



# **A Web Application and Reader Point of Care CMOS Diagnostic Development Tool to Enhance the Efficacy of Blood Analysis**



**University College London**

**Donal McLaughlin<sup>1</sup>**

**MSc Computer Science**

**Dr Graham Roberts & Professor Mark Handley**

**Submission Date – 05 September 2018**

**2018 MSc Individual Project**

<sup>1</sup>Disclaimer: This report is submitted as part requirement for the MSc Computer Science at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

## **Acknowledgements**

My heartfelt gratitude to Dr Graham Roberts & Professor Mark Handley for their continuous support throughout the project. I would also express sincere thanks to all the team at NIBEC. A special mention to Dr David Steele and Dr Shasridran Raj for their guidance.

## Abstract

As a result of ever increasing pressures on the NHS, it is the fundamental objective of this project to develop and manufacture a Point-Of-Care device (Rapsens) whereby blood analysis via lateral flow tests are provided as a screening tool for laboratory testing. This supersedes the technology currently used for this process dovetailed to being highly cost effective whilst also being significantly more efficacious.

Lateral flow tests have dominated rapid diagnostics for over 30 years however, quantifying such a test, is in its early research and development stages. This highlights the significance and timeliness of this project in developing a low-cost, accurate, lightweight, and disposable digital reader technology. The Rapsens reader uses the novel approach of a CMOS-sensor camera to capture data from the Lateral Flow Strip, sending the image (using a Particle Photon microcontroller) via the TCP protocol to an Node.js TCP server within an Amazon Web Services' EC2 instance. Additionally, within the EC2 instance, a web server is used to host a web application to display, automatically analyse via a ROI and query results presented by the LFI Reader using HTML, CSS and JavaScript (and their corresponding libraries). Image analysis was primarily implemented using Python and the OpenCV vision libraries.

Rapsens enables higher quality diagnostics, better decision making and improved algorithm development due to related calibration curves demonstrating higher accuracy readings. These readings are repeatable with less error (0.5% in some cases) and artefacts are statistically controlled due to the wide angle of imaging (across blotchy changing elements) and averaging techniques as used during image processing. Rapsens' enhancements versus techniques using flat-bed scanners and image processing software (ImageJ) are illustrated with graphs that demonstrate reduced error bars and higher coefficient of determination (i.e. improved  $R^2$  fitting). The fundamental characteristics of this device and system facilitates a propensity for extrapolation.

## Contents

<b>1.0</b>	<b>INTRODUCTION</b>	7
1.1	Problem Domain	7
1.2	Project Aims and Goals	8
1.3	Personal Aims	8
1.4	Project Management	9
1.5	Report Overview	10
<b>2.0</b>	<b>RESEARCH BACKGROUND</b>	11
2.1	Introduction	11
2.1.1	Lateral Flow Immunoassay	11
2.1.2	CMOS VS CCD	12
2.2	Literature Review of Competitive Technologies	13
2.3	Project Technology research	14
2.3.1	Hardware and Firmware Research	14
2.3.2	Server-side Development	15
2.3.3	Client-side Development	17
<b>3.0</b>	<b>REQUIREMENTS</b>	18
3.1	Problem Statement	18
3.2	Requirements Gathering	19
3.2.1	Persona	19
3.2.2	Scenario	19
3.2.3	Storyboard with sketches	20
3.3	Requirements	21
3.4	Use Cases	24
<b>4.0</b>	<b>DESIGN AND IMPLEMENTATION</b>	25
4.1	Design, Housing and Assembly Architecture Implementation	25
4.2	Hardware Implementation	27
4.3	Firmware Implementation	28
4.4	Cloud Implementation	29
4.5	Image Receiver Script Implementation	31
4.6	Web Application Implementation	32
4.6.1	Login and Registration	32
4.6.2	Image Gallery, Manual and Automatic Image Analysis	32
4.6.3	Record Viewer and CSV downloader	34

4.6.4	View Automatically Detected Regions Of Interest.....	34
4.7	Python Manual Analysis Implementation .....	35
4.8	Python Auto Analysis Implementation.....	36
<b>5.0</b>	<b>EVALUATION</b> .....	38
5.1	Analysis.....	38
5.2	Browser Compatibility and Web testing Tool .....	42
5.3	Automated system testing for the web application.....	42
5.4	Alpha and Beta Acceptance testing .....	42
<b>6.0</b>	<b>CONCLUSIONS</b> .....	43
6.1	Project Goals and Requirements Evaluation.....	43
6.1.2	Cloud Architecture and Web Application Evaluation.....	43
6.2	Personal Aims Evaluation .....	44
6.4	Future Work .....	44
	<b>References</b> .....	46
	Appendix A .....	48
	System Manual.....	48
	Photon Firmware.....	48
	Virtual Machine .....	48
	Relational Database .....	49
	Appendix B .....	50
	User Manual .....	50
	Taking a reading using the Rapsens reader.....	50
	Analysing image data sent from the reader.....	50
	Appendix C .....	52
	Supporting documentation .....	52
	Use cases .....	52
	Screen Shots of the Web Application.....	56
	Selerium IDE test. ....	58
	Appendix D .....	60
	Code Listings.....	60

## List of Figures

Figure 2.1 A sandwich lateral flow diagnostic platform.....	12
Figure 2.2 ESEQuant Lateral Flow and the Detekt Chameleon Mini.....	13
Figure 2.3 (a) Particle Photon (b) Arducam.....	14
Figure 2.4 Lab colour space visualisation.....	17
Figure 3.1 - The LED-Photodiode Sensor Experimental Set-up.....	18
Figure 3.2 - The Rapsens Storyboard.....	20
Figure 3.3 – Functional and Non-Functional Requirements.....	21
Figure 3.4 – Project Use Cases.....	24
Figure 4.1 – Early Sketches of Rapsens; 3D CAD drawings; 3D Printer and assembly of Rapsens system.....	26
Figure 4.2 – Breadboard layout for Rapsens Prototype; Electrical schematic drawings; Breadboard and assembly of Rapsens system.....	28
Figure 4.3 – Overall schematic and systems diagram of the project’s key components.....	31
Figure 4.4 – Code Extract from index.php.....	34
Figure 4.5 – Automatically selected ‘Regions of Interest’ for the Rapsens Image Lateral Flow strips.....	35
Figure 4.6 – Code Extract from imageAnalysis.py.....	35
Figure 4.7 – Code Extract from autoThresholding.py.....	37
Figure 5.1 – Batch B: Pregnancy test strip hCG dose Response.....	39
Figure 5.2 – Batch C: Pregnancy test strip hCG dose Response.....	39
Figure 5.3 – Batch D: Pregnancy test strip hCG dose Response.....	40
Figure 5.4 – Flat Bed Scanner and ImageJ method showing variation between batches.....	40
Figure 5.5 – Automatic detection of ROI for Rapsens System showing variation between batches.....	41
Figure 5.6 – Manual detection of ROI for Rapsens System showing variation between batches.....	41
Figure 5.7 – SortSite Performance testing results .....	42

## Nomenclature

LFIA	Lateral Flow Immunoassays
LFD	Lateral Flow Device
PoCT	Point of Care Test(ing)
RDT	Rapid Diagnostic Test
CCD	Charge Coupled Device
CMOS	Complementary metal-oxide-semiconductor
AWS	Amazon Web Services
LED	Light Emitting Devices
IDE	Integrated Development Environment
NIBEC	Nanotechnology and Integrated Bioengineering Centre
hCG	Human Chorionic Gonadotropin
ROI	Regions of Interest
CAD	Computer Aided Design
IOT	Internet of Things
ECC	Elastic Cloud Compatibility
RDS	Relational Database System
VM	Virtual Machine
TCP	Transmission Control Protocol

## 1.0 INTRODUCTION

### 1.1 Problem Domain

Cognisant of increasing pressures experienced by the National Health Service, whereby time restraints and financial constraints necessitate the development of accurate Point of Care technologies with the capability to deliver instantaneous results is becoming a pre-requisite to the NHS. Accessing results from a blood test within the NHS can range from hours to weeks [1] which may result in the postponement of treatment for some patients, leading to a serious decline in their health. There exists a gap in the market for an intermediate Point of Care device whereby blood analysis via lateral flow tests are provided as a screening tool for laboratory testing. The current, predominant method of reading Lateral Flow strips is viewed and calculated by the naked eye and therefore results can be subjective, as minuscule variations within the colour change, due to analyte changes, can result in a misdiagnosis. Therefore, improved lateral flow developer systems are required to enhance calibration, algorithms, predictivity and overall accuracy of decision making.

The purpose of this project is to develop and manufacture an Internet of Things Point of Care digital medical diagnostic device which can semi-quantitatively analyse Lateral Flow Immunoassays using innovative CMOS (Complementary Metal-Oxide-Semiconductor) detector technologies. The Lateral Flow Immunoassay Reader will capture image data from the CMOS detector and transmit the data to a server within Amazon Web Services (AWS). A Lateral Flow analytical environment (in the form of a web-application is hosted on Amazon Web Services using an Apache Web Server) was designed and developed within this project, allowing users to view Lateral Flow tests, manually analyse each strip or allow the software to automatically analyse the test thus paving a way for higher quality diagnostics and lower False Positives/negatives.

This project was carried out in association with Ulster University's, Nanotechnology and Integrated Bioengineering Centre (NIBEC), a network of research groups which specialise in medical devices, sensors and advanced materials used within sensors, electronics, photonics, and tissue engineering. NIBEC provided invaluable expertise in the field of lateral flow analysis as well as the hardware (Particle Photon, Electronic components and Arducam camera modules) and licensed software (MATLAB and Solid Edge CAD) essential to fulfil the requirements within this project.



## 1.2 Project Aims and Goals

The predominant aim of this project is to design and develop a prototype based Lateral Flow Immunoassay CMOS-sensor based Reader which can send image-data to the cloud, thus allowing a 'lateral flow developer' to analyse the test-strip instantaneously using a web application which is in contrast to optical transmission/reflectance manual readers or even the naked eye approach, which are inherently fraught with large errors. Therefore, the primary goals of this project are:

- to design a robust and easy-to-use Lateral Flow Immunoassay (LFI) CMOS-sensor based Reader system (including 3-D printed plastics and LED illuminators), image acquisition via an Arducam image sensor and the transmittance of data to the cloud for analysis (written in python), calibration (via manual or automation detection of test lines) and visualisation via a specifically designed web-site (HTML, CSS and JavaScript).
- to create a clear and intuitive user-interface via a web application which is used to display, automatically analyse and query results presented by the LFI Reader.
- to allow the user to automatically detect regions of interest via the CMOS imaging data as collected via the new prototype system (which allows higher sensitivity and better signal integration of full images compared to transmissive/reflective modes in LFD's),
- to determine the concentration of, for example, hCG in a sample or manually select regions of interest within the strips in order to check the set-up for hCG calibration curves and thus higher accuracy devices demonstrated in regard to point of care devices

## 1.3 Personal Aims

The majority of the technologies used in this project were new to the author therefore time was set aside at the beginning of the project to learn and to become familiar with each technology. The following list highlights an overview of personal aims which were proposed and subsequently attained during the production of this project:

- Gain experience designing and developing a complete and fully functional IOT device
- Learn how to develop Arduino firmware (Wiring C/C++) using the Particle Framework (Particle Cloud)
- Learn to use Solid Edge Computer-Aided-Design to develop a robust physical prototype using Interaction Design methodologies and thus print 3-d plastics via additive technologies.

- Gain further experience using the client-side web technologies: HTML, CSS and JavaScript as well as obtain new experiences with their libraries such as jQuery and Ajax.
- Server script hosting using the JavaScript run-time environment Node.js
- Learn and use Python for image analysis using libraries such as OpenCV and Numpy
- Experience using Amazon Web Services EC2, EBS and RDS instances and server-side programming languages such as PHP to query the MySQL database using SQL
- Use of the Ubuntu Virtual Machine to host an Apache HTTP Server as well as the Image Analysis logic

#### 1.4 Project Management

The project was implemented using the four stages of the Unified Process (UP) methodologies defined by Arlow and Neustadt [2]: Inception, Elaboration, Construction and Transition, where each stage is highly iterative. The Inception phase of the project occurred throughout all the early stages of the project and consisted of gathering and analysing requirements using Interaction Design techniques described by Preece [3]. Once the requirements were established, an Elaboration Phase was carried out to design and establish the system's architecture. This was completed by creating use-cases, conceptual diagrams and architectural diagrams. The largest and most time-consuming part of the project was the Construction Phase which required the learning and applying of the technologies to create a final prototype. This phase consisted of debugging problems and iterating earlier phases to ensure the project was delivered on time. The final phase of the project was the Transition Phase which consisted of user testing and unit testing which led to further refinements within the Constructions Phases.

Although only one person was working on this project, a Scrum system (used in the agile framework) [4] was adapted allowing the author to work in *sprints* (timeboxed iterations) which typically lasted one week. After each sprint, the progress was self-reviewed which helped with time management and progression throughout the project allowing the completion of key milestones.

## 1.5 Report Overview

### **Chapter 2.0 Research**

This chapter consists of Background Information which will convey a situation analysis with problem identification; a Literature Review which contains a comprehensive literature search of medical/biomedical devices already in the market; and a Technological Review which presents the research and comparative study to establish most effective technologies customisable for the project.

### **Chapter 3.0 Requirements and Analysis**

A detailed problem statement is established, in conjunction with a structured list of Function and Non-Functional Requirements using the MoSCoW prioritisation method. Comprehensive primary research (20 depth interviews) was undertaken collating valuable information in the form of persona, scenario and storyboard. Use Cases were then created.

### **Chapter 4.0 Design and Implementation**

A detailed description of the design and implementation process is presented. This includes the whole planning, design and manufacture of all aspects of the prototype.

### **Chapter 5.0 Evaluation**

A detailed evaluation process is conveyed, outlining the criteria necessary to produce the most efficacious prototype in correlation to the required specification.

### **Chapter 6.0 Conclusions**

This chapter presents the drawing of conclusions and verification of the results. Recommendations for future amendments and evolution of the prototype are proffered.

## 2.0 RESEARCH BACKGROUND

### 2.1 Introduction

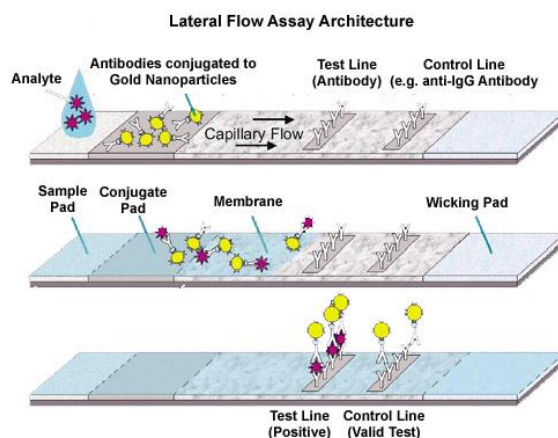
This section is split into three subsections: Background Information, Literature Review and Technologies Research; Background Information will give an insight into how Lateral Flow Immunoassays operate and how they are used from day-to-day. The Literature Review subsection will evaluate publications related to the topic of quantitative Lateral Flow Immunoassay analysis and how these pieces of work relate to this project. Lastly, the Technologies Research subsection will elaborate on the chosen technologies used within this project with comparisons between open-source alternative technologies. Technologies consist of: programming languages and their libraries, hardware, firmware, software, database engines, cloud services and Computer Aided Design software (CAD).

#### 2.1.1 Lateral Flow Immunoassay

Lateral Flow Immunoassay is a medical diagnostic method used to confirm the absence or presence of a target analyte (such as the Human Chorionic Gonadotropin, hCG hormone which is raised during pregnancy and used as gold standard in lateral flow development) by interpreting a colour change of the test line. [5] Most tests using this method are purely done on a qualitative basis where the user detects a colour change using the naked eye. This works well for some tests such as the pregnancy test (which tests for the hCG hormone) where a visible line appears when the hCG hormone is present (i.e. positive pregnancy result). However, this project explores a semi-quantitative approach where the end-result will display the concentration of the hCG hormone in the Developer's blood in thousandth international units per millilitre (mIU/ml).

Figure 2.1 gives an illustrated understanding on how a sandwich Lateral Flow Strip produces a result. The target analyte (usually based in blood or urine) is placed onto the sample pad of the Lateral Flow Assay. The analyte will then migrate along the strip to a region known as the 'conjugate pad'. If the target is present, the immobilised conjugated antibodies and gold colloid labels will bind to the target and then continue along the strip via capillary flow. The test line region is printed-on top of a nitrocellulose membrane which consists of binding reagents which will attach onto the target. A colour line should appear with an intensity proportional to the concentration of the target. This project will use gold colloid (however latex or magnetic nanoparticles could have been utilised [6]) as it results in a distinctive red colour plus gold is stable under exposure to heat and light, and any degradation is caused by the instability of the protein.[7]

As published by the critical review paper ‘Lab on Chip’ [5], lateral flow tests have dominated rapid diagnostics for over 30 years however, quantifying such a test, is in its early research and development stages. This highlights the significance and timeliness of this project in developing a low-cost, accurate, lightweight, and disposable digital reader technology.



**Figure 2.1 A sandwich lateral flow diagnostic platform**

### 2.1.2 CMOS VS CCD

Both Charge Coupled Devices- CCDs and CMOS imagers offer excellent imaging-performance in today’s market. CCDs have in the past provided the specification benchmarks in the scientific, photographic, and industrial applications that demand the highest image quality (as measured in quantum-efficiency and noise) without any real consideration of size/cost. CMOS imagers offer more integration (more functions on the chip), lower power dissipation (at the chip level), lower cost and the possibility of smaller system size, but they have often required trade-offs between image quality and device cost although this is dramatically improving due to high demand. CMOS designers have devoted great efforts to achieving high image quality, while CCD designers have focused on reducing power requirements and pixel sizes. Thus, you can find CCDs in low-cost low-power cell phone cameras and CMOS sensors in high-performance professional and industrial cameras, directly contradicting the early stereotypes. This project requires an interrogatable solution, highly suitable for digital processing and at potentially low cost.

## 2.2 Literature Review of Competitive Technologies

This subsection will list devices already on the market as well as published papers which relate to quantitate analysis of lateral flow immunoassay.

**Qiagen's ESEQuant Lateral Flow Reader**[8] as shown in Figure 2.2, is used for Research and Development when developing custom fluorescence or colorimetric tests. Within its specifications, measurement time is approximately 40 seconds and results can be displayed on PC software as well as a mobile phone app. Additionally, the reader has a built-in RFID reader which can detect what type of strip has been inserted. The ESEQuant reader does not use a CMOS sensor in contrast to this project and additionally, the ESEQuant reader instructs 'users' into using lateral flow holders (plastic registration cases) for precise measurements. Whilst taking images with CMOS sensors, the Field-Of-View is large (60°) allowing the software to auto-detect the regions of interest.

