0)OD(0.1)-Arabinose(1.25)	G07	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.009765625)
D08	pBad Lux-ON(37.0)OD(0.1)-Arabinose(1.25)	G08	pBad Lux-ON(37.0)OD(0.1)-Water(2.0uL)
D09	pBad Lux-ON(37.0)OD(0.1)-Arabinose(1.25)	G09	pBad Lux-ON(37.0)OD(0.1)-Water(2.0uL)
D10	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.625)	G10	pBad Lux-ON(37.0)OD(0.1)-Water(2.0uL)
D11	pBad Lux-ON(37.0)OD(0.1)-Arabinose(0.625)	G11	pBad Lux-ON(37.0)OD(0.1)-Water(2.0uL)

Chapter 10. References

[1] L. Chantranupong, R. L. Wolfson, and D. M. Sabatini, 'Nutrient-Sensing Mechanisms across Evolution', *Cell*, vol. 161, no. 1, pp. 67–83, Mar. 2015, doi: 10.1016/J.CELL.2015.02.041.

[2] S. Otto, E. P. Bruni, H. Harms, and L. Y. Wick, 'Catch me if you can: dispersal and foraging of Bdellovibrio bacteriovorus 109J along mycelia', *The ISME Journal 2017 11:2*, vol. 11, no. 2, pp. 386–393, Nov. 2016, doi: 10.1038/ismej.2016.135.

[3] M. Waso, S. Khan, and W. Khan, 'Assessment of predatory bacteria and prey interactions using culture-based methods and EMA-qPCR', *Microbiol Res*, vol. 228, p. 126305, Nov. 2019, doi: 10.1016/J.MICRES.2019.126305.

[4] G. H. Wadhams and J. P. Armitage, 'Making sense of it all: bacterial chemotaxis', *Nature Reviews Molecular Cell Biology 2004 5:12*, vol. 5, no. 12, pp. 1024–1037, Dec. 2004, doi: 10.1038/nrm1524.

[5] K. Shimizu, 'Metabolic Regulation of a Bacterial Cell System with Emphasis on Escherichia coli Metabolism', *ISRN Biochem*, vol. 2013, pp. 1–47, Feb. 2013, doi: 10.1155/2013/645983.

[6] H. Wedler and R. Wambutt, 'A temperature-sensitive lambda cI repressor functions on a modified operator in yeast cells by masking the TATA element', *Mol Gen Genet*, vol. 248, no. 4, pp. 499–505, Aug. 1995, doi: 10.1007/BF02191651.

[7] S. Gupta and V. Kakkar, 'Development of Environmental Biosensors for Detection, Monitoring, and Assessment', *Nanotechnology in the Life Sciences*, pp. 107–125, 2020, doi: 10.1007/978-3-030-34544-0_7/TABLES/1.

[8] A. Haleem, M. Javaid, R. P. Singh, R. Suman, and S. Rab, 'Biosensors applications in medical field: A brief review', *Sensors International*, vol. 2, p. 100100, Jan. 2021, doi: 10.1016/J.SINTL.2021.100100.

[9] I. del Valle, E. M. Fulk, P. Kalvapalle, J. J. Silberg, C. A. Masiello, and L. B. Stadler, 'Translating New Synthetic Biology Advances for Biosensing Into the Earth and Environmental Sciences', *Front Microbiol*, vol. 11, p. 3513, Feb. 2021, doi: 10.3389/FMICB.2020.618373/BIBTEX.

[10] V. . Naresh and N. Lee, 'A Review on Biosensors and Recent Development of Nanostructured Materials-Enabled Biosensors', *A Review on Biosensors and*

Recent Development of Nanostructured Materials-Enabled Biosensors. Sensors, vol. 21, p. 1109, 2021, doi: 10.3390/s21041109.

[11] Y. Li *et al.*, 'Taccalonolides: Structure, semi-synthesis, and biological activity', *Front Pharmacol*, vol. 13, p. 2866, Aug. 2022, doi: 10.3389/FPHAR.2022.968061/BIBTEX.

[12] L. Xiang *et al.*, 'Biodegradation of aromatic pollutants meets synthetic biology', *Synth Syst Biotechnol*, vol. 6, no. 3, pp. 153–162, Sep. 2021, doi: 10.1016/J.SYNBIO.2021.06.001.

[13] D. Ucar, Y. Zhang, and I. Angelidaki, 'An overview of electron acceptors in microbial fuel cells', *Front Microbiol*, vol. 8, no. APR, p. 643, Apr. 2017, doi: 10.3389/FMICB.2017.00643/BIBTEX.

[14] B. F. Sun, 'Total synthesis of natural and pharmaceutical products powered by organocatalytic reactions', *Tetrahedron Lett*, vol. 56, no. 17, pp. 2133–2140, Apr. 2015, doi: 10.1016/J.TETLET.2015.03.046.

[15] K. C. Nicolaou, D. Vourloumis, N. Winssinger, and P. S. Baran, 'The Art and Science of Total Synthesis at the Dawn of the Twenty-First Century**', doi: 10.1002/(SICI)1521-3773(20000103)39:1.

[16] R. A. Kjonaas, P. E. Williams, D. A. Counce, and L. R. Crawley, 'Synthesis of ibuprofen in the introductory organic laboratory', *J Chem Educ*, vol. 88, no. 6, pp. 825–828, Jun. 2011, doi: 10.1021/ED100892P/ASSET/IMAGES/MEDIUM/ED-2010-00892P_0001.GIF.

[17] C. Q. Li, H. M. Lei, Q. Y. Hu, G. H. Li, and P. J. Zhao, 'Recent Advances in the Synthetic Biology of Natural Drugs', *Front Bioeng Biotechnol*, vol. 9, p. 640, Jul. 2021, doi: 10.3389/FBIOE.2021.691152/BIBTEX.

[18] S. K. Yadav and P. Guleria, 'Steviol glycosides from Stevia: biosynthesis pathway review and their application in foods and medicine', *Crit Rev Food Sci Nutr*, vol. 52, no. 11, pp. 988–998, Nov. 2012, doi: 10.1080/10408398.2010.519447.

[19] G. E. Wen *et al.*, 'The first total synthesis of rebaudioside R', *Org Biomol Chem*, vol. 18, no. 1, pp. 108–126, Dec. 2019, doi: 10.1039/C9OB02422K.

[20] E. Baloglu and D. G. I. Kingston, 'A New Semisynthesis of Paclitaxel from Baccatin III', *J Nat Prod*, vol. 62, no. 7, pp. 1068–1071, Jul. 1999, doi: 10.1021/NP990040K.

[21] C. Li, R. Zhang, J. Wang, L. M. Wilson, and Y. Yan, ‘Protein engineering for improving and diversifying natural products biosynthesis’, *Trends Biotechnol*, vol. 38, no. 7, p. 729, Jul. 2020, doi: 10.1016/J.TIBTECH.2019.12.008.

[22] J. M. Woodley, ‘Protein engineering of enzymes for process applications’, *Curr Opin Chem Biol*, vol. 17, no. 2, pp. 310–316, Apr. 2013, doi: 10.1016/J.CBPA.2013.03.017.

[23] R. J. Marcheschi, L. S. Gronenberg, and J. C. Liao, ‘Protein engineering for metabolic engineering: current and next-generation tools’, *Biotechnol J*, vol. 8, no. 5, p. 545, May 2013, doi: 10.1002/BIOT.201200371.

[24] B. van den Burg, G. Vriend, O. R. Veltman, G. Venema, and V. G. H. Eijsink, ‘Engineering an enzyme to resist boiling’, *Proc Natl Acad Sci U S A*, vol. 95, no. 5, pp. 2056–2060, Mar. 1998, doi: 10.1073/PNAS.95.5.2056/ASSET/7E450772-2797-4C18-B458-F333D25D9FBA/ASSETS/GRAPHIC/PQ0484305004.JPEG.

[25] M. Goldsmith and D. S. Tawfik, ‘Enzyme engineering: reaching the maximal catalytic efficiency peak’, *Curr Opin Struct Biol*, vol. 47, pp. 140–150, Dec. 2017, doi: 10.1016/J.SBI.2017.09.002.

[26] R. Chen, ‘Enzyme engineering: rational redesign versus directed evolution’, *Trends Biotechnol*, vol. 19, no. 1, pp. 13–14, Jan. 2001, doi: 10.1016/S0167-7799(00)01522-5.

[27] H. Chen, L. Ma, H. Dai, Y. Fu, H. Wang, and Y. Zhang, ‘Advances in Rational Protein Engineering toward Functional Architectures and Their Applications in Food Science’, *J Agric Food Chem*, vol. 70, no. 15, pp. 4522–4533, Apr. 2022, doi: 10.1021/ACS.JAFC.2C00232/ASSET/IMAGES/MEDIUM/JF2C00232_0009.GIF.

[28] S. R. Nirantar, ‘Directed Evolution Methods for Enzyme Engineering’, *Molecules*, vol. 26, no. 18, Sep. 2021, doi: 10.3390/MOLECULES26185599.

[29] A. Kawamura and T. Miyata, ‘Biosensors’, *Biomaterials Nanoarchitectonics*, pp. 157–176, Jan. 2016, doi: 10.1016/B978-0-323-37127-8.00010-8.

[30] N. Bhalla, P. Jolly, N. Formisano, and P. Estrela, ‘Introduction to biosensors’, *Essays Biochem*, vol. 60, no. 1, p. 1, Jun. 2016, doi: 10.1042/EBC20150001.

[31] G. Rocchitta *et al.*, ‘Enzyme Biosensors for Biomedical Applications: Strategies for Safeguarding Analytical Performances in Biological Fluids’, *Sensors 2016, Vol. 16, Page 780*, vol. 16, no. 6, p. 780, May 2016, doi: 10.3390/S16060780.

[32] A. Schreiber, R. Feldbrügge, G. Key, J. F. C. Glatz, and F. Spener, ‘An immunosensor based on disposable electrodes for rapid estimation of fatty acid-binding protein, an early marker of myocardial infarction’, *Biosens Bioelectron*, vol. 12, no. 11, pp. 1131–1137, Dec. 1997, doi: 10.1016/S0956-5663(97)00003-1.

[33] S. B. Murugaiyan, R. Ramasamy, N. Gopal, and V. Kuzhandaivelu, ‘Biosensors in clinical chemistry: An overview’, *Adv Biomed Res*, vol. 3, no. 1, p. 67, 2014, doi: 10.4103/2277-9175.125848.

[34] J. Wang, ‘Glucose Biosensors: 40 Years of Advances and Challenges’, doi: 10.1002/1521-4109.

[35] S. M. Khor, J. Choi, P. Won, and S. H. Ko, ‘Challenges and Strategies in Developing an Enzymatic Wearable Sweat Glucose Biosensor as a Practical Point-Of-Care Monitoring Tool for Type II Diabetes’, *Nanomaterials*, vol. 12, no. 2, Jan. 2022, doi: 10.3390/NANO12020221.

[36] F. Ricci and G. Palleschi, ‘Sensor and biosensor preparation, optimisation and applications of Prussian Blue modified electrodes’, *Biosens Bioelectron*, vol. 21, no. 3, pp. 389–407, Sep. 2005, doi: 10.1016/J.BIOS.2004.12.001.

[37] K. Hult and P. Berglund, ‘Enzyme promiscuity: mechanism and applications’, *Trends Biotechnol*, vol. 25, no. 5, pp. 231–238, May 2007, doi: 10.1016/J.TIBTECH.2007.03.002.

[38] R. B. Leveson-Gower, C. Mayer, and G. Roelfes, ‘The importance of catalytic promiscuity for enzyme design and evolution’, *Nature Reviews Chemistry 2019 3:12*, vol. 3, no. 12, pp. 687–705, Nov. 2019, doi: 10.1038/s41570-019-0143-x.

[39] J. Ang, E. Harris, B. J. Hussey, R. Kil, and D. R. McMillen, ‘Tuning response curves for synthetic biology’, *ACS Synth Biol*, vol. 2, no. 10, pp. 547–567, Oct. 2013, doi: 10.1021/SB4000564/ASSET/IMAGES/LARGE/SB-2013-000564_0009.JPEG.

[40] G. Liu, ‘Grand Challenges in Biosensors and Biomolecular Electronics’, *Front Bioeng Biotechnol*, vol. 9, Aug. 2021, doi: 10.3389/FBIOE.2021.707615.

[41] K. Miura, A. C. Orcutt, O. v. Muratova, L. H. Miller, A. Saul, and C. A. Long, ‘Development and Characterization of a Standardized ELISA Including a Reference Serum on Each Plate to Detect Antibodies Induced by Experimental Malaria Vaccines’, *Vaccine*, vol. 26, no. 2, p. 193, Jan. 2008, doi: 10.1016/J.VACCINE.2007.10.064.

[42] S. Sakamoto *et al.*, ‘Enzyme-linked immunosorbent assay for the quantitative/qualitative analysis of plant secondary metabolites’, *J Nat Med*, vol. 72, no. 1, pp. 32–42, Jan. 2018, doi: 10.1007/S11418-017-1144-Z/FIGURES/6.

[43] A. Thiha and F. Ibrahim, ‘A Colorimetric Enzyme-Linked Immunosorbent Assay (ELISA) Detection Platform for a Point-of-Care Dengue Detection System on a Lab-on-Compact-Disc’, *Sensors (Basel)*, vol. 15, no. 5, pp. 11431–11441, May 2015, doi: 10.3390/S150511431.

[44] S. Aydin, ‘A short history, principles, and types of ELISA, and our laboratory experience with peptide/protein analyses using ELISA’, *Peptides (N.Y.)*, vol. 72, pp. 4–15, Mar. 2015, doi: 10.1016/J.PEPTIDES.2015.04.012.

[45] P. Kralik and M. Ricchi, ‘A basic guide to real time PCR in microbial diagnostics: Definitions, parameters, and everything’, *Front Microbiol*, vol. 8, no. FEB, p. 108, Feb. 2017, doi: 10.3389/FMICB.2017.00108/BIBTEX.

[46] A. Demeke Teklemariam, M. Samaddar, M. G. Alharbi, R. R. Al-Hindi, and A. K. Bhunia, ‘Biosensor and molecular-based methods for the detection of human coronaviruses: A review’, *Mol Cell Probes*, vol. 54, p. 101662, Dec. 2020, doi: 10.1016/J.MCP.2020.101662.

[47] P. Kotrade, E. M. Sehr, E. Wischnitzki, and W. Brüggemann, ‘Comparative transcriptomics-based selection of suitable reference genes for normalization of RT-qPCR experiments in drought-stressed leaves of three European Quercus species’, *Tree Genet Genomes*, vol. 15, no. 3, pp. 1–12, Jun. 2019, doi: 10.1007/S11295-019-1347-4/TABLES/4.

