IMPERIAL COLLEGE LONDON, DIVISION OF MOLECULAR BIOSCIENCES, CENTRE FOR SYNTHETIC BIOLOGY AND INNOVATION

Tools and Technologies for Enabling Characterisation in Synthetic Biology

Jonathan Charles Smith

19/07/2016

Submitted for the Degree of Doctor of Philosophy

Declaration of Originality

I hereby declare that this project was entirely my own work and that any additional sources of information have been duly cited.

Copyright Declaration

Table of Contents

1 Table of Figures

2 Table of Tables

3 List of Abbreviations

Abbreviation	Meaning
AHL	Acyl-Homoserine Lactone
APD	Avalanche Photodiode Detector
API	Application Programming Interface
CAD	Computer Aided Design
CCD	Charge Coupled Device
DICOM	Digital Imaging and Communications in Medicine
DICOM-SB	Digital Imaging and Communications in Medicine Synthetic Biology
DNA	Deoxyribonucleic Acid
DPAD	Data Processing Analysis and Display
emCCD	Electron Multiplying Charge Coupled Device
EDTA	Ethylenediaminetetraacetic acid
FIFO	First In First Out
GFP	Green Fluorescent Protein
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IVC	In Vitro Comparmentalisation
IVTT	In Vitro Transcription Translation
JDBC	Java Database Connectivity
LB	Lysogeny Broth
LIMS	Laboratory Information Management System
MIAME	Minimum Information about a Microarray Experiment
MIBBI	Minimum Information for Biological and Biomedical Investigations
MINSEQE	Minimum Information about a high-throughput SEQuencing Experiment
mRNA	Messenger Ribonucleic Acid
OD	Optical Density
PCR	Polymerase Chain Reaction
PDMS	Polydimethyl Siloxane
RBS	Ribosome Binding Site

ReST	Representational State Transfer
rpm	revolutions per minute
SBOL	Systems Biology Open Language
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TAE	Tris base, acetic acid and EDTA
TCP	Transmission Control Protocol
UV	Ultra Violet

4 Abstract

Synthetic Biology represents a movement to utilise biological organisms for novel applications through the use of rigorous engineering principles. These principles rely on a solid and well versed understanding of the underlying biological components and functions (relevant to the application). In order to achieve this understanding, reliable behavioural and contextual information is required (more commonly known as characterisation data). Focussing on lowering the barrier of entry for current research facilities to regularly and easily perform characterisation assays will directly improve the communal knowledge base for Synthetic Biology and enable the further application of rational engineering principles.

Whilst characterisation remains a fundamental principle for Synthetic Biology research, the high time costs, subjective measurement protocols, and ambiguous data analysis specifications, deter regular performance of characterisation assays. Vitally, this prevents the valid application of many of the key Synthetic Biology processes that have been derived to improve research yield (with regards to solving application problems) and directly prevent the intended goal of addressing the *ad hoc* nature of modern research from being realised.

Designing new technologies and tools to facilitate rapid 'hands off' characterisation assays for research facilities will improve the uptake of characterisation within the research pipeline. To achieve this two core problem areas have been identified that limit current characterisation attempts in conventional research. Therefore, it was the primary aim of this investigation to overcome these two core problems to promote regular characterisation.

The first issue identified as preventing the regular use of characterisation assays was the user-intensive methodologies and technologies available to researchers. There is currently no standardised characterisation equipment for assaying samples and the methodologies are heavily dependent on the researcher and their application for successful and complete characterisation. This study proposed a novel high throughput solution to the characterisation problem that was capable of low cost,

concurrent, and rapid characterisation of simple biological DNA elements. By combining *in vitro* transcription-translation with microfluidics a potent solution to the characterisation problem was proposed. By utilising a completely *in vitro* approach along with excellent control abilities of microfluidic technologies, a prototype platform for high throughput characterisation was developed.

The second issue identified was the lack of flexible, versatile software designed specifically for the data handling needs that are quickly arising within the characterisation speciality. The lack of general solutions in this area is problematic because of the increasing amount of data that is both required and generated for the characterisation output to be considered as rigorous and of value. To alleviate this issue a novel framework for laboratory data handling was developed that employs a plugin strategy for data submission and analysis. Employing a plugin strategy improves the shelf life of data handling software by allowing it to grow with the needs of the speciality. Another advantage to this strategy is the increased ability for well documented processing and analysis standards to arise that are available for all researchers. Finally, the software provided a powerful and flexible data storage schema that allowed all currently conceivable characterisation data types to be stored in a well-documented manner.

The two solutions identified within this study increase the amount of enabling tools and technologies available to researchers within Synthetic Biology, which in turn will increase the uptake of regular characterisation. Consequently, this will potentially improve the lateral transfer of knowledge between research projects and reduce the need to perform *ad hoc* experiments to investigate facets of the fundamental biological components being utilised.

5 Background and Introduction

5.1 Synthetic Biology

Recent efforts to standardise Life Sciences by applying an engineering paradigm (Serrano, 2007) have led to the emergence of the field of Synthetic Biology, which aims to design and manufacture novel or reconstituted biological constructs with functional applications as an end point (Royal Academy of Engineering, (2009)). To facilitate this, the field has embraced three core tenets derived from the engineering paradigm: abstraction, decoupling and standardisation. Abstraction aims to break down the complexity of biological systems to define discrete functional components, thereby increasing specialisation within these areas and decreasing the need to understand areas other than those of interest to the researcher. Decoupling reduces the complexity of achieving applications by breaking down the investigations into discrete steps, also generating areas of specialisation related to the type of investigative step being performed. Finally, standardisation allows increased understanding through the use of common tools and methods and gives rise to common annotations for biological functions and activity. A comprehensive review of these can be found in the excellent review by Andrianantoandro *et al*. (Andrianantoandro et al., 2006). Briefly, these tenets aim to reduce the *ad hoc* nature of biotechnology research through systematic design and rational engineering by improving the transfer of information between research investigations, decreasing the required knowledge to operate in more complex applications, and increasing specialisation within discrete areas of investigation. These tenets have hastened the arrival of novel applications for the field, including work on the production of a synthetic microbe (Gibson et al., 2010) and the industrialisation of artemisinin production for use as an antimalarial drug (Ro et al., 2006). Previously such applications would have taken many more research hours to achieve but by reducing the *ad hoc* nature of the research and increasing the lateral transfer of knowledge *post hoc* applications can be realised at increased rates without re-inventing the wheel.

The seminal work performed by Elowitz *et al.* (Elowitz and Leibler, 2000) and Gardner *et al.* (Gardner et al., 2000) in designing synthetic gene networks that emulate computational circuits has established the applicability of the engineering paradigm to biological cells. By identifying components with discrete functions and combining them using rational, computationally aided, design principles these researchers have achieved an oscillatory biological circuit and a biological toggle switch respectively. Key to their success is both the abstraction of the complexity of the biological processes into separate functional layers and the decoupling of the normal research workflow into discrete investigative steps.

To reduce the complexity of engineering biological organisms, a classification system for biological components has been developed. By applying the tenet of abstraction to conventional biological understanding the concept of biological parts, devices, and systems has arisen (the reader is referred a summary of this by Drew Endy (Endy, 2005)). Specifically, biological parts are responsible for a core biological function (e.g. a promoter), a device is a combination of biological function parts that may result in a directly measurable outcome (e.g. an expression vector), and a system utilises multiple biological devices to achieve a function application. Furthermore, a common concept within Synthetic Biology is to abstract everything that is not of interest to the investigation (e.g. the non-synthetic cell circuitry and behaviours) and consider it to be part of the expression 'chassis'. Decoupling the chassis and the synthetic circuitry conventionally represents a core aim of an investigation as it prevents confounding effects on the synthetic circuitry.

In engineering, the application of a cyclical workflow (known as the engineering cycle, Figure 1) is accepted as a technique that allows for the iterative improvement of experiment designs. These improvements can arise through a variety of mechanisms, including, but not limited to: the progressive improvement of associated technologies, the improvement of human experience, and an increase in knowledge about the system (Kitney et al., 2007). Separating and cyclising the steps within a biological investigation (as denoted in the engineering cycle) allows for the following potential achievements: a reduction in the need for *ad hoc* tool and method development and an increase in the specialisation of researchers. Specialisation of expertise is considered, by convention, to lead directly to an increase in the development of niche tools and technologies that in turn, will iteratively improve upon the area of expertise.

Applying the engineering paradigm to biology has not only created tools and technologies within the investigative steps highlighted in Figure 1, but is helping highlight the need for tools and technologies at the interface of these steps. The capabilities of core enabling technologies such as DNA synthesis have increased dramatically as it attempts to match the increasing demands for the rapid prototyping of biological parts. As testament to this, the complete synthesis of the *M. mycoides* (Gibson et al., 2010) emphasises the sheer scale of DNA synthesis and assembly that is now possible with these technologies. The development of tools within the computational step of the engineering workflow has also increased dramatically. As standards for annotating biological parts (Galdzicki et al., 2014, Roehner et al., 2014) and modelling their behaviour (Hill et al., 2008) have been identified and taken up into the field, it has given rise to powerful bioinformatics tools that greatly decrease the time required for selecting and determining the optimal path to achieve desired applications. Conventionally, these tools are known as computer aided design (CAD) tools and they generally facilitate the modelling of a synthetic system's behaviour by selecting the biological parts that constitute the system. These CAD tools rely on libraries of biological information such as collections of biological parts that have been tied to any associated information pertaining to the part's performance under specific environmental conditions. Generally, the acquisition of information associated with a part is known as characterisation. More specifically, these computational tools rely on detailed behaviour profiles or specifications in order to model how these parts are likely to behave *in* vivo (Hill *et al.* 2008).

Repositories of biological parts and their coupled information (as required by the tools introduced above) have therefore arisen in turn and cyclically allow CAD tools to flourish by increasing their ability to predict the behaviour of devices *a priori* (Macdonald *et al.* 2011). An example of biological parts (with associated behavioural information) is the BioBricks^{TM} Registry of Standard Biological Parts (www.partsregistry.org) which stores the physical DNA of a biological part and its characterisation information. As such, the BioBricks^{TM} Registry represents a foundational technology that enables Synthetic Biology and its tools to progress and reduce the need for *ad hoc* research.

One of the more essential characteristics of an effective biological part repository is the requirement that a biological part and its associated characterisation information be obtained in a consistent fashion to promote interoperability and allow both researchers and CAD tools to inspect and use the information in a meaningful manner. It is at the interfaces between researcher and archivist (the conceptual curator of a biological parts repository) that detailed standards for both the biological and information based components are required. For the previously provided example of the BioBricks^{TM} repository, a standard exists for biological part submissions whereby parts must be flanked with specific prefix and suffix DNA sequences to facilitate standardised assembly methods. One of the standardised assembly method used by the BioBricks^{TM} repository is known as RFC10 (http://parts.igem.org/Help:Standards/Assembly/RFC10) and it allows for rapid biological part assembly with known scar sequences insulating each part. The diagram found in Figure 2 provides an overview of the most basic assembly method.

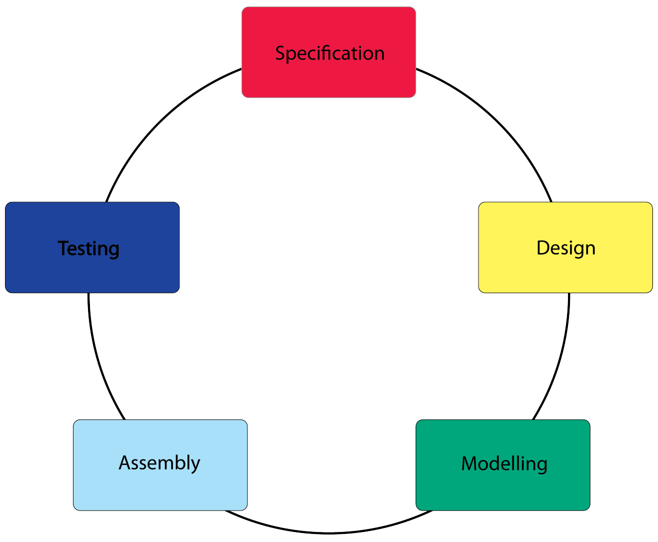


Figure 1: The standard engineering cycle used in fields such as software engineering, aeronatuics and mechanical engineering.

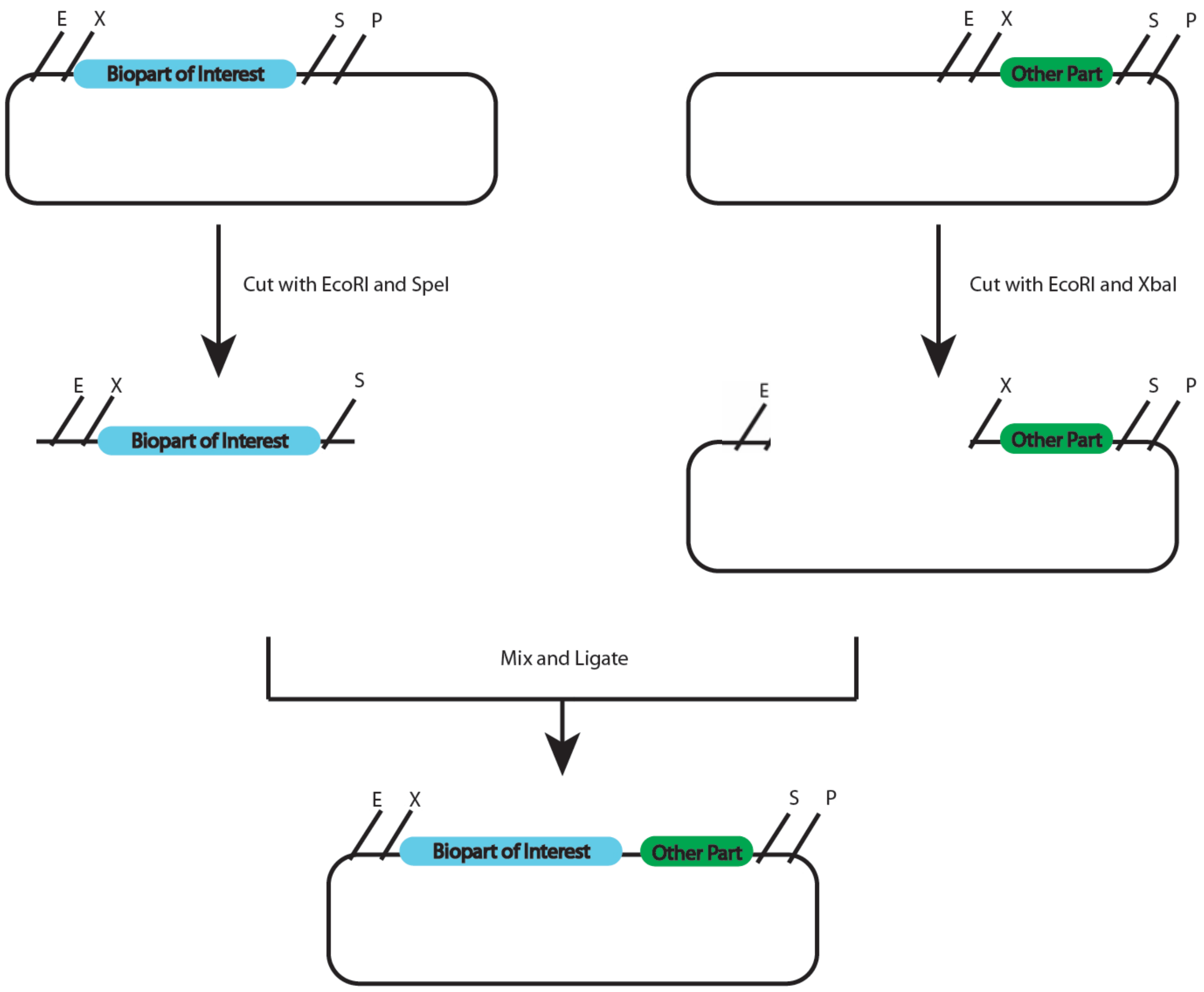


Figure 2: The BioBrick^{TM} assembly method (RFC10). Assembly using the above method helps reform a plasmid that conforms to the normal BioBrick^{TM} standard. Assembly is performed by cutting the two plasmids independently with the indicated enzymes and after purification of the relevant parts the two samples can be ligated into a unified plasmid. E, X, S and P correspond to restriction enzyme sites for EcoRI, SpeI, XbaI and PstI.

5.2 Characterisation in Synthetic Biology

As intimated above, underpinning the three core tenets of Synthetic Biology is the concept of characterisation. Characterisation is herein identified as the description of key attributes and or behaviours of the biological components that constitute the sample under investigation within fixed environmental parameters. Characterisation facilitates the successful implementation of both the engineering cycle and the core tenets, as it generates the required information about biological component performance and the effect of the biological components on the overall system. Figure 3 is a schematic representation of the information provided by characterisation assays. Additionally, characterisation is required to help ensure a system's robustness and predictability as well as to define performance targets and thresholds for the activation of a system, as each of the components' behaviours must be sufficiently defined to predict all outcomes in the face of nature's complexity (Endy, 2005, Canton et al., 2008, Kelly et al., 2009, Arkin, 2008, Marguet et al., 2007, Heinemann and Panke, 2006).

In turn, detailed characterisation information can help promote rapid prototyping of complex biological circuits (where target application specifications exist), by using CAD tools to promote *in silico* modelling of biological circuit behaviour. Using CAD tools facilitates rapid assessment of a design's viability without the arduous process of trial and error and optimisation (MacDonald et al., 2011).

The value of characterisation data lies not only in its ability to promote modelling of potential solutions to an application, but also serves as a foundation for the lateral transfer of knowledge between research projects. Collaborative characterisation across all institutions performing Synthetic Biology research would greatly reduce the amount of overlap in the early stages of research and greatly increase the speed at which solutions to application problems are realised.

Characterisation is a concept and technique that can be applied to many of the areas of biological investigation. In point of fact a great deal of work has been done to quantify and understand a variety of these areas, such as the behaviour of enzyme catalytic activity, but for the purposes of this investigation, future references to characterisation refer specifically to the quantification and description of the effects of a biological part's DNA sequence at the unit, functional, and system levels within a predetermined environment.

The first hurdle in establishing viable characterisation strategies for the field has been defining the behaviours that need to be quantified. Previous work established many of the necessary measurements for characterisation data to be considered sufficiently comprehensive, most of which can be found in the seminal work by Canton *et al.* (Canton et al., 2008). Canton *et al.* also promote the use of biological part datasheets (conventionally used by other engineering disciplines) to describe biological part behaviour. Specifically, these datasheets should define the biological part as well as quantify its behaviour under described environmental conditions.

The second hurdle to establishing a standardised characterisation toolset has been the characterisation methodology itself. Canton *et al.* were the first group to define a characterisation methodology. Predominantly their aim is to absolutely quantify the desirable behaviours of the biological part in question, generating the aforementioned datasheets. More recently, in the study by Kelly *et al*. (Kelly et al., 2009) the authors hypothesise that reducing inter-lab environmental variations would increase the versatility of Canton *et al.*'s characterisation method. To accomplish this, they describe biological part activity in relative units (as opposed to absolute units) by normalising the characterisation data using a standard reference part's activity. In the case of Kelly *et al.* they are attempting to characterise promoters and so have chosen to use a standard reference promoter to normalise the expression levels. Kelly *et al*.'s conclusions emphasise that any data acquired by a characterisation platform should be both versatile and robust in its acquisition, such that cross-experiment variation is reduced and the statistical confidence in observed behaviours is increased.

Using the absolute quantification method described by Canton *et al.*, calibration curves for fluorescence and OD to reporter units are required which can be time consuming to generate; however, such results can be used directly in mathematical modelling. The relative quantification method is considered more cross-experiment friendly, as effects in the characterisation data due to environmental variations are normalised out. The relative quantification method has been the most widely cited and has been used in *Escherichia coli* (Kelly et al., 2009, Canton et al., 2008), cyanobacteria (Huang et al., 2010), and yeast (Blount et al., 2012), and is considered sufficient for computational modelling (Ellis et al., 2009).

The most common types of characterisation to date involve biological parts that regulate the amount of target protein produced and green fluorescent protein (GFP) is generally used as the reporter protein. Characterisation methods not only attempt to minimise environmental variation, but they also attempt to reduce the number of confounding variables from the expression host. To achieve this, minimal expression plasmids or vectors are created for the sole purpose of characterisation that attempt to reduce context dependency of the biological part being characterised. The expression host (e.g. the bacteria responsible for producing the reporter) is selected using criteria that aim to reduce the amount of background interference the expression media will cause (i.e. no interfering products or intermediates).

Despite this, research into characterisation protocols and technologies is limited at best (Canton et al., 2008, Kelly et al., 2009, Huang et al., 2010, Zhang et al., 2007), and published work on Synthetic Biology's applications, exhibits only a small degree of component characterisation (Gibson et al., 2010, Ro et al., 2006, Sinha et al., 2010, Beisel and Smolke, 2009, Stricker et al., 2008, Win and Smolke, 2007, Anderson et al., 2007, Alper et al., 2005). Much of the more recent research does little to address the lack of foundational developments (either in component characterisation or method standardisation), which are essential in realising Synthetic Biology's promise. This is largely due the time requirement and ambiguous nature of characterisation work, which is especially emphasised, as an example, by the variety of *E. coli* strains that have been used as an expression medium (Canton et al., 2008, Kelly et al., 2009, Lee et al., 2011, Singh et al., 2012, Pasotti et al., 2012) for bacterial biological part characterisation.

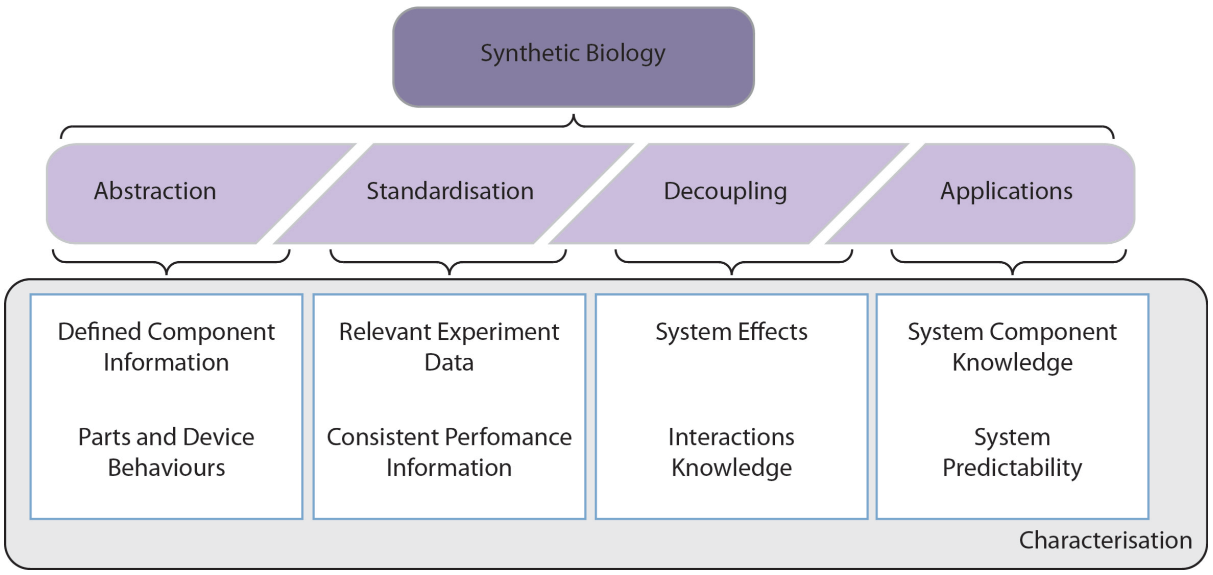


Figure 3: The purview of characterisation in Synthetic Biology. Showing how characterisation underlies the fundamental tenets of Synthetic Biology and is useful in all potential applications.

5.3 Creating Enabling Tools and Technologies for DNA-Based Biological Part Characterisation in Synthetic Biology

As intimated above, very little characterisation of biological parts is being performed in Synthetic Biology, as it requires a high investment of research hours for very little obvious potential gain within a normal research environment. However, the potential gain increases significantly when used in conjunction with the engineering cycle, as CAD tools can make direct use of the information to accelerate research. Specifically, the small amount of potential gain arises because researchers tend only to concern themselves with their current on-going project, but this mind set does little to reduce the *ad hoc* nature of research and provides minimal lateral transfer of knowledge to *post hoc* investigations. To improve the uptake of characterisation throughout Synthetic Biology research, it is essential to create tools and technologies that will reduce the number of hours it requires. Similar to movements in DNA Synthesis, a reduction in the cost (here the cost is man hours) will greatly improve its utility and therefore uptake across laboratories. The high time cost of characterisation work can be attributed to two core problems: firstly the number of man hours required to characterise a single sample is very high, and secondly the handling of the large volume of data that is produced during characterisation assays.

To overcome the first problem, high throughput technologies have to be designed specifically for the characterisation of biological parts. These technologies should aim to reduce the amount of user input required to perform characterisation by taking a ‘plug and play’ approach. An approach like this would allow researchers to rapidly and autonomously generate characterisation data. The advantage of such an approach is that researchers would be capable of determining if their biological parts or devices are sufficient in solving the application that they are targeting, prior to rounds of optimisation and attenuation.

The second problem can be solved computationally by designing a framework for handling characterisation data. The framework could reduce the need for manual data curation, increase the speed at which data is analysed and even display it in a meaningful manner. Furthermore, computational tools would aid in defining a common storage and analysis toolset that would reduce the need for *ad hoc* analysis techniques to be used.

The research described herein proposed a solution to each of the problems that serve as a bottleneck for the uptake of characterisation in Synthetic Biology research. The proposed solution to the first problem was to design a high throughput technology that fills the gap in foundational characterisation technologies, see section 5.4, and secondly a novel computational framework for the handling of characterisation data, see section 5.5. To overcome the lack of enabling technologies for characterisation work a variety of disciplines need to be coordinated. Whilst the biological methodologies for characterisation are somewhat established, the methodologies for observing, recording and analysing the characterisation process are still extremely lab dependent. Therefore, establishing common tools and technologies will serve to standardise these methodologies and hopefully facilitate the uptake of characterisation into the research pipeline.

5.4 A High Throughput DNA Part Expression Characterisation Platform

5.4.1 Aims for the Investigation

The work undertaken herein represents efforts to standardise, automate and improve upon existing characterisation methods by engineering a novel foundational platform for characterisation and rapid design prototyping. A high throughput, high detail, and automated characterisation platform can be achieved by adopting a multidisciplinary approach that integrates biological characterisation assays (Canton et al., 2008, Kelly et al., 2009, Chappell et al., 2013) within the platform. By adopting this approach, solutions already present within each of the discrete disciplines can be combined, producing a cohesive and well understood solution to the characterisation problem.

The first milestone established for the platform was to characterise the performance of individual biological parts, namely, promoters, ribosome binding sites (RBSs), and terminators. Promoters are of particular importance because of their ubiquitous presence in Synthetic Biology research (Danino et al., 2010, Fussenegger, 2010, Kötter et al., 2009), their large effect on protein production (Kensy et al., 2009, Khalil and Collins, 2010, Gulati et al., 2009), their ease of use (Alper et al., 2005, Danino et al., 2010, Fussenegger, 2010, Kötter et al., 2009, Kensy et al., 2009, Khalil and Collins, 2010, Gulati et al., 2009, Kobayashi et al., 2004, Chappell et al., 2013), and their pervasiveness in the characterisation literature which makes them an ideal candidate for platform validation (Canton et al., 2008, Kelly et al., 2009, Zhang et al., 2007, Chappell et al., 2013). Promoters are responsible for the binding of the transcription machinery and are the predominant control point for the production of mRNA that ultimately leads to reporter production. The advantages of using promoters as the primary test set for the success of the technology is that they are an established standard for characterisation methods (Canton et al., 2008, Kelly et al., 2009, Chappell et al., 2013).

The targets for the platform output (i.e. characterisation data) have been informed through two seminal papers from the characterisation literature, namely, Canton *et al.*'s: Refinement and standardization of synthetic biological parts and devices (Canton et al., 2008), and Kelly *et al.*'s:

Measuring the activity of BioBrick^{TM} promoters using an *in vivo* reference standard (Kelly et al., 2009). In the first paper, Canton *et al*. successfully characterise a biological sender-receiver device using Acyl-Homoserine Lactone (AHL) as a small molecule inducer (the input to the system). The results from their research quantify all the behaviours and parameters (confounding or otherwise) associated with their biological device. These include the steady state reporter levels over varied input concentrations, the dynamic performance of the reporter over time at a specific input concentration, the interaction of the reporter with other similar small molecules, and the performance of the reporter over an extended period of time. Subsequent to this, the authors successfully parameterize the device (describe the components of the device in mathematical terms) within a computational model, allowing accurate prediction of the device's behaviour given certain environmental constraints. As intimated earlier, a computational end point for characterisation data is becoming increasingly common within the field (Stricker et al., 2008, Danino et al., 2010, Fussenegger, 2010, Bayer and Smolke, 2005, Friedland et al., 2009, Beisel et al., 2008, Basu et al., 2004), for example the outcomes from Canton *et al*.'s research represent ideal specifications for a characterisation platform. Therefore it is essential that any data produced using the proposed technology be directly useful to biological CAD tools.

To achieve these specifications a detection system that is high-resolution and high-throughput must be used, as well as methods and techniques that eliminate confounding affects (such as generation time and chassis interactions). It was decided that *in-vitro* transcription-translation mixtures were to be used as the expression medium, to further reduce environmental variations within the characterisation data as well as increase the speed at which the data could be acquired (Chappell et al., 2013). Furthermore, microfluidic droplets were to be used to encapsulate the expression medium in order to generate large numbers of repeats (improving the data quality through quantity) and handle multiple characterisation reactions concurrently. To integrate such a disparate group of disciplines, detailed specifications were created for both the platform as a whole and at each of the interfaces between the fields. As such, it was an early aim to generate platform specifications to enable the use of a cyclic workflow for the creation of this technology.

5.4.2 *In-Vitro* Transcription-Translation systems as an Expression Medium

In 2006 George Church and Antony Forster (Forster and Church, 2006) presented a review proposing possible applications for Synthetic Biology using *in vitro* chassis homologues. Primarily, the authors hypothesise that an *in vitro* transcription and translation (IVTT) system provides a more flexible platform for engineering biology as it allows for tighter control of environmental variables and any reactions that may be occurring in the background, behind the reaction of interest. IVTT systems have been conceptualised as genetic circuit 'breadboards' by some, as they are becoming quickly recognised for their fast turnaround, reduced complexity and well-studied nature (Forster *et al.* 2007, (Hockenberry and Jewett, 2012).

One of the key uses for IVTT mixtures has been, and still is, within protein production systems (Khnouf et al., 2009, Dittrich et al., 2005, He, 2008, Jewett et al., 2008, Kara and James, 2009) because of their high protein production rates and relative chassis simplicity (reducing the amount of required downstream purification) (He, 2008). Previous studies show that IVTT mixtures are restricted by their inability to activate several reaction pathways in parallel; however, this does not present a problem for a foundational characterisation chassis as it requires minimal network complexity (Shin and Noireaux, 2010, Shin and Noireaux, 2012). Generally these studies have helped establish that IVTT mixtures provide all the essential transcription and translation machinery required for producing proteins (Zubay, 1973, Zubay, 1980, Nevin and Pratt, 1991) which is essential for a cell-free characterisation platform.

More recently, the Noireaux lab and colleagues have produced an exhaustive set of investigations covering IVTT's utility within the field of synthetic biology. These investigations have focused predominantly on a novel, all E. *coli* based transcription translation system coined TX-TL (Garamella et al., 2016). Noireaux *et al.* have shown that the system is ideal, not only for high volume protein production, but also suitable for the expression of complex biological systems. Much of Noireaux *et al.*'s work was published after the conclusion of the work performed in this study, but it represents an interesting avenue for improvement for the platform.

The primary advantages of using IVTT as the characterisation chassis are reduced environmental effects, reduced cell cycle dynamics, and reduced metabolic effects (Forster and Church, 2006). In essence these advantages help reduce variation between characterisation assays whilst maintaining versatility in both the types of biological components that can be investigated, and the types of assays that can be performed (Jewett et al., 2008, Kara and James, 2009, Noireaux et al., 2003, Noireaux and Libchaber, 2004). Whilst current methods for characterisation (as described in Canton *et al*. (Canton et al., 2008) and Kelly *et al*.'s papers (Kelly et al., 2009) address most of the characterisation assay requirements, the data produced includes some drawbacks, such as a high context dependency (Kelly et al., 2009) and time requirement (Canton *et al*.'s 2008 study took over two years to complete).

Additionally, IVTT as an expression medium allows for accurate parameterisation of all of its constituents without all of the metabolic and cellular dynamics that are present *in vivo*, which presents the opportunity to model the chassis in its entirety (Takahashi et al., 2015). This advantage relates back to the concept of decoupling, whereby the activity of a functional part of interest can be separated from the normal background signal patterns. This decoupling is especially important when considering the lead in time for assaying the biological part of interest, which historically, has been high due to the need to work with live cells. Instead, IVTT mixtures leverage cell extracts whose behaviours are independent of transformation efficiencies, culture times, and general network characterisation issues.

One of the core standards in Synthetic Biology is the RFC10 BioBrick^{TM} assembly format for biological constructs (www.partsregistry.org). This standard has been shown to work well in conjunction with IVTT in previous and on-going work (Chappell et al., 2013) and therefore represents the core technique used to assemble the DNA in this work. In most circumstances these represent improvements on one or more aspects of the original RFC10 standard but in order to maximise compatibility with ongoing and future work this investigation has maintained usage of the RFC10 standard.

A platform based upon IVTT expression enhances versatility and the ability of the platform to expand into the variety of modern IVTT applications that are arising, including mRNA recovery and metabolic pathway engineering (Kara and James, 2009, Chappell et al., 2013). For example, future iterations of the proposed platform could characterise more complex biological reactions, including biological devices and quantifying mRNA production in conjunction with protein expression (He, 2008, Jewett et al., 2008, Yang et al., 2009). Previous studies have already established the viability of IVTT as a characterisation medium (Shin and Noireaux, 2010, Shin and Noireaux, 2012), not only for simple biological part characterisation but also for complex function biological circuits.

Recent work by James Chappell *et al.* shows that an IVTT chassis is viable for characterisation of simple biological parts (Chappell et al., 2013). Their work shows an effective methodology for performing characterisation using IVTT at a rate that surpasses conventional characterisation methodologies. The work performed in their study exemplifies the type of characterisation assay that was targeted for use within the high throughput characterisation platform. Furthermore this work, and others have shown the possibility of using linear DNA as the expression backbone rather than the more traditional plasmid based transformation vectors (Sun et al., 2014).

5.4.3 Microfluidics and Synthetic Biology

The microfluidics field covers the investigation of chemical reactions on the nanolitre to picolitre scale (Gulati et al., 2009). Much of the on-going research in this field is driven by the desire for miniaturisation in the fields of biotechnology and medicine. Employing microfluidic techniques leads to reduced consumption of reagents, tighter control over reactions, and detailed quantification of the analytes being investigated (deMello, 2006). By leveraging techniques from the semiconductor industry, such as soft-lithography as a fabrication technology, the microfluidics field has generated a host of solutions to biological, biomedical and biochemical problems.

Microfluidic investigations generally utilise poly-dimethyl sulphate (PDMS) 'chips' to manipulate low volume liquid streams for tight control over reaction components and environments although other examples of chip substrates do occur. Combining this flexible and fast turnaround fabrication methodology with the ability to manipulate and control small fluid volumes imparts advantages that allow for high-throughput data acquisition (Srisa-Art et al., 2009, Srisa-Art et al., 2007, Wang et al., 2009), low cost per reaction (deMello, 2006, Agresti et al., 2010, Niu et al., 2009, Zhang et al., 2006, Huebner et al., 2008, Huebner et al., 2009), automation (Millington et al., 2010) and fine manipulation of the reactants (Gulati et al., 2009, Forster and Church, 2006, Doktycz and Simpson, 2007, Griffiths and Tawfik, 2006, Tawfik and Griffiths, 1998, deMello, 2006). These advantages make microfluidics an ideal investigative tool for medium to long term biochemical reactions such as those targeted by characterisation assays.

Many examples exist for the use of microfluidic technologies in conjunction with IVTT to enhance the cell free synthesis of proteins (Georgi et al., 2016). These examples utilise the high power of microfluidics to control and coordinate reactions to produce proteins that may otherwise be toxic to living cells, or require non-natural resources to be consumed . Microfluidic technologies are increasingly being used in 'at patient diagnosis solutions' where a diagnosis can be performed at the patient's bedside rather than in a dedicated laboratory. These solutions typically increase the speed at which a diagnosis can be obtained without sacrificing sensitivity. Additionally, these small devices should serve to reduce the overall cost for assays as they purport to reduce the reagent consumption and hands on time requirements (e.g. laboratory staff) for a diagnosis to be acquired (Yang et al., 2015, McGrath et al., 2011).

An extension of the microfluidics field is droplet microfluidics, whereby analytes are encapsulated by pumping the reagents into an immiscible fluid (deMello, 2006). The flow instabilities between the two immiscible fluids can cause spontaneous droplet formation. The formation of these droplets conceptually compartmentalises the reactions, acting much like individual reactions in their own right (Tawfik and Griffiths, 1998, Griffiths and Tawfik, 2006). Thus, these droplets represent chemically inert micro-bio-reactors (Tawfik and Griffiths, 1998, Schaerli and Hollfelder, 2009, Kong et al., 2007, Williams et al., 2006) which can be considered as small cell-like capsules (Griffiths and Tawfik, 2006, Viskari and Landers, 2006). By considering the droplets in this way, the characterisation platform created below can utilise the low reagent consumption, high droplet homogeneity and large sampled parameter space that the microfluidics droplet technology offers thereby addressing the platform's requirements. The exceptional ability of microfluidics to sample very large parameter spaces is best exemplified by their proposed usages in high throughput screening platforms (Du et al., 2016). These platforms require quick assessment of hundreds or more reactions to confirm the presence of specific molecules or their precursors. These technologies look set to shake up the current screen market that relies on large fluid handling robot infrastructures and microtiter plates (Dittrich and Manz, 2006, Neuzi et al., 2012, Hong et al., 2009).

By relying on a fabricated 'chip' structure, the microfluidics technology is inherently modular and this allows for a variety of detection techniques, droplet incubation, and existing microfluidic designs to be validated and used. For example, the characterisation platform could be coupled with existing microfluidic designs such as on-chip PCR (Zhang et al., 2006, Williams et al., 2006), electroporation (Wang et al., 2009), DNA sequencing (Paegel et al., 2003), and DNA assembly (Zhou et al., 2004). There is also an increasing number of detection methods associated with microfluidics, each targeted to different application specifications. Examples of detection methods in microfluidics include avalanche photodiode detectors (APDs), charge coupled detectors (CCDs or emCCDs), and electrochemical detection methods (Baker et al., 2009). These technologies are increasingly combined with technologies to monitor, assess and validate the construction quality of microfluidic devices (Kawano et al., 2015). The work done in a recent paper by Kawano *et al.*, demonstrates a novel detection solution for assessing both the fluorescence of samples within a microfluidic chip, and also to visualises chip details, such as the channel walls or microbubble imperfections, without using prior (non-integrated) techniques such as scanning electron microscopy. A great deal of microfluidics' power as an investigative technology is derived from its synergy and integration with a variety of disparate disciplines involved in the detection of small molecules or reactions and utilising them in an in-line and automated manner (Maceiczyk et al., 2015).

A paper by Agresti *et al*. (Agresti et al., 2010) exemplifies the industrialisation potential of the microfluidics technology. In this study, droplets were formed in a primary device followed by detection and sorting in a secondary device. Capabilities to screen and sort biological constructs for their levels of production, as shown in Agresti *et al*.'s research, allow for a high-throughput, high-resolution characterisation platform to move towards a rapid prototyping device or mutagenesis library selection device, especially when combined with the existing microfluidic modules mentioned above.

5.4.4 High Throughput DNA-Based Biological Part Characterisation Platform Overview

Using the principles discussed above, a design for the platform was envisaged. This design called for the generation of encapsulated picolitre scale characterisation reactions within microfluidic channel devices, which allowed for fine grained control over observation of the droplets and their manipulation over time. The characterisation reactions were to be based around the the BioBrick^{TM} RFC10 standard as Chappell *et al.* have already established the feasibility of this. Obviously one of the major hurdles that needed to be overcome is the interaction between the characterisation reaction and the microfluidic device.

Some precedence is established for work at the interface of IVTT and microfluidics. It has been established that IVTT has previously been used in conjunction with microfluidics as a protein expression system (Khnouf et al., 2009, Dittrich et al., 2005, Noireaux and Libchaber, 2004, Doktycz and Simpson, 2007, Griffiths and Tawfik, 2006, Tawfik and Griffiths, 1998). In a core example of this Khnouf *et al*. (Khnouf et al., 2009) use un-encapsulated IVTT as a protein expression system in microchannel arrays, as an expandable protein production platform. While the core aim of the paper is to investigate the IVTT expression system for industrialised protein production, lessons can be learned that inform the platform's design, such as the versatility of the platform to expand into DNA vectors that are not based around the BioBrick^{TM} RFC10 standard. As an extension of this, Griffiths and Tawfik (Tawfik and Griffiths, 1998, Griffiths and Tawfik, 2006) have encapsulated IVTT with a technique known as *in vitro* compartmentalisation (IVC), emulating the compartment-like nature of *E. coli* cells. More relevantly, Dittrich *et al.*'s work in their 2005 paper shows that proteins can be quantified using fluorescence as a reporter. Dittrich *et al.* attempt to quantify the fluorescence levels of droplets containing a single GFP producing vector. Unfortunately, this work did not exhibit the reproducible characteristics that are desirable in a foundational technology, but their efforts were used to inform the work performed in section 6.

Richard Murray's lab has recently demonstrated that a cell free approach for engineering biological systems allows rapid novel biological system development, and that the information obtained within these *ex vivo* characterisation assays can be translated to *in* vivo (Niederholtmeyer et al., 2015). In some of their most relevant work Niederholtmeyer *et al.* utilise microfluidic bioreactors to characterise the oscillatory profiles of three, four and five ring oscillators. Unique to their characterisation methodology, is the emulation of cell cycle behaviour through discontinuous reagent flow. In contrast to this, the work proposed here attempts to encapsulate IVTT mixtures in small cell-like compartments, reducing reagent volume and promoting reagent cycling through built in regenerative mechanisms.