**Detekt Biomedical's Chameleon Mini** [9] has similar features to the ESEQuant, however it is more tailored for the public rather than lateral flow developers. The device is still in development stage but can be purchased in kits starting at \$1095. Images and data (collected via a simple Si sensor array as shown) is sent to an iOS device and processed with the mobile application and optionally can be stored within the cloud. The Chameleon Mini's architecture differs from what is required for this project, in that the Rapsens case, the image is sent directly to the cloud where it can then be analysed using a web application (instead of a mobile application).



**Figure 2.2 A Qiagen's ESEQuant Lateral Flow Reader and the Detekt Biomedical's Chameleon Mini**

With reference to the paper, '**Smartphone accessory for quantitative chemiluminescence-based lateral flow immunoassay**', [10] Zangheri *et al.* built a 3-D printed rig which could hold a lateral flow strip into position and was mounted onto Samsung Galaxy SII Plus smartphone. The smartphone has a Charge-Coupled-Device (CCD) camera (more complex to address than CMOS cameras) and was sensitive enough to detect and quantify cortisol in real saliva samples. Using

this method has a major drawback, as the 3-D printed rig must be redesigned for each smartphone (additionally, most smartphone models will take images using different sensors) and CCD cameras are now a less favourable technology.

## 2.3 Project Technology research

The research into the technologies used in this project can be split into three sections: Hardware and Firmware, server-side development and client-side development.

### 2.3.1 Hardware and Firmware Research

The main challenging hardware objective of this project was to build an IOT device which sends data to a remote server within Amazon Web Services. During the design phase of the project, research was focused on which device should be used to capture and transmit images of the lateral flow strip to the cloud. Two options were considered: Arduino MKR1000 and the Photon Particle microcontroller platforms integrated with Arducan CMOS Camera. Both microcontroller devices use the *Wiring* Integrated Development Platform (IDE) which includes the C/C++ library Wiring which aides with input/output firmware production. A learning phase of the project was set aside for the author to build up their skills with firmware and electronics development.

***Rationale for using a Particle Photon*** - A basic Arduino Uno requires a Wi-Fi shield to be attached which increases the price as well as the size of the device. Arduino produce a board called the Arduino MKR1000 which does not require an Arduino Uno, however the MKR1000 only has 256KB of



**Figure 2.3 (a) Particle Photon (b) Arducam**

flash memory and 32KB of RAM which is significantly lower than the Photon Particle which has 1MB of flash memory and 128KB of RAM. More memory is beneficial for this project as the reader is tasked to capture and transmit a lot of data. The Particle Photon has a built in Wi-Fi chip and is much less expensive than any other alternatives from Arduino or Raspberry Pi.

***Particle to Server challenges*** - Initial experiments were conducted on the Photon Particle which consisted of sending temperature data to Particle's "*Device Cloud*". Using Particle webhooks, this data can be sent to Lambda Functions (written in Node.js) in Amazon Web Services via an API Gateway where the data can be stored in a MySQL Relational Database (Amazon RDS). Although using Particle's "*Device Cloud*" was intuitive, it was restrictive as it limits the number of devices which can connect to the Device Cloud as well as the number of actions which can be sent to the "*Device Cloud*". Through further experimentation, the Photon was eventually declared successful in writing data to a Node.js server running separately to Particle's "*Device Cloud*".

**The Arducan CMOS Camera** - For the camera module, Arducam [15] is the primary manufacturer for cameras designed to be used with Arduino platforms which distribute open source hardware and software systems. The model used is the OV5640 5MP MINI PLUS which supports JPEG compression thus minimising the quantity of data sent to the cloud. Arducam camera's also uses CMOS sensors which are more power efficient than CCD sensors thus increasing battery life plus can be addressed effectively via microprocessors.

### 2.3.2 Server-side Development

**Web applications and database systems** - Within the modules COMPGC02 App Design and Database and Information Management Systems COMPGC06, the author gained experience developing and deploying web applications and database systems on Microsoft Azure. Azure fulfilled the requirements for this project however the author opted to use Amazon Web Services to explore its vast choice of services.

**AWS deployment** - In the early stages of the project, Amazon S3 buckets were considered for holding captured pictures however particularly in development, S3 buckets can run up large costs with large files being saved into them. It was then decided to use a Virtual Machine, allowing the files to be saved within the virtual machine. The client requested a Virtual Machine to be used to ease the transfer of files and data from AWS to their secure server.

To host a Virtual Machine, AWS provides an Amazon Elastic Compute Cloud (EC2) web service which offers resizable cloud computing allowing the developer to automatically scale up or down computing capacity depending on usage. This is vital as the developer only pays for services which are in use, thus keeping costs to a minimum. Ubuntu was chosen for the VM operating system as it can be installed onto a local computer which can then be easily deployed onto AWS.

Connecting and managing the Virtual Machine can primarily be done using Windows Command Prompt or Putty whilst using the Secure Shell (SSH) cryptographic network protocol. WinSCP is an open source SFTP client and SCP client for Windows allowing quick and easy file transfer between a local and a cloud-based machine using a user-interface similar to Window's File Explorer.

**Cloud Communication** - To send images from the Photon to the cloud, two communication protocols were considered: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP is the most commonly used protocol used on the internet and it has an advantageous edge over UDP. In the context of this project, TCP is the preferred protocol, as it guarantees the ordered, reliable, and error-checked delivery of a stream of bytes in the correct order which is



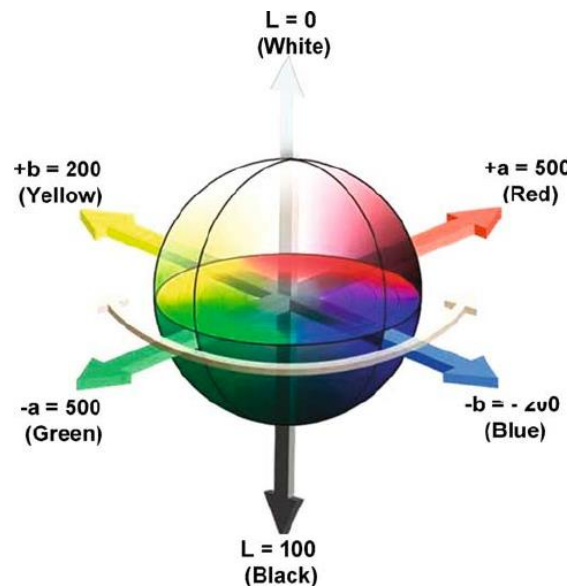
ideal as the Photon cannot send one full image in one packet due to its lack of memory. Furthermore, UDP does not guarantee delivery, arrival time, and order of arrival which may lead to missing or corrupted images.

Within the Virtual Machine, a script is required to listen to incoming TCP packets (transmitting the image) being sent to a particular IP address and particular port where the image can be directly saved onto the Virtual Machine. Node.js is used for the script as it consists of an event-based architecture and non-blocking I/O therefore provides the optimal usage of the CPU and memory which ultimately increases the efficiency of the server. Non-blocking is invaluable for scalability of the project as commands can execute concurrently, consequently allowing multiple devices to send multiple images up to server at the same time. Node.js also allows the use of callbacks which can be sent back to the device signalling success or failure.

**Database MySQL** - A simple database is also required for the projects primarily storing User Account Information and the Lateral Flow Test data/results within separate tables. Amazon Web Services provide a distributed relational database service (RDS) where database instances of a chosen engine can be deployed. The MySQL engine was chosen for this project as its preinstalled WampServer (Windows, Apache, MySQL, PHP) software stack, was used initially in the development stage on a local machine prior to the deployment stage. Apache HTTP was used as the web server for the same reason. PHP is an open source scripting language which can be embedded into the HTML of a web application. PHP was used as it can query a relational database using SQL (Structured Query Language). To aid constructing SQL queries, JetBrains's DataGrip database IDE was used for its intuitive user interface.

**Image Analysis** - The PHP executes a script which will ultimately conduct image analysis on each image which will return the resultant concentration of the analyte. Python has a vast range of image processing libraries, however OpenCV (Open Source Computer Vision Library) was primarily used alongside NumPy which is the fundamental library for scientific computing when using Python.

**Control Line Detection** - To automatically detect the control line, research was conducted into various "colour models" such as RGB (Red, Green, Blue), HSV/L (Hue, Saturation, Value/Lightness) and  $L^*a^*b^*$ . CIE  $L^*a^*b^*$  colour space was devised by the International Commission on Illumination and was chosen for this project as CIE  $L^*a^*b^*$  is designed to be perceptually uniform with human colour vision and device-independent (the model defines colours independently of how they are displayed). [11]



**Figure 2.4** Lab colour space visualisation [12]

Although the images were sent directly to AWS, the *Device Cloud* can be used as a useful debugging tool. `Particle.publish()` can be used to publish events created by the photon. Python also has a logging library which is profoundly useful with debugging routines. Apache also produces error logs for PHP and front-end exceptions.

### 2.3.3 Client-side Development

**For the front-end (client-side) development.** - HTML5 (Hyper Text Mark-up Language) was used to build a template for each web page, CSS3 (Cascading Style Sheets) was used to style and format each HTML element and JavaScript was used to create a dynamic web page. AJAX and jQuery were used to manipulate the HTML element '<canvas>' allowing regions-of-interest coordinates to be extracted then sent to a Python script (via PHP).

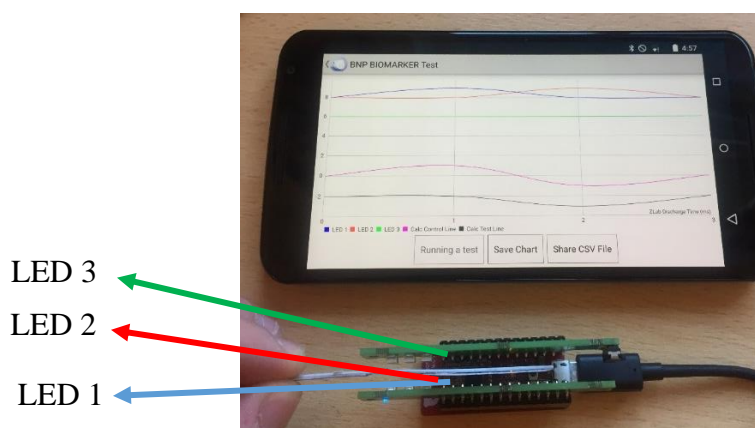
Chrome DevTools provided an invaluable debugging platform for real-time error-checking and handling. It also provided quantitative reports of the speed of the website which helped develop a faster running website. The DevTools additionally featured a tool allowing the user to tester their website's responsiveness to changing screen sizes (desktop and, mobile platforms).

## 3.0 REQUIREMENTS

### 3.1 Problem Statement

As identified through the literature review (Chapter 2.2) there is no device available commercially which successfully delivers accurate point of care-blood-test results on a timely, accurate and cost-effective manner. Currently, within NIBEC, Lateral Flow Strips are developed and this process is labour intensive; is susceptible to user-bias/error and hardware dependence (see Figure 3.1 which is based on transmitting LED-Sensor readers). Current problems include the primary method of testing whereby, the lateral flow strips are scanned using a commercial flat-bed scanner (or LED-Sensor systems) and then analysed using an open-source image processing tool (ImageJ) to analyse the images. This alone reduces the accuracy due to machine set-up variance, human error and a poor understanding of the software source. The proposed method adopted by this project improves the efficacy of blood test results whilst also removing dependencies on third-party hardware and software in addition to, human error.

The fundamental objective of this project is to create a fully functional Lateral Flow Immunoassay Reader which can be applied to multifarious medical applications. This project detects the concentration of the hCG hormone which is raised during early stages of pregnancy. This project is unique as it utilises the CMOS detector (currently in its innovative embryonic stage) as a prototype diagnostic device, providing a developer with accurate and high-speed analysis.



**Figure 3.1 - The LED-Photodiode Sensor Experimental Set-up; Individual LEDs highlighted.**

## 3.2 Requirements Gathering

Emanating from the culmination of an extensive literature review, analysing the current devices available, in addition to a series of depth interviews (semi-structured, qualitative research methods) from key personnel including biomedical/medical, academic and microfluidic engineers and related representatives; a comprehensive framework was developed providing the key determinants to producing the most efficacious diagnostic device. Highlighted below is a synopsis of the critical information as ascertained from all the interviews characterised in the form of a persona and a scenario as is presented below:-

### 3.2.1 Persona



Michael is a 40-year-old Diagnostics Engineer working in NIBEC who is tasked with the development and manufacture of lateral flow strips. His discipline is based on a biomedical background however he lacks the in-depth vision processing experience. He is frustrated with the cumbersome, current methodologies he uses when analysing the lateral strips. This is particularly apparent with larger batches where the process is laborious.

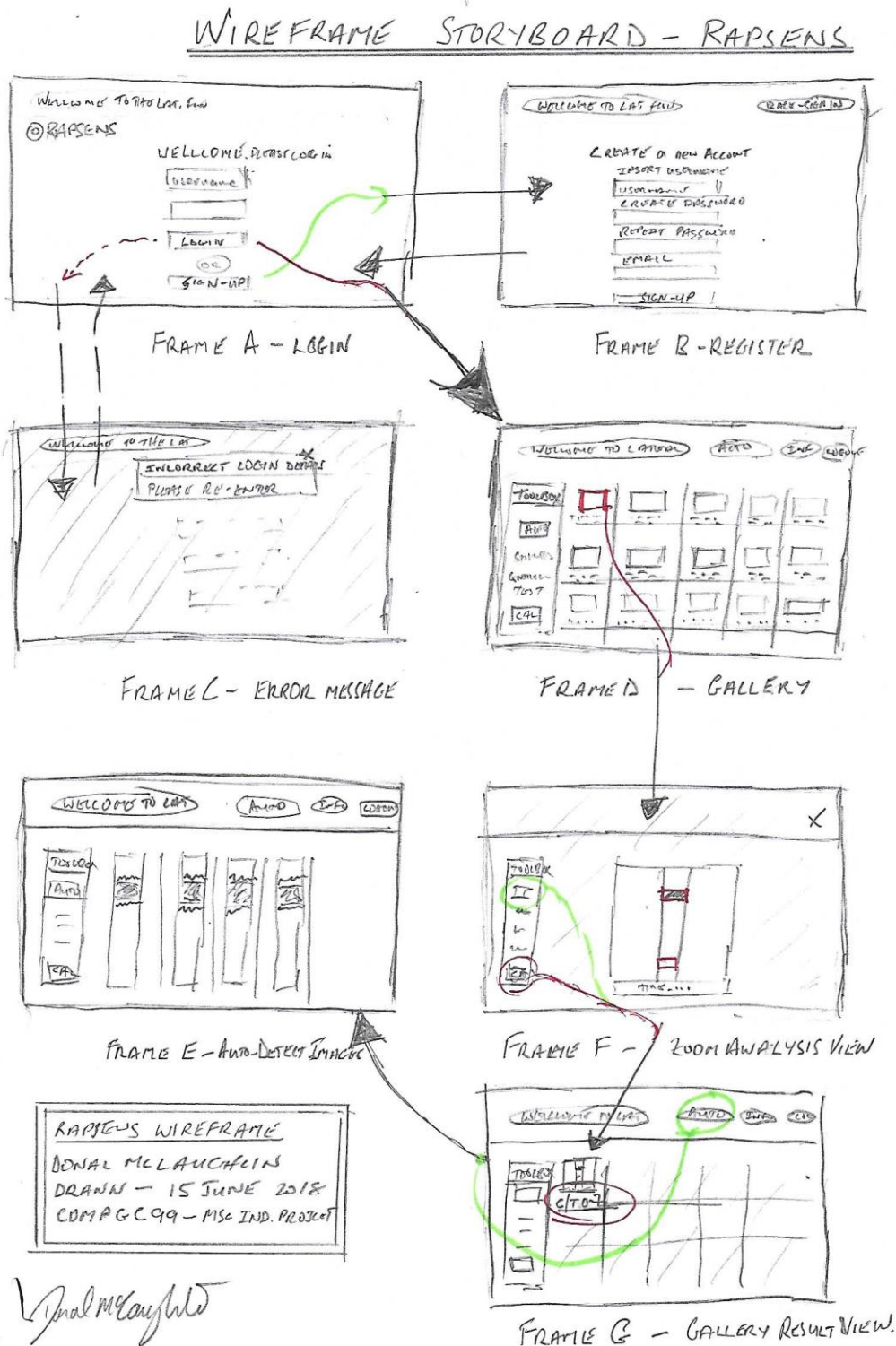
### 3.2.2 Scenario

Michael has successfully manufactured a batch of lateral flow strips which is required for the inclusion to the Qualcomm Xprize Tricorder Grand Challenge and the deadline is imminent. He is solely responsible for analysing the lateral flow strips and he is fearful that the deadline will not be met, he is using the only method available to him, the protracted system involving the flatbed scanner and third-party image analysis software.

Alternatively, by using the RapSens Reader, he inserts each lateral flow strip separately whereby this data is captured and sent to the cloud. On completion of this process, he logs into the web application and starts to draw regions of interest pertaining to each strip's test line and control line which provides results corresponding to the specific concentration of the analyte per strip. Furthermore, Michael has the capability to allow the software to automatically draw the regions of interest thus increasing accuracy between results and having beneficial time reductions. He can now plot accurate calibration lines (light intensity from strips versus concentration of analyte) through his data with high value 'least square fit correlations' thus allowing line equations to be calculated and quantitative values to be derived via an evolving algorithm. Higher sensitivities and specificities of the diagnostic outcome is the key product value, and this

allows his research projects to be pitched at the 'Low' False Positive/False Negative diagnostic market.

### 3.2.3 Storyboard with sketches



**Figure 3.2 - The Rapsens Storyboard. An initial sketch of the User Interface of the web application**

### 3.3 Requirements

After analysing the interviews and creating the persona, scenario and storyboard for a potential user of the product, a set of functional and non-functional requirements were created. As the project only lasted three months, the MoSCoW prioritisation method[13] was applied to the requirements to ensure the most important requirements were fulfilled. 'Must have' requirements are features which are critical to the successful completion of the project. 'Should have' requirements are important features but are not vital for the completion of a successful project. 'Could have' requirements are desirable and may reduce impact if they are not implemented. Finally, 'Wont have' requirements are features which cannot be delivered within the given time but provides a scope of the project.