[48] H. Zhang *et al.*, ‘Determination of Advantages and Limitations of qPCR Duplexing in a Single Fluorescent Channel’, *ACS Omega*, vol. 6, no. 34, pp. 22292–22300, Aug. 2021, doi: 10.1021/ACSOMEGA.1C02971/ASSET/IMAGES/MEDIUM/AO1C02971_M012.GIF.

[49] S. Bustin and J. Huggett, ‘qPCR primer design revisited’, *Biomol Detect Quantif*, vol. 14, p. 19, Dec. 2017, doi: 10.1016/J.BDQ.2017.11.001.

[50] M. Hicks, T. T. Bachmann, and B. Wang, ‘Synthetic biology enables programmable cell-based biosensors’, *ChemPhysChem*, vol. 21, no. 2, pp. 132–144, 2019, doi: 10.1002/cphc.201901191.

[51] J. K. Rogers, N. D. Taylor, and G. M. Church, ‘Biosensor-based engineering of biosynthetic pathways’, *Curr Opin Biotechnol*, vol. 42, pp. 84–91, Dec. 2016, doi: 10.1016/J.COPBIO.2016.03.005.

[52] B. K. Verma, A. A. Mannan, F. Zhang, and D. A. Oyarzún, ‘Trade-Offs in Biosensor Optimization for Dynamic Pathway Engineering’, *ACS Synth Biol*, vol. 11, no. 1, pp. 228–240, Jan. 2022, doi: 10.1021/ACSSYNBIO.1C00391/SUPPL_FILE/SB1C00391_SI_002.XLSX.

[53] M. Merkx, B. Smith, and M. Jewett, ‘Engineering Sensor Proteins’, *ACS Sens*, vol. 4, no. 12, pp. 3089–3091, Dec. 2019, doi: 10.1021/ACSSENSORS.9B02459.

[54] Y. J. Chen *et al.*, ‘Development of a highly sensitive enzyme-linked immunosorbent assay (ELISA) through use of poly-protein G-expressing cell-based microplates’, *Scientific Reports 2018 8:1*, vol. 8, no. 1, pp. 1–11, Dec. 2018, doi: 10.1038/s41598-018-36192-8.

[55] K. L. Garner, ‘Principles of synthetic biology’, *Essays Biochem*, vol. 65, no. 5, pp. 791–811, Nov. 2021, doi: 10.1042/EBC20200059.

[56] L. Katz, Y. Y. Chen, R. Gonzalez, T. C. Peterson, H. Zhao, and R. H. Baltz, ‘Synthetic biology advances and applications in the biotechnology industry: a perspective’, *J Ind Microbiol Biotechnol*, vol. 45, no. 7, pp. 449–461, Jul. 2018, doi: 10.1007/S10295-018-2056-Y.

[57] D. Endy, ‘Foundations for engineering biology’, *Nature 2005 438:7067*, vol. 438, no. 7067, pp. 449–453, Nov. 2005, doi: 10.1038/nature04342.

[58] A. Casini, M. Storch, G. S. Baldwin, and T. Ellis, ‘Bricks and blueprints: methods and standards for DNA assembly’, *Nature Reviews Molecular Cell Biology 2015 16:9*, vol. 16, no. 9, pp. 568–576, Jun. 2015, doi: 10.1038/nrm4014.

[59] M. Storch, A. Casini, B. Mackrow, T. Ellis, and G. S. Baldwin, ‘BASIC: A Simple and Accurate Modular DNA Assembly Method’, *Methods Mol Biol*, vol. 1472, pp. 79–91, 2017, doi: 10.1007/978-1-4939-6343-0_6.

[60] R. Chao, Y. Yuan, and H. Zhao, ‘Recent advances in DNA assembly technologies’, *FEMS Yeast Res*, vol. 15, no. 1, p. 1, Feb. 2015, doi: 10.1111/1567-1364.12171.

[61] S. Marillonnet and R. Grützner, ‘Synthetic DNA Assembly Using Golden Gate Cloning and the Hierarchical Modular Cloning Pipeline’, *Curr Protoc Mol Biol*, vol. 130, no. 1, p. e115, Mar. 2020, doi: 10.1002/CPMB.115.

[62] G. Mısırlı, B. Yang, K. James, and A. Wipat, ‘Virtual Parts Repository 2: Model-Driven Design of Genetic Regulatory Circuits’, *ACS Synth Biol*, vol. 10, no. 12, pp. 3304–3315, Dec. 2021, doi: 10.1021/ACSSYNBIO.1C00157/ASSET/IMAGES/MEDIUM/SB1C00157_0006.GIF.

[63] J. Beal *et al.*, ‘Multicolor plate reader fluorescence calibration’, *Synth Biol*, vol. 7, no. 1, pp. 1–9, Nov. 2022, doi: 10.1093/SYNBIO/YSAC010.

[64] J. A. McLaughlin *et al.*, ‘The Synthetic Biology Open Language (SBOL) Version 3: Simplified Data Exchange for Bioengineering’, *Front Bioeng Biotechnol*, vol. 8, p. 1009, Sep. 2020, doi: 10.3389/FBIOE.2020.01009/BIBTEX.

[65] C. Y. Baldwin and K. B. Clark, ‘Design Rules: The Power of Modularity’, *Design Rules*, Dec. 2000, doi: 10.7551/MITPRESS/2366.001.0001.

[66] V. Modrak and Z. Soltysova, ‘Development of the Modularity Measure for Assembly Process Structures’, *Math Probl Eng*, vol. 2021, 2021, doi: 10.1155/2021/4900748.

[67] J. Bonvoisin, F. Halstenberg, T. Buchert, and R. Stark, ‘A systematic literature review on modular product design’, *http://dx.doi.org/10.1080/09544828.2016.1166482*, vol. 27, no. 7, pp. 488–514, Jul. 2016, doi: 10.1080/09544828.2016.1166482.

[68] C. Y. Baldwin and K. B. Clark, ‘Modularity in the design of complex engineering systems’, *Underst Complex Syst*, vol. 2006, pp. 175–205, 2006, doi: 10.1007/3-540-32834-3_9/COVER.

[69] S. Garcia and C. T. Trinh, ‘Modular design: Implementing proven engineering principles in biotechnology’, *Biotechnol Adv*, vol. 37, no. 7, p. 107403, Nov. 2019, doi: 10.1016/J.BIOTECHADV.2019.06.002.

[70] P. Opgenorth *et al.*, ‘Lessons from Two Design-Build-Test-Learn Cycles of Dodecanol Production in Escherichia coli Aided by Machine Learning’, *ACS Synth Biol*, vol. 8, no. 6, pp. 1337–1351, Jun. 2019, doi: 10.1021/ACSSYNBIO.9B00020/SUPPL_FILE/SB9B00020_SI_005.XLSX.

[71] J. Beal and M. Rogers, ‘Levels of autonomy in synthetic biology engineering’, *Mol Syst Biol*, vol. 16, no. 12, p. e10019, Dec. 2020, doi: 10.15252/MSB.202010019.

[72] J. Tanevski, L. Todorovski, and S. Džeroski, ‘Process-based design of dynamical biological systems’, *Scientific Reports 2016 6:1*, vol. 6, no. 1, pp. 1–13, Sep. 2016, doi: 10.1038/srep34107.

[73] E. Appleton, C. Madsen, N. Roehner, and D. Densmore, ‘Design Automation in Synthetic Biology’, *Cold Spring Harb Perspect Biol*, vol. 9, no. 4, Apr. 2017, doi: 10.1101/CSHPERSPECT.A023978.

[74] J. T. MacDonald, C. Barnes, R. I. Kitney, P. S. Freemont, and G. B. v. Stan, ‘Computational design approaches and tools for synthetic biology’, *Integrative Biology*, vol. 3, no. 2, pp. 97–108, Feb. 2011, doi: 10.1039/C0IB00077A.

[75] T. S. Jones, S. M. D. Oliveira, C. J. Myers, C. A. Voigt, and D. Densmore, ‘Genetic circuit design automation with Cello 2.0’, *Nat Protoc*, vol. 17, no. 4, pp. 1097–1113, Apr. 2022, doi: 10.1038/S41596-021-00675-2.

[76] L. Watanabe *et al.*, ‘IBIOSIM 3: A Tool for Model-Based Genetic Circuit Design’, *ACS Synth Biol*, vol. 8, no. 7, pp. 1560–1563, Jul. 2019, doi: 10.1021/ACSSYNBIO.8B00078/ASSET/IMAGES/MEDIUM/SB-2018-000785_0002.GIF.

[77] J. Naylor *et al.*, ‘Simbiotics: A Multiscale Integrative Platform for 3D Modeling of Bacterial Populations’, *ACS Synth Biol*, vol. 6, no. 7, pp. 1194–1210, Jul. 2017, doi: 10.1021/ACSSYNBIO.6B00315/SUPPL_FILE/SB6B00315_SI_001.PDF.

[78] J. R. Porter and E. Batchelor, ‘Using Computational Modeling and Experimental Synthetic Perturbations to Probe Biological Circuits’, *Methods Mol Biol*, vol. 1244, p. 259, 2015, doi: 10.1007/978-1-4939-1878-2_12.

[79] K. Malcl *et al.*, ‘Standardization of Synthetic Biology Tools and Assembly Methods for Saccharomyces cerevisiae and Emerging Yeast Species’, *ACS Synth Biol*, vol. 11, no. 8, pp. 2527–2547, Aug. 2022, doi: 10.1021/ACSSYNBIO.1C00442/ASSET/IMAGES/LARGE/SB1C00442_0009.JPEG.

[80] J. E. Bird, J. Marles-Wright, and A. Giachino, ‘A User’s Guide to Golden Gate Cloning Methods and Standards’, *ACS Synth Biol*, vol. 2022, pp. 3551–3563, Nov. 2022, doi: 10.1021/ACSSYNBIO.2C00355/ASSET/IMAGES/LARGE/SB2C00355_0005.JPEG.

[81] V. de Lorenzo, N. Krasnogor, and M. Schmidt, ‘For the sake of the Bioeconomy: define what a Synthetic Biology Chassis is!’, *N Biotechnol*, vol. 60, pp. 44–51, Jan. 2021, doi: 10.1016/J.NBT.2020.08.004.

[82] B. L. Adams, ‘The Next Generation of Synthetic Biology Chassis: Moving Synthetic Biology from the Laboratory to the Field’, *ACS Synth Biol*, vol. 5, no. 12, pp. 1328–1330, Dec. 2016, doi: 10.1021/ACSSYNBIO.6B00256/ASSET/IMAGES/LARGE/SB-2016-002563_0001.JPEG.

[83] D. Arella, M. Dilucca, and A. Giansanti, ‘Codon usage bias and environmental adaptation in microbial organisms’, *Molecular Genetics and Genomics*, vol. 296, no. 3, p. 751, May 2021, doi: 10.1007/S00438-021-01771-4.

[84] Z. Lipinszki *et al.*, ‘Enhancing the Translational Capacity of E. coli by Resolving the Codon Bias’, *ACS Synth Biol*, vol. 7, no. 11, pp. 2656–2664, Nov. 2018, doi: 10.1021/ACSSYNBIO.8B00332/ASSET/IMAGES/LARGE/SB-2018-003328_0006.JPEG.

[85] V. Chandrasekaran *et al.*, ‘Mechanism of ribosome stalling during translation of a poly(A) tail’, *Nature Structural & Molecular Biology 2019 26:12*, vol. 26, no. 12, pp. 1132–1140, Nov. 2019, doi: 10.1038/s41594-019-0331-x.

[86] C. Elena, P. Ravasi, M. E. Castelli, S. Peirú, and H. G. Menzella, ‘Expression of codon optimized genes in microbial systems: Current industrial applications and perspectives’, *Front Microbiol*, vol. 5, no. FEB, p. 21, 2014, doi: 10.3389/FMICB.2014.00021/BIBTEX.

[87] D. J. Skelton, L. E. Eland, M. Sim, M. A. White, R. J. Davenport, and A. Wipat, ‘Codon optimisation for maximising gene expression in multiple species and microbial consortia’, doi: 10.1101/2020.06.30.177766.

[88] Y. Boada *et al.*, ‘Characterization of Gene Circuit Parts Based on Multiobjective Optimization by Using Standard Calibrated Measurements’, *Chembiochem*, vol. 20, no. 20, pp. 2653–2665, Oct. 2019, doi: 10.1002/CBIC.201900272.

[89] J. Gilman, L. Walls, L. Bandiera, and F. Menolascina, ‘Statistical Design of Experiments for Synthetic Biology’, *ACS Synth Biol*, vol. 10, no. 1, pp. 1–18, Jan. 2021, doi: 10.1021/ACSSYNBIO.0C00385/ASSET/IMAGES/MEDIUM/SB0C00385_0004.GIF.

[90] E. Balsa-Canto, L. Bandiera, and F. Menolascina, ‘Optimal Experimental Design for Systems and Synthetic Biology Using AMIGO2’, *Methods in Molecular Biology*, vol. 2229, pp. 221–239, 2021, doi: 10.1007/978-1-0716-1032-9_11/FIGURES/6.

[91] O. Gallup, H. Ming, and T. Ellis, ‘Ten future challenges for synthetic biology’, *Engineering Biology*, vol. 5, no. 3, pp. 51–59, Sep. 2021, doi: 10.1049/ENB2.12011.

[92] L. Grozinger *et al.*, ‘Pathways to cellular supremacy in biocomputing’, *Nature Communications 2019 10:1*, vol. 10, no. 1, pp. 1–11, Nov. 2019, doi: 10.1038/s41467-019-13232-z.

[93] A. S. Khalil and J. J. Collins, ‘Synthetic biology: applications come of age’, *Nature Reviews Genetics 2010 11:5*, vol. 11, no. 5, pp. 367–379, May 2010, doi: 10.1038/nrg2775.

[94] R. García-Granados, J. A. Lerma-Escalera, and J. R. Morones-Ramírez, ‘Metabolic engineering and synthetic biology: Synergies, future, and challenges’, *Front Bioeng Biotechnol*, vol. 7, no. MAR, p. 36, 2019, doi: 10.3389/FBIOE.2019.00036/BIBTEX.

[95] W. Liu and R. Jiang, ‘Combinatorial and high-throughput screening approaches for strain engineering’, *Appl Microbiol Biotechnol*, vol. 99, no. 5, pp. 2093–2104, Feb. 2015, doi: 10.1007/S00253-015-6400-0/FIGURES/3.

