EEEE3001

Final Year Individual Project Dissertation

Compact high-efficiency illumination systems for in-orbit microscopy

AUTHOR:	**Mr Dean Morris**
ID NUMBER:	**20275270**
SUPERVISOR:	**Dr. Kevin Webb**
MODERATOR:	**Dr. Peter Christopher**
DATE:	**09 May 2024**

This third year project Dissertation is submitted in part fulfilment of the requirements of the degree of Bachelor of Engineering.

Contents

List of Figures

List of Tables

List of Equations

Glossary

Term	Definition
Caenorhabditis elegans (*C. elegans)*	Transparent nematode.
Low Earth Orbit (LEO)	An orbit close to Earth's surface typically less than 1000km in altitude.
Printed Circuit Board (PCB)	A method to connect components via copper conductors.
Message Queuing Telemetry Transport (MQTT)	Lightweight internet messaging protocol.
Graphical User Interface (GUI)	Allows a user to interact with a program in a visual way.
Light Emitting Diode (LED)	A semiconductor-based device that will emit light once current is ran through it.
Secure Copy Protocol (SCP)	A command line utility to securely transfer files from one device to another.
User Datagram Protocol (UDP)	An internet communication protocol.
Pulse Width Modulation (PWM)	A rectangular wave signal of certain frequency with varying duty cycle.
Single Board Computer (SBC)	Is an entire compute made on a single circuit board, including processor and memory.
General-Purpose Input/Output (GPIO)	A GPIO (General-Purpose Input/Output) is a versatile digital pin on an electronic circuit, like MCUs or MPUs, controllable by software to serve as either input, output, or both.
Back Focal Length (BFL)	A measurement of the last surface of a lens to its image plane.
Light Dependent Resistor (LDR)	Passive component that works via photoconductivity, meaning it decreases in resistance with increasing light intensity.

1.0 Introduction

Under micro gravity astronauts experience negative health consequences, including disruptions to their central nervous system, cardiovascular system, musculoskeletal atrophy, immune system dysfunction and increased intracranial pressure [2]. Investigating the underlying biological mechanisms causing these health consequences, requires a model system that can be economically placed in low earth orbit (LEO), within the size and weight requirements for a 1U CubeSat. Therefore, *C. elegans* are a suitable option as are a eukaryotic system with good genetic homology of 59% [3] with key systems in the human. Given the correct research environment, *Caenorhabditis elegans* will work well as the eukaryote for the experiments due to the similarities in their DNA to human DNA, and modest cultivation needs, such as low agar requirements for food and humidity. The life cycle of *C. elegans* is rapid and well-characterised, making it highly suitable for genetic studies. Its genome was fully sequenced in 1998 [4], establishing it as a canonical model in fields such as neuroscience, metabolic studies, and genetics. This extensive use in both low Earth orbit (LEO) and on Earth highlights its versatility and the broad applicability.

Hence an imaging system allows for improved understanding of the C. *elegans* biology by visualising growth in terms of size, volume, and behaviour over time. This requires microscopic imaging which is greatly improved through the use of contrast enhancement known as phase contrast [5].

Thus, to most effectively study this, my thesis provides a compact and cost-effective microscopy solution that utilises phase contrast imaging, to meet the requirements of operating experiments in LEO. The future of these experiments aims to allow better understanding of the impact that microgravity and radiation pose to human health, in terms of muscle loss, with the main importance of this research to be to decrease or eliminate muscle loss in astronauts and as a result allow astronauts more time to do experiments and research instead of spending a significant amount of time on counter measures to mitigate these negative health consequences. Therefore, making manned experiments in LEO more affordable due to being able to complete more work in the same period of time.

1.1 Aims

The primary objective of this thesis is to design and implement a compact, power-efficient, low-cost, high-efficiency prototype microscope, that will fit within a 1U CubeSat payload for the WormSail mission. All costs for the microscope solution should be £250 or less and provide flexibility in the design such that there is a degree of genericism, allowing the design to be applicable in a wide range of projects such as other CubeSat missions, projects with limited space (e.g. cell culture incubators), or even as a teaching tool due to the low-cost nature of the design.

1.2 Specification

- Design a custom PCB that will facilitate the illumination system comprising of 3 LED rings of colours red, blue, and green, with a central cyan LED.
- Choose components needed for the optical setup, including lenses and camera.
- Develop a C++ code to run on a Raspberry Pi 4B, to automatically take images, perform post processing and store the output to the Raspberry Pi's SD card.
- Design of custom enclosure for the microscope including ability to manually adjust the specimen height to allow for optimal positioning with respect to the lens configuration and camera choice.
- Integrate all sub-sections including the PCB illumination system, optical system, and software design.
- The final system should have the ability to take brightfield and darkfield images using the custom illumination PCB in a condenser free setup.
- ➢ The first stretch goal for the project is to provide a force analysis of the custom enclosure with the PCB design to ensure it would withstand launch forces.
- ➢ An additional stretch goal is to design a high level, easy to user graphical user interface to be able to control the microscope wirelessly using an internet messaging protocol called Message Queuing Telemetry Transport (MQTT).
- ➢ Final stretch goal is once the images have been taken and processed, automatically send back all images to the Linux PC running the GUI.

1.3 Deliverables

1.) LED based illumination system integrated on a custom PCB, with various LED rings of different colours, provide multimodal imaging in brightfield and darkfield, in series or parallel. Where the use of a mono camera would provide better fill factor and higher sensitivity.

2.) Code developed to allow automatic control of the microscope through pre-defined parameters as a program running on the Raspberry Pi 4B. It will have the capability to process raw data from the camera into a useable format for the analysis of the *C. elegan*s throughout experiments.

3.) High level GUI developed to allow the user to control the microscope wirelessly, with a corresponding program running on the Raspberry Pi 4B to allow this functionality.

4.) A comprehensively developed and rigorously tested prototype system design to take contrast enhanced images, with µm scale resolution to be able to resolve *C. elegans* of size 50µm diameter and 1000µm length [6]. The prototype system needs to fit in the 1U CubeSat specifications of $10cm^3$ [7], and the power required for the system must be able to run off the CubeSats' 5V 4A DC supply. While being able to withstand stresses involved with a rocket launch to LEO.

5.) A thesis, in which contains methodology of the project including, system modelling, system development and validation.

6.) A presentation to demonstrate the prototype working and the findings from the data derived from use of the working prototype. In addition, a reflection on the initial aim, objectives, and deliverables, and if all factors were achieved, if not the reason for each is stated.

The diagram shown in Figure 1 provide a high-level overview of the project and how the four main components interact such as stating what each section allows for and how each section communicates.

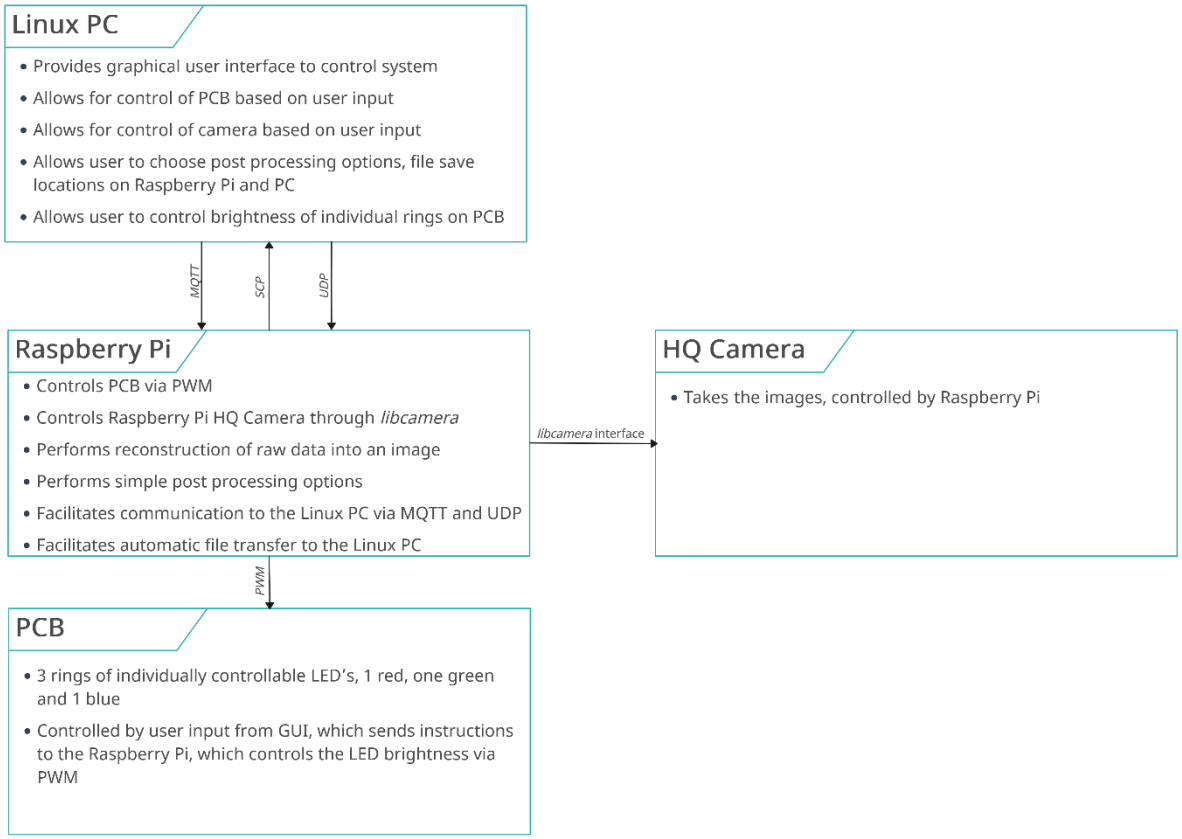

Figure 1: High level overview of entire system, stating the main tasks each component does along with the communication method between the four components.