Req. ID	Functional Requirements	Category	Priority
FReq1	The system shall allow Users to insert Lateral Flow Strips into the Reader	Reader	Must
FReq2	The system shall allow Users to take Lateral Flow Strip readings using a button	Reader	Must
FReq3	The system shall prompt the User when a reading is taken place	Reader	Must
FReq4	The system shall prompt the User if there is connectivity issues	Reader	Must
FReq5	The system shall allow Users to login using a username and password pair	Web Application	Must
FReq6	The system shall allow Users to register for an account	Web Application	Must
FReq7	The system shall allow Users to logout	Web Application	Must
FReq8	The system shall allow Users to view Lateral Flow images taken from the reader	Web Application	Must
FReq9	The system shall allow Users to manually draw Regions Of Interest on the images to calculate the concentration of the analyte	Web Application	Must
FReq10	The system shall allow Users to automatically obtain the concentration of the analyte without drawing Regions Of Interest	Web Application	Must
FReq11	The system shall allow Users to view Regions of Interest automatically plotted by the system	Web Application	Must
FReq12	The system shall allow Users to view information on how to use the analytical tools	Web Application	Must

FReq13	The system shall allow the User to view analysis results in the form of a table	Web Application	Should
FReq14	The system shall allow the User to download test results into a .csv format	Web Application	Could
FReq15	The system shall have a simple display screen	Reader	Wont
FReq16	The system can detect Lateral flow strips testing for other analytes	Reader	Wont

Req. ID	Non-Functional Requirements	Category	Priority
NFReq1	The system shall be intuitive to use	User Experience	Must
NFReq2	The system shall be portable	Reader	Must
NFReq3	The system shall allow to insert lateral flow strips with thickness of 2.5mm $\pm$ 0.3mm	Reader	Must
NFReq4	The system shall have no ambient light interference	Reader	Must
NFReq5	The system shall deliver a reliable quantification of Lateral Flow Immunoassay analyte concentration	Web Application	Must
NFReq6	The system shall display high quality images allowing for accurate analysis	Web Application	Must
NFReq7	The system shall be hosted on an Ubuntu Virtual Machine	Web Application	Must
NFReq8	The system shall store user information and image analysis into a relational database	Web Application	Must
NFReq9	The system shall output a result in less that 5 seconds	Web Application	Must
NFReq10	The system shall be secure against malicious attacks	Other	Must
NFReq11	The system shall be built using a Photon Particle	Reader	Should
NFReq13	The system shall send event messages to monitor reader status	Reader	Should
NFReq14	The system shall be battery powered	Reader	Should
NFReq15	The system shall be deployed and hosted on AWS	Web Application	Should
NFReq16	The system shall be functional 24 hours, 360 days a year	Other	Should

NFReq17	The system shall terminate all session variables if web browser is closed or after 30 minutes of inactivity	Web Application	Should
NFReq18	The system shall be compatible with major web browsers: Chrome, Internet Explorer and Safari.	Web Application	Could
NFReq19	The system shall have data recovery strategies ready	Other	Could
NFReq20	The system shall track and report website traffic and usage	Other	Could
NFReq21	The system shall be compatible with mobile devices	Web Interface	Wont

***Figure 3.3 – Functional and Non-Functional Requirements***



### 3.4 Use Cases

A detailed list of Use Cases can be found in Appendix C. A Use Case Diagram was not required as it did not provide any further useful information on how the Use Cases interact with the system and the User. Additionally, there is only one actor (Developer) interacting with the system which will lead to a minimalist use case diagram. Figure 3.4 shows a summary of the Use Cases produced and displayed in Appendix C. Within each Use Case, alternative flows are added which highlight error recovery methods. The Use Case templates were modelled using a technique demonstrated by Arlow and Neustadt in *UML 2 and the unified process*. [2]

ID	Use Cases	System
UC01	TakeReading	Reader
UC02	Register	Web Interface
UC03	Login	Web Interface
UC04	LogOut	Web Interface
UC05	ManualAnalysis	Web Interface
UC06	AutomaticAnalysis	Web Interface
UC07	ViewAutomaticAnalysisImages	Web Interface
UC08	CheckHelpInformation	Web Interface
UC09	ViewHistoryOfResults	Web Interface
UC10	DownloadCSVOFResults	Web Interface

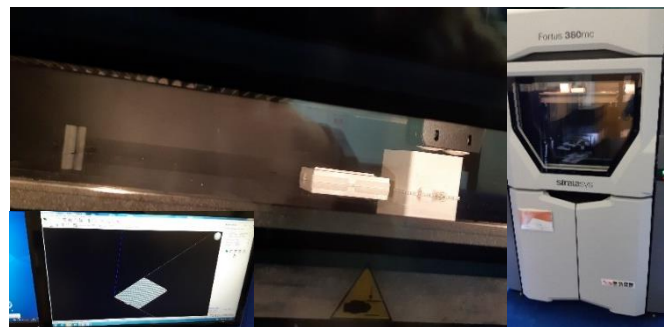
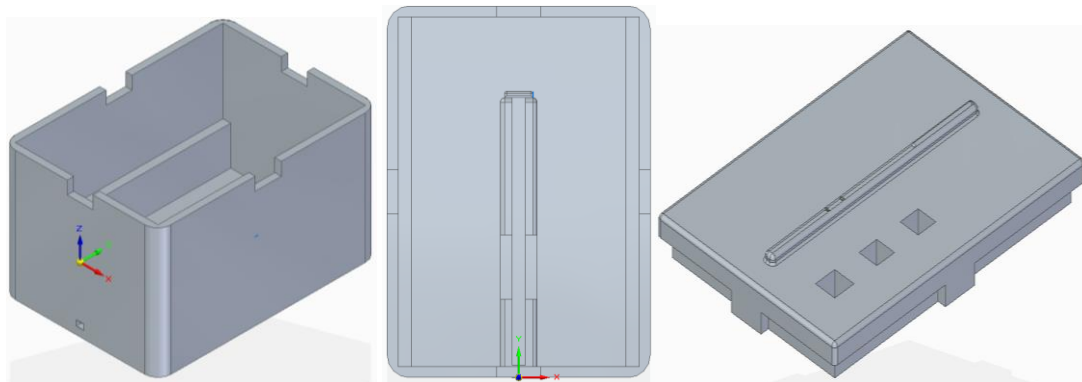
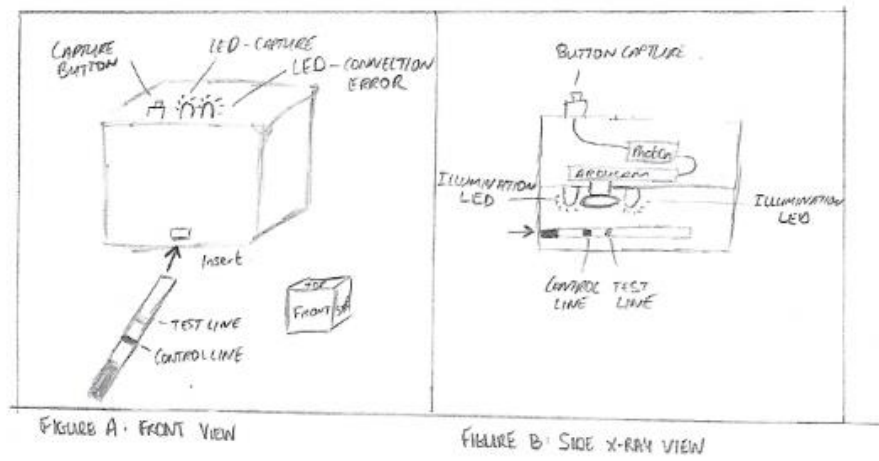
**Figure 3.4 – Project Use Cases**

## 4.0 DESIGN AND IMPLEMENTATION

### 4.1 Design, Housing and Assembly Architecture Implementation

A case assembly was designed using the CAD design package Solid Edge. The author was able to attempt six key-stage iterations, using visualisation methods and additive printing to team-discuss before finalising a set of drawings (saved as Solid Edge Part files) which were converted to .stl (3D object) files suitable for 3-D printing.

The high tolerance housing prototype was made from Acrylonitrile Butadiene Styrene (ABS) and was produced in 3 parts to allow assembly of electronics, 2 boards plus CMOS camera with designed lens , sensors and associated cabling/wireless set-up. A base-plate provided the important slider-rail to allow strip alignment and accurate registration with the camera. It was important that the material was light-proof and black paint spraying was required to prevent background ambient light leaking. Careful assembly of the components was necessary to ensure no breakages; good wireless alignment and accurate/robust positioning of the CMOS sensor and associated lens which had to be designed and modified to achieve a suitable optical focal length/view.



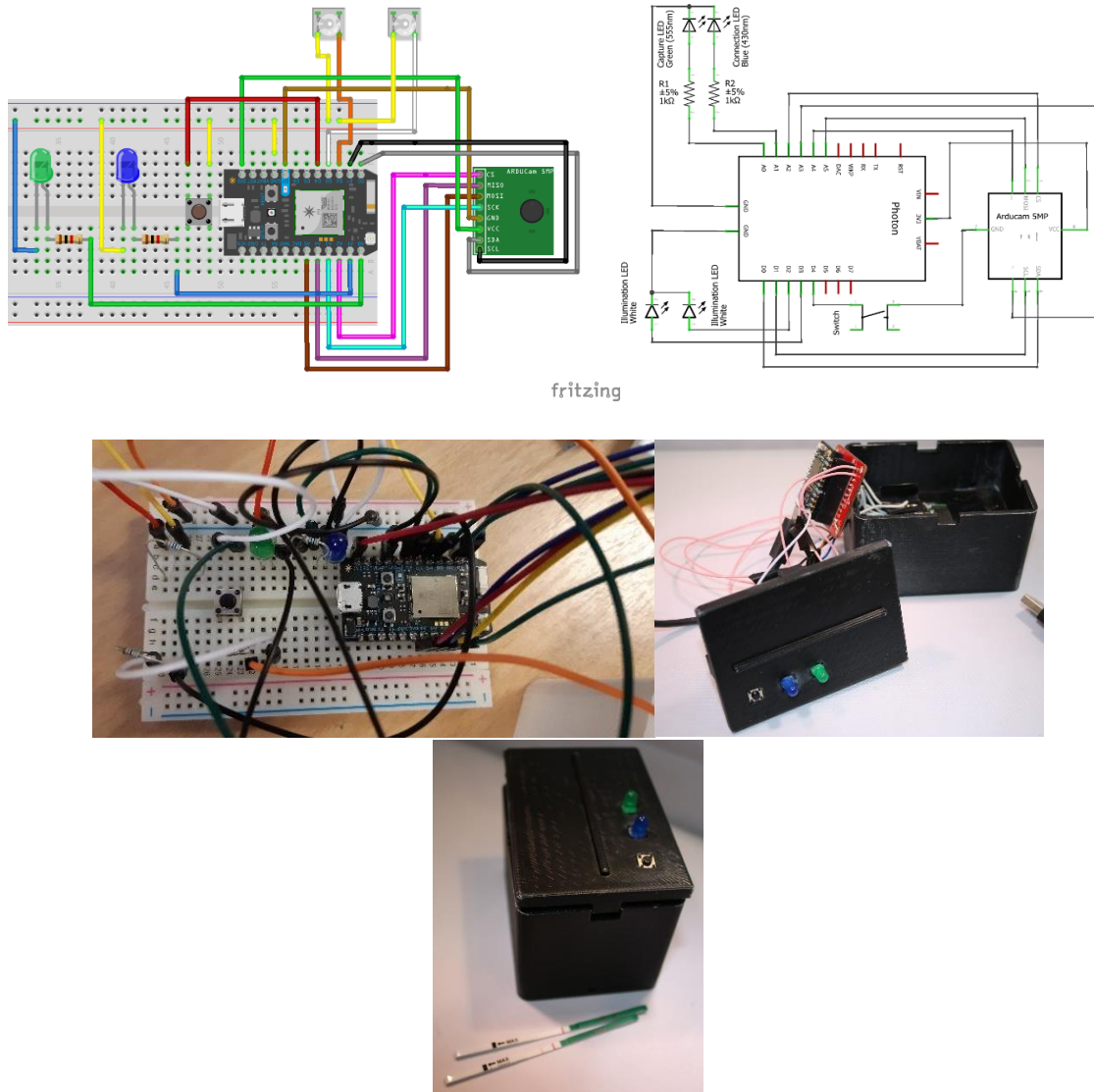
**Figure 4.1 – Early Sketches of Rapsens; 3D CAD drawings; 3D Printer and assembly of Rapsens system.**

## 4.2 Hardware Implementation

Figure 4.2 illustrates a breadboard view of the Lateral Flow Reader and a circuit schematic view of the Reader. The final prototype was assembled with all connections soldered to reduce movement between the contacts, reducing inter-lead capacitance which contributed to lost / error image data. The Reader consists of the following components:

- A Photon Particle
- An Arducam OV5640 PLUS MINI
- 2 Through-hole LEDs
- 2 Surface Mount (SMD) LEDs
- 2 1K $\Omega$  Resistors and,
- A push button switch

The Photon can be powered via the on-board USB Micro B connector or directly via the VIN pin. To increase portability, a battery shield was connected to the VIN pin. The Green Through-hole LED was used to communicate to the user whether an image was currently being captured and sent to the server. The Blue Through-hole LED tells the user that the Reader is having difficulty connecting to the server. Two Surface Mount (SMD) LEDs were positioned on either side of the CMOS sensor and directed towards the lateral flow strip, thus isolating the light imaging area within a light sealed zone. Two SMD LEDs were used to distribute the light evenly onto the strip to prevent shadowing and unbalanced lighting which will lead to skewed results when analysing the 'mean-grey intensity'. The SMD LEDs produced high intensity light which had to be significantly reduced by using heavy duty resistors (3K $\Omega$ ) to reduce voltage and reducing the analogue value (reducing the pulse width of the pulse width modulation PWM) to the SMD LED pin.



**Figure 4.2 – Breadboard layout for Rapsens Prototype; Electrical schematic drawings; Breadboard and assembly of Rapsens system.**

### 4.3 Firmware Implementation

As the Particle uses the Wiring IDE, the firmware consists of two functions: *setup()* and *loop()*. The *setup()* function is run once at the beginning of the program and is used to define initial environmental variables, whilst the *loop()* function repeats until the device is reset or turned off. Within the Lateral Flow Reader's firmware, the *setup()* function: configures all the LEDs as outputs, sets the push button switch as a 'pullup input', checks if the CMOS sensor is connected to the Arducam shield, ensures the correct sensor is used with the firmware, establishes a connection to the server and prepares the sensor for capturing images.

Initially, the loop() function checks if the photon can connect to the server at the IP address 54.85.56.7 and port 5550. If the connection between the server and the photon cannot be established, the Blue LED turns on and remains on until a connection is made. The loop function loops continuously with no events until the push button is pushed (set LOW). At this moment, the Green LED will illuminate as well as the two SMD LEDs which will illuminate the lateral flow strip. At this point, the fifo read pointer is set to zero and the capture completion flag is removed from previous captures. The Arducam shield will then capture and store the image onto the 8MB onboard frame buffer. If the capture takes too long (greater than 30 seconds), the capture will terminate. This is likely to be caused by a disconnection between the CMOS sensor module and the Arducam shield. If the capture was successful, the burst read operation is called, thus reading multiple bytes out of the frame buffer. The image data must be separated into multiple fragments as the photon (256KB of RAM) does not have the required memory to save the image (roughly 400-700KB). The data is then sent to the server as a series of bytes using an array called buffer. The array is parsed using the tx\_buffer index until the maximum buffer size is reached for each fragment and is then sent to the server. This is repeated until the complete buffer has been sent. After successful completion, the Green LED and the two SMD LEDs are switched off. The button switch is then set to HIGH allowing the process to repeat itself if the push button switch is pushed (set LOW).

The loop() function also has multiple failsafe functions such as: if the photon cannot connect to the node.js TCP server; if the image takes too long to capture (or takes too long to send to the server) and if the image size is too large (greater than 2MB). The firmware will also detect whether a strip has not been inserted and the connection LED will blink twice indicating no image was sent.

#### 4.4 Cloud Implementation

Within Amazon Web Services, a Virtual Private Cloud (VPC) was created to contain both an Amazon RDS instance and an Amazon Elastic Cloud Computing (EC2) instance. An Elastic IP was assigned to the EC2 instance ensuring a static IP was created which does not change in the event of the EC2 being shut down and restarted. The EC2 instance was shut down during periods of inactivity to reduce costs. The EC2 instance was deployed with an Ubuntu Server 16.04 LTS Machine Image and Elastic Storage Block. The Machine Image provides a basis for the root volume such as the operating system, the application server and the applications [14]. At the end of the project, this will allow an easy handover of the Virtual Machine and storage block to the clients at NIBEC. Packages were then installed onto the Ubuntu machine: Python 2.7

(including required libraries), Apache HTTP, MySQL, PHP and required JavaScript libraries such as jQuery and Node.js.

Amazon Relational Database Service (RDS) was used to run a database instance using a MySQL engine. Amazon RDS provided invaluable services which allowed intuitive connection to the database instance using JetBrains's DataGrip to generate SQL statements for creating and managing the database. The database consists of only two tables: users and imageAnalysis. The users table stores login information for each user such as username, password and email address. The password is inserted into the database using the hashing algorithm MD5 which adds a layer of security. The imageAnalysis table consists of six columns: image (image name), testCoord (coordinates of the test line), contCoord (coordinates of the control line), contMeanGray (mean grey intensity of the control line), testMeanGray (mean grey intensity of the control line) and contOtest (ratio between the control line and the test line). These values are inserted into the database by the Python script and selected by PHP-SQL queries within the web application.

Throughout the Photon's firmware, publish events were sent to the Device Cloud which gave real-time status updates of how the Reader was performing. Such events published were: detection of the CMOS sensor to the Arducam board; start and end of image capture, TCP packets pushed to the server and error messages such as: images being too large and disconnection from the server.

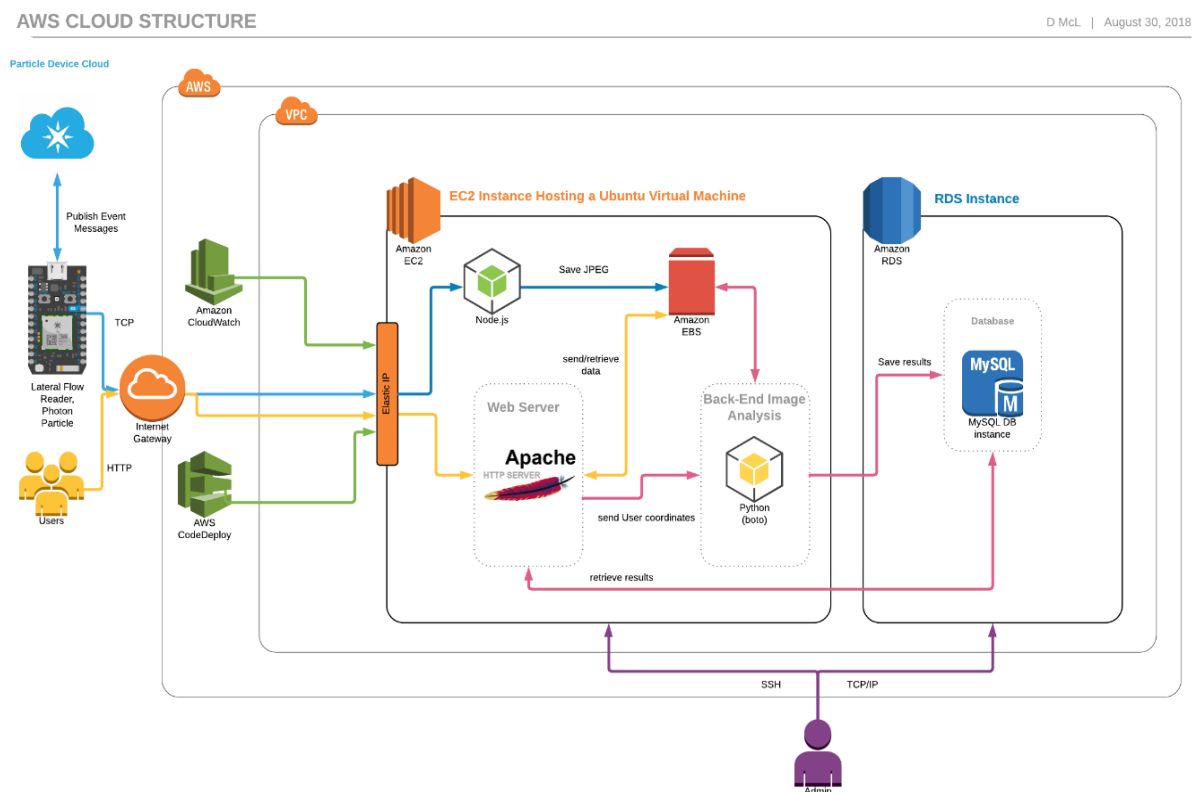
When an image has successfully been taken, the image data is split into a buffer array and is sent to the TCP server in the EC2 instance as a series of bytes. The TCP server was created using Node.js which listened for the TCP packets, joined them together back into a complete array and saved the data as a JPEG into the EC2 instance's Elastic Block Storage. Within the EC2 instance, the Apache HTTP web server stores, processes and delivers web pages (HTML, CSS, JavaScript) to the users. The images are retrieved from their directory, /Images/, and displayed on the web page using a PHP foreach loop.