[96] L. A. Mitchell *et al.*, ‘Versatile genetic assembly system (VEGAS) to assemble pathways for expression in S. cerevisiae’, *Nucleic Acids Res*, vol. 43, no. 13, pp. 6620–6630, Jul. 2015, doi: 10.1093/NAR/GKV466.

[97] G. A. Khoury, J. Smadbeck, C. A. Kieslich, and C. A. Floudas, ‘Protein folding and de novo protein design for biotechnological applications’, *Trends Biotechnol*, vol. 32, no. 2, pp. 99–109, Feb. 2014, doi: 10.1016/J.TIBTECH.2013.10.008.

[98] A. Krivoruchko and J. Nielsen, ‘Production of natural products through metabolic engineering of Saccharomyces cerevisiae’, *Curr Opin Biotechnol*, vol. 35, pp. 7–15, Dec. 2015, doi: 10.1016/J.COPBIO.2014.12.004.

[99] N. M. Markina, A. A. Kotlobay, and A. S. Tsarkova, ‘Heterologous Metabolic Pathways: Strategies for Optimal Expression in Eukaryotic Hosts’, *Acta Naturae*, vol. 12, no. 2, p. 28, Aug. 2020, doi: 10.32607/ACTANATURAE.10966.

[100] C. J. Paddon *et al.*, ‘High-level semi-synthetic production of the potent antimalarial artemisinin’, *Nature 2013 496:7446*, vol. 496, no. 7446, pp. 528–532, Apr. 2013, doi: 10.1038/nature12051.

[101] L. M. Adleman, ‘Molecular Computation of Solutions to Combinatorial Problems’, *Science (1979)*, vol. 266, no. 5187, pp. 1021–1024, Nov. 1994, doi: 10.1126/SCIENCE.7973651.

[102] ‘The Project Gutenberg eBook of The Story Of The Living Machine, by H.W. Conn.’ https://www.gutenberg.org/files/16487/16487-h/16487-h.htm (accessed Nov. 30, 2022).

[103] M. B. Elowitz and S. Leibier, ‘A synthetic oscillatory network of transcriptional regulators’, *Nature 2000 403:6767*, vol. 403, no. 6767, pp. 335–338, Jan. 2000, doi: 10.1038/35002125.

[104] T. S. Gardner, C. R. Cantor, and J. J. Collins, ‘Construction of a genetic toggle switch in Escherichia coli’, *Nature 2000 403:6767*, vol. 403, no. 6767, pp. 339–342, Jan. 2000, doi: 10.1038/35002131.

[105] A. Goñi-Moreno and P. I. Nikel, ‘High-Performance Biocomputing in Synthetic Biology–Integrated Transcriptional and Metabolic Circuits’, *Front Bioeng Biotechnol*, vol. 7, no. MAR, p. 40, 2019, doi: 10.3389/FBIOE.2019.00040.

[106] A. E. Friedland, T. K. Lu, X. Wang, D. Shi, G. Church, and J. J. Collins, ‘Synthetic gene networks that count’, *Science (1979)*, vol. 324, no. 5931, pp. 1199–1202, May 2009, doi: 10.1126/SCIENCE.1172005/SUPPL_FILE/FRIEDLAND.SOM.PDF.

[107] X. Li, L. Rizik, V. Kravchik, M. Khoury, N. Korin, and R. Daniel, ‘Synthetic neural-like computing in microbial consortia for pattern recognition’, *Nature Communications 2021 12:1*, vol. 12, no. 1, pp. 1–12, May 2021, doi: 10.1038/s41467-021-23336-0.

[108] J. J. Tabor *et al.*, ‘A Synthetic Genetic Edge Detection Program’, *Cell*, vol. 137, no. 7, pp. 1272–1281, Jun. 2009, doi: 10.1016/j.cell.2009.04.048.

[109] P. Fu, ‘Grand Challenges in Synthetic Biology to be Accomplished’, *Front Bioeng Biotechnol*, vol. 0, p. 2, 2013, doi: 10.3389/FBIOE.2013.00002.

[110] B. Saltepe, E. Ş. Kehribar, S. S. Su Yirmibeşoğlu, and U. Ö. Şafak Şeker, ‘Cellular Biosensors with Engineered Genetic Circuits’, *ACS Sens*, vol. 3, no. 1, pp. 13–26, Jan. 2018, doi: 10.1021/ACSSENSORS.7B00728/ASSET/IMAGES/LARGE/SE-2017-00728E_0005.JPEG.

[111] G. S. Hossain, M. Saini, R. Miyake, H. Ling, and M. W. Chang, ‘Genetic Biosensor Design for Natural Product Biosynthesis in Microorganisms’, *Trends Biotechnol*, vol. 38, no. 7, pp. 797–810, Jul. 2020, doi: 10.1016/J.TIBTECH.2020.03.013.

[112] S. Slomovic, K. Pardee, and J. J. Collins, ‘Synthetic biology devices for in vitro and in vivo diagnostics’, *Proc Natl Acad Sci U S A*, vol. 112, no. 47, pp. 14429–14435, Nov. 2015, doi: 10.1073/PNAS.1508521112/ASSET/4DBE653E-8604-42A8-8C71-CA6E5838A898/ASSETS/GRAPHIC/PNAS.1508521112FIG02.JPEG.

[113] H. Kaur, R. Kumar, J. N. Babu, and S. Mittal, ‘Advances in arsenic biosensor development – A comprehensive review’, *Biosens Bioelectron*, vol. 63, pp. 533–545, Jan. 2015, doi: 10.1016/J.BIOS.2014.08.003.

[114] D. Liu, T. Evans, and F. Zhang, ‘Applications and advances of metabolite biosensors for metabolic engineering’, *Metab Eng*, vol. 31, pp. 35–43, Sep. 2015, doi: 10.1016/J.YMBEN.2015.06.008.

[115] K. Pardee *et al.*, ‘Rapid, Low-Cost Detection of Zika Virus Using Programmable Biomolecular Components’, *Cell*, vol. 165, no. 5, pp. 1255–1266, May 2016, doi: 10.1016/j.cell.2016.04.059.

[116] S. Ravikumar, M. G. Baylon, S. J. Park, and J. il Choi, ‘Engineered microbial biosensors based on bacterial two-component systems as synthetic biotechnology platforms in bioremediation and biorefinery’, *Microb Cell Fact*, vol. 16, no. 1, pp. 1–10, Apr. 2017, doi: 10.1186/S12934-017-0675-Z/FIGURES/4.

[117] Y. Liu, Y. Liu, and M. Wang, ‘Design, optimization and application of small molecule biosensor in metabolic engineering’, *Front Microbiol*, vol. 8, no. OCT, p. 2012, Oct. 2017, doi: 10.3389/FMICB.2017.02012/BIBTEX.

[118] B. M. Woolston, T. Roth, I. Kohale, D. R. Liu, and G. Stephanopoulos, ‘Development of a formaldehyde biosensor with application to synthetic methylotrophy’, *Biotechnol Bioeng*, vol. 115, no. 1, pp. 206–215, Jan. 2018, doi: 10.1002/BIT.26455.

[119] S. Sridhar, C. M. Ajo-Franklin, and C. A. Masiello, ‘A Framework for the Systematic Selection of Biosensor Chassis for Environmental Synthetic Biology’, *ACS Synth Biol*, vol. 11, no. 9, pp. 2909–2916, Sep. 2022, doi: 10.1021/ACSSYNBIO.2C00079/ASSET/IMAGES/LARGE/SB2C00079_0002.JPEG.

[120] B. Wang, M. Barahona, and M. Buck, ‘A modular cell-based biosensor using engineered genetic logic circuits to detect and integrate multiple environmental signals’, *Biosensors and Bioelectronic*, vol. 40, pp. 368–376, 2012, doi: 10.1016/j.bios.2012.08.011.

[121] G. Yuan *et al.*, ‘Plant-Based Biosensors for Detecting CRISPR-Mediated Genome Engineering’, *ACS Synth Biol*, vol. 10, no. 12, pp. 3600–3603, Dec. 2021, doi: 10.1021/ACSSYNBIO.1C00455/ASSET/IMAGES/LARGE/SB1C00455_0001.JPEG.

[122] P. Banerjee and A. K. Bhunia, ‘Mammalian cell-based biosensors for pathogens and toxins’, *Trends Biotechnol*, vol. 27, no. 3, pp. 179–188, Mar. 2009, doi: 10.1016/J.TIBTECH.2008.11.006.

[123] J. K. Jung *et al.*, ‘Cell-free biosensors for rapid detection of water contaminants’, *Nature Biotechnology 2020 38:12*, vol. 38, no. 12, pp. 1451–1459, Jul. 2020, doi: 10.1038/s41587-020-0571-7.

[124] K. Khambhati, G. Bhattacharjee, N. Gohil, D. Braddick, V. Kulkarni, and V. Singh, ‘Exploring the Potential of Cell-Free Protein Synthesis for Extending the Abilities of Biological Systems’, *Front Bioeng Biotechnol*, vol. 7, p. 248, Oct. 2019, doi: 10.3389/FBIOE.2019.00248/BIBTEX.

[125] V. Libis, B. Delépine, and J. L. Faulon, ‘Expanding Biosensing Abilities through Computer-Aided Design of Metabolic Pathways’, *ACS Synth Biol*, vol. 5, no. 10, pp. 1076–1085, Oct. 2016, doi: 10.1021/ACSSYNBIO.5B00225/SUPPL_FILE/SB5B00225_SI_001.PDF.

[126] B. Townshend, J. S. Xiang, G. Manzanarez, E. J. Hayden, and C. D. Smolke, ‘A multiplexed, automated evolution pipeline enables scalable discovery and characterization of biosensors’, *Nature Communications 2021 12:1*, vol. 12, no. 1, pp. 1–15, Mar. 2021, doi: 10.1038/s41467-021-21716-0.

[127] C. E. French, K. de Mora, N. Joshi, A. Elfick, J. Haseloff, and J. Ajioka, *SYNTHETIC BIOLOGY AND THE ART OF BIOSENSOR DESIGN*. National Academies Press (US), 2011. Accessed: Nov. 30, 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK84465/

[128] R. Kent and N. Dixon, ‘Contemporary Tools for Regulating Gene Expression in Bacteria’, *Trends Biotechnol*, vol. 38, no. 3, pp. 316–333, Mar. 2020, doi: 10.1016/J.TIBTECH.2019.09.007.

[129] Y. Liu, Y. Zhuang, D. Ding, Y. Xu, J. Sun, and D. Zhang, ‘Biosensor-Based Evolution and Elucidation of a Biosynthetic Pathway in Escherichia coli’, *ACS Synth Biol*, vol. 6, no. 5, pp. 837–848, May 2017, doi: 10.1021/ACSSYNBIO.6B00328/SUPPL_FILE/SB6B00328_SI_001.PDF.

[130] R. K. Jha, S. Chakraborti, T. L. Kern, D. T. Fox, and C. E. M. Strauss, ‘Rosetta comparative modeling for library design: Engineering alternative inducer specificity in a transcription factor’, *Proteins: Structure, Function, and Bioinformatics*, vol. 83, no. 7, pp. 1327–1340, Jul. 2015, doi: 10.1002/PROT.24828.

[131] C. A. Miller, J. M. Ho, S. E. Parks, and M. R. Bennett, ‘Macrolide Biosensor Optimization through Cellular Substrate Sequestration’, *ACS Synth Biol*, vol. 10, no. 2, pp. 258–264, Feb. 2021, doi: 10.1021/ACSSYNBIO.0C00572/SUPPL_FILE/SB0C00572_SI_001.PDF.

[132] E. L. C. de Los Santos, J. T. Meyerowitz, S. L. Mayo, and R. M. Murray, 'Engineering Transcriptional Regulator Effector Specificity Using Computational Design and in Vitro Rapid Prototyping: Developing a Vanillin Sensor', *ACS Synth Biol*, vol. 5, no. 4, pp. 287–295, Apr. 2016, doi: 10.1021/ACSSYNBIO.5B00090/SUPPL_FILE/SB5B00090_SI_002.ZIP.

[133] R. Daniel, J. R. Rubens, R. Sarpeshkar, and T. K. Lu, 'Synthetic analog computation in living cells', *Nature 2013 497:7451*, vol. 497, no. 7451, pp. 619–623, May 2013, doi: 10.1038/nature12148.

[134] F. Ceroni *et al.*, 'Burden-driven feedback control of gene expression', *Nature Methods 2018 15:5*, vol. 15, no. 5, pp. 387–393, Mar. 2018, doi: 10.1038/nmeth.4635.

[135] P. B. Rainey and S. D. Quistad, 'Toward a dynamical understanding of microbial communities', *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 375, no. 1798, May 2020, doi: 10.1098/RSTB.2019.0248.

[136] R. J. Whitaker and J. F. Banfield, 'Population genomics in natural microbial communities', *Trends Ecol Evol*, vol. 21, no. 9, pp. 508–516, Sep. 2006, doi: 10.1016/j.tree.2006.07.001.

[137] S. Haruta and N. Kanno, 'Survivability of Microbes in Natural Environments and Their Ecological Impacts', *Microbes Environ*, vol. 30, no. 2, p. 123, Jun. 2015, doi: 10.1264/JSME2.ME3002RH.

[138] S. Giri, S. Waschina, C. Kaleta, and C. Kost, 'Defining Division of Labor in Microbial Communities', *J Mol Biol*, vol. 431, no. 23, pp. 4712–4731, Nov. 2019, doi: 10.1016/J.JMB.2019.06.023.

[139] A. Lykidis *et al.*, 'Multiple syntrophic interactions in a terephthalate-degrading methanogenic consortium', *The ISME Journal 2011 5:1*, vol. 5, no. 1, pp. 122–130, Aug. 2010, doi: 10.1038/ismej.2010.125.

[140] F. Villarreal *et al.*, 'Synthetic microbial consortia enable rapid assembly of pure translation machinery', *Nature Chemical Biology 2017 14:1*, vol. 14, no. 1, pp. 29–35, Nov. 2017, doi: 10.1038/nchembio.2514.

[141] A. R. T. Figueiredo and J. Kramer, 'Cooperation and Conflict Within the Microbiota and Their Effects On Animal Hosts', *Front Ecol Evol*, vol. 8, p. 132, May 2020, doi: 10.3389/FEVO.2020.00132/BIBTEX.

[142] L. Guo, X. He, and W. Shi, 'Intercellular communications in multispecies oral microbial communities', *Front Microbiol*, vol. 5, no. JULY, 2014, doi: 10.3389/FMICB.2014.00328.