Risk Mitigation

Below shows the risk resister set at the start of the project, including potential risks with mitigation strategies and quantified pre and post mitigation value.

	Mitigation Actions	Impact on project and rating (1-3, 1 meaning low impact 3 meaning high impact) pre and post mitigation	Likelihood of risk (1-3, 1 being low chance and 3 being high chance) pre and post mitigation	When the risk is likely to happen	Quantified risk pre and post mitigation = (Impact on project * likelihood)
Components ordered that don't arrive or are delayed	Adding 1 week to the expected delivery estimate to help account for shipping delays.	Delays to prototyping and if components had to be reorder if out of stock could mean change in components, and potential specification change. Pre mitigation: 3 Post mitigation: 2	Pre mitigation: 1 Post mitigation: 1	This risk is only likely to happen in week 4 and 6 as this is when the time plan is set to order components.	Pre mitigation= 3*1 = 3 Post mitigation= 2*1 = 2
Running out of time to complete all sections	Mostly mitigated by the Gannt chart through error and error dependency bars, and for critical aspects such as the LEDs the original plan is to use phosphor LEDs, and backup plan is to use RGB LEDs to produce white light.	Falling too far behind schedule could mean resorting to the backup plan of using white LEDs with no phosphor coating. Pre mitigation: 2 Post mitigation: 1	Pre mitigation: 3 Post mitigation: 1	This risk is likely to occur midway through the project as can see if the project is on track.	Pre mitigation= 2*3 = 6 Post mitigation= 1*1 = 1
Custom PCBs shipping delays	As the technicians say PCBs normally take 2 weeks to arrive, the time plan allows for 5 weeks from ordering to when they are needed. And asking my supervisor to check over the design before I order the PCBs	If the PCBs are late or arrive on time but there was a defect on the design would result in delays to testing the PCB and potential extra cost if design changes had to be made. Pre mitigation: 3 Post mitigation: 1	Pre mitigation: 2 Post mitigation: 2	This is likely to occur at week 21 as this is when the design is finished for the PCB layout and will be ordered	Pre mitigation= 3*2 = 6 Post mitigation= 1*2 =2
3D printing failure for microscope hardware	Not every print will be successful so the largest parts will be printed first due to higher likelihood of failure.	Be delayed by a time to print a part. Pre mitigation: 2 Post mitigation: 1	Pre mitigation: 1 Post mitigation: 1	This is likely to happen through out the project as CAD design tends to be iterative.	Pre mitigation= 2*1 = 2 Post mitigation= 1*1 = 1

2 Design and Implementation

2.1 System Modelling

The development of the phase contrast microscope required a certain level of system modelling, which could involve simulations, analytical calculations, or empirical testing to ensure the effectiveness of selected tools for the intended applications. The modelling conducted, extended across various domains, encompassing the evaluation of potential software libraries for image processing as well as the conceptual design of a user-friendly GUI layout. Furthermore, the project entailed quantitative modelling, such as performing detailed analytical calculations and conducting comparative assessments to identify the most suitable camera for the task of imaging *C. elegans* worms. Additionally, the efficiency of the image processing program was scrutinised through memory tests and by timing the duration required for image analysis tasks, as well as for the transfer of files between the Raspberry Pi 4 and the Linux-based computer that hosted the GUI.

2.1.1 OpenCV vs rawpy for raw image reconstruction and post-processing

Originally the plan was to use OpenCV for image capture and imaging processing, as outlined in the Gantt Chart and in the project proposal specification section under the software subheading. This decision was made due to the wide range of features, a large community making for easier debugging, prior knowledge and use of the library, and that C/C++ were the only languages known, leading to a preference to stick with these due to concerns about the time required to learn a new language. However, in the research and development phase, it was discovered that a limitation of OpenCV was that it only supported saving post-processed images as 8-bit depth. Although the bit depth of the images was not a point in the specification, discussions with the supervisor led to the conclusion that a higher bit depth was a necessary feature, necessitating a different approach.

After further research, a potential solution to this issue was identified: a Python library called rawpy, which handles the conversion from raw data into a viewable format such as a .TIFF, through the use of various demosaic algorithms presented to the user and provides a variety of post processing options as part of the library. Table 1 compares OpenCV to rawpy to determine the better library for the image processing application.

Feature	OpenCV	rawpy
Supported Bit Depth	8-bit depth	16-bit depth
Processing Time	Generally faster due to C/C++ implementation	Slightly longer due to the nature of the Python language
Language Compatibility	Native support for C/C++	Python-based, requiring integration with C++ code
Integration Complexity	Seamless integration for C/C++ environments	More complex due to cross language integration

Post-Processing Options	Wide range of options available	Limited, number of options
Primary Use Case	Broad image processing application	Specialised in processing raw image data
Ability to Control Camera	Yes	No
Number of demosaic algorithms available	2	13
Library File Size	1GB-6GB	≤ 100MB
Ability to Control Camera	Yes	No

Table 1: Comparison of OpenCV vs rawpy for raw image reconstruction and post processing.

Reflecting upon the comparisons drawn in Table 1, it was observed that rawpy presented substantial advantages in essential aspects, notably in its support for higher bit depth and a more extensive selection of demosaic algorithms. Although rawpy's processing times will be marginally longer than those of OpenCV, the difference would be negligible, due to rawpy's architecture as a high-level Python interface to the C-based LibRaw library. This setup ensures that the bulk of computationally intensive tasks are efficiently handled by LibRaw's compiled C code, with minimal overhead from the Python layer.

A further point in favour of rawpy, as stated in Table 1, and pertains to its library size. OpenCV, being a comprehensive image processing suite, has a considerably larger library file size. In contrast, rawpy, with its focused functionality, requires less storage space, aligning more closely with the project's requirements for higher bit depth processing and a broader range of demosaic options without compromising the system's overall performance on the CubeSat.

However, the transition to rawpy was not without its challenges. As there is higher complexity in the integration with existing C++ components, a narrowed scope of post-processing functionalities, the inability of built-in camera control and slightly higher memory usage. Consequently, the project adapted by employing the libcamera command line interface for image acquisition, diverging from the previously utilised OpenCV approach, however this is justifiable as the method for camera control makes no overall difference to functionality of the project.

To validate rawpy's effectiveness in generating 16-bit depth images, the ImageMagick Linux command line utility was employed, verifying the bit depth of images processed with rawpy. The command used and its output are presented in Figure 2, serving as a preliminary confirmation of the project's capability to autonomously capture and store phase contrast images, thus fulfilling a critical requirement outlined in the project specifications. To optimise this further an ideal solution would use a bit packing approach that would allow 12-bit images to share a 16-bit image stream without the need for padding, making it more efficient for communications from space.

Figure 2: Proof of rawpy processed 16 bit-depth image using ImageMagick command line utility.

2.1.2 Modelling of GUI Layout

To model a design for the microscope control GUI, I used QT Creator, as this allows for widget-based applications with a drag and drop approach for the design and layout of the GUI, the final layout is shown in Figure 3. Initial research was done to discover what widgets were available with QT, and the layout of the GUI was done from a high-level perspective to abstract away the complexity from the user, while providing a layout that makes sense and is easy for anyone to understand with the benefit of making the GUI non-specific to this application of imaging *C. elegan* worms, allows for this GUI to be used on any future project that has a Raspberry Pi to capture images, and is also not tied to any camera as long as the camera used is compatible with the libcamera command line interface.

Figure 3: Layout of the final GUI that was created with a QT widget application inside QT Creator.

2.1.3 Camera Selection

Camera selection was an essential part of the project as it will determine what lenses can be used, the resolution and thus how much resolvable detail, as higher resolution means smaller pixel size. As discussed in section 2.1.1 the project used the libcamera command line interface for taking images, for this reason it was sensible to use an official Raspberry Pi camera for the project to ensure compatibility with libcamera. Raspberry Pi provides an extensive hardware specification table for all six official Raspberry Pi camera modules. Camera Module v1, v2, Camera Module 3 and Camera Module 3 Wide are not fit for this application due to no ability to change the lens configuration, which was a requirement for the project's success, as the ability to change lens allowed for a lens kit to be used and tested to find the most appropriate

lens. The last two camera the HQ Camera and the Global Shutter (GS) Camera were both viable options and an in-depth comparison is shown in Table 2.