From the gallery web page (index.php), users can either draw their regions of interest or allow the web application to draw regions of interest automatically. Either way, the coordinates (strings) are passed to a Python script for image analysis. The result, which is the ratio of the mean grey intensity of the control line against the test line, is inserted onto the MySQL database where the results can be displayed on the web page using a PHP echo.

For an Administrator to manage and monitor the EC2 instance, they would use the Secure Shell (SSH) protocol alongside a private key file generated by the EC2 console. To access the RDS database instance, the admin must fill in the correct username and password which was registered for that particular database instance or endpoint.

To automate the deployments to the EC2 instance, AWS CodeDeploy was used to push and pull content from the instance to a GitHub Repository. Amazon CloudWatch was used to monitor and optimize instance events using its wide range of logs, metrics and troubleshooting tools.

Figure 4.3 portrays the design architecture of the overall project including how each individual component interacts with one another.



**Figure 4.3 – Overall schematic and systems diagram of the project’s key components.**

## 4.5 Image Receiver Script Implementation

As the Photon’s firmware transmits the image data as a series of bytes, a script must be created to receive this series of bytes, combine them and save the complete image onto the server. To do this, a Node.js script was used (main.js). The ‘net’ module was imported as it supports an asynchronous network API for producing stream-based TCP servers. The TCP server listens to incoming TCP packets on port 5550. If a connection between the TCP server and the Photon has been established, each image data fragment is read then passed as a parameter to the function



'delayBufferFn', where each segment is pushed onto an array called buffer. After a period of inactivity set using the setTimeout function, (i.e. no further TCP packets are sent to the server) the complete image data is sent to the saveImage function where it is stored into a subdirectory called Images using the File System Module. Each image's file name is saved as the date and time (to milliseconds) thus, making each image unique unless, two images are received at the same time (up to 0.01 seconds). This problem is outside of this project's scope as only one device has been made. To resolve this in the future, the image data could be transmitted along with a device ID to distinguish between each device. AWS does not yet support connecting Particle devices to AWS's IOT service, this feature would help with organising and monitoring IOT devices connected to AWS.

## 4.6 Web Application Implementation

### 4.6.1 Login and Registration

To access and analyse data sent from the Rapsens reader, the user must register and login. The web application homepage is login.php and to access any other webpage on the web application, PHP session variables must be created. The username and password of the user was used as the PHP session variables and before any other web page is loaded, the username variable and password must match the username-password pair which has been stored in the database (users table). The session variables are destroyed after 30 minutes of inactivity or, if the browser is closed down or; if the user logs out of the application. Using HTML <input> components are secured against malicious attacks such as HTML and SQL injections. This was accomplished by trimming, removing slashes, removing html special characters and 'escaping' special characters (using mysqli\_real\_escape\_string) of the data inserted and posted by the <input> components.

### 4.6.2 Image Gallery, Manual and Automatic Image Analysis

All web pages after the login screen contain (using PHP include statement) a header and footer which are implemented using PHP files. The header.php contain links to other webpages allowing the user to easily navigate through the web application. The header.php file also contains the database connector method which is required for all web pages. Each web page contains a Google Analytics Script allows future web developers of the web application to analyse traffic visiting the web application.

After a database connection has been established, a foreach loop is used to display the images which have been saved from the TCP server Node.js script. These images are inserted into a HTML <div> where the time and date of capture for each image is displayed underneath. The

resultant concentration of the analyte is displayed underneath the date and time if analysis has taken place (if analysis has not been done, 'Pending' will be displayed).

To start analysis of a particular image, the user will double click an image and a modal will display an enlarged image. To manually draw on the image, a HTML `<canvas>` element and five JavaScript functions are used. When the user left clicks on the `<canvas>` element the `mousedown()` function is called and the starting X and Y coordinates are displayed in the toolbox. When the user then drags the mouse (whilst still holding down left click), the `mousemove()` function is called which updates the coordinates (in real-time) within the toolbox. When the user is satisfied with their chosen Region of Interest box, they will release the left click on their mouse and the `mouseup()` function is called which add the coordinates of the Region of Interest square to a hidden `<input>` form field. This is repeated for the test line. (It should be noted that no external library was used to help with the programming of the manual draw ROI). To generate the concentration, the 'Calculate' button is clicked which will execute a post event. Using the 'Auto Calculate' button is ultimately less time consuming as the user only has to select an image.

There are two post methods present within the same webpage, as conveyed in Figure 4.4 Code was created to execute different Python scripts for their corresponding input. Case A conforms to the Auto Calculate button and only passes the image name as a parameter to the `autoThresholding.py` script. On the other hand, Case B corresponds to the manually drawn regions of interest button (Calculate button) and passes three parameters to the `imageAnalysis.py` script: image name, test line coordinates and control line coordinates. The `shell_exec()` function was used to execute the python scripts via the Ubuntu shell (no results were returned as they were stored in the database instead).

```

if ($_SERVER['REQUEST_METHOD'] == 'POST'){
    if (isset($_POST["form"])) {
        switch ($_POST['form']){
            case "A":
                $imgName = $_POST['aimnamejstophp'];
                echo $imgName;
                $cmd = "python autoThresholding.py $imgName";
                $output = shell_exec($cmd);
                break;
            case "B":
                $contCoords = $_POST['xcontCoords'];
                $testCoords = $_POST['ytestCoords'];
                $imgName = $_POST['imgnamejstophp'];
                $cmd = "python imageAnalysis.py $contCoords $testCoords $imgName";
                $output = shell_exec($cmd);
                break;
        }
    }else{
        echo "error";
    }
}
}

```

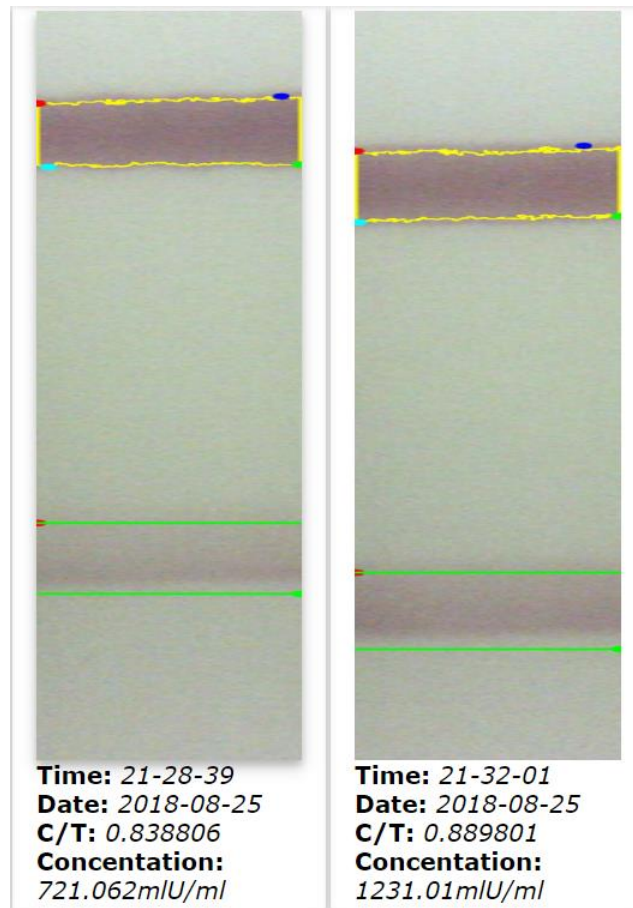
**Figure 4.4 – Code Extract from index.php**

#### 4.6.3 Record Viewer and CSV downloader

For further analytical research outside the scope of this project's application, the user can view a history of all records of image analysis conducted on the web application and download a csv file displaying the records. The recordsViewer.php displays a table with a detailed view of each image's analysed features. DataTables is a jQuery plugin which was used to format the table allowing search and sorting features to be implemented.

#### 4.6.4 View Automatically Detected Regions Of Interest

Within the autoThresholding.py script, the images are saved into the "autoImage" directory with an overlay of the automatically drawn regions of interest. The autoImages.php displays the images with the overlay using a foreach loop similar to the loop used in the index.php (Section 4.6.2). The resultant images of the Python script autoThresholding.py are displayed in a format as seen in Figure 4.5.



**Figure 4.5 – Automatically selected ‘Regions of Interest’ for the Rapsens Image Lateral Flow strips**

#### 4.7 Python Manual Analysis Implementation

Initially, the connection is established to the MySQL statement using the `mysql.connector.connect()` function. Throughout both python scripts, loggings are used as a platform for debugging. As the manual script is imported into the auto-detection script, an if statement (Figure 4.6) was created ensuring the main function was not called twice.

```
if len(sys.argv) > 2:
    contCord = str(sys.argv[1]) # Control Coordinates
    testCord = str(sys.argv[2]) # Test Coordinates
    imageNameold = str(sys.argv[3]) # Image Name
    main(imageNameold, contCord, testCord)
```

**Figure 4.6 – Code Extract from *imageAnalysis.py***

The `main()` method contains three arguments: the image name, the control line coordinates and the test line coordinates. Within the manual selection scenario, the three function arguments

are passed to the python script as command line arguments. The command line arguments are in a String format where the coordinates were comma separated. Firstly, the main function converts the coordinate strings into lists, then assigning each list component to their appropriate X/Y and test/control line label. Hereafter, the image is edited and processed using the computer vision library OpenCV abbreviated to 'cv' in the code. Using OpenCV's algorithms, the image is read and saved as a variable (img) and then converted to greyscale. Using the two sets of formatted coordinates, the image is cropped into two variables: crop\_img\_cont and crop\_img\_test. Using the mean() within OpenCV, the mean grey intensity of each new cropped image is calculated and then a control line to test line ratio is calculated. This ratio can then be used to predict the value of the concentration of the hCG hormone. The image name, coordinates of both the control line and test line, the mean grey density of both the control line and test line and the 'control-line test-line' ratio are recorded on to the MySQL database. The resultant ratio is returned and presented in the appropriate HTML components within the web application.

#### 4.8 Python Auto Analysis Implementation

When the 'Auto Calculate' button is pressed within the web application, the selected image's name (dateAndTime.jpg) is passed as a command line parameter to the python script: autoThresholding.py. The image is initially cropped to ensure only the centre part of the strip undergoes the image analysis. This is manually done within the python script as the strips are fixed into position by barriers within the reader. Only the centre region of the strip is analysed as shadowing and excessive clustering of the gold colloid around the strips edge leads to skewed results.

Using the same OpenCV module, the image is read and stored as a variable (uncroppedImage) and converted to the L\*a\*b\* colour space (discussed in Chapter 2). Using a L\*a\*b\* upper bound of (77, 0, 0) and a lower bound of (159, 255, 255), a mask could be applied to the image. The mask highlighted the region (in white) which corresponded to the control line whilst the rest of the image was left black.

Using the mask, contours (Figure 4.7) could be drawn around the control test line. For increased accuracy, it is advised to convert the image into a binary image, hence why a threshold mask was applied. As the findContours() method manipulates the variable that is passed into it and the threshold image (thresh) is copied. Instead of plotting every single contour point, cv2.CV\_CHAIN\_APPROX\_SIMPLE is used to remove all redundant contours points, thus compressing the contour and saving memory. Likewise, cv2.RETR\_EXTERNAL only retrieves the

extreme contour points. The third line in Figure 4.6 is a precaution and checks whether OpenCV 2 or OpenCV 3 is installed and acts accordingly. The variable 'c' then is saved as the contour drawn around the control line. If multiple contours are found within the image (created by shadowing or irregular clustering found on the strip due to a manufacture defect), the maximum area is selected.

```
# find contours in thresholded image, then choose the largest one
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
c = max(cnts, key=cv2.contourArea)
```

**Figure 4.7 – Code Extract from *autoThresholding.py* [17]**

Using these contour points, it is possible to find the extreme contour points which correspond to the top and bottom of the control line. Using these points, a fixed distance is added to the coordinates thus locating the test line. For a given batch of strips, the distance between the control line and the test line remains constant. Now that the control lines coordinates and the test line coordinates have been found, the main function within the manual detect script is imported and the main function is labelled with the selected image name, and the two sets of coordinates are passed as arguments.

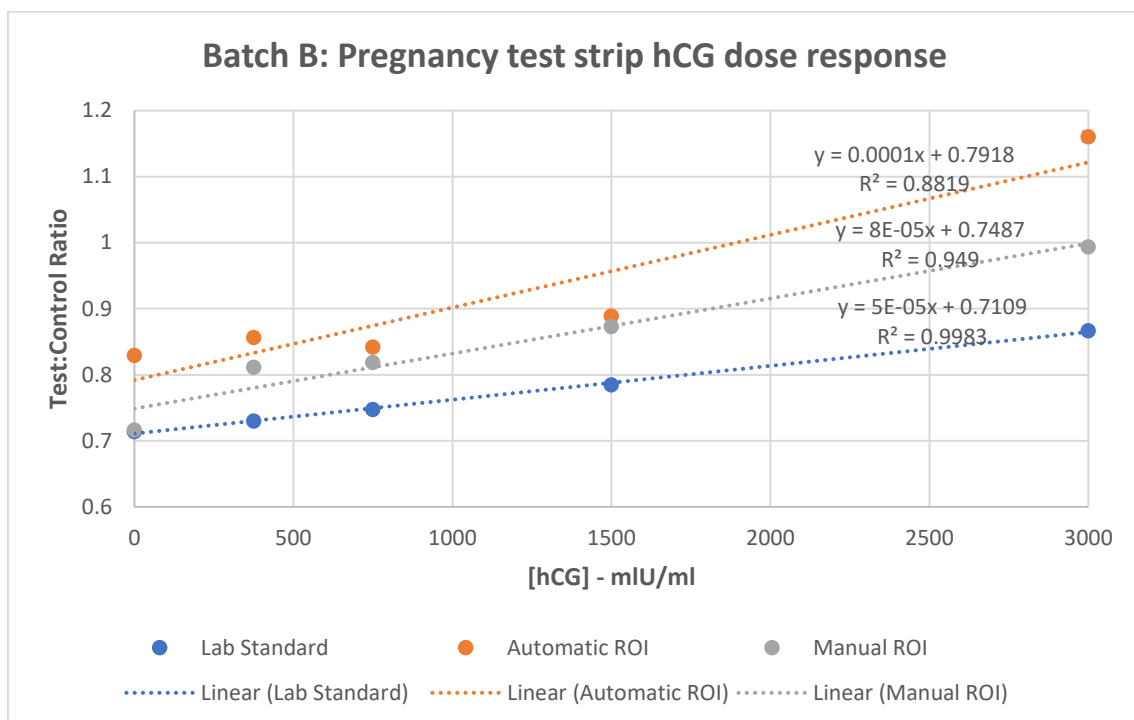
## 5.0 EVALUATION

### 5.1 Analysis

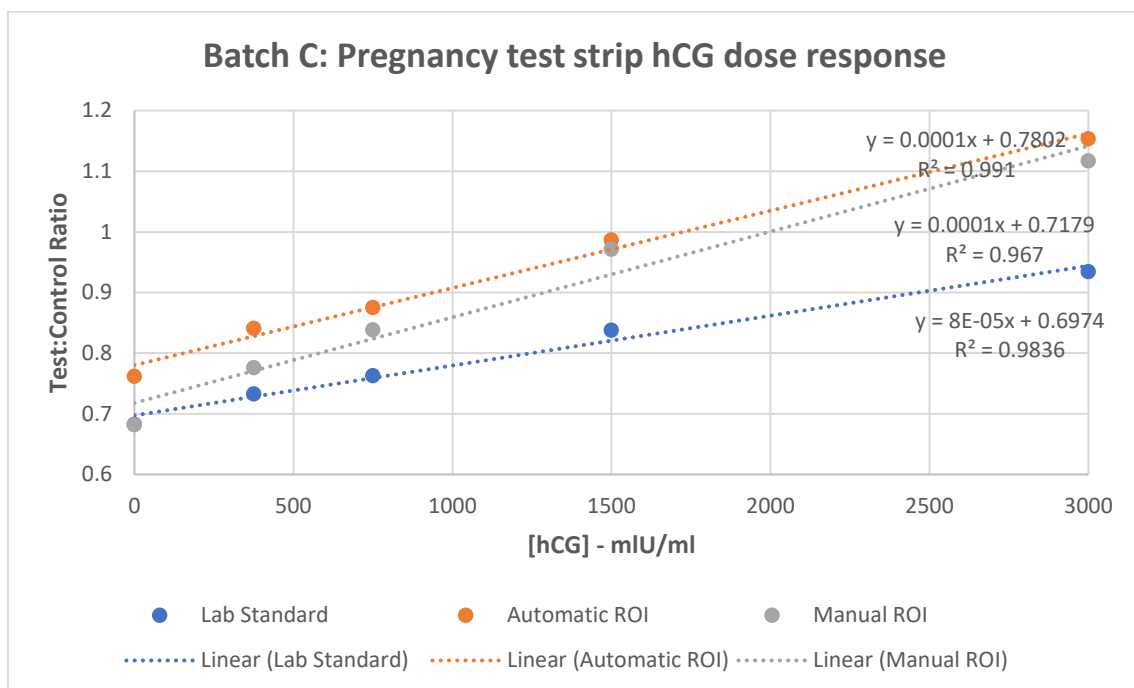
To determine the reproducibility and sensitivity of the reader, a set of tests were carried out (approximately 70 tests). Using three batches of lateral flow strips, which have been doped with the hCG pregnancy hormone, a set of reproducibility, error analysis and sensitivity experimental tests were executed. Each batch consists of 5 strips each containing the following individual concentrations of hCG: 0 mIU/ml, 375 mIU/ml, 750 mIU/ml, 1500 mIU/ml and 3000 mIU/ml.