[143] S. B. von Bodman, J. M. Willey, and S. P. Diggle, ‘Cell-Cell Communication in Bacteria: United We Stand’, *J Bacteriol*, vol. 190, no. 13, p. 4377, Jul. 2008, doi: 10.1128/JB.00486-08.

[144] G. P. Dubey and S. Ben-Yehuda, ‘Intercellular nanotubes mediate bacterial communication’, *Cell*, vol. 144, no. 4, pp. 590–600, Feb. 2011, doi: 10.1016/j.cell.2011.01.015.

[145] L. C. M. Antunes and R. B. R. Ferreira, ‘Intercellular communication in bacteria’, *http://dx.doi.org/10.1080/10408410902733946*, vol. 35, no. 2, pp. 69–80, 2009, doi: 10.1080/10408410902733946.

[146] R. T. Pena *et al.*, ‘Relationship between quorum sensing and secretion systems’, *Front Microbiol*, vol. 10, no. JUN, p. 1100, 2019, doi: 10.3389/FMICB.2019.01100/BIBTEX.

[147] S. Mukherjee and B. L. Bassler, ‘Bacterial quorum sensing in complex and dynamically changing environments’, *Nature Reviews Microbiology 2019 17:6*, vol. 17, no. 6, pp. 371–382, Apr. 2019, doi: 10.1038/s41579-019-0186-5.

[148] M. B. Miller and B. L. Bassler, ‘Quorum sensing in bacteria’, *Annu Rev Microbiol*, vol. 55, pp. 165–199, 2001, doi: 10.1146/ANNUREV.MICRO.55.1.165.

[149] W. C. Fuqua, S. C. Winans, and E. P. Greenberg, ‘Quorum sensing in bacteria: the LuxR-LuxI family of cell density-responsive transcriptional regulators.’, *J Bacteriol*, vol. 176, no. 2, p. 269, 1994, doi: 10.1128/JB.176.2.269-275.1994.

[150] S. T. Rutherford and B. L. Bassler, ‘Bacterial Quorum Sensing: Its Role in Virulence and Possibilities for Its Control’, *Cold Spring Harb Perspect Med*, vol. 2, no. 11, 2012, doi: 10.1101/CSHPERSPECT.A012427.

[151] N. Kylilis, Z. A. Tuza, G. B. Stan, and K. M. Polizzi, ‘Tools for engineering coordinated system behaviour in synthetic microbial consortia’, *Nature Communications 2018 9:1*, vol. 9, no. 1, pp. 1–9, Jul. 2018, doi: 10.1038/s41467-018-05046-2.

[152] M. v. Trushin, ‘Light-mediated “conversation” among microorganisms’, *Microbiol Res*, vol. 159, no. 1, pp. 1–10, Apr. 2004, doi: 10.1016/J.MICRES.2003.11.001.

[153] M. Diender, I. Parera Olm, and D. Z. Sousa, ‘Synthetic co-cultures: novel avenues for bio-based processes’, *Curr Opin Biotechnol*, vol. 67, pp. 72–79, Feb. 2021, doi: 10.1016/J.COPBIO.2021.01.006.

[154] L. Goers, P. Freemont, and K. M. Polizzi, ‘Co-culture systems and technologies: taking synthetic biology to the next level’, *J R Soc Interface*, vol. 11, no. 96, Jul. 2014, doi: 10.1098/RSIF.2014.0065.

[155] A. Burmeister *et al.*, '(Optochemical) Control of Synthetic Microbial Coculture Interactions on a Microcolony Level', *ACS Synth Biol*, vol. 10, no. 6, pp. 1308–1319, Jun. 2021, doi: 10.1021/ACSSYNBIO.0C00382/SUPPL_FILE/SB0C00382_SI_001.PDF.

[156] B. Lagoa-Costa, C. Kennes, and M. C. Veiga, 'Influence of feedstock mix ratio on microbial dynamics during acidogenic fermentation for polyhydroxyalkanoates production', *J Environ Manage*, vol. 303, Feb. 2022, doi: 10.1016/J.JENVMAN.2021.114132.

[157] R. U. Sheth, V. Cabral, S. P. Chen, and H. H. Wang, 'Manipulating Bacterial Communities by in situ Microbiome Engineering', *Trends Genet*, vol. 32, no. 4, p. 189, Apr. 2016, doi: 10.1016/J.TIG.2016.01.005.

[158] N. S. McCarty and R. Ledesma-Amaro, 'Synthetic Biology Tools to Engineer Microbial Communities for Biotechnology', *Trends Biotechnol*, vol. 37, no. 2, p. 181, Feb. 2019, doi: 10.1016/J.TIBTECH.2018.11.002.

[159] B. D. Karkaria, N. J. Treloar, C. P. Barnes, and A. J. H. Fedorec, 'From Microbial Communities to Distributed Computing Systems', *Front Bioeng Biotechnol*, vol. 8, p. 834, Jul. 2020, doi: 10.3389/FBIOE.2020.00834/BIBTEX.

[160] I. E. Müller, J. R. Rubens, T. Jun, D. Graham, R. Xavier, and T. K. Lu, 'Gene networks that compensate for crosstalk with crosstalk', *Nature Communications 2019 10:1*, vol. 10, no. 1, pp. 1–8, Sep. 2019, doi: 10.1038/s41467-019-12021-y.

[161] J. Macía, F. Posas, and R. v. Solé, 'Distributed computation: the new wave of synthetic biology devices', *Trends Biotechnol*, vol. 30, no. 6, pp. 342–349, Jun. 2012, doi: 10.1016/J.TIBTECH.2012.03.006.

[162] J. M. Smith, R. Chowdhry, and M. J. Booth, 'Controlling Synthetic Cell-Cell Communication', *Front Mol Biosci*, vol. 8, p. 1321, Jan. 2022, doi: 10.3389/FMOLB.2021.809945/BIBTEX.

[163] A. I. Curatolo *et al.*, 'Cooperative pattern formation in multi-component bacterial systems through reciprocal motility regulation', *Nature Physics 2020 16:11*, vol. 16, no. 11, pp. 1152–1157, Aug. 2020, doi: 10.1038/s41567-020-0964-z.

[164] E. Osmekhina *et al.*, 'Controlled communication between physically separated bacterial populations in a microfluidic device', *Communications Biology 2018 1:1*, vol. 1, no. 1, pp. 1–7, Jul. 2018, doi: 10.1038/s42003-018-0102-y.

[165] N. I. Johns, T. Blazejewski, A. L. C. Gomes, and H. H. Wang, ‘Principles for designing synthetic microbial communities’, *Curr Opin Microbiol*, vol. 31, pp. 146–153, Jun. 2016, doi: 10.1016/J.MIB.2016.03.010.

[166] P. S. Losoi, V. P. Santala, and S. M. Santala, ‘Enhanced Population Control in a Synthetic Bacterial Consortium by Interconnected Carbon Cross-Feeding’, *ACS Synth Biol*, vol. 8, no. 12, pp. 2642–2650, Dec. 2019, doi: 10.1021/ACSSYNBIO.9B00316/SUPPL_FILE/SB9B00316_SI_001.PDF.

[167] T. Xia, M. A. Eiteman, and E. Altman, ‘Simultaneous utilization of glucose, xylose and arabinose in the presence of acetate by a consortium of Escherichia coli strains’, *Microb Cell Fact*, vol. 11, no. 1, pp. 1–9, Jun. 2012, doi: 10.1186/1475-2859-11-77/FIGURES/6.

[168] P. Jayaraman, K. Devarajan, T. K. Chua, H. Zhang, E. Gunawan, and C. L. Poh, ‘Blue light-mediated transcriptional activation and repression of gene expression in bacteria.’, *Nucleic Acids Res*, vol. 44, no. 14, pp. 6994–7005, 2016, doi: 10.1093/nar/gkw548.

[169] J. D. Hunter, ‘Matplotlib: A 2D graphics environment’, *Comput Sci Eng*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[170] Y. M. Wang *et al.*, ‘Single-molecule studies of repressor-DNA interactions show long-range interactions’, *Proc Natl Acad Sci U S A*, vol. 102, no. 28, pp. 9796–9801, Jul. 2005, doi: 10.1073/PNAS.0502917102/SUPPL_FILE/02917FIG13.JPG.

[171] E. Roberts, A. Magis, J. O. Ortiz, W. Baumeister, and Z. Luthey-Schulten, ‘Noise Contributions in an Inducible Genetic Switch: A Whole-Cell Simulation Study’, *PLoS Comput Biol*, vol. 7, no. 3, p. e1002010, 2011, doi: 10.1371/JOURNAL.PCBI.1002010.

[172] O. Purcell, C. S. Grierson, M. D. Bernardo, and N. J. Savery, ‘Temperature dependence of ssrA-tag mediated protein degradation’, *J Biol Eng*, vol. 6, no. 1, pp. 1–3, Jul. 2012, doi: 10.1186/1754-1611-6-10/FIGURES/1.

[173] U. Vogel and K. F. Jensen, ‘The RNA chain elongation rate in Escherichia coli depends on the growth rate’, *J Bacteriol*, vol. 176, no. 10, pp. 2807–2813, 1994, doi: 10.1128/JB.176.10.2807-2813.1994.

[174] H. Bremer and P. P. Dennis, ‘Modulation of Chemical Composition and Other Parameters of the Cell at Different Exponential Growth Rates’, *EcoSal Plus*, vol. 3, no. 1, Feb. 2008, doi: 10.1128/ECOSAL.5.2.3.

[175] P. K. Grant *et al.*, ‘Orthogonal intercellular signaling for programmed spatial behavior’, *Mol Syst Biol*, vol. 12, no. 1, 2016, doi: 10.15252/msb.20156590.

[176] Y. Taniguchi *et al.*, ‘Quantifying E. coli proteome and transcriptome with single-molecule sensitivity in single cells’, *Science*, vol. 329, no. 5991, pp. 533–538, Jul. 2010, doi: 10.1126/SCIENCE.1188308.

[177] M. R. Maurizi, ‘Proteases and protein degradation in Escherichia coli’, *Experientia*, vol. 48, no. 2, pp. 178–201, Feb. 1992, doi: 10.1007/BF01923511.

[178] N. Saeidi, M. Arshath, M. W. Chang, and C. L. Poh, ‘Characterization of a quorum sensing device for synthetic biology design: Experimental and modeling validation’, *Chem Eng Sci*, vol. 103, pp. 91–99, Nov. 2013, doi: 10.1016/J.CES.2012.12.016.

[179] A. Pai, Y. Tanouchi, C. H. Collins, and L. You, ‘Engineering multicellular systems by cell-cell communication’, *Curr Opin Biotechnol*, vol. 20, no. 4, p. 461, Aug. 2009, doi: 10.1016/J.COPBIO.2009.08.006.

[180] S. Wellington and E. Peter Greenberg, ‘Quorum sensing signal selectivity and the potential for interspecies cross talk’, *mBio*, vol. 10, no. 2, 2019, doi: 10.1128/MBIO.00146-19/SUPPL_FILE/MBIO.00146-19-SF006.TIF.

[181] A. Raychaudhuri, A. Jerga, and P. A. Tipton, ‘Chemical mechanism and substrate specificity of RhlI, an acylhomoserine lactone synthase from Pseudomonas aeruginosa’, *Biochemistry*, vol. 44, no. 8, pp. 2974–2981, Mar. 2005, doi: 10.1021/BI048005M/ASSET/IMAGES/LARGE/BI048005MH00002.JPEG.

[182] A. Claussen *et al.*, ‘Kinetic Model for Signal Binding to the Quorum Sensing Regulator LasR’, *International Journal of Molecular Sciences 2013, Vol. 14, Pages 13360-13376*, vol. 14, no. 7, pp. 13360–13376, Jun. 2013, doi: 10.3390/IJMS140713360.

[183] K. J. Sappington, A. A. Dandekar, K. I. Oinuma, and E. P. Greenberg, ‘Reversible signal binding by the Pseudomonas aeruginosa quorum-sensing signal receptor LasR’, *mBio*, vol. 2, no. 1, 2011, doi: 10.1128/MBIO.00011-11/ASSET/1C5B6046-7461-4D38-878C-5B58D22676A3/ASSETS/GRAPHIC/MBO0011110910004.JPEG.

[184] S. Hernando-Amado, M. Alcalde-Rico, T. Gil-Gil, J. R. Valverde, and J. L. Martínez, ‘Naringenin Inhibition of the Pseudomonas aeruginosa Quorum Sensing Response Is Based on Its Time-Dependent Competition With N-(3-Oxo-

dodecanoyl)-L-homoserine Lactone for LasR Binding', *Front Mol Biosci*, vol. 7, p. 25, Feb. 2020, doi: 10.3389/FMOLB.2020.00025/BIBTEX.

[185] N. Chowdhury and A. Bagchi, 'Identification of ligand binding activity and DNA recognition by RhlR protein from opportunistic pathogen Pseudomonas aeruginosa—a molecular dynamic simulation approach', *Journal of Molecular Recognition*, vol. 31, no. 12, p. e2738, Dec. 2018, doi: 10.1002/JMR.2738.

[186] K. Brenner, D. K. Karig, R. Weiss, and F. H. Arnold, 'Engineered bidirectional communication mediates a consensus in a microbial biofilm consortium', *Proc Natl Acad Sci U S A*, vol. 104, no. 44, pp. 17300–17304, Oct. 2007, doi: 10.1073/PNAS.0704256104/SUPPL_FILE/04256SUPPTEXT.PDF.

[187] G. Medina, K. Juárez, B. Valderrama, and G. Soberón-Chávez, 'Mechanism of Pseudomonas aeruginosa RhlR Transcriptional Regulation of the rhlAB Promoter', *J Bacteriol*, vol. 185, no. 20, p. 5976, Oct. 2003, doi: 10.1128/JB.185.20.5976-5983.2003.

[188] C. J. van Dolleweerd *et al.*, 'MIDAS: A Modular DNA Assembly System for Synthetic Biology', *ACS Synth Biol*, vol. 7, no. 4, pp. 1018–1029, Apr. 2018, doi: 10.1021/ACSSYNBIO.7B00363/SUPPL_FILE/SB7B00363_SI_001.PDF.

[189] E. Weber, C. Engler, R. Gruetzner, S. Werner, and S. Marillonnet, 'A Modular Cloning System for Standardized Assembly of Multigene Constructs', *PLoS One*, vol. 6, no. 2, p. e16765, 2011, doi: 10.1371/JOURNAL.PONE.0016765.