	HQ Camera	GS Camera
Size	38 x 38 x 18.4mm (excludes lens)	38 x 38 x 19.8mm (29.5mm with adapter and dust cap)
Weight	30.4 grams	34 grams
Still Resolution	12.3 Megapixels	1.58 Megapixels
Sensor Resolution	4056 x 3040 pixels	1456 x 1088 pixels
Sensor Size Diagonal	7.9mm	6.3mm
Pixel Size	1.55µm x 1.55µm	3.45µm x 3.45µm
Lens Mount	C/CS mount or M12-mount	C/CS mount
Shutter Type	Rolling	Global
Cost Manufacturer Listed	$50 or £38.89	$50 or £38.89

Table 2: Comparison of Raspberry Pi HQ Camera and Raspberry Pi GS Camera. All data from [1]

Table 2 shows the comparison of HQ Camera and the GS Camera, this showed that on all but one metric the HQ Camera was the better choice, due to smaller size and weight but the main advantage of the HQ Camera over the GS Camera is the higher resolution and smaller pixel size. This is because pixel size is one of the main parameters in the resolvable detail from an image. However, where the GS Camera does have an advantage is having a global shutter, this means the entire sensor is exposed to light simultaneously, and thus all pixels start and stop collecting light data at the same time. The advantage this gives, is that when imaging moving specimens no time domain artifacts will occur in the image if the specimen is moving. Rolling shutter like on the HQ Camera works by scanning across the scene either vertically or horizontally meaning the whole image is not captured at the same time, resulting in the potential for motion artefact if the specimen moves during capture. As the aim of the project was to accurately image *C. elegan* worms with µm scale resolution the HQ Camera was chosen, and as *C. elegan* are slow moving results in less chance for distortion in the images.

2.1.4 Performance Evaluation of rawpy: Memory Usage and Processing Time Analysis

2.1.4.1 Rawpy Memory Usage Analysis

To accurately assess the memory consumption of rawpy, the memory_profiler package was used (Table 3). This tool offers a detailed view of memory usage by each

line of code, providing a more comprehensive analysis compared to Python's built-in *tracemalloc*. The latter is limited to tracking memory allocations solely within Python code, as is unable to track allocations made by external C libraries. As rawpy relies on the C library LibRaw for its core functionality, *tracemalloc* would not capture the complete memory usage. In contrast, memory_profiler monitors both Python-specific and overall process memory allocations, thereby providing a more accurate insight into the memory allocation of rawpy. This provides a valuable insight, as even though for development purposes a Raspberry Pi 4B is being used, when implemented in the intended scenario of a CubeSat a Raspberry Pi 0W would be used as this Single Board Computer (SBC) has a lower power draw, same GPIO pinout and has a CSI camera port (although a different cable or adapter would be needed as the Raspberry Pi 0W uses a 22 pin CSI connector and the Raspberry pi 4B uses a 15 pin CSI connector). However, the Raspberry Pi 0W comes with a maximum of 512MB of RAM and thus was important to model/test the memory usage to ensure compatibility between the test platform and what would be used for the final implementation onto the CubeSat. Table 3 shows the output of the memory profiler.

As shown in Table 3, the *raw.postprocess()* function uses the most memory of 243.06 Mb and this was the highest amount of memory used by the program overall. Meaning that if ran on a Raspberry Pi 0W it would consume less than half of the available RAM and thus would work as expected when implemented on the CubeSat, and thus advances the progress on meeting the specification point of a system that can autonomously capture phase contrast images.

Line #	Mem usage	Mem usage (MB)	Increment	Occurrences	Line Contents
8	42.0 MiB	44.04	42.0 MiB	1	@profile
9					def process_image(filename):
10					# Step 1: Process raw image and save as TIFF
11	42.0 MiB	44.0	0.0 MiB	1	path = filename.decode()
12	42.0 MiB	44.0	0.0 MiB	1	print(f”Processing image at path: {path}”)
14	114.3 MiB	119.9	-116.4 MiB	2	with rawpy.imread(path) as raw:
15	231.8 MiB	243.1	188.7 MiB	1	rgb = raw.postprocess(output_bps=16)
17	114.3 MiB	119.85	0.0 MiB	1	tiff_path = ‘linear.tiff’
18	115.7 MiB	121.32	1.4 MiB	1	imageio.imsave(tiff_path, rgb)

Table 3: Output from the memory profiler on a simplified Python program to show memory usage per line with extra column to show memory usage in MB.

2.1.4.2 Rawpy Processing Time Analysis

Processing time analysis had to be done to find the average processing time for rawpy to reconstruct the raw data into a viewable format and perform simple post processing. This was important data needed by the GUI, this is because if you allow the user to take another image while the previous is still processing can result in unexpected behaviour or loss of the previous image, an attempted was made to make this work multi-threaded, however the complexity of providing this for the image processing Python function that is called by a C++ program proved to be more difficult than expected, thus wasn't able to get working after around 5-7 hours of trying. Having performed this analysis, it gives the average processing time length, where the real time shows the time actual real-world time taken to run the program, the user time is higher than the real time as this takes a summation of the time it took all cores to run the program. The program was not written in a way to take advantage of parallelisation, but the functionality is within the underlying C library LibRaw and thus is abstracted away from the user, thus if the program had no degree of parallelisation, it would take the user time rather than the real time.

	Test Number										
	1	2	3	4	5	6	7	8	9	10	Avg
Real (s)	2.84	2.75	2.78	2.78	2.80	2.77	2.77	2.77	2.77	2.76	2.78
User (s)	4.40	4.48	4.40	4.37	4.47	4.50	4.52	4.48	4.51	4.49	4.46

Table 4: Program run time for rawpy to construct image from raw data and perform basic post processing, multiple run times recorded and averaged.

2.1.5 Lens Characterisation

A large portion of the project revolved around optimising the lens optics with the chosen Raspberry Pi HQ camera. Justification for camera selection can be found at section 2.1.3. The lens choice is paramount to the success of the project as it determines various factors such as magnification, FOV and minimum object distance, which, in turn, sets whether and in how much context the *C. elegan* worms will be imageable. Thus, a lens had to be found that would allow for high enough magnification to resolve details of the *C. elegans*, with an appropriate FOV diameter to capture the entire length and width of the body of the C. elegans, while also taking the minimum object distance into account as not to exceed the $10cm^3$ size requirement of the CubeSat.

Due to the complexity of needed parameters, it was decided to purchase a lens kit [8], since the budge-compatible lenses for the Pi camera format are poorly described (if at all) in supplied data sheets. The set allowed trade-offs to be mapped, performing data analysis of the needed parameters to judge whether the selected lens was fit for purpose. The selected lens kit came with five different lenses, all with different optical specifications but share the same CS mounting system, only giving information on estimated parameters (e.g. focal length) of each lens. The values stated would

typically be enough to categorise the lenses and perform analytical calculations, however, as will be shown in this section of the report, these given parameters were proven to be mainly inaccurate. Because of this significant time was spent characterising the lenses for accurate measurements that can be relied upon in the final choice for lens selection.

2.1.5.1 Finite Conjugate Imaging

The first experiments done to categorise the lens kit was for individual lenses meaning a finite conjugate imaging system. Figure 4 shows a plot of object distance vs the size of a 500µm feature in pixels with the corresponding magnification. To determine the magnification value, the size of the 0.5mm feature had to be converted from pixels to mm, given that from Table 2 the pixel size for the Raspberry Pi HQ camera is 1.55µm x 1.55µm, Equation 1 outlines the formula needed for the conversion.

$$I_{Size} = N_{Pixels} * P_{Size}(mm) \quad (1)$$

Where:

I_{Size} denotes image size of 0.5mm scale bar.

N_{Pixels} denotes number of pixels.

P_{Size} denotes pixel size in mm.

Equation 2 shows the formula to calculate the magnification.

$$Magnification = \frac{I_{Size}}{A_{Size}} \quad (2)$$

Where:

A_{Size} denotes actual size of 0.5mm scale bar.

	M01	M02	M03	M05	M07
EFL (mm)	25	16	12	8	6
MOD (mm)	200	200	200	300	300

Table 5: Table summarising the EFL and MOD in mm stated by the datasheet.

In figure 4, six measurements were taken per lens, this was needed as each lens had an adjustable focal length. By rotating this setting, it adjusts the BFL, for CS mount one 360-degree rotation is the pitch length of a CS mount (0.75mm) and thus each full rotation adjusts the BFL by 0.75mm, and one measurement was made for each BFL adjustment in increments of 0.75mm. The object distance to front lens is defined as the minimum distance in mm to achieve optimal focus of the lens. Highlighted in the black box in figure 4 are all data points for each BFL adjustment that would fit within the CubeSat 10cm^3 limit. However, this is not practical as there still needs to be room to fit the illumination system and house the specimens. Given this the all the data points inside the grey box in Figure 4 are a more reasonable maximum size for the object distance while considering space required for the illumination system and specimens.