A review by Gulati *et al*. (Gulati et al., 2009) considers the potential of Microfluidics to become a foundational technology for Synthetic Biology, especially for characterisation assays. Principally Gulati *et al*. highlight some of the advantages of the microfluidics technology, whilst covering some of the technological gaps that are present within Synthetic Biology. More recently, work produced by the Hasty Laboratory at Caltech (Stricker et al., 2008, Bennett and Hasty, 2009, Prindle et al., 2014) exemplify modern applications of a microfluidic platform to Synthetic Biology. It should be noted that these represent implementations that are only relevant to their application of interest and are neither robust nor high throughput enough to function as the basis for a high throughput characterisation platform where the sample behaviour being measured has unknown boundaries (i.e. the assay measurements are being performed *a priori*). More recently, Linshiz *et al.*, have demonstrated an end-to-end microfluidic solution that facilitates the design, construction, testing and analysis of biological parts and devices. Whilst Linshiz *et al.*'s solution still requires more conventional lab based techniques such as cell culturing, it is on the right track for abstracting away many of the more common problems associated with meeting Synthetic Biology's application targets (Linshiz et al., 2016).

The work performed in this investigation was based upon the belief that a synergy between disciplines, such as those discussed during the enumeration of potential detection technologies in microfluidics, would greatly benefit Synthetic Biology. The plethora of Synthetic Biology's applications and the multitude of possible (and sometimes required) observations for characterisation assays, require robust solutions and high quality data (Andrianantoandro et al., 2006, Arkin, 2008, Heinemann and Panke, 2006, Khalil and Collins, 2010, Gulati et al., 2009, Jungmann et al., 2008). As discussed previously, these reactions can vary across types and time, and require large numbers in order to have statistical power in describing the effects of biological parts on biological systems. It therefore seems intuitive to suggest that the advantages mentioned above, with regards to microfluidic technologies, promote microfluidics as a powerful solution to the characterisation problem.

In summary, the aim of this portion of the study was to create the high throughput characterisation platform by generating multiple characterisation reaction droplets within a microfluidic device. These reactions needed to be both repeats of a single reaction type, as well as multiple reaction types, in order to increase the throughput of the platform and allow multiplexing of characterisation assays. To achieve this, it was imperative that once a viable droplet manipulation and data detection protocol was established, that effort was emphasised into the front-end sample input section of the platform. The work performed also represented an extension of a previous investigation into the feasibility of a microfluidics based characterisation technology. This work demonstrated a first round of design iterations, some of which were used within the work detailed below either in further tests, or as springboards to further design tweaks. Where possible and where clear delineations between the two works exist, such as any prior design iterations that informed the work below, distinct work will be marked as such.

Over the course of this investigation, a novel microfluidic droplet formation technology was being developed in parallel (Gielen et al., 2013). Within their investigation Gielen *et al.*, develop a robotic technology to automate the encapsulation of microfluidics droplets, forming small reaction compartments on demand. This technology provides a unique methodology to encapsulate a large number of reaction types as well as produce droplets in great quantities. When the work below was first performed, the technology was still in its infancy, and was only adopted later in the work.

5.5 Handling of Characterisation Data in Synthetic Biology

5.5.1 Aims for the Investigation

Section 5.3 highlighted the second core limiting factor for the uptake of regular DNA part characterisation within a Synthetic Biology research pipeline. Namely, the large volume of data required for comprehensive coverage of all possible behaviours (and the requirement for this data to be compatible across multiple research institutions), the variety of data analysis techniques available for processing the data, and any metadata necessary to contextualise the biological part under investigation. As Synthetic Biology makes inroads towards a large volume and data driven discipline, useful corollaries can be drawn from fields that have experienced similar transitions such as the more traditional biological sciences, as well as medicine (Shah and Tenenbaum, 2012), and computing (V. Knyazkov et al., 2012). These corollaries tend to teach and inform enabling solutions that can be brought across to Synthetic Biology. Three types of solutions were identified as areas for integration within a data handling framework for Synthetic Biology including: tools for handling large data volumes, solutions that reduced the need for manually processing data, and the introduction of translatable standards across data handling layers. Examples implementations of these solutions include: the DICOM database (Mildenberger et al., 2002), the SCOP database (Murzin et al., 1995) and the UniProt catalogue (Consortium, 2013). The work performed below used the three solutions enumerated above to derive a flexible and robust solution to the data handling problem in Synthetic Biology (see Section 5.5.5).

5.5.2 Provisioning for and Handling DNA Part Characterisation Data Volumes

Data volume has not been a traditional concern of biological laboratories, and under standard throughput it has generally be considered sufficient to allow individual researchers to manage their data. With the introduction of more high-throughput technologies such as DNA sequencing, automated microscopic imaging, and high speed, high resolution fluorescence detection techniques, the amount of data produced by rigorous experiments has grown exponentially. These technologies are increasingly being used throughout the biological sciences as the technologies grow cheaper or become more approachable. Even the predominantly adopted 'gold standard' for characterisation of a single biological part relies on a large number of data points for it to be minimally viable. Traditionally this entails that for a single biological part within a sample, three independent replicates (different bacterial seed colonies) of three repeats (Canton et al., 2008, Kelly et al., 2009, Chappell et al., 2013) are required. Assuming cultures are observed every 30 minutes over a four hour assay, where both the optical density (OD) and fluorescence of the sample are measured, a total of 144 data points are required per sample (this value does not include standardising the recorded data using a reference sample).

Whilst the volume of data is not yet comparable to the example databases mentioned above, establishing storage and processing methodologies early within the data driven field of biological part characterisation will increase the uptake of characterisation techniques, as the methodologies will speed up the time needed and decrease the difficulty in obtaining useful results. There are several techniques that can be employed to help facilitate better data handling including centralising data storage moving towards a database architecture, as opposed to the traditional flat file model used by researchers (almost ubiquitously) today, as well as designing any storage models with flexibility in mind, in order to accommodate the evolving standards that are typical of new and rapidly developing fields such as Synthetic Biology.

Centralised data storage promotes the safe and organised management of data and whilst it is not universally applicable, the multitude of advantages in the case of research data (and in this specific case DNA part characterisation data), promote its adoption for centres employing high throughput technologies. The core advantages purported by centralised data storage techniques include common storage standards, transferability and share-ability, data resilience, and scalability. A traditional example of centralised data storage is the concept of a database. A database can take many forms but traditionally consists of SQL (Structured Query Language) or NoSQL type formats. These formats come in a variety of implementation specific 'flavours' but generally adhere to a set of core concepts that promote data resilience, storage standards, as well documented data access, and data operation optimisations. Implementing these techniques can generally increase the organisation of data throughout organisations, promote cross interrogation of data sets that would traditionally remain separate, as well as provide routes for data audit and tracking analyses to be performed. The additional rigidity associated with a centralised data storage infrastructure does not imply a reduction in the flexibility of the data types that can be handled. Instead, flexibility can be reacquired by careful design of the common storage formats, specifically by promoting adaptability in data formats, metadata content and descriptive information. Flexibility of this kind is inherently necessary in emerging fields as standards tend to shift, improve and refine over time.

Common storage formats generally promote better labelling practices of data as they can require semi-structured data submission, which in turn can reduce data submission errors and promote data cross examination. Developing a well-documented and open storage standard promotes an increase in understanding of data structure, meaning and analysis routes. Having a common storage infrastructure can also promote data sharing and transfer as, alongside common storage formats, it promotes understanding of data formats as well as documents data access routes and policies. Data resilience is a core advantage offered by the centralised data storage technique, as it allows for redundancy infrastructures to be put in place as well as ensuring that data is not lost or discarded during laboratory transitionary phases (such as the leaving of a key staff member). Finally, centralised data storage offers a wide variety of scalability options to improve performance, uptake and general utility. These scalability options are a direct reflection of the maturity of the techniques as well as their common use throughout modern IT infrastructures.

5.5.3 Reducing the Hands-on Nature of Data Analysis for Synthetic Biology's DNA Part Characterisation

Biological and big data analysis have traditionally been difficult for non-specialised research individuals to approach, either because of the complexity of the plethora of statistical tools available, or the inherent requirement for specialised knowledge (such as programming) to perform the analysis. This difficulty has been mitigated through the creation of novel analytics toolboxes and platforms, but these often fail to keep up with rapidly developing fields or are technology and technique dependant. Biological data analysis is also becoming more computationally intensive and increasingly requires access to dedicated processing or analysis environments. Examples of this complexity are present within Synthetic Biology already, where complex genetic systems can be simulated with non-deterministic algorithms, in order to approximate and estimate biological behaviours. In the investigation below, several computation techniques were leveraged in order to counter the hands-on nature of current characterisation data analysis in Synthetic Biology.

As stated earlier, these issues are more often mitigated through the development of specialised software, and examples of this can be seen in the DNA sequencing field where companies, such as Illumina (Illumina, Inc.), have recently developed dedicated cloud based platforms for the analysis of sequencing data with their BaseSpace platform. Another example is the Galaxy project (https://galaxyproject.org/) which promotes the creation and management of biological data analysis pipelines in a modular manner, by leveraging small isolated processing or analysis components. Both of these software examples present modern, flexible architectures for data analysis and are emulated in the novel software developed below to provide an integrated data storage and analysis solution.

The purpose of this investigation was not only to achieve a specific solution to the characterisation problem in gene expression, but also derive a novel platform that would accommodate future developments in the field, such as changes in standards, analysis techniques and the technologies used. Additionally, by taking inspiration from the Galaxy modular architecture, a route for iterative and open development for data processing or analysis modules becomes apparent, which would encourage the collaborative development of tools, techniques and standards with regards to data in Synthetic Biology.

5.5.4 Leveraging Standards to Improve Characterisation Data Handling in Synthetic Biology

Engineering disciplines have almost always identified standards as a common necessity in increasing the understanding of systems, maximising their scope of use (by ensuring non-specialists can access the knowledge), promoting workflow standardisation and increasing the cross pollination of ideas through the cross examination of common principles (in this case it is the cross interrogation of the data that provides this). Synthetic Biology has recognised the need for standardisation at the methodological level, but still suffers at the data storage level.

Examples of common storage formats do exist for sets of specific data types. Two such storage methods were mentioned briefly before, namely the Digital Imaging and Communications in Medicine standard for medical imaging (DICOM) and the Synthetic Biology Open Language (SBOL). The DICOM standard was an evolution of a previous standard (ACR/NEMA), which was first used in 1993 (Bidgood and Horii, 1992). Since then the standard has not only grown to encompass interpretation and display standards for medical images, but also to design services for their use such as data transmission and data redundancy and storage. The DICOM standard was so successful in its uptake across medical infrastructure that it has informed the development of a Synthetic Biology analogue (DICOM-SB). DICOM-SB serves to provide an extensible, re-usable model for Synthetic Biology's data, whilst also providing servers for data exchange (Sainz de Murieta et al., 2016). DICOM-SB may become the *de facto* standard for data storage in Synthetic Biology, but was only recently finished. The Synthetic Biology Open Language (which purportedly can be used in conjunction with the DICOM-SB storage and handling standard) can be used to describe the sequence associated information, similar to that of more conventional storage formats such as GenBank (Benson et al., 2005). The most recent version of which allows for a wide range of components and dynamic interactions to be described. These dynamic interactions can be directly represented by associating SBOL files and Systems Biology Markup Language (SBML) files. As these standards become more and more accepted, as well as more and more mature, their integration into common infrastructures will become more important. These common storage formats also extend directly into high throughput experimental results from microarrays and high-throughput sequencing as exemplified in the ArrayExpress repository (Kolesnikov et al., 2015). ArrayExpress offers an international data repository service for storing high-throughput functional genomics experiments to promote reuse and storage redundancy. The ArrayExpress adheres to core established data standards for storing microarray and sequencing data (MIAME and MINSEQE respectively). ArrayExpress also provides a common portal for experiment archiving whilst brokering the actual data storage to storage delegates such as the European Nucleotide Archive.

Another example of the data types that need to be stored in biological experiments are the metadata information that describe the types of biological measurements, their context and their methodologies. Most recently, a large conglomeration of biological standards has arisen under the BioSharing umbrella. BioSharing provide a curated web based interface for search registries on standards, databases and policies within the biological sciences. These standards include the aforementioned MIAME and MINSEQE standards but also include the notable MIBBI standard (Minimum Information for Biological and Biomedical Investigations). MIBBI provides a set of checklists that aim to define the minimum information set necessary to describe a biological investigation.

The difficulty in establishing data analysis standards directly prevents the uptake of characterisation within the Synthetic Biology research pipeline, as it requires a considerable time investment in order to independently research the most robust and performant methodologies. This problem is best exemplified by the variety of statistical analyses and data visualisation techniques used within the previously discussed canonical characterisation work (Canton et al., 2008, Kelly et al., 2009, Chappell et al., 2013), which either uses confidence intervals or standard deviation to calculate the error in sample results. Furthermore, the lack of consensus regarding the required contextual information for characterisation data can also slow the uptake of characterisation. Some of the difficulty in establishing standardised data analyses is the perceived need for Synthetic Biology's characterisation work to produce analogues to the engineering datasheets as described in Canton et al.'s 2008 characterisation paper. Whilst this movement is commendable in furthering the application of engineering to biology, such a rigid output does little to reflect the inherent complexity of biological systems and their possible applications. It is difficult to identify what contextual information is required when the possible usages of biological parts are so wide and varied. Both the lack of established analysis standards, and the required contextual information further confound the difficulty in producing viable characterisation data that meets Synthetic Biology's implied characterisation standards (Canton et al., 2008).

5.5.5 Data Processing, Analysis and Display – A De Novo Data Handling Software Suite for Synthetic Biology

In the work below, a comprehensive data handling framework was designed and created. The software was built with the Java programming language which is known for its cross-platform compatibility as well as its wide array of software libraries to build upon. This software solution was engineered with the primary target of handling gene expression characterisation data, but also with sufficient flexibility so as to allow other data types, analysis modules, and displays, to be dynamically plugged in to the system. Beyond this, the software was designed to help centralise, coordinate and control data storage and analysis by adopting a client server model that helps decouple the various system logics that underpin the software. The client portion of the software was designed to provide common visualisation tools, data loading modules and network components and the server side of the software was designed to provide multithreaded network infrastructure, multithreaded plugin execution infrastructures, as well as all of the database abstraction infrastructures which specifically utilised the JDBC driver to interact with an H2SQL database.

This dynamic polymorphic and plugin-able design helped maximising the versatility of the software by allowing it to be adapted to new data analysis techniques, standards and methodologies, whilst maintaining stability in its storage solutions, thereby providing a consistent and centralised information source for all users. As such, the platform was designed to handle multiple information sources, facilitate batch loading of data, perform consistent data analysis that are contextually relevant, and present any available results in a meaningful manner.

It was also important for the software to maximise its utility across large and complex organisational structures whilst requiring minimal infrastructure changes. To achieve, this the software was designed to work under a client/server model which completely decouples the data storage and analysis from the data input and display. This model was a direct implementation of the centralisation techniques described above. The software was designed to be wholly self-serving, not relying on web engines and other software installations. The latter of these has long term implications for the software's utility. Namely, whilst independence and ease of setup were maximised (by removing dependencies on other software), the ability for additional intelligence to be placed between the client and server was reduced (implying that performance increasing techniques such as load balancing and distributed computing are not directly available). This may appear to be a long term drawback for the software with regards to big data analysis (which often has higher computational demands and therefore benefits greatly from small performance enhancements), however the problem can actually be mitigated by utilising the plugin architecture described above, to delegate computationally intensive tasks to more suitable infrastructures and then re-integrating any results produced from the analysis.

Finally, the software aims to provide a platform that can quickly take up a storage standard, or even accommodate multiple standards, by providing flexible and non-constrictive metadata storage. One of the key guiding principles was that any storage infrastructure put in place would not define a new standard in itself, which often times tends to pollute and confound progress for a standard's uptake. Instead it was a core aim for the software produced in the work below to allow the quick uptake of situationally relevant data standards when specifically called for, but to generally be agnostic in the technology and experiment types, in order to provide maximum versatility in the software implementation as well as to ensure the future utility of the software.

6 The Microfluidic *In-Vitro* Characterisation Platform Results and Discussion

6.1 Platform Design and Testing Strategy

As outlined in section 5.4, the overarching platform strategy was to utilise IVTT as an expression medium for DNA constructs, whilst encapsulated in a droplet surrounded by an oil phase within a microfluidic device. Beyond this, automated data analysis was to be established allowing for increased speed and a reduction in the *ad hoc* nature of data processing. In theory this combination of disciplines achieves:

1) The high throughput demands of the platform by exploiting microfluidics' ability to produce high numbers of reactions encapsulated in droplets (Srisa-Art et al., 2007, Srisa-Art et al., 2009, Wang et al., 2009)
2) The accurate characterisation demands of the platform by employing IVTT's ability to represent a biological system's behaviour under controlled conditions (Chappell et al., 2013)
3) A standardised analysis methodology, reducing the *ad hoc* nature of characterisation data analysis
4) High-throughput data acquisition and contextualisation with automated data handling (including statistical confidence inference) (Canton et al., 2008)

To achieve a high-throughput microfluidic *in vitro* characterisation platform, multiple initial constraints were outlined for the platform during the design process, each relating to the key aims of the platform. These initial constraints informed the design by limiting the number and types of techniques and experimental methodologies that were suitable for use. From these constraints a set of explicit platform specifications, for both the scope and the direction of the platform, were derived (these are summarised in Figure 4).

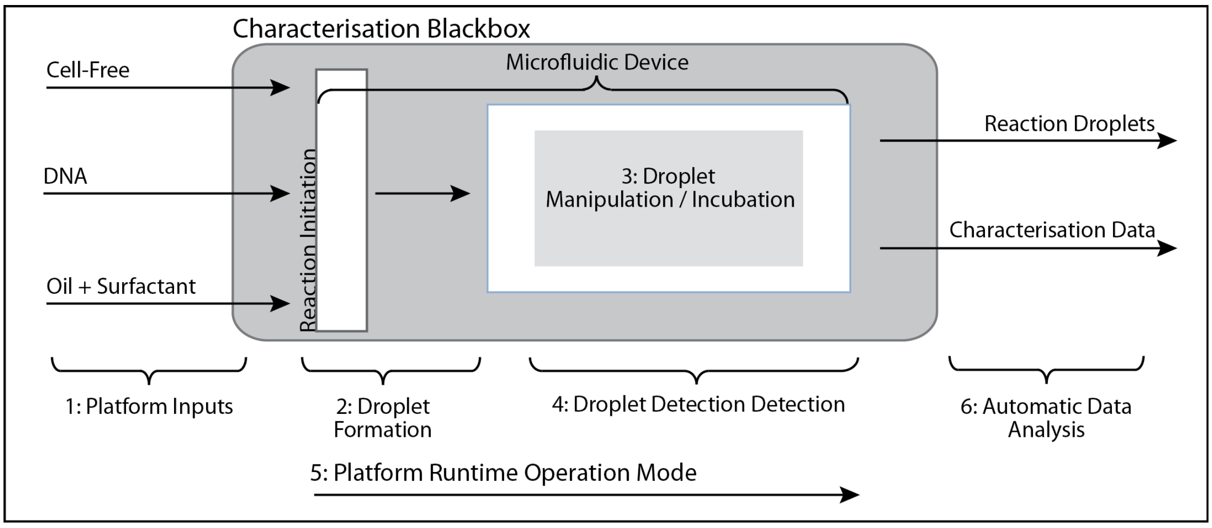

Characterisation Blackbox
Cell-Free
DNA
Oil + Surfactant
Microfluidic Device
Reaction Initiation
3: Droplet Manipulation / Incubation
Reaction Droplets
Characterisation Data
1: Platform Inputs
2: Droplet Formation
4: Droplet Detection Detection
6: Automatic Data Analysis
5: Platform Runtime Operation Mode

Figure 4: Illustrates the overall platform schema. Different sections of the platform design are broken into modules to allow for faster iteration cycling. These modules are marked numerically according to when they affect a normal platform workflow.

6.1.1 Design constraints

The initial set of platform design constraints was derived from the requirement that any data acquired must be directly comparable to previous *in vitro* characterisation studies performed by Chappell et al. (Chappell et al., 2013) as their study represents a complete comparison of *in vitro* vs. *in vivo* characterisation techniques. Considering this, all data acquired needed to summarise a dynamic profile of GFP production over time, under controlled environmental conditions, and be directly attributable to a single reaction type that entered the platform. Reactions studied by the platform must conform to the characterisation adage of only containing a single distinct change in the reaction mixture (usually the DNA template) for each non-repeat sample being investigated. The design constraints derived from this are summarised below:

- All reactions must be tracked and recorded over a minimum duration of four hours, or a time that directly corresponds to a representative timeframe of GFP production within the expression medium
- A minimum of two distinct sample reactions must be studied concurrently in order to standardise the data sets, specifically the sample of interest and the reference sample.

Multiple reactions are highly preferable in order to maximise the environmental similarity and to compete with existing characterisation techniques

To improve upon existing characterisation techniques and take full advantage of droplet microfluidics' innate benefits, the platform also needed to either exceed the number of concurrent reactions investigated during an experimental run, or to exceed the statistical confidence in any experimental results by increasing the number of reactant repeats when compared to the current characterisation methods. The platform should require similar or less manual involvement once the characterisation has begun, as current characterisation techniques require no user intervention once the reaction has started (Canton et al., 2008, Kelly et al., 2009, Chappell et al., 2013). Considering this, the following design constraints were produced:

- Reduced reagent consumption in comparison to conventional analytic methodologies (deMello, 2006)
- High numbers of samples (to allow characterisation multiplexing) and their repeats (to increase the statistical confidence in observed characterisation data) (Srisa-Art et al., 2007, Srisa-Art et al., 2009)
- Consistent droplet formation to ensure that droplets are congruent and therefore directly comparable from a fluorescence standpoint
- Detailed time-course datasets equalling or exceeding current time-course observation durations
- High levels of automation and stability to prevent the need for user intervention

As described by Chappell *et al.* (Chappell et al., 2013) and Kelly *et al.* (Kelly et al., 2009) control over environmental factors is essential for the characterisation process because of their high impact upon data acquired throughout all characterisation techniques (Canton et al., 2008, Kelly et al., 2009, Chappell et al., 2013). As such, a set of constraints regarding environmental variation was also created:

- Equal or improved control (in comparison to in in-lab techniques) over environmental factors affecting the samples under investigation
- Temperature of all reactants specifically must be controllable as this can vary the duration of the GFP production lifecycle in IVTT greatly (Chappell et al., 2013)
- Preventative measures must be taken to reduce or remove the sample evaporation present within current characterisation techniques (an issue that was found by (Chappell et al., 2013)) in order to maintain correct sample volume estimation
- Accurate determination of reaction initiation in order to match the detailed result sets produced by current techniques

A well-defined data analysis methodology allows for increased value in cross-comparison data as greater assurances would be present on whether the data was handled in the same manner. Inferred statistical confidence levels would also have greater value as all data-sets would have been treated in the same way. Therefore a set of design constraints pertaining to the data analysis were also derived:

- All data must be analysed in accordance with the techniques outlined by Kelly *et al.* (Kelly et al., 2009) (i.e. using an internal reference standard to minimise the impact of environmental variation)
- Representations of variation and standard deviation should be displayed in a consistent manner ensuring their cross-compatibility
- The propagation of error should be consistent throughout each analysis step
- Inferred statistical coefficients (such as R^2 coefficients) should be calculated in the same manner using all available data
- Differing data sources must result in a similar stored data pattern to ensure the cross-compatibility of datasets

6.1.2 Platform Iterations and Modularity

An obvious but important consideration when designing this platform were the interactions between each of the disciplines (Synthetic Biology, Microfluidics and Engineering) used during its construction and operation, and how to ensure that any unpredictability derived from their interaction was minimised. To better identify, isolate and improve upon any issues that arose in the testing process, an iterative testing process was adopted. The iterative testing process directly follows the engineering cycle and cyclically designs and tests possible solutions to the problem, thus over the course of the entire investigation (including work performed previously) five core design iterations were designed, developed and tested.

The overall guiding principle throughout the iterative platform testing process was to simplify and reduce the number of points of failure throughout the system across all disciplines. The outcome of some of these minimisations is described further in section 6.4. Any desirable complexity removed during the minimisation process could later be re-added to the platform once a successful proof of concept had been obtained. In order to allow such a 'minimisation mind-set' to work alongside the design and testing of such a complex platform, the platform schema was broken down into separate modular sections. Each module pertains to a specific task required by the platform to operate according to the above constraints and to their implied specifications.

Including modularity into the initial platform design allows for faster identification of any points of failure in the platform. Furthermore, the modules allow for faster swapping of alternate solutions without having to redesign the entire platform. This technique has proved powerful across the fields of engineering, computing and even within Synthetic Biology, and as such was adopted early as a paradigm for the creation of the characterisation platform. When used in conjunction with the engineering cycle a powerful and agile workflow was enacted.

Each of the numbered modules within the platform schema (Figure 4) represents a point of investigation for this study, and varying any of these allows different emergent behaviours to arise within the platform. Each of these behaviours had different consequences with regard to achieving the initial aims for the platform. Individual points of investigation are discussed in dedicated sections below, however many of these points of investigation were studied in parallel or in a combinatorial manner. To summarise these investigations in a concise manner five key platform iterations are described in section 6.4 that represent a distillation of all of the concurrent work performed on the *in-vitro* microfluidic characterisation platform. These platform iterations attempt to summarise the various issues and data sets that were acquired throughout the lifecycle of this investigation and should be viewed as stepping stones towards the current platform solution.

An overview of each of the platform iterations, a generalised indication of their modular content and an indication of the core lessons learnt from each of the iterations can be found below in Table 1.

Table 1: A summary of the most salient points tested and observed from each platform design iteration.

Iteration Number	Key Modules	Key Information
1	Serpentine Channel	On-chip continuous flow designs likely to fail due to long channel requirements
2	Modular Chips	Many chip interfaces create significant points of failure
3	Trapping Design	Cup type designs cause large amounts of droplet shearing
4	Parking Design	Droplet formation using flow focussing geometries and IVTT mixture is not consistent enough for a high throughput technology
5	Chip-less	Bidirectional flow causes unpredictable merging and is not robust enough for a high throughput technology

6.1.3 Design Specifications and Platform Schema

From the design constraints (section 6.1.1), an initial set of performance specifications were created which directly informed the platform schema displayed in Figure 4**.** Six potential modules of investigation are labelled within this figure.

Inputs into the characterisation platform represent the first potential module for investigation (Figure 4 Module 1). Ideally the platform is capable of characterising multiple expression vectors concurrently, with minimal environmental influence upon the results; however, the number of inputs into the platform is linked to the number of samples that can be characterised at the same time, and therefore alternative input multiplexing solutions need to be investigated. The specification of using IVTT as an expression medium represented an effort to reduce the environmental impact upon characterisation data (Chappell et al., 2013). However, environmental variation was also attenuated through several other complementary methodologies. Input sample evaporation is reduced or eliminated through proper fluid handling and the innate properties of microfluidic encapsulation in an oil phase (deMello, 2006), whilst temperature is maintained throughout the platform by ensuring that the oil phase and reaction-phases are kept under constant heating conditions.

The second module in the platform is droplet formation (Figure 4 Module 2); many techniques exist for forming droplets in microfluidics. Key examples include the T-Junction channel geometry (Song et al., 2006), flow focussing channel geometry (Song et al., 2006) and a novel compartment-on-demand robot (Gielen et al., 2013). Each of these techniques was used to investigate the optimal droplet formation strategy in order to achieve the high-throughput droplet formation specifications discussed above. Key points of droplet formation that were under investigation included: droplet formation homology, droplet formation frequency and droplet contents. These points represent core discriminants for a working platform vs. a sub-optimal platform (optimality is from the user's perspective, whereby a sub-optimal platform represents a technology which is unlikely to be used in comparison to current gold standard characterisation techniques, see section 5.2).

With regards to the droplet manipulation module (Figure 4 Module 3) a notable requirement was to establish a first-in first-out (FIFO) system for droplets so that any data acquired for a specific encapsulated reaction could be directly attributed to the inputs for that encapsulation. Previously described techniques for protein expression in IVTT such as (Agresti et al., 2010, Griffiths and Tawfik, 2006), are not applicable to the platform because their incubation methods do not guarantee this FIFO. Maintaining a FIFO droplet relationship directly affects the aforementioned specifications regarding the types of data that the platform is required to produce, because if droplet ordering is not maintained then droplet identity is difficult to establish. If a droplet's identity is unknown, its expression profile cannot be directly associated with an expression vector and therefore the data becomes useless. Previous work using encapsulated PCR reactions (Zhang et al., 2006) showed that droplet identity could be preserved whilst manipulating a high number of droplets; however, of concern was the incubation time required for the reaction to occur as well as the ability to detect the progress of reactions throughout their lifecycle across a high number of droplets. The difficulty in maintaining FIFO droplet ordering is preventing any droplet shearing or coalescence within the microfluidic chips. The more complex the design or manipulation, the more likely this is. Many groups have got around this requirement by sorting the droplets after the fact using a form of barcoding, or being agnostic to the original droplet contents (Eastburn et al., 2015).

Detection of reactions within microfluidic chips (Figure 4 Module 4) encompasses (but is not limited to) microscope based detection techniques. As such there is a wealth of methods and literature to draw upon. Predominantly the platform requires accurate fluorescence detection with a good periodicity for obtaining observations (sampling frequency). Crucially, it is also necessary to be able to discern droplet boundaries when the observation is under way, so as to maintain the previously discussed FIFO droplet ordering. Failure to identify droplet boundaries would result in the previously discussed droplet identity issues. All detection techniques investigated suffer (to different degrees), from issues when the contents of a droplet are not homogeneous (either due to a physical process or because of bad mixing of the droplet contents). Droplet in-homogeneity was potentially an issue as some of the information regarding the droplet contents is lost when it is outside the area of observation. The two key observation techniques used in the study are detailed in section 8.3 and their advantages and disadvantages are discussed in section 6.4.

Module 5 (Figure 4) alludes to the different modes of operation that can be used in microfluidics, and with regards to the platform one of three modes was used:

- Continuous flow was the one of the modes of operation, whereby the oil phase within the microfluidic channels is always flowing and droplets never come to a rest.
- Stop-flow was another mode, where droplets can be stopped within the platform to allow for incubation.
- Cyclic-flow is a mode where droplets are cycled back-and-forth across the point of detection.

Data analysis represents the last independent module shown in Figure 4 (Module 6) but it is just as important when designing a realised high-throughput platform. To date a variety of commonly used technologies with automated data handling exists within laboratories, and for a technology to compete at this level the platform must offer the same degree of usability. An 'automated droplet identification and fluorescence evaluation' toolset was created to achieve this and is detailed in section 6.5.2 and section 6.5.3. The data analysis performed attempts to mitigate environmental variation using the techniques described by (Kelly et al., 2009), whereby finalised datasets have their environmental variation normalised by directly referencing an internal and established standard.

6.1.4 Testing Strategy

The primary testing strategy underlying each of the five iterations of the platform conformed to a simple stepwise methodology. Firstly chip functionality was assessed by testing the system with purified GFP, which also allowed for fluorescence calibration of the detectors. Secondly, a single reactant run assessed whether or not GFP, expressed by a J23100 expression vector in IVTT (assembled as per section 6.2.1), was produced and detected inside of a reaction droplet. Thirdly, the multi-sample behaviour of the platform was assessed by using a five reaction run containing different expression vectors (the corresponding constitutive promoters are listed in Table 2) which represented the range of end-point GFP concentrations likely to be observed within the platform. Finally, the platform was to be tested using the entire promoter library should the platform prove capable (as indicated in Table 10 (section 10.1)).

Table 2: List standard promoters used in this investigation. Generally used within an expression vector to determine the platforms ability to characterise a wide range of GFP fluorescence levels. * Calculated using BBa_J23100 as a reference, values obtained from (www.partsregistry.org)

Promoter Name (as per BioBrick^{TM} Registry)	Relative Expression Level*
BBa_J23100	1
BBa_J23104	0.72
BBa_J23108	0.51
BBa_J23105	0.24
BBa_J23113	0.01

6.2 Biological Precursors for Platform Operation

6.2.1 Expression Vector Design

The expression vector was designed using biological parts available on the BioBrick^{TM} Registry, which has been previously used within the discipline of characterisation in Synthetic Biology (Chappell et al., 2013). The core components of the expression vector are listed in section 8.1.1 Table 8, and an example of the archetype expression vector can be found in Figure 56. The RBS of choice for the archetypal expression vector was BioBrick^{TM} part B0034 which represents a well characterised and relatively standard RBS within the registry, and as such was considered a good candidate for assessing the viability of the platform. The core expression target, namely the GFP encoding gene, corresponds to the widely cited GFPmut3b variant which has shown high fluorescence levels and has a fast protein folding time (Iizuka et al., 2011). The terminator (BioBrick^{TM} part B0015) for the expression sequence was selected because it is in fact a double terminator, which prevents as much read-through (i.e. non-termination of the mRNA production process) of the gene as possible. The plasmid backbone (BioBrick^{TM} part pSB1A2) was selected as a whole from the registry as it has a minimal sequence length (2079bp) and contains a high copy number replication origin (pUC19-derived pMB1), leading to approximately 100-300 plasmids per cell (this increases the yield of a DNA purification step). Expression vector pSB1A2 also contains an ampicillin resistance gene to allow for plasmid uptake selection during the assembly and purification stages described in section 8.1. This expression vector represents a minimal DNA plasmid for GFP production and has been used in the characterisation work performed by (Chappell et al., 2013).

As mentioned in section 8.1.1, two sources were used for obtaining constitutive promoter sequences. The sequences obtained from the BioBrick^{TM} Registry represent a standard constitutive promoter library that is used regularly within the Synthetic Biology field (Kelly et al., 2009) where each promoter sequence has detailed characterisation data associated with it. The constitutive promoter sequences obtained from the BioFab Registry, represent a set of randomly generated sequences and any characterisation data associated with them was generated through the semi-automated characterisation processes used by the BioFab Registry (http://biofab.synberc.org/). A target promoter set encompassing these two collections allows the platform to be tested against detailed and well-accepted characterisation data as well as automated high-throughput characterisation data (obtained from the BioBrick^{TM} Registry and the BioFab Registry respectively). To rapidly generate the necessary primer sequences to obtain these promoters a short script was created in MATLAB. The source for this code can be found in Figure 60. As seen within this figure, forward primers are prefixed with the sequence AATTCGCGGCCGCTTCTAGAG and suffixed with TA whilst reverse primers are prefixed with GCGCCGGCGAAGATCTC and suffixed with ATGATC. These sequences generated overhangs corresponding to the digestions performed on the standardised backbone so that the annealed primers could be ligated directly into the backbone. Ultimately only four constitutive promoter designs were used throughout the testing of this work and their specific names are listed in Table 2.

6.2.2 IVTT Use within the Platform

IVTT reactions were initialised in one of three ways based on the iteration of the platform. One method initialised reactions by splitting the DNA component and the IVTT component so that when mixed (on-chip before droplet formation) a definite time for reaction onset was obtained. The second method, also initiating reactions on-chip (allowing accurate reaction initiation quantification), split the IVTT sample across two separate syringes constituting pre-mix in one and cell extract in the other, DNA was then added to the pre-mix sample. Finally, the reaction could be initialised off-chip (by mixing the IVTT sample and the DNA sample in a syringe) and then piped into the microfluidic devices directly. Initialising reactions in this manner prevents droplets from having identical reaction durations when they are observed (the time that has passed within the droplet since the reaction was initiated). The concentration of expression vector mixed with the IVTT solution was informed by previous work which directly demonstrated the optimal concentration outside of microfluidic droplets (Chappell et al., 2013).

To confirm that the presence of oil had little effect on the measurable fluorescence produced by a reaction containing IVTT and a GFP expression vector, a simple experiment was performed. The experiment consisted of a side-by-side comparison of GFP production from a reaction containing oil and one without. The results are shown in Figure 5. Whilst there is evidence of the oil's impact on the fluorescence readings, the general profile remains the same. On-chip the effect of the oil will be less pronounced as the amount of oil between the objective and the sample will be exceedingly small.

Figure 7 depicts a test that was performed late in the iteration cycle to determine whether or not the dilution of a portion of the IVTT mixture would have a demonstrable effect on fluorescence levels over times. This was performed when determining the impact of droplet formation geometries as well as their modes of operation (i.e. whether or not reactions could be initiated on chip by separating the IVTT components). These considerations are better discussed in Section 6.4.3.

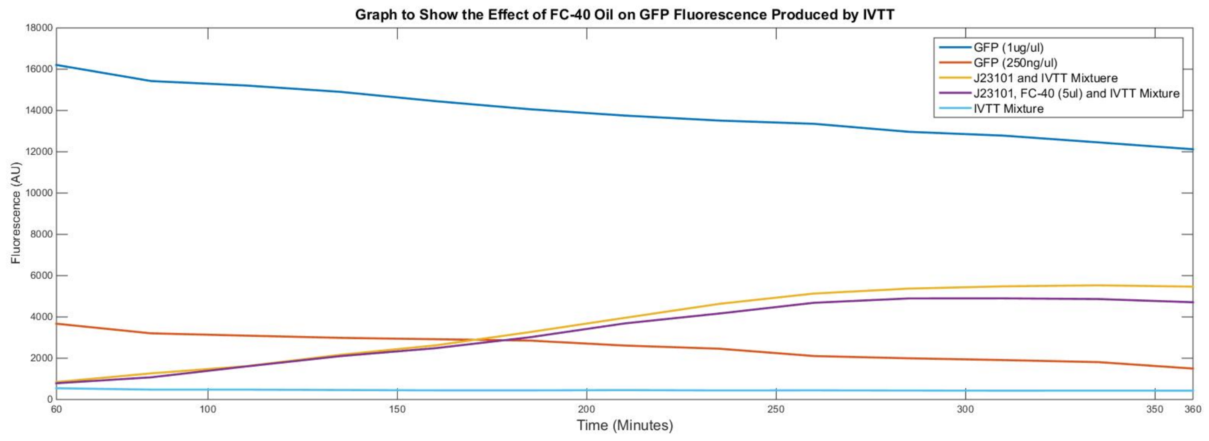


Figure 5: Diagram depicting the effect of oil on GFP fluorescence. The graph plots observed raw fluorescence against the time of the assay. Dark Blue: Purified GFP at 1µg/µl. Red: Purified GFP at 250ng/µl. Yellow: IVTT and GFP expression vector with the J23101 promoter. Purple: Identical to Yellow except with 5µl of FC-40 oil. Light Blue: IVTT mixture only

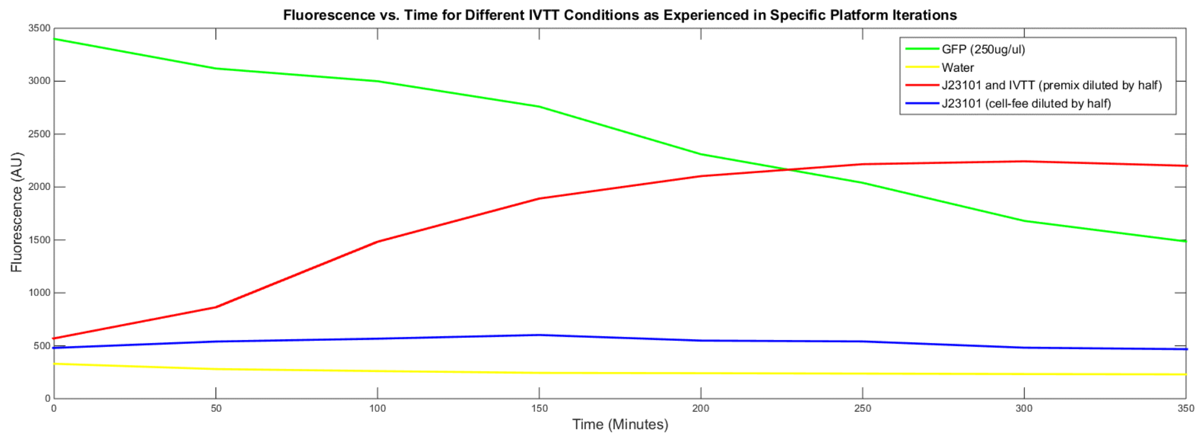


Figure 6: Production of GFP under different conditions over time. Green is pure GFP at a dilution of 250ng/µl. Red contains a GFP Expression Vector with J23101 and IVTT with the pre-mix portion diluted by 1 in 2. Blue is the same as green except the cell-free portion has been diluted by 1 in 2. Yellow is water.

6.3 Microfluidic Precursors for Platform Operation

6.3.1 Overview

Throughout the course of the investigation a slew of variations in chip fabrication, mode of operation and oil types were tested in order to overcome small shortcomings in chips. These variants were applied in an *ad hoc.* manner, as needed, to attempt to improve the performance of some designs. The principles and techniques used are given below and a summary of any and all techniques that were used within each variation can be found in its relevant section.

6.3.2 Surfactant Variants

As discussed in section 8.2.4 three combinations of oil and surfactant were used throughout the course of this investigation. As a general rule the combination of FC-40 oil and RainDance^{TM} surfactant performed best with regards to overall chip longevity and reduction in droplet merging, however, this combination was not available for the duration of the study as the RainDance^{TM} surfactant became unavailable (either commercially or through a Materials Transfer Agreement) due to the original providers (http://raindancetech.com/science/) removing it from the market to use with their proprietary technologies. To overcome this, some work was done to synthesise the surfactant in house but the cost became prohibitive.

As a substitute the majority of later work was performed using the FC-40 and 1H,1H,2H,2H-Perfluoro-1-octanol combination, which performed acceptably in most cases at preventing degradation of the microfluidic devices but was not as effective at preventing droplet merging.

The fluorinated oils and their corresponding surfactants appear to perform consistently better within the platform than their mineral oil counterparts. This is consistent with the current trend within the microfluidic literature and is likely due to FC-40's higher immiscibility with organic compounds (Holtze et al., 2008, Theberge et al., 2010).

6.3.3 Treatment Variants

Section 8.2.3 discusses a subsequent treatment step in the fabrication of the microfluidic devices that was possible (the application of Duxback) and was used when severe accumulation of droplet debris along the channel walls was observed. As a general rule treatment was applied to each of the platform iterations as a secondary investigation to assess whether or not the platform's performance improved.

6.3.4 Syringes, Tubing and Syringe Pumps

A variety of syringes were used to act as either inputs or outputs to the platform. Generally Hamilton syringes (Hamilton Company) with volumes from 50 – 500µl were used. Either fixed or disposable needles were used depending on the tubing required for the platform.

The tubing used depended on the type of chip that was currently being tested. Tubing was obtained from Cole-Parmer (www.colepalmer.co.uk) and had inner diameters raging from 100µm to 500µm**.** These choices were based on the type of droplet manipulation being performed, the types of detection used and the types of interface between the chip and the tubing.

Standard syringe pumps capable of pumping small volumes at low flow rates in both infuse and withdraw modes were used.

Syringe to tubing interfaces were generally direct insertion of the tubing into the syringes (or vice versa) and sealed either by wax or by melting the external tubing respectively. These techniques are flawed as the connection stability can vary greatly between experiments. Primarily these techniques are employed purely for their speed and flexibility.

Tubing to chip interfaces were either direct insertion into chip micro capillary tubing (corresponding to chip fabrication type 1 from section 8.2.2) or direct insertion into the chip itself (either perpendicular to or parallel to the channels within the chip, corresponding to chip fabrication type 2 from section 8.2.2).