[190] G. Røkke, E. Korvald, J. Pahr, O. Øyås, and R. Lale, 'BioBrick assembly standards and techniques and associated software tools', *Methods in Molecular Biology*, vol. 1116, pp. 1–24, 2014, doi: 10.1007/978-1-62703-764-8_1/FIGURES/13.

[191] T. Ellis, T. Adie, and G. S. Baldwin, 'DNA assembly for synthetic biology: from parts to pathways and beyond', *Integr Biol (Camb)*, vol. 3, no. 2, pp. 109–118, 2011, doi: 10.1039/C0IB00070A.

[192] M. English, J. Buckley, and J. J. Collins, 'Investigating software modularity using class and module level metrics', *Software Quality Assurance: In Large Scale and Complex Software-intensive Systems*, pp. 177–200, Jan. 2016, doi: 10.1016/B978-0-12-802301-3.00008-9.

[193] K. Otto, K. Hölttä-Otto, T. W. Simpson, D. Krause, S. Ripperda, and S. K. Moon, 'Global Views on Modular Design Research: Linking Alternative Methods to Support Modular Product Family Concept Development', *Journal of Mechanical*

Design, Transactions of the ASME, vol. 138, no. 7, Jul. 2016, doi: 10.1115/1.4033654/376182.

[194] A. Tamsir, J. J. Tabor, and C. A. Voigt, 'Robust multicellular computing using genetically encoded NOR gates and chemical "wires"', *Nature 2010 469:7329*, vol. 469, no. 7329, pp. 212–215, Dec. 2010, doi: 10.1038/nature09565.

[195] P. L. Voyvodic *et al.*, 'Plug-and-play metabolic transducers expand the chemical detection space of cell-free biosensors', *Nature Communications 2019 10:1*, vol. 10, no. 1, pp. 1–8, Apr. 2019, doi: 10.1038/s41467-019-09722-9.

[196] J. Macia *et al.*, 'Implementation of Complex Biological Logic Circuits Using Spatially Distributed Multicellular Consortia', *PLoS Comput Biol*, vol. 12, no. 2, p. e1004685, Feb. 2016, doi: 10.1371/JOURNAL.PCBI.1004685.

[197] M. Kamali and K. Hewage, 'Life cycle performance of modular buildings: A critical review', *Renewable and Sustainable Energy Reviews*, vol. 62, pp. 1171–1183, Sep. 2016, doi: 10.1016/J.RSER.2016.05.031.

[198] T. R. Browning, 'Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities', *IEEE Trans Eng Manag*, vol. 63, no. 1, pp. 27–52, Feb. 2016, doi: 10.1109/TEM.2015.2491283.

[199] Y. H. Kim, L. K. Park, S. Yiacoumi, and C. Tsouris, 'Modular Chemical Process Intensification: A Review', *https://doi.org/10.1146/annurev-chembioeng-060816-101354*, vol. 8, pp. 359–380, Jun. 2017, doi: 10.1146/ANNUREV-CHEMBIOENG-060816-101354.

[200] E. M. Dashofy, A. van der Hoek, and R. N. Taylor, 'A comprehensive approach for the development of modular software architecture description languages', *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 14, no. 2, pp. 199–245, Apr. 2005, doi: 10.1145/1061254.1061258.

[201] W. G. Griswold *et al.*, 'Modular software design with crosscutting interfaces', *IEEE Softw*, vol. 23, no. 1, pp. 51–60, Jan. 2006, doi: 10.1109/MS.2006.24.

[202] H. Tsukune *et al.*, 'Modular manufacturing', *J Intell Manuf*, vol. 4, pp. 163–183, 1993.

[203] I. Sommerville, *Software Engineering*, 10th ed. Pearson Education, 2016. Accessed: Nov. 30, 2022. [Online]. Available: https://www.academia.edu/50882590/Sommerville_Software_Engineering_10ed

[204] T. R. Gruber, ‘Toward principles for the design of ontologies used for knowledge sharing?’, *Int J Hum Comput Stud*, vol. 43, no. 5–6, pp. 907–928, Nov. 1995, doi: 10.1006/IJHC.1995.1081.

[205] G. Mlslrll *et al.*, ‘SBOL-OWL: An Ontological Approach for Formal and Semantic Representation of Synthetic Biology Information’, *ACS Synth Biol*, vol. 8, no. 7, pp. 1498–1514, Jul. 2019, doi: 10.1021/ACSSYNBIO.8B00532/SUPPL_FILE/SB8B00532_SI_001.PDF.

[206] K. Eilbeck *et al.*, ‘The Sequence Ontology: a tool for the unification of genome annotations’, *Genome Biology 2005 6:5*, vol. 6, no. 5, pp. 1–12, Apr. 2005, doi: 10.1186/GB-2005-6-5-R44.

[207] M. Courtot *et al.*, ‘Controlled vocabularies and semantics in systems biology’, *Mol Syst Biol*, vol. 7, no. 1, p. 543, Jan. 2011, doi: 10.1038/MSB.2011.77.

[208] M. A. Harris *et al.*, ‘The Gene Ontology (GO) database and informatics resource’, *Nucleic Acids Res*, vol. 32, no. suppl_1, pp. D258–D261, Jan. 2004, doi: 10.1093/NAR/GKH036.

[209] K. Degtyarenko *et al.*, ‘ChEBI: a database and ontology for chemical entities of biological interest’, *Nucleic Acids Res*, vol. 36, no. Database issue, p. D344, Jan. 2008, doi: 10.1093/NAR/GKM791.

[210] E. Demir *et al.*, ‘BioPAX – A community standard for pathway data sharing’, *Nat Biotechnol*, vol. 28, no. 9, p. 935, Sep. 2010, doi: 10.1038/NBT.1666.

[211] M. Kostylev, A. E. Otwell, R. E. Richardson, and Y. Suzuki, ‘Cloning Should Be Simple: Escherichia coli DH5α-Mediated Assembly of Multiple DNA Fragments with Short End Homologies’, *PLoS One*, vol. 10, no. 9, Sep. 2015, doi: 10.1371/JOURNAL.PONE.0137466.

[212] H. Jeong, H. J. Kim, and S. J. Lee, ‘Complete Genome Sequence of Escherichia coli Strain BL21’, *Genome Announc*, vol. 3, no. 2, pp. 134–149, 2015, doi: 10.1128/GENOMEA.00134-15.

[213] K. E. Duncker, Z. A. Holmes, and L. You, ‘Engineered microbial consortia: strategies and applications’, *Microbial Cell Factories 2021 20:1*, vol. 20, no. 1, pp. 1–13, Nov. 2021, doi: 10.1186/S12934-021-01699-9.

[214] S. Gupta, T. D. Ross, M. M. Gomez, J. L. Grant, P. A. Romero, and O. S. Venturelli, ‘Investigating the dynamics of microbial consortia in spatially structured environments’, *Nature Communications 2020 11:1*, vol. 11, no. 1, pp. 1–15, May 2020, doi: 10.1038/s41467-020-16200-0.

[215] K. Vishwakarma, N. Kumar, C. Shandilya, S. Mohapatra, S. Bhayana, and A. Varma, ‘Revisiting Plant–Microbe Interactions and Microbial Consortia Application for Enhancing Sustainable Agriculture: A Review’, *Front Microbiol*, vol. 11, p. 3195, Dec. 2020, doi: 10.3389/FMICB.2020.560406/BIBTEX.

[216] M. M. Jessop-Fabre and N. Sonnenschein, ‘Improving reproducibility in synthetic biology’, *Front Bioeng Biotechnol*, vol. 7, no. FEB, p. 18, 2019, doi: 10.3389/FBIOE.2019.00018/BIBTEX.

[217] X. Li *et al.*, ‘Design of stable and self-regulated microbial consortia for chemical synthesis’, *Nature Communications 2022 13:1*, vol. 13, no. 1, pp. 1–9, Mar. 2022, doi: 10.1038/s41467-022-29215-6.

[218] C. J. Myers *et al.*, ‘A standard-enabled workflow for synthetic biology’, *Biochem Soc Trans*, vol. 45, no. 3, pp. 793–803, Jun. 2017, doi: 10.1042/BST20160347.

[219] D. Ross, P. D. Tonner, and O. B. Vasilyeva, ‘Method for reproducible automated bacterial cell culture and measurement’, *Synth Biol*, vol. 7, no. 1, Nov. 2022, doi: 10.1093/SYNBIO/YSAC013.

[220] J. A. Yeow, P. K. Ng, K. S. Tan, T. S. Chin, and W. Y. Lim, ‘Effects of Stress, Repetition, Fatigue and Work Environment on Human Error in Manufacturing Industries’, *Journal of Applied Sciences*, vol. 14, no. 24, pp. 3464–3471, Dec. 2014, doi: 10.3923/JAS.2014.3464.3471.

[221] D. Bryce *et al.*, ‘Round Trip: An Automated Pipeline for Experimental Design, Execution, and Analysis’, *ACS Synth Biol*, vol. 11, no. 2, pp. 608–622, Feb. 2022, doi: 10.1021/ACSSYNBIO.1C00305/SUPPL_FILE/SB1C00305_SI_001.PDF.

[222] I. Otero-Muras and P. Carbonell, ‘Automated engineering of synthetic metabolic pathways for efficient biomanufacturing’, *Metab Eng*, vol. 63, pp. 61–80, Jan. 2021, doi: 10.1016/J.YMBEN.2020.11.012.

[223] M. Storch, M. C. Haines, and G. S. Baldwin, ‘DNA-BOT: a low-cost, automated DNA assembly platform for synthetic biology’, *Synth Biol*, vol. 5, no. 1, Jan. 2020, doi: 10.1093/SYNBIO/YSAA010.

[224] D. I. Walsh *et al.*, ‘Standardizing Automated DNA Assembly: Best Practices, Metrics, and Protocols Using Robots’, *SLAS Technol*, vol. 24, no. 3, p. 282, Jun. 2019, doi: 10.1177/2472630318825335.

[225] N. Hillson *et al.*, ‘Building a global alliance of biofoundries’, *Nature Communications 2019 10:1*, vol. 10, no. 1, pp. 1–4, May 2019, doi: 10.1038/s41467-019-10079-2.

[226] N. Rupp, K. Peschke, M. Köppl, D. Drissner, and T. Zuchner, ‘Establishment of low-cost laboratory automation processes using AutoIt and 4-axis robots’, *SLAS Technol*, vol. 27, no. 5, pp. 312–318, Oct. 2022, doi: 10.1016/J.SLAST.2022.07.001.

[227] M. A. Torres-Acosta, G. J. Lye, and D. Dikicioglu, ‘Automated liquid-handling operations for robust, resilient, and efficient bio-based laboratory practices’, *Biochem Eng J*, vol. 188, p. 108713, Dec. 2022, doi: 10.1016/J.BEJ.2022.108713.

[228] D. Kumar, K. Achuthan, B. Nair, and S. Diwakar, ‘Online bio-robotics labs: Open hardware models and architecture’, *International Conference on Robotics and Automation for Humanitarian Applications, RAHA 2016 - Conference Proceedings*, May 2017, doi: 10.1109/RAHA.2016.7931877.

[229] S. M. Brooks and H. S. Alper, ‘Applications, challenges, and needs for employing synthetic biology beyond the lab’, *Nature Communications 2021 12:1*, vol. 12, no. 1, pp. 1–16, Mar. 2021, doi: 10.1038/s41467-021-21740-0.

[230] G. Linshiz, N. Stawski, S. Poust, C. Bi, J. D. Keasling, and N. J. Hillson, ‘PaR-PaR laboratory automation platform’, *ACS Synth Biol*, vol. 2, no. 5, pp. 216–222, May 2013, doi: 10.1021/SB300075T/SUPPL_FILE/SB300075T_SI_002.ZIP.

[231] L. Gautier, ‘Reproducibility in the lab: the proof is in the protocol’, *Biotechniques*, vol. 73, no. 2, pp. 71–74, Aug. 2022, doi: 10.2144/BTN-2022-0074/ASSET/IMAGES/LARGE/FIGURE2.JPEG.

[232] E. Appleton, D. Densmore, C. Madsen, and N. Roehner, ‘Needs and opportunities in bio-design automation: four areas for focus’, *Curr Opin Chem Biol*, vol. 40, pp. 111–118, Oct. 2017, doi: 10.1016/J.CBPA.2017.08.005.

[233] ‘How Synthace Works | Synthace Experiment Platform’. https://www.synthace.com/how-it-works (accessed Nov. 30, 2022).

[234] E. C. Hayden, ‘The automated lab’, *Nature 2014 516:7529*, vol. 516, no. 7529, pp. 131–132, Dec. 2014, doi: 10.1038/516131a.

[235] P. Carbonell, T. Radivojevic, and H. García Martín, ‘Opportunities at the Intersection of Synthetic Biology, Machine Learning, and Automation’, *ACS Synth Biol*, vol. 8, no. 7, pp. 1474–1477, Jul. 2019, doi: 10.1021/ACSSYNBIO.8B00540/ASSET/IMAGES/LARGE/SB-2018-00540F_0002.JPEG.

[236] C. Armer, S. Golas, T. Kalil, and E. Debenedictis, ‘Barriers to Academic Use of Commercial Cloud Labs’, doi: 10.33552/OJRAT.2022.01.000511.

[237] M. I. Sadowski, C. Grant, and T. S. Fell, ‘Harnessing QbD, Programming Languages, and Automation for Reproducible Biology’, *Trends Biotechnol*, vol. 34, no. 3, pp. 214–227, Mar. 2016, doi: 10.1016/J.TIBTECH.2015.11.006.

[238] E. J. Chory, D. W. Gretton, E. A. DeBenedictis, and K. M. Esvelt, ‘Enabling high-throughput biology with flexible open-source automation’, *Mol Syst Biol*, vol. 17, no. 3, p. 9942, Mar. 2021, doi: 10.15252/MSB.20209942.

[239] V. Gupta, J. Irimia, I. Pau, and A. Rodríguez-Patón, ‘BioBlocks: Programming Protocols in Biology Made Easier’, *ACS Synth Biol*, vol. 6, no. 7, pp. 1230–1232, Jul. 2017, doi: 10.1021/ACSSYNBIO.6B00304/SUPPL_FILE/SB6B00304_SI_001.PDF.

[240] V. Ananthanarayanan and W. Thies, ‘Biocoder: A programming language for standardizing and automating biology protocols’, *Journal of Biological Engineering 2010 4:1*, vol. 4, no. 1, pp. 1–13, Nov. 2010, doi: 10.1186/1754-1611-4-13.

[241] ‘PyLabRobot/pylabrobot: A hardware agnostic platform for liquid handling’. https://github.com/pylabrobot/pylabrobot (accessed Nov. 30, 2022).