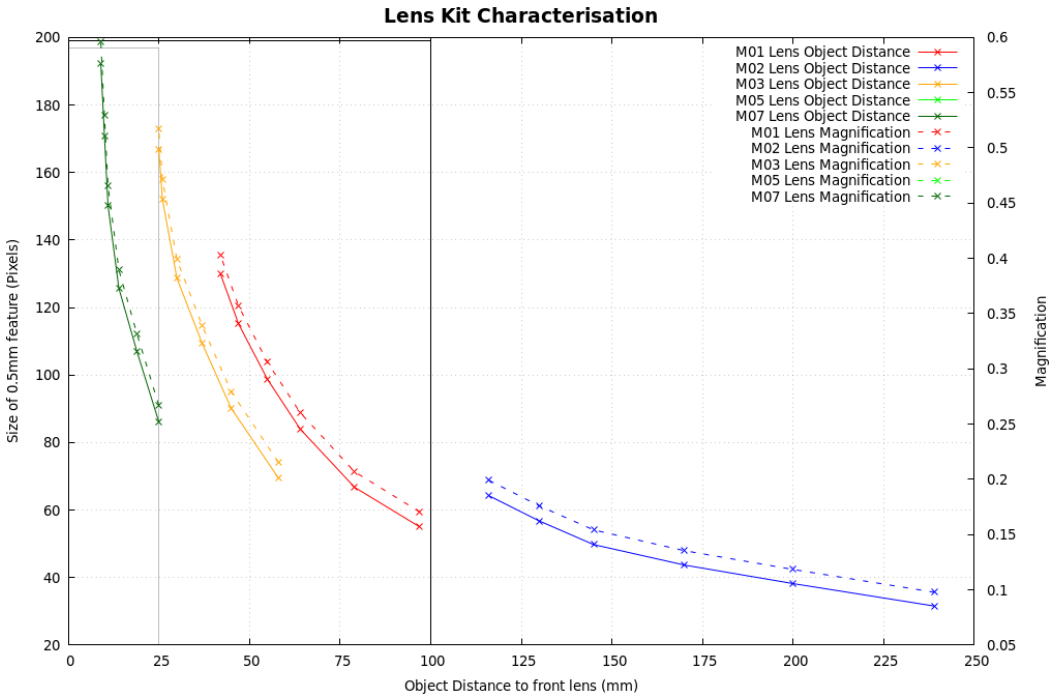

Figure 4: Measurements for size of 500 µm feature in pixels, magnification, and object distance for 0.75mm BFL adjustment.

Even though the M07 lens would be suitable due to its magnification of approximately 0.73 and low minimum object distance, by transitioning to an infinite conjugate imaging system as show in section 2.1.5.2 a similar minimum object distance was achieved with a magnification greater than 5 times what the single M07 lens was capable of, for the best lens pair in the infinite conjugate imaging setup. This allows for a wider application scope as can image the developmental stages of the C. *elegans* in higher detail but sacrifices in FOV.

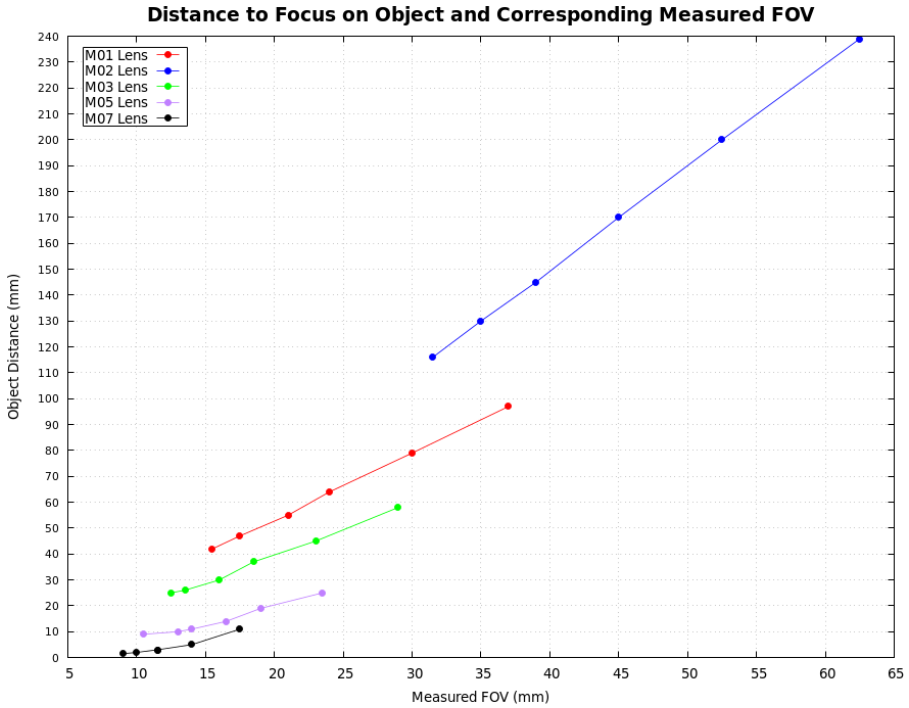

Figure 5: Graph of object distance in mm vs measured FOV.

When comparing the results from Figure 4 and 5 to the given datasheet parameters in Table 5 proves how inaccurate the given datasheet was and thus the reason for further modelling into the lenses to produce data that can be relied upon.

2.1.5.2 Infinite Conjugate Imaging

As discussed in section 2.1.5.1 the finite conjugate imaging system had shortcomings in terms of the magnification, with improvements also possible to the object distance. To improve in these areas a technique called infinite conjugate imaging was employed, the concept is simple and consists of using two lenses coupled together, where the lens where the light enters is the objective lens, and the second lens is connected to the camera to focus light on the sensor and is known as a tube lens. Specifically, the configuration used is a reverse macro technique where the two lenses are coupled together with no gap between the objective lens and the tube lens. The advantage of this technique is that the magnification becomes a function of the ratio of the focal lengths of the two lenses, as shown in Equation 3 [9], and thus has the benefit of low minimum object distance, similar to the M07 lens in a finite conjugate system but with a higher magnification that what was possible with the finite conjugate imaging system, analysed in section 2.1.5.1.

$$Magnification = \frac{f_{Tube\ Lens}}{f_{Objective}} \tag{3}$$

Where:

$f_{Tube\ Lens}$ is the focal length of the tube lens.

$F_{Objective}$ is the focal length of the objective lens.

In addition, the use of an infinite conjugate imaging system allows for a reduced object distance due to the physics of how the light passes through the optical setup. As shown in figure 6 [10] utilising an infinite conjugate system causes the light beams that pass through the objective lens to project an image at infinity. Meaning that divergent light rays are converted into parallel beams of light, thus the magnification remains constant, regardless of adjustments made to the distance between the objective lens and the tube lens [ref], this allows for reduced object distance as the parallel light rays can be focused onto the camera sensor by the tube lens at a closer distance than would be possible if the rays were diverging from a point closer than its normal minimum focusing distance.

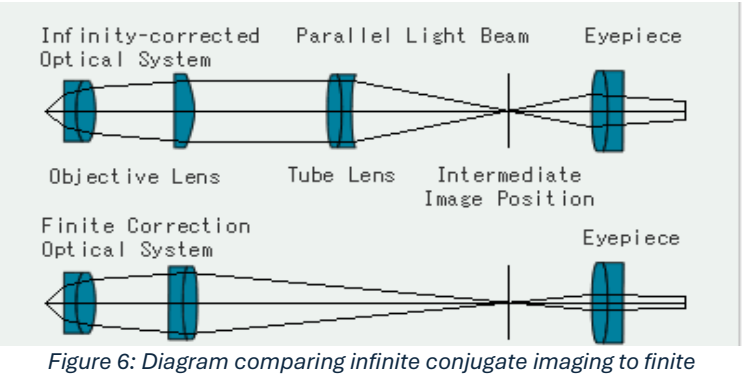

Figure 6: Diagram comparing infinite conjugate imaging to finite conjugate imaging to show how the light rays become parallel in an infinite conjugate system.

One final benefit of infinite conjugate imaging is that the distance between the objective lens and tube lens can be varied which allows for a wider potential application range, as different optical components can be placed in between the objective lens and the tube lens to change the parameters of the microscope, such as optical filters could be placed here to filter out unwanted wavelengths of light. This allows for a system that is much more expandable with the capability for a much wider range of applications and experiments.

However there also limitations to an infinite conjugate imaging system, such as higher complexity for setup and higher cost due to the need of two lenses. More importantly, the more optical elements in the system will result on higher amounts of light loss, making the image appear darker and issues such as vignetting can occur which can be due to apertures/stops in the lenses. Further discussion of vignetting explained in section 2.1.5.3. This is also key when choosing lenses as high-end lenses from companies such as Edmund Optics can come with more optical surfaces when aberrations are attempted to be corrected for, where each optical component reduces the light throughput. Thus, by understanding the potential which aberrations could occur in the system then the selected lens should only account for these in order to keep the light loss low.

2.1.5.3 Lens Selection for Infinite Conjugate Imaging System

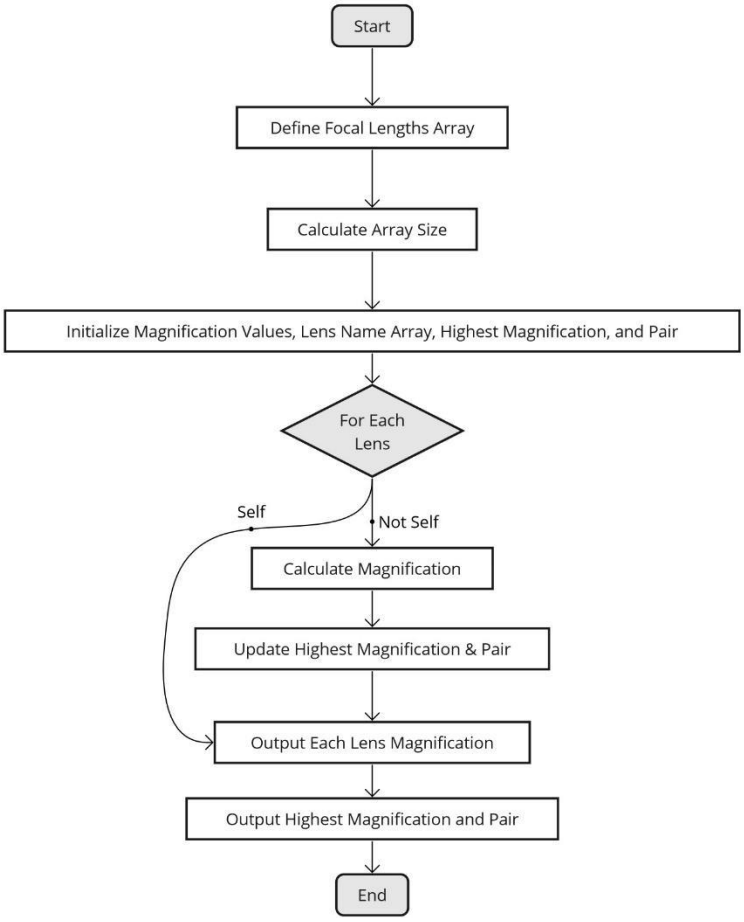

Figure 7: Flowchart showing the operation of the program wrote to calculate the magnification for every lens pair.