6.4 Platform Iterations

6.4.1 Overview

As discussed previously, the next sections describe each of the platform designs that were tested as well as listing any variants. Variants, where pertinent are described in a table, providing a qualitative outcome description.

6.4.2 Platform Iteration 1: Long Chip Design

Platform Iteration 1 was designed in previous work but is listed here to demonstrate the full cycle of platform iterations performed. Notably, this design was also tested with two detectors, the latter of which was performed during this investigation. This iteration of the platform used an entirely on-chip set of geometries for droplet manipulation, including: reaction initiation, droplet formation, droplet incubation and droplet observation. Whilst this appears to be counter-intuitive with respect to the 'minimisation mind-set' discussed in section 6.1.2, it should be noted that each portion of the chip was designed separately and in a modular manner so that the geometry designs could be re-used throughout multiple chip iterations. Three variants of the chip were created, each with a different channel width (30µm, 50µm, and 100µm), allowing determination of the optimum channel geometry. Initial tests of the system were to verify the chip design's operability; as such the chip would be tested with purified GFP and an initial IVTT reaction containing the standard constitutive expression promoter as discussed in section 6.1.4.

The chips were fabricated as per section 8.2.2 and as such allowed direct connection of any inputs and outputs to any tubing. This type of fabrication was selected because it was considered the most robust and ensured a long chip life time which was important for the reaction incubation step.

500µl input syringes were used in conjunction with 100µm inner diameter tubing for the inputs into the chip and a single short stretch of tubing was used as the output. Initially this output tubing was connected directly to a waste reservoir but later it was also connected to a syringe set in withdraw (suction) mode. Flow rates of approximately 1µl per minute were used for the reactant inputs, the oil phase was set to approximately 2µl per minute and when applicable the output syringe was set to withdraw fluid at approximately 3µl per minute. Tubing was connected to the syringes by directly inserting the needle into the tubing and securing with some common wax.

As shown in Figure 7a, the chip contained a triple input that could be manipulated to allow different numbers of inputs into the chip by plugging the inputs with sealed tubing. When testing the platform with purified GFP a single input into the chip could be used; whilst testing with the basic IVTT reaction two inputs could be used to allow on chip reaction initialisation as per section 8.1.11.

Droplets were formed by using a relatively standard T-junction channel schema (Figure 7b) whereby the reaction mixture is entered into a continuously flowing oil phase at a perpendicular angle. Droplet formation using a T-Junction design represents a trade-off for ease of fabrication vs. long term degradation due to droplet debris accumulation. Later platform iterations used the more stable flow focussing channel geometries for droplet formation. An excellent discussion on the various droplet formation channel geometries can be found in Song *et al.*'s paper on 'Reaction in Droplets in Microfluidic Channels' (Song et al., 2006).

Short stretches of highly convoluted channel were also used after the droplet formation geometries (Figure 7c) as these promote chaotic advection (Song et al., 2006) within the droplets. Chaotic advection is important as a tool for mixing the reactants within the droplets ensuring their uniformity and allowing for accurate determination of reaction initiation (Song et al., 2006). Droplet incubation in this iteration is achieved through an extremely long serpentine channel Figure 7d, which consists of extended stretches of channel travelling across the chip and joined together by short 'U' bends. For the 50µm width channel design the length of the continuous channel was approximately 20m with a footprint equalling a microscope coverslip. This extreme length in combination with low flow rates of the oil phase allowed extended incubation times to be achieved on chip.

Detection of the droplets was achieved by recording the droplet fluorescence at a variety of points along the serpentine channel as the droplets arrived at that point. Detection was achieved through a confocal microscope in combination with an APD as described in section 8.3.1. Droplets were to be observed at a variety of points along the channel, roughly corresponding to 10 minute incubation durations. This allowed the generation of a time series whilst capturing data from all of the droplets (by ensuring that the next data set began recording in the next observation location along the channel). The positioning of the microscope was manually adjusted, based on the droplet travel distance, in order to obtain these observations.

Multiple oil and surfactant combinations were tested with this chip design and are listed in section 8.2.4. The IVTT reaction was initialised using the first IVTT reaction technique described in section 8.1.11. Purified GFP (section 8.1.10) was also tested in parallel at a concentration of 1µg per µl.

The runtime operation of the chip involved first ensuring that it was completely saturated with oil phase to reduce as much back pressure (the pressure exerted on the liquid phase, working against the pressure incurred from the input syringes) from the presence of air as possible. The chips were allowed to form a limited number of droplets (the number varied per test but ranged from ~10 to ~100) after which all of the inputs except for the oil phase were shut off. This allowed for a continuous flow paradigm where the droplets were constantly moving through the chip's channels.

Initial results for this platform iteration were not encouraging. Due to the length of the channel within the microfluidic device a great deal of back pressure built up eventually preventing any droplet movement within the channels. Attempts to ameliorate this pressure included adding a second syringe withdrawing fluid on the output allowing the device to completely fill with oil before forming droplets. However, the devices ultimately failed catastrophically usually by breaching sections between the serpentine channels or between the PDMS and the microscope slide. This back-pressure also led to inconsistent droplet formation and droplet sizes such that the device design was rendered inappropriate to achieve the initial specifications. All three of the channel widths fabricated in this iteration suffered the same issues, with the 100µm channel surviving the longest. It was also quickly realised that even with an on-chip channel length of 20 meters the duration of droplet travel from one end of the device to the other would not suffice in covering the initial specification of a 4 hour reaction duration (with actual durations ranging from approximately 30 minutes to 2 hours). Table 3 contains a summarised overview of some of the key iteration variants that were tested.

Table 3: A table detailing some of the key variants attempt within platform iteration 1. These attempts were sometimes informed by later work which lead to iterations being revisited.

Variant	Key Observations
Original	Droplets progress some distance before slowing to a stop. Long term operation resulted in consistent device failure
Withdraw syringe on output	As above, with an increased duration until failure
Duxback Treatment	No noticeable change on either of the above modes of operation
Sizes	100 µm channel survived the longest with 30 µm rarely even producing droplets in a stable manner
Oil pre-fill	Oil alone could not travel the entire length of the chip in any device variant

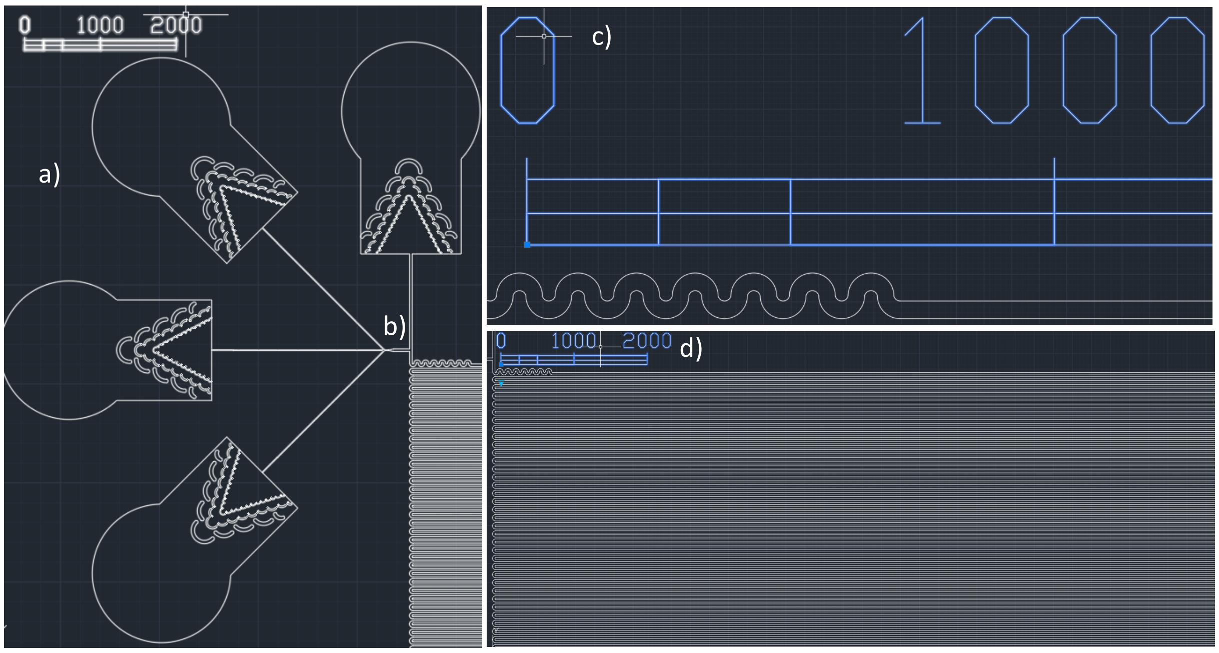


Figure 7: Salient design motifs used in the construction of platform iterations 1's microfluidic chip. Design a) The triple input geometry that allows on chip initialisation of reactions with additional room for marker reactions. Port usages from bottom left clockwise are as follows: GFP only, IVTT reaction mixture part 1, IVTT reaction mixture part 2 and finally the oil b) The T-Junction geometry used for droplet formation. c) The mixing channel designed to promote chaotic advection within the droplets. d) A portion of the long serpentine like channel used for droplet incubation.

6.4.3 Platform Iteration 2: Modular Chip Design

Platform Iteration 2 took a more modular approach by separating the droplet forming and droplet detection devices of the microfluidic geometries and linking them with tubing. Theoretically this allowed for long incubation times to occur between the chips (within tubing of a length corresponding to the incubation duration). Overall this set of designs provided greater flexibility and versatility in comparison to the previous iteration as multiple runtime types could be tested. Iteration 2 was also designed and tested in previous work, but was revisited in this investigation in order to inform Iterations 4 and 5. This revisiting included testing the already fabricated devices with different observation mechanisms (as detailed below).

Chip fabrication, oils, surfactants, tubing and syringes were almost identical to those employed in iteration 1 of the platform. The only difference was that chips were treated (as per section 8.2.3) from the outset in order to improve droplet formation stability and to minimise the accumulation of droplet debris that had been observed during platform iteration 1.

Chip inputs were identical to those used in Iteration 1 (Figure 7), i.e. employing a triple input design that could be modified as per the required number of inputs using sealed tubing. Reaction initiation was again performed on chip by merging two streams, the first containing IVTT and the second containing the GFP expression vector of interest. Droplet formation was performed using a T-Junction like geometry (Figure 9A). The inputs into the detection device were slightly different to those used in the previous iteration. As can be seen in Figure 9 (2 and 3) no input filters are used (these input filters normally help reduce impurities in the input solutions that would be detrimental to normal chip operation) as these would trap droplets within the inputs and prevent them from continuing through the chip channels.

The core difference between this design and iteration 1 is that droplets are incubated off-chip in the tubing that links the devices. This design improved upon the previous iteration in two core ways. Firstly, distinct observation points could be created by chaining detection devices (Figure 9B) and secondly, backpressure could be reduced within the channels by stopping the oil phase during incubation steps (generally corresponding to when the droplets are within the tubing linking devices) thus greatly reducing the length of the channel (on and off chip) required to achieve a four hour incubation period. Droplet content homogeneity was ensured by including mixing geometries before and after the droplets entered the tubing for incubation. This allowed droplets to be re-mixed prior to detection, ensuring that accurate sampling of droplet fluorescence occurred.

Detection of droplet fluorescence was performed using the APD confocal setup as described in section 8.3.1. To promote better sampling of the droplets an alternative detection device was designed (Figure 9C) that contained sections of channel that squeezed the droplets (Figure 9C - 1). These squeeze points represented where the point of detection (i.e. the microscope objective) would be located. Principally this squeeze point increased the volume of droplet sampled by the APD confocal setup, thereby improving the confidence in the data's representation of the reaction state. When this iteration was revisited, the fluorescence microscope was used to assess whether the detection portion of the designs could be used instead of iteration 4's designs.

With respect to runtime operation, the off-chip incubation step allows for oil flow to be stopped and incubation to occur without extending the channel length. Predominantly this methodology allows for the chaining of multiple microfluidic detection devices using linking tubing to achieve the same incubation specification as Iteration 1 (Figure 10), however, should droplet incubation within tubing prove to be successful, an alternative runtime operation mode could be employed. This alternative would be to incubate the droplets in the tubing linking the formation and detection devices, passing them across the objective, and then reversing the flow, allowing the droplets to be incubated again in the same stretch of tubing. This would greatly decrease the complexity (the number of devices) of this iteration as well as potentially eliminate any of the previously observed backpressure issues.

When device chaining was first tested, several key issues became apparent. Firstly the backpressure in the channel was still present, leading to inconsistent droplet formation and extreme difficulty when filling the device with oil. Chained detection devices also led to instability as either the stage or the device had to be moved to the point of observation, which directly resulted in some droplets merging within the tubing or the chain failing completely and becoming disconnected. Reducing the number of chained devices increased the stability of the flow and droplet formation consistency improved to a degree. Upon incubating droplets and pushing them to the first detection device it became apparent that droplets were aggregating in the input well of the detection device and coalescing such that the FIFO droplet ordering could not be guaranteed.

Despite this, it was decided to perform a further set of experiments using a single detection device setup to form droplets and allow incubation. Any aggregate or single droplets could then be pushed into the squeeze section in order to observe a time course of GFP expression as a proof of concept. After allowing an aggregate droplet to sit within the point of detection over a long period of time (corresponding to approximately an hour and detecting the droplet's fluorescence every 5 minutes) it was discovered that little-to-no fluorescence was being observed. To confirm the observed results above a series of reactions were setup in microplate format to mimic the reaction initiation steps performed in Iterations 1 + 2. Essentially this was to test whether the dilution of the IVTT mixture that occurs through the mixing of two inputs within the microfluidic device prevented or at least severely slowed down GFP production. Figure 6 shows the outcome of this test. As theorised, a one in two dilution of the cell-extract portion of the IVTT led to little or no observable GFP fluorescence over time.

When re-visited, this iteration was tested using GFP alone in order to assess whether or not part of the design could be reused. Whilst individual GFP droplets were possible to observe at early time points, repeated passes of the droplets through the input or output ports quickly lead to uncontrolled droplet coalescence and loss of droplet identity. A set of images taken during the early time points of this test can be found in Figure 8.

Table 4: A table detailing some of the key variants attempt within platform iteration 2. These attempts were sometimes informed by later work which lead to iterations being revisited.

Variant	Key Observations
Original	Backpressure issues were still present, chip module interfaces were likely points of failure
Single detector chip	No backpressure issues, but droplet coalescence and loss of identity was observed at the input and output ports as well as no fluorescence production
Sizes	100 µm channel proved to be the most stable and encountered a small enough resistance at squeeze points unlike its smaller counterparts
Revist	This iteration was later revisited when the lack of fluorescence production was overcome using other input mechanisms but the droplet ordering issues could not be overcome
Revist + Raindance surfactant	Uncontrolled droplet coalescence was reduced unless the ports became blocked. FIFO issues could not be overcome here.

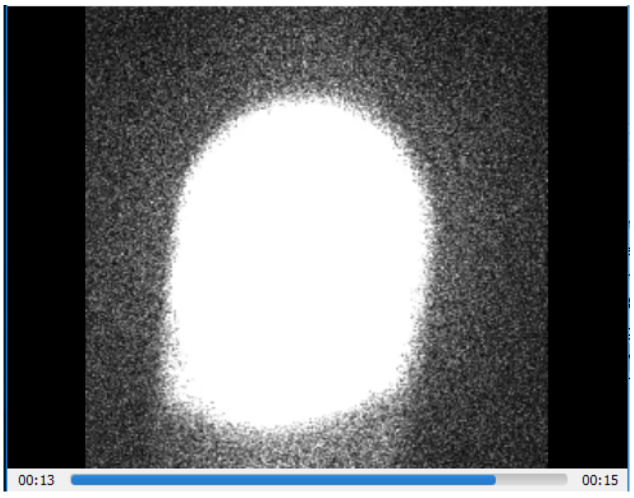


Figure 8: An example of a GFP droplet being observed as it passed through standard 100µm channel. The GFP concentration was 1µg/µl and the image was acquired at 40x magnification.

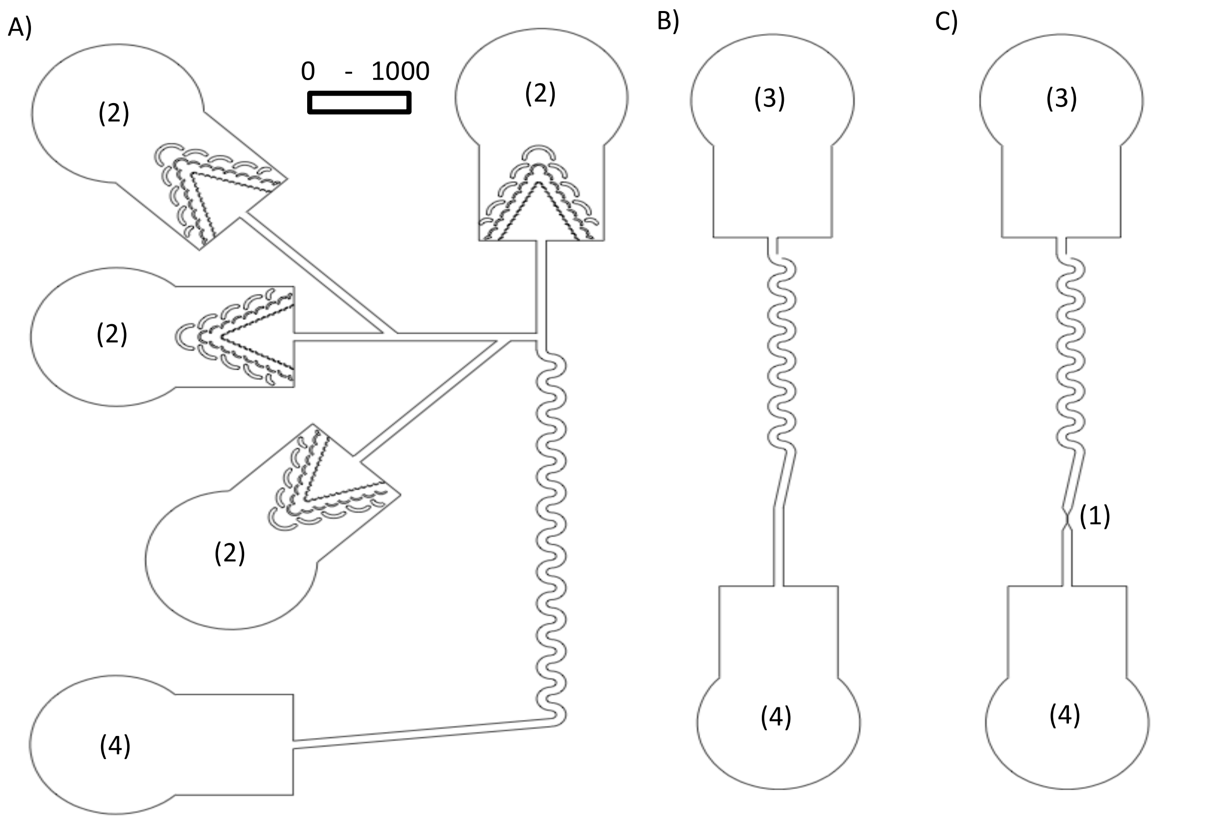


Figure 9: Platform iteration 2 chip designs. A) Droplet Formation Module in which the ports were used in the same manner as Iteration 1, B) Droplet Detection Module 1, C) Droplet Detection Module 2 with indicated Squeeze point (1). (2) The original filter inputs. (3) The inputs connection without filters. (4) The output connections.

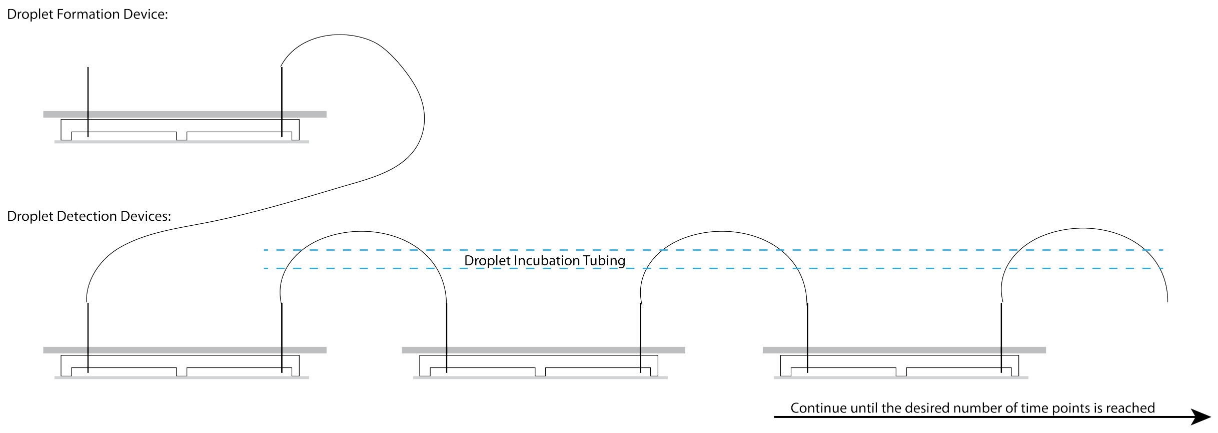

Droplet Formation Device:
Droplet Detection Devices:
Droplet Incubation Tubing
Continue until the desired number of time points is reached

Figure 10: Diagram depicting the chaining of platform iteration 2's detection chips. This allows high numbers of observation points without long internal microfluidic channels.

6.4.4 Platform Iteration 3: Trapping Chip Design

Platform iteration 3 took a different approach to droplet incubation in order to overcome the backpressure issues observed in both platform iteration 1 and 2. The microfluidic chip was designed to store droplets in specific microfluidic geometries rather than passing them across a fixed point of observation. As a result the detection methodology was more similar to that of iteration 1 whereby the point of observation would change, however, rather than representing the moment of observation along a time course assay, each observation would now represent a single droplet at that moment.

To increase the turnaround of chip design and construction a new fabrication technique was adopted. This fabrication technique (section 8.2.2 Type 2) reduced the stability of the finished product but greatly increased the speed at which new chips could be tested. As noted in section 8.2.2 (Type 2) several additional fabrication techniques could be applied to vary the finished product. The greatest speed was achieved by bonding the device directly to a thin layer of PDMS and therefore this technique was used predominantly to perform initial tests on a platform design. If the device proved to work under the most basic of conditions device fabrication was altered to bonding directly to a microscope coverslip.

To overcome the reaction initiation issues identified in iteration 2, IVTT and GFP expression vector DNA were now pre-mixed in the syringes beforehand. This means that only oil and the pre-mixed reactions were inputs into the platform, greatly reducing the overall complexity of the experimental methodology.

Droplet formation was performed using a modified T-Junction geometry (Figure 11a) to help alleviate some of the perceived inconsistencies in droplet size and quality. The new geometry allowed for better fluid pressure equalisation between each of the inputs using hydrodynamic focussing (Song et al., 2006) and a more focussed droplet budding area.

Droplet incubation was achieved by reproducing the cup-like geometries as previously described by Niu *et al.* (Niu et al., 2009) in which droplets would be stored progressively as they are formed (Figure 11b). Droplet progression through the cup geometries is achieved through a build-up of pressure on the captured droplet from the droplet following it.

Detection of droplet fluorescence was again achieved through the APD confocal microscope setup, however this device iteration lacked the squeeze points designed in platform iteration 2. The removal of the squeeze geometry from this design arose from the lack of droplet movement; as droplets are kept stationary within the cup-like geometries squeeze points were not feasible. Detection of each droplet involved moving the microscope stage to each droplet's corresponding location.

Despite the droplets remaining stationary within the device the oil phase of the platform could remain flowing; however droplet formation was limited to the number of cup-like geometries available and so the reaction phase of the platform would have to be stopped to prevent further droplets from forming, therefore pushing all of the droplets into their next cup. Thus the runtime operation of the platform was similar to that of a stop-flow mode of operation.

Issues with this device design arose quickly as rather than droplets pushing the preceding droplet out of the cup-like geometry, either droplet merging or shearing of the droplet through the pillars occurred. This behaviour meant that the cup-like geometries were never filled with droplets. At most, only a run of five cups could be filled before complete device failure. Both of these scenarios indicate a failure in the iteration as FIFO and therefore droplet identity cannot be maintained. Other trade-offs were also present within this design such as the inherent limitation on the number of droplets that could be formed (limited by the number of droplet cups available for occupation).

Reaction initiation within the syringe also represents a serious concern for the platform as it has several drawbacks. These include an increase in the amount of IVTT used per characterisation assay and a greater difficulty in reaction multiplexing. Finally the amount of time that a reaction has been allowed to proceed is convoluted by the reaction being initiated off-chip (and not at the moment of droplet formation) and by the moment of observation which is greatly affected by the need to move the microscope stage.

Table 5: A table detailing some of the key variants attempt within platform iteration 3. These attempts were sometimes informed by later work which lead to iterations being revisited.

Variant	Key Observations
Original	No backpressure issues observed, but droplet shearing and coalescing occurring at the traps
Raindance surfactant	Reduced but did not eliminate the coalescence of droplets at the trapping geometries. It also improved stability of droplet formation to a small degree but was still inefficient at producing homogenous droplets
Duxback	This variant used surface treatment to increase the hydrophobicity of the chip surface. In theory this should have increased the proclivity of droplets to remain together and not accrue within the geometries, but it had very little effect on the iterations performance

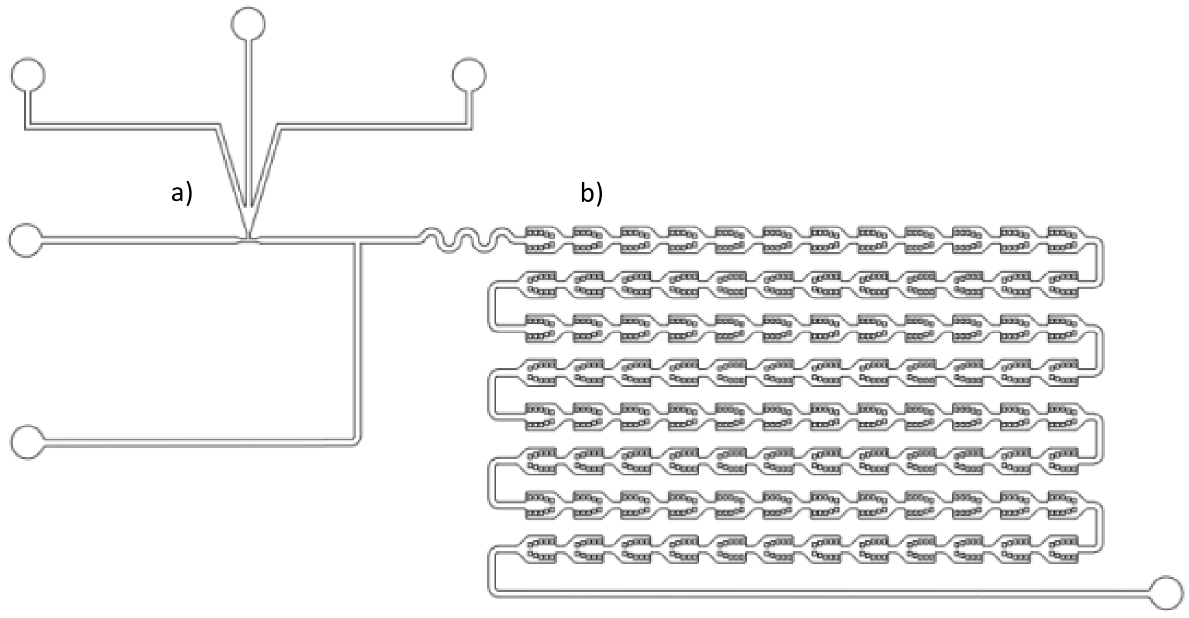


Figure 11: Platform iteration 3's microfluidic chip designs. This device uses improved droplet formation geometries at (a). Also present are droplet capture geometries at (b). The input ports for the device are as follows (clockwise from the bottom left port): pressure release tuning, oil phase, IVTT reaction, GFP, unused (plugged).

6.4.5 Platform Iteration 4: Parking Chip Design

Platform iteration 4 discarded the concept of droplet trapping and instead attempted to slow droplets and then 'park' them in a linear order before the point of detection. Droplets could then be passed through the point of observation at which point they encounter a secondary 'parking zone' which again attempts to preserve droplet ordering. The device shown in Figure 12 was designed specifically to allow accurate characterisation of a single reaction sample; once this is established higher throughput and multiplexing amendments could be investigated. A second version of this device was designed without the flow focussing geometries; this design was intended to be used with a newly developed compartment-on-demand robotic front end for droplet formation (Gielen et al., 2013).

Chip fabrication Type 2 was used to create initial device prototypes. Modifications were made to this fabrication step during the later stages of platform iteration 4 testing, whereby the PDMS chip portion was plasma bonded directly to coverslips as opposed to a thin layer of PDMS. The reasoning behind this change was to allow the confocal microscope's objective lens to better focus on the centre of the microfluidic channel, improving the sampling of the droplet's contents.

Reaction initiation was again initiated off-chip in the input syringe, however the geometries used for droplet manipulation ('parking') potentially allowed for another method of reaction initiation that could be further investigated if iteration 4 of the platform proved to be functional.

Droplet formation was performed using a flow focussing geometry (Figure 12B) in order to improve its consistency. The flow focussing geometries employ a four way junction with sample entering the junction from the top and oil entering the junction from both of the sides (leaving droplets to exit at the bottom channel). This technique has been shown to improve droplet formation under certain circumstances as well as reduce the 'wetting' of the channel walls (Srisa-Art et al., 2007) (which occurs often in T-junction geometries along the channel wall opposite the solution input channel).

As shown in Figure 12C the channel on both sides of the point of observation have been couched with a pillar like geometry. This geometry allows oil to pass freely around droplets between the pillars, which cause droplets to 'park' between the pillars. The pillar geometries would also not result in the droplet shearing that was observed during iteration 3, as no pillars are directly in the droplets' path. Once the pillar geometry is full of droplets and a new one attempts to enter the pillared zone, the droplet closest to the exit of the pillars will be pushed through the point of observation. This device's geometries were based around those used in Niu *et al.*'s paper on Electro-Coalescence of Digitally Controlled Droplets (Niu et al., 2009) which utilised similar droplets but also imparted an electric field to cause droplet coalescence. The premise behind this choice of geometry was to allow later iterations of the platform to control droplet coalescence in order to facilitate multiplexing within the platform by separating the IVTT and expression vector solutions into discrete droplets. This would allow multiple samples to be input into the device and reaction initiation to be controlled on-chip.

Droplet detection was again performed using the APD confocal microscope setup in combination with squeeze point droplet geometries for better sampling of droplet contents. The re-introduction of the squeeze points is a reflection of the device's runtime operation mode.

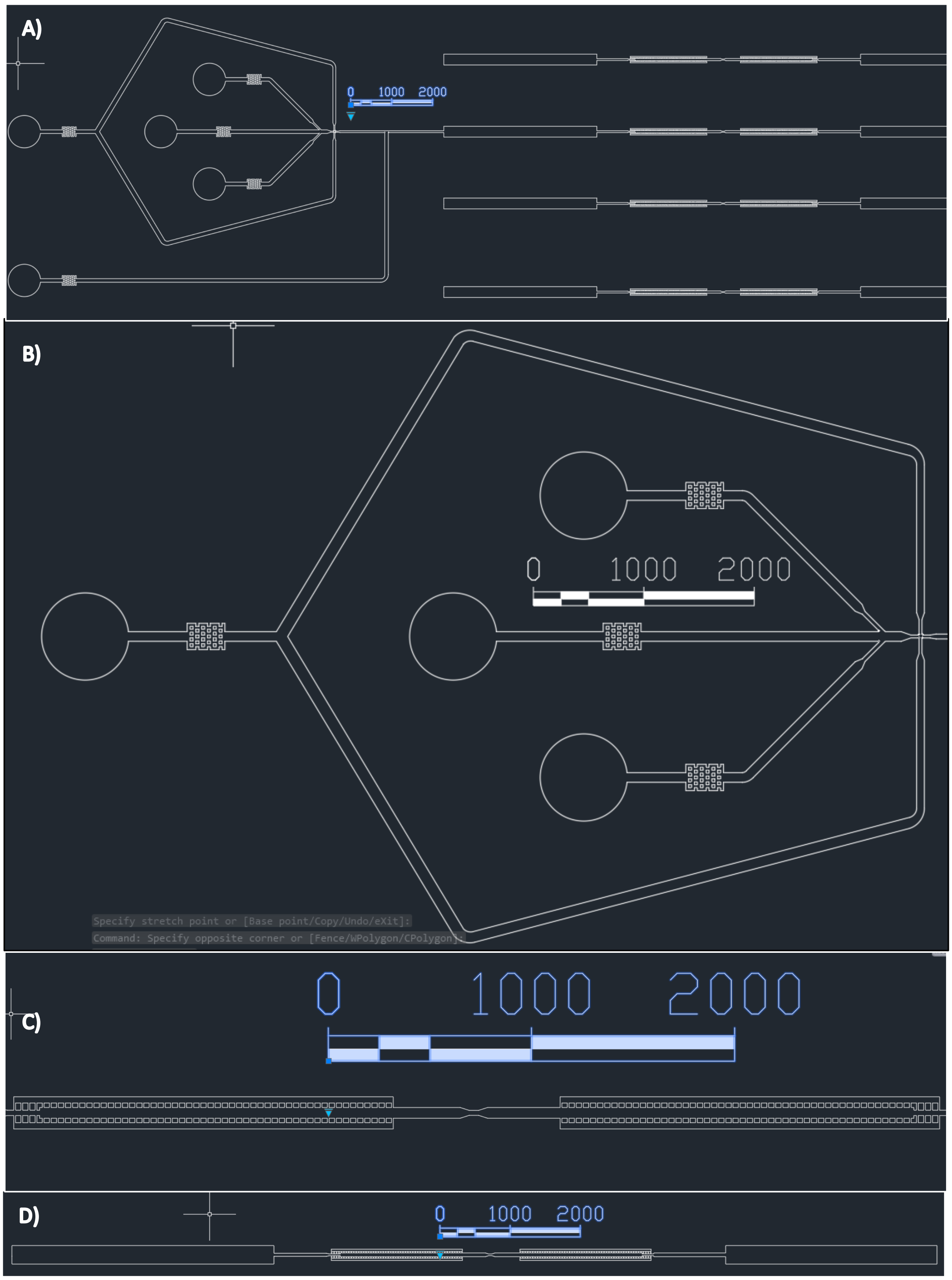
Device runtime operation would essentially follow a stop-flow methodology whereby droplets were allowed to fill the parking zones but the oil phase would be stopped before the first droplet in the pillar geometry moved to pass through the point of observation. This would allow for extended periods of incubation within the pillar geometries before each observation.

The flow focussing geometries used in this device design improved the stability of the device as little-to-no accumulation of droplet debris was observed during droplet formation. However, once used, the device still needed to be thoroughly cleaned before it could be re-used. Droplet formation was still inconsistent with a wide range of droplet sizes being observed with an incoherent periodicity in their formation. This is likely due to the highly viscous nature of IVTT which would rather aggregate as opposed to separate. The pillar geometries of the device functioned well in slowing the droplets down but leaving them parked in such close contact quickly resulted in unpredictable droplet coalescence, again this is likely due to IVTT's viscosity.

Despite these observations it was decided to attempt to characterise a time course fluorescence production profile for the BBa_J23100 sample mentioned in section 6.1.4. To achieve this, droplets were continuously formed and recorded as they passed across the point of observation. The assay lasted for approximately an hour before the droplet formation geometry had degraded to the point where droplets were no longer being formed. Figure 13 shows the results of this assay whereby data points represent averages of the droplets that had passed the objective within thirty seconds. An increasing trend can be seen in fluorescence over time indicating that when some of the more fundamental issues in the platform are addressed a viable characterisation technique may be realised. The variability in the data is extremely high across the time profile and is due to the inconsistencies in the morphologies of the droplets being observed (as a result of the deteriorating droplet formation and the inconsistent droplet merging occurring at the parking geometries). The inconsistency in the data quality is also representative of a fundamental flaw in this (and the previous) iteration of the platform: the APD confocal microscope samples an extremely small volume of the droplets. This small sampling volume is an artefact of confocal microscopy which uses point illumination and a pinhole to eliminate out of focus signals (Minsky, 1988). The additional module (designed for use with the compartment-on-demand robot) was not used in this iteration as the fundamental detection issues had to be addressed first. The low increase in fluorescence is likely due to the cold reaction and lack of mixing occurring within the syringe.

Table 6: A table detailing some of the key variants attempt within platform iteration 4. These attempts were sometimes informed by later work which lead to iterations being revisited.

Variant	Key Observations
Original	Inhomogeneous droplet formation and occasional droplet coalescence
Raindance surfactant	Little to no droplet coalescence



```
Specify stretch point or [Base point/Copy/Undo/eXit]:
Command: Specify opposite corner or [Fence/WPolygon/CPolygon]:
```

Figure 12: Salient design motifs used in platform iteration 4's microfluidic device design A) The Standard Device as a whole. B) Flow Focussing Geometry, oil is input from the left most port and the other ports are used from top to bottom as: IVTT mixture, GFP, nothing (plugged) C) Droplet Parking Geometry. D) Version of Iteration 4's device design without a droplet formation region.

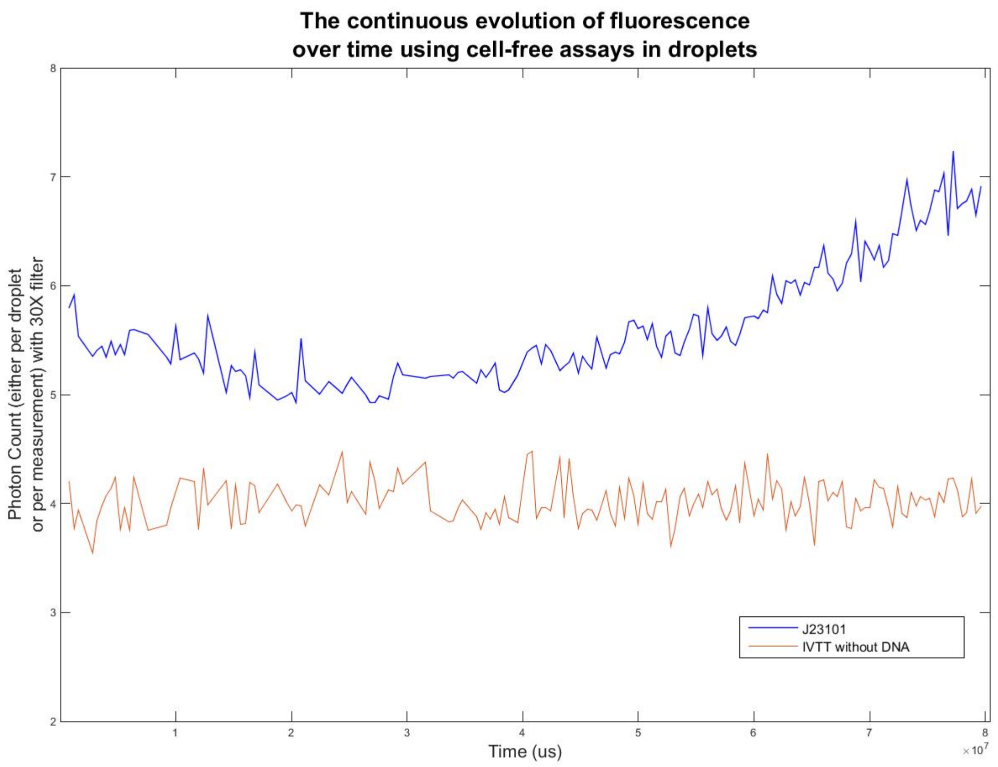


Figure 13: Graph to show the production of fluorescence against droplets number. Results are obtained from continuously produced droplets that contain IVTT and BBa_J23100 expression vector which implies that each sampled droplet corresponds to a 50µs snapshot of a time course dataset. The two expression profiles were obtained over the course of separate runs to prevent droplet coalescence.

6.4.6 Platform Iteration 5: Chip-less Design

Platform iteration 5 attempted to greatly reduce the complexity of the platform by removing the microfluidic chip altogether and work with the newly developed compartment-on-demand microfluidic robot (henceforth called the droplet robot). This reduced some of the stability issues associated with previous iterations as there are no tubing to chip interfaces present. These interfaces were constantly problematic throughout previous iterations as they represented points of failure that were always subject to small environmental fluctuations such as placement and the quality of seals. A diagram representing this iteration's overall design can be found in Figure 14.

Device fabrication was performed by adhering the tubing directly to microscope coverslips using a small amount of PDMS. This reduced the scattering of light from the cylindrical surface of the tubing and prevented movement during observation.

Reactions were initiated inside PCR tubes which in turn were placed on the droplet robot's carousel. Droplet formation occurred by passing the tubing end between the oil and surface interface (deMello, 2006). The robot used in this iteration was an early prototype of that published by Niu *et al.* (Gielen et al., 2013) and therefore could be tweaked in a variety of ways to improve droplet formation consistency.

Droplets were then 'sucked' towards the point of observation using a syringe set to withdraw mode (rates varied between 2 – 5µl per second). Incubation of the reactions was achieved by stopping droplets prior to the point of observation (as per iteration 4). The tubing used to form and incubate the droplets had an inner diameter of 100µm corresponding to the channel width diameters used in previous assays.

Detection of droplet fluorescence was performed using the wide-field fluorescence microscope as described in section 8.3.2. Images were captured at 500µs intervals as the droplets passed by the point of observation in order to ensure that droplet identity could be established within the image set. The move to wide-field detection methods was to improve the overall amount of the droplet sampled during observation, thereby ensuring that no sampling inconsistencies were embedded in the datasets.

Platform runtime operation followed a cyclic approach whereby droplets would be passed across the point of observation for fluorescence detection. After detection, droplets would be returned to their original position (by placing the oil phase syringe in infuse mode) and allowed to incubate for 10 minutes. The process would then be repeated to obtain a full time course dataset.

This platform iteration improved the stability and consistency by removing much of the complexity previously present. A large amount of time was invested in calibrating the prototype version of the robot to form droplets from the highly viscous IVTT. These calibrations included creating custom sample selection heads (Figure 14) to facilitate the type of tubing used in this assay, as well as custom software (see section 8.2.5 – full source available in the supplementary materials) to allow the droplet robot to function without the proprietary LabVIEW software (National Instruments Corp.). After droplets were successfully formed a time course assay verifying the production of fluorescence from droplets, encapsulating the expression vector containing BBa_J23100, was performed. Figure 15 shows the outcome of this assay as analysed according to section 6.5.3. Whilst the outcome of this assay included viable fluorescence data, droplet merging still occurred within the tubing and can likely be attributed to either surfactant insufficiencies (at this point in time the RainDance^{TM} surfactant was unavailable) or the flow instabilities created by changing droplet directions.