[242] ‘dgretton/pyhamilton: Python for Hamilton liquid handling robots’. https://github.com/dgretton/pyhamilton (accessed Nov. 30, 2022).

[243] W. Ouyang *et al.*, ‘An Open-Source Modular Framework for Automated Pipetting and Imaging Applications’, *Adv Biol*, vol. 6, no. 4, p. 2101063, Apr. 2022, doi: 10.1002/ADBI.202101063.

[244] K. T. Walker *et al.*, ‘CONTAIN: An open-source shipping container laboratory optimised for automated COVID-19 diagnostics’, *bioRxiv*, p. 2020.05.20.106625, May 2020, doi: 10.1101/2020.05.20.106625.

[245] T. Sanderson and J. C. Rayner, ‘PlasmoTron: an open-source platform for automated culture of malaria parasites’, doi: 10.1101/241596.

[246] A. S. Karim *et al.*, ‘Modular cell-free expression plasmids to accelerate biological design in cells’, *Synth Biol*, vol. 5, no. 1, p. 19, Jan. 2020, doi: 10.1093/SYNBIO/YSAA019.

[247] R. J. Grant *et al.*, ‘Achieving Accurate Compound Concentration in Cell-Based Screening: Validation of Acoustic Droplet Ejection Technology’, *SLAS Discovery*, vol. 14, no. 5, pp. 452–459, May 2009, doi: 10.1177/1087057109336588.

[248] M. A. Crone *et al.*, ‘A role for Biofoundries in rapid development and validation of automated SARS-CoV-2 clinical diagnostics’, *Nature Communications 2020 11:1*, vol. 11, no. 1, pp. 1–11, Sep. 2020, doi: 10.1038/s41467-020-18130-3.

[249] J. W. Yeoh *et al.*, ‘SynBiopython: an open-source software library for Synthetic Biology’, *Synth Biol*, vol. 6, no. 1, Nov. 2021, doi: 10.1093/SYNBIO/YSAB001.

[250] J. Duo, H. Dong, B. DeSilva, and Y. J. Zhang, ‘A generic template for automated bioanalytical ligand-binding assays using modular robotic scripts in support of discovery biotherapeutic programs’, *Bioanalysis*, vol. 5, no. 14, pp. 1735–1750, 2013, doi: 10.4155/BIO.13.154.

[251] S. H. Friedman *et al.*, ‘MultiCellDS: a standard and a community for sharing multicellular data’, doi: 10.1101/090696.

[252] N. J. Hillson, H. A. Plahar, J. Beal, and R. Prithviraj, ‘Improving synthetic biology communication: Recommended practices for visual depiction and digital submission of genetic designs’, *ACS Synth Biol*, vol. 5, no. 6, pp. 449–451, Jun. 2016, doi: 10.1021/ACSSYNBIO.6B00146/ASSET/IMAGES/LARGE/SB-2016-001468_0001.JPEG.

[253] E. Oberortner and D. Densmore, ‘Web-Based Software Tool for Constraint-Based Design Specification of Synthetic Biological Systems’, *ACS Synth Biol*, vol. 4, no. 6, pp. 757–760, Jun. 2015, doi: 10.1021/SB500352B/ASSET/IMAGES/MEDIUM/SB-2014-00352B_0002.GIF.

[254] N. Dalchau, P. K. Grant, P. Vaidyanathan, C. Spaccasassi, C. Gravill, and A. Phillips, ‘Scalable dynamic characterization of synthetic gene circuits’, *bioRxiv*, no. 5, p. 635672, Aug. 2019, doi: 10.1101/635672.

[255] A. A. K. Nielsen *et al.*, ‘Genetic circuit design automation’, *Science (1979)*, vol. 352, no. 6281, Apr. 2016, doi: 10.1126/SCIENCE.AAC7341/SUPPL_FILE/NIELSEN.SM.PDF.

[256] M. J. Czar, Y. Cai, and J. Peccoud, ‘Writing DNA with GenoCAD™’, *Nucleic Acids Res*, vol. 37, no. suppl_2, pp. W40–W47, Jul. 2009, doi: 10.1093/NAR/GKP361.

[257] M. Zhang, J. A. McLaughlin, A. Wipat, and C. J. Myers, ‘SBOLDesigner 2: An Intuitive Tool for Structural Genetic Design’, *ACS Synth Biol*, vol. 6, no. 7, pp. 1150–1160, Jul. 2017, doi: 10.1021/ACSSYNBIO.6B00275/ASSET/IMAGES/MEDIUM/SB-2016-00275D_0011.GIF.

[258] J. A. McLaughlin *et al.*, ‘SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology’, *ACS Synth Biol*, vol. 7, no. 2, pp. 682–688, Feb. 2018, doi: 10.1021/ACSSYNBIO.7B00403/ASSET/IMAGES/MEDIUM/SB-2017-004037_0007.GIF.

[259] G. Mlslrll, J. Beal, T. E. Gorochowski, G. B. Stan, A. Wipat, and C. J. Myers, ‘SBOL Visual 2 Ontology’, *ACS Synth Biol*, vol. 9, no. 4, pp. 972–977, Apr. 2020, doi: 10.1021/ACSSYNBIO.0C00046/ASSET/IMAGES/MEDIUM/SB0C00046_0004.GIF.

[260] G. Misirli *et al.*, ‘A Computational Workflow for the Automated Generation of Models of Genetic Designs’, *ACS Synth Biol*, vol. 8, no. 7, pp. 1548–1559, Jul. 2019, doi: 10.1021/ACSSYNBIO.7B00459/ASSET/IMAGES/MEDIUM/SB-2017-004597_0012.GIF.

[261] S. Konur *et al.*, ‘Toward Full-Stack in Silico Synthetic Biology: Integrating Model Specification, Simulation, Verification, and Biological Compilation’, *ACS Synth Biol*, vol. 10, no. 8, pp. 1931–1945, Aug. 2021, doi: 10.1021/ACSSYNBIO.1C00143/ASSET/IMAGES/LARGE/SB1C00143_0009.JPEG.

[262] B. Hatch, L. Meng, J. Mante, J. A. McLaughlin, J. Scott-Brown, and C. J. Myers, ‘VisBOL2 - Improving Web-Based Visualization for Synthetic Biology Designs’, *ACS Synth Biol*, vol. 10, no. 8, pp. 2111–2115, Aug. 2021, doi: 10.1021/ACSSYNBIO.1C00147/ASSET/IMAGES/MEDIUM/SB1C00147_0004.GIF.

[263] R. Kent and N. Dixon, ‘Systematic Evaluation of Genetic and Environmental Factors Affecting Performance of Translational Riboswitches’, *ACS Synth Biol*, vol. 8, no. 4, pp. 884–901, Apr. 2019, doi: 10.1021/ACSSYNBIO.9B00017/ASSET/IMAGES/LARGE/SB-2019-00017Y_0009.JPEG.

[264] A. M. Banks *et al.*, ‘Key reaction components affect the kinetics and performance robustness of cell-free protein synthesis reactions’, *Comput Struct Biotechnol J*, vol. 20, pp. 218–229, Jan. 2022, doi: 10.1016/J.CSBJ.2021.12.013.

[265] P. Carbonell *et al.*, ‘An automated Design-Build-Test-Learn pipeline for enhanced microbial production of fine chemicals’, *Communications Biology 2018 1:1*, vol. 1, no. 1, pp. 1–10, Jun. 2018, doi: 10.1038/s42003-018-0076-9.

[266] E. Bartocci and P. Lió, ‘Computational Modeling, Formal Analysis, and Tools for Systems Biology’, *PLoS Comput Biol*, vol. 12, no. 1, p. e1004591, 2016, doi: 10.1371/JOURNAL.PCBI.1004591.

[267] L. Grozinger, E. Heidrich, and Á. Goñi-Moreno, ‘An electrogenetic toggle switch model’, *Microb Biotechnol*, 2022, doi: 10.1111/1751-7915.14153.

[268] L. Endler *et al.*, ‘Designing and encoding models for synthetic biology’, *J R Soc Interface*, vol. 6, no. SUPPL. 4, Aug. 2009, doi: 10.1098/RSIF.2009.0035.FOCUS.

[269] A. Niarakis and T. Helikar, ‘A practical guide to mechanistic systems modeling in biology using a logic-based approach’, *Brief Bioinform*, vol. 22, no. 4, Jul. 2021, doi: 10.1093/BIB/BBAA236.

[270] J. W. Yeoh, K. B. I. Ng, A. Y. Teh, J. Y. Zhang, W. K. D. Chee, and C. L. Poh, ‘An Automated Biomodel Selection System (BMSS) for Gene Circuit Designs’, *ACS Synth Biol*, vol. 8, no. 7, pp. 1484–1497, Jul. 2019, doi: 10.1021/ACSSYNBIO.8B00523/SUPPL_FILE/SB8B00523_SI_001.PDF.

[271] M. Prado Casanova, ‘Noise and Synthetic Biology: How to Deal with Stochasticity?’, *NanoEthics 2020 14:1*, vol. 14, no. 1, pp. 113–122, Apr. 2020, doi: 10.1007/S11569-020-00366-4.

[272] D. J. Wilkinson, ‘Stochastic modelling for quantitative description of heterogeneous biological systems’, *Nature Reviews Genetics 2009 10:2*, vol. 10, no. 2, pp. 122–133, Feb. 2009, doi: 10.1038/nrg2509.

[273] E. Sharon *et al.*, ‘Probing the effect of promoters on noise in gene expression using thousands of designed sequences’, *Genome Res*, vol. 24, no. 10, pp. 1698–1706, Oct. 2014, doi: 10.1101/GR.168773.113.

[274] Á. Goñi-Moreno, I. Benedetti, J. Kim, and V. de Lorenzo, ‘Deconvolution of Gene Expression Noise into Spatial Dynamics of Transcription Factor-Promoter Interplay’, *ACS Synth Biol*, vol. 6, no. 7, pp. 1359–1369, Jul. 2017, doi: 10.1021/ACSSYNBIO.6B00397/SUPPL_FILE/SB6B00397_SI_002.ZIP.

[275] G. Simoni, F. Reali, C. Priami, and L. Marchetti, ‘Stochastic simulation algorithms for computational systems biology: Exact, approximate, and hybrid methods’, *Wiley Interdiscip Rev Syst Biol Med*, vol. 11, no. 6, p. e1459, Nov. 2019, doi: 10.1002/WSBM.1459.

[276] S. K. Hahl and A. Kremling, ‘A Comparison of Deterministic and Stochastic Modeling Approaches for Biochemical Reaction Systems: On Fixed Points, Means, and Modes’, *Front Genet*, vol. 7, no. AUG, p. 157, Aug. 2016, doi: 10.3389/FGENE.2016.00157.

[277] J. Twycross, L. R. Band, M. J. Bennett, J. R. King, and N. Krasnogor, ‘Stochastic and deterministic multiscale models for systems biology: An auxin-transport case study’, *BMC Syst Biol*, vol. 4, no. 1, pp. 1–11, Mar. 2010, doi: 10.1186/1752-0509-4-34/TABLES/6.

[278] R. Stoof, A. Wood, and Á. Goñi-Moreno, ‘A Model for the Spatiotemporal Design of Gene Regulatory Circuits †’, *ACS Synth Biol*, vol. 8, no. 9, pp. 2007–2016, Sep. 2019, doi: 10.1021/ACSSYNBIO.9B00022/SUPPL_FILE/SB9B00022_SI_002.ZIP.

[279] S. M. Keating *et al.*, ‘SBML Level 3: an extensible format for the exchange and reuse of biological models’, *Mol Syst Biol*, vol. 16, no. 8, p. e9110, Aug. 2020, doi: 10.15252/MSB.20199110.

[280] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka, ‘LibSBML: an API Library for SBML’, *Bioinformatics*, vol. 24, no. 6, pp. 880–881, Mar. 2008, doi: 10.1093/BIOINFORMATICS/BTN051.

[281] H. Takizawa *et al.*, ‘LibSBMLSim: a reference implementation of fully functional SBML simulator’, *Bioinformatics*, vol. 29, no. 11, pp. 1474–1476, Jun. 2013, doi: 10.1093/BIOINFORMATICS/BTT157.

[282] S. Hoops *et al.*, ‘COPASI—a COmplex PAthway SImulator’, *Bioinformatics*, vol. 22, no. 24, pp. 3067–3074, Dec. 2006, doi: 10.1093/BIOINFORMATICS/BTL485.

[283] J. L. Wilmoth *et al.*, ‘A microfluidics and agent-based modeling framework for investigating spatial organization in bacterial colonies: The case of Pseudomonas aeruginosa and H1-Type VI secretion interactions’, *Front Microbiol*, vol. 9, no. FEB, p. 33, Feb. 2018, doi: 10.3389/FMICB.2018.00033/BIBTEX.

[284] S. Koshy-Chenthittayil, L. Archambault, D. Senthilkumar, R. Laubenbacher, P. Mendes, and A. Dongari-Bagtzoglou, ‘Agent Based Models of Polymicrobial Biofilms and the Microbiome—A Review’, *Microorganisms 2021, Vol. 9, Page 417*, vol. 9, no. 2, p. 417, Feb. 2021, doi: 10.3390/MICROORGANISMS9020417.

[285] T. E. Gorochowski, ‘Agent-based modelling in synthetic biology’, *Essays Biochem*, vol. 60, no. 4, p. 325, Nov. 2016, doi: 10.1042/EBC20160037.

[286] P. Almela, A. Justel, and A. Quesada, ‘Heterogeneity of Microbial Communities in Soils From the Antarctic Peninsula Region’, *Front Microbiol*, vol. 12, p. 280, Feb. 2021, doi: 10.3389/FMICB.2021.628792/BIBTEX.

[287] T. Pecht, A. C. Aschenbrenner, T. Ulas, and A. Succurro, ‘Modeling population heterogeneity from microbial communities to immune response in cells’, *Cellular and Molecular Life Sciences 2019 77:3*, vol. 77, no. 3, pp. 415–432, Nov. 2019, doi: 10.1007/S00018-019-03378-W.

[288] Y. Han and F. Zhang, ‘Heterogeneity coordinates bacterial multi-gene expression in single cells’, *PLoS Comput Biol*, vol. 16, no. 1, p. e1007643, 2020, doi: 10.1371/JOURNAL.PCBI.1007643.

[289] E. G. Sweeney *et al.*, ‘Agent-Based Modeling Demonstrates How Local Chemotactic Behavior Can Shape Biofilm Architecture’, *mSphere*, vol. 4, no. 3, Jun. 2019, doi: 10.1128/MSPHERE.00285-19/SUPPL_FILE/MSPHERE.00285-19-S0001.PDF.