To test the reproducibility of the CMOS detector and software, the strips were analysed five times (removing and reinserting the strip). This was carried out to ensure that the same result is generated within a reasonable uncertainty and slight misalignment errors (within the barriers of the reader) do not affect result accuracy. The reader proved to be very reliable regarding reproducibility, as repeated analysis of the same strip only varied by 0.5% over the full range. This does not overlap with other concentrations as illustrated within the error bars of Figure 5.1, Figure 5.2 and Figure 5.3.

There exists a variation between the batches as shown in Figure 5.4, Figure 5.5 and Figure 5.6. All three methods show the same trend in variation therefore, it can be ruled out that the Rapsens device is not at fault, in causing the variable results. It is highly possible (according to NIBEC biochemists) that there could be manufacturing defects between each batch, such as different gold colloid chemistries/ functionalisation methods used or a variation in the quantity of gold colloid used. Additionally, as shown in all three methods, there is a minuscule distinguishable difference between the hCG concentrations of 375 mIU/ml, 750 mIU/ml across all batches. Even by eye, it is hard to distinguish between the two concentrations. This may be caused by the small incremental differences between the concentrations. The concentrations double in increments however, at lower concentrations, this incremental difference may be too small (incremental difference 375 mIU/ml) to detect measurable change in comparison to higher concentrations (incremental difference >1500 mIU/ml).

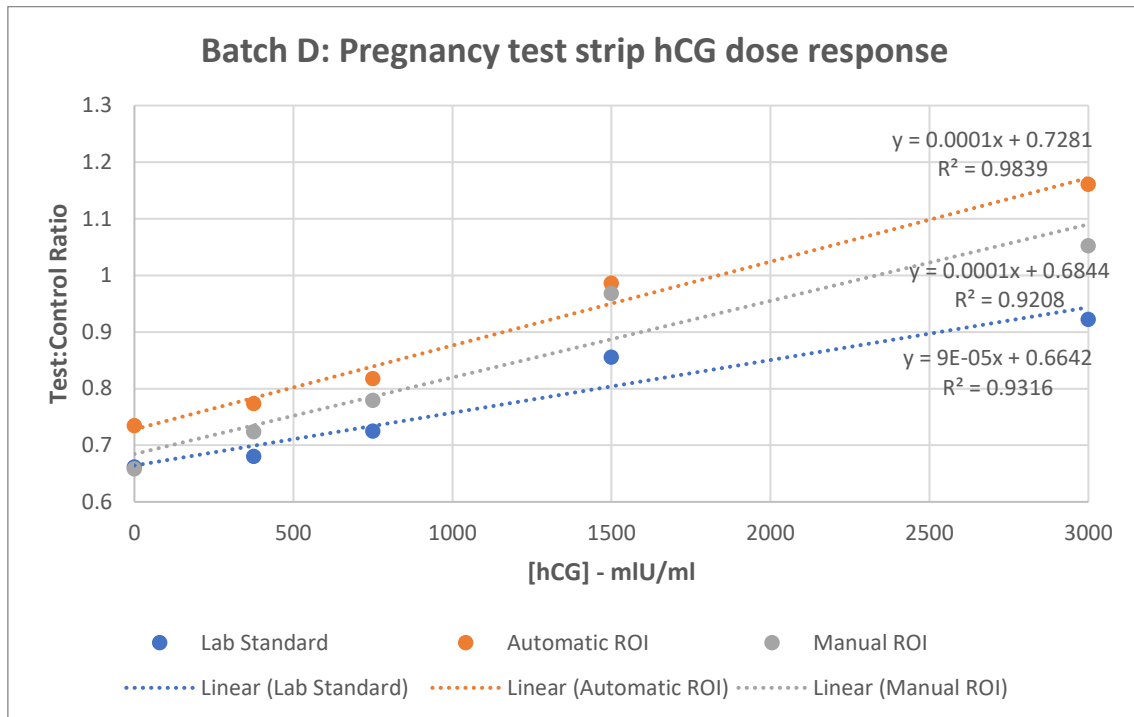


**Figure 5.1 – Batch B: Pregnancy test strip hCG dose Response**

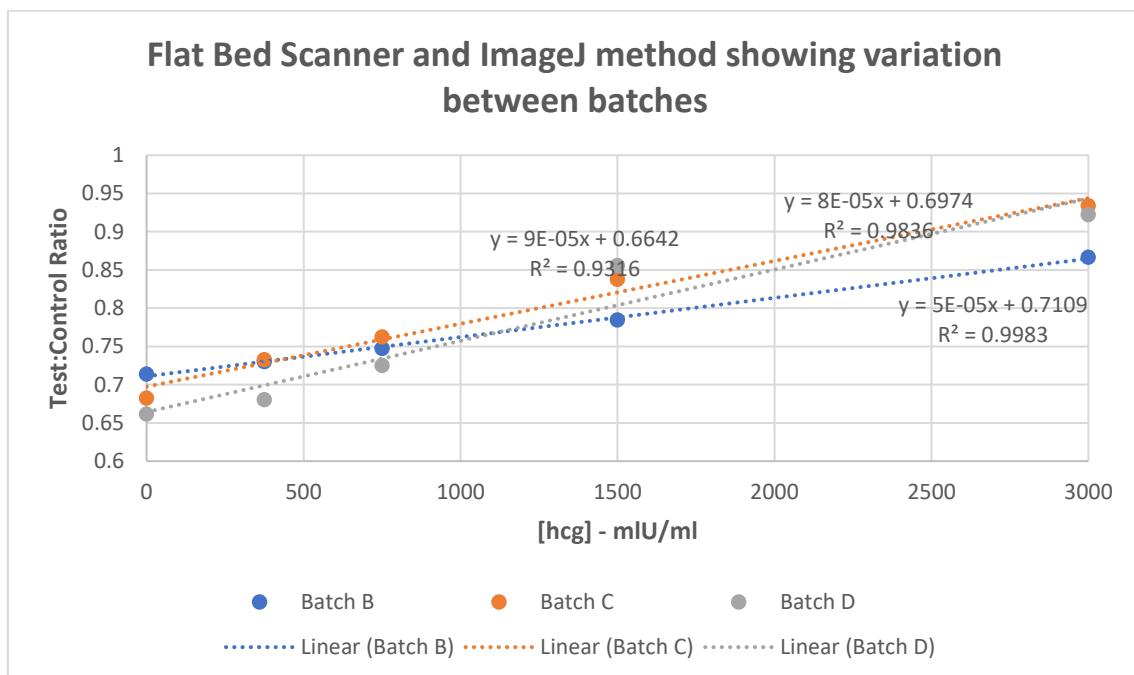


**Figure 5.2 – Batch C: Pregnancy test strip hCG dose Response**

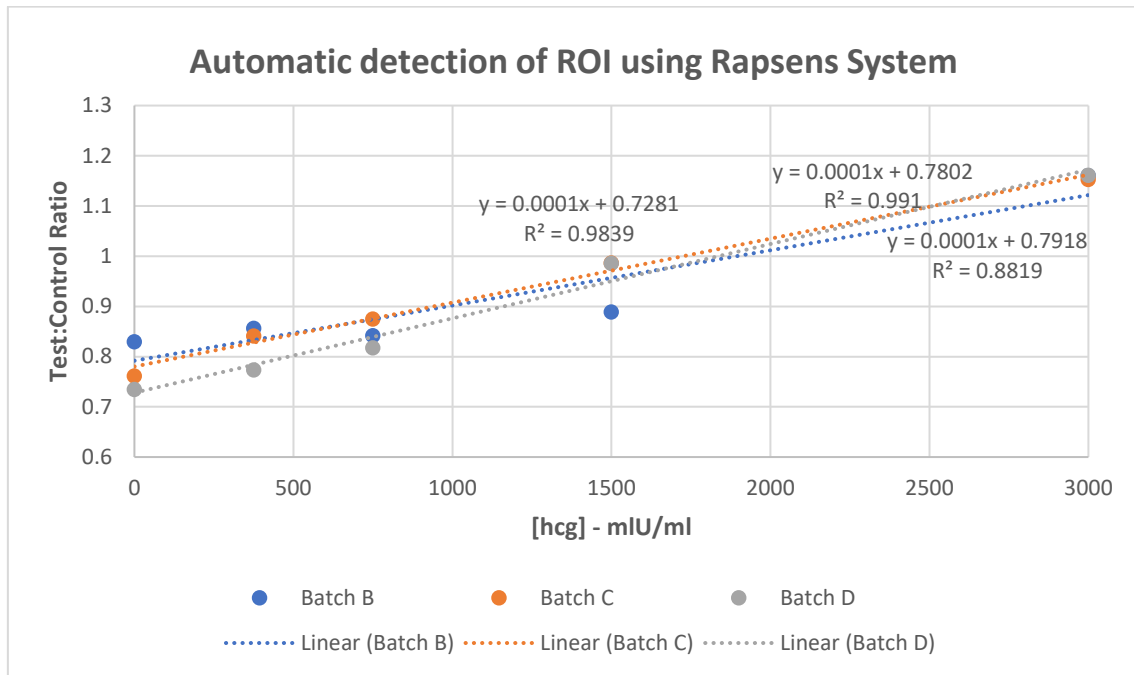




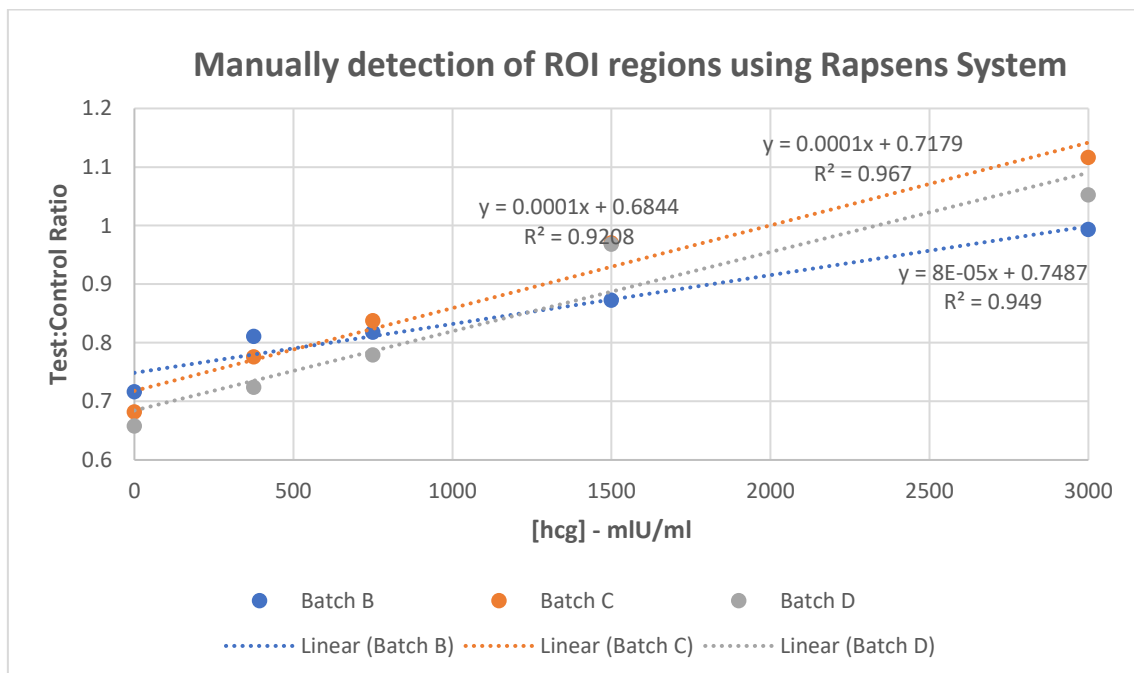
**Figure 5.3 – Batch D: Pregnancy test strip hCG dose Response**



**Figure 5.4 – Flat Bed Scanner and ImageJ method showing variation between batches.**



**Figure 5.5 – Automatic detection of ROI for Rapsens System showing variation between batches.**



**Figure 5.6 – Manual detection of ROI for Rapsens System showing variation between batches.**

## 5.2 Browser Compatibility and Web testing Tool

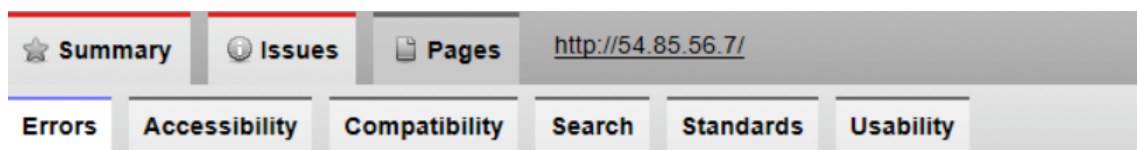
Figure 5.7 was generated using SortSite[16] which is a web crawler tool which analyses the web application's performance and browser compatibility. The results showed that the all features implemented on the web application are compatible with all the major web browsers and there are no broken links or server configuration issues.

This tab shows pages that exhibit browser-specific behavior, or trigger browser bugs.

Browser	IE	Edge	Firefox	Safari	Opera	Chrome	iOS	Android	
Version	11	17	61	≤ 10 11	54	68	≤ 9 10 11	≤ 3 4*	
Critical Issues	✓	✓	✓	✓	✓	✓	✓	✓	✓
Major Issues	✓	✓		✓	✓		✓	✓	✓
Minor Issues	✓	✓		✓	✓		✓	✓	✓

**Key**

- Missing content or functionality
- Major layout or performance problems
- Minor layout or performance problems



This tab shows site quality issues, including broken links and server configuration problems.

- ✓ **Broken links** - No issues found.
- ✓ **Server configuration**
- ✓ **ASP, ASP.NET and PHP script errors** - No issues found.
- ✓ **Internet RFCs** - No issues found.

**Figure 5.7 – SortSite Performance testing results.**

## 5.3 Automated system testing for the web application

To test all the functionalities within the use cases which apply to the web application, Selium IDE was rendered to most suitable software testing framework on this occasion. A complete automated walkthrough of the web application was carried out with no errors as seen in Appendix C.

## 5.4 Alpha and Beta Acceptance testing

The final testing phases is undertaken to gauge the interaction and response from the client. Initially a round of alpha acceptance testing was conducted by the author ensuring Interaction Design Principles are in situ and to ensure no bugs remain within the system. It is subsequently revealed to the client within a round of beta acceptance testing. This is a time consuming process whereby the client is led carefully through the process (train the trainer). It was established at this meeting that the prototype and the system was successfully commensurate to the Functional and Non-Functional Requirements specified in Chapter 3.0.

## 6.0 CONCLUSIONS

### 6.1 Project Goals and Requirements Evaluation

Overwhelmingly, as postulated in this thesis, the Lateral Flow Immunoassay (LFI) CMOS-sensor based Reader system and web application developed and manufactured for this project is both more efficacious and cost effective in comparison to the current process. The project was successful in completing all the MoSCoW Must have, Should have and Could have functional and non-functional requirements. A final, fully operational prototype which fulfils all the diagnostic development criteria and objectives of this project is available for demonstration or viewed via a submitted video. Furthermore, a publication is in progress and will be submitted in September 2018. The success of this project can be delineated by the following criteria: -

#### 6.1.1 Hardware, Firmware and Housing Evaluation

The project successfully delivered a robust and easy-to-use Lateral Flow Immunoassay (LFI) CMOS-sensor based Reader system which transmits image data via the TCP protocol to a server within Amazon Web Services. The hardware was constructed at minimal cost, in addition to generating high quality images (2592x1944px). Large volumes of data were passed between the Arducam shield and the Photon Particle therefore, connections had to be perfectly soldered ensuring no loose or flimsy connections. Images appeared broken as data was lost due to slack connections however increasing buffer sizes and securing connections ensured images were sent correctly.

The firmware was programmed to accommodate the photon's small memory size by sending the image data in parts (as the image size is larger than the photon's memory). As well as the reader communicating efficiently with the server, the human-computer interaction aspect of the reader was explored which resulted in the reader making use of LEDs to communicate competently with the user. The firmware also safeguards against hardware issues and user errors such as, forgetting to insert a lateral flow strip.

The housing (which was fabricated using Computer-Aided-Design and 3-D printing) successfully held lateral flow strips into position, blocked out all ambient lighting and stowed the hardware into a lightweight but durable container.

#### 6.1.2 Cloud Architecture and Web Application Evaluation

An Elastic Cloud Computing (EC2) instance hosted a Ubuntu Virtual Machine which successfully combined image data and saved images (sent from the photon) onto the Virtual Machine. Within the Virtual Machine, an Apache HTTP web server hosted a web application with a clear and

intuitive user-interface. The user can draw their own regions of interests or allow the web application to automatically draw them. It was demonstrated that the reader has a high threshold of reproducibility as repeated analysis of the same strip only varied by 0.5% over the full range of strips.

It was also established when the user draws the regions of interest manually, a larger variation in the data was found in comparison to the web application automatically detecting the regions of interest. This is due to user-bias whereby the user may select the leading edge of the test line and not the complete line (which may not be fully visible). A similarly large variation was found when using already existing methods.

## 6.2 Personal Aims Evaluation

The personal aims of this project were primarily focused on learning new programming languages, designing and developing a complete and fully functional IOT device and deploying the system on Amazon Web Services which will totally enhance the rudimentary technique currently utilised. The actualisation of this project necessitated the learning of several programming languages simultaneously including: Wiring to program the Photon's firmware; Python for image analysis using OpenCV and Numpy libraries; Node.js to create a TCP server for receiving image data. Furthermore, prerequisite to the project the author used: Computer-Aided-Design software (Solid Edge) to 3-D print housing for the reader; a Ubuntu Virtual Machine to host an Apache HTTP web server as well as the Image Analysis logic; and Amazon Web Services EC2, EBS and RDS instances to host all cloud applications.

In furtherance, this project capitalised and consolidated upon the author's previous experience with the web technologies HTML, CSS and JavaScript to template and style a dynamic web application. Additionally, server-side programming languages such as PHP to query the MySQL database using SQL were further enhanced in this project.

Unique to the technicalities of this project, soldering and electronic skills were learned and adopted to design and manufacture the finished product.

## 6.4 Future Work

As already iterated, the proposed device and system was successfully manufactured and it demonstrates that it supersedes what is currently available. The process from design to the manufacture and implementation of the finished device and system created a superior, viable and cost-effective alternative without major impediments. On analysis of the project and the experience of the overall process, it could be proffered that a superficial adjustment for future

work would include, a minor adjustment to the positioning of the illumination LEDs. This would eradicate errors relating to the proximity of the lateral flow strips to the CMOS detector.

In the future, to improve the calibration of the system, more lateral flow strips with varying known concentrations could be used thus improving the accuracy of the system.

The advantage of such a system is its ability to be replicated for a multitude of lateral flow strips which detect a diverse range of analytes.

Additional areas of future research and adaptations could include the implementation of a machine learning algorithm whereby, previously known concentrations of analyte train (and test) a model which could be used to predict future lateral flow tests.

With reference to the web application, customisation features such as allowing the lateral flow developer to calibrate the thresholding technique (if different levels of gold colloid are used) as well as varying the distance between the control line and test line. This allows the lateral flow developer to use the system for other types of lateral flow strips in development.