Despite this, efforts were made to investigate how a five reaction sample run would function with the current platform iteration. Whilst some image data was collected, a variety of issues prevented the information from achieving the overall platform aims. Firstly, the merging of droplets inferred that droplet identity could no longer be ascertained, in turn preventing any acquired image data being associated with a particular reaction sample. This issue was further confounded by the droplet robot inconsistently forming droplets indicating that the original repeat number per sample could not be established. Secondly, the platform iteration began to fail over time from backpressure issues (the tubing length used in this assay was approximately 1.5m and could not be reduced with this version of the droplet robot). These backpressure issues presented in either flow failure throughout the system or seal failure at the oil phase syringe.

Figure 16 shows data that was acquired in a mixed promoter run. Originally each sample contained 10 droplets separated by a marker droplet with a large spacing either side. Whilst droplet merging occurred within each sample type the spacing and markers between sample allowed for sufficient separation to occur to prevent identity loss. This demonstrates some of the potential multiplexing ability of the compartment on demand robot although with a loss of statistical confidence as there are now no reaction repeats.

Table 7: A table detailing some of the key variants attempt within platform iteration 5. These attempts were sometimes informed by later work which lead to iterations being revisited.

Variant	Key Observations
Original	Backpressure issues
Raindance surfactant + Duxback	Reduced backpressure, observed droplet merging over the cyclic flow

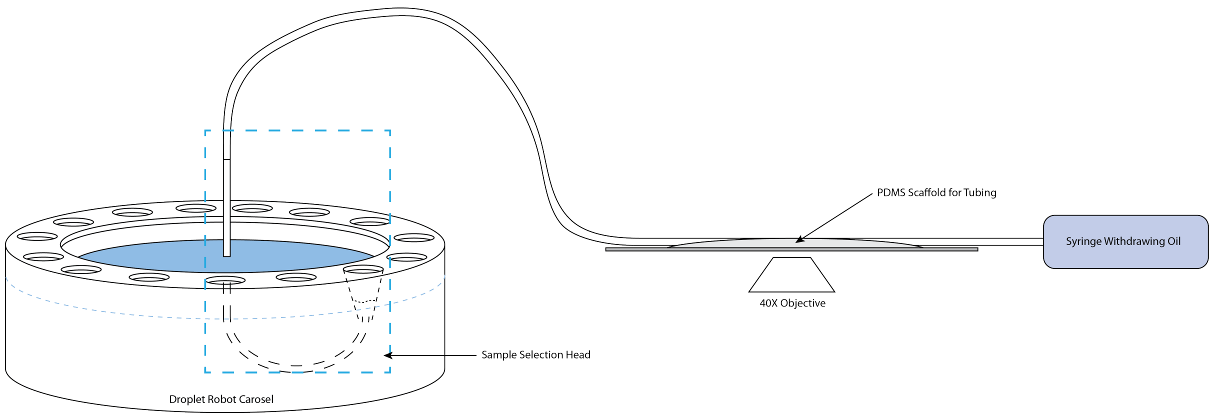


Figure 14: Schematic of platform iteration 5's overall design. Operation is performed by withdrawing oil / samples from the robot until the desired amount of droplets are formed and then cycling the droplets across the point of observation

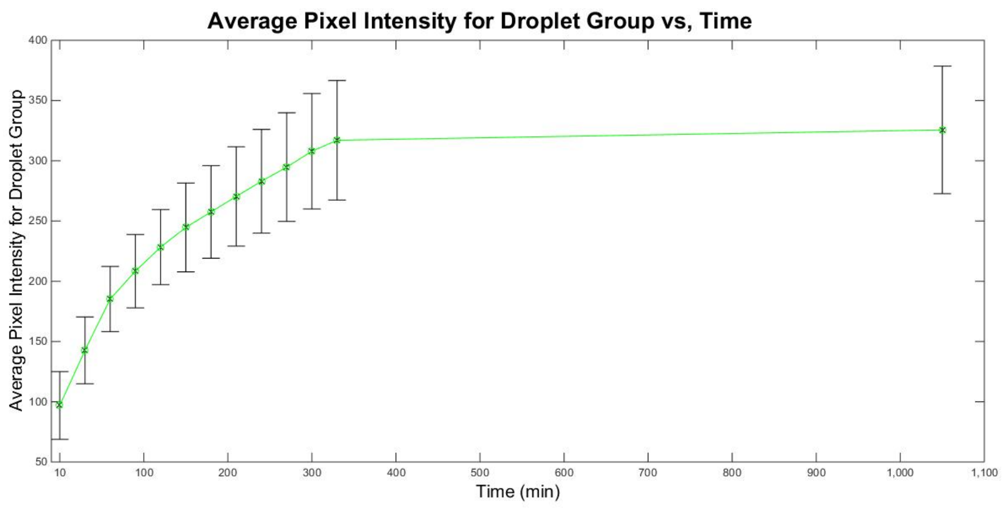


Figure 15: Graph showing the production of fluorescence over time. Fluorescence is recorded as pixel intensity. Droplets containing expression vector BBa_J23100. Error bars are the standard deviation of the averaging operations of the droplets. The number of droplets used per observation point to calculate the average fluorescence is 5

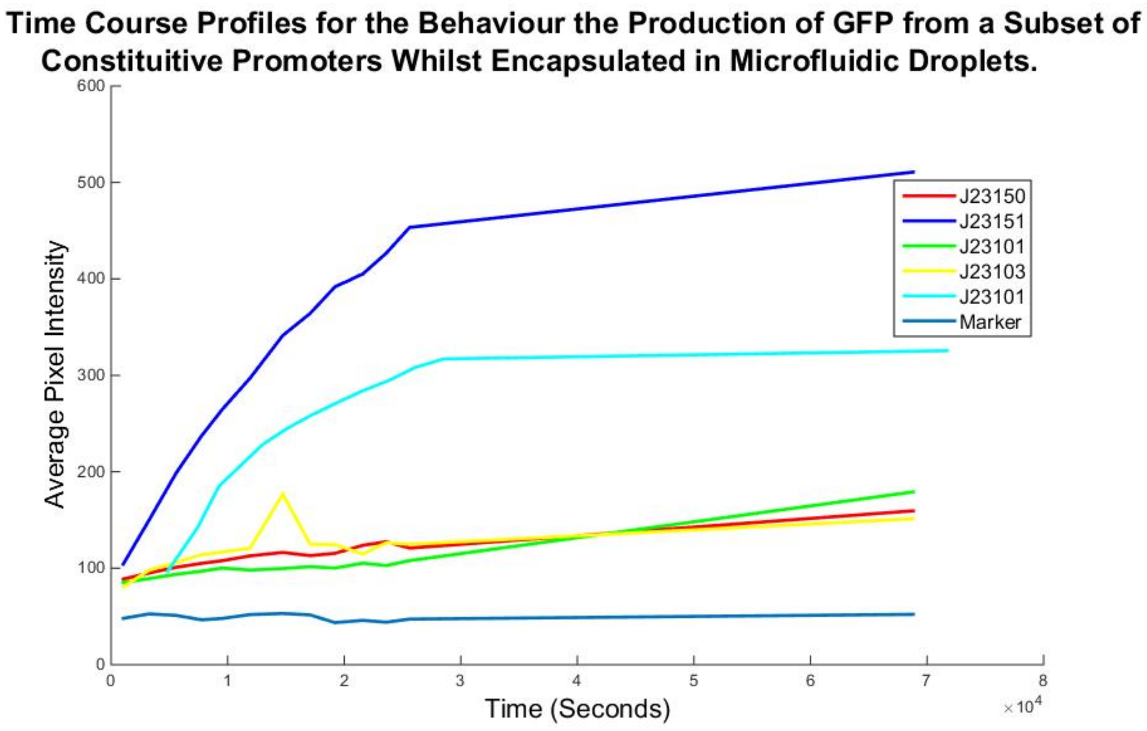


Figure 16: Graph showing the production of fluorescence over time for a variety of promoters. Due to droplet merging, only a single representative and clearly identifiable sample was used to determine the sample's fluorescence. The background levels were measured in a droplet containing IVTT and tracer dye but no expression vector.

6.5 Droplet Data Analysis

6.5.1 Overview

Two key types of data were produced over the course of the microfluidic experiments, either from the APD or from the fluorescence microscope. These data sets can be quite complex to extract meaningful information from, especially given the dynamic and heterogeneous nature of the samples. Microfluidic droplets are not only (usually) mobile, but their shapes and sizes can be non-uniform in these prototype devices. The next two sections detail the full data extraction processes that were created to handle these data types. A note for all readers, commentary detailing the operations of a MATLAB script can be found in green.

6.5.2 Data Analysis for Microfluidic Droplets Detected Using an APD

In all cases the APD was set to record photon counts every 50µs. Any data analysis script would have to identify where the droplets were located in this dataset and to average the intensity for the identified droplet. A threshold would also be required to remove noise from the system as small particles and imperfect chips would result in background fluorescence. To achieve this, the data analysis was broken down into a series of steps:

1) Load data from native format into a MATLAB compliant format
2) Use a standard low pass filter to remove base noise in the data
3) Data trimming
4) Threshold the data for meaningful information
5) Remove anomalous troughs from the data that are likely to affect droplet identification
6) Remove anomalous peaks from the data that are likely to affect droplet identification
7) Identify droplets candidates within the data set
8) Retrieve core results from data candidates

Each of these steps is further detailed below with sample scripts for how they were performed and an example data set (taken from actual raw data) with depictions of how the data set is manipulated.

6.5.2.1 APD Data Loading

The script responsible for step 1 attempts to load raw APD data (in .dat format) from its normal file form into the MATLAB environment. It is relatively simple in MATLAB as many raw data types are handled automatically. The data output from the APD is a line separated (i.e. new data is on a new line in the file) numeric format to 3 decimal places (although photon count is always an integer). To handle this file appropriately ‘dlmread’ was used (a native MATLAB function) to read the files and infer their delimiter (what separates the data) automatically. The script and an example of loaded data are shown in Figure 18. Time values for each observation are calculated by multiplying the number of the observation by 50 (corresponding to 50µs, the sampling frequency of the APD).

```
%% Function for droplet data loading
% Function for loading droplet data into a single column of data with the
% corresponding time values in the second column
function [output] = LoadDropletData()
    % Get file
    [fileName, path] = uigetfile('*.dat');
    if isequal(fileName, 0)
        error('No file was selected')
    end

    % Read the file and build the dataset
    file = fullfile(path, fileName);
    data = dlmread(file);
    time = (1 : size(data, 1))’ * 50;

    % Output the dataset
    output(:, 1) = data;
    output(:, 2) = time;
end
```

Figure 17: A function for loading data produced by the APD into a MATLAB working environment

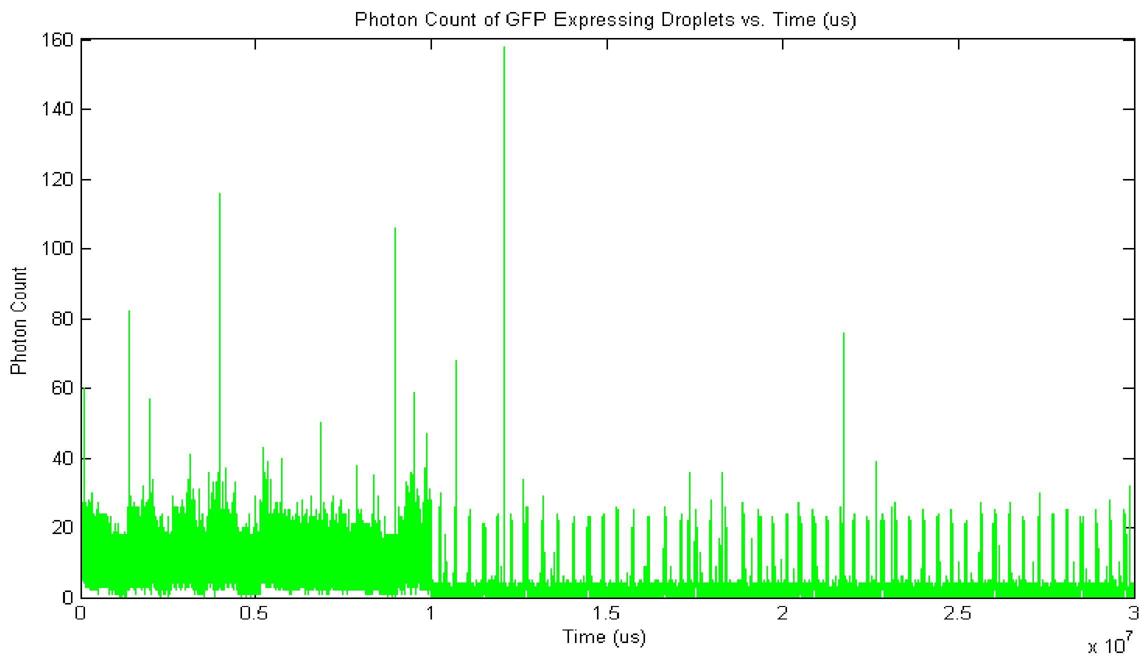


Figure 18: Example data for raw APD data having loading into a MATLAB native format corresponding to step 1.

6.5.2.2 Filtering the APD Data

Filtering the data allows us to transform the source by looking for a signal that corresponds to frequencies lower that a certain cut-off. In actuality, this allows us to remove noisy, highly variable data in favour of the slower more episodic signal that represents the passing of droplets across the objective. Filtering the APD data corresponds to step 2 in the above list. An example of the processed data is shown in Figure 19, the input for which was the outputs obtained from the previous step. The filter properties were generated using MATLABs Signal Processing Toolbox^{TM} (required for this portion of the script) but are generally tuned to ignore (or reduce) highly variable fluctuations that are representative of noise within the detection system.

```
%% Function for applying a low band pass filter to droplet data
% Expected input is the output from LoadDropletData
% Note this function requires the signal processing toolbox
function [outputData] = FilterDropletData(inputData)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(
        error('Unknown input data');
    end

    % Only apply filter to the data not to the time
    actualData = inputData(:, 1);

    % Apply the filter (detailed below)
    resultantData = filter(LowPassFilter, actualData);

    % Ensure all data is positive
    resultantData = (resultantData + abs(resultantData)) / 2;

    outputData(:, 1) = resultantData;
    outputData(:, 2) = inputData(:, 2);
end

%LOWPASSFILTER Returns a discrete-time filter object
% M-File generated by MATLAB(R) 7.0 and the Signal Processing Toolbox 6.2.
% Generated on: 28-Jun-2009 16:59:16
% Equiripple Lowpass filter designed using the FIRPM function.
% All frequency values are in Hz.
function [Hd] = LowPassFilter
    Fs = 20000;              % Sampling Frequency
    Fpass =  300;            % Passband Frequency
    Fstop =  400;            % Stopband Frequency
    Dpass = 0.057501127785;  % Passband Ripple
    Dstop = 0.0001;          % Stopband Attenuation
    dens  = 20;              % Density Factor

    % Calculate the order from the parameters using FIRPMORD.
    [N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass, Dstop]);

    % Calculate the coefficients using the FIRPM function.
    b  = firpm(N, Fo, Ao, W, {dens});
    Hd = dfilt.dffir(b);
end
```

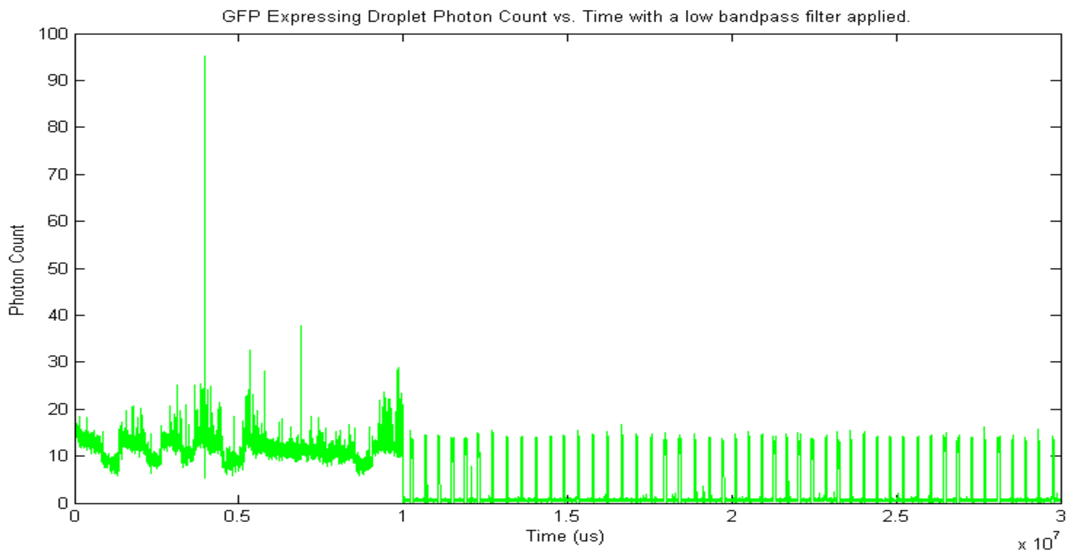


Figure 19: Above shows a script to perform low band pass filtering on APD data (step 2). The script is divided into two sections with the first managing the data and the second applying the filter. The script thresholds were generated automatically by MATLAB's Signal Processing Toolbox. Below shows example data after the application of a low-pass filter. The source for this data was the inputs displayed in step 1.

6.5.2.3 APD Data Trimming

Trimming the data (step 3) to remove any extraneous information that would affect droplet identification is essential. Data sections that need to be trimmed include background oil or any marked droplets used. Figure 20 shows the script and an example data set (the input of which was the results from the last step). This step is user curated and requires visual feedback therefore a short graphical user interface handling script is used before the data is trimmed to identify its location. This script can be found in section 10.2 with an example of the process shown in Figure 61. The reason this step is applied after the low band pass filter step is to allow better identification of the regions of data to be trimmed.

```
%% Function for trimming droplet data
function [outputData] = TrimDropletData(inputData, headFrom, headTo, tailFrom, tailTo)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(1, 1))
        error('Unknown input data');
    end

    outputData = inputData;

    if tailFrom ~= length(inputData)
        outputData(tailFrom : tailTo, :) = [];
    end

    if headTo ~= 1
        outputData(headFrom : headTo, :) = [];
    end
end
```

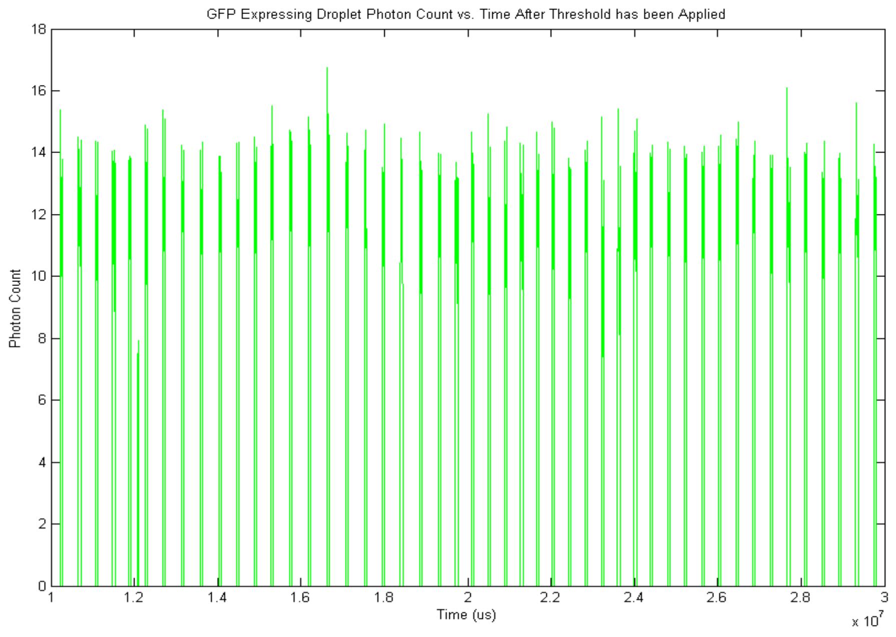


Figure 20: Above shows a script responsible for enacting step 3 of APD Data Processing. Whilst below displays example data for the trimming of data, corresponding to step 3. This example uses the data from the previous step and displays the outcome after it has been processed*APD Data Thresholding*

The 4th step is to apply a threshold to all of the data. This removes any background photon counts that were observed from the oil phase and any impurities within it. The example shown in Figure 21 has a threshold of 5 photons per observation applied to it (this was a general threshold used throughout this study except for when analysing non fluorescent data). It can be seen that much of the background noise between the pronounced peaks has been removed, which greatly eases droplet identification.

```
%% Function for applying a threshold to droplet data
% Apply a threshold to the dataset using the provided photon count. Any
% values below this threshold will be set to 0 as they are considered noise
% Data provided should be in the same form as the output from droplet
% filtering
function [outputData] = ThresholdDropletData(inputData, threshold)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(1, 1))
        error('Unknown input data');
    end

    % Check the input
    if ~isnumeric(threshold) && threshold <= 0
        error('Invalid threshold value')
    end

    % Apply the threshold
    inputData(inputData(:, 1) < threshold) = 0;
    outputData = inputData;
end
```

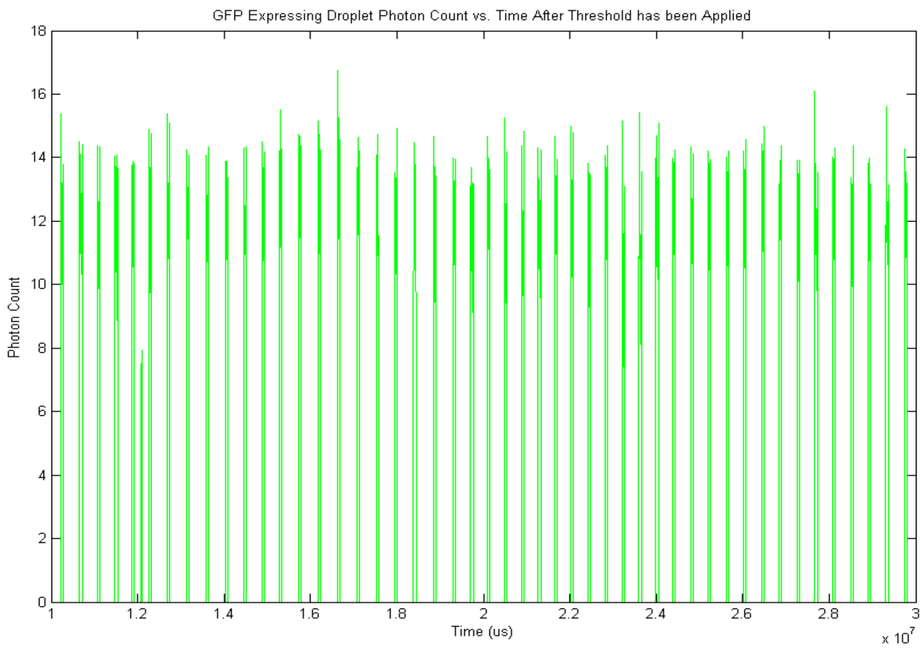


Figure 21: Script and example data for the application of a threshold to the data, corresponding to step 4. Above is the script responsible for applying the threshold. Below is the sample data used in the previous step after it has had a threshold applied.

6.5.2.5 Erroneous Trough Identification in APD Droplet Data

Step 5 is to identifies stretches within the dataset that contain no data and are below a given threshold in length (i.e. if a threshold of 5 is applied, then stretches of 5 data points or less are identified). This script (Figure 23) essentially identifies erroneous troughs in the dataset that may arise from heterogeneity in the droplets' morphologies. A threshold of 5 was generally used and corresponds to 250µs of time passing. Such a short timespan between droplets is extremely unlikely and therefore can be attributed to the aforementioned heterogeneity. These troughs are important to identify as they will affect droplet identification. An example data set (having also been processed using the previous steps) can be found in Figure 22.

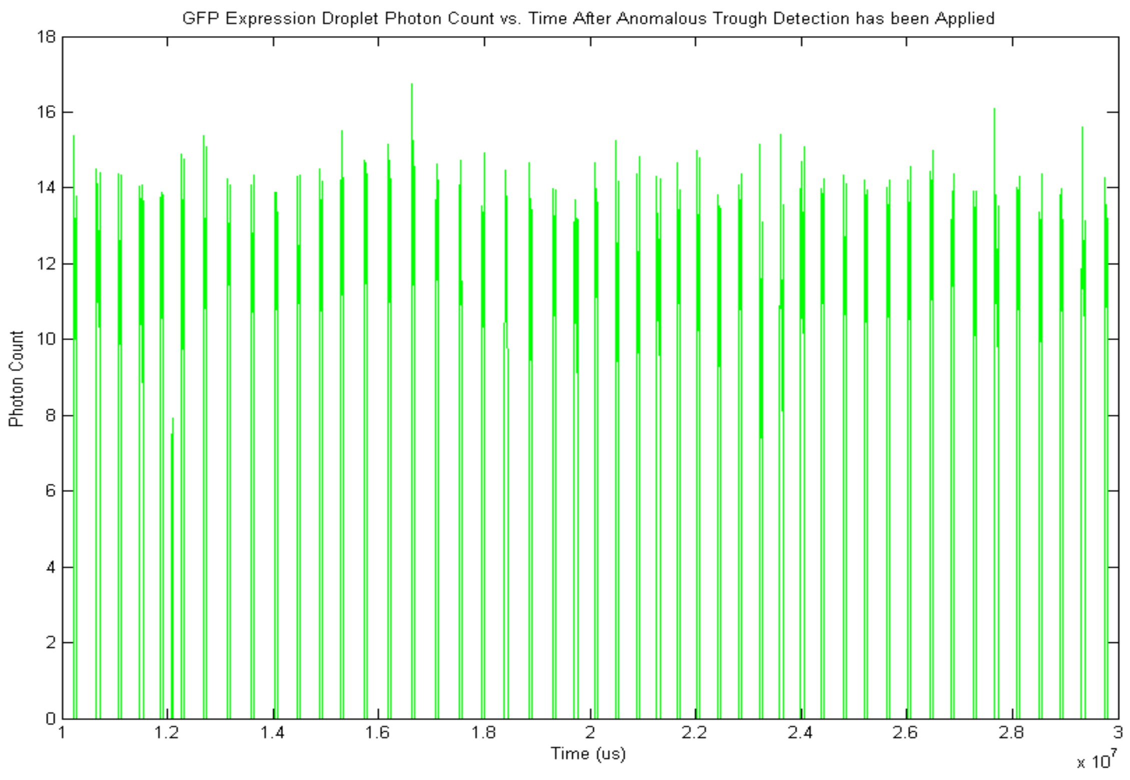


Figure 22: Example data set for the removal of erroneous troughs that may hinder droplet identification. The data used was from the previous step and corresponds to the actions performed in step 4.

```
%% Function to remove anomalous troughs from the data
% Checks for troughs within the data of a given tolerance (i.e. if a trough
% is wider than the tolerance it is considered real not anomalous). Data
% input type expected is the results of ThresholdDropletData
% Note: this should be used with a very stringent tolerance as the wider
% the trough, the less valid the replacement photon count becomes
function [outputData] = RemoveAnomalousTroughs(inputData, tolerance)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(1, 1))
        error('Unknown input data');
    end

    % Check the input
    if ~isnumeric(tolerance) && tolerance <= 0
        error('Invalid threshold value')
    end

    % Preallocate
    outputData = inputData;
    binaries = zeros(size(inputData(:, 1), 1), 1);
    firstTurningPoint = 1;

    % Identify zero elements of the input data
    indices = inputData(:, 1) == 0;
    binaries(indices) = 1;

    % Look for turning points
    turningPoints = diff(binaries);
    turningPointIndices = find(turningPoints ~= 0);
    lastTurningPoint = length(turningPointIndices);

    % Identify if there is data present at the dataset start (head)
    if turningPoints(turningPointIndices(1)) == -1
        % Ignore this as it does not count as a trough
        firstTurningPoint = 2;
    end

    % Identify reshape parameters and ignore any troughs at the end of the
    % data (tail)
    turningPointPairValue = lastTurningPoint - (firstTurningPoint - 1);
    if mod(turningPointPairValue, 2) ~= 0
        if turningPoints(turningPointIndices(lastTurningPoint)) == 1
            lastTurningPoint = lastTurningPoint - 1;
            turningPointPairValue = lastTurningPoint - (firstTurningPoint - 1);
        end
    end

    % Remove anomalous Troughs
    turningPointPairNumber = turningPointPairValue / 2;
    troughIndices = reshape(...
        turningPointIndices(firstTurningPoint : lastTurningPoint, 1), ...
        2, ...
        turningPointPairNumber)';
    for i = 1 : turningPointPairNumber
        if troughIndices(i, 2) - troughIndices(i, 1) <= tolerance
            outputData(troughIndices(i, 1) + 1 : troughIndices(i, 2), 1) = ...
                mean([inputData(troughIndices(i, 1)), inputData(troughIndices(i, 2) + 1)]);
        end
    end
end
```

Figure 23: Script for anomalous trough removal in APD data corresponding to step 5.

6.5.2.6 Remove Anomalous Peaks in Droplet Data from the APD

Step 6 is similar to step 5 as it improves the likelihood of droplet detection except that anomalous peaks are removed (Script: Figure 25, example data: Figure 24). This is likely to occur when small particles that are highly reflective pass across the point of observation. A threshold of 10 is generally used here (corresponding to 500µs of data), again this threshold choice is based on droplets being larger than this timeframe.

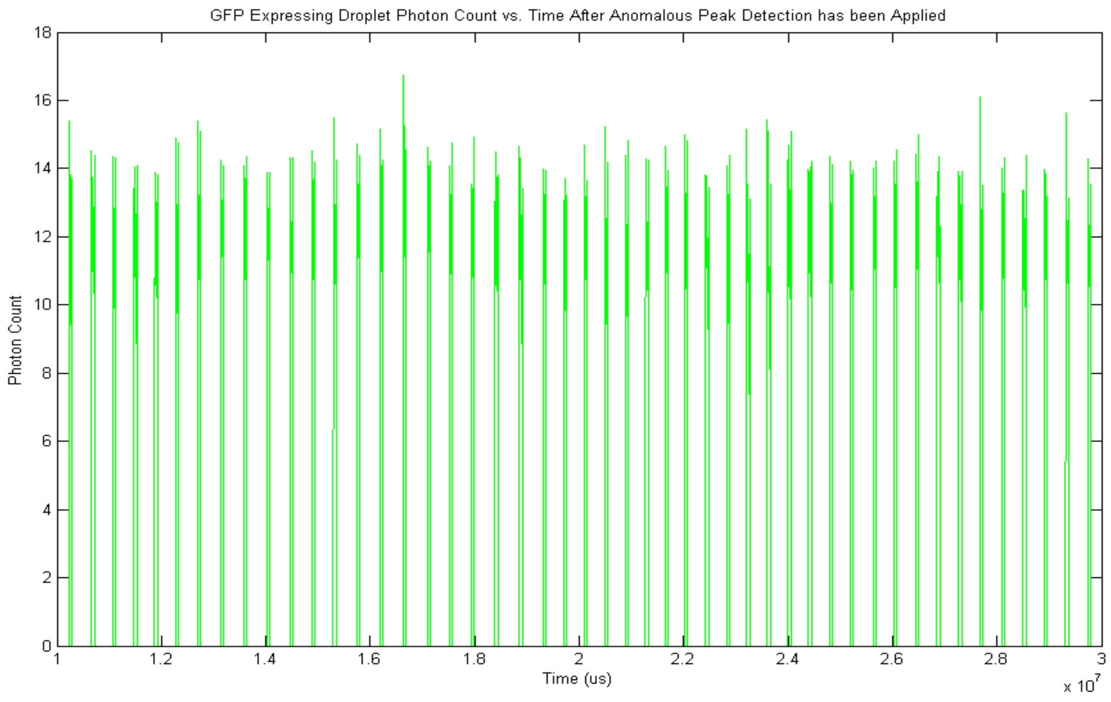


Figure 24: An example data set showing anomalous peak detection corresponding to step 6. The source data for this example plot was the output of the demonstration data from step 5

```
%% Function to remove anomalous peaks from the data
% Checks for peaks in the data less than or equal to a given tolerance
% Anything identified is set to 0. Data input type expected is the output
% from RemoveAnomalousTroughs
%
% This function helps remove peaks from dust or impurities in the oil phase
% of the microfluidic device. Tolerance choices should be based below the
% predicted droplet width. Confidence in this function can only be assured
% if the droplet formation step is consistent.
%
% Note: This function will ignore non-zero data at the tail and head of the
% data sets even if they are below the tolerance width. This behaviour is
% maintained throughout the later functions.
function [outputData] = RemoveAnomalousPeaks(inputData, tolerance)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(1, 1))
        error('Unknown input data');
    end

    % Check the input
    if ~isnumeric(tolerance) && tolerance <= 0
        error('Invalid threshold value')
    end

    % Preallocate
    outputData = inputData;
    binaries = zeros(size(inputData(:, 1), 1), 1);
    firstTurningPoint = 1;

    % Identify non-zero elements of the input data
    indices = inputData(:, 1) ~= 0;
    binaries(indices) = 1;

    % Look for turning points
    turningPoints = diff(binaries);
    turningPointIndices = find(turningPoints ~= 0);
    lastTurningPoint = length(turningPointIndices);

    % Identify if there is data present at the dataset start (head)
    if turningPoints(turningPointIndices(1)) == -1
        % Ignore this as it does not count as a trough
        firstTurningPoint = 2;
    end

    % Identify reshape parameters and ignore any peaks at the end of the
    % data (tail)
    turningPointPairValue = lastTurningPoint - (firstTurningPoint - 1);
    if mod(turningPointPairValue, 2) ~= 0
        if turningPoints(turningPointIndices(lastTurningPoint)) == 1
            lastTurningPoint = lastTurningPoint - 1;
            turningPointPairValue = lastTurningPoint - (firstTurningPoint - 1);
        end
    end

    % Remove anomalous peaks
    turningPointPairNumber = turningPointPairValue / 2;
    peakIndices = reshape(...
        turningPointIndices(firstTurningPoint : lastTurningPoint, 1), ...
        2, ...
        turningPointPairNumber)';
    for i = 1 : turningPointPairNumber
        if peakIndices(i, 2) - peakIndices(i, 1) <= tolerance
            outputData(peakIndices(i, 1) + 1 : peakIndices(i, 2), 1) = 0;
        end
    end
end
```

Figure 25: Script for anomalous peak removal. The script identifies extremely short regions of non-zero data by looking for turning points (where the direction of change in fluorescence equals 0). These turning points, thanks to our previous filtering, can be used to find sharp, erroneous peaks. This script is used in step 6.

6.5.2.7 Droplet Identification in Processed APD Droplet Data

The 7th step (correspond to Figure 26) in the droplet data processing pipeline identifies the leading and tail ends of droplets using the script found in Figure 27 and step 8 (Figure 28) uses these indicators to calculate pertinent droplet properties.

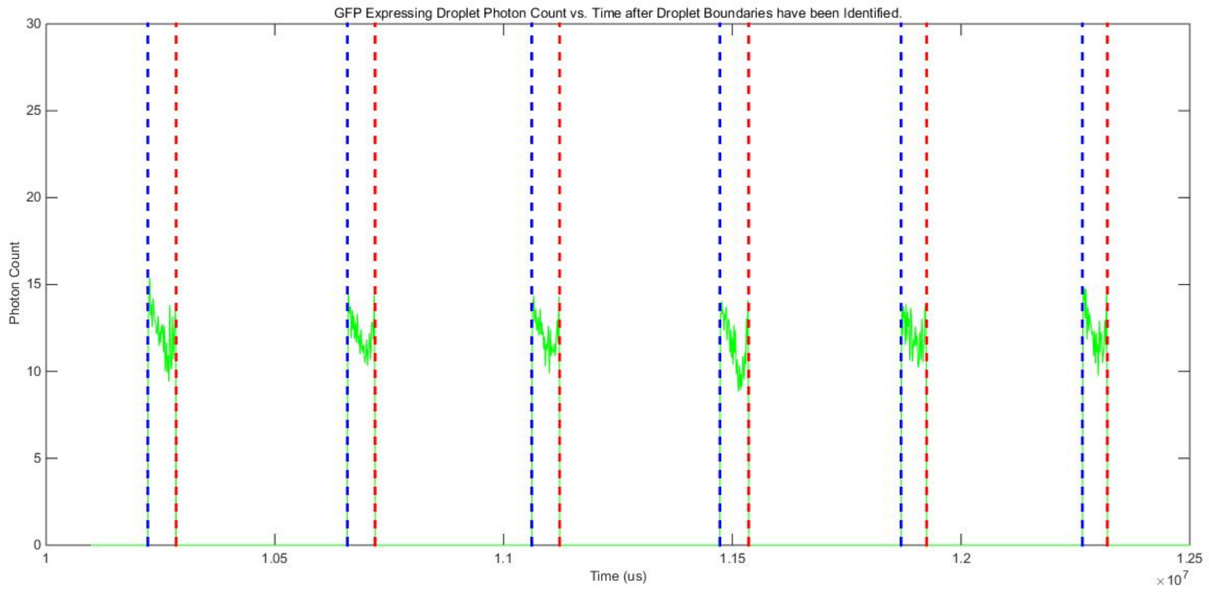


Figure 26: Example dataset showing the identification of droplet boundaries where the leading edge is marked in blue and the tailing edge in red. The source for this image was the output data from the previous step, please note that for clarity this has been magnified to only cover a few droplets, whilst the actual dataset used included many.

```
%% Function to identify droplets within the provided data
% Data input type expected is the output from RemoveAnomalousPeaks
% Data output is as follows:
%       OutputData:
%               column 1 == start index of droplets
%               column 2 == end index of droplets
%
%       InputData:
%               Same as inputData :)
%
% Note: This function will ignore non-zero data at the tail and head of the
% data sets even if they are below the tolerance width. This behaviour is
% maintained throughout the later functions. If droplets are present at
% these locations the user is advised to merge the datasets containing the
% rest of the data or manually add a zero data point at the head and tail
% end of the data
function [dropletIndices, inputData] = IdentifyDroplets(inputData)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(1, 1))
        error('Unknown input data');
    end

    % Preallocate
    binaries = zeros(size(inputData(:, 1), 1), 1);
    firstTurningPoint = 1;

    % Identify zero elements of the input data
    indices = inputData(:, 1) ~= 0;
    binaries(indices) = 1;

    % Look for turning points
    turningPoints = diff(binaries);
    turningPointIndices = find(turningPoints ~= 0);
    lastTurningPoint = length(turningPointIndices);

    % Identify if there is data present at the dataset start (head)
    if turningPoints(turningPointIndices(1)) == -1
        % Ignore this as it does not count as a trough
        firstTurningPoint = 2;
    end

    % Identify reshape parameters and ignore any peaks at the end of the
    % data (tail)
    turningPointPairValue = lastTurningPoint - (firstTurningPoint - 1);
    if mod(turningPointPairValue, 2) ~= 0
        if turningPoints(turningPointIndices(lastTurningPoint)) == 1
            lastTurningPoint = lastTurningPoint - 1;
            turningPointPairValue = lastTurningPoint - (firstTurningPoint - 1);
        end
    end

    % Save Peaks Locations
    turningPointPairNumber = turningPointPairValue / 2;
    dropletIndices = reshape(...
        turningPointIndices(firstTurningPoint : lastTurningPoint, 1), ...
        2, ...
        turningPointPairNumber)';
    dropletIndices(:, 2) = dropletIndices(:, 2) + 1;
end
```

Figure 27: Script to find the edges of droplets (droplet boundaries). This corresponds to step 7.

6.5.2.8 Extracting Droplet Properties from Droplets Identified in APD Data

The final step (step 8) serves to extract useful information from the regions of data labelled as 'within' a droplet. To do this the average droplet photon count, the average photon count's variance and an average time for the droplet's observation are calculated. Using this information, droplet identities can then be de-convoluted (i.e. identify which droplets contain what samples) and then processed in the same manner as section 8.4.3.The script described in Figure 29 performs this step and an example data, having been processed, is found in Figure 28.

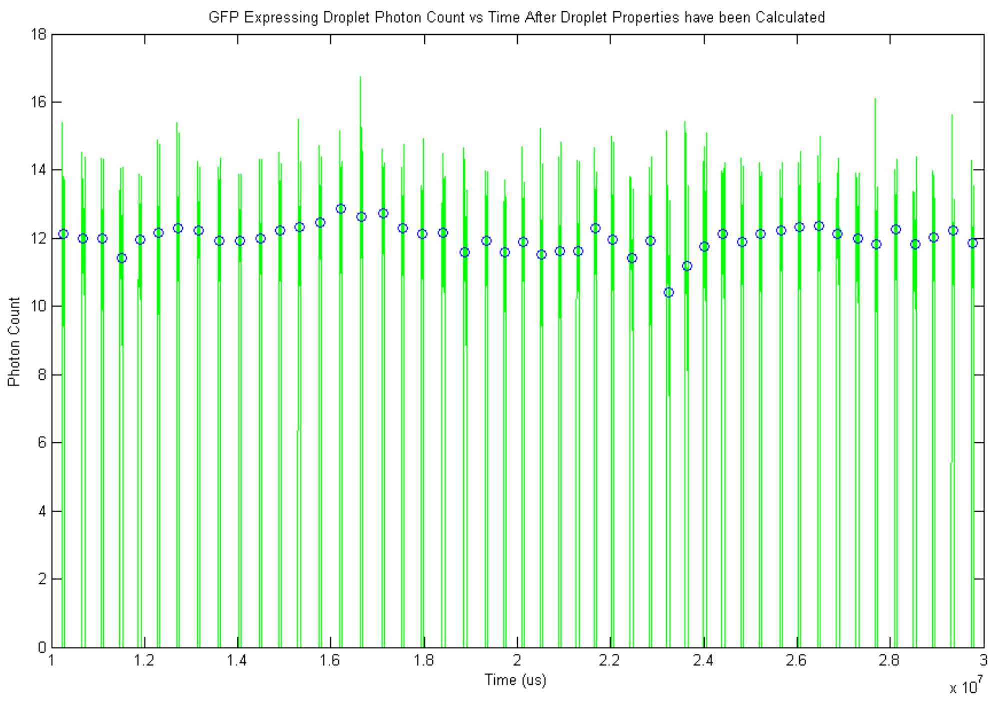


Figure 28: Script and example data showing the identification of droplets and their average photon count. Above is the script and below is the example data with average photon count indicated by blue circles.

```
%% Function to identify droplet properties.
% This function identifies the average photon count and time of observation
% for the droplet as well as the variance in the droplet photon count
%
% Data is output in the following manner:
%           column 1 == average photon count of the droplet
%           column 2 == variance in the photon count of the droplet
%           column 3 == average time for droplet observation
%           column 4 == width of the droplet in number of observations
%
% Note: Data input expected is the output from IdentifyDroplets
function [outputData] = GatherDropletData(inputData, dropletIndices)
    % Handle unknown data types
    if ~ismatrix(inputData) || size(inputData, 2) ~= 2 || ~isnumeric(inputData(1, 1))
        error('Unknown input data');
    end

    % Handle indices matrix
    if ~ismatrix(dropletIndices) || size(dropletIndices, 2) ~= 2 || ~isnumeric(dropletIndices(1,
1))
        error('Unknown droplet indices input data');
    end

    % Pre-allocate output data
    outputData = zeros(size(dropletIndices, 1), 4);

    % Build output data
    for i = 1 : size(dropletIndices, 1)
        dropletData = inputData(dropletIndices(i, 1) : dropletIndices(i, 2), 1);
        dropletTimes = inputData(dropletIndices(i, 1) : dropletIndices(i, 2), 2);

        outputData(i, 1) = mean(dropletData);
        outputData(i, 2) = var(dropletData);
        outputData(i, 3) = mean(dropletTimes);
        outputData(i, 4) = length(dropletTimes);
    end
end
```

Figure 29: The GatherDropletData script, corresponding to the final step in the APD Droplet Data processing pipeline.