As infinite conjugate imaging requires two lenses, and thus given the lens kit with five lenses, results in 20 different lens combinations when excluding using the same lens for both, although using the same lens for both would be a good solution as the aberrations would cancel out, however there is only one of each lens and thus can't be used. Because of this a C++ program was wrote to find the magnification base on the focal lengths of the lenses stated in the data sheet, Figure 7 shows a flowchart of how this program worked to calculate the magnification of each lens pair based of the formula stated in Equation 3.

The output results from this C++ program were formatted into a heat map to clearly indicate the magnification of every lens pair and can be seen in Figure 8.

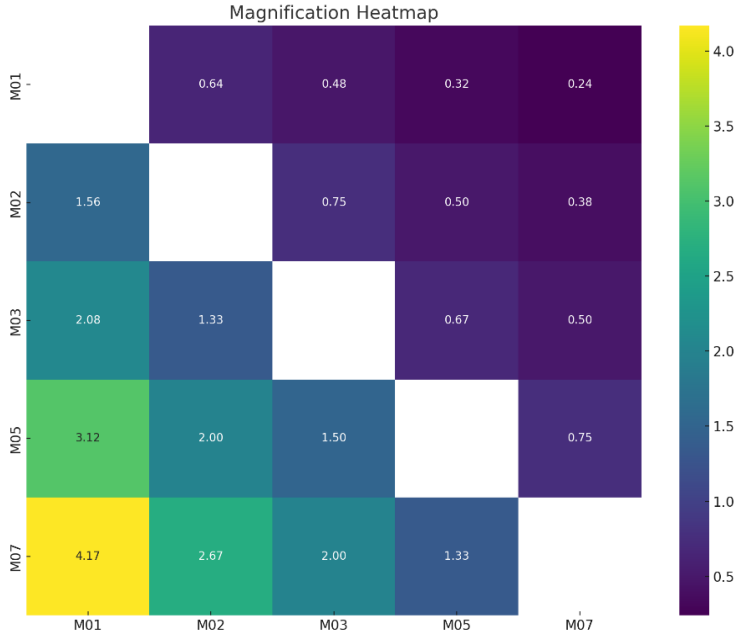

Figure 8: Magnification heatmap for each lens pair, based on datasheet focal lengths.

With the data found in Figure 8, all lens configurations with a magnification greater than one, were tested by taking an image of a 0.5mm scale bar so that the magnification and FOV can be determined along with the corresponding object distance for optimum focus for each lens pair configuration. Due to the inaccurate nature of the data sheet the magnification was measured as to compare to the theoretical results based on the datasheet parameters.

As an example, the highest theoretical magnification based on the datasheet values was 4.17 and was from the lens pair M07/M01, however when processed in ImageJ to calculate the magnification by using Equations 1 and 2 from section 2.1.5.1, the measured magnification was 2.18 which is half of the expected theoretical value. This demonstrates that the given datasheet parameters are not accurate and motivated the characterisation experiments described. From Figure 9, Figure 10, Figure 11, it is shown that the two best lens pairs are M02/M01 and M02/M05, as M02/M01 provides the best magnification with an FOV large enough to fit an adult *C. elegan* worm (average length of 1mm [11]) fully in frame with an 8mm minimum object distance. M02/M05 was also a good choice as it allows for a slightly larger FOV by compromising in a reduction in magnification in comparison to M02/M01 while still having a minimum object distance of 7.5mm.

As well as the measurements for magnification measurements were taken for the FOV, object distance and vignetting percentage (where the vignetting was calculated from Figure 10 as a white background allows for full view of the vignetting), the results are shown in Figure 11.

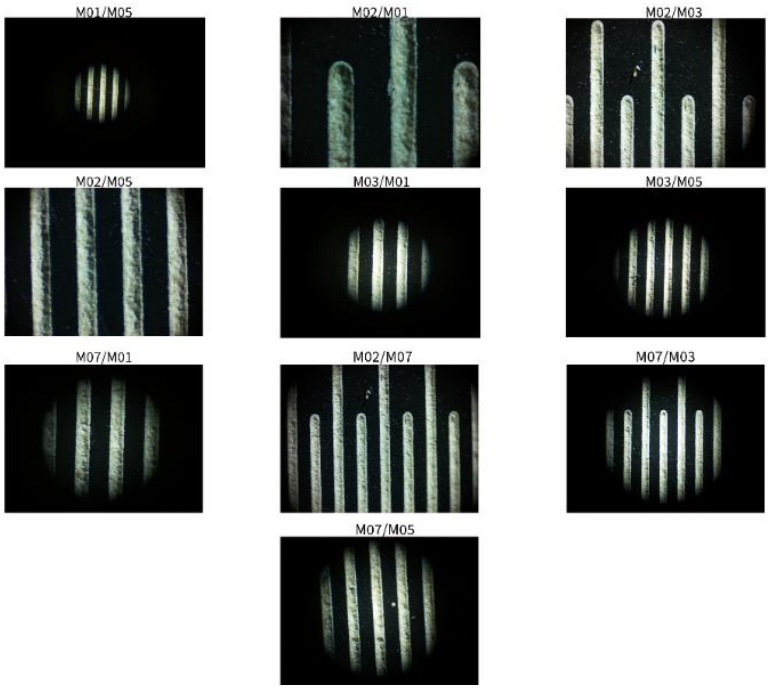

Figure 4: Image of 500µm scale bar for each lens pair.

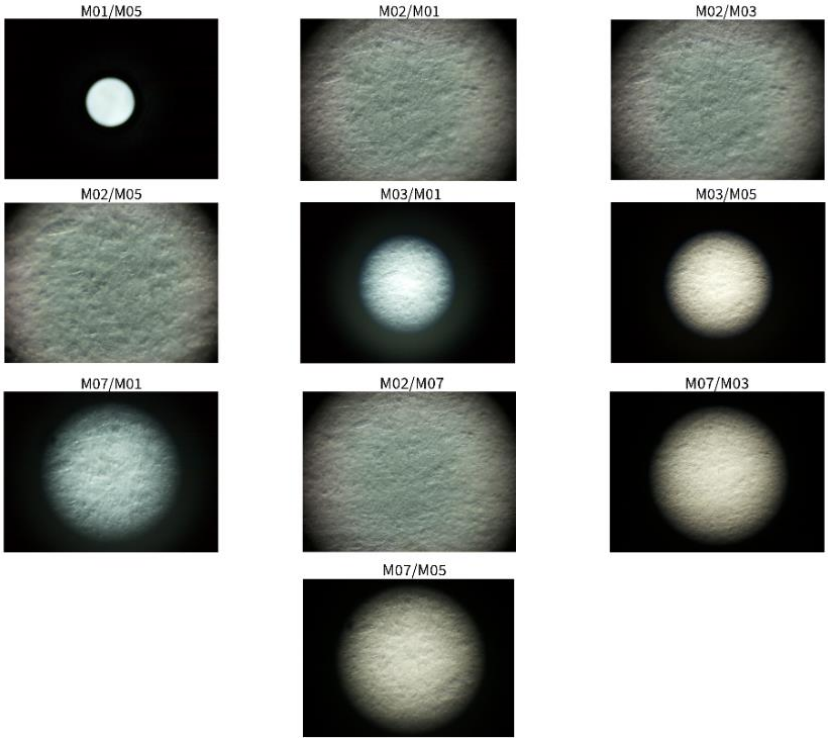

Figure 5: Vignetting tests with white paper background to see vignetting more effectively.

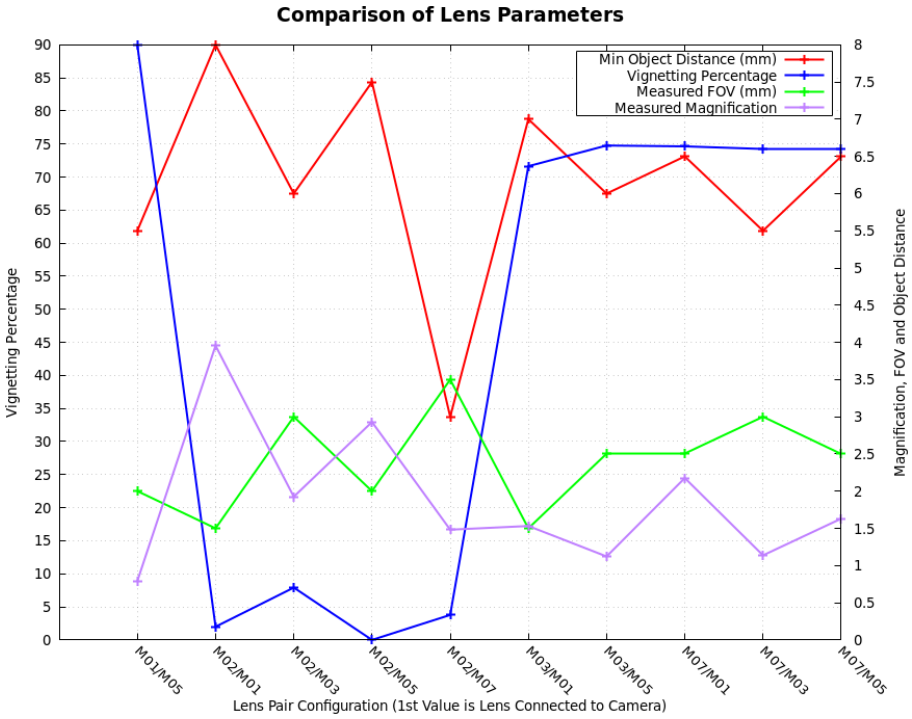

Figure 11: Measured data of magnification, FOV, object distance and vignetting percentage from each lens pair.