Ultimately, the future evolution of the system could lead to a mass market Point-Of-Care diagnostic device which conforms with NICE (National Institute for Health and Care Excellence) guidelines thereby reducing waiting times and alleviating financial pressures within the NHS.

## References

- [1] nhs.uk. (2018). *Blood tests*. [online] Available at: <https://www.nhs.uk/conditions/blood-tests/> [Accessed 2 Sep. 2018].
- [2] Arlow, J. and Neustadt, I. (2013). *UML 2 and the unified process*. Upper Saddle River, NJ: Addison-Wesley.
- [3] Preece, J., Sharp, H. and Rogers, Y. (2015). *INTERACTION DESIGN - BEYOND HUMAN-COMPUTER INTERACTION*. 4th ed. Chichester: Wiley, p.332.
- [4] Schwaber, K. (2004). *Agile project management with Scrum*. Redmond, Wash.: Microsoft Press.
- [5] Yetisen, A., Akram, M. and Lowe, C. (2013). *Paper-based microfluidic point-of-care diagnostic devices*. *Lab on a Chip*, 13(12), p.2210.
- [6] Jacinto, M., Trabuco, J., Vu, B., (2018). *Enhancement of lateral flow assay performance by electromagnetic relocation of reporter particles*. *PLOS ONE*, 13(1), p.e0186782.
- [7] Fisher, A. and Sirk, S. (2003). *Development of a Quantum Dot-Lateral Flow Assay*. *Quantum Dot Corporation*. [online] Available at: <https://trs.jpl.nasa.gov/bitstream/handle/2014/38284/03-2035.pdf?sequence=1> [Accessed 17 Aug. 2018].
- [8] ESEQuant *Lateral Flow Reader* - QIAGEN. [online] Available at: <https://www.qiagen.com/kr/products/custom-solutions/automation/ese-instruments/esequant-lateral-flow-reader/> [Accessed 23 Aug. 2018].
- [9] Idetekt.com. (2018). *Reader Systems* / *iDetekt.com*. [online] Available at: <http://idetekt.com/reader-systems/> [Accessed 23 Aug. 2018].
- [10] Zangheri, M., Cevenini, L., Anfossi, L., Baggiani, C., Simoni, P., Di Nardo, F. and Roda, A. (2015). *A simple and compact smartphone accessory for quantitative chemiluminescence-based lateral flow immunoassay for salivary cortisol detection*. *Biosensors and Bioelectronics*, 64, pp.63-68.
- [11] Lindbloom, B. (2018). *Uniform Perceptual Lab*. [online] [Brucelindbloom.com](http://www.brucelindbloom.com). Available at: <http://www.brucelindbloom.com/index.html?UPLab.html> [Accessed 27 Aug. 2018].

- [12] Singh, B., Parwate, D. and Shukla, S. (2009). *Radiosterilization of Fluoroquinolones and Cephalosporins: Assessment of Radiation Damage on Antibiotics by Changes in Optical Property and Colorimetric Parameters*. AAPS PharmSciTech, 10(1), pp.34-43.
- [13] Clegg, D. and Barker, R. (1994). *Fast-track*. Wokingham, England: Addison-Wesley Pub. Co.
- [14] Docs.aws.amazon.com. (2018). *Amazon Machine Images (AMI)* - Amazon Elastic Compute Cloud. [online]
- [15] Jackson, (2018). *Arduino Based Camera*. [online] Arduino Based Camera. Available at: <http://www.arducam.com/> [Accessed 23 Aug. 2018].
- [16] SortSite, *Website Error Checker: Accessibility & Link Checker - SortSite*. [online] Powermapper.com. Available at: <https://www.powermapper.com/products/sortsite/> [Accessed 3 Sep. 2018].
- [17] Rosebrock, A. (2018). *Finding extreme points in contours with OpenCV - PyImageSearch*. [online] PyImageSearch. Available at: <https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/> [Accessed 5 Sep. 2018].
- [18] W3schools.com. (2018). *W3Schools Online Web Tutorials*. [online] Available at: <https://www.w3schools.com/> [Accessed 5 Sep. 2018].
- [19] Nielson, Jakob (2005) *Ten Usability Heuristics*, [http://www.useit.com/papers/heuristic/heuristic\\_list.html](http://www.useit.com/papers/heuristic/heuristic_list.html)
- [21] Preece, J., Sharp, H. and Rogers, Y. (2015). *Interaction Design – Beyond Human Computer Interaction*. 4th ed. Chichester: Wiley
- [22] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994). *Design patterns*. 1st ed. Addison Wesley.
- [23] Middlecamp, D. (2018). *Sending Photos and Video Over the Internet!*. [online] Hackster.io. Available at: <https://www.hackster.io/middleca/sending-photos-and-video-over-the-internet-db583f> [Accessed 5 Sep. 2018].
- [24] Tod, R. (2018). *Tutorial: Creating and managing a Node.js server on AWS, part 1*. [online] Hacker Noon. Available at: <https://hackernoon.com/tutorial-creating-and-managing-a-node-js-server-on-aws-part-1-d67367ac5171> [Accessed 5 Sep. 2018].



## Appendix A

### System Manual

#### Photon Firmware

Within the Firmware directory contains the firmware for the Rapsens' Photon Particle microcontroller. If another TCP server is created, change lines 26 and 28 within the firmware file (main.ino) to alter the destination IP address and port number of the TCP packets. The Arducam library files are located in the folder 'libraries'.

After altering the photon's firmware, the script must be built and flashed using the shell:

**build:** *particle flash Photon tcp\_photographer.bin*

**flash:** *particle compile photon temp\_build --saveTo tcp\_photographer.bin*

#### Virtual Machine

The Virtual Machine used within this project was Ubuntu 16.04 LTS. To access the files within Amazon Web Services's EC2 instance, use a secure shell client with the private key file (keypairec2180718.pem). Alternatively, the user can insert the private-key file into the C:/ssh directory on their local machine and type the follow command into the command prompt:

```
ssh -i "keypairec2180718.pem" ubuntu@ec2-54-85-56-7.compute-1.amazonaws.com
```

The following packages and libraries must be installed using the Ubuntu shell in order for the application to work:

<b>Git</b>	<i>apt-get install git</i>
<b>Apache Server</b>	<i>apt-get install apache2</i>
<b>MySQL</b>	<i>apt-get install mysql-server</i>
<b>PHP</b>	<i>apt-get install php</i>
<b>Pip</b>	<i>apt-get install python-pip</i>
<b>Python 2.7</b>	<i>apt install python2.7 python-pip</i>
<b>OpenCV</b>	<i>pip install opencv-python</i>
<b>Numpy</b>	<i>pip install --user numpy</i>

**Node.js** *apt-get install -y nodejs*

Insert the 'CloudDirectory' directory into the Ubuntu Virtual Machine directory: /var/www/html.

To start the TCP server within the Virtual Machine, navigate to the directory containing the main.js Node.js script (/var/www/html/server/) and in the shell type: *node main.js*

If successful, a shell will display and the console log 'Listening for TCP packets on port 5550' will appear. Also, if the photon has successfully connected to the Node.js script, the console will read 'Connection Received'. If the connection is unsuccessful, check the publish events on the Particle's Device Cloud (<https://console.particle.io>).

### Relational Database

The relational database (engine: MySQL) used within this project contained two tables, 'user' and 'imageAnalysis'. There is no relationship between each table. The imageAnalysis table is used for storing results generated by the image analysis python scripts whilst the user table stores username-password key pairs.

imageAnalysis table:

<b>image</b>	varchar(40)
<b>testCoord</b>	varchar(25)
<b>contCoord</b>	varchar(25)
<b>testMeanGray</b>	float
<b>contMeanGray</b>	float
<b>testOcont</b>	float
<b>contOtest</b>	float
<b>concentration</b>	float

user table:

<b>username</b>	varchar(30)
<b>password</b>	varchar(30)
<b>confirmation</b>	int(1)
<b>email</b>	varchar(30)

## Appendix B

### User Manual

#### Taking a reading using the Rapsens reader

1. Ensure the Photon Particle is connected to power. This can be done by inserting a micro-usb directly into the Photon or connecting a 3.3 V battery into the battery shield.
2. If connection to cloud is successful, the photon's onboard LED will 'breath' cyan. If there is connectivity issues, the blue led will turn on and the user is prompted to the event messages within the Photon Particle Device Cloud. Most likely cases for this occurring will be: node.js TCP server is not running or the Photon is not connected to WiFi. Ensure the photon is connected to a WiFi connection which is broadcasting at 2.4GHz.
3. If all external LEDs are off, insert the lateral flow strip into the Rapsens reader.
4. Click the push button switch on the lid.
5. The green LED will illuminate and stay on. This shows that the image is being sent to the cloud. Uploads usually last approximately 5 seconds and once the image has been uploaded, the green LED will turn off. However, if the green LED turns on and then flashes three times, this indicates that there is no strip inserted into the reader and image upload will be canceled until the button is pressed again (with a strip inserted).

#### Analysing image data sent from the reader

1. Type <http://54.85.56.7> into your browser. You will be directed to the Rapsens login page. On this page, you can login or register for an account.
2. Once logged in, the gallery page will be displayed (index.php). Within the gallery page, images are displayed (oldest first) in a five column grid. Underneath each image, the time and date of image capture is displayed as well as the analysed concentration, if calculated (the concentration will display 'Pending').
3. To analyse a particular image, double click an image. The image will enlarge. To manually draw regions of interest around the control line and test line: click in the left mouse button and drag the mouse. If the user is satisfied with region of interest box, release the left click on the mouse. Repeat for the test line. Press the 'Calculate' button.
4. The gallery page will be displayed again and the concentration of the previously selected image will have updated.
5. Alternatively, the user can allow the web application to automatically detect the regions of interest. To accomplish this, double click a particular image and click the 'Auto

Calculate' button. Similarly to bullet point 4, the gallery page will be displayed again and the concentration of the previously selected image will have updated.

6. To view the automatically detected region of interest, click the 'Auto detect Images' link at the top of the page.
7. A similarly styled grid as seen in the gallery page will be displayed however regions of interest are drawn onto each analysed image.
8. To view all analysis results in a table format, click 'Table' at the top of the page. Within this page, the table can be downloaded to the user's local machine in a .csv file format.

## Appendix C

### Supporting documentation

#### Use cases

<b>Name</b>	TakeReading
<b>ID</b>	UC01
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	None
<b>Main Flow</b>	<ol style="list-style-type: none"><li>1. The use case starts when the Developer removes packaging from Lateral Flow Strip.</li><li>2. The Developer applies the specimen onto the sample pad of the Lateral Flow Strip.</li><li>3. The Developer holds the Lateral Flow Strip by the green holder and insert it into the reader.</li><li>4. Ensuring that none of the waning LEDs are turned on, the Developer pushes the button on the lid of the reader.</li><li>5. Ensuring that none of the waning LEDs are turned on, the Developer pushes the button on the lid of the reader.</li></ol>
<b>Post Conditions</b>	The System responds turning on the green LED, informing the Developer that the Reader is analysing the Lateral Flow Strip. The data sent from the reader will be stored in a remote server.
<b>Alternative Flows</b>	If the System indicates that the blue LED is illuminated, this indicates that there is a connectivity issue due to an unstable or weak internet connection.

<b>Name</b>	Register
<b>ID</b>	UC02
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	None
<b>Main Flow</b>	<ol style="list-style-type: none"><li>1. The use case starts when the Developer inserts <a href="http://54.85.56.7">http://54.85.56.7</a> into web browsers address bar</li><li>2. The Developer presses Sign Up</li><li>3. The System responds by opening the Registration Form Page</li></ol>

4. The Developer inserts their information into the presented forms
5. The Developer presses the Sign Up button

**Post Conditions** A Developer account has been set up

**Alternative Flows** If the Developer inserts a username/email that has already been taken or the same password has not been re-entered correctly, the System presents an alert prompting the Developer of their error.

**Name** Login

**ID** UC03

**Primary Actors** Developer

**Secondary Actors** None

**Preconditions** UC02: Registration

**Main Flow**

1. The use case starts when the Developer inserts <http://54.85.56.7> into web browsers address bar
2. The Developer inserts their information into the username and password fields
3. Press the Login button
4. The System conducts validation checks

**Post Conditions** The System will direct the Developer to the List of Lateral Flow Images if correct information is entered.

**Alternative Flows** If Developer enters details incorrectly, the System presents an alert prompting the Developer of their error.

**Name** LogOut

**ID** UC04

**Primary Actors** Developer

**Secondary Actors** None

**Preconditions** UC03: Login

**Main Flow**

1. The use case starts when the Developer clicks Logout

**Post Conditions** The System will direct the Developer to the Login page

**Alternative Flows** None

**Name** ManualAnalysis

<b>ID</b>	UC05
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	UC03: Login
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The use case starts when the Developer clicks an image which they want to analyse</li> <li>2. The System will enlarge the image</li> <li>3. The Developer will draw a region of interest around the control line</li> <li>4. The System will display coordinates of the region of interest around the control line</li> <li>5. The Developer will draw a region of interest around the test line</li> <li>6. The System will display coordinates of the region of interest around the test line</li> <li>7. The Developer will click Calculate</li> </ol>
<b>Post Conditions</b>	The System will produce a resultant concentration of the analyte and add result to a database
<b>Alternative Flows</b>	If the Developer incorrectly draws regions of interest, the System presents an alert prompting the Developer of their error.

<b>Name</b>	AutomaticAnalysis
<b>ID</b>	UC06
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	UC03: Login
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The use case starts when the Developer clicks an image which they want to analyse</li> <li>2. The System will enlarge the image</li> <li>3. The Developer will click Auto Calculate</li> </ol>
<b>Post Conditions</b>	The System will produce a resultant concentration of the analyte and add result to a database. An image displaying the automatically detected regions of interest will be recorded.
<b>Alternative Flows</b>	none

<b>Name</b>	ViewAutomaticAnalysisImages
-------------	-----------------------------

<b>ID</b>	UC07
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	UC03: Login
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The use case starts when the Developer clicks Auto Detect Images</li> <li>2. The System will display a list of images with Automatic region of interests markings on them</li> </ol>
<b>Post Conditions</b>	The System will display the Automatic Regions of Interest Images Page
<b>Alternative Flows</b>	none

<b>Name</b>	CheckHelpInformation
<b>ID</b>	UC08
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	UC03: Login
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The use case starts when the Developer clicks Information</li> </ol>
<b>Post Conditions</b>	The System will jump to the end of the page which will display helpful information of how to use the tools on the website
<b>Alternative Flows</b>	The Developer can manually scroll to the end of the page.

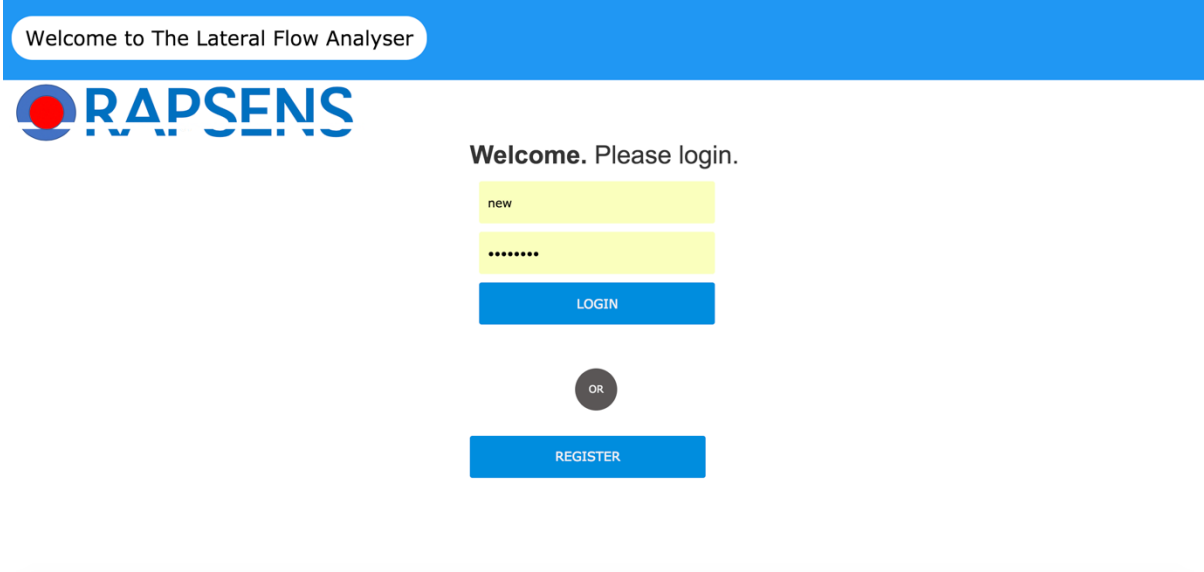
<b>Name</b>	ViewHistoryOfResults
<b>ID</b>	UC09
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	UC03: Login
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The use case starts when the Developer clicks History</li> <li>2. The System will display the web page which contains a table of past results</li> </ol>
<b>Post Conditions</b>	The User can then browse a table of a table containing a backlog of results which have been generate and saved from a database. The table can be sorted.



<b>Alternative Flows</b>	none
<b>Name</b>	DownloadCSVOfResults
<b>ID</b>	UC10
<b>Primary Actors</b>	Developer
<b>Secondary Actors</b>	None
<b>Preconditions</b>	UC09: ViewHistoryOfResults
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. The use case starts when the System displays the web page containing a table of past results</li> <li>2. The Developer will click 'Generate CSV file'</li> </ol>
<b>Post Conditions</b>	The System will jump to the end of the page which will display helpful information of how to use the tools on the website
<b>Alternative Flows</b>	The Developer can manually scroll to the end of the page.