6.5.3 Data Analysis for Microfluidic Droplets Detected Using a Fluorescence Microscope

The analysis of image data required some pre-processing (as described in section 8.4.2) in order to obtain images in an acceptable format (.tiff image format). The images are then processed using a MATLAB script. The final script for image analysis can be found in Figure 30. This script requires the Image Processing Toolbox^{TM} that can acquired separately to the core MATLAB framework. Figure 31 shows snapshots of the underlying image analysis pipeline, the data within the figure is a subset of the data used to produce the data shown in Figure 15 and Figure 16.

Firstly, images were loaded into MATLAB itself. All images for an entire time point were loaded at once which allowed computational identification of the optimum images to use for analysis. This principle is crucial in the automated analysis of images as the shutter speed and image capture steps of data acquisition are not aware of the contents of the image prior to capture. As such, the fluorescence microscopes low sampling frequency may capture many empty image frames as well as droplets that cross between images.

To overcome the limitations of the image capture, a derivation of a freely available script by Mark Brookes (http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html) was created to identify images that contain droplets or to identify images that contain the largest representative portion of droplets should the droplets cross multiple images. The script derives an average fluorescence per image and considers these as a fluorescence time series. The fluorescence time series has a threshold applied to it in order to reduce random perturbation effects in background light levels (normally with a value corresponding to an average pixel intensity of 16 which corresponded to the background light during acquisition). The peaks within this time series are identified and back correlated to the image source, an example of this peak finding is shown in Figure 31 (Top).

Once the images containing the target information were identified the images were transformed to black and white for better determination of the droplet boundaries within the image Figure 31 (Left). This function uses a threshold value for cutting off irrelevant light signals; to prevent loss of low fluorescence data a very low threshold was used at this step (0.0007). A low threshold value increases the computational work needed to analyse the data at a later stage and is likely inefficient at its current value. This threshold value was chosen because a wide range of fluorescence is likely across all of the constitutive promoters that were to be studied.

Regions of fluorescence within each image were identified Figure 31 (Right) and then using the area of these regions the droplets within each image were calculated. To achieve this, the largest area belonging to a region of fluorescence was indicated as a potential droplet. An arbitrary droplet minimum area was used to reduce false positives (corresponding to 10000 data points). This assumption removed all extraneous regions in test cases, however, such a cut-off also allows for small droplets being ignored. Droplets as small as those identified by this cut-off are assumed to either be erroneous (i.e. droplet debris within the channel) or not of interest because they would not accurately represent the entire droplet contents (i.e. the majority of the droplet has passed the objective at the time of observation).

Fluorescence intensity information was then derived from the original image data (i.e. prior to black and white transformation) using the image region identified in the previous steps. This intensity information was averaged across the area of the droplet and the standard deviation was calculated.

```
%% Function to load an image set corresponding to a set of droplets
% This function loads and processes a set of images in tiff format that
% corresponds to a single set of observations.

function [dropletProperties] = IdentifyDroplets()
    % Identify files that correspond to the period of observation
    dirPath = uigetdir;
    files = dir(dirPath);
    fileNumber = size(files, 1);

    % Preallocate
    viableImages = 0;
    imagesProperties = cell(1, 3);


    % Loop through provided files to load the data and generate the average
    % fluorescence per image
    for i = 1 : fileNumber
        % Check its a file we can understand
        filePath = strcat(dirPath, filesep, files(i).name);
        [~, name, ext] = fileparts(filePath);
        if ~strcmp(ext, '.tiff')
            continue;
        end


        % Load the image data
        imageData = imread(filePath);


        % Calculate the average intensity for the image
        imageAverageIntensity = mean2(imageData);

        % Store this image data
        viableImages = viableImages + 1;
        imagesProperties{viableImages, 1} = imageData;
        imagesProperties{viableImages, 2} = imageAverageIntensity;
        imagesProperties{viableImages, 3} = name;
    end

    % Threshold fluorescence
    thresholdData = [imagesProperties{:, 2}]';
    thresholdData(thresholdData <= 16) = 0;

    % Identify images of interest
    imageIndices = PeakFind([imagesProperties{:, 2}]);
    peakNumber = length(imageIndices);

    % Preallocate
    identifiedDropletNumber = 0;
    dropletProperties = cell(1, 4);

    % Loop through peak locations
    for i = 1 : peakNumber
       % Get this image’s information
        imageProperties = imagesProperties(imagesIndixes(i, 1));
        imageData = imageProperties{1, 1};

        % Transform the image into black and white
        bwImageData = im2bw(imageData, 0.007);

        % Generate the region properties based on the black and white image
        imageRegionProperties = regionprops(bwImageData, 'all');
        imageAreas = [imageRegionProperties.Area];

        % Identify the largest
        maxAreaIndex = find(imageAreas == max(imageAreas), 1);

        % Get region properties
        regionProperties = imageRegionProperties(maxAreaIndex, 1);
        regionPixelIndices = [regionProperties.PixelIdxList];

        % Check if the region is too small to be a viable droplet
        if (length(regionPixelIndices) < 10000)
            continue;
        end
        % Acquire the parameters of interest
        identifiedDropletNumber = identifiedDropletNumber + 1;
        regionData = imageData(regionPixelIndices);
        dropletProperties(identifiedDropletNumber, 1) = ...
            imagesProperties(imageIndices(i, 1), 3);
        dropletProperties(identifiedDropletNumber, 2) = {regionData};
        dropletProperties(identifiedDropletNumber, 3) = {mean(regionData)};
        dropletProperties(identifiedDropletNumber, 4) = {std2(regionData)};
    end
end

% Function to identify peaks in a 2d dataset
% This function was derived from the findp function by Mike Brooks 2005
% available online at:
% http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
function [imageLocations] = PeakFind(sourceData)
    imageNumber = length(sourceData);

    % Look for turning points
    turningPoints = diff(sourceData);
    upTurningPoints = find(turningPoints > 0, 1);
    downTurningPoints = find(turningPoints < 0, 1);

    % Check that there is a peak
    if ~isempty(upTurningPoints) && ~isempty(downTurningPoints)
        % Identify the previous rises
        risesAcceleration = upTurningPoints;
        risesAcceleration(2 : end, 1) = diff(upTurningPoints);
        pastRises = ones(imageNumber, 1);
        pastRises(turningPoints + 1, 1) = 1 - risesAcceleration;
        pastRises(1, 1) = 0;
        lastRises = cumsum(pastRises);

        % Identify the previous falls
        fallsAcceleration = downTurningPoints;
        fallsAcceleration(2 : end, 1) = diff(downTurningPoints);
        pastFalls = ones(imageNumber, 1);
        pastFalls(downTurningPoints + 1, 1) = 1 - fallsAcceleration;
        pastFalls(1, 1) = 0;
        lastFalls = cumsim(pastFalls);

        % Identify future rises
        futureRises = ones(imageNumber, 2);
        futureRises = futureRises - 2;
        futureRises([1; upTurningPoints + 1]) = ...
            [risesAcceleration - 1; imageNumber - upTurningPoints(end) - 1];
        nextRises = cumsum(futureRises);

        % Identify future falls
        futureFalls = ones(imageNumber, 2);
        futureFalls = futureFalls - 2;
        futureFalls([1; downTurningPoints + 1]) = ...
            [fallsAcceleration - 1; imageNumber - downTurningPoints(end) - 1];
        nextFalls = cumsum(futureFalls);

        % Identify peaks locations
        imageLocations = find(...
            (lastRises < nextRises) ... % Identify between rises
            & (nextFalls < lastFalls) ... % Identify between falls
            & (floor((nextFalls - lastRises) / 2) == 0)); % Is a peak

    else
        imageLocations = [];
    end
end
```

Figure 30: Script for processing images produced by the fluorescence microscope. See the annotations in green for a more detailed breakdown of how the script works in each section.

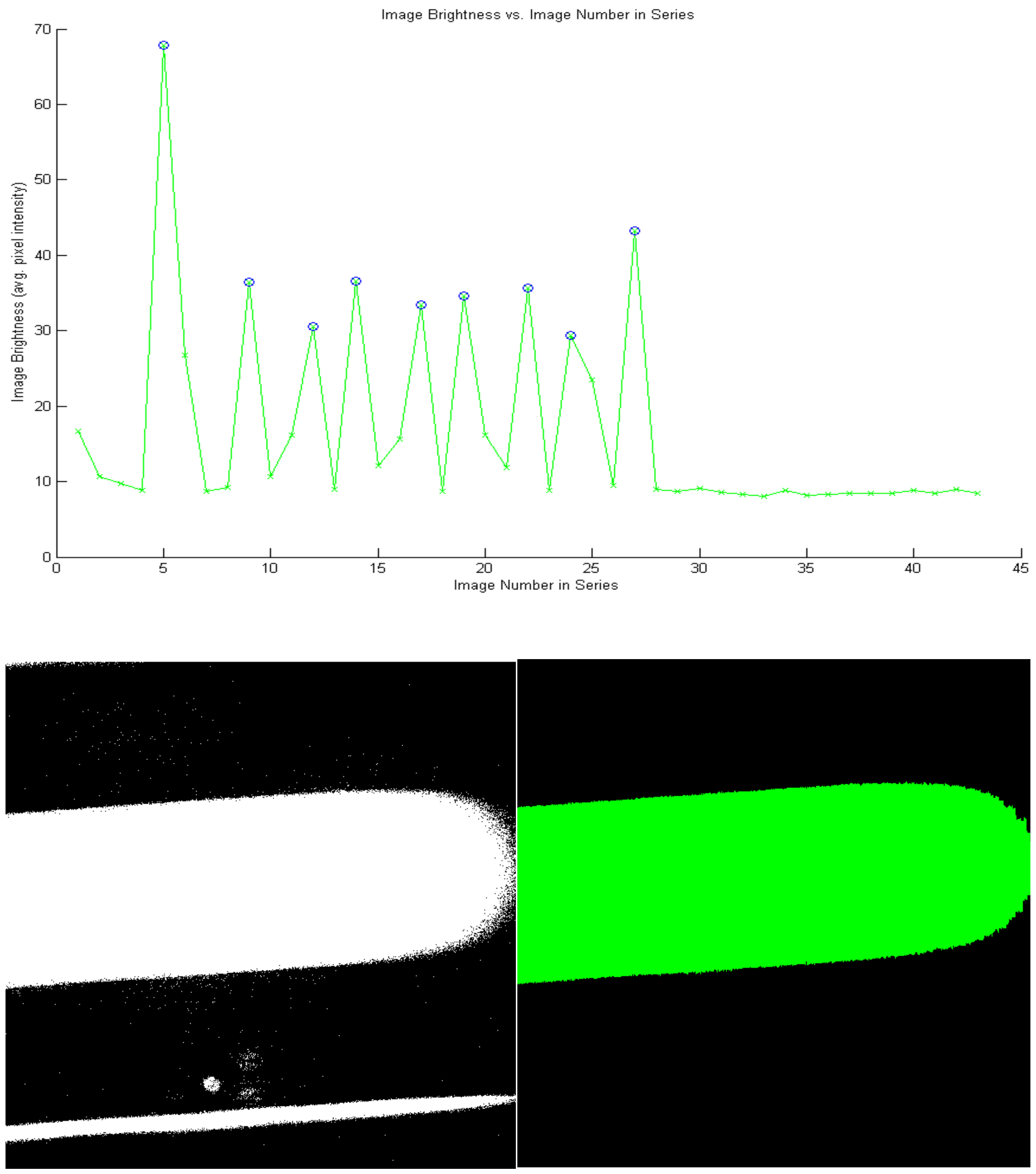


Figure 31: Depiction of the image analysis process. Top: Shows the average fluorescence per image across a short capture window. The peaks have been successfully identified (blue). Left: shows the image corresponding to the first peak after being transformed to black and white. Right: shows the identified pixels in the image from the left that correspond to the perceived droplet. Each of these images were generating using a section of the script above.

6.6 Outcome Summary

Over the course of the platform iterations a great deal of progress has been made in establishing a high throughput *in-vitro* characterisation platform for Synthetic Biology by identifying the fundamental challenges associated with a microfluidic orientated design. Until platform iteration 5 it had been difficult to identify a platform design that was capable of characterising a single expression vector, let alone one capable of achieving the high throughput aims of this investigation. However, each of the iterations served to identify flaws in the underlying methodology and slowly increase the stability of the platform.

The data analysis pipelines for droplet microfluidics, as described above, were successfully used to extract meaningful data sets from both APD data and fluorescence microscope images. They have shown demonstrable success in processing the source data into meaningful information for use throughout the iterations produced above. Given the availability of MATLAB, the signal processing toolbox, and the image analysis toolbox, these scripts should be transferable to any modern (v.11+) MATLAB environment. Given the amount of post processing necessary to use the data in a modern assay, it was considered unnecessary to package the scripts into a single executable.

The success in the iterative process in this investigation can only be attributed to the fundamental design principle of modularity that was employed from the outset. Including this design principle from the moment of the platform's inception allowed the identified 'modules' (Figure 4) to be hot-swapped without re-designing the entire platform, and thereby increased the platform prototypes' turnover rate. This modularity also greatly improved the ability to identify the points of failure within a design and use the existing research to select alternative solutions.

Interestingly, the iterations proposed in this study are not the only solutions to each problem encountered. Adam Abate's lab has had great success in improving droplet formation stability by employing a valve based tuneable flow focusing geometry (Romero and Abate, 2012). This kind of geometry may have had better success than those trialled in iteration 4 and may have led to a different solution being investigated for Iteration 5.

Another key take point to consider is that no fluid simulations of the microfluidic devices were performed in this study. Software exists that can help predict the behaviours of the devices within the channels. One of the more common examples of this kind of software is known as COMSOL Multiphysics (COMSOL Inc.). COMSOL is a Multiphysics simulator that includes a microfluidics module for simulating microfluidics devices. Primarily the software was not used due to the mixed type of devices being employed throughout the investigation, as well as the most common failure point of all the devices being the chip build quality, but it would certainly have helped in identifying some of the back pressure issues observed in Iteration 1.

Whilst basic fluorescence quantification for an expression vector was achieved across multiple droplets (Figure 15), high throughput *in*-vitro characterisation (within the constraints of our initial platform aims) was not. As also noted, through a manual curation of the source data, it was possible to profile multiple expression vectors in the same reaction run (Figure 16), albeit with no droplet repeats to provide any statistical confidence in the levels. To achieve the level of characterisation outlined in section 6.1.3 it is necessary to have a minimum of two expression vectors' fluorescence being quantified concurrently with more than one droplet being recorded per vector type. The characterisation of fluorescence production from the target expression vector and the standard expression vector (BBa_J23100) is necessary to normalise environmental variations in the assay. Due to the inability to discern droplet identity consistently in the latest iteration of the platform, concurrent quantification of droplet fluorescence cannot be achieved.

Many of the issues identified throughout the course of the platform iterations can be attributed to the immaturity of the microfluidics field with regards to automated and repeatable hands-off technologies at picolitre volumes (deMello, 2006). Despite microfluidics' ability to allow an inordinate level of control over samples and produce excellent data quality using a variety of detection techniques, microfluidics solutions still require a great deal of maintenance and adjustment from a user's stand point.

This statement is best exemplified by the backpressure issues that have arisen throughout the platform iterations. Hands-off, automated, solutions to the platform design, using the currently available technologies, require tubing lengths and inner diameters that are not conducive to either long term use or consistent droplet formation. In essence, this negates the initial platform aims as the platform cannot currently represent a standalone technology for characterisation in synthetic biology.

Another small but no less important issue with the platform was identified throughout the iterative process: the control of reaction initiation and its impact on multiplexing within the platform. As discovered in platform iteration 2 (Figure 6) the initial technique used for reaction initiation failed to result in detectable levels of fluorescence production. To solve this, the technique of in-syringe or in-tube (off-chip) reaction initiation was adopted. This technique eventually yielded detectable fluorescence levels but removed much of the multiplexing capability desirable in the platform. Off-chip reaction initiation also complicates data analysis, as ascribing reaction duration values to droplets detected at points of observation is no longer consistent over a droplet set. This concept is important because when droplets are imaged they will represent different reaction durations. This is different to on-chip reaction initiation because reaction initiation would occur with a delay corresponding to the droplet's position in the chip channels allowing the reaction duration of each droplet in a droplet set to be identical (i.e. all droplets would therefore be at the same stage of reaction progression despite being sampled at different times).

Despite these two core issues, much of the fundamental work associated with creating a functional platform has been achieved, and solutions to these two issues have been identified. The availability of solutions to these two core problems indicates that a realised version of the characterisation platform is within reach. To overcome the issue of backpressure new versions of the droplet robot (Gielen et al., 2013) are capable of forming droplets within the footprint of a microscope's stage which would greatly reduce the length of tubing required. The newer versions of the droplet robot also facilitate better control over droplet formation and morphology with increased flexibility in the inner diameter of the tubing being used. Both of these imply that greater control can be had over droplets which could theoretically reduce droplet merging and therefore expedite the process of accurate droplet identification by maintaining the FIFO droplet ordering. Multiplexing of reactions within the platform is also possible using the microfluidic geometries described in iteration 4 Figure 12). By alternating droplets of IVTT and expression vector and forming these droplets with different volumes, controlled droplet coalescence could re-allow on-chip reaction initiation. Such volumetric control over droplet formation can be achieved with the droplet robot by adjusting the residence time of the sample selection head within the reaction phase (i.e. larger droplets can be formed by allowing more IVTT to be withdrawn to form a droplet compared to the expression vector). By once again separating the IVTT and expression vector components of the characterisation reactions, multiple samples could once again be characterised in a single assay using only a single sample of IVTT. This reinforces one of the key advantages of microfluidics as a tool for manipulating biological reactions; the principle of reduced reagent consumption. Should the amount of IVTT required to assay 15 concurrent expression vectors be reduced to a single pot, the platform's attractiveness as an in-lab technology would increase greatly as the cost of commercial IVTT can be quite prohibitive to routine characterisation.

Fundamentally, *in-vitro* characterisation has been shown to compare well to *in-vivo* characterisation under basic conditions (Chappell et al., 2013) and its utility as a rapid prototyping technique for expression vectors, before detailed experimental work is performed, is extremely appealing, but the issues when characterising complex biological circuits (Chappell et al., 2013) cannot be overlooked. Using microfluidics as a characterisation tool in synthetic biology actually presents options for dealing with characterisation of complex devices. As shown in Villar *et al.*'s paper (Villar et al., 2011), droplets can be allowed to communicate through small molecules that can freely diffuse into the oil phase within the chips. Taking advantage of this, smaller less-complex components of the overall device could be encapsulated in a serial manner and designed to communicate through the oil phase, therefore mimicking the device complexity that was previously an issue for characterisation. This would allow device components to be characterised individually as well as the characterisation of overall device behaviour by monitoring reporters in the final droplet.

7 Data - Processing, Analysis and Display Software Results and Discussion

7.1 Overview

To overcome many of the issues described in section 5.5, a data handling framework called Data – Processing, Analysis and Display (DPAD) was developed that is specifically designed to handle the large variety of data sources and characterisation assay types present within biological part characterisation. Specifically, the DPAD software is designed to provide a highly flexible, network ready data storage and analysis framework. In brief, the software was designed to run in an organisational environment, be it large or small, and allow individual research to save data from their workstations to a centralised location. This centralised location then stores, processes and analyses the data as specified by the researcher.

To achieve a viable framework for handling characterisation data, the conventional data handling procedure must be broken down into a series of discrete operations, as shown in Figure 32, which represents the core work flow for the software. By decoupling the data handling procedure into these operations, data can be handled and stored in a step-wise manner to ensure that the outcome of each data manipulation operation can be saved for validation later. The steps shown in Figure 32 directly inform the software's operation and affect the implementation of both the plugin and the database topologies (described in section 7.2.3 and section 7.2.2 respectively). The first operation in the data handling procedure is the loading the data into a format that is understood by the software. This is followed by the processing of data to generate a profile for a single sample, with a single observation type over time. The next step allows the amalgamation of sample data to produce meaningful characterisation data. Finally, the data is displayed according to the characterisation type being performed. These operations directly correlate to the types of plugins available within the DPAD software framework where plugins for the loading/processing, analysis, and display can be created independently of one another.

To maintain flexibility in the types of data processing and analysis that can be performed, specifically in our case characterisation, and increase the overall utility of the platform, a variety of data input types must be handled. It is for this reason that each of the operations has been deferred to plugins so that the software framework can be expanded dynamically by the researchers according to their needs. The need for this flexibility becomes apparent when looking at the data sources used across the evolution of characterisation in Synthetic Biology. Early attempts used microplate readers as the single source of data (Kelly et al., 2009) whilst current implementations pair this information with data from flow cytometers. This is also apparent in the characterisation platform produced above (see section 6), which utilised either APD data or image data to characterise the microfluidic droplets.

The overall utility and efficiency of the software is increased by its ability to handle data batches, i.e. the amount of data that can be handled by plugins is only limited by the implementation of the plugins themselves. The final solution to the DPAD software breaks down the implementation into three core building block architectures that abstract many of the important functional components of the software away from the core runtime environment. Namely: the flexibility of the platform is predicated on a plugin architecture that allows the framework to be dynamically expanded, as the needs of researchers and their equipment types grow over time. The software is also based around a core database architecture that allows for versatile sample data storage as well as rapid and flexible sample context annotation. Finally, the network based architecture of the platform allows for centralisation of the data storage and submission of data from any computer with the software installed. These three architectures are described below and constitute a summary of the salient implementation results of the DPAD software.

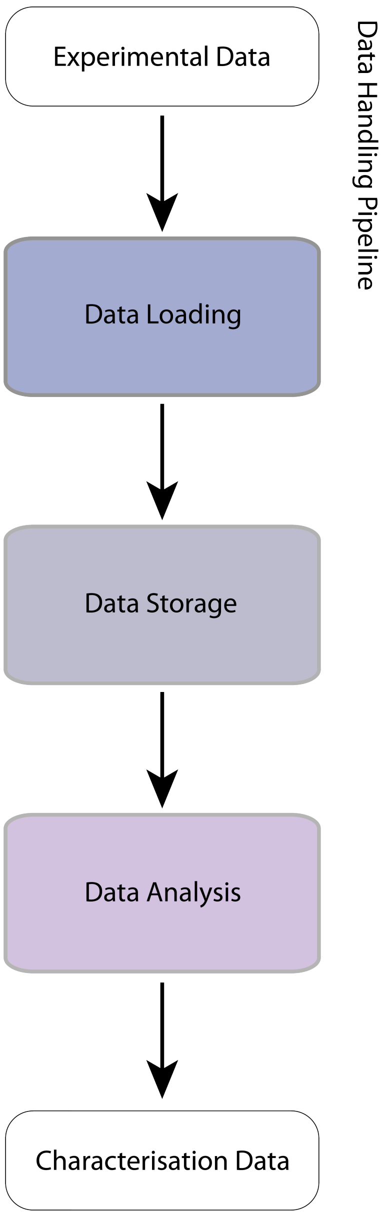


Figure 32: Data handling pipeline breakdown. The steps responsible for producing accurately characterised data that are distinctly represented within the DPAD software

7.2 DPAD Software Implementation

7.2.1 Network Architecture

The DPAD software utilises a network architecture that allows the software to operate in three distinct states. These terms are used to distinguish the different ways in which the software can be used. As shown in Figure 33 the three modes of operation are client, server and local. The client mode of operation is responsible for user based interactions with the software and the runtime environment for the plugins (see section 7.2.3). The server mode of operation controls and limits access to the software's database as well as monitoring and handling the logic required for multiple client connections. The local mode of operation is responsible for mimicking a server connection by creating a server implementation that it is hidden to the user but provides the same user experience. Local modes of operation are important as they allow the software to be used when no network connections are available or for when users do not wish to submit their data to a centralised repository. By convention, the underlying network architecture implementation type used is known as 'thick client' whereby all data is processed and manipulated on the client's computer and all data is stored and accessed through the server. Choosing to implement a network architecture promotes the centralisation of data storage, which in the long term can be used to reduce the amount of *ad hoc* research being performed. Promoting automatic data submission to a centralised repository also promotes the lateral transfer of knowledge to other software users if and when a permissions system is implemented (see section 7.3).

The network implementation relies upon a multi-threaded, asynchronous, event-driven network application that is freely available to open-source and commercial applications (http://netty.io/). This application provides a high-performance, highly scalable scaffold for custom network implementations, which sets the stage for a highly versatile network implementation for multiple concurrent users within a dedicated server environment. Utilising an asynchronous, event-driven model for the network application allows multi-threaded software implementations to achieve high-performance as no queuing of network information processing and responses occurs. To achieve a secure and rapid network implementation the communication of information between the client and server was broken down into a series of message handlers along the communication pipeline. Information being transmitted is limited to implementations of the Packet class (see Figure 35). A diagrammatic representation of these handlers can be found in Figure 34 as well as a short description of the handler's function. Of interest within the pipeline are the optional encryption handlers which, when the software is not operating in local mode, secure all information transmission to and from the server using RSA encryption (Katzenbeisser, 2001).

These layers of communication are built upon the transmission control protocol (TCP) directly rather than leveraging some of the existing communication standards such as Simple Object Access Protocol (SOAP (Mein et al., 2002) and Representational State Transfer (ReST (Jakl, 2005)). This decision was made for three core reasons. Firstly, direct implementation of a custom communication layer between the client and server allows for state aware communication. State aware communication is essential for the general operation and encryption layers of the DPAD software which utilise context specific network handlers depending on the client state (e.g. encrypted or unencrypted). Secondly, by implementing a custom communication layer, the software provides highly performant, compact and efficient network messaging as the implementation leverages direct access to byte buffer frameworks and direct 'packet in' to 'packet out' connections, which reduces the need for additional structure specific information (such as hypertext transfer protocol, HTTP, headers) to be transferred as the message can only be interpreted in one manner (by the mapped packet, with the correct network context). Finally, DPAD relies upon constant uptime in network connections in order to reduce the potential for sensitive data to be intercepted. This is achieved by maintaining a 'heartbeat' communication between the client and server that prevents identity based attacks from pretending to be clients after successful authentication. These choices do have a downside, which is that generally HTTP messages can be passed between the client and server without firewall issues, but a custom TCP implantation generally requires specific ports to be opened up within the organisation deploying DPAD. However, the reasons given above sufficiently outweigh the overheads incurred by using a custom TCP implementation.

Network communication within the software is monitored by a dedicated network manager thread which, when a client first connects to the server, generates a distinct session that handles all communication between the server's database and the client in a secure manner (see section 7.2.2). The network managers sort network communication and hand them off to the correct client sessions as well as allowing sessions to communicate with the correct client. Network sessions are responsible for handling the data communicated through the pipeline and for regenerating packets into meaningful information. Network sessions hand off a context specific network protocol to the packets to allow them to be handled accordingly. Network protocols are based around the current state that a connection between the client and the server is in (e.g. whether the user has successfully logged in and obtained an encryption key yet). By handling the newly obtained (through network transfer) packets with a network protocol rather than the network session, packets are limited in the data that they can access, adding a further layer of security to the network communication.

Network communication is not freely available to plugin implementations, see section 7.2.3**,** as this would defeat many of the security layers in place, instead a series of common packet types are made available to the plugins through the plugin API, see section 7.2.3**,** which should cover all the necessary network communications. Abstracting the network communication in this way maintains flexibility and allows rapid error tracing whilst minimising the security risks for centralised data storage and decreasing the complexity of implementing custom data handling plugins.

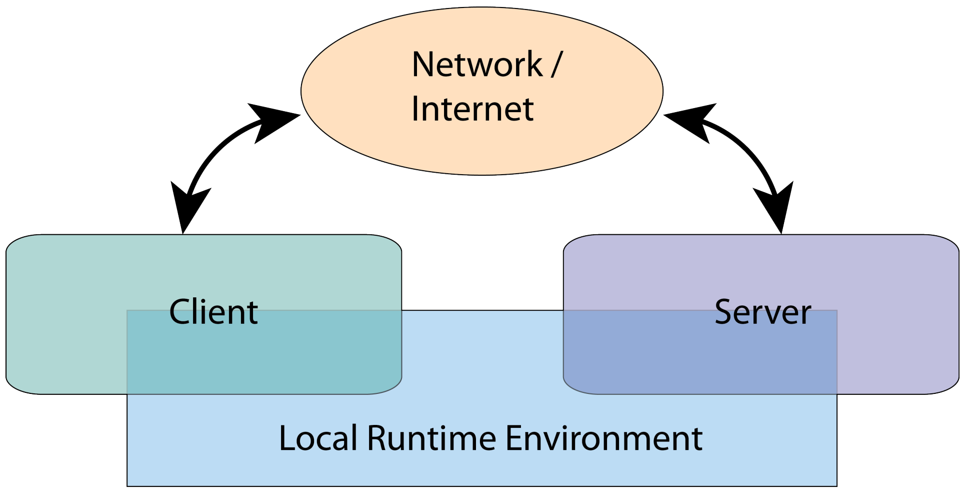


Figure 33: Runtime architecture for the DPAD software. The software can run in client, server or local modes where local runtime creates a hidden server for the user to interact with.

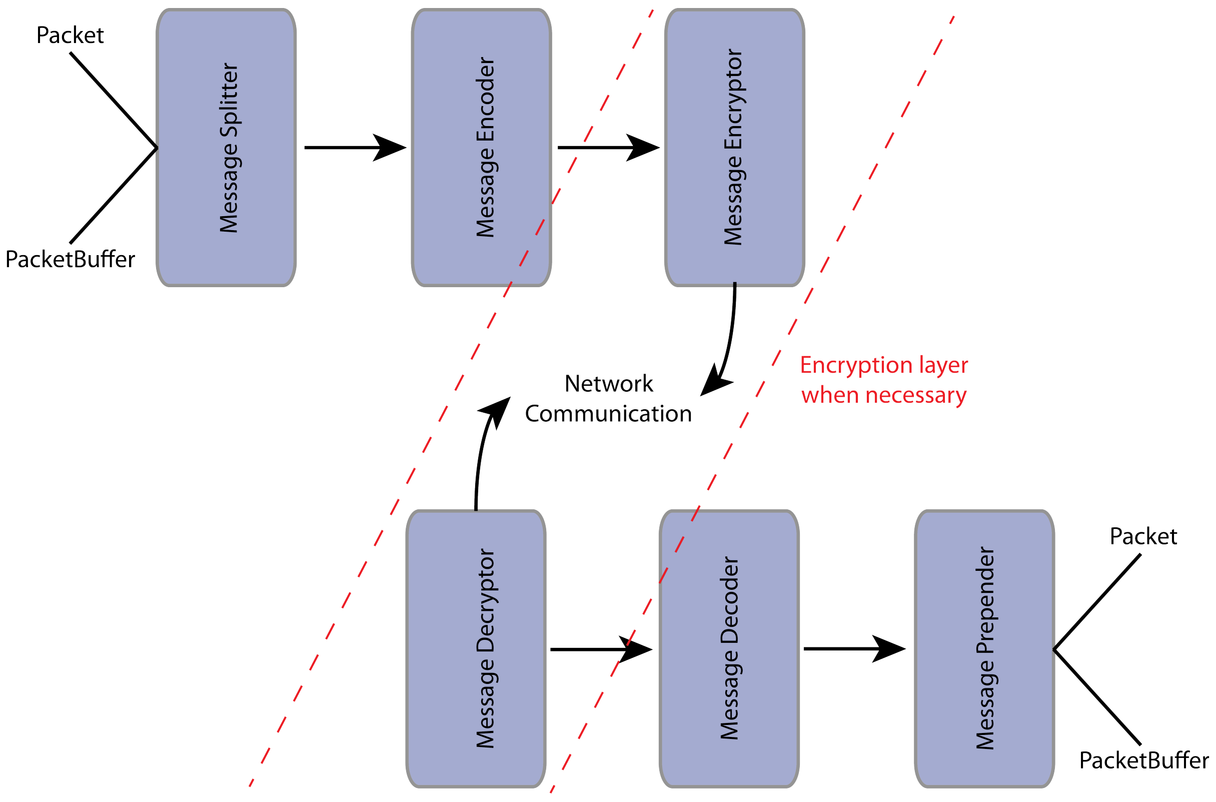


Figure 34: Schematic of the network pipeline. The pipeline operates in both directions from the client to the server. The message splitter first encodes a packet identifier into the byte stream, after which the packet contents are encoded into the byte stream. If the network channel requires security the byte stream is encrypted. On the other side of the channel (i.e. after the network communication) the process is reversed.

```java
package jonathansmith.dpad.common.network.packet;

import java.io.IOException;
import com.google.common.collect.BiMap;
import jonathansmith.dpad.common.network.protocol.INetworkProtocol;

/**
 * Created by Jon on 26/03/14.
 * Abstract packet class. Parent for all networking packets.
 */
public abstract class Packet {

    /**
     * Empty constructor to allow for packet registration without data
     */
    public Packet() { }

    /**
     * Generate an empty packet for automatic regeneration of the packet from the byte stream
     * that is transmitted
     * through the packet pipeline
     * @param integerClassBiMap the list of packets that are pre-registered
     * @param packetId the id of the packet within the packet list
     * @return the empty packet
     * @throws Exception if the packet could not be initialised
     */
    public static Packet getEmptyPacket(BiMap<Integer, Class<? extends Packet>>
                                        integerClassBiMap, int packetId) throws Exception {
        Class clazz = integerClassBiMap.get(packetId);
        return clazz == null ? null : (Packet) clazz.newInstance();
    }

    /**
     * Used to determine whether the packet needs to be processed immediately or should enter
     * the packet queue.
     * @return true if the packet should be processed out of order. Conventionally used for
     * setup purposes only.
     * Return false if the packet should be processed normally.
     */
    public boolean isUrgent() {
        return false;
    }

    /**
     * Function called to generate the packet data from the byte stream
     * @param packetBuffer the byte stream containing the packet's data
     * @throws IOException
     */
    public abstract void readPacketData(PacketBuffer packetBuffer) throws IOException;

    /**
     * Function to write the packet data into the bytestream using convenience methods in the
     * {@link jonathansmith.dpad.common.network.packet.PacketBuffer}
     * class
     * @param packetBuffer the packet buffer to write the data to
     * @throws IOException
     */
    public abstract void writePacketData(PacketBuffer packetBuffer) throws IOException;

    /**
     * Function called to handle the packet after it has been reconstituted from the byte
     * stream
     * @param networkProtocol a situational network protocol dependent on the state of the
     * network connection.
     */
    public abstract void processPacket(INetworkProtocol networkProtocol);

    /**
     * Convenience method for debugging that converts the packet into a string
     * @return summary of the packet
     */
    public abstract String payloadToString();
}
```

Figure 35: The abstract packet class that is necessary for all network communication

7.2.2 Database Architecture

The DPAD software's database architecture is designed to be fast, capable of encryption and have a small memory impact. The database runs on the server side of the software and cannot directly be interacted with by the user. To access the database contents a client must query through the server through the network architecture. Database access within the server side of the software is limited only to a network session, which is only obtained once a client has connected. This implementation prevents any external access to the database and maintains a one to one relationship between database connections and client connections.

The database core implementation used within the software is known as H2 (http://www.h2database.com/) and is freely available. H2 utilises the structured query language (SQL) for communicating with the server, requires an extremely small amount of memory and relies purely on the Java programming language which provides maximum compatibility with the current implementation of the software. Despite H2's well documented performance (http://www.h2database.com/html/performance.html) some issues have been reported with large numbers of concurrent database queries causing locks in the database input or output. To allow maximum flexibility in the software, the type of core database implementation can be changed with relative ease in future versions of the software, as long as the database still uses SQL for communication. This would allow the software to utilise more robust, industry tested SQL databases should the need ever arise.

The Hibernate ORM (http://hibernate.org/) application is embedded within the software to manage communications between the server and the database. Hibernate ORM offers a highly scalable, reliable and high performance to database creation and access. Hibernate ORM allows mapping java application objects (including inheritance and relationship properties) directly into the database without having to engineer an SQL schema directly. To achieve this, the Hibernate ORM communicates with the H2 database using the Java Database Connectivity API (JDBC) which allows direct communication with SQL databases. The application offers lazy loading, optimistic locking and specialised data fetching strategies, all of which improve the input/output performance of data into the database. The database schema is set programmatically within the DPAD software and either automatically generated or validated when the server starts up.

To improve the maximum concurrent users capable of querying the server at any one time the C3P0 connection pooling application (http://www.mchange.com/projects/c3p0/) has also been embedded into the software. This application is responsible for managing the number of connections that are open between the database at any one time and recycling both memory and connections when they become closed or unused.

The database schema for the DPAD software is shown pictographically in Figure 36 and explicitly in Figure 37. The core data storage record is the dataset record which essentially contains time course data for a single measurement type and the corresponding times. Each dataset record must be appended with a sample tag record and a measurement type tag record, but can be tagged with any number of additional context tags records. A dataset record must also contain a plugin record reference so that the source of the data can be traced. Finally, each dataset record must be owned by an experiment record which in turn must be owned by a user record. This allows datasets to be loosely grouped without forcing a direct empirical relationship with each other. Allowing datasets to 'own' any number of tags increases the flexibility of the software to make use of metadata. The context tag is simply a name and a description with no limit on the contents, therefore these tags could be used to include anything from the bacterial strain used or the ambient temperature during data acquisition.

Finally, the software enforces database schema contracts (meaning the schema cannot be changed) by only allowing table specific (table relates to a specific portion of the database schema, i.e. the user database record schema) objects to access their contents. These objects are stateless (i.e. they are not dependent on the client who is using them) and are freely available if the client has acquired a database connection. The table specific objects have the necessary SQL queries embedded within them to enforce the types of manipulation that can be performed on database records.

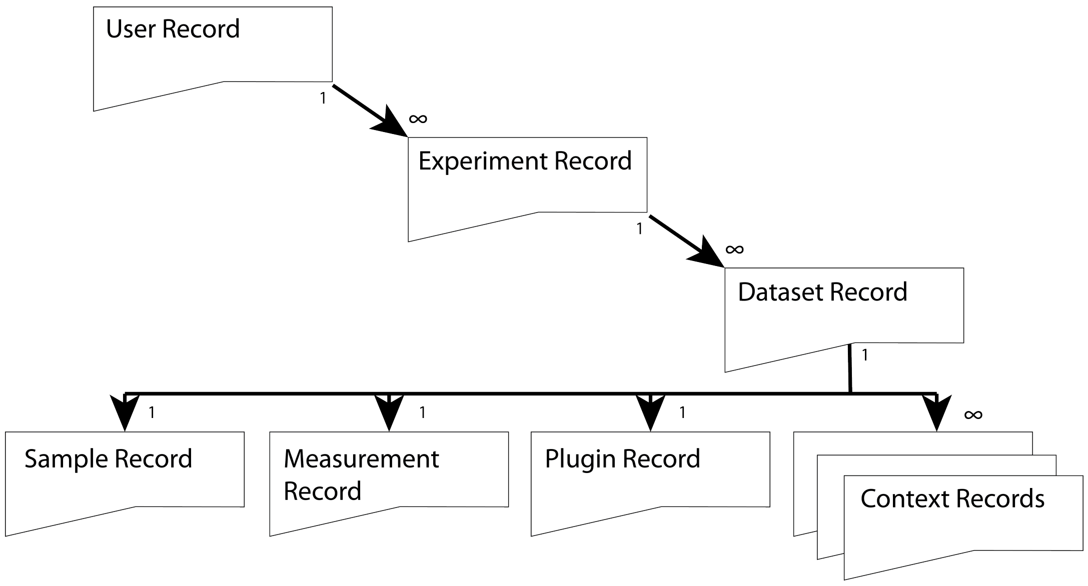


Figure 36: Database schema for the DPAD software. Relationships are either one to one or one to infinite as indicated by the annotations on the arrows.

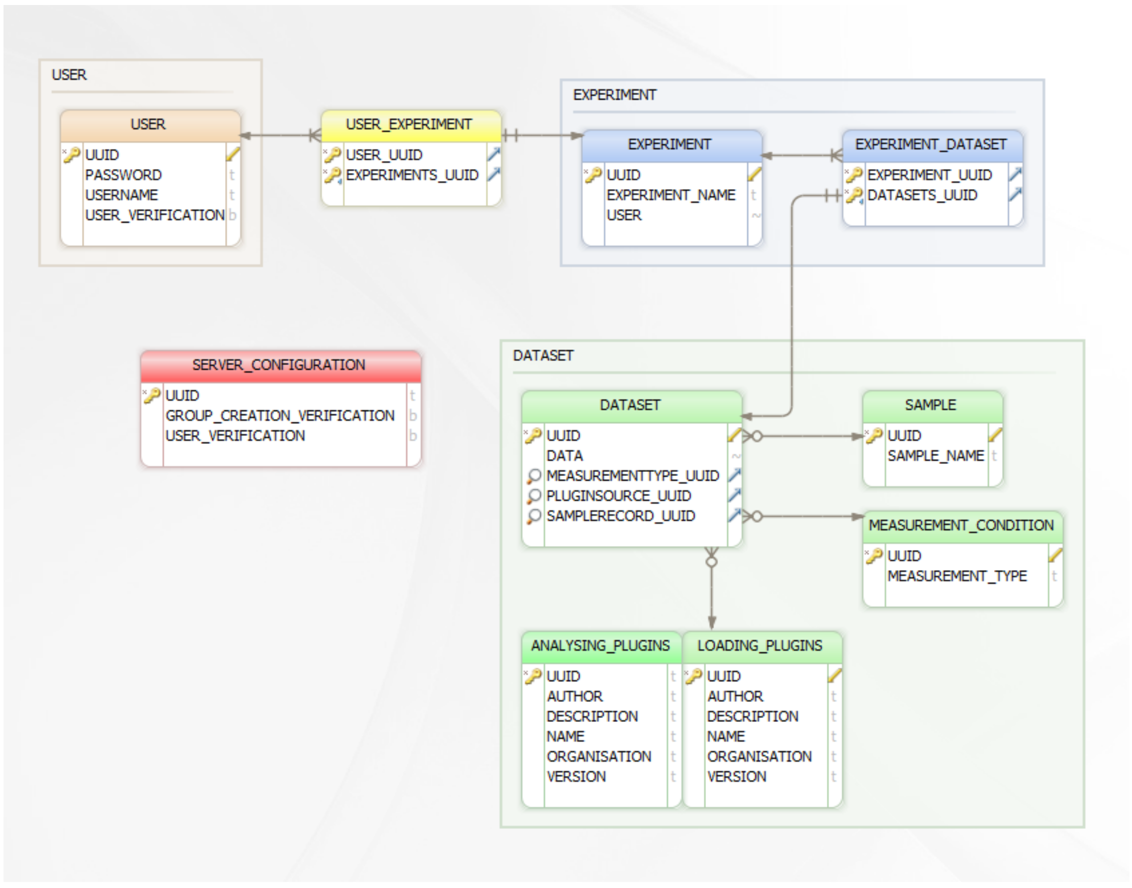


Figure 37: Full Schema for the DPAD H2 Database. Tables are arranged into the core functional sections.