[290] P. G. Jayathilake *et al.*, ‘A mechanistic Individual-based Model of microbial communities’, *PLoS One*, vol. 12, no. 8, p. e0181965, Aug. 2017, doi: 10.1371/JOURNAL.PONE.0181965.

[291] M. Gutiérrez, P. Gregorio-Godoy, G. Pérez Del Pulgar, L. E. Munoz, S. Sáez, and A. Rodríguez-Patón, ‘A New Improved and Extended Version of the Multicell Bacterial Simulator gro’, *ACS Synth Biol*, vol. 6, no. 8, pp. 1496–1508, Aug. 2017, doi: 10.1021/ACSSYNBIO.7B00003/SUPPL_FILE/SB7B00003_SI_003.ZIP.

[292] B. Li *et al.*, ‘NUFEB: A massively parallel simulator for individual-based modelling of microbial communities’, *PLoS Comput Biol*, vol. 15, no. 12, p. e1007125, 2019, doi: 10.1371/JOURNAL.PCBI.1007125.

[293] T. J. Lambert, ‘FPbase: a community-editable fluorescent protein database’, *Nature Methods 2019 16:4*, vol. 16, no. 4, pp. 277–278, Mar. 2019, doi: 10.1038/s41592-019-0352-8.

[294] E. Csibra and G.-B. Stan, ‘FPCountR: Absolute protein quantification using fluorescence measurements’, *bioRxiv*, p. 2021.12.06.471413, Sep. 2022, doi: 10.1101/2021.12.06.471413.

[295] J. D. Pedelacq and S. Cabantous, ‘Development and Applications of Superfolder and Split Fluorescent Protein Detection Systems in Biology’, *Int J Mol Sci*, vol. 20, no. 14, Jul. 2019, doi: 10.3390/IJMS20143479.

[296] F. Hussain *et al.*, ‘Engineered temperature compensation in a synthetic genetic clock’, *Proc Natl Acad Sci U S A*, vol. 111, no. 3, pp. 972–977, Jan. 2014, doi: 10.1073/PNAS.1316298111/SUPPL_FILE/SM02.MOV.

[297] N. D. Taylor *et al.*, ‘Engineering an allosteric transcription factor to respond to new ligands’, *Nature Methods 2015 13:2*, vol. 13, no. 2, pp. 177–183, Dec. 2015, doi: 10.1038/nmeth.3696.

[298] A. Veliz-Cuba *et al.*, ‘Sources of Variability in a Synthetic Gene Oscillator’, *PLoS Comput Biol*, vol. 11, no. 12, p. e1004674, 2015, doi: 10.1371/JOURNAL.PCBI.1004674.

[299] T. Knight, ‘Draft Standard for Biobrick Biological Parts’, 2007. https://dspace.mit.edu/bitstream/handle/1721.1/45138/BBFRFC10.txt (accessed Nov. 30, 2022).

[300] D. T. Gillespie, ‘A general method for numerically simulating the stochastic time evolution of coupled chemical reactions’, *J Comput Phys*, vol. 22, no. 4, pp. 403–434, Dec. 1976, doi: 10.1016/0021-9991(76)90041-3.

[301] F. Bergmann and lilijap, ‘copasi/basico: Release 0.7’, Nov. 2021, doi: 10.5281/ZENODO.5723018.

[302] M. v. Rouches, Y. Xu, L. B. G. Cortes, and G. Lambert, ‘A plasmid system with tunable copy number’, *Nature Communications 2022 13:1*, vol. 13, no. 1, pp. 1–12, Jul. 2022, doi: 10.1038/s41467-022-31422-0.

[303] ‘Part:pSB1C3 - parts.igem.org’. https://parts.igem.org/Part:pSB1C3 (accessed Nov. 30, 2022).

[304] K. Kipper *et al.*, ‘Structure-guided approach to site-specific fluorophore labeling of the lac repressor LacI’, *PLoS One*, vol. 13, no. 6, p. e0198416, Jun. 2018, doi: 10.1371/JOURNAL.PONE.0198416.

[305] W. Gilbert and B. Müller-Hill, ‘ISOLATION OF THE LAC REPRESSOR’, *Proceedings of the National Academy of Sciences*, vol. 56, no. 6, pp. 1891–1898, Dec. 1966, doi: 10.1073/PNAS.56.6.1891/ASSET/8BF8F836-7A1D-4292-A0D7-DB9C632E0B46/ASSETS/PNAS.56.6.1891.FP.PNG.

[306] N. A. Becker, J. P. Peters, and L. J. Maher, ‘Mechanism of promoter repression by Lac repressor–DNA loops’, *Nucleic Acids Res*, vol. 41, no. 1, p. 156, 2013, doi: 10.1093/NAR/GKS1011.

[307] B. sen Chen, C. Y. Hsu, and J. J. Liou, ‘Robust design of biological circuits: Evolutionary systems biology approach’, *J Biomed Biotechnol*, vol. 2011, 2011, doi: 10.1155/2011/304236.

[308] J. Xu and K. S. Matthews, ‘Flexibility in the Inducer Binding Region is Crucial for Allostery in the Escherichia coli Lactose Repressor’, *Biochemistry*, vol. 48, no. 22, p. 4988, Jun. 2009, doi: 10.1021/BI9002343.

[309] M. Stamatakis and N. v. Mantzaris, ‘Comparison of Deterministic and Stochastic Models of the lac Operon Genetic Network’, *Biophys J*, vol. 96, no. 3, pp. 887–906, Feb. 2009, doi: 10.1016/J.BPJ.2008.10.028.

[310] K. A. Goodson, Z. Wang, A. R. Haeusler, J. D. Kahn, and D. S. English, ‘LacI-DNA-IPTG loops: Equilibria among conformations by single-molecule FRET’,

Journal of Physical Chemistry B, vol. 117, no. 16, pp. 4713–4722, Apr. 2013, doi: 10.1021/JP308930C/SUPPL_FILE/JP308930C_SI_001.PDF.

[311] J. K. Rogers, C. D. Guzman, N. D. Taylor, S. Raman, K. Anderson, and G. M. Church, 'Synthetic biosensors for precise gene control and real-time monitoring of metabolites', *Nucleic Acids Res*, vol. 43, no. 15, p. 7648, Sep. 2015, doi: 10.1093/NAR/GKV616.

[312] A. Banerjee, I. Weaver, T. Thorsen, and R. Sarpeshkar, 'Bioelectronic measurement and feedback control of molecules in living cells', *Scientific Reports 2017 7:1*, vol. 7, no. 1, pp. 1–7, Oct. 2017, doi: 10.1038/s41598-017-12655-2.

[313] A. A. Mannan, D. Liu, F. Zhang, and D. A. Oyarzún, 'Fundamental Design Principles for Transcription-Factor-Based Metabolite Biosensors', *ACS Synth Biol*, vol. 6, no. 10, pp. 1851–1859, Oct. 2017, doi: 10.1021/ACSSYNBIO.7B00172/SUPPL_FILE/SB7B00172_SI_002.XLSX.

[314] P. Kiratisin, K. D. Tucker, and L. Passador, 'LasR, a Transcriptional Activator of Pseudomonas aeruginosa Virulence Genes, Functions as a Multimer', *J Bacteriol*, vol. 184, no. 17, p. 4912, Sep. 2002, doi: 10.1128/JB.184.17.4912-4919.2002.

[315] F. Wu, D. J. Menn, and X. Wang, 'Quorum-Sensing Crosstalk-Driven Synthetic Circuits: From Unimodality to Trimodality', *Chem Biol*, vol. 21, no. 12, pp. 1629–1638, Dec. 2014, doi: 10.1016/J.CHEMBIOL.2014.10.008.

[316] E. G. Suneby, L. R. Herndon, and T. L. Schneider, 'Pseudomonas aeruginosa LasR·DNA Binding Is Directly Inhibited by Quorum Sensing Antagonists', *ACS Infect Dis*, vol. 3, no. 3, pp. 183–189, Mar. 2017, doi: 10.1021/ACSINFECDIS.6B00163/SUPPL_FILE/ID6B00163_SI_001.PDF.

[317] J. R. D. Naylor, 'An integrative modelling framework for multicellular systems', 2019, Accessed: Nov. 29, 2022. [Online]. Available: http://theses.ncl.ac.uk/jspui/handle/10443/4796

[318] G. A. Bocharov and F. A. Rihan, 'Numerical modelling in biosciences using delay differential equations', *J Comput Appl Math*, vol. 125, no. 1–2, pp. 183–199, Dec. 2000, doi: 10.1016/S0377-0427(00)00468-4.

[319] W. Dubitzky, K. Kurowski, and B. Schott, *Large-Scale Computing*. Wiley, 2012.

[320] B. J. Swihart, B. Caffo, B. D. James, M. Strand, B. S. Schwartz, and N. M. Punjabi, 'Lasagna plots: A saucy alternative to spaghetti plots', *Epidemiology*, vol. 21, no. 5, p. 621, Sep. 2010, doi: 10.1097/EDE.0B013E3181E5B06A.

[321] R. Stoof and A. And´ And´angel Goñi-Moreno, ‘Modelling co-translational dimerisation for programmable nonlinearity in synthetic biology’, 2020, doi: 10.1101/2020.07.10.196667.

[322] R. J. K. Ngo, J. W. Yeoh, G. H. W. Fan, W. K. S. Loh, and C. L. Poh, ‘BMSS2: A Unified Database-Driven Modeling Tool for Systematic Biomodel Selection’, *ACS Synth Biol*, vol. 11, no. 8, pp. 2901–2906, Aug. 2022, doi: 10.1021/ACSSYNBIO.2C00123/ASSET/IMAGES/LARGE/SB2C00123_0002.JPEG.

[323] N. Anesiadis, W. R. Cluett, and R. Mahadevan, ‘Model-driven design based on sensitivity analysis for a synthetic biology application’, *Computer Aided Chemical Engineering*, vol. 29, pp. 1446–1450, Jan. 2011, doi: 10.1016/B978-0-444-54298-4.50068-4.

[324] Y. Boada, A. Vignoni, G. Reynoso-Meza, and J. Picó, ‘Parameter identification in synthetic biological circuits using multi-objective optimization’, *IFAC-PapersOnLine*, vol. 49, no. 26, pp. 77–82, Jan. 2016, doi: 10.1016/J.IFACOL.2016.12.106.

[325] J. Beal *et al.*, ‘Comparative analysis of three studies measuring fluorescence from engineered bacterial genetic constructs’, *PLoS One*, vol. 16, no. 6, p. e0252263, Jun. 2021, doi: 10.1371/JOURNAL.PONE.0252263.

[326] J. Beal *et al.*, ‘Quantification of bacterial fluorescence using independent calibrants’, *PLoS One*, vol. 13, no. 6, p. e0199432, Jun. 2018, doi: 10.1371/JOURNAL.PONE.0199432.

[327] H. M. Davey and M. K. Winson, ‘Using Flow Cytometry to Quantify Microbial Heterogeneity’, *Current Issues in Molecular Biology 2003, Vol. 5, Pages 9-15*, vol. 5, no. 1, pp. 9–15, Jan. 2003, doi: 10.21775/CIMB.005.009.

[328] B. D. Özel Duygan and J. R. van der Meer, ‘Recent advances in microbial community analysis from machine learning of multiparametric flow cytometry data’, *Curr Opin Biotechnol*, vol. 75, p. 102688, Jun. 2022, doi: 10.1016/J.COPBIO.2022.102688.

[329] A. L. Heins *et al.*, ‘Quantitative flow cytometry to understand population heterogeneity in response to changes in substrate availability in escherichia coli and saccharomyces cerevisiae chemostats’, *Front Bioeng Biotechnol*, vol. 7, no. AUG, p. 187, 2019, doi: 10.3389/FBIOE.2019.00187/BIBTEX.

[330] B. Wang, R. I. Kitney, N. Joly, and M. Buck, ‘Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology’, *Nature*

Communications 2011 2:1, vol. 2, no. 1, pp. 1–9, Oct. 2011, doi: 10.1038/ncomms1516.

[331] B. I. Davenport, J. Tica, and M. Isalan, ‘Reducing metabolic burden in the PACEmid evolver system by remastering high-copy phagemid vectors’, *Engineering Biology*, vol. 6, no. 2–3, pp. 50–61, Jun. 2022, doi: 10.1049/ENB2.12021.

[332] F. Ceroni, R. Algar, G. B. Stan, and T. Ellis, ‘Quantifying cellular capacity identifies gene expression designs with reduced burden’, *Nat Methods*, vol. 12, no. 5, pp. 415–418, Apr. 2015, doi: 10.1038/NMETH.3339.

[333] S. Chen, M. Larsson, R. C. Robinson, and S. L. Chen, ‘Direct and convenient measurement of plasmid stability in lab and clinical isolates of E. coli’, *Scientific Reports 2017 7:1*, vol. 7, no. 1, pp. 1–11, Jul. 2017, doi: 10.1038/s41598-017-05219-x.

[334] R. González-Cabaleiro, A. M. Mitchell, W. Smith, A. Wipat, and I. D. Ofiteru, ‘Heterogeneity in pure microbial systems: Experimental measurements and modeling’, *Front Microbiol*, vol. 8, no. SEP, p. 1813, Sep. 2017, doi: 10.3389/FMICB.2017.01813/TEXT.

[335] P. Du *et al.*, ‘De novo design of an intercellular signaling toolbox for multi-channel cell–cell communication and biological computation’, *Nature Communications 2020 11:1*, vol. 11, no. 1, pp. 1–11, Aug. 2020, doi: 10.1038/s41467-020-17993-w.

[336] R. Silva-Rocha and V. De Lorenzo, ‘Engineering multicellular logic in bacteria with metabolic wires’, *ACS Synth Biol*, vol. 3, no. 4, pp. 204–209, Apr. 2014, doi: 10.1021/SB400064Y/SUPPL_FILE/SB400064Y_SI_001.PDF.

[337] M. Thiry and D. Cingolani, ‘Optimizing scale-up fermentation processes’, *Trends Biotechnol*, vol. 20, no. 3, pp. 103–105, Mar. 2002, doi: 10.1016/S0167-7799(02)01913-3.

[338] D. W. Lendrem *et al.*, ‘Lost in space: design of experiments and scientific exploration in a Hogarth Universe’, *Drug Discov Today*, vol. 20, no. 11, pp. 1365–1371, Nov. 2015, doi: 10.1016/J.DRUDIS.2015.09.015.