2.1.5.4 Modelling Microscope Resolving Power

This entire section was a novelty section as was not previously discussed with the supervisor but provide a more in-depth analysis of the lens configurations. To find the resolving power of the selected lenses in section 2.1.5.3, tests were performed to see what level of detail the lens pairs selected were capable of. This was done by imaging the RGB pixels present in a Samsung Galaxy Tab S6 Lite tablet, the reason for this is because the present an effective way to image small objects as the RGB pixels are made up of three sub pixels, and thus gives an effective way to determine the resolving power of the microscope, by which sections of the RGB pixels can be resolved. For all tests carried out the microscope took images of a pure white screen set to 50% brightness, white was chosen as it means that all red, green, and blue pixels will be set to the same intensity, summing to be perceived as white light by the human eye.

For both lens pairs the diagonal size of one pixel was calculated based off of known parameters, of the specs of the Samsung Galaxy Tab S6 Lite (SM-P610) tablet [12]. Table 6 shows the needed parameters of the tablet to calculate the RGB pixel size.

Samsung Galaxy Tab S6 Lite (SM-P610) Parameters				
Resolution	Diagonal Screen Size	PPI	Screen Width	Screen Height
2000x1200	10.4 inches	224	226mm	135

Table 6: Parameters of Samsung Galaxy Tab S6 Lite needed for calculation of RGB pixel sizes.

Using the data from table 5, the diagonal pixel size can be calculated as shown in equations 4-7.

$$D_{Resolution} = \sqrt{W_{Pixels}^2 + H_{Pixels}^2} \tag{4}$$

Where $D_{Resolution}$ is the diagonal resolution, W_{Pixels} is the width in pixels, H_{Pixels} is the height in pixels.

$$D_{Resolution} = \sqrt{2000^2 + 1200^2} = 2332.38\ Pixels \tag{5}$$

Thus, given a PPI of 224 the diagonal size per RGB pixel can be calculated as shown in equation 6.

$$P_{Size}(inches) = \frac{1}{PPI} = \frac{1}{224} = 0.00446\ inches \tag{6}$$

$$\mathrm{P_{Size}(mm)} = 0.113\mathrm{mm} \tag{7}$$

2.1.5.4.1 M02/M01 Lens Selection Resolving Power

As the M02/M01 lens has the highest magnification it will have a higher resolving capability than the M02/M05 lens pair providing the image quality does not suffer. As seen in Figure 12 this lens pair provides good resolving capability as is able to see not only a pixel, where one pixel is defined as a red, green, and blue sub pixel, it is able to see each individual sub pixel that comprises one full pixel.

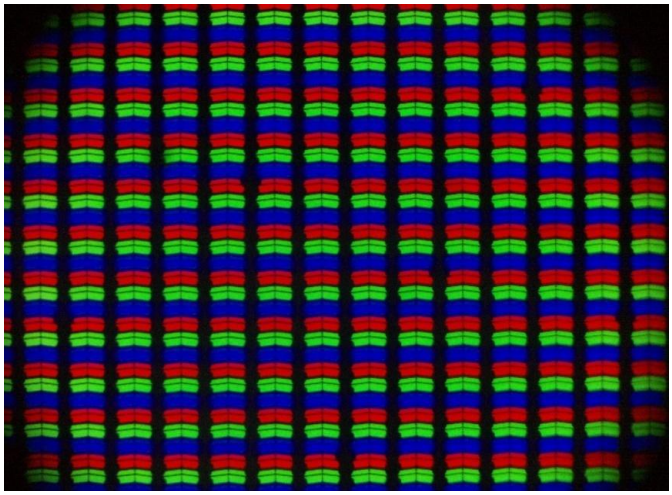

Figure 12: Image of the RGB Pixels of a purely white screen at 50% brightness of a Samsung Galaxy Tab S6 Lite, with M02/M01 lens pair.

	M02/M01 - 8mm Object Distance					
	Size in Pixels	Size Magnified (mm)	Real World Size (mm)	Known Pixel Size (mm)	Screen Size Calculated from Pixel Size	% Error Between Calculated and Measured Screen Sizes
Diagonal Size of Pixel	351.836	0.545	0.138	0.113	0.116	2.530
Width SubPixel	214.375	0.332	0.084	N/A	225.681	0.141
Height SubPixel	92.125	0.143	0.036	N/A	N/A	N/A
Pixel Height	286.562	0.444	0.112	N/A	134.747	0.187
Column Gap	73.594	0.114	0.029	N/A	57.676	N/A

Table 7: Sizes and parameters of the RGB pixels of the Samsung Tab S6 Lite, with M02/M01 lens pair.

To determine the size of the pixels and sub pixels of the lens configuration ImageJ was used to find all the dimensions in pixels of each element such as the all the pixel sizes and the width of the column gap. All the data from this is summarised in Table 7, as the diagonal size of one RGB pixel is known based off the calculations in Equations 4-7 and are calculated from parameters from the manufacturer, this becomes the reference to check for accuracy for example the column “screen size calculated from pixel size” for the diagonal size of pixel row, is calculated using Pythagoras theorem from the width and height of the sub pixel, and is shown to be close to the actual value calculated in Equation 7, and is within margin of error.

To find the resolving power of the microscope the same data used in Table 7 was used to find the half rise time of the red, green, and blue profile plots for each lens. This gave a distance value in pixels, however to verify against the specification this needs to be in µm, to do this the scale for each lens was found by using the images from the magnification tests in Figure 9 to find how many pixels comprise a 500µm distance in ImageJ, this result was then converted into µm using ImageJ set scale function, and was finally divided by 5 to get a 100µm scale as this is a more appropriate size for the specimens being imaged, the scales are summarised in Table 8.

	M02/ M01	M02/ M05	M03/ M01	M03/ M05	M07/ M01	M07/ M03	M07/ M05	M02/ M07	M01/ M05	M02/ M03
Scale	2.552	1.89	0.99	0.72	1.40	0.73	1.05	0.96	0.51	1.24

Table 8: Scale in pixels/µm for every lens combination.

	Resolution in Pixels and in µm					
	Red (pix)	Green (Pix)	Blue (Pix)	Red µm	Green µm	Blue µm
M02/M01	7.5	8.5	11.5	2.94	3.33	4.50
M02/M05	11	12.5	17	5.82	6.61	8.99
M03/M01	12	13.5	19	12.15	13.66	19.23
M03/M05	12	13	20	16.57	17.96	27.62
M07/M01	14.5	20	31	10.33	14.25	22.07
M07/M03	8.5	13	19	11.58	17.71	25.88
M07/M05	10.5	12.5	17	9.98	11.88	16.15
M02/M07	10.5	11	16.5	10.98	11.51	17.25
M01/M05	15.5	22	38	30.39	43.14	74.50
M02/M03	10.5	11	18	8.45	8.86	14.49

Table 9: Resolving power of each lens configuration in pixels and in µm.

The data in table 9 shows the resolving power in pixels and in µm, a colour scale is applied to show the best selections, where a higher value is worse for both data units. As shown the M02/M01 lens pair had the best resolving power with red being the best channel at almost 3µm. This meets the criteria of µm resolution and is better than required as with this resolving power, it is also possible to image *C. elegans* L1 larvae of size 15µm diameter and 250µm long [6].

2.1.5.6 Chromatic Aberration Analysis

Following on from the data in section 2.1.5.5 it can be seen that for the blue sub-pixels in Figure 12, that it is not as in focus as it is for the red and green sub-pixels. This is due to chromatic aberration caused by the lens pair, where the blue light coming through the lens is being focused to a different plane than the red and green light, meaning the blue light is not optimally focused on the imaging sensor of the camera causing a blurring affect.

An analysis of this was performed by taking the captured RGB image and using ImageJ to split the image into its red, green, and blue spectrums, the result of which can be seen in Figure 13 and a profile plot is recorded by capturing the edge transition from a pixel to the black background. Where the justification for the analysis, is it allows for a quantitative analysis and allows for a refinement in the merit function used to dictate the lens selection to meet the specification point of imaging *C. elegan* worms.