7.2.3 Plugin Architecture

The DPAD software contains a plugin architecture that enables flexibility in both the types of data that can be loaded into the software as well as the types of analysis that can be performed. The plugin architecture allows for plugins to be developed using the DPAD software API without knowledge of the actual software's internal functionality. Plugins for the architecture are of two varieties: data loading plugins and data analysis plugins.

Data loading plugins are responsible for loading data from the client computer and submitting them to the server side database. This process can involve de-convoluting complex datasets such as mixed measurement types or batch type data. The implementation is entirely dependent on the plugin and is not limited in scope. Data analysis plugins are capable of querying server side data that a user has access to (see section 7.2.2 and section 7.3). Data analysis plugins are responsible for combining multiple datasets in meaningful characterisation data.

To achieve this, the Java Simple Plugin Framework (JSPF) (https://code.google.com/p/jspf/) application has been embedded within the software implementation. This application allows dynamic loading of separate applications given that they implement a specific interface from the DPAD API (this interface is dependent on whether they are a loading or an analysing type plugin). The JSPF application completely masks any of the actual plugin code from the server, instead relying only on the aforementioned interfaces to communicate. This is advantageous as it minimises interactions to an extremely controllable level that ensures security on the server side.

The DPAD API is the main crossover point of information between plugins and the software itself. Primarily communication is provided through two key interfaces (namely ILoadingPlugin and IAnalysingPlugin) both of which inherit the IPlugin interface, see Figure 38. Predominantly information is gathered about the plugin by obtaining a plugin record from the plugin which is submitted to the server alongside a dataset in order to trace its origins. Actual runtime operation of a plugin is obtained by retrieving an ordered list of tasks (objects that inherit the IPluginTask, see the interface in Figure

36) which contains functions that allow injection of the software's API. The API allows for display changing, event thread access and controlled communication to the server. The list of tasks obtained from the plugin represents an ordered set of operations that should be performed by the main application runtime.

Two core plugins are already available for the current implementation of the DPAD software. These plugins are designed to be examples for future plugin implementations and exhibit all of the necessary functions that a plugin would conventionally use. The first plugin is designed to load microplate data in a tab separated value format. An example of the data can be found in section 10.4. This example data can be de-convoluted into a time course data set for 96 samples with two measurement types. The second plugin can query the server for datasets in order to generate characterisation data according to the Kelly method (Kelly et al., 2009). Using the example data provided in the supplementary materials it is possible to produce characterisation for all of the 96 wells on the microplate in a batch-wise manner. This dramatically decreases the time required to perform characterisation analysis whilst preserving the data in a meaningful way.

```java
package jonathansmith.dpad.api.common.plugin;

import java.util.LinkedList;

import net.xeoh.plugins.base.Plugin;

/**
 * Created by Jon on 28/05/2014.
 * <p/>
 * Core plugin class. Not to be used directly. Merely inherited either by
 * {@link jonathansmith.dpad.api.common.plugin.ILoaderPlugin}
 * or {@link jonathansmith.dpad.api.common.plugin.IAnalyserPlugin}
 * To implement this class you must compile with jspf found at
 * <a href="https://code.google.com/p/jspf/">JSPF</a>
 */
public interface IPlugin<T extends IPluginRecord> extends Plugin {

    /**
     * Return the plugin record type
     *
     * @return either an ILoaderPluginRecord or an IAnalyserPluginRecord depending on the type
     * of plugin
     */
    T getPluginRecord();

    /**
     * Return the list of tasks that this plugin should run when the user selects it
     *
     * @return
     */
    LinkedList<IPluginTask> getPluginRuntimeTasks();
}
```

Figure 38: The interface class that acts as a contract for all plugins for the DPAD software.

```java
package jonathansmith.dpad.api.plugins.tasks;

import jonathansmith.dpad.api.plugins.runtime.IPluginRuntime;

/**
 * Created by Jon on 29/09/2014.
 * <p/>
 * A contract for plugin tasks. Objects implementing this will be automatically run in a
 * thread to perform the task.
 */
public interface IPluginTask {

    /**
     * Return the task name
     *
     * @return
     */
    String getTaskName();

    /**
     * Function called when the task is going to be run.
     */
    void runTask(IPluginRuntime runtime);

    /**
     * Function called to force code within the runTask function to finish early.
     */
    void killTask(IPluginRuntime runtime);
}
```

Figure 39: The abstract task object that dictates the actions that can be performed by a plugin

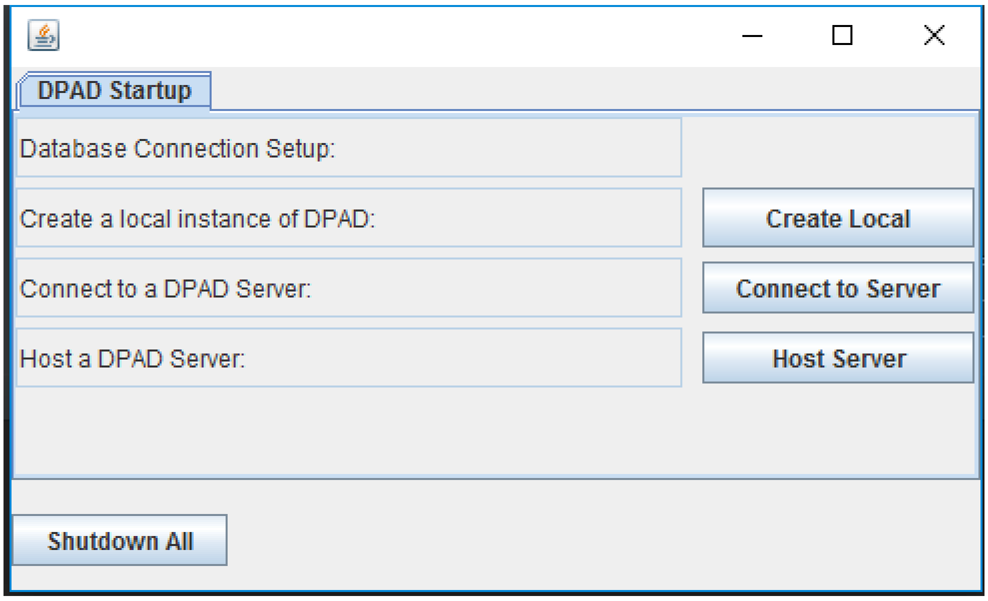
```
package jonathansmith.dpad.api.plugins.runtime;

import java.util.HashSet;

import jonathansmith.dpad.api.database.DatasetRecord;
import jonathansmith.dpad.api.database.ExperimentRecord;
import jonathansmith.dpad.api.plugins.data.Dataset;
import jonathansmith.dpad.api.plugins.display.IPluginDisplay;
import jonathansmith.dpad.api.plugins.events.IEventThread;

/**
 * Created by Jon on 29/09/2014.
 * <p/>
 * All accessible methods for plugins
 */
public interface IPluginRuntime {

    /**
     * Change the current display within the plugin runtime environment
     *
     * @param pluginDisplay
     */
    void changeDisplay(IPluginDisplay pluginDisplay);

    /**
     * Return the main event thread for the plugin environment
     *
     * @return
     */
    IEventThread getEventThread();

    /**
     * Build standard progressbar display
     */
    void buildProgressbarDisplay();

    /**
     * Record an error to the engine
     *
     * @param errorMessage
     */
    void error(String errorMessage);

    /**
     * Submit a dataset to the server
     *
     * @param dataset
     */
    void sumbitDataset(Dataset dataset);

    /**
     * Obtain experiments available to the user
     */
    void getAvailableExperiments();

    /**
     * Return lazy loaded datasets (do not contain full information)
     *
     * @param interestedRecords
     */
    void getDatasetsForExperiments(HashSet<ExperimentRecord> interestedRecords);

    /**
     * Fully load datasets
     *
     * @param lazyLoadedDatasets
     */
    void getFullDatasetInformation(HashSet<DatasetRecord> lazyLoadedDatasets);
}
```
Figure 40: IPluginRuntime interface. Acts as a contractual interface between the engine and the plugin.

```java
package jonathansmith.dpad.api.plugins.display;

/**
 * Created by Jon on 29/09/2014.
 */
public interface IPluginDisplay {

    /**
     * Return the display panel responsible for handling your toolbar
     *
     * @return
     */
    DisplayPanel getDisplayToolbar();

    /**
     * Return the display panel that will act as the core of your plugin
     * @return
     */
    DisplayPanel getDisplayPanel();

    /**
     * Called when a display becomes active
     */
    void onDisplayActivation();

    /**
     * Called when a display is updated
     */
    void onDisplayUpdate();

    /**
     * Called when a display is destroyed
     */
    void onDisplayDestroy();
}
```
Figure 41: IPluginDisplay contract. Used to create plugin displays.

```java
package jonathansmith.dpad.api.plugins.events;

/**
 * Created by Jon on 23/03/14.
 * <p/>
 * Methods for subscribing to and posting events
 */
public interface IEventThread {

    /**
     * Method to add an event listener into the event listener pool
     *
     * @param listener to add
     */
    void addEventListener(IEventListener listener);

    /**
     * Method to remove an event listener from the event listener pool
     *
     * @param listener to remove
     */
    void removeListener(IEventListener listener);

    /**
     * Post an event into the event thread for other listeners to respond to
     *
     * @param event
     */
    void postEvent(Event event);
}
```
Figure 42: IEventThread contract. Allows events to be sent and received using the core internal event thread.

7.3 DPAD Demonstration

The DPAD software was demonstrated using some of the data acquired above that charts the behaviour of IVTT when diluted. The aim for this demonstration was to show the software functioning, the user successfully loading a plugin located within the plugins folder and the user loading, processing and displaying the demonstration data.


DPAD Startup

Database Connection Setup:

Create a local instance of DPAD: Create Local

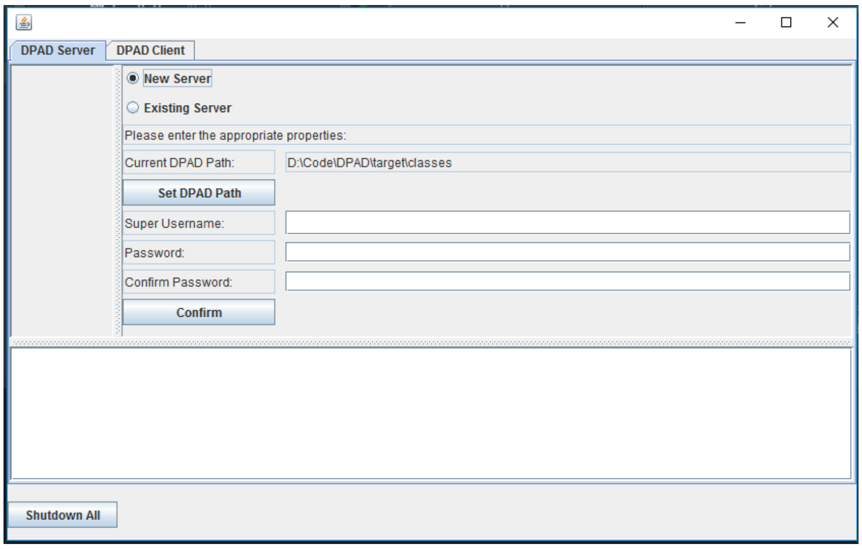
Connect to a DPAD Server: Connect to Server

Host a DPAD Server: Host Server

Shutdown All

Figure 43: The GUI created on start up of the DPAD software.

When first run, the software provides the user with a set of options on what runtime environment they wish to use with the software. These options: create local, connect to server and host server correspond to the three runtime environments described in section 7.1.


DPAD Server | DPAD Client

[✓ New Server]

Existing Server

Please enter the appropriate properties:

Current DPAD Path: D:\Code\DPAD\target\classes

Set DPAD Path

Super Username:

Password:

Confirm Password:

Confirm

Shutdown All

Figure 44: The options displayed for creating a new server within the DPAD software.

Figure 44 demonstrates the options pane that is displayed should the user choose to setup a local server or to host a server. From this pane, the user is then presented with the main client window (Figure 45).

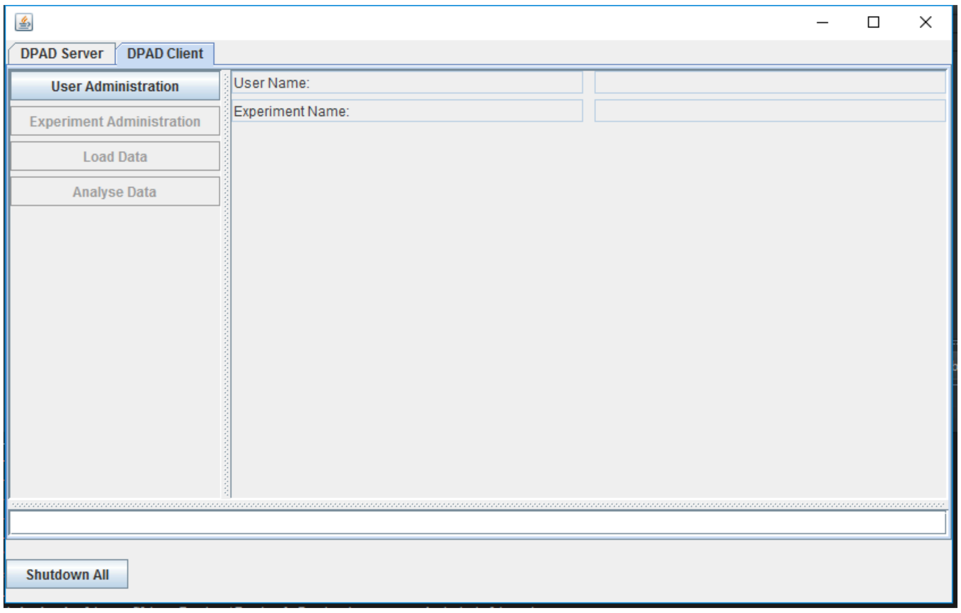
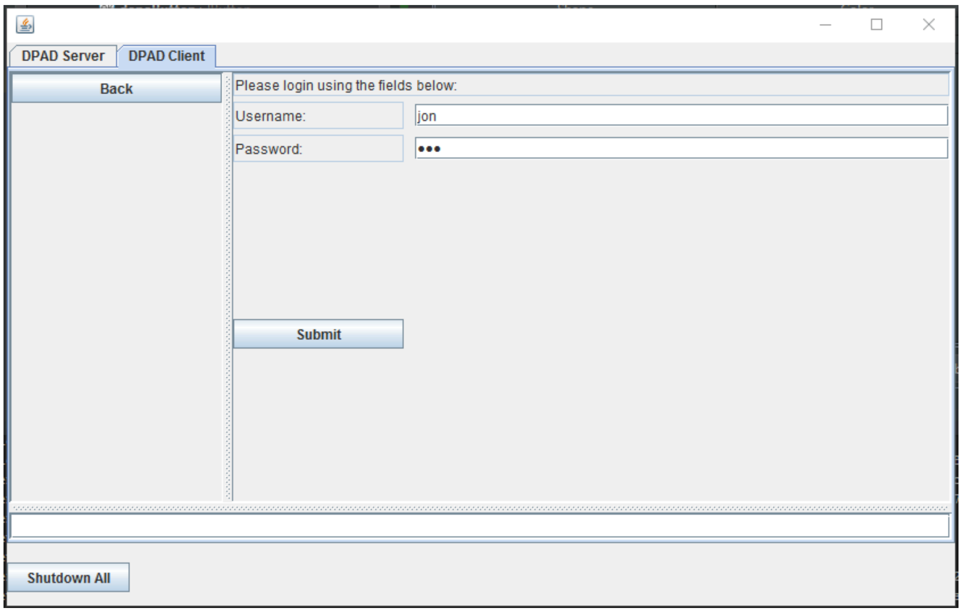


Figure 45: The main client window in the DPAD software.

The main client window, when run in local mode, will be added as a tab in the existing server window to allow both runtimes to be monitored simultaneously. The first and only step available to the user within the main client window is for the user to log in. Until this action is performed no subsequent steps are permitted for security reasons.



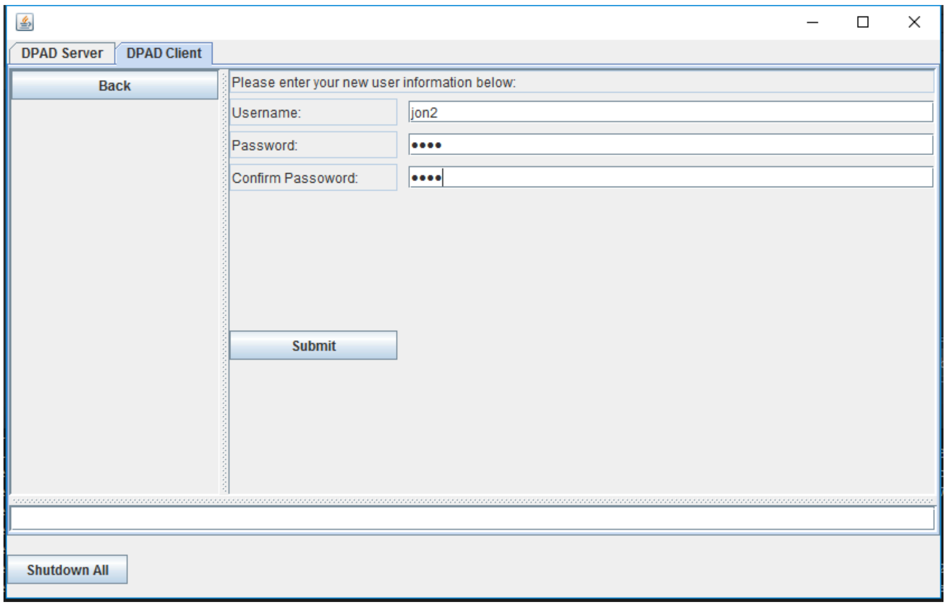
Please login using the fields below:

Username: jon

Password: •••

Figure 46: The client login window for the DPAD software.

Figure 46 shows the client's login window. An authentication attempts is sent, via encrypted key exchange, to the server to confirm a user's identity. Depending on how the server is configured, open registration of users can be enabled, which provides the 'User Administration' panel with the additional option of adding a new user (Figure 47).



Please enter your new user information below:

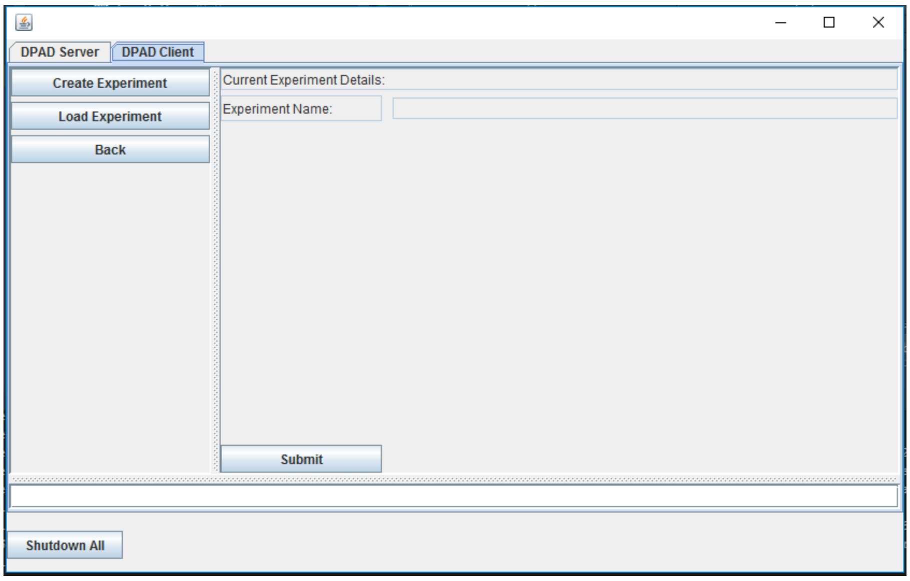
Username: jon2

Password: ••••

Confirm Passoword: ••••

Figure 47: The new user window for the DPAD software.

Once the user has successfully registered and logged on, the user's network state is switched to the secure communication protocols as described in section 7.2.1.

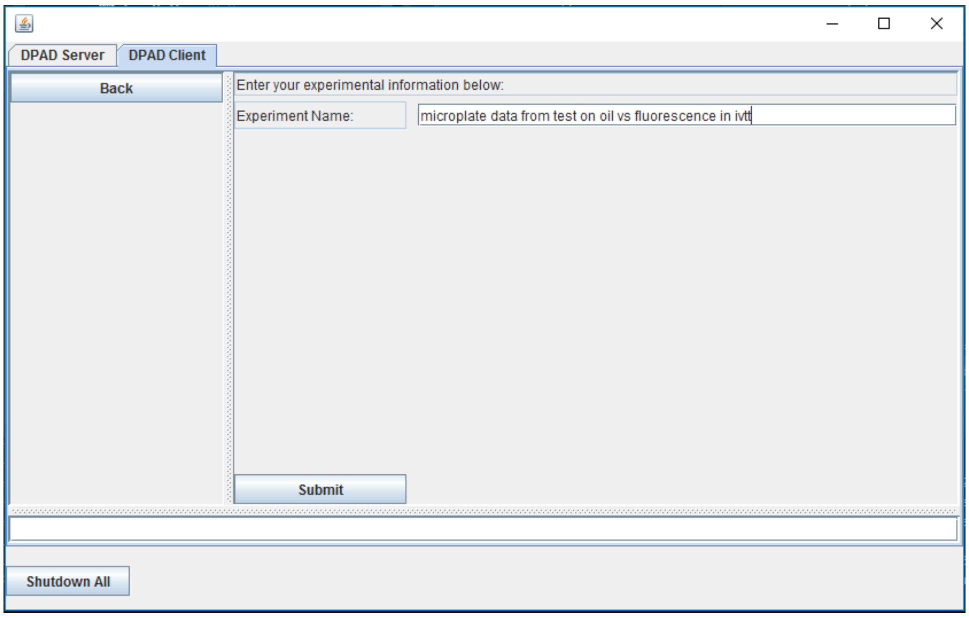


Current Experiment Details:

Experiment Name:

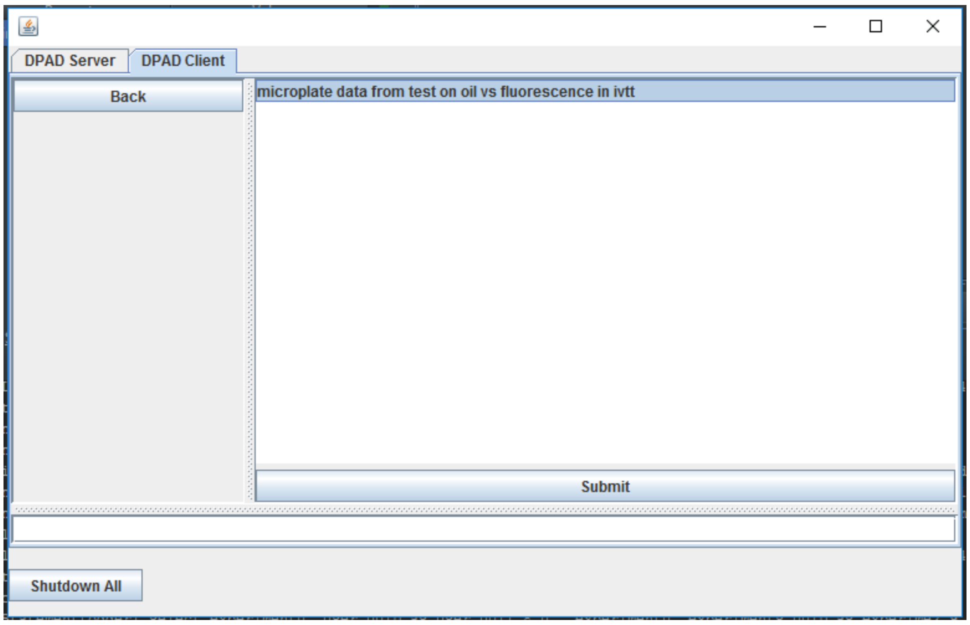
Figure 48: The experiment administration section of the DPAD software.

Figure 48 displays the experiment administration panel which allows users to create a new experiment or to load an existing experiment. As described in section 7.2.2, an experiment provides an object aggregate for any data that gets loaded into the system. As a general rule, data analyses are done on entire experimental data sets (and are not limited to just data loaded in one iteration) as the information is usually contextually relevant.



Enter your experimental information below:

Experiment Name: microplate data from test on oil vs fluorescence in ivtt



microplate data from test on oil vs fluorescence in ivtt

Figure 49: The experiment creation screen for the DPAD software. The upper image shows the screen displayed when a new experiment is selected whilst the lower displays the screen displayed when the user attempts to load an existing experiment.

Once the experiment has been set for the work that is about to follow (either for data loading or for data analysis), it is considered contextually relevant to all future actions, until the experiment is switched. This implies that all operations performed by the user subsequent to this step, become associated with the experiment object.

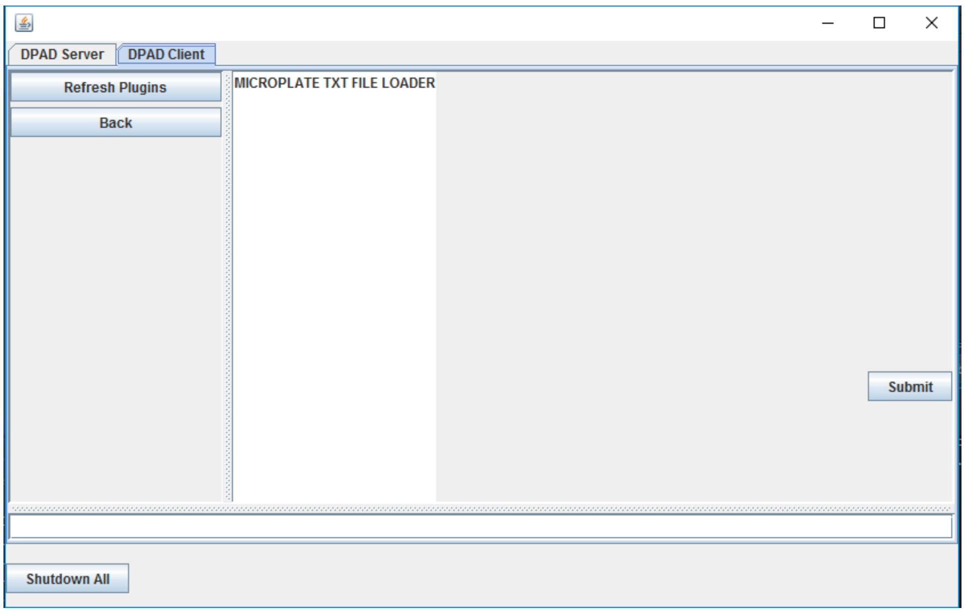
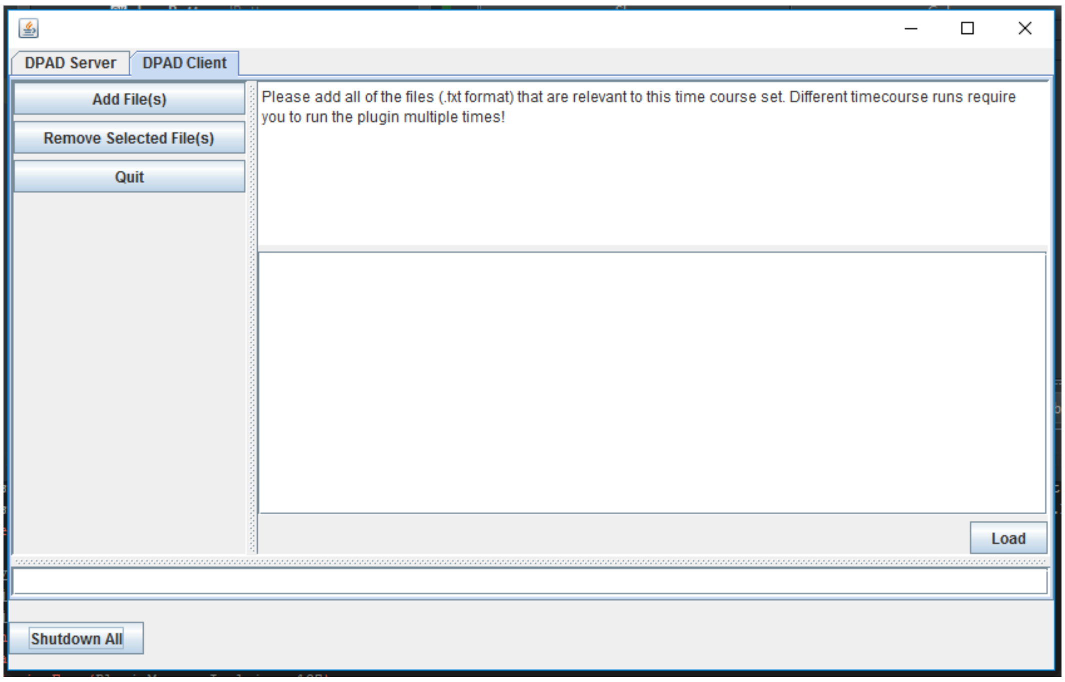


Figure 50: The screen displayed in the DPAD software when Load Data is selected on the main panel.

Figure 50 demonstrates the screen that is displayed once the load data option has been selected from the main navigation window. The ability to load data is only available once the experiment has been successfully set. Figure 50 demonstrates the first visual implementation of section 7.2.3 as a list of available loading plugins has been dynamically generated (and can be refreshed if the user changes them mid-session).

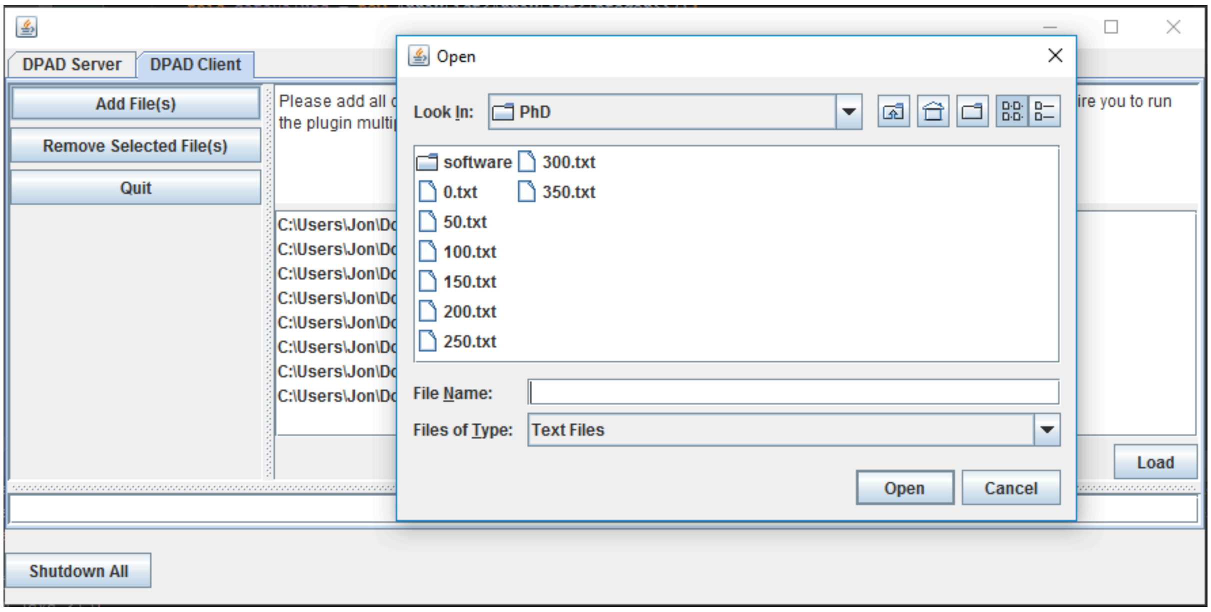
In this demonstration the only available plugin displayed is the ‘Microplate TXT File Loader’, which was designed to load longitudinal (i.e. timecourse) data from a microplate experiment into the DPAD software. Once this plugin is selected it is dynamically loaded into the Java classpath to allow its use within the DPAD software.



Please add all of the files (.txt format) that are relevant to this time course set. Different timecourse runs require you to run the plugin multiple times!

Figure 51: Step 1 in the microplate text file loader plugin.

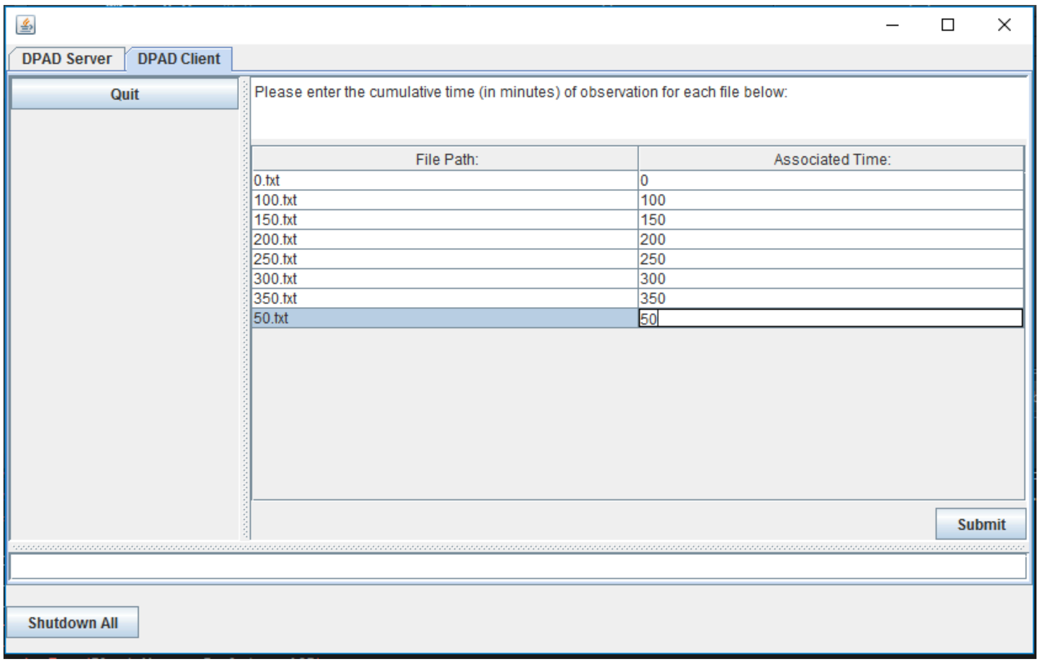
Figure 51 displays the first logical step performed by the microplate plugin, the ability to track files for loading. By navigating through a file selection dialog (Figure 52) the user is able to select all of the files pertinent to a longitudinal study.



Please add all c
the plugin multi
C:\Users\Jon\Dc
C:\Users\Jon\Dc
C:\Users\Jon\Dc
C:\Users\Jon\Dc
C:\Users\Jon\Dc
C:\Users\Jon\Dc
C:\Users\Jon\Dc
C:\Users\Jon\Dc
ire you to run

Figure 52: The modal dialog box displayed to the user within the microplate text file loader plugin for file selection.

Once the user has selected the files of interest, they are then provided with a screen that allows the assignment of time contexts to each of the files (Figure 53).

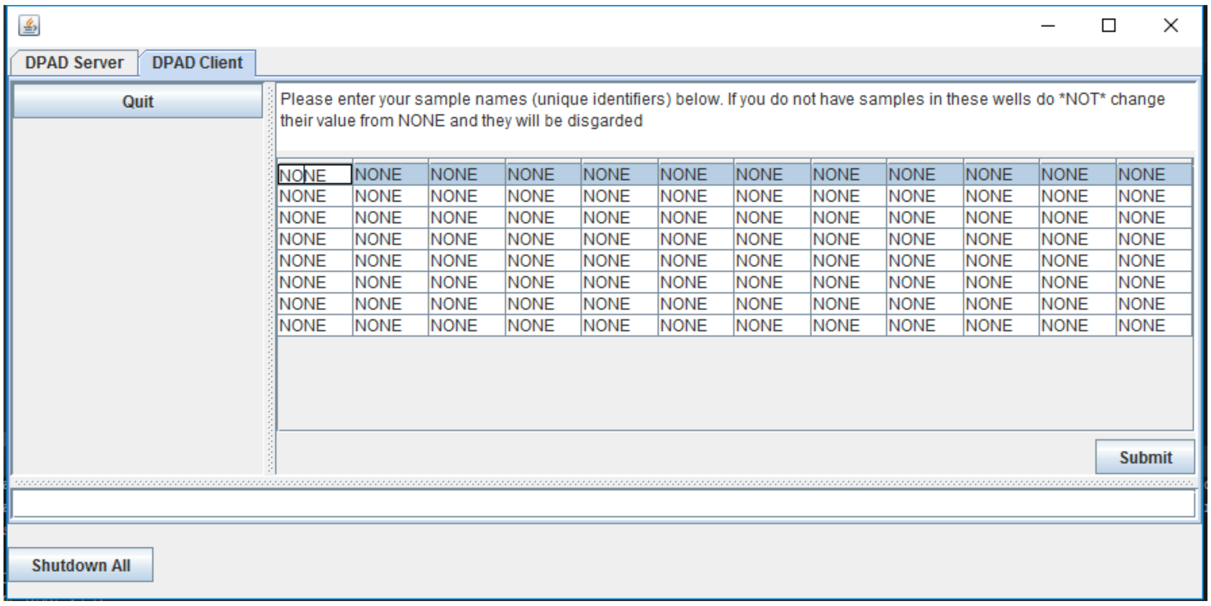


Please enter the cumulative time (in minutes) of observation for each file below:

File Path:	Associated Time:
0.txt	0
100.txt	100
150.txt	150
200.txt	200
250.txt	250
300.txt	300
350.txt	350
50.txt	50

Figure 53: Step 2 in the microplate text loader plugin.

The final step in the microplate text loading plugin is to assign the sample names to the wells provided. For this specific implementation, the number of wells displayed is dynamically generated depending on the files provided, ensuring that microplate size is always correct with regards to your data.



Please enter your sample names (unique identifiers) below. If you do not have samples in these wells do *NOT* change their value from NONE and they will be disgarded

NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE
NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE	NONE

Figure 54: Step 3 in the microplate text loading plugin.

Once the user has provided this information, the plugin then loads the data into database for permanent storage. The result of this loading process is shown in Figure 55, which shows a graph of the data loaded by the previous plugin.

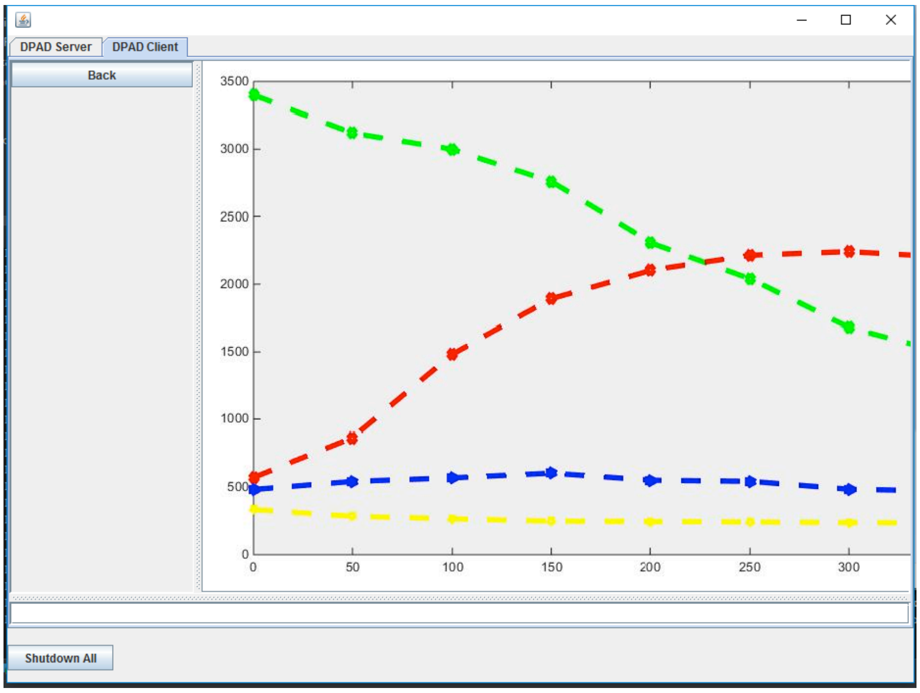


Figure 55: The outcome of loading data into the DPAD software using the microplate text file loading plugin.

7.4 Outcomes

The DPAD software represents a novel approach to handling characterisation data, one which incorporates all of the stages necessary to produce high quality, meaningful information. By breaking down the software into several key architectures, the internal runtime environment for the software have become sufficiently abstract to allow a flexible plugin framework to function. Employing a plugin framework allows the software to be both versatile and robust in its application with potential to serve multiple data types across all fields of research. It has been tested on Windows 7 but should be completely cross-platform compatible. The source code for the software can be found at http://www.github.com/JonathanCSmith/dpad and is licenced under the lesser general public licence (LGPL) version 2.1 and as such is fully open source.

The DPAD API is the core enabling feature of the plugin framework. It provides the ability to access the software in an abstracted and modular manner. Primarily, access occurs through the IPluginRuntime interface (Figure 40) which acts as a contract for all plugin – server interactions. IEventThread (Figure 42) and IPluginDisplay (Figure 41) are also contractual obligations, where the first allows the plugins to send events within the software, and the second allows plugins to submit their own displays. See the supplementary information for the full DPAD API. Section 7.3 demonstrates the power of this architecture, showing that the loading / processing and analysis logic is completely separated from the core runtime logic of the main program by demonstrating the start to finish loading of a ubiquitous data type. This separation allows new plugins to be created over time, as well as improvements to be made. Moreover, the plugin architecture allows organisations to build locally relevant plugins that can reduce and standardise workloads by standardising data workflows.

It was imperative that any database schema created when designing the software imposed minimal structure onto the disparate data types that are likely to arise within characterisation work. Achieving this allowed potential plugins to be as diverse as possible, which in turn ensures that as the field of characterisation evolves, the software will be capable of meeting those evolving needs. The use of a loose context information tagging system allows for future compatibility with complex plugins such as datasheet generation (Canton et al., 2008), as information can be appended to datasets at any point.