## Screen Shots of the Web Application

### Login Page



Welcome to The Lateral Flow Analyser

**RAPSENS**

**Welcome. Please login.**

new

\*\*\*\*\*

LOGIN

OR

REGISTER

## Login Page with Modal

Welcome to The Lateral Flow Analyser

**Incorrect Login details** ✕

Please Re-enter your login details!

new

\*\*\*\*\*

LOGIN

OR

REGISTER

## Image Gallery Page

Welcome to The Lateral Flow Analyser

Auto detect Images Information Logout

Toolbox (Drag to move)

Auto Calculate

Selected Image

Control Coordinates

Test Coordinates

Calculate

<b>Time:</b> 21-12-41 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.829678 Concentration: 629.78mIU/ml	<b>Time:</b> 21-13-51 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.82824 Concentration: 615.402mIU/ml	<b>Time:</b> 21-14-44 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.829406 Concentration: 627.06mIU/ml	<b>Time:</b> 21-15-26 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.83294 Concentration: 662.404mIU/ml	<b>Time:</b> 21-16-12 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.826879 Concentration: 601.788mIU/ml
<b>Time:</b> 21-20-02 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.855028 Concentration: 883.284mIU/ml	<b>Time:</b> 21-20-44 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.861064 Concentration: 943.637mIU/ml	<b>Time:</b> 21-21-26 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.851059 Concentration: 843.588mIU/ml	<b>Time:</b> 21-22-10 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.857651 Concentration: 909.512mIU/ml	<b>Time:</b> 21-23-04 <b>Date:</b> 2018-08-25 <b>C/T:</b> 0.857408 Concentration: 907.079mIU/ml

## Selected Image Zoom with Manual ROI Selection

Welcome to The Lateral Flow Analyser

Auto detect Images Information Logout ✕

Toolbox (Drag to move)

2018-08-25\_21-12-41.jpg

Auto Calculate

Selected Image

2018-08-25\_21-12-41.jpg

Control Coordinates

1004,1488,228,340

Test Coordinates

1080,1088,244,256

Calculate

Time: 21-12-41  
Date: 2018-08-25  
C/T: 0.796236  
Concentration:  
-104.636mIU/ml

Time: 21-16-12  
Date: 2018-08-25  
C/T: 0.826879  
Concentration:  
601.788mIU/ml

Time: 21-20-02  
Date: 2018-08-25  
C/T: 0.855028  
Concentration:  
883.284mIU/ml

Time: 21-23-04  
Date: 2018-08-25  
C/T: 0.857408  
Concentration:  
907.079mIU/ml

Time: 21-12-41  
Date: 2018-08-25

## Automatic Detection ROI Page

Welcome to The Lateral Flow Analyser
Table
Gallery
Auto detect Images
Information
Logout

Time: 21-12-41  
Date: 2018-08-25  
C/T: 0.829678  
Concentration: 629.78mIU/ml

Time: 21-13-51  
Date: 2018-08-25  
C/T: 0.82824  
Concentration: 615.402mIU/ml

Time: 21-14-44  
Date: 2018-08-25  
C/T: 0.829406  
Concentration: 627.06mIU/ml

Time: 21-15-26  
Date: 2018-08-25  
C/T: 0.83294  
Concentration: 662.404mIU/ml

Time: 21-16-12  
Date: 2018-08-25  
C/T: 0.826879  
Concentration: 601.788mIU/ml

## Table View Page of Image Analysis

Welcome to The Lateral Flow Analyser
Table
Gallery
Auto detect Images
Information
Logout

Show 10 entries
Search:

Image	Test Coordinates	Control Coordinates	Test Mean Grey Intensity	Control Mean Grey Intensity	Ratio: Test/Control	Ratio: Control/Test	Concentration mIU/ml	Days Pregnant
2018-08-25_21-12-41.jpg <a href="#">View</a>	1085, 1334, 1254, 1436	1085, 1334, 193, 375	176.122	146.124	1.20529	0.829678	629.78	0
2018-08-25_21-13-51.jpg <a href="#">View</a>	1085, 1334, 1252, 1434	1085, 1334, 191, 373	176.258	145.984	1.20738	0.82824	615.402	0
2018-08-25_21-14-44.jpg <a href="#">View</a>	1085, 1334, 1252, 1434	1085, 1334, 191, 373	175.997	145.973	1.20568	0.829406	627.06	0
2018-08-25_21-15-26.jpg <a href="#">View</a>	1085, 1334, 1255, 1441	1085, 1334, 194, 380	176.088	146.671	1.20057	0.83294	662.404	0
2018-08-25_21-16-12.jpg <a href="#">View</a>	1085, 1334, 1259, 1438	1085, 1334, 198, 377	176.283	145.765	1.20937	0.826879	601.788	0
2018-08-25_21-20-02.jpg <a href="#">View</a>	1040,1528,1384,1520	1032,1488,304,412	158.705	129.995	1.22085	0.819098	523.983	0
2018-08-25_21-20-44.jpg <a href="#">View</a>	1068,1468,1388,1496	1044,1452,304,416	160.935	131.854	1.22055	0.819301	526.009	0

Selerium IDE test.

A complete walkthrough of the web application was successfully completed with no errors and the desired result was displayed.

1. Running 'Complete Walkthrough'
2. 1.open on /login.php... OK
3. 2.click on css=a > #submit... OK
4. 3.click on name=username... OK
5. 4.type on name=username with value Newuser... OK
6. 5.click on css=strong > strong... OK
7. 6.click on name=password... OK
8. 7.type on name=password with value Newuser... OK
9. 8.type on name=confirmpassword with value Newuser... OK
10. 9.type on name=email\_address with value newuser@outlook.com... OK
11. 10.click on id=submit... OK

12. 11.click on linkText=Back to Login... OK
13. 12.click on id=username... OK
14. 13.type on id=username with value Newuser... OK
15. 14.type on id=password with value newuser... OK
16. 15.click on id=submit... OK
17. 16.click on id=0... OK
18. 17.click on id=0... OK
19. 18.doubleClick on id=0... OK
20. 19.mouseDownAt on id=canvas with value 364.3999938964844,81... OK
21. 20.mouseMoveAt on id=canvas with value 364.3999938964844,81... OK
22. 21.mouseUpAt on id=canvas with value 364.3999938964844,81... OK
23. 22.click on id=canvas... OK
24. 23.mouseDownAt on id=canvas with value 373.3999938964844,353... OK
25. 24.mouseMoveAt on id=canvas with value 373.3999938964844,353... OK
26. 25.mouseUpAt on id=canvas with value 373.3999938964844,353... OK
27. 26.click on id=canvas... OK
28. 27.click on css=form[name="formB"] > #submit... OK
29. 28.click on id=0... OK
30. 29.click on id=0... OK
31. 30.doubleClick on id=0... OK
32. 31.click on id=submit... OK
33. 32.click on linkText=Auto detect Images... OK
34. 33.click on linkText=Table... OK
35. 34.click on id=submit... OK
36. 35.click on linkText=Logout... OK
37. **'Automatic Image Analysis' completed successfully**

## Appendix D

### Code Listings

This section contains the primary PHP, Python, Node.js and Photon Firmware (.ino) files which were used in the creation of this project. The complete set of code used will be handed in separately. The index.php page contains the most back-end logic in comparison to the other PHP files.

*Photon Particle Firmware – main.ino*

```
/*
Photon Particle firmware for the Quick Check Lateral Flow Immunoassay.
This firmware is used in conjunction with:
    AWS EC2 Instance IP:      54.85.56.7
    AWS EC2 Instance Port:    5550
    Arducam Camera Module:    OV5640 5MP MINI PLUS
    Image Resolution:         2592x1944px
Images are captured and are sent in TCP packets to a recipient Node.js script
running in AWS.
Due to the Photon's memory capacity, one image must be sent in multipled TCP
packets.
Open-source camera libraries and example scripts are published by
www.arducam.com.

@modified    28/08/18
@author:     Donal McLaughlin
*/

#include "ArduCAM.h"
#include "memorysaver.h"
SYSTEM_THREAD(ENABLED);

#define VERSION_SLUG "7n"

//Connect to TCP Client Server on PC
TCPClient client;

//Elastic IP for EC2 instance containing Node.js server
#define SERVER_ADDRESS "54.85.56.7"
//Node.js listens at port 5550
#define SERVER_TCP_PORT 5550
//1024
#define TX_BUFFER_MAX 4096

uint8_t buffer[TX_BUFFER_MAX + 1];
```

```

int tx_buffer_index = 0;
int led = D7;           //Onboard led
int ledouter = A0;      //LED outer, on if image capturing is in progress
int button = D4;        //Push Button Switch
int illlled1 = D3;      //Illumination LED 1
int illlled2 = D2;      //Illumination LED 2
int conled = A1;        //Connection LED - On if cant connect to TCP server
int buttonpush = 0;     //Button switch - 0 when off - 1 when pushed

// set pin A2 as the slave select for the ArduCAM shield
const int SPI_CS = A2;
// camera module used: OV5640
ArduCAM myCAM(OV5640, SPI_CS);

//-----SETUP-----//
void setup()
{
    // Set pin modes for all the LEDs and PBS
    pinMode(led, OUTPUT);
    pinMode(ledouter, OUTPUT);
    pinMode(illlled1, OUTPUT);
    pinMode(illlled2, OUTPUT);
    pinMode(conled, OUTPUT);
    pinMode(button, INPUT_PULLUP);

    Particle.publish("status", "Hello, Version: " + String(VERSION_SLUG));
    delay(1000);

    uint8_t vid,pid;
    uint8_t temp;

    //Set baseline frequency for communication between I2C slave devices
    Wire.setSpeed(CLOCK_SPEED_100KHZ);
    Wire.begin();

    Serial.begin(115200);
    Serial.println("ArduCAM camera starting!");

    // set the SPI_CS as an output:
    pinMode(SPI_CS, OUTPUT);

    // Initialise Serial Peripheral Interface (SPI)
    //Outputs: SCK, MOSI
    //Pull low: SCK, MOSI
    SPI.begin();

    while(1) {

```

```

        //Check for CMOS sensor, events published to particle cloud and
console
        Particle.publish("status", "checking for camera");
        Serial.println("Checking for camera...");

        //Check if the ArduCAM SPI bus is OK
        myCAM.write_reg(ARDUCHIP_TEST1, 0x55);
        temp = myCAM.read_reg(ARDUCHIP_TEST1);
        if(temp != 0x55){
            Serial.println("SPI interface Error!");
            Serial.println("myCam.read_reg: " + String(temp));
            delay(5000);
        }
        else {
            break;
        }
        Particle.process();
    }

    Particle.publish("status", "Camera found.");

    while(1){
        //Check if the camera module type is OV5640
        myCAM.rdSensorReg16_8(OV5640_CHIPID_HIGH, &vid);
        myCAM.rdSensorReg16_8(OV5640_CHIPID_LOW, &pid);

        Serial.println(F("OV5640 detected."));
        Particle.publish("status", "OV5640 detected: " +
String::format("%d:%d", vid, pid));
        break;
    }

    Serial.println("Camera found, initializing...");

    //Change MCU mode of the sensor (RGB or JPEG MODE)
    myCAM.set_format(JPEG);
    delay(100);
    //initialise hardware information of the photon (SPI chip select port and
//image sensor slave address)
    myCAM.InitCAM();
    delay(100);
    // Close auto exposure mode
    uint8_t _x3503;
    myCAM.wrSensorReg16_8(0x5001, _x3503|0x01);
    //Set exposure value
    myCAM.wrSensorReg16_8(0x3500, 0x00);

```

```

myCAM.wrSensorReg16_8(0x3501,0x79);
myCAM.wrSensorReg16_8(0x3502,0xe0);

myCAM.set_bit(ARDUCHIP_TIM, VSYNC_LEVEL_MASK);
delay(100);
//Clear flag after one image is buffed to photon memory
myCAM.clear_fifo_flag();
delay(100);

myCAM.write_reg(ARDUCHIP_FRAMES,0x00);
delay(100);
//Set JPEG resolution
myCAM.OV5640_set_JPEG_size(OV5640_320x240);

// wait 1 second
delay(1000);

//Connect to Node.js server in AWS EC2 instance
client.connect(SERVER_ADDRESS, SERVER_TCP_PORT);
}

//-----LOOP-----//

void loop()
{
    //Cannot connect to client: Blue LED turns on and message sent to
    Particle cloud
    //Repeated until TCP server is found
    if (!client.connected()) {
        //client.stop();
        Particle.publish("status", "Reconnect to TCP Server...");
        if (!client.connect(SERVER_ADDRESS, SERVER_TCP_PORT)) {
            digitalWrite(conled, HIGH);
            delay(1000);
            return;
        }
    }
    digitalWrite(conled, LOW);
    int buttonState = digitalRead(button);
    int imgcount = 0;
    if (buttonState == LOW){
        buttonpush = 0;
        digitalWrite(ledouter, HIGH);
        analogWrite(illlled1, 1);
        analogWrite(illlled2, 1);

        Particle.publish("status", "Taking a picture...");
        Serial.println("Taking a picture...");
    }
}

```



```

digitalWrite(led, HIGH);

//Set image resolution at max resolution at: 2592x1944
myCAM.OV5640_set_JPEG_size(OV5640_2592x1944); //works
//-----
delay(100);

myCAM.flush_fifo();
delay(100);

myCAM.clear_fifo_flag();
delay(100);
//Capture and Store frame data on 8MB frame buffer
myCAM.start_capture();
delay(100);

unsigned long start_time = millis(),
              last_publish = millis();

/*
wait for the photo to be done
*/
while(!myCAM.get_bit(ARDUCHIP_TRIG , CAP_DONE_MASK)) {
    Particle.process();
    delay(10);

    unsigned long now = millis();
    if ((now - last_publish) > 1000) {
        Particle.publish("status", "waiting for photo " + String(now-
start_time));
        last_publish = now;
    }

    if ((now-start_time) > 30000) {
        Particle.publish("status", "bailing...");
        break;
    }
}
delay(100);

int length = myCAM.read_fifo_length();
Particle.publish("status", "Image size is " + String(length));
Serial.println("Image size is " + String(length));

uint8_t temp = 0xff, temp_last = 0;
int bytesRead = 0;

```

```

if(myCAM.get_bit(ARDUCHIP_TRIG, CAP_DONE_MASK))
{
    delay(100);
    Serial.println(F("Capture Done."));
    Particle.publish("status", "Capture done");

    if(length < 430000){
        Particle.publish("status", "No strip detected");
        digitalWrite(conled, HIGH);
        delay(1000)
        digitalWrite(conled, LOW);
        delay(1000)
        digitalWrite(conled, HIGH);
        delay(1000)
        digitalWrite(conled, LOW);
        break;
    }

    //read multiple bytes out of the frame buffer
    myCAM.set_fifo_burst();

    tx_buffer_index = 0;
    temp = 0;

    while( (temp != 0xD9) | (temp_last != 0xFF) )
    {
        temp_last = temp; // set last temp
        temp = myCAM.read_fifo(); // read data
        bytesRead++;

        buffer[tx_buffer_index++] = temp; //
        //index >= 4096
        if (tx_buffer_index >= TX_BUFFER_MAX) {
            //client.write(buffer array, length of buffer);
            client.write(buffer, tx_buffer_index);
            Particle.publish("Writing to server");

            tx_buffer_index = 0;

            Particle.process();
        }
        // If the image is larger than 2MB, stop sending to the server
        if (bytesRead > 2048000) {
            break;
        }
    }
}

```

```

    if (tx_buffer_index != 0) {
        client.write(buffer, tx_buffer_index);
        Particle.publish("Writing to server (small)");
    }

    //Clear the capture done flag
    myCAM.clear_fifo_flag();

    Serial.println(F("End of Photo"));
}

//Photo capture finish: turn off Green LED and illumination LEDs
digitalWrite(ledouter, LOW);
digitalWrite(led, LOW);
analogWrite(illlled1, 0);
analogWrite(illlled2, 0);
buttonState = HIGH;

}else{
    digitalWrite(ledouter, LOW);
}
}

```

*TCP Revicer Script (Node.js) – main.js*

```

/*
TCP Server File
This node.js script listens for TCP packets on port 5550 and pushes each
packet onto an array (buffer[]).
After 10seconds of inactivity, the buffer array is converted into a variable
and saved as a JPG within
the EC2 instance ../server/image/date-time.jpg.
@modified    28/08/18
@author:     Donal McLaughlin
*/

//File system module
var fs = require('fs');
//Provides an asynchronous network API for creating stream-based TCP or IPC
servers
var net = require('net');

var settings = {

```

```

    ip: "127.0.0.1",
    port: 5550
  };

var moment = require('moment');

var buffer = [];
var bufferTimer = null;
var bufferDelay = 10000;

/*
If TCP packet transmission complete, add to Virtual Machine
*/
var saveImage = function() {
  //save image name as DateAndTime.jpg
  var datetime = new Date().toISOString().replace(/T/, '_').      //
  replace T with a space
  replace(/\.+/ , '').
  replace(/:/, '-').replace(/:/, '-')
  console.log("saving image");
  var data = Buffer.concat(buffer);
  var filename = "images/" + datetime + ".jpg";
  console.log("writing " + data.length + " bytes to file");
  //Write File.jpg
  fs.writeFileSync(filename, data);
  //Image transfer complete, reset buffer array for next image.
  buffer = [];
}

//If inactivity greater than 10s (i.e. finished transmission of TCP packets),
save image
var delayBufferFn = function(section) {
  if (bufferTimer) {
    clearTimeout(bufferTimer);
  }
  //push section of image data onto the buffer array
  buffer.push(section);
  //wait 10seconds, interrupted if new packet is added to buffer[]
  bufferTimer = setTimeout(saveImage, bufferDelay);
}

//Create the TCP server
var server = net.createServer(function (socket) {
  //Connection between photon and TCP server successful
  console.log("Connection received");

  socket.on('readable', function() {
    //Read TCP packet (a section of the image data)

```

```

        var data = socket.read();
        //Add to buffer array and await end of transmission from photon
        delayBufferFn(data);
    });
});
//Console output for listening to port 5550
server.listen(settings.port, function () {
    console.log('Listening for TCP packets on port ' + settings.port + '
...');
});
});

```

*Gallery Web Page – index.php*

```

<?php
include('template/header.php');
$dirname= "server/images/";
$images = glob($dirname."*.*");
$i=0;
$imgCountTable=0;
$selectedImage="";
$imgarray = array();
$contCoords = $testCoords = $output = "";
$cot = "";
$toc = "";
$conn = mysqli_connect("lateralflowinstance.c59wk2ctliok.us-east-
1.rds.amazonaws.com", "Nibec", "Nibec2018", "lateralflowschema");
//If no connection to database, display error message
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
//Two forms: Auto Detect (Case A) and Manual Detect (Case B)
if ($_SERVER['REQUEST_METHOD'] == 'POST'){
    if (isset($_POST["form"])) {
        switch ($_POST['form']){
            case "A":
                $imgName = $_POST['aimnamejstophp'];
                echo $imgName;
                $cmd = "python autoThresholding.py $imgName 2>&1";
                $output = shell_exec($cmd);
                break;
            case "B":
                $contCoords = $_POST['xcontCoords'];
                $testCoords = $_POST['ytestCoords'];
                $imgName = $_POST['imgnamejstophp'];
                $cmd = "python imageAnalysis.py $contCoords $testCoords
$imgName 2>&1";

```

```

        $output = shell_exec($cmd);
        break;
    }
}
}

//Generate a table of images
echo '<table id="table"><tr>';
    foreach($images as $image){
        //Time
        $baseNameNoType=strtotime(str_replace(".jpg", "", basename($image)));
        $newformat = date('H:i d/m/Y', $baseNameNoType);

        if(getimagesize($image) === false){
        }else{
            if($imgCountTable<5){
                $basename = basename($image);
                $pieces=explode("_", $basename);
                $recieveDate=$pieces[0];
                $recieveTimejpg=$pieces[1];
                $recieveTime=str_replace(".jpg", "", $recieveTimejpg);
                echo '<td><div class="w3-card w3-container" style="min-height:100%
min-width:100%"><div onClick="getID('.$i.')">Date: '.$recieveDate.'"
style="width:100%;max-width:300px" class="hover-shadow cursor"; cursor: zoom-
in></div>';
                echo "<b>Time:</b> <i>".$recieveTime."</i><br><b>Date:</b>
<i>".$recieveDate."</i><br>";
                //Print results under each photo
                echo "<b>C/T:</b> ";
                $sql = "SELECT contOtest, concentration FROM
imageAnalysis WHERE image='".$basename."'";
                $result = mysqli_query($conn, $sql);
                if (mysqli_num_rows($result) > 0) {
                    while($row = mysqli_fetch_assoc($result)) {
                        echo "<i>".$row['contOtest']."</i><br>";
                        echo "<b>Concentration:</b>
<i>".$row['concentration']."mlU/ml</i></td></div>";
                    }
                }else {
                    echo "<i>Pending</i></td></div>";
                }

                $imgarray[$i] = $basename;
                $imgCountTable++;
            }else{