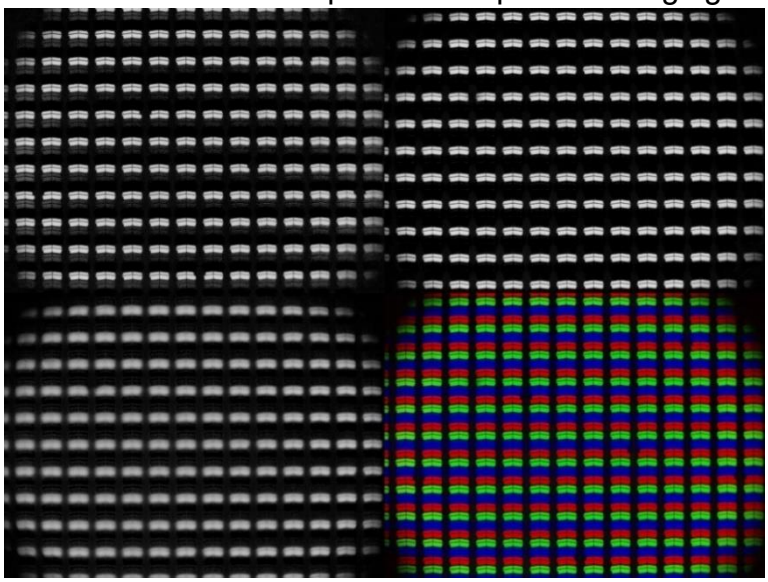

Figure 13:RGB image and the individual red, blue and green spectrum of the RGB pixels of the Samsung Tab S6 Lite tablet, for the M02/M01 lens pair. Top left is the red spectrum, top right is the green spectrum, bottom left is the blue spectrum and bottom right is the full RGB image.

From there to show the extent of the chromatic aberration each colour spectrum's profile was plotted and is shown in Figure 14 for the original data and the normalised data. The data shows that the M02/M01 lens pair provides proficient management of red and green wavelengths whereas the blue performance of the lens is poor due to chromatic aberration, if desired both red and green could be defocussed, to allow the blue light to focus on the imaging plane correctly and thus be in focus.

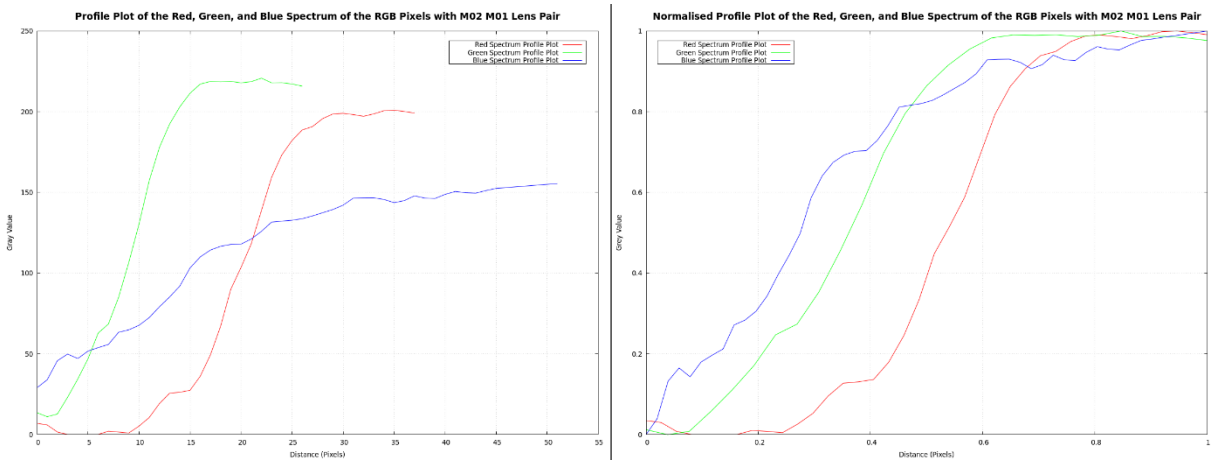

Figure 14: Profile plot of red, green, and blue pixels for edge transition from black background to pixel until intensity stabilises, for M02/M01 lens pair for normalised and original data.

The chromatic aberration is determined quantitively by fitting a logistic curve to the profile plot of a red, green, or blue pixel (using MATLAB with the curve fitting toolbox) found from ImageJ where a logistic curve follows the equation below in Equation 8.

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \quad (8)$$

Where L is the curves maximum value, x is the input to the logistic equation, x_0 is the x-value inflection point, e is the base of the natural logarithm and finally k is the rise time coefficient. The MATLAB script that utilises the curve fitting toolbox, it iterates values for L, x_0 and k to find the curve that best fits the given data. For this analysis only the outputted k value is of interest, as this is the rise time coefficient where a higher value correlates to a sharper transition from a red, green, or blue pixel to the black background.

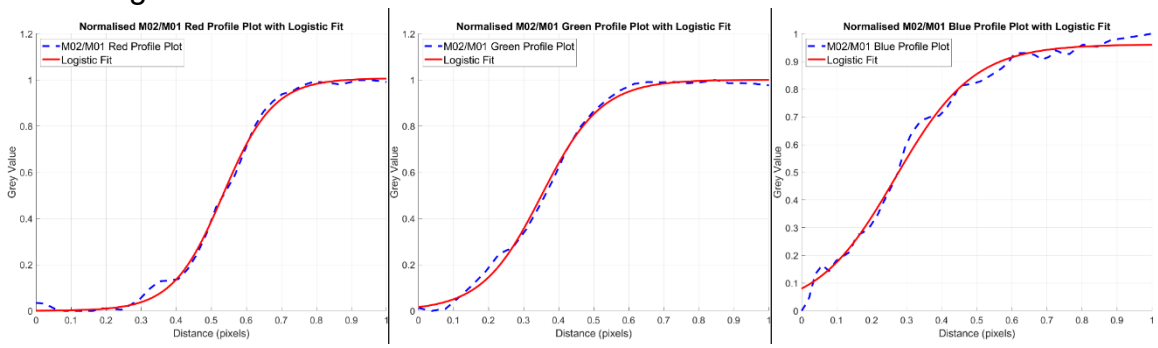

Figure 15: Graphs showing logistic curve fitted to the normalised data with MATLAB curve fitting toolbox for the M02/M01 lens pair for each colour channel.

The resolution is assessed based on the rise coefficient of the logistic function fitted to the normalised profile plot data by MATLAB; a higher value for the rise coefficient correlates with a more concentrated image, indicative of better focus. The result of which is visually discernible in the RGB visualisation showcased in Figure 13 and is reinforced by the comparison of the colour channels profile plots depicted in Figure 14 and 15 as it is clearly seen the rise time for the blue profile is much less than for red or green. In addition, the analytical results are summarised in Table 10 with all other values for other lens pairs for ease of comparison and so that lens selection can be selected for based off of the chromatic aberration and will be part of the merit function used to decide on the final lens pair.

	Red	Green	Blue	% Diff Red/green	% Diff Blue/green	Transmission Red	Transmission Green	Transmission Blue
M02-M01	13.72	12.32	8.96	11.36%	37.50%	0.91	0.65	0.774
M02-M05	9.38	11.05	8.17	15.11%	35.25%	0.94	0.76	0.801
M03-M01	8.63	9.723	6.87	11.24%	41.53%	0.9	0.71	0.7654
M03-M05	8.52	9.364	6.23	9.01%	50.30%	0.93	0.83	0.7921
M07-M01	7.185	11.22	7.02	35.96%	59.83%	0.9	0.69	0.6622
M07-M03	13.48	8.99	6.04	49.94%	48.84%	0.92	0.83	0.6853
M07-M05	9.73	11.81	8.58	17.61%	37.65%	0.92	0.69	0.6853
M02-M07	9.79	10.52	7.14	6.94%	47.34%	0.93	0.76	0.693
M01-M05	6.72	11.68	6.66	42.47%	75.38%	0.91	0.69	0.7654
M02-M03	9.96	10.52	6.34	5.32%	65.93%	0.93	0.79	0.801

Table 10: Rise coefficient transmission, % difference of a logistic function fitted to the profile plot data from ImageJ using MATLAB curve fitting toolbox and the transmission efficiency or each colour channel.

2.2.5.5 Light Transmission Efficiency Analysis

This was the final lens characterisation parameter considered as part of the analysis and was also a novelty section as none of the contents if this section was previously discussed with the supervisor. The aim was to find the light transmission efficiency of each lens for red, green, and blue colour channels. To do this a comparison is needed from the light intensity through the lens vs the intensity without the lens, where the light source is the same distance from the sensor for both scenarios.

The device used to measure the light intensity is a light dependent resistor (LDR) on a break-out board called KY-018, this measures the relative light intensity, from 0-1023 thus when compared to the value received without the lens the transmission efficiency can be calculated, the microcontroller used to read the data from the sensor was an Arduino Nano mounted to a piece of Vero board. If a more accurate test were to be conducted to measure the light value in lux instead of an efficiency a module such as the VEML7700 lux sensor from Adafruit could be used.

Shown in Figure 16 and Figure 17 is the testing setup for with the lens and without the lens, the dome shaped 3D print that holds the LED is designed such that the tip of the LED once inserted is 10mm from the lens casing. And in Figure 18, where there is no lens, it is the same distance from the tip of the LED to the sensor for both configurations

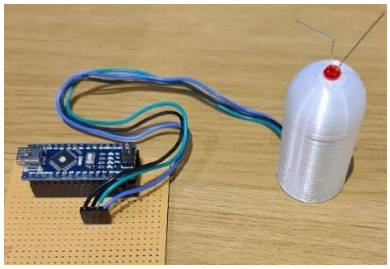

Figure 16: Setup for no lens light intensity value, needed to as baseline to calculate transmission efficiency.