Alternative database software applications are also available for data handling; however, many of these software applications are not freely available. Much of the database software that is comparable to the DPAD software is based around the online parts registries (e.g. www.partsregistry.org and http://biofab.synberc.org/), which do not openly declare the underlying software architectures. The paper by Macdonald *et al.*(MacDonald et al., 2011) discusses some of the potential uses and approaches for software application in Synthetic Biology but predominantly focuses on CAD, modelling or DNA assembly software tools rather than software under the data handling umbrella. The closest software examples from a function standpoint are the commercial Laboratory Information Management Systems (LIMS) which aim to track any and all lab information and manage workflows. The DPAD software offers some of the same flexibility in data handling, with easy to use graphical user interfaces (GUIs) and centralised data storage; however, many of the key advantages of the DPAD software are not found elsewhere (such as the plugin framework) and will expand the usability and versatility of the platform as a whole.

The software includes a high functioning graphing package (JFreeChart - http://www.jfree.org/jfreechart/) that can produce high quality publication level graphs that are capable of being completely written to .pdf file formats. This is of great interest for future version of the software as it allows the complete and automated production of the aforementioned datasheets (Canton et al., 2008). In turn, this would likely increase the prevalence of both characterisation and datasheets with the Synthetic Biology pipeline.

Future versions of the software could move to a more robust permissions system which would allow for a more hierarchical data access topology. By doing this it is possible to associate not only a user with their data, but also grouping users with a laboratory. A topology like this would improve the lateral transfer of information between collaborative users. This is important as other users can weigh in on data to provide previously unseen insight and increase the collective information processing abilities of grouped users. The software could also be developed to include a third plugin type responsible for 'tagging' samples and annotating them properly. Doing this allows a layer of quality control to be applied at the final data analysis stage of the runtime environment, (i.e. data analysis plugins could refuse to accept data that had not been processed by specific plugins). Employing a quality control assurance step into the data analysis process may decrease the independence of individual plugin implementations. It would help assure that any data submitted to the database follows defined conventions and constraints. By standardising the data prior to analysis it is possible to limit the wide variety of contextual information available to experimenters to only that which is necessary for the analysis to be fully realised. Some features of the software can still be improved upon, especially when large datasets become a concern as the user has to sift through multiple data entries before finding their desired dataset. To improve the software, future iterations will make use of more sophisticated search algorithms and better refinement of database queries.

8 Materials and Methods

8.1 Biological Techniques

8.1.1 DNA Manipulation

Predominantly the DNA used in this study involved a standardised expression vector where only the promoter region preceding the target for expression (GFP) was varied. To create these vectors a common methodology set was used as detailed below. Essentially this methodology consisted of created a standard linear stretch of DNA corresponding to a promoter-less expression vector and then inserting the promoter targeted for characterisation.

The linearised empty expression vector was created by obtaining the parts listed in Table 8 (without the promoter region) and assembled according to the BioBrick^{TM} Assembly Method (see Figure 2 for more information) (Shetty et al., 2008). A schema of the archetypal expression vector can be found in Figure 56. The circular backbone was then inserted into *E. coli* (section 8.1.7) for replication and purified (section 8.1.8). The backbone was then linearized by digestion (section 8.1.2) generating specific overhangs designed for ligation with other BioBrick^{TM} parts. The backbone inserts (i.e. the promoters) were created by designing and obtaining two complementary oligonucleotides that when annealed (section 8.1.3) resulted in a known promoter region, affixed on both ends with a predetermined sequence corresponding to a complementary overhang to those generated by the digestion of the circularised backbone, an example of this can be seen in Figure 57.

Promoter sequences were either obtained from the BioBrick^{TM} parts registry or from the BioFab registry (www. biofab.synberc.org) and adjusted using a short MATLAB script (Figure 60) to contain the necessary overhangs. The sequences for each of the primers can be found in Table 10 (section 10.1). All oligonucleotides were obtained from Integrated DNA Technologies Inc.

Table 8: GFP Expression vector parts list. Variable indicates a promoter region corresponding to the oligonucleotide annealing produced using the sequences found in Table 10 *Naming convention from BioBrick^{TM} Registry

Part Type	Part Name
Constitutive Promoter	Variable
Ribosome Binding Site (RBS)	BBa_B0034 *
Fluorescence Reporter	GFPmut3b
Terminator	BBa_B0015 *
Vector Backbone	pSB1A2

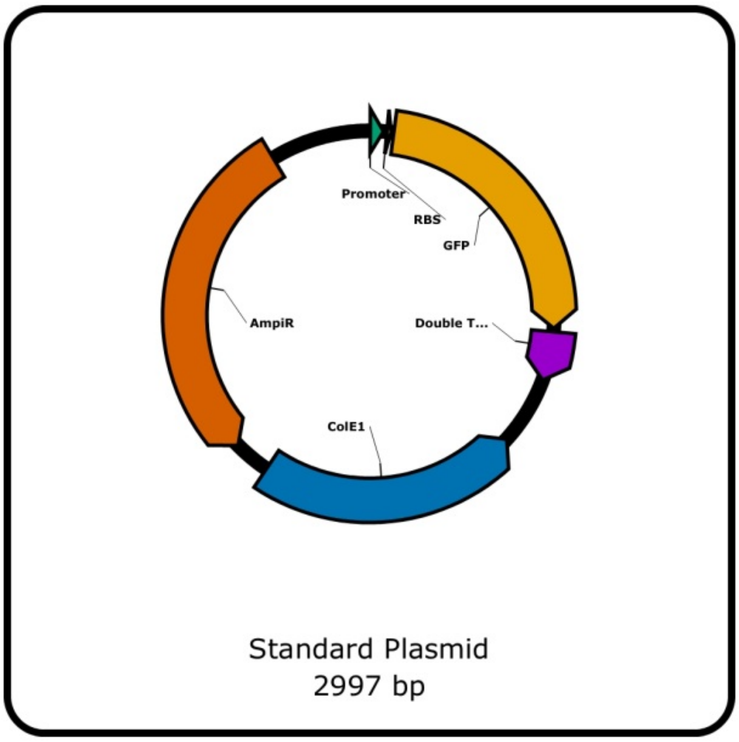


Figure 56: A plasmid map depicting the archetype expression vector's component parts. The archetypal plasmid contains the parts listed in Table 8

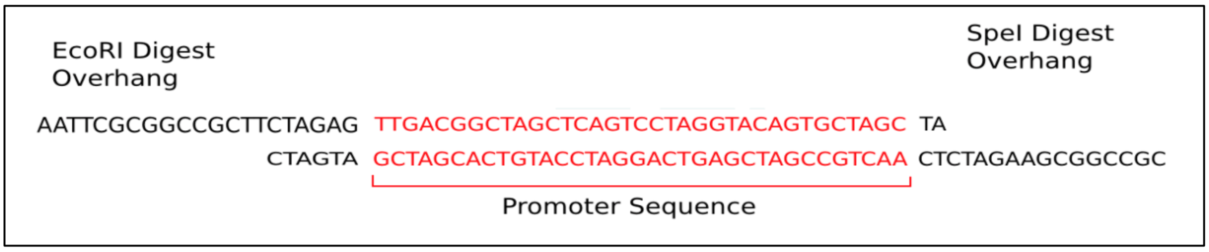

EcoRI Digest Overhang
SpeI Digest Overhang
AATTCGCGGCCGCTTCTAGAG TTGACGGCTAGCTCAGTCCTAGGTACAGTGCTAGC TA
CTAGTA GCTAGCACTGTACCTAGGACTGAGCTAGCCGTCAA CTCTAGAAGCGGCCGC
Promoter Sequence

Figure 57: An example depicting the primer design used to insert promoter regions into a pre-cut expression vector backbone. The promoter region is couched in sequences corresponding to an EcoRI and SpeI double digestion.

8.1.2 Digestion

Digestions were performed using EcoRI and XbaI (according to the New England Biolab guidelines) at 5 units of enzyme per µg of target DNA. Enzyme volume never exceeded 10% of total digestion volume and glycerol volumes were maintained below 5% of total digestion volume. Digestions were incubated for two hours at 37^{o}C.

8.1.3 Primer Annealing

10µl reactions containing 10ng per µl of each primer oligonucleotide were annealed using 1µl of 10x annealing buffer (10mM Tris pH 7.5-8.0, 50mM NaCl, 1mM EDTA) and double distilled H_2O. Reactions were heated within a PCR block to 85^{o}C and allowed to slow cool to room temperature.

8.1.4 Ligations

10µl ligation reactions were performed using an insert to vector molar ratio of 3:1 with 25ng of vector. T4 DNA ligase was used according to the manufacturer's guidelines (New England Biolabs) and incubated at room temperature for approximately two hours. Insert and vector concentrations were estimated using agarose gel electrophoresis (section 8.1.5).

8.1.5 Agarose Gel Electrophoresis

Agarose Gels were prepared using a 1% weight to volume ratio of agarose to Tris-Acetate-EDTA (TAE) with 0.5x GelRed (Cambridge Bioscience) added as a nucleic acid stain. Gels were loaded with samples using DNA loading buffer (10mM EDTA, 50% Glycerol, 0.1% Bromophenol blue) and the sample ladder was 1Kb Plus DNA Ladder from Invitrogen. Gels were run in TAE buffer at 100V for approximately 40 minutes. DNA fragments were visualised using a UV trans-illuminator (BioRad).

8.1.6 Electroporation Cell Preparation

Electrocompetent XL1-Blue (Agilent Genomics) *E. coli* cells were prepared by growing cells to mid-log phase in LB broth (1% w/v bactotryptone, 1% w/v NaCl, 0.5% w/v bacto-yeast extract, double distilled H_2O) containing 20µg per ml of tetracycline at 37^{o}C. Cells were then pelleted by centrifugation at 4000 x g for 15 minutes at 4^{o}C. Pelleted cells were first washed with double distilled ice cold H_2O and re-suspended using a 1/400 volume of 10% v/v ice cold glycerol and double distilled H_2O. Cell suspensions were stored in 50µl aliquots at -80^{o}C.

8.1.7 Electroporation

Electroporation was performed using 50µl of electro-competent XL1-Blue cells prepared as per section 8.1.6 and 1µl of plasmid DNA or post-ligation DNA. Transformation was performed inside a 0.1cm electro-cuvette using a BioRad Gene Pulser under settings of 200Ω, 25mF and 1.67kV. 450µl of LB broth was then added to the cells and then incubated at 37^{o}C for 30 minutes. After incubation cells were plated on LB agar containing ampicillin (100µg per ml) and incubated overnight at 37^{o}C.

8.1.8 DNA Purification

DNA purification was performed using Qiagen QIAfilter Plasmid Maxi Kit according to the manufacturer's guidelines.

8.1.9 DNA Concentration Quantification

DNA Concentration and purity was quantified using a Thermo Scientific NanoDrop^{TM} 1000 Spectrophotometer using 1µl of target DNA according to the manufacturer's guidelines.

8.1.10 GFP Purification

An expression vector containing a polyhistidine-tagged GFPmut3b was constructed by inserting GFPmut3b into a pProEX-Htb backbone (obtained from Invitrogen) which contains the polyhistidine-tag DNA sequence. *E.* coli XL1-Blue electrocompetent cells were transformed using the vector. Colonies were picked from the resultant plates and grown in 4ml of LB broth with ampicillin (100µg per ml) and grown for 6 hours at 37^{o}C. 1ml of this was used to inoculate an overnight culture consisting of 100ml of LB Broth with ampicillin (100µg per ml) for growth under the same conditions. 50ml of this overnight culture was then used to inoculate 1000ml of LB with ampicillin (100µg per ml), which in turn was grown under the same conditions until an O.D. 600 of 0.5 was observed. 1mM of isopropyl β-D-1-thiogalactopyranoside (IPTG) was then added to induce GFP production. After a further 3 hours of growth at 37°C the cells were centrifuged at 4000 x g for 15 minutes and the supernatant was removed.

Cells were then re-suspended in 50ml of Buffer A (10mM imidazole, 150mM NaCl, 50mM Tris pH 7.5 and 5% glycerol) and sonicated. Sonication was performed twice with a minute rest in-between using two second pulses at 50% amplitude. The cell lysate was then centrifuged at 17,000rpm for 45 minutes and the supernatant loaded into a prepared 1ml HisTrap column in an ÄKTA System (GE Life Sciences).

The HisTrap column was prepared by equilibrating the column with 10 column volumes (CV) of Buffer A. The system was run at 0.8ml per minute with UV detection set to 280nm. Supernatant was loaded after the Buffer A. Proteins without His-Tags were removed by adding 10 CV of 10% Buffer B (500mM imidazole, 150mM NaCl, 50mM Tris pH 7.5 and 5% glycerol) after which a gradient of 10-100% Buffer B was applied causing target protein elution.

The desired elution fraction (containing purified GFP) was identified first by fluorescence and then through 15% SDS-PAGE for purity assessment.

8.1.11 *In Vitro* Transcription-Translation Preparation

E. coli S30 circular extract cell-free system was prepared according to Promega Ltd.'s guidelines and used as the IVTT expression medium throughout the course of this study. The expression solution is composed of two core reagents, the cell extract and the pre-mix which contains all the prerequisites for protein expression, such as: tRNAs, DNTPs, amino acids and energy regenerating mechanisms (Katzen et al., 2005). 1µl of approximately 1000ng/µl of expression vector DNA (purified as per section 8.1.8) was added to the IVTT mixture per reaction.

8.1.12 Microplate Assays

Reactions were prepared in 96 well plates (Greiner Bio-One) and measured in a POLARstar Omega plate reader from BMG Labtech. Measurements were made by exciting the samples with 485nm wavelength light and emission was detected at 520nm. Measurements were taken every 15 minutes and the reactions were held at 30^{o}C for four hours.

8.2 Microfluidic Techniques

8.2.1 Microfluidic Chip Design

Designs for the microfluidic chips were created using the AutoCAD software framework. Designs were printed to acetate masks by Microlithograph services Ltd. The devices were printed in negative (channel locations were clear, channel exterior was blacked out) to facilitate fabrication stages.

8.2.2 Microfluidic Chip Fabrication

Microfluidic chips were fabricated using two core methodologies. Each presented their own advantages and disadvantages and are discussed in section 8.4. All microfluidic fabrication steps were performed in a clean room environment to reduce environmental contamination of the chips.

Permanent chrome masks were generated from the acetates to prevent acetate degradation (which would affect the quality of the chips). Chrome masks were etched into soda-lime glass plates coated in chrome and AZ 1518 positive photoresist (MicroChemicals). The acetate masks were secured to the glass substrate and exposed to UV light (350-400nm) using a UV lamp (Optical Associated, Inc.) at 20mW per cm^2 for 10 seconds. Areas of positive photoresist exposed to the UV light (i.e. the channels) become removable using a stripper solution (AZ 100 MicroChemicals) leaving unprotected chrome in its wake. The chrome was then etched using Standard Chromium Etchant (Sigma Aldrich) for 10 seconds and washed with double distilled H_2O resulting in an exact copy of the original acetate but preserved on a chrome/glass plate.

Silicon wafer masters (a cast for chip fabrication) were created by first cleaning a blank silicon wafer using a mixture of hydrogen peroxide and sulphuric acid (Pirahna Etch 3:1 ratio created by slowly adding hydrogen peroxide to the sulphuric acid). A thin layer of negative photoresist was deposited across the clean wafer (SU-8 MicroChemicals). Either SU-8 50 or SU-8 100 was used depending on the desired channel depth (predominantly SU-8 100 was used). SU-8 was deposited evenly across the wafer surface using a spin coater (Laurell Technologies Corp.) with different spin speeds and durations depending on the desired photoresist thickness (Table 9 shows the various combinations used to obtain specific channel thicknesses). Each spin speed was preceded by a ramp up time of 5 seconds starting from 300rpm. Wafers were then soft baked to remove excess solvent (according to Table 9) consisting of a pre-bake at 65^{o}C and a soft bake at 95^{o}C.

Chrome masks were placed on the wafers and exposed to UV light, under the same conditions as described previously, according to the desired channel depth (Table 9). Areas of negative photoresist exposed to UV light (the channels) undergo a chemical crosslinking which prevents SU-8 developer from removing it. Crosslinking is allowed to progress during a final-bake at 95^{o}C.

Finally, the wafers were treated with SU-8 developer (MicroChemicals) for 5 – 10 minutes during which the undeveloped photoresist is removed (corresponding to the areas of the design that are not part of the microfluidic channels). This section of device fabrication corresponds to the first step shown in Figure 58**.**

Table 9: Master fabrication properties according to channel depth

Channel Depth (µm)	SU-8	Spin Speed (rpm)	Spin Duration (sec)	Pre-Bake Duration (min)	Soft-Bake Duration (min)	UV Exposure Duration (sec)	Final-Bake Duration (min)
30	50	3000	40	5	10	5	10
50	100	3000	30	6	20	12	20
100	100	2000	30	10	30	15	45

Polydimethylsiloxane (PDMS) was used as the substrate for the microfluidic chips. PDMS was created using a Sylguard 184 kit (Dow Corning Corp.) and prepared in a 10:1 ratio of substrate and catalyst. The liquid PDMS was then poured over the silicon master (Figure 58 step 2). Excess air was removed from the cast and PDMS by placing them within a vacuum chamber for 30 minutes (preventing air bubbles from distorting the device when they expand during the curing phase). PDMS was cured on a hotplate at 75^{o}C for 90 minutes. Chips were removed by cutting the PDMS around the device design and processed in two distinct manners generating two distinct chip types (Type 1 & Type 2) both of which are shown schematically in Figure 42.

Type 1

Type 1 fabrication produces extremely sturdy microfluidic chip that use a microscope slide as a support to increase stability. Microscope glass slides were marked with the input and output locations using the acetate masks. A 1mm diamond coated drill bit was then used to carefully pierce the slide without cracking the glass. The slides were then washed using a sonicator water bath, first in 96% filtered ethanol and then in de-ionised water, for 10 minutes each.

The slides were then bonded to the PDMS chip using a plasma cleaner (Diener Electronics) under vacuum by carefully aligning the input/output ports and the holes drilled in the slides. The orientation of the bonding located the slides against the surface of the PDMS that did not contain the channel moulds (therefore the device is not yet sealed). Micro capillary tubing was pushed through the holes of the slides, through the PDMS into the voids corresponding to the input/output port channels. The sections of tubing that pierced the PDMS were removed to prevent clogging and the entire device was cleaned in a sonicator as stated previously. The channels were then sealed by plasma bonding a microscope coverslip to the other PDMS surface. The tubing was secured in place (and the holes in the glass slide were sealed) using a two part epoxy resin ensuring that no gaps were left in the slide. The chips were given a final bake in an oven at 65^{o}C overnight.

Type 2

Type 2 fabrication requires much less time to achieve as it uses no chip support. Fabrication begins by partially curing an extremely thin layer (less than 1mm) of PDMS at 65^{o}C for 7 minutes (until the surface was a tacky or a stringy solid rather than a liquid). Meanwhile the PDMS chips were pierced using a 1mm biopsy needle at the locations of the input and output ports. The chips were then applied (channel side down) to the partially cured PDMS and allowed to fully cure overnight at 65°C in an oven.

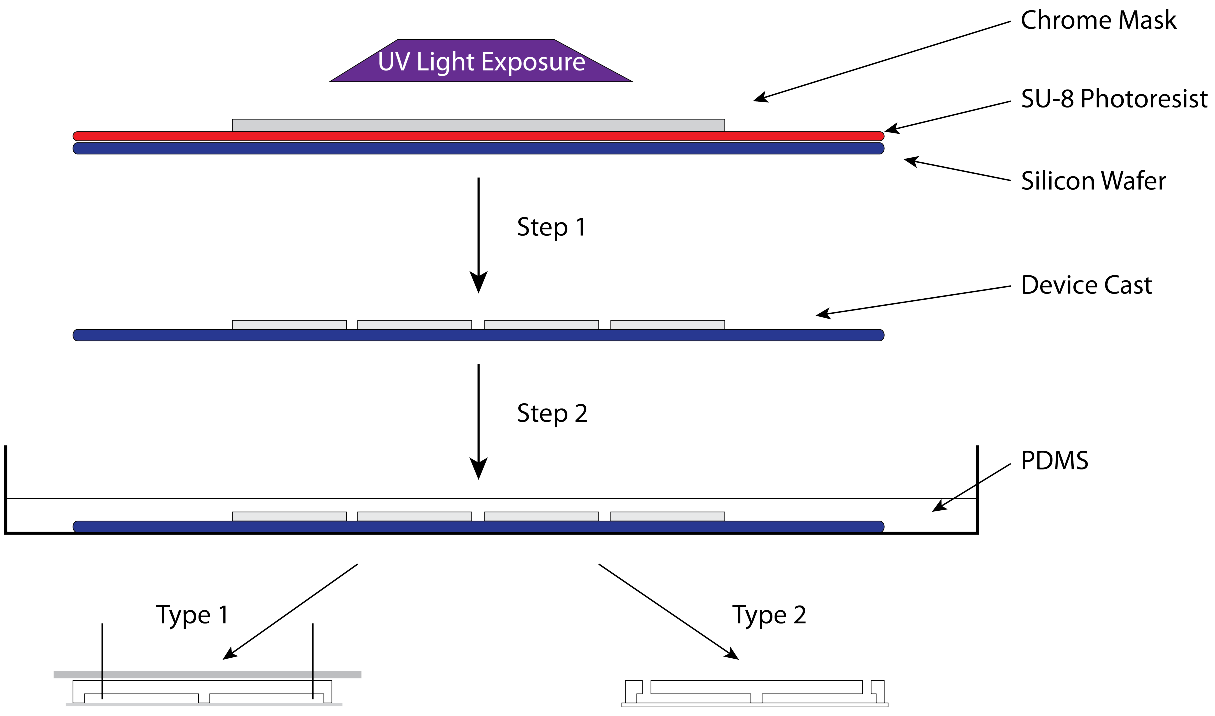


Figure 58: Overview of microfluidic chip fabrication including the divergent chip types. Type 1 and Type 2 differ in their overall rigidity of the final chips. 1 chips are extremely robust but take many man hours to manufacture. Type 2 chips tend to be quick to manufacture but have a shorter shelf life.

8.2.3 Microfluidic Chip Treatments

Devices were occasionally (see section 6) treated with Duxback solution (DuxBack Ltd.) to prevent leakage, contamination and decrease the 'stickyness' of the walls with respect to droplets. Duxback works by fluorinating the PDMS channel surface and therefore increasing its hydrophobicity.

8.2.4 Microfluidic Chip Oil Phases and Surfactants

A variety of oil phases and surfactants can be used with microfluidic devices. In this study either mineral oil (Sigma Aldrich) or Fluorinert® FC-40 oil (Sigma Aldrich) was used. Adding surfactant to the oil phase serves to reduce droplet merging. Mineral oil was combined with either Abil EM 90 (3% v/v - Golsdschmidt GmbH) surfactant or Span 80 (1.5% v/v - Sigma Aldrich) surfactant. FC-40 was combined with either RainDance^{TM} surfactant (1.8% wt/wt ratio, RainDance Technologies Inc.) or 1H,1H,2H,2H-Perfluoro-1-octanol surfactant**.**

8.2.5 Compartment-on-demand Robot Software

Originally the compartment-on-demand robot (Gielen et al., 2013) used a custom LabVIEW (National Instruments Corp.) workspace, however, a custom software implementation of this workspace was also created, using the Java programming language to allow computers without the LabVIEW software to use the robot. The software used the freely available Phidget drivers and Phidget Java Application Programming Interface (API) (http://www.phidgets.com/) to interface with the carousel of the robot, and solenoid control was achieved through the RXTX Java API (https://github.com/rxtx/rxtx).

8.3 Detection Techniques

8.3.1 Avalanche Photodiode Detector (APD)

Detection was performed by combining an Omnichrome 488nm CW air-cooled argon ion laser (Melles Griot) with a confocal microscope (Zeiss) and an avalanche photodiode detector. The laser was operated at 60-80% power unless stated otherwise. The path of the laser is depicted in Figure 59. The laser first passes through a glass filter (usually a 1/20th filter) and a polariser to attenuate the laser intensity. The laser beam then passes through an iris to ensure alignment after which it is reflected by a dichroic mirror (DC2: HQ505DCLP, AHF Analysentechnik AG) into the 60x objective lens. Emission is passed back to the dichroic mirror (corresponding to an emission wavelength of 511nm) which allows the emitted light to pass. Emitted light is filtered to reduce non-specific light and passed through a pinhole, finally leading to the APD detector.

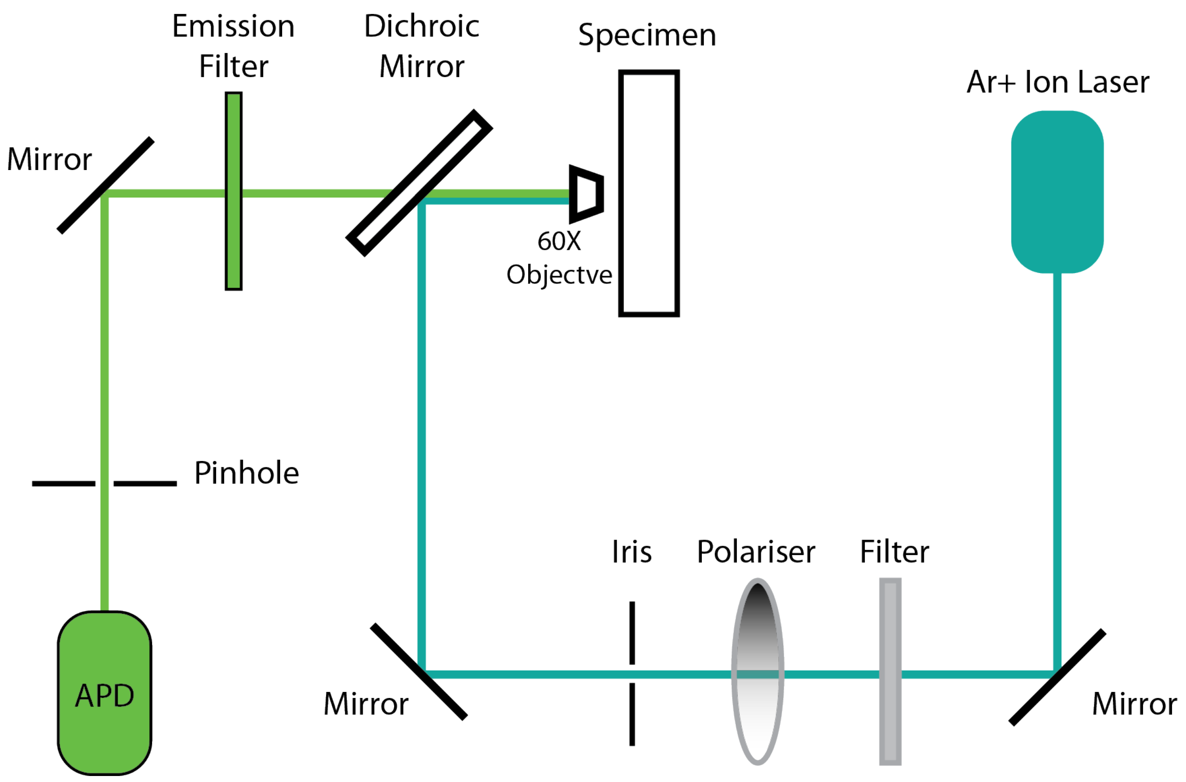


Figure 59: The APD confocal microscope setup used in the investigation. Primarily, the diagram depicts the light path for the argon laser and how it is attenuated for both excitation of the sample and how the data is recorded at the APD.

8.3.2 Fluorescence Wide-Field Microscope

Wide-Field Fluorescence Microscope assays used a TI-E Super-research Inverted Microscope (Nikon Ltd.) with a mercury lamp excitation source and a C-FL Epi-Fl GFP-B filter set. Detection was performed using a CoolSnap HQ2 CCD (Photometrics). Data capture was performed by using NIS Elements Advanced Research software.

8.4 Analysis Techniques

8.4.1 APD Droplet Detection

Data analysis for the APD data was performed as per section 6.5.2 using a custom set of MATLAB scripts.

8.4.2 Image Analysis

Images were loaded into ImageJ (from Nikon's proprietary image format '.nd2') using the Nikon ND2 reader. Image stacks were converted to .tiff and separated into individual images. ImageJ was used to generate overlay images whilst custom MATLAB scripts were used to calculate the time course fluorescence profile of the droplets (as per section 6.5.3).

8.4.3 Characterisation Data Analysis

To analyse constitutive promoter data, repeat droplets were averaged to calculate the mean fluorescence value per sample per time point.

Next, the times corresponding to steady state GFP production for the dataset as a whole were identified. The two closest data points to the previously established steady state GFP production were then used to calculate a rate of fluorescence production as per: Equation 1 for each sample investigated.

$$Rate\ of\ Fluorescence\ Production(X) = (Fluorescence(X)_{Time2} - Fluorescence(X)_{Time1})/(Time2 - Time1)$$

Equation 1: Equation for calculating the rate of fluorescence production for a given sample (X)

Relative strengths of the promoters can then be calculated by referencing the internal standard (BBa_J23100) using the equation shown in Equation 2.

$$\mathit{Relative\ Strength}\ (X)$$

$$= \mathit{Rate\ of\ Fluorescence\ Production}(X)/\mathit{Rate\ of\ Fluorescence\ Production}\ (\mathit{Std})$$

Equation 2: Equation for calculating the relative strength of a constitutive promoter using a reference

8.5 Data Processing Analysis and Display Software

For the D-PAD software two languages were predominantly used; firstly MATLAB was used as the core programming language for the system as work was being done in parallel with this language already. As the complexity of the software increased the language was changed to JAVA so that greater flexibility in the user interface as well as a more robust software framework could be achieved.

9 Conclusion

In conclusion this study attempted to improve the uptake of characterisation within the Synthetic Biology research pipeline by providing enabling technologies for the characterisation speciality. By reducing the apparent complexity and time involvement of current characterisation techniques it is hoped that more researchers will employ characterisation within their workflow. Fundamentally, this would improve the capability of predictive modelling as well as improve the amount of communal information available for biological parts, devices and systems.

Specifically the work above has developed a prototype characterisation platform that could potentially fulfil the current gap in high throughput characterisation technologies using some of the newly developed solutions to the lack of robust technologies available in the microfluidics field. By utilising more robust version of the compartment-on-demand microfluidic robot in combination with either shorter tubing or improved suction, the realisation of a massively multiplexable, rapid characterisation technology is entirely plausible. It should be acknowledged that the current platform iteration is not suitable for high throughput characterisation work as the immaturity of the microfluidics field leads to a high user involvement in the platform's operation. The platform suggested above has successfully shown both the high throughput and multiplexing capabilities of the proposed solution, albeit in an independent manner. Further work needs to be done to refine the technology and improve the reproducibility. These improvements can likely be found with the newer versions of the compartment on demand robot in combination with COMSOL modelling to ensure minimal backpressure issues. The technology could also be made more robust by adopting less prototypic assembly strategies (such as those used to join microtubing with syringes) as these represent a common failure point across all the system tests. By trading off on some of the small efficiencies gained by reducing droplet size it would be possible to make use of aluminium milling techniques and Luer-Lock type connections to ensure optimal connectivity between technology components.

The work above also created a data handling framework for characterisation data that enables consistent data storage whilst maintaining a flexible data manipulation mind set. The DPAD software offers a highly concurrent, robust solution to the data handling problem, one that is capable of highly scalable, flexible and robust data processing. The DPAD software also offers a variety of potential points of improvements that would enable it to become and an enterprise-level centralised data handling solution for many research institutions at the same time. Theoretically, this would greatly improve the rate at which a uniform characterisation database is realised and increase the rate of establishing standards and lower the ambiguity of much of the characterisation speciality. The work above has demonstrated the utility of the software by loading a common type of data through the plugin architecture and displaying it to the user in a meaningful manner. In future, the software can be improved to integrate flow cytometry data by designing custom plugins. This would tie in with some of the more recent advances in biological DNA part characterisation and provide a more comprehensive toolset (Hirst, 2014). Further testing should also be performed across larger network infrastructures to confirm concurrent data accessibility, network stability and usability across a more wide ranging user group.

The two solutions developed here, when joined together, can theoretically offer a comprehensive solution to the characterisation problems that were laid out in the beginning of this work. It is hoped that such complex infrastructural developments can be adopted and in-lined into everyday workflows in order to increase the throughput of biological part characterisation in Synthetic Biology.

10 Appendices

10.1 List of Promoter Sequences Generated

Table 10: Full list of promoters used in this study. Promoter naming convention is the registry name (either BioBrick^{TM} or BioFAB) suffixed with FW for the forward primer and RV for the reverse.

Name	Sequence
J23100FW	aattcgcggccgcttctagagttgacggctagctcagtcctaggtacagtgctagcta
J23100RV	ctagtagctagcactgtacctaggactgagctagccgtcaactctagaagcggccgcg
J23101FW	aattcgcggccgcttctagagtttacagctagctcagtcctaggtattatgctagcta
J23101RV	ctagtagctagcataatacctaggactgagctagctgtaaactctagaagcggccgcg
J23106FW	aattcgcggccgcttctagagtttacggctagctcagtcctaggtatagtgctagcta
J23106RV	ctagtagctagcactatacctaggactgagctagccgtaaactctagaagcggccgcg
J23114FW	aattcgcggccgcttctagagtttatggctagctcagtcctaggtacaatgctagcta
J23114RV	ctagtagctagcattgtacctaggactgagctagccataaactctagaagcggccgcg
J23113FW	aattcgcggccgcttctagagctgatggctagctcagtcctagggattatgctagcta
J23113RV	ctagtagctagcataatccctaggactgagctagccatcagctctagaagcggccgcg
J23116FW	aattcgcggccgcttctagagttgacagctagctcagtcctagggactatgctagcta
J23116RV	ctagtagctagcatagtccctaggactgagctagctgtcaactctagaagcggccgcg
J23104FW	aattcgcggccgcttctagagttgacagctagctcagtcctaggtattgtgctagcta
J23104RV	ctagtagctagcacaatacctaggactgagctagctgtcaactctagaagcggccgcg
J23105FW	aattcgcggccgcttctagagtttacggctagctcagtcctaggtactatgctagcta
J23105RV	ctagtagctagcatagtacctaggactgagctagccgtaaactctagaagcggccgcg
J23107FW	aattcgcggccgcttctagagtttacggctagctcagccctaggtattatgctagcta
J23107RV	ctagtagctagcataatacctagggctgagctagccgtaaactctagaagcggccgcg
J23108FW	aattcgcggccgcttctagagctgacagctagctcagtcctaggtataatgctagcta
J23108RV	ctagtagctagcattatacctaggactgagctagctgtcagctctagaagcggccgcg
J23109FW	aattcgcggccgcttctagagtttacagctagctcagtcctagggactgtgctagcta
J23109RV	ctagtagctagcacagtccctaggactgagctagctgtaaactctagaagcggccgcg
J23110FW	aattcgcggccgcttctagagtttacggctagctcagtcctaggtacaatgctagcta
J23110RV	ctagtagctagcattgtacctaggactgagctagccgtaaactctagaagcggccgcg
J23111FW	aattcgcggccgcttctagagttgacggctagctcagtcctaggtatagtgctagcta
J23111RV	ctagtagctagcactatacctaggactgagctagccgtcaactctagaagcggccgcg
J23112FW	aattcgcggccgcttctagagctgatagctagctcagtcctagggattatgctagcta
J23112RV	ctagtagctagcataatccctaggactgagctagctatcagctctagaagcggccgcg
J23115FW	aattcgcggccgcttctagagtttatagctagctcagcccttggtacaatgctagcta
J23115RV	ctagtagctagcattgtaccaagggctgagctagctataaactctagaagcggccgcg
J23117FW	aattcgcggccgcttctagagttgacagctagctcagtcctagggattgtgctagcta

J23117RV	ctagtagctagcacaatccctaggactgagctagctgtcaactctagaagcggccgcg
J23118FW	aattcgcggccgcttctagagttgacggctagctcagtcctaggtattgtgctagcta
J23118RV	ctagtagctagcacaatacctaggactgagctagccgtcaactctagaagcggccgcg
J23119FW	aattcgcggccgcttctagagttgacagctagctcagtcctaggtataatgctagcta
J23119RV	ctagtagctagcattatacctaggactgagctagctgtcaactctagaagcggccgcg
apFAB168FW	aattcgcggccgcttctagagttattcctaatcatccggctcgtataatgtgtggata
apFAB168RV	ctagtatccacacattatacgagccggatgattaggaataactctagaagcggccgcg
apFAB188FW	aattcgcggccgcttctagagttattccttaatcatcggctcgtataatgtgtggata
apFAB188RV	ctagtatccacacattatacgagccgatgattaaggaataactctagaagcggccgcg
apFAB189FW	aattcgcggccgcttctagagtttttccttaatcatcggctcgtataatgtgtggata
apFAB189RV	ctagtatccacacattatacgagccgatgattaaggaaaaactctagaagcggccgcg
apFAB190FW	aattcgcggccgcttctagagttctgcgttaatcatcggctcgtataatgtgtggata
apFAB190RV	ctagtatccacacattatacgagccgatgattaacgcagaactctagaagcggccgcg
apFAB197FW	aattcgcggccgcttctagagttgacaattaatcatcggctcttaggttttgtggata
apFAB197RV	ctagtatccacaaaacctaagagccgatgattaattgtcaactctagaagcggccgcg
apFAB200FW	aattcgcggccgcttctagagttgacaattaatcatcggctcttagggtttgtggata
apFAB200RV	ctagtatccacaaaccctaagagccgatgattaattgtcaactctagaagcggccgcg
apFAB210FW	aattcgcggccgcttctagagttcgccgttaatcatcggctcgtataatgtgtggata
apFAB210RV	ctagtatccacacattatacgagccgatgattaacggcgaactctagaagcggccgcg
apFAB212FW	aattcgcggccgcttctagagttcgtttttaatcatcggctcgtataatgtgtggata
apFAB212RV	ctagtatccacacattatacgagccgatgattaaaaacgaactctagaagcggccgcg
apFAB215FW	aattcgcggccgcttctagagtttcaggttaatcatcggctcgtataatgtgtggata
apFAB215RV	ctagtatccacacattatacgagccgatgattaacctgaaactctagaagcggccgcg
apFAB217FW	aattcgcggccgcttctagagttgacaattaatcatcggctcctagggtttgtggata
apFAB217RV	ctagtatccacaaaccctaggagccgatgattaattgtcaactctagaagcggccgcg
apFAB220FW	aattcgcggccgcttctagagttgacaattaatcatcggctcatatggtctgtggata
apFAB220RV	ctagtatccacagaccatatgagccgatgattaattgtcaactctagaagcggccgcg
apFAB235FW	aattcgcggccgcttctagagttgacaattaatctccggctcttacggtatgtggata
apFAB235RV	ctagtatccacataccgtaagagccggagattaattgtcaactctagaagcggccgcg
apFAB236FW	aattcgcggccgcttctagagttgacaattaatcatcggctcctatggtgtgtggata
apFAB236RV	ctagtatccacacaccataggagccgatgattaattgtcaactctagaagcggccgcg
apFAB237FW	aattcgcggccgcttctagagttgacaattaatcatcggctcataacctttgtggata
apFAB237RV	ctagtatccacaaaggttatgagccgatgattaattgtcaactctagaagcggccgcg
apFAB238FW	aattcgcggccgcttctagagttgacaattaatcatcggctcgtaggttttgtggata
apFAB238RV	ctagtatccacaaaacctacgagccgatgattaattgtcaactctagaagcggccgcg
apFAB240FW	aattcgcggccgcttctagagtttgaatttaatcatcggctcgtataatgtgtggata
apFAB240RV	ctagtatccacacattatacgagccgatgattaaattcaaactctagaagcggccgcg

apFAB241FW	aattcgcggccgcttctagagttaacattaatcatccggctcgtataatgtgtggata
apFAB241RV	ctagtatccacacattatacgagccggatgattaatgttaactctagaagcggccgcg
apFAB247FW	aattcgcggccgcttctagagttgacaattaatcatcggctcttaggttctgtggata
apFAB247RV	ctagtatccacagaacctaagagccgatgattaattgtcaactctagaagcggccgcg
apFAB249FW	aattcgcggccgcttctagagttgacaattaatcatcggctcgtagattttgtggata
apFAB249RV	ctagtatccacaaaatctacgagccgatgattaattgtcaactctagaagcggccgcg
apFAB250FW	aattcgcggccgcttctagagttgacaattaatcatcggctcctacgatttgtggata
apFAB250RV	ctagtatccacaaatcgtaggagccgatgattaattgtcaactctagaagcggccgcg
apFAB255FW	aattcgcggccgcttctagagttcacaattaatcatcggctcgtataatgtgtggata
apFAB255RV	ctagtatccacacattatacgagccgatgattaattgtgaactctagaagcggccgcg
apFAB260FW	aattcgcggccgcttctagagtttactgttaatcatcggctcgtataatgtgtggata
apFAB260RV	ctagtatccacacattatacgagccgatgattaacagtaaactctagaagcggccgcg
apFAB264FW	aattcgcggccgcttctagagtttatctttaatcatcggctcgtataatgtgtggata
apFAB264RV	ctagtatccacacattatacgagccgatgattaaagataaactctagaagcggccgcg
apFAB265FW	aattcgcggccgcttctagagttgactattaatcatcggctcatccattatgtggata
apFAB265RV	ctagtatccacataatggatgagccgatgattaatagtcaactctagaagcggccgcg
apFAB266FW	aattcgcggccgcttctagagttgacaattaatcatcggctcttaggctatgtggata
apFAB266RV	ctagtatccacatagcctaagagccgatgattaattgtcaactctagaagcggccgcg
apFAB269FW	aattcgcggccgcttctagagttgacaattaatcatcggctcgtaggctgtgtggata
apFAB269RV	ctagtatccacacagcctacgagccgatgattaattgtcaactctagaagcggccgcg
apFAB270FW	aattcgcggccgcttctagagttgacaattaatcatcggctcataaagtgtgtggata
apFAB270RV	ctagtatccacacactttatgagccgatgattaattgtcaactctagaagcggccgcg
apFAB272FW	aattcgcggccgcttctagagttgacaattaatcatcggctcgtagagtttgtggata
apFAB272RV	ctagtatccacaaactctacgagccgatgattaattgtcaactctagaagcggccgcg
apFAB276FW	aattcgcggccgcttctagagttgtctattaatcatcggctcgtataatgtgtggata
apFAB276RV	ctagtatccacacattatacgagccgatgattaatagacaactctagaagcggccgcg
apFAB279FW	aattcgcggccgcttctagagttgttctttaatcatcggctcgtataatgtgtggata
apFAB279RV	ctagtatccacacattatacgagccgatgattaaagaacaactctagaagcggccgcg
apFAB295FW	aattcgcggccgcttctagagttgcctcttaatcatcggctcgtataatgtgtggata
apFAB295RV	ctagtatccacacattatacgagccgatgattaagaggcaactctagaagcggccgcg
apFAB303FW	aattcgcggccgcttctagagttgaatcttaatcatcggctcgtataatgtgtggata
apFAB303RV	ctagtatccacacattatacgagccgatgattaagattcaactctagaagcggccgcg
apFAB324FW	aattcgcggccgcttctagagttgacaattaatcatcggctcttaggctttgtggata
apFAB324RV	ctagtatccacaaagcctaagagccgatgattaattgtcaactctagaagcggccgcg
apFAB325FW	aattcgcggccgcttctagagttgacaattaatatccggctcgtagcgtctgtggata
apFAB325RV	ctagtatccacagacgctacgagccggatattaattgtcaactctagaagcggccgcg