[339] S. Swain, B. Ranjan Jena, S. Beg, S. Swain, B. R. Jena, and S. Beg, ‘Design of Experiments for the Development of Biotechnology Products’, *Design of Experiments for Pharmaceutical Product Development*, pp. 171–188, 2021, doi: 10.1007/978-981-33-4351-1_10.

[340] A. Berepiki, R. Kent, L. F. M. MacHado, and N. Dixon, ‘Development of High-Performance Whole Cell Biosensors Aided by Statistical Modeling’, *ACS Synth Biol*, vol. 9, no. 3, pp. 576–589, Mar. 2020, doi: 10.1021/ACSSYNBIO.9B00448/ASSET/IMAGES/LARGE/SB9B00448_0009.JPEG.

[341] C. C. Azubuike, M. G. Edwards, A. M. R. Gatehouse, and T. P. Howard, ‘Applying Statistical Design of Experiments To Understanding the Effect of Growth Medium Components on Cupriavidus necator H16 Growth’, *Appl Environ Microbiol*, vol. 86, no. 17, Sep. 2020, doi: 10.1128/AEM.00705-20.

[342] A. J. Lopatkin and J. J. Collins, ‘Predictive biology: modelling, understanding and harnessing microbial complexity’, *Nature Reviews Microbiology 2020 18:9*, vol. 18, no. 9, pp. 507–520, May 2020, doi: 10.1038/s41579-020-0372-5.

[343] L. Hocharoen, S. Noppiboon, and P. Kitsubun, ‘Process Characterization by Definitive Screening Design Approach on DNA Vaccine Production’, *Front Bioeng Biotechnol*, vol. 8, Oct. 2020, doi: 10.3389/FBIOE.2020.574809.

[344] M. L. Jones, D. W. Rivett, A. Pascual-Garria, and T. Bell, ‘Relationships between community composition, productivity and invasion resistance in semi-natural bacterial microcosms’, *Elife*, vol. 10, Oct. 2021, doi: 10.7554/ELIFE.71811.

[345] T. Bell, J. A. Newman, B. W. Silverman, S. L. Turner, and A. K. Lilley, ‘The contribution of species richness and composition to bacterial services’, *Nature*, vol. 436, no. 7054, pp. 1157–1160, Aug. 2005, doi: 10.1038/NATURE03891.

[346] D. R. Amor, C. Ratzke, and J. Gore, ‘Transient invaders can induce shifts between alternative stable states of microbial communities’, *Sci Adv*, vol. 6, no. 8, 2020, doi: 10.1126/SCIADV.AAY8676.

[347] D. W. Rivett and T. Bell, ‘Abundance determines the functional role of bacterial phylotypes in complex communities’, *Nat Microbiol*, vol. 3, no. 7, pp. 767–772, Jul. 2018, doi: 10.1038/S41564-018-0180-0.

[348] T. Chung, D. L. Weller, and J. Kovac, ‘The Composition of Microbial Communities in Six Streams, and Its Association With Environmental Conditions, and Foodborne Pathogen Isolation’, *Front Microbiol*, vol. 11, p. 1757, Jul. 2020, doi: 10.3389/FMICB.2020.01757/BIBTEX.

[349] J. Andres Martinez *et al.*, ‘Controlling microbial co-culture based on substrate pulsing can lead to stability through differential fitness advantages’, 2022, doi: 10.1101/2022.02.18.480836.

[350] H. Sassi, T. M. Nguyen, S. Telek, G. Gosset, A. Grünberger, and F. Delvigne, ‘Segregostat: a novel concept to control phenotypic diversification dynamics on the example of Gram-negative bacteria’, *Microb Biotechnol*, vol. 12, no. 5, pp. 1064–1075, Sep. 2019, doi: 10.1111/1751-7915.13442.

[351] C. H. Gao, H. Cao, P. Cai, and S. J. Sørensen, ‘The initial inoculation ratio regulates bacterial coculture interactions and metabolic capacity’, *The ISME Journal 2020 15:1*, vol. 15, no. 1, pp. 29–40, Sep. 2020, doi: 10.1038/s41396-020-00751-7.

[352] R. V. Kapoore, G. Padmaperuma, S. Maneein, and S. Vaidyanathan, ‘Co-culturing microbial consortia: approaches for applications in biomanufacturing and bioprocessing’, *https://doi.org/10.1080/07388551.2021.1921691*, vol. 42, no. 1, pp. 46–72, 2021, doi: 10.1080/07388551.2021.1921691.

[353] Q. M. Dudley, K. C. Anderson, and M. C. Jewett, ‘Cell-Free Mixing of Escherichia coli Crude Extracts to Prototype and Rationally Engineer High-Titer Mevalonate Synthesis’, *ACS Synth Biol*, vol. 5, no. 12, pp. 1578–1588, Dec. 2016, doi: 10.1021/ACSSYNBIO.6B00154/SUPPL_FILE/SB6B00154_SI_001.PDF.

[354] X. Liu, L. Li, and G.-R. Zhao, ‘Systems Metabolic Engineering of Escherichia coli Coculture for De Novo Production of Genistein’, *Cite This: ACS Synth. Biol*, vol. 2022, pp. 1746–1757, 2022, doi: 10.1021/acssynbio.1c00590.

[355] I. M. Kasli, O. R. T. Thomas, and T. W. Overton, ‘Use of a design of experiments approach to optimise production of a recombinant antibody fragment in the periplasm of Escherichia coli: selection of signal peptide and optimal growth conditions’, *AMB Express*, vol. 9, no. 1, pp. 1–14, Dec. 2019, doi: 10.1186/S13568-018-0727-8/FIGURES/8.

[356] C. Singleton, J. Gilman, J. Rollit, K. Zhang, D. A. Parker, and J. Love, ‘A design of experiments approach for the rapid formulation of a chemically defined medium for metabolic profiling of industrially important microbes’, *PLoS One*, vol. 14, no. 6, p. e0218208, Jun. 2019, doi: 10.1371/JOURNAL.PONE.0218208.

[357] A. Goñi-Moreno, M. Amos, and F. de la Cruz, ‘Multicellular Computing Using Conjugation for Wiring’, *PLoS One*, vol. 8, no. 6, p. e65986, Jun. 2013, doi: 10.1371/JOURNAL.PONE.0065986.

[358] D. Oesterhelt, ‘Bacteriorhodopsin as an Example of a Light-Driven Proton Pump’, *Angewandte Chemie International Edition in English*, vol. 15, no. 1, pp. 17–24, Jan. 1976, doi: 10.1002/ANIE.197600171.

[359] L. B. Motta-Mena *et al.*, ‘An optogenetic gene expression system with rapid activation and deactivation kinetics’, *Nat Chem Biol*, vol. 10, no. 3, p. 196, 2014, doi: 10.1038/NCHEMBIO.1430.

[360] P. Jayaraman, K. Devarajan, T. K. Chua, H. Zhang, E. Gunawan, and C. L. Poh, ‘Blue light-mediated transcriptional activation and repression of gene expression in bacteria’, *Nucleic Acids Res*, vol. 44, no. 14, pp. 6994–7005, Aug. 2016, doi: 10.1093/NAR/GKW548.

[361] M. Mansouri and M. Fussenegger, ‘Synthetic biology-based optogenetic approaches to control therapeutic designer cells’, *Curr Opin Syst Biol*, vol. 28, p. 100396, Dec. 2021, doi: 10.1016/J.COISB.2021.100396.

[362] E. J. Olson and J. J. Tabor, ‘Optogenetic characterization methods overcome key challenges in synthetic and systems biology’, *Nature Chemical Biology 2014 10:7*, vol. 10, no. 7, pp. 502–511, Jun. 2014, doi: 10.1038/nchembio.1559.

[363] A. Baumschlager and M. Khammash, ‘Synthetic Biological Approaches for Optogenetics and Tools for Transcriptional Light-Control in Bacteria’, *Adv Biol*, vol. 5, no. 5, p. 2000256, May 2021, doi: 10.1002/ADBI.202000256.

[364] E. Brodl, A. Winkler, and P. Macheroux, ‘Molecular Mechanisms of Bacterial Bioluminescence’, *Comput Struct Biotechnol J*, vol. 16, p. 551, Jan. 2018, doi: 10.1016/J.CSBJ.2018.11.003.

[365] A. J. Syed and J. C. Anderson, ‘Applications of bioluminescence in biotechnology and beyond’, *Chem Soc Rev*, vol. 50, no. 9, pp. 5668–5705, May 2021, doi: 10.1039/D0CS01492C.

[366] T. Kaku, K. Sugiura, T. Entani, K. Osabe, and T. Nagai, ‘Enhanced brightness of bacterial luciferase by bioluminescence resonance energy transfer’, *Scientific Reports 2021 11:1*, vol. 11, no. 1, pp. 1–10, Jul. 2021, doi: 10.1038/s41598-021-94551-4.

[367] J. K. Tung, K. Berglund, C.-A. Gutekunst, U. Hochgeschwender, and R. E. Gross, ‘Bioluminescence imaging in live cells and animals’, *Neurophotonics*, vol. 3, no. 2, p. 1, Apr. 2016, doi: 10.1117/1.NPH.3.2.025001.

[368] S. Hennig, G. Rödel, and K. Ostermann, ‘Artificial cell-cell communication as an emerging tool in synthetic biology applications’, *J Biol Eng*, vol. 9, no. 1, pp. 1–12, Aug. 2015, doi: 10.1186/S13036-015-0011-2/FIGURES/2.

[369] W. Bacchus and M. Fussenegger, ‘Engineering of synthetic intercellular communication systems’, *Metab Eng*, vol. 16, no. 1, pp. 33–41, 2013, doi: 10.1016/J.YMBEN.2012.12.001.

[370] T. Chakraborty and S. v. Wegner, ‘Cell to Cell Signaling through Light in Artificial Cell Communities: Glowing Predator Lures Prey’, *ACS Nano*, vol. 15, no. 6, pp. 9434–9444, Jun. 2021, doi: 10.1021/ACSNANO.1C01600/SUPPL_FILE/NN1C01600_SI_007.AVI.

[371] M. Sureda-Vives and K. S. Sarkisyan, ‘Bioluminescence-Driven Optogenetics’, *Life*, vol. 10, no. 12, pp. 1–11, Dec. 2020, doi: 10.3390/LIFE10120318.

[372] ‘Team:Tokyo-NoKoGen/Project/overview - 2012.igem.org’. https://2012.igem.org/Team:Tokyo-NoKoGen/Project/overview (accessed Nov. 30, 2022).

[373] ‘Team:Peking/Project/Communication/Results - 2012.igem.org’. https://2012.igem.org/Team:Peking/Project/Communication/Results (accessed Nov. 30, 2022).

[374] K. P. Szymula, M. S. Magaraci, M. Patterson, A. Clark, S. G. Mannickarottu, and B. Y. Chow, ‘An Open-Source Plate Reader’, *Biochemistry*, vol. 58, no. 6, pp. 468–473, Feb. 2019, doi: 10.1021/ACS.BIOCHEM.8B00952/SUPPL_FILE/BI8B00952_SI_002.PDF.

[375] T. Danino, O. Mondragón-Palomino, L. Tsimring, and J. Hasty, ‘A synchronized quorum of genetic clocks’, *Nature*, vol. 463, no. 7279, p. 326, Jan. 2010, doi: 10.1038/NATURE08753.

[376] M. Tehranirokh, A. Z. Kouzani, P. S. Francis, and J. R. Kanwar, ‘Microfluidic devices for cell cultivation and proliferation’, *Biomicrofluidics*, vol. 7, no. 5, p. 051502, Oct. 2013, doi: 10.1063/1.4826935.

[377] X. Ouyang, J. Hoeksma, R. J. M. Lubbers, T. K. Siersma, L. W. Hamoen, and J. den Hertog, ‘Classification of antimicrobial mechanism of action using dynamic bacterial morphology imaging’, *Scientific Reports 2022 12:1*, vol. 12, no. 1, pp. 1–12, Jul. 2022, doi: 10.1038/s41598-022-15405-1.

[378] F. Wong *et al.*, ‘Cytoplasmic condensation induced by membrane damage is associated with antibiotic lethality’, *Nature Communications 2021 12:1*, vol. 12, no. 1, pp. 1–15, Apr. 2021, doi: 10.1038/s41467-021-22485-6.

[379] N. Kylilis, Z. A. Tuza, G.-B. Stan, and K. M. Polizzi, ‘Tools for engineering coordinated system behaviour in synthetic microbial consortia’, *Nat Commun*, vol. 9, no. 1, p. 2677, Dec. 2018, doi: 10.1038/s41467-018-05046-2.

[380] S. Regot *et al.*, ‘Distributed biological computation with multicellular engineered networks’, *Nature*, vol. 469, no. 7329, pp. 207–211, Jan. 2011, doi: 10.1038/nature09679.

[381] A. Urrios, E. Gonzalez-Flo, D. Canadell, E. de Nadal, J. Macia, and F. Posas, ‘Plug-and-Play Multicellular Circuits with Time-Dependent Dynamic Responses’, *ACS Synth Biol*, vol. 7, no. 4, pp. 1095–1104, Apr. 2018, doi: 10.1021/acssynbio.7b00463.

[382] A. Goñi-Moreno, M. Amos, and F. de la Cruz, ‘Multicellular Computing Using Conjugation for Wiring’, *PLoS One*, vol. 8, no. 6, p. e65986, Jun. 2013, doi: 10.1371/journal.pone.0065986.

[383] C. Gregor, K. C. Gwosch, S. J. Sahl, and S. W. Hell, ‘Strongly enhanced bacterial bioluminescence with the ilux operon for single-cell imaging’, *Proc Natl Acad Sci U S A*, vol. 115, no. 5, pp. 962–967, Jan. 2018, doi: 10.1073/PNAS.1715946115/SUPPL_FILE/PNAS.1715946115.SM05.MOV.

[384] H. Kobayashi, L. P. Picard, A. M. Schönegge, and M. Bouvier, ‘Bioluminescence resonance energy transfer–based imaging of protein–protein interactions in living cells’, *Nature Protocols 2019 14:4*, vol. 14, no. 4, pp. 1084–1107, Mar. 2019, doi: 10.1038/s41596-019-0129-7.

[385] F. Liu, J. Mao, T. Lu, and Q. Hua, ‘Synthetic, Context-Dependent Microbial Consortium of Predator and Prey’, *ACS Synth Biol*, vol. 8, no. 8, pp. 1713–1722, Aug. 2019, doi: 10.1021/ACSSYNBIO.9B00110/SUPPL_FILE/SB9B00110_SI_001.PDF.