Figure 17: Setup for light intensity value with a lens, used in comparison to the base line value to calculate transmission

Figure 18: Internal of the connector on the mount side of the lens that houses the LDR.

	Lens Analogue Light Intensity and Transmission Efficiency Values					
	No Lens	M01	M02	M03	M05	M07
Red	920	863	892	885	890	878
Green	556	427	472	515	500	501
Blue	852	654	734	766	755	760
Transmission efficiency red	N/A	93.804	96.956	96.195	96.739	95.434
Transmission efficiency green	N/A	76.798	84.892	92.625	89.928	90.107
Transmission efficiency blue	N/A	76.760	86.150	89.906	88.615	89.201

Table 11: Results from light transmission efficiency tests showing both received analogue value and transmission efficiency when compares to the no lens received analogue value.

2.2.5.6 Developed Merit Function for Lens Pair Selection

2.2.5.6.1 Generic Merit Function

Symbol	Description	Weight Value
ω_D	Weight for minimum object distance	0.25
ω_R	Weight for resolution	0.2
ω_V	Weight for vignetting	0.175
ω_M	Weight for magnification	0.15
ω_L	Weight for light transmission efficiency	0.1
ω_F	Weight for Field of View (FOV)	0.125
$f_D(D)$	Function calculating performance based on minimum object distance	N/A
$f_R(R)$	Function calculating performance based on resolution	N/A
$f_V(V)$	Function calculating performance based on vignetting	N/A
$f_M(M)$	Function calculating performance based on magnification	N/A
$f_L(L)$	Function calculating performance based on light transmission efficiency	N/A
$f_F(F)$	Function calculating performance based on field of view	N/A
T_{Red1} T_{Red2}	Transmission Efficiency for red light through first lens (T_{Red1}) and through the second lens (T_{Red2})	N/A
T_{Green1} T_{Green2}	Transmission Efficiency for green light through first lens (T_{Green1}) and through the second lens (T_{Green2})	N/A
T_{Blue1} T_{Blue2}	Transmission Efficiency for blue light through first lens (T_{Blue1}) and through the second lens (T_{Blue2})	N/A
$R_{red\ normalised}$	Resolving Power for red spectrum in µm normalised	N/A
$R_{green\ normalised}$	Resolving Power for green spectrum in µm normalised	N/A
$R_{blue\ normalised}$	Resolving Power blue spectrum in µm normalised	N/A

Table 12: Table summarising the symbols, description, and weight values for the merit function.

The merit function developed incorporates the analyses from section 2.1.5 into an equation that produces a single numerical output dictating its fitness for purpose in the application of imaging C. elegans, where a higher value dictates a more optimal lens pair. When used to calculate the merit function value for each lens pair the best configuration is found.

In the generic form the merit function takes the form shown in Equation 9, the function makes use of weights such that the more important parameters can be set at a higher impact to the output which are shown in Table X, with the symbols and descriptions.

$$f(x) = \left(\omega_D \cdot f_D(D)\right) + \left(\omega_R \cdot f_R(R)\right) + \left(\omega_V \cdot f_V(V)\right) + \left(\omega_M \cdot f_M(M)\right) + \left(\omega_L \cdot f_L(L)\right) + \left(\omega_F \cdot f_F(F)\right) \quad (9)$$

2.2.5.6.2 Expanded Merit Function

Section 2.2.5.6.1 presents the merit function in a generalised form such that it does not state the functionality of all the parameters in Table 12. Shown below is the final merit function in the expanded state, such that this was the function used to calculate the values for each lens pair.

Overall, the merit function is straightforward, for minimum object distance, magnification and FOV is just a linear contribution to the final output value, where the contribution amount is set by the weight assigned to each parameter. For resolution and light transmission efficiency as these have multiple parameters a weighted average is done so that all parameters are included in the function and the output of this is associated with a weight to set how much influence it has on the final output. Finally, for the vignetting parameter it only considers vignetting percentages less than 10% in the analysis, and as the values are percentages the value is normalised, like the other parameters the weight governs the contribution amount to the final output value.

$$f(x) = (\omega_D \cdot D) + \left(\omega_R \cdot \left(\omega_{\text{red}} \cdot \frac{1}{R_{red\ normalised}} \cdot \omega_{\text{green}} \cdot \frac{1}{R_{green\ normalised}} + \omega_{\text{red}} \cdot \frac{1}{R_{blue\ normalised}}\right)\right) + \omega_{\text{V}} \cdot \begin{cases} 1 - \dfrac{V}{100} & if\ V \leq 10\% \\ not\ included\ in\ analysis & if\ V > 10\% \end{cases} + (\omega_M \cdot M) + \omega_L \cdot \left(\omega_{Red}\ (T_{Red1} \cdot T_{Red2}) + \omega_{Green} \cdot (T_{Green1} \cdot T_{Green2}) + \omega_{Blue}(T_{Blue1} + T_{Blue2})\right) + (\omega_{\text{FOV}} \cdot F) \quad (10)$$

In the theme of good practice engineering presented in this thesis, the calculations for this merit function were developed into a MATLAB script, which is made generic such that it can be used to assess different lens pair solutions in the future once all required data has been gathered and can be used in for different experiments it was not originally intended for due to the flexibility the script provides in being able to alter the value of every parameters weight so the user can easily adapt these to suit the new requirements.

Shown in Table 13, is the final output from the merit function from the MATLAB script that implements the formula in Equation 10. As M02/M01 is the highest value and thus best lens pair, this will be the lens used for the final version.

M02/M01	M02/M05	M02/M03	M02/M07
5.64	5.58	5.26	4.44

Table 13: Table summarising the output from a MATLAB script that implements the formula in Equation 10 to select optimum lens pair.

2.2.5.6.1 Explanation Behind Merit Function Development

When developing the merit function, minimum object distance, magnification and FOV are set at a linear scale such that their contribution to the overall output is determined only by the value of the weight and the data value such as the minimum object distance for a certain lens pair. For the resolution component of the merit function, as this consists of three different values, one for red, green, and blue a weighted average was performed based on the individual weights for red, green, and blue and the overall weight set for the resolution. In terms of the vignetting component only lens pairs that

are below 10% vignetting are considered as any value above this becomes impractical for the application. For values of vignetting less than or equal to 10%, it is scaled to a value between 0-1 where 0% vignetting is best the and the contrary is the worst. Finally, for the lens transmission efficiency the overall lens pair transmission efficiency is calculated for each colour channel where total light loss is defined as transmission efficiency of the first lens multiplied by the transmission efficiency of the second lens. They are the multiplied by the weights assigned for each colour channel for a weighted average to get the final value for this parameter.

2.2 System Development

2.2.1 PCB Design for the Illumination System

2.2.1.1 First PCB Design for the Illumination System

The PCB design for the illumination system consisted of a large section of this project, due to five different iterations of the design although not all were manufactured. This PCB design was difficult due to the space limitations the CubeSat requirements impose of 10cm^3 to help mitigate this only SMD comments were used as the space required is drastically smaller than through hole components. The first board design was a single ring (of the needed three) and used an entirely different LED driver configuration to the final design, consisting of a TLC5947 constant current driver IC, this IC provides 24 addressable channels each with channel capable of delivering 30mA [13], while this design had the benefit of each LED being fully addressable both in colour and brightness, the board was just within the size requirements of the CubeSat and the design would not physically scale well with the two additional needed LED rings, and thus was not manufactured. The 3D model is show in Figure 19.

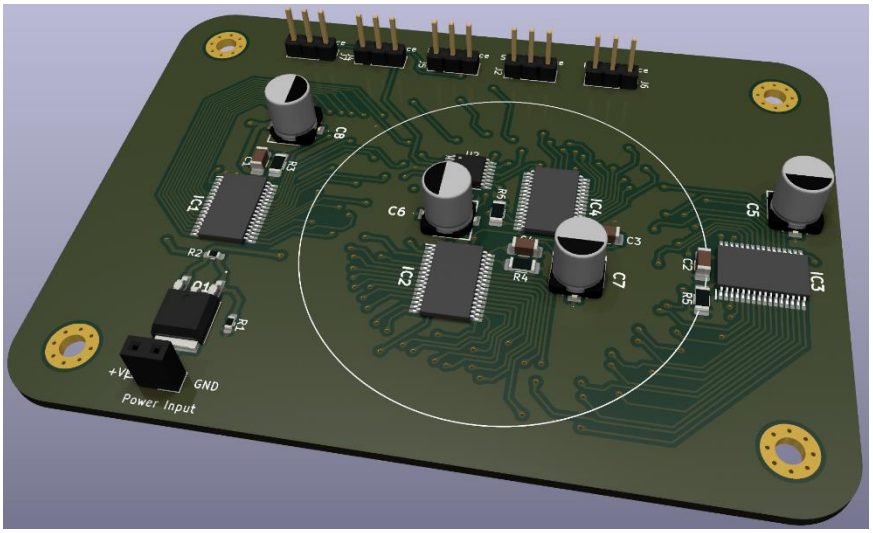

Figure 19: Render of the 1st PCB design for the illumination system, utilising 4 TLC5947 constant current driver IC's.

2.2.1.2 Second PCB Design for the Illumination System

Taking what was learnt from the original PCB design, it was evident that the board had to be made more compact and thus the constant current driver IC from the original PCB was