10.2 Promoter Sequence Generation Script

```
%% BuildPrimers
% Short script for generating primers from a cell array of strings (in
% column format corresponding to the promoter sequences).
% The input cell array can have other information attached to it that will
% be prepended to the result
%
% NOTE: This function requires the bioinformatics toolbox
%%
function [primerSequences] = BuildPrimers(primerSeq)
    % Check if we have an input
    if (iscell(primerSeq) && isempty(primerSeq))
        error('Input was not valid');
    end

    % Ensure it's not empty
    numSequences = size(primerSeq, 1);
    if numSequences < 1
        error('Input was not valid');
    end

    % Ensure the first column has strings
    if iscellstr(primerSeq(:, 1))
        error('Input does not contain strings in column 1');
    end

    % Prefill arrays
    primerSequences = cell(numSequences * 2, 1);
    sequenceCopy = cell(numSequences, 1);
    sequenceCopy(:, 1) = primerSeq(:, 1);
    additionalInformation = size(primerSeq, 2);

    % Affix Forward Primers
    forwardPrimers = strcat('AATTCGCGGCCGCTTCTAGAG', sequenceCopy(:, 1), 'TA');

    % Affix Reverse Primers
    reversePrimers = cellfun(@seqrcomplement, sequenceCopy(:, 1), 'uni', false);
    reversePrimers = cellfun(@fliplr, reversePrimers(:, 1), 'uni', false);
    reversePrimers = strcat('GCGCCGGCGAAGATCTC', reversePrimers(:, 1), 'ATGATC');

    % Set to uppercase as a standard
    forwardPrimers = cellfun(@upper, forwardPrimers, 'uni', false);
    reversePrimers = cellfun(@upper, reversePrimers, 'uni', false);

    for n = 1 : size(primerSequences, 1)
        index = (n * 2) - 1;

        % Reverse Sequence
        if mod(index, 2) == 1
            sequence = forwardPrimers(index, 1);
            direction = 'F';

        % Forward Sequence
        else
            sequence = reversePrimers(index, 1);
            direction = 'R';
        end

        primerSequences(index, 1) = sequence;
        primerSequences(index, 2) = direction;

        if additionalInformation > 1
            primerSequences(index, 3 : 2 + (additionalInformation - 1)) = ...
                primerSeq(n, 2 : additionalInformation);
        end
    end
end
```

Figure 60: Script for generating compatible primers for the generation of promoter sequences

10.3 APD Data Analysis – Data Selection Script

```
%% Function to identify regions of the data to remove
% Displays a gui to allow the user to identify regions of the data to trim
function [headFrom, headTo, tailFrom, tailTo] = IdentifyDataTrimZones(inputData)
    % Build GUI and get our communication variable
    guiHandle = BuildDataSelectionGUI();
    userData = get(guiHandle, 'UserData');

    % Build plot
    set(userData.axes, 'XLim', [0 inputData(end, 2)]);
    set(userData.axes, 'YLim', [0, max(inputData(:, 1))]);
    userData.plot = plot(inputData(:, 2), inputData(:, 1), 'g');
    userData.axisSize = axis(userData.axes);

    % Set max width and height
    userData.maxWidth = inputData(end, 2);
    userData.maxHeight = max(inputData(:, 1));
    set(guiHandle, 'UserData', userData);

    % Loop while the user is doing nothing in the GUI
    while true
        % Update userData
        userData = get(guiHandle, 'UserData');

        % If we are finished
        if userData.isFinished
            break;
        end

        % No selection mode has been chosen
        if userData.runtime == 0
            waitfor(guiHandle, 'UserData');
            continue;
        end

        % User has chosen a selection
        waitforbuttonpress;
        userData = get(guiHandle, 'UserData');
        userData.oldpointer = get(guiHandle, 'Pointer');
        set(guiHandle, 'UserData', userData);
        set(guiHandle, 'Pointer', 'ibeam');
        set(guiHandle, 'WindowButtonMotionFcn', {@MouseMove, userData});
        set(guiHandle, 'WindowButtonUpFcn', {@MouseUp, userData});
        uiwait;
    end

    % Save head end trimming data
    headFrom = 1;
    if isfield(userData, 'headSelection')
        headTo = userData.headSelection(2);
    else
        headTo = 1;
    end

    % Save tail end trimming data
    tailTo = length(inputData);
    if isfield(userData, 'tailSelection')
        tailFrom = userData.tailSelection(1);
    else
        tailFrom = length(inputData);
    end

    close(guiHandle);
end

% Core function for displaying the trimming GUI
function [guiHandle] = BuildDataSelectionGUI()
    % Core communication variable
    userData.isFinished = false;
    userData.runtime = 0;

    % Main figure
    guiHandle = figure(...
        'Units','characters',...
        'PaperUnits',get(0,'defaultfigurePaperUnits'),...
        'Name','selectLinearData',...
        'PaperPosition',get(0,'defaultfigurePaperPosition'),...
        'PaperSize',get(0,'defaultfigurePaperSize'),...
        'PaperType',get(0,'defaultfigurePaperType'),...
        'Position',[103.8 29.1538461538462 112 32.3076923076923],...
        'UserData', userData,...
        'Tag','figure1',...
        'Visible','on');

    userData.figure = guiHandle;

    % Main axes
    h2 = axes(...
        'Parent',guiHandle,...
        'Units','characters',...
        'Position',[-0.2 6.61538461538462 112 25.6153846153846],...
        'CameraPosition',[0.5 0.5 9.16025403784439],...
        'CameraPositionMode',get(0,'defaultaxesCameraPositionMode'),...
        'Tag','axes1');

    userData.axes = h2;
    set(guiHandle, 'UserData', userData);

    % Edit start data button
    h7 = uicontrol(...
        'Parent',guiHandle,...
        'Units','characters',...
        'Callback',@(hObject,eventdata)startDataSelection(userData),...
        'Position',[11.6 3.38461538461539 17 1.92307692307692],...
        'String',{  'Select Start Data' },...
        'Tag','pushbutton1');

    % Edit end data button
    h8 = uicontrol(...
        'Parent',guiHandle,...
        'Units','characters',...
        'Callback',@(hObject,eventdata)endDataSelection(userData),...
        'Position',[51.6000000000001 3.38461538461539 17 1.92307692307692],...
        'String',{  'Select End Data' },...
        'Tag','pushbutton3');

    % Finish button
    h9 = uicontrol(...
        'Parent',guiHandle,...
        'Units','characters',...
        'Callback',@(hObject,eventdata)finish(userData),...
        'Position',[92.8 0.692307692307693 17 1.92307692307692],...
        'String',{  'Finish' },...
        'Tag','pushbutton5');
end

% Function to handle when the user selects edit start data
function [] = startDataSelection(userData)
    % Build relevant rectangle properties for start data
    userData = get(userData.figure, 'UserData');

    % Build rectangle
    hold on
    xv = [0, 0, 0, 0, 0];
    yv = [0, userData.maxHeight, 0, userData.maxHeight, 0];
    rectangleHandle = fill(xv, yv, 'b');
    set(rectangleHandle, 'facealpha', 0.5, 'linestyle', '--', 'edgecolor', 'b');
    hold off

    % Save rectangle properties
    userData.currentRectangle = rectangleHandle;
    userData.startRectangle = rectangleHandle;
    userData.currentRectangleStart = 0;
    userData.runtime = 1;
    set(userData.figure, 'UserData', userData);
end

% Function to handle when the user selects edit end data
function [] = endDataSelection(userData)
```

```
    % Build relevant rectangle properties for end data
    userData = get(userData.figure, 'UserData');

    % Build rectangle
    hold on
    xv   =   [userData.maxWidth,   userData.maxWidth,   userData.maxWidth,   userData.maxWidth,
userData.maxWidth];
    yv = [0, userData.maxHeight, 0, userData.maxHeight, 0];
    rectangleHandle = fill(xv, yv, 'r');
    set(rectangleHandle, 'facealpha', 0.5, 'linestyle', '--', 'edgecolor', 'r');
    hold off

    % Save rectangle properties
    userData.currentRectangle = rectangleHandle;
    userData.endRectangle = rectangleHandle;
    userData.currentRectangleStart = userData.maxWidth;
    userData.runtime = 2;
    set(userData.figure, 'UserData', userData);
end

% Function to handle when the user presses finish.
function [] = finish(userData)
    % Inform the wait loop that we are done
    userData = get(userData.figure, 'UserData');
    userData.isFinished = true;
    set(userData.figure, 'UserData', userData);
end

% Function to handle whenever the mouse moves
function [] = MouseMove(src, event, userData) %#ok
    % Build and collect the relevant data
    userData = get(userData.figure, 'UserData');
    mousenew = get(userData.axes, 'CurrentPoint');
    xy = mousenew(1, 1:2);
    axis(userData.axisSize);

    % Build the rectangle
    xv = [userData.currentRectangleStart(1), ...
        xy(1), ...
        xy(1), ...
        userData.currentRectangleStart(1), ...
        userData.currentRectangleStart(1)];
    yv = [0, ...
        0, ...
        userData.maxHeight, ...
        userData.maxHeight, ...
        0];

    % Update the current rectangle
    set(userData.currentRectangle, 'xdata', xv, 'ydata', yv);
    set(userData.figure, 'UserData', userData);
end

function [] = MouseUp(src, event, userData) %#ok
    % Gather our communication variable
    userData = get(userData.figure, 'UserData');

    % Store the values
    mousenew = get(userData.axes, 'CurrentPoint');
    xy = mousenew(1, 1:2);

    % Update variables based on the selection type
    switch userData.runtime
        case 1
            userData.headSelection = [0, closestPoint(userData, xy(1))];

        case 2
            userData.tailSelection = [closestPoint(userData, xy(1)), userData.maxWidth];
    end

    % Revert pointer and current rectangle selection
    set(userData.figure, 'Pointer', userData.oldpointer);
    userData.runtime = 0;
    userData.currentRectangle = [];

    % Update variables
    set(userData.figure, 'UserData', userData);
    set(userData.figure, 'WindowButtonMotionFcn', '');
    set(userData.figure, 'WindowButtonUpFcn', '');
    uiresume;
end

% Function to determine the index along the x axis closest to the mouse
function [xIndex] = closestPoint(userData, xMouse)
    dx = (userData.axisSize(2) - userData.axisSize(1));
    ratio = dx / length(get(userData.plot, 'xData'));
    xIndex = floor(xMouse / floor(ratio));
end
```

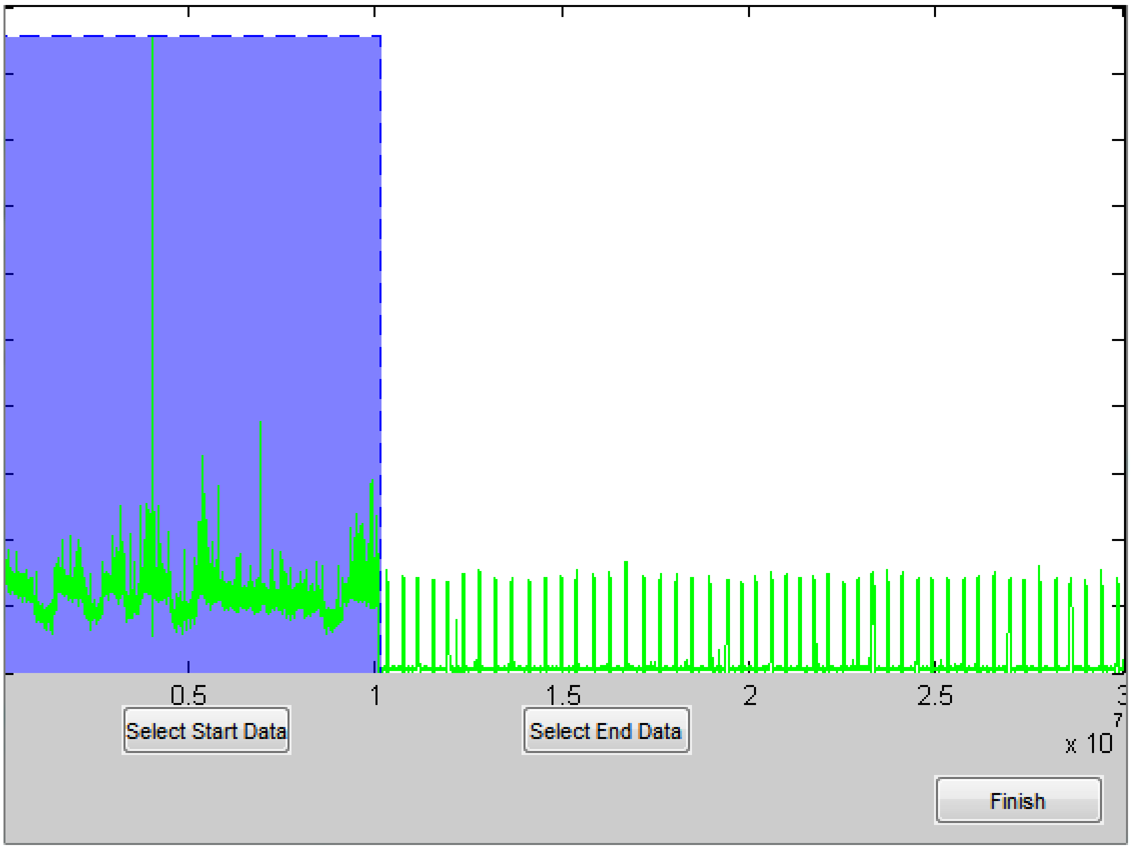


Figure 61: Example of how data is selected for trimming using the script above.

10.4 Sample Microplate Data for the DPAD Software

The sections below display the useful data present within the sample data set used to derive the affect of dilution on fluorescence production by J23101 on IVTT. A1 corresponds to pure GFP (250ng/ul), A2 corresponds to J23101 and IVTT where the premix was diluted by half, A3 corresponds to J23101 where the cell-free portion was diluted by half and A4 corresponds to water.

<u>0.txt</u>

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	3412	570	480	330	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

<u>50.txt</u>

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	3117	864	540	276	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

100.txt

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	2994	1483	567	261	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

150.txt

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	2761	1891	602	244	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

200.txt

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	2315	2103	549	241	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

250.txt

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	2028	2216	541	237	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

300.txt

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	1680	2243	482	233	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

350.txt

[Plate: M 485/528]

	1	2	3	4	5	6	7	8	9	10	11	12
A	1486	2201	468	230	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x	x	x	x	x
C	x	x	x	x	x	x	x	x	x	x	x	x
D	x	x	x	x	x	x	x	x	x	x	x	x
E	x	x	x	x	x	x	x	x	x	x	x	x
F	x	x	x	x	x	x	x	x	x	x	x	x
G	x	x	x	x	x	x	x	x	x	x	x	x
H	x	x	x	x	x	x	x	x	x	x	x	x

11 References

Registry of Standard Biological Parts [Online]. Available: http://partsregistry.org/wiki/index.php?title=Assembly:Standard_assembly.

2009. Synthetic biology: scope, applications and implications. *London: The Royal Academy of Engineering.*

AGRESTI, J. J., ANTIPOV, E., ABATE, A. R., AHN, K., ROWAT, A. C., BARET, J.-C., MARQUEZ, M., KLIBANOV, A. M., GRIFFITHS, A. D. & WEITZ, D. A. 2010. Ultrahigh-throughput screening in drop-based microfluidics for directed evolution. *Proceedings of the National Academy of Sciences*, 107.9, 4004-4009.

ALPER, H., FISCHER, C., NEVOIGT, E. & STEPHANOPOULOS, G. 2005. Tuning genetic control through promoter engineering. *Proceedings of the National Academy of Sciences of the United States of America,* 102**,** 12678-12683.

ANDERSON, J. C., VOIGT, C. A. & ARKIN, A. P. 2007. Environmental signal integration by a modular AND gate. *Mol Syst Biol,* 3.1, 133.

ANDRIANANTOANDRO, E., BASU, S., KARIG, D. K. & WEISS, R. 2006. Synthetic biology: new engineering rules for an emerging discipline. *Mol Syst Biol,* 2.1

ARKIN, A. 2008. Setting the standard in synthetic biology. *Nat Biotech,* 26**,** 771-774.

BAKER, C. A., DUONG, C. T., GRIMLEY, A. & ROPER, M. G. 2009. Recent advances in microfluidic detection systems. *Bioanalysis,* 1**,** 967-75.

BASU, S., MEHREJA, R., THIBERGE, S., CHEN, M.-T. & WEISS, R. 2004. Spatiotemporal control of gene expression with pulse-generating networks. *Proceedings of the National Academy of Sciences of the United States of America,* 101**,** 6355-6360.

BAYER, T. S. & SMOLKE, C. D. 2005. Programmable ligand-controlled riboregulators of eukaryotic gene expression. *Nat Biotech,* 23**,** 337-343.

BEISEL, C. L., BAYER, T. S., HOFF, K. G. & SMOLKE, C. D. 2008. Model-guided design of ligand-regulated RNAi for programmable control of gene expression. *Mol Syst Biol,* 4.1, 224.

BEISEL, C. L. & SMOLKE, C. D. 2009. Design Principles for Riboswitch Function. *PLoS Comput Biol,* 5**,** e1000363.

BENNETT, M. R. & HASTY, J. 2009. Microfluidic devices for measuring gene network dynamics in single cells. *Nat Rev Genet,* 10**,** 628-638.

BENSON, D. A., KARSCH-MIZRACHI, I., LIPMAN, D. J., OSTELL, J. & WHEELER, D. L. 2005. GenBank. *Nucleic Acids Research,* 33**,** D34-D38.

BIDGOOD, W. D., JR. & HORII, S. C. 1992. Introduction to the ACR-NEMA DICOM standard. *Radiographics,* 12**,** 345-55.

BLOUNT, B. A., WEENINK, T., VASYLECHKO, S. & ELLIS, T. 2012. Rational Diversification of a Promoter Providing Fine-Tuned Expression and Orthogonal Regulation for Synthetic Biology. *PLoS ONE,* 7**,** e33279.

CANTON, B., LABNO, A. & ENDY, D. 2008. Refinement and standardization of synthetic biological parts and devices. *Nat Biotech,* 26**,** 787-793.

CHAPPELL, J., JENSEN, K. & FREEMONT, P. S. 2013. Validation of an entirely in vitro approach for rapid prototyping of DNA regulatory elements for synthetic biology. *Nucleic Acids Res,* 41**,** 3471-81.

CONSORTIUM, T. U. 2013. Update on activities at the Universal Protein Resource (UniProt) in 2013. *Nucleic Acids Research,* 41**,** D43-D47.

DANINO, T., MONDRAGON-PALOMINO, O., TSIMRING, L. & HASTY, J. 2010. A synchronized quorum of genetic clocks. *Nature,* 463**,** 326-30.

DEMELLO, A. J. 2006. Control and detection of chemical reactions in microfluidic systems. *Nature,* 442**,** 394-402.

DITTRICH, P. S., JAHNZ, M. & SCHWILLE, P. 2005. A new embedded process for compartmentalized cell-free protein expression and on-line detection in microfluidic devices. *Chembiochem,* 6**,** 811-4.

DITTRICH, P. S. & MANZ, A. 2006. Lab-on-a-chip: microfluidics in drug discovery. *Nat Rev Drug Discov,* 5**,** 210-8.

DOKTYCZ, M. J. & SIMPSON, M. L. 2007. Nano-enabled synthetic biology. *Mol Syst Biol,* 3**,** 125.

DU, G., FANG, Q. & DEN TOONDER, J. M. J. 2016. Microfluidics for cell-based high throughput screening platforms—A review. *Analytica Chimica Acta,* 903**,** 36-50.

EASTBURN, D. J., HUANG, Y., PELLEGRINO, M., SCIAMBI, A., PTÁČEK, L. J. & ABATE, A. R. 2015. Microfluidic droplet enrichment for targeted sequencing. *Nucleic Acids Research*, 43.13 e86

ELLIS, T., WANG, X. & COLLINS, J. J. 2009. Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat Biotech,* 27**,** 465-471.

ELOWITZ, M. B. & LEIBLER, S. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature,* 403**,** 335-338.

ENDY, D. 2005. Foundations for engineering biology. *Nature,* 438**,** 449-453.

FORSTER, A. C. & CHURCH, G. M. 2006. Synthetic biology projects in vitro. *Genome Research,* 17**,** 000.

FRIEDLAND, A. E., LU, T. K., WANG, X., SHI, D., CHURCH, G. & COLLINS, J. J. 2009. Synthetic Gene Networks That Count. *Science,* 324**,** 1199-1202.

FUSSENEGGER, M. 2010. Synthetic biology: Synchronized bacterial clocks. *Nature,* 463**,** 301-302.

GALDZICKI, M., CLANCY, K. P., OBERORTNER, E., POCOCK, M., QUINN, J. Y., RODRIGUEZ, C. A., ROEHNER, N., WILSON, M. L., ADAM, L., ANDERSON, J. C., BARTLEY, B. A., BEAL, J., CHANDRAN, D., CHEN, J., DENSMORE, D., ENDY, D., GRUNBERG, R., HALLINAN, J., HILLSON, N. J., JOHNSON, J. D., KUCHINSKY, A., LUX, M., MISIRLI, G., PECCOUD, J., PLAHAR, H. A., SIRIN, E., STAN, G.-B., VILLALOBOS, A., WIPAT, A., GENNARI, J. H., MYERS, C. J. & SAURO, H. M. 2014. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat Biotech,* 32**,** 545-550.

GARAMELLA, J., MARSHALL, R., RUSTAD, M. & NOIREAUX, V. 2016. The All E. coli TX-TL Toolbox 2.0: A Platform for Cell-Free Synthetic Biology. *ACS Synthetic Biology,* 5**,** 344-355.

GARDNER, T. S., CANTOR, C. R. & COLLINS, J. J. 2000. Construction of a genetic toggle switch in Escherichia coli. *Nature,* 403**,** 339-342.

GEORGI, V., GEORGI, L., BLECHERT, M., BERGMEISTER, M., ZWANZIG, M., WUSTENHAGEN, D. A., BIER, F. F., JUNG, E. & KUBICK, S. 2016. On-chip automation of cell-free protein synthesis: new opportunities due to a novel reaction mode. *Lab Chip,* 16**,** 269-81.

GIBSON, D. G., GLASS, J. I., LARTIGUE, C., NOSKOV, V. N., CHUANG, R.-Y., ALGIRE, M. A., BENDERS, G. A., MONTAGUE, M. G., MA, L., MOODIE, M. M., MERRYMAN, C., VASHEE, S., KRISHNAKUMAR, R., ASSAD-GARCIA, N., ANDREWS-PFANNKOCH, C., DENISOVA, E. A., YOUNG, L., QI, Z.-Q., SEGALL-SHAPIRO, T. H., CALVEY, C. H., PARMAR, P. P., HUTCHISON, C. A., SMITH, H. O. & VENTER, J. C. 2010. Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome. *Science,* 329**,** 52-56.

GIELEN, F., VAN VLIET, L., KOPROWSKI, B. T., DEVENISH, S. R. A., FISCHLECHNER, M., EDEL, J. B., NIU, X., DEMELLO, A. J. & HOLLFELDER, F. 2013. A Fully Unsupervised Compartment-on-Demand Platform for Precise Nanoliter Assays of Time-Dependent Steady-State Enzyme Kinetics and Inhibition. *Analytical Chemistry,* 85**,** 4761-4769.

GRIFFITHS, A. D. & TAWFIK, D. S. 2006. Miniaturising the laboratory in emulsion droplets. *Trends in biotechnology,* 24**,** 395-402.

GULATI, S., ROUILLY, V., NIU, X., CHAPPELL, J., KITNEY, R. I., EDEL, J. B., FREEMONT, P. S. & DEMELLO, A. J. 2009. Opportunities for microfluidic technologies in synthetic biology. *Journal of The Royal Society Interface,* rsif20090083.

HE, M. 2008. Cell-free protein synthesis: applications in proteomics and biotechnology. *N Biotechnol,* 25**,** 126-32.

HEINEMANN, M. & PANKE, S. 2006. Synthetic biology—putting engineering into biology. *Bioinformatics,* 22**,** 2790-2799.

HILL, A. D., TOMSHINE, J. R., WEEDING, E. M. B., SOTIROPOULOS, V. & KAZNESSIS, Y. N. 2008. SynBioSS: the synthetic biology modeling suite. *Bioinformatics,* 24**,** 2551-2553.

HIRST, C. 2014. Automated BioPart characterisation for synthetic biology.

HOCKENBERRY, A. J. & JEWETT, M. C. 2012. Synthetic in vitro circuits. *Curr Opin Chem Biol,* 16**,** 253-9.

HOLTZE, C., ROWAT, A. C., AGRESTI, J. J., HUTCHISON, J. B., ANGILE, F. E., SCHMITZ, C. H. J., KOSTER, S., DUAN, H., HUMPHRY, K. J., SCANGA, R. A., JOHNSON, J. S., PISIGNANO, D. & WEITZ, D. A. 2008. Biocompatible surfactants for water-in-fluorocarbon emulsions. *Lab on a Chip,* 8**,** 1632-1639.

HONG, J., EDEL, J. B. & DEMELLO, A. J. 2009. Micro- and nanofluidic systems for high-throughput biological screening. *Drug Discov Today,* 14**,** 134-46.

HUANG, H.-H., CAMSUND, D., LINDBLAD, P. & HEIDORN, T. 2010. Design and characterization of molecular tools for a Synthetic Biology approach towards developing cyanobacterial biotechnology. *Nucleic Acids Research,* 38**,** 2577-2593.

HUEBNER, A., BRATTON, D., WHYTE, G., YANG, M., DEMELLO, A. J., ABELL, C. & HOLLFELDER, F. 2009. Static microdroplet arrays: a microfluidic device for droplet trapping, incubation and release for enzymatic and cell-based assays. *Lab on a Chip,* 9**,** 692-698.

HUEBNER, A., SHARMA, S., SRISA-ART, M., HOLLFELDER, F., EDEL, J. B. & DEMELLO, A. J. 2008. Microdroplets: A sea of applications? *Lab on a Chip,* 8**,** 1244-1254.

IIZUKA, R., YAMAGISHI-SHIRASAKI, M. & FUNATSU, T. 2011. Kinetic study of de novo chromophore maturation of fluorescent proteins. *Anal Biochem,* 414**,** 173-8.

JAKL, M. 2005. Representational State Transfer. Citeseer.

JEWETT, M. C., CALHOUN, K. A., VOLOSHIN, A., WUU, J. J. & SWARTZ, J. R. 2008. An integrated cell-free metabolic platform for protein production and synthetic biology. *Mol Syst Biol,* 4**,** 220.

JUNGMANN, R., RENNER, S. & SIMMEL, F. C. 2008. From DNA nanotechnology to synthetic biology. *HFSP J,* 2**,** 99-109.

KARA, C. & JAMES, S. 2009. Cell-Free Systems for Metabolic Engineering. *The Metabolic Pathway Engineering Handbook.* CRC Press.

KAWANO, Y., OTSUKA, C., SANZO, J., HIGGINS, C., NIREI, T., SCHILLING, T. & ISHIKAWA, T. 2015. Expanding imaging capabilities for microfluidics: applicability of darkfield internal reflection illumination (DIRI) to observations in microfluidics. *PLoS One,* 10**,** e0116925.

KATZEN, F., CHANG, G. & KUDLICKI, W. 2005. The past, present and future of cell-free protein synthesis. *Trends Biotechnol,* 23**,** 150-6.

KATZENBEISSER, S. 2001. *Recent advances in RSA cryptography,* Boston, MA, Kluwer Academic Publishers.

KELLY, J., RUBIN, A., DAVIS, J., AJO-FRANKLIN, C., CUMBERS, J., CZAR, M., DE MORA, K., GLIEBERMAN, A., MONIE, D. & ENDY, D. 2009. Measuring the activity of BioBrick promoters using an in vivo reference standard. *Journal of Biological Engineering,* 3**,** 4.

KENSY, F., ENGELBRECHT, C. & BUCHS, J. 2009. Scale-up from microtiter plate to laboratory fermenter: evaluation by online monitoring techniques of growth and protein expression in Escherichia coli and Hansenula polymorpha fermentations. *Microbial Cell Factories,* 8**,** 68.

KHALIL, A. S. & COLLINS, J. J. 2010. Synthetic biology: applications come of age. *Nat Rev Genet,* 11**,** 367-79.

KHNOUF, R., BEEBE, D. J. & FAN, Z. H. 2009. Cell-free protein expression in a microchannel array with passive pumping. *Lab Chip,* 9**,** 56-61.

KITNEY, R. I., FREEMONT, P. & ROUILLY, V. 2007. Engineering a molecular predation oscillator. *IET Synthetic Biology***,** 1-3, 68-70.

KOBAYASHI, H., KÆRN, M., ARAKI, M., CHUNG, K., GARDNER, T. S., CANTOR, C. R. & COLLINS, J. J. 2004. Programmable cells: Interfacing natural and engineered gene networks. *Proceedings of the National Academy of Sciences of the United States of America,* 101**,** 8414-8419.

KOLESNIKOV, N., HASTINGS, E., KEAYS, M., MELNICHUK, O., TANG, Y. A., WILLIAMS, E., DYLAG, M., KURBATOVA, N., BRANDIZI, M., BURDETT, T., MEGY, K., PILICHEVA, E., RUSTICI, G., TIKHONOV, A., PARKINSON, H., PETRYSZAK, R., SARKANS, U. & BRAZMA, A. 2015. ArrayExpress update--simplifying data submissions. *Nucleic acids research,* 43**,** D1113-6.

KONG, D. S., CARR, P. A., CHEN, L., ZHANG, S. & JACOBSON, J. M. 2007. Parallel gene synthesis in a microfluidic device. *Nucleic Acids Research,* 35**,** e61.

KÖTTER, P., WEIGAND, J. E., MEYER, B., ENTIAN, K.-D. & SUESS, B. 2009. A fast and efficient translational control system for conditional expression of yeast genes. *Nucleic Acids Research,* 37**,** e120.

LEE, T. S., KRUPA, R. A., ZHANG, F., HAJIMORAD, M., HOLTZ, W. J., PRASAD, N., LEE, S. K. & KEASLING, J. D. 2011. BglBrick vectors and datasheets: A synthetic biology platform for gene expression. *J Biol Eng,* 5**,** 12.

LINSHIZ, G., JENSEN, E., STAWSKI, N., BI, C., ELSBREE, N., JIAO, H., KIM, J., MATHIES, R., KEASLING, J. D. & HILLSON, N. J. 2016. End-to-end automated microfluidic platform for synthetic biology: from design to functional analysis. *J Biol Eng,* 10**,** 3.

MACDONALD, J. T., BARNES, C., KITNEY, R. I., FREEMONT, P. S. & STAN, G.-B. V. 2011. Computational design approaches and tools for synthetic biology. *Integrative Biology,* 3**,** 97-108.

MACEICZYK, R. M., LIGNOS, I. G. & DEMELLO, A. J. 2015. Online detection and automation methods in microfluidic nanomaterial synthesis. *Current Opinion in Chemical Engineering,* 8**,** 29-35.

MARGUET, P., BALAGADDE, F., TAN, C. & YOU, L. 2007. Biology by design: reduction and synthesis of cellular components and behaviour. *Journal of the Royal Society, Interface / the Royal Society,* 4**,** 607-23.

MCGRATH, S. C., SCHIELTZ, D. M., MCWILLIAMS, L. G., PIRKLE, J. L. & BARR, J. R. 2011. Detection and quantification of ricin in beverages using isotope dilution tandem mass spectrometry. *Anal Chem,* 83**,** 2897-905.

MEIN, G., PAL, S., DHONDU, G., ANAND, T. K., STOJANOVIC, A., AL-GHOSEIN, M. & OEUVRAY, P. M. 2002. Simple object access protocol. Google Patents.

MILDENBERGER, P., EICHELBERG, M. & MARTIN, E. 2002. Introduction to the DICOM standard. *European Radiology,* 12**,** 920-927.

MILLINGTON, D. S., SISTA, R., ECKHARDT, A., ROUSE, J., BALI, D., GOLDBERG, R., COTTEN, M., BUCKLEY, R. & PAMULA, V. 2010. Digital Microfluidics: A Future Technology in the Newborn Screening Laboratory? *Seminars in Perinatology,* 34**,** 163-169.

MINSKY, M. 1988. Memoir on inventing the confocal scanning microscope. *Scanning,* 10**,** 128-138.

MURZIN, A. G., BRENNER, S. E., HUBBARD, T. & CHOTHIA, C. 1995. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol,* 247**,** 536-540.

NEUZI, P., GISELBRECHT, S., LANGE, K., HUANG, T. J. & MANZ, A. 2012. Revisiting lab-on-a-chip technology for drug discovery. *Nat Rev Drug Discov,* 11**,** 620-32.

NEVIN, D. E. & PRATT, J. M. 1991. A coupled in vitro transcription-translation system for the exclusive synthesis of polypeptides expressed from the T7 promoter. *FEBS Letters,* 291**,** 259-263.

NIEDERHOLTMEYER, H., SUN, Z. Z., HORI, Y., YEUNG, E., VERPOORTE, A., MURRAY, R. M. & MAERKL, S. J. 2015. Rapid cell-free forward engineering of novel genetic ring oscillators. *Elife,* 4**,** e09771.

NIU, X., GIELEN, F., DEMELLO, A. J. & EDEL, J. B. 2009. Electro-coalescence of digitally controlled droplets. *Anal Chem,* 81**,** 7321-5.

NOIREAUX, V., BAR-ZIV, R. & LIBCHABER, A. 2003. Principles of cell-free genetic circuit assembly. *Proceedings of the National Academy of Sciences,* 100**,** 12672-12677.

NOIREAUX, V. & LIBCHABER, A. 2004. A vesicle bioreactor as a step toward an artificial cell assembly. *Proceedings of the National Academy of Sciences of the United States of America,* 101**,** 17669-17674.

PAEGEL, B. M., BLAZEJ, R. G. & MATHIES, R. A. 2003. Microfluidic devices for DNA sequencing: sample preparation and electrophoretic analysis. *Curr Opin Biotechnol,* 14**,** 42-50.

PASOTTI, L., POLITI, N., ZUCCA, S., CUSELLA DE ANGELIS, M. G. & MAGNI, P. 2012. Bottom-up engineering of biological systems through standard bricks: a modularity study on basic parts and devices. *PLoS ONE,* 7**,** e39407.

PRINDLE, A., SELIMKHANOV, J., LI, H., RAZINKOV, I., TSIMRING, L. S. & HASTY, J. 2014. Rapid and tunable post-translational coupling of genetic circuits. *Nature,* 508**,** 387-391.

RO, D.-K., PARADISE, E. M., OUELLET, M., FISHER, K. J., NEWMAN, K. L., NDUNGU, J. M., HO, K. A., EACHUS, R. A., HAM, T. S., KIRBY, J., CHANG, M. C. Y., WITHERS, S. T., SHIBA, Y., SARPONG, R. & KEASLING, J. D. 2006. Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature,* 440**,** 940-943.

ROMERO, P. A. & ABATE, A. R. 2012. Flow focusing geometry generates droplets through a plug and squeeze mechanism. *Lab on a Chip,* 12**,** 5130-5132.

ROEHNER, N., OBERORTNER, E., POCOCK, M., BEAL, J., CLANCY, K., MADSEN, C., MISIRLI, G., WIPAT, A., SAURO, H. & MYERS, C. J. 2014. Proposed Data Model for the Next Version of the Synthetic Biology Open Language. *ACS Synthetic Biology,* 4.1 57-71.

SAINZ DE MURIETA, I., BULTELLE, M. & KITNEY, R. I. 2016. Toward the First Data Acquisition Standard in Synthetic Biology. *ACS Synth Biol*, 5 (8), 817-826.

SCHAERLI, Y. & HOLLFELDER, F. 2009. The potential of microfluidic water-in-oil droplets in experimental biology. *Molecular BioSystems,* 5**,** 1392-1404.

SERRANO, L. 2007. Synthetic biology: promises and challenges. *Mol Syst Biol,* 3.1, 158.

SHAH, N. H. & TENENBAUM, J. D. 2012. The coming age of data-driven medicine: translational bioinformatics' next frontier. *J Am Med Inform Assoc,* 19**,** e2-4.

SHETTY, R., ENDY, D. & KNIGHT, T., JR. 2008. Engineering BioBrick vectors from BioBrick parts. *Journal of Biological Engineering,* 2**,** 1-12.

SHIN, J. & NOIREAUX, V. 2010. Efficient cell-free expression with the endogenous E. Coli RNA polymerase and sigma factor 70. *J Biol Eng,* 4**,** 8.

SHIN, J. & NOIREAUX, V. 2012. An E. coli cell-free expression toolbox: application to synthetic gene circuits and artificial cells. *ACS Synth Biol,* 1**,** 29-41.

SINGH, A. V., FERRI, M., TAMPLENIZZA, M., BORGHI, F., DIVITINI, G., DUCATI, C., LENARDI, C., PIAZZONI, C., MERLINI, M., PODESTA, A. & MILANI, P. 2012. Bottom-up engineering of the surface roughness of nanostructured cubic zirconia to control cell adhesion. *Nanotechnology,* 23**,** 475101.

SINHA, J., REYES, S. J. & GALLIVAN, J. P. 2010. Reprogramming bacteria to seek and destroy an herbicide. *Nat Chem Biol,* 6**,** 464-470.

SONG, H., CHEN, D. L. & ISMAGILOV, R. F. 2006. Reactions in droplets in microfluidic channels. *Angewandte Chemie (International ed. in English),* 45**,** 7336-56.

SRISA-ART, M., DEMELLO, A. J. & EDEL, J. B. 2007. High-Throughput DNA Droplet Assays Using Picoliter Reactor Volumes. *Analytical Chemistry,* 79**,** 6682-6689.

SRISA-ART, M., DEMELLO, A. J. & EDEL, J. B. 2009. High-throughput confinement and detection of single DNA molecules in aqueous microdroplets. *Chemical Communications***,** 6548-6550.

STRICKER, J., COOKSON, S., BENNETT, M. R., MATHER, W. H., TSIMRING, L. S. & HASTY, J. 2008. A fast, robust and tunable synthetic gene oscillator. *Nature,* 456**,** 516-519.

SUN, Z. Z., YEUNG, E., HAYES, C. A., NOIREAUX, V. & MURRAY, R. M. 2014. Linear DNA for rapid prototyping of synthetic biological circuits in an Escherichia coli based TX-TL cell-free system. *ACS Synth Biol,* 3**,** 387-97.

TAKAHASHI, M. K., HAYES, C. A., CHAPPELL, J., SUN, Z. Z., MURRAY, R. M., NOIREAUX, V. & LUCKS, J. B. 2015. Characterizing and prototyping genetic networks with cell-free transcription–translation reactions. *Methods,* 86**,** 60-72.

TAWFIK, D. S. & GRIFFITHS, A. D. 1998. Man-made cell-like compartments for molecular evolution. *Nat Biotech,* 16**,** 652-656.

THEBERGE, A. B., COURTOIS, F., SCHAERLI, Y., FISCHLECHNER, M., ABELL, C., HOLLFELDER, F. & HUCK, W. T. S. 2010. Microdroplets in Microfluidics: An Evolving Platform for Discoveries in Chemistry and Biology. *Angewandte Chemie International Edition,* 49**,** 5846-5868.

V. KNYAZKOV, K., V. KOVALCHUK, S., N. TCHUROV, T., V. MARYIN, S. & V. BOUKHANOVSKY, A. 2012. CLAVIRE: e-Science infrastructure for data-driven computing. *Journal of Computational Science,* 3**,** 504-510.

VILLAR, G., HERON, A. J. & BAYLEY, H. 2011. Formation of droplet networks that function in aqueous environments. *Nat Nano,* 6**,** 803-808.

VISKARI, P. J. & LANDERS, J. P. 2006. Unconventional detection methods for microfluidic devices. *Electrophoresis,* 27**,** 1797-810.

WANG, S., ZHANG, X., WANG, W. & LEE, L. J. 2009. Semicontinuous flow electroporation chip for high-throughput transfection on mammalian cells. *Analytical Chemistry,* 81**,** 4414-21.

WILLIAMS, R., PEISAJOVICH, S. G., MILLER, O. J., MAGDASSI, S., TAWFIK, D. S. & GRIFFITHS, A. D. 2006. Amplification of complex gene libraries by emulsion PCR. *Nat Methods,* 3**,** 545-50.

WIN, M. N. & SMOLKE, C. D. 2007. A modular and extensible RNA-based gene-regulatory platform for engineering cellular function. *Proceedings of the National Academy of Sciences,* 104**,** 14283-14288.

YANG, Y.-H., KIM, T.-W., PARK, S.-H., LEE, K., PARK, H.-Y., SONG, E., JOO, H.-S., KIM, Y.-G., HAHN, J.-S. & KIM, B.-G. 2009. Cell-Free Escherichia coli-Based System To Screen for Quorum-Sensing Molecules Interacting with Quorum Receptor Proteins of Streptomyces coelicolor. *Applied and Environmental Microbiology,* 75**,** 6367-6372.

YANG, J., SELVAGANAPATHY, P. R., GOULD, T. J., DWIVEDI, D. J., LIU, D., FOX-ROBICHAUD, A. E. & LIAW, P. C. 2015. A microfluidic device for rapid quantification of cell-free DNA in patients with severe sepsis. *Lab Chip,* 15**,** 3925-33.

ZHANG, A. L., LIU, H., YANG, M. M., GONG, Y. S. & CHEN, H. 2007. Assay and characterization of a strong promoter element from B. subtilis. *Biochemical and biophysical research communications,* 354**,** 90-5.

ZHANG, C., XU, J., MA, W. & ZHENG, W. 2006. PCR microfluidic devices for DNA amplification. *Biotechnology Advances,* 24**,** 243-284.

ZHOU, X., CAI, S., HONG, A., YOU, Q., YU, P., SHENG, N., SRIVANNAVIT, O., MURANJAN, S., ROUILLARD, J. M., XIA, Y., ZHANG, X., XIANG, Q., GANESH, R., ZHU, Q., MATEJKO, A., GULARI, E. & GAO, X. 2004. Microfluidic PicoArray synthesis of oligodeoxynucleotides and simultaneous assembling of multiple DNA sequences. *Nucleic Acids Res,* 32**,** 5409-17.

ZUBAY, G. 1973. In vitro synthesis of protein in microbial systems. *Annu Rev Genet,* 7**,** 267-87.

ZUBAY, G. 1980. The Isolation and Properties of CAP, the Catabolite Gene Activator. *Methods in Enzymology,* 75**,** 856-877.