```

```

        echo '</tr><tr>';
        $imgCountTable=0;
    }

}

    $i++;
}
echo '</tr></table>';
//End of table of images
?>

<!-- Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-123889074-1"></script>
<script>
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());
    gtag('config', 'UA-123889074-1');
</script>
<!-- END OF GOOGLE ANALYTICS -->

<!DOCTYPE html>
<html>
<title>Lateral Flow IOT Reader</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://www.w3schools.com/lib/w3-theme-black.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.3.0/css/font-awesome.min.css">
<link rel="stylesheet" href="style.css">
<!-- <script type="text/javascript" src="roiscript.js"></script> -->
<body>

<!-- Top of Page Button -->
<script src="buttons.js"></script>
<button onclick="topFunction()" id="topBtn" title="Go to top">Top</button>
<button onclick="window.location.reload()" id="refreshBtn" title="Refresh Page">Refresh</button>

<!-- //////////MODAL//////////////////////////////////////-->

<!-- HTML MODAL TEMPLATE -->
<div id="myModal" class="modal" scroll="no" style="overflow: hidden">
    <span class="close">&times;</span>

```

```

        <canvas id="canvas" class="modal-content" style="cursor:
crosshair"></canvas>
    <div id="caption"></div>
</div>

<?php $js_imgarray = json_encode($imgarray); ?>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></scri
pt>
<script>

clickedID=0;

function getID(clickedID){
    // Get the modal
    var modal = document.getElementById('myModal');

    //Array
    var jsarrayimg = <?php echo $js_imgarray; ?>;
    var imageName = jsarrayimg[clickedID];
    var canvas = document.getElementById('canvas');
    var ctx = canvas.getContext('2d');

var drag = false;
var imageObj = null;
var testControlSwitch = 0; // Control Line = 0, Test Line = 1
var conX1 = 0, //Control Coordinates
    conX2 = 0,
    conY1 = 0,
    conY2 = 0,
    testX1 = 0, //Test Coordinates
    testX2 = 0,
    testY1 = 0,
    testY2 = 0,
    X1 = 0, //General Coordinates
    X2 = 0,
    Y1 = 0,
    Y2 = 0;
    clickedImg="/server/images/"+imageName;

    // Get the image and insert it inside the modal - use its "alt" text as a
caption
    var img = document.getElementById(clickedID); //selected img

    var captionText = document.getElementById("caption");

```



```

img.onclick = function(){
    modalImg = new Image();
    modalImg.onload = function () { ctx.drawImage(modalImg, 0, 0,
ctx.canvas.width, ctx.canvas.height); };
    modalImg.src = "/server/images/"+imageName;
    ctx.canvas.width = modalImg.naturalWidth/4;
    ctx.canvas.height = modalImg.naturalHeight/4;
    canvas.addEventListener('mousedown', mouseDown, false);
    canvas.addEventListener('mouseup', mouseUp, false);
    canvas.addEventListener('mousemove', mouseMove, false);
    modal.style.display = "block";
    captionText.innerHTML = this.alt;
    document.getElementById("iimgnamejstophp").innerHTML = imageName;
    document.getElementById("aiimgnamejstophp").innerHTML = imageName;
    document.getElementById("aimgnamejstophp").innerHTML = imageName;
    document.getElementById("aimgnamejstophp").value = imageName;

}

// Get the <span> element that closes the modal
var span = document.getElementsByClassName("close")[0];

// When the user clicks on <span> (x), close the modal
span.onclick = function() {
    modal.style.display = "none";
    document.getElementById("iimgnamejstophp").innerHTML = "Select an
Image";
}

function mouseDown(e) {
    rect.startX = e.pageX - this.offsetLeft;
    rect.startY = e.pageY - this.offsetTop;
    drag = true;
}

function mouseUp() {
    drag = false;
    if(testControlSwitch == 0){
        //Control Line
        console.log("Control");
        conX1=rect.startX;
        conX1=X1;
        conX2=X2;
        conY1=Y1;
        conY2=Y2;
        document.getElementById("contCoords").innerHTML = conX1 + "," + conX2
+ "," + conY1 + "," + conY2;
    }
}

```

```

        addToForm(conX1 + "," + conX2 + "," + conY1 + "," + conY2);
        console.log(conX1, conX2, conY1, conY2);
        testControlSwitch++; //Switch
    }else{
        //Test Line
        console.log("Test");
        testX1=X1;
        testX2=X2;
        testY1=Y1;
        testY2=Y2;
        console.log(testX1, testX2, testY1, testY2);
        document.getElementById("testCoords").innerHTML = testX1 + "," +
testX2 + "," + testY1 + "," + testY2;
        addToForm(testX1 + "," + testX2 + "," + testY1 + "," + testY2);
        testControlSwitch=0; //Switch
        //Final Coordinates for submission
        coordinates = [{"conX1": conX1, "conX2": conX2, "conY1": conY1,
"conY2": conY2, "testX1": testX1, "testX2": testX2, "testY1": testY1,
"testY2": testY2 , "imageName": imageName}];
        console.log("Coordinates");
        console.log(coordinates);
    }
}

function addToForm(inputCoords) {

    if(testControlSwitch == 0){

        var element = document.getElementById("xcoordid");
        if (element!=null){
            element.parentNode.removeChild(element);
        }

        var x = document.createElement("INPUT");
        x.setAttribute("type", "text");
        x.setAttribute("name", "xcontCoords");
        x.setAttribute("id", "xcoordid");
        x.setAttribute("value", inputCoords);

        document.body.appendChild(x);

    }else{
        var element = document.getElementById("ycoordid");
        if (element!=null){
            element.parentNode.removeChild(element);
        }
    }
}

```

```

        var y = document.createElement("INPUT");
        y.setAttribute("type", "text");
        y.setAttribute("name", "ytestCoords");
        y.setAttribute("id", "ycoordid");
        y.setAttribute("value", inputCoords);

        document.body.appendChild(y);
    }
}

function mouseMove(e) {
    if (drag) {
        if(testControlSwitch==1){

        }
        ctx.clearRect(0, 0, 500, 500);
        ctx.drawImage(modalImg, 0, 0, ctx.canvas.width, ctx.canvas.height);
        rect.w = (e.pageX - this.offsetLeft) - rect.startX;
        rect.h = (e.pageY - this.offsetTop) - rect.startY;
        ctx.strokeStyle = 'red';
        ctx.strokeRect(rect.startX, rect.startY, rect.w, rect.h);
        X1=rect.startX*4;
        Y1=rect.startY*4;
        X2=(rect.w + rect.startX)*4;
        Y2=(rect.h + rect.startY)*4;

        widthOfCont = rect.w;
        heightOfCont = rect.y;

        if(testControlSwitch == 0){
            document.getElementById("contCoords").innerHTML = X1 + "," + X2 +
            "," + Y1 + "," + Y2;
            document.getElementById("xcontCoords").value = X1 + "," + X2 +
            "," + Y1 + "," + Y2;

            document.getElementById("iimgnamejstophp").innerHTML = imageName;
            document.getElementById("imgnamejstophp").value = modalImg.src;

        }else{
            document.getElementById("testCoords").innerHTML = X1 + "," + X2 +
            "," + Y1 + "," + Y2;
            document.getElementById("ytestCoords").value = X1 + "," + X2 +
            "," + Y1 + "," + Y2;

            document.getElementById("iimgnamejstophp").innerHTML = imageName;
            document.getElementById("imgnamejstophp").value = modalImg.src;
        }
    }
}

```

```

    }

    }
}

} //End of getID()

</script>

<!-- TOOLBOX -->
<script>
$.getScript("draggableToolbox.js", function() {
    dragElement(document.getElementById("movable"));
});
</script>

<div id="movable">
    <div id="mydivheader">Toolbox (Drag to move)</div>

    <form name="formA" class="form" action="<?php echo
$_SERVER["PHP_SELF"];?>" method="POST" enctype="multipart/form-data"
autocomplete="off">
        <input type="hidden" name="form" value="A">
        <p id="aimgnamejstophp" name="aimgnamejstophp"><br></p>
        <input type="hidden" name="aimgnamejstophp" id="aimgnamejstophp"
required></input>
        <input type="submit" name="submit" id="submit" class="button"
value="Auto Calculate"></input>
    </form>

    <!--      Manual Form      -->
    <form name="formB" class="form" action="<?php echo
$_SERVER["PHP_SELF"];?>" method="POST" enctype="multipart/form-data"
autocomplete="off">

        <input type="hidden" name="form" value="B">
        <p><br><b>Selected Image</b> </p>
        <i>
        <p id="iimgnamejstophp" name="imgnamejstophp"><br></p>
        <input type="hidden" name="imgnamejstophp"
id="imgnamejstophp"></input>
        </i>

        <p><b>Control Coordinates</b></p>

        <i>
        <p id="contCoords" name="contCoords"><br></p>
        <input type="hidden" name="xcontCoords" id="xcontCoords"
required></input>

```

```

</i>
        <p><b>Test Coordinates</b></p>

        <i>
            <p id="testCoords" name="testCoords"><br></p>
            <input type="hidden" name="ytestCoords" id="ytestCoords"
required></input><br>
        </i>
        <input type="submit" name="submit" id="submit" class="button"
value="Calculate"></input>
    </form>
    <div class=results>
        <!--?php echo $output; ?-->
    </div>
</div>

</body>
</html>
<?php include('template/footer.php'); ?>

```

*Automatically Detect ROI – autoThresholding.py*

```

# import the necessary packages
import imutils
import cv2
import numpy as np
import logging
import sys

logging.basicConfig(filename='logs.log',level=logging.DEBUG,
format='%(asctime)s %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')
logging.warning('Start: Python exectuting.....
%s', str(sys.argv))
#Distance to test line
distance = 1025

imageNameold = "server/images/" + str(sys.argv[1])

imageName = imageNameold.replace("http://54.85.56.7/", "")
noPathimage = imageName.replace("server/images/", "")

#Cropping coordinates for test image
conX1 = 1084
conX2 = 1336

```

```

conY1 = 36
conY2 = 1836

class AutoThresholding(object):

    def __init__(self, imageName):
        self.imageName = imageName
        # load the image, convert it to grayscale, and blur it slightly

        uncroppedImage = cv2.imread(imageName) #

        image = uncroppedImage[int(conY1):int(conY2), int(conX1):int(conX2)]

    def formatImg(uncroppedImage):
        labuncropped = cv2.cvtColor(uncroppedImage, cv2.COLOR_BGR2LAB)
        lab = labuncropped[int(conY1):int(conY2), int(conX1):int(conX2)]

        lower_bound = np.array([77, 0 , 0])
        upper_bound = np.array([159, 255 , 255])
        thresh = cv2.inRange(lab, lower_bound, upper_bound)
        return thresh

    thresh = formatImg(uncroppedImage)

    def drawContours(thresh):
        # find contours in thresholded image, then choose the largest
        # one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                                cv2.CHAIN_APPROX_SIMPLE)
        cnts = cnts[0] if imutils.is_cv2() else cnts[1]
        c = max(cnts, key=cv2.contourArea)
        return c

    c = drawContours(thresh)

    # determine the most extreme points along the contour
    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])

    #adjust for cropping changes to the coordinates
    X1 = extLeft[0] + conX1
    X2 = extRight[0] + conX1
    autocontY1 = extTop[1] +conY1
    autocontY2 = extBot[1] +conY1
    autotestY1 = autocontY1 + distance + conY1

```

```

autotestY2 = autocontY2 + distance + conY1

autoContCordList = X1,X2,autocontY1,autocontY2
autoTestCordList = X1,X2,autotestY1,autotestY2

def formatLists(list):
    stringify = str(list)
    removeOpenBrack = stringify.replace("(", "")
    removeCloseBrack = removeOpenBrack.replace(")", "")
    return removeCloseBrack

autoContCordList = formatLists(autoContCordList)
autoTestCordList = formatLists(autoTestCordList)

from imageAnalysis import ManualDetect
ManualDetect(imageNameold, autoContCordList,autoTestCordList)

a = tuple((extLeft[0],extTop[1]+distance))
a1 = tuple((extRight[0],extTop[1]+distance))
b = tuple((extRight[0],extBot[1]+distance))
b1 = tuple((extLeft[0],extBot[1]+distance))
if autocontY1>750:
    autoTestCordList = autoContCordList
    autoContCordList = X1,X2,autocontY1-(distance*2),autocontY2-
(distance*2)

    autoContCordList = str(autoContCordList)

    autoContCordList = autoContCordList.replace("(", "")
    autoContCordList = autoContCordList.replace(")", "")
    a = tuple((extLeft[0],extTop[1]-distance))
    a1 = tuple((extRight[0],extTop[1]-distance))
    b = tuple((extRight[0],extBot[1]-distance))
    b1 = tuple((extLeft[0],extBot[1]-distance))

def saveAutoROIImg(image, c, extLeft, extRight,extTop, extBot,
a,a1,b,b1):
    # draw the outline of the object, then draw each of the
    # extreme points, where the left-most is red, right-most
    # is green, top-most is blue, and bottom-most is teal
    #Draw contours
    cv2.drawContours(image, [c], -1, (0, 255, 255), 2)
    #Control line contours
    cv2.circle(image, extLeft, 8, (0, 0, 255), -1)
    cv2.circle(image, extRight, 8, (0, 255, 0), -1)
    cv2.circle(image, extTop, 8, (255, 0, 0), -1)
    cv2.circle(image, extBot, 8, (255, 255, 0), -1)
    #Test line contours

```

```

cv2.circle(image, a, 8, (0, 0, 255), -1)
cv2.line(image, a, a1, (0, 255, 0), thickness=3, lineType=8)
cv2.circle(image, b, 8, (0, 255, 0), -1)
cv2.line(image, b1,b, (0, 255, 0), thickness=3, lineType=8)

# save the image with automatically drawn ROI
cv2.imwrite("autoImage/"+noPathimage, image)

saveAutoROIImg(image, c, extLeft, extRight,extTop, extBot, a,a1,b,b1)

imageObj = AutoThresholding(imageName)

```

*Image Analysis – imageAnalysis.py*

```

import sys
import logging
import cv2
import MySQLdb
#from skimage import io
import numpy as np
import mysql.connector

mydb = mysql.connector.connect(
host="lateralflowinstance.c59wk2ctliok.us-east-1.rds.amazonaws.com",
user="Nibec",
passwd="Nibec2018",
database="lateralflowschema"
)

logging.basicConfig(filename='logs.log',level=logging.DEBUG,
format='%(asctime)s %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')
logging.warning('Python exectuting %s', str(sys.argv))
contCord = ""
testCord = ""
imageNameold = ""

class ManualDetect(object):
    logging.debug('Enters Class')
    def __init__(self, imgarg, contCord, testCord):
        self.imgarg = imgarg
        self.contCord = contCord
        self.testCord = testCord
        ManualDetect.main(self, imgarg, contCord, testCord)

```



```

def main(self, imgarg, contCord, testCord):
    logging.debug('DEBUG: %s %s %s', self.imgarg, self.contCord,
self.testCord)
    imageName = imgarg.replace("http://54.85.56.7/", "")
    noPathimage = imageName.replace("server/images/", "")
    contCordList = contCord.split(",")
    testCordList = testCord.split(",")

    conX1 = contCordList[0]
    conX2 = contCordList[1]
    conY1 = contCordList[2]
    conY2 = contCordList[3]
    testX1 = testCordList[0]
    testX2 = testCordList[1]
    testY1 = testCordList[2]
    testY2 = testCordList[3]

    def formatImg(imageName):
        try:
            img = cv2.imread(imageName) #image
        except:
            result = "Can't find image!"
            logging.warning('Cant find image')

        try:
            img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            return img_gray          #convert image to greyscale
        except:
            logging.warning('Not Converted to BANDW')

    img_gray = formatImg(imageName)

    try:
        crop_img_cont = img_gray[int(conY1):int(conY2),
int(conX1):int(conX2)]
    except:
        result = "Can't Crop image - control line"
        logging.warning('Cant crop control')

    try:
        crop_img_test = img_gray[int(testY1):int(testY2),
int(testX1):int(testX2)]
    except:
        result = "Can't Crop image - test line"
        logging.warning('Cant crop test')

    try:
        meanGrayCont = cv2.mean(crop_img_cont)

```

```

        meanGrayCont = meanGrayCont[0]

        meanGrayTest = cv2.mean(crop_img_test)
        meanGrayTest = meanGrayTest[0]

        toc = meanGrayTest/meanGrayCont
        cot = meanGrayCont/meanGrayTest

        concentration = (cot - 0.7667)/0.0001    #Concentration of
Automatic detection
        #concentration = (cot - 0.717)/0.0001    #Concentration of
Manual detection
        timePregnant = 0
        logging.warning(concentration)

    except:
        logging.warning('Calculations failed')

    #try:
        var = (noPathimage, testCord, contCord,meanGrayTest,meanGrayCont,
toc, cot, concentration, timePregnant)
        var2 = (testCord, contCord,meanGrayTest,meanGrayCont, toc, cot,
concentration, timePregnant, noPathimage)
        def addToDatabase(var, var2):
            try:
                mycursor = mydb.cursor(buffered=True)

                mycursor.execute("SELECT image, COUNT(*) FROM
imageAnalysis WHERE image = %s GROUP BY image",(noPathimage,))
                row_count = mycursor.rowcount

                if row_count == 0:
                    logging.warning('No entry in db present, adding new
entry')

                    sql = "INSERT INTO imageAnalysis ( image, testCoord,
contCoord,testMeanGray,contMeanGray, testOcont, contOtest, concentration,
timePregnant) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s);"
                    var = (noPathimage, testCord,
contCord,meanGrayTest,meanGrayCont, toc, cot, concentration,
timePregnant)

                    mycursor.execute(sql, var)
                    mydb.commit()

            else:
                logging.warning('Entry Present, updating entry')

```

```

        mycursor.execute("UPDATE imageAnalysis SET
testCoord=%s, contCoord=%s,testMeanGray=%s,contMeanGray=%s, testOcont=%s,
contOtest=%s, concentration=%s, timePregnant=%s WHERE image = %s",var2)
        mydb.commit()
    except:
        result = "Error when inserting data into database"

    addToDatabase(var, var2)

logging.warning(imageNameold)
if len(sys.argv) > 2:
    logging.debug('More than two arguments')
    contCord = str(sys.argv[1]) # Control Coordinates
    testCord = str(sys.argv[2]) # Test Coordinates
    imageNameold = str(sys.argv[3]) # Image Name
    ManualDetect(imageNameold, contCord, testCord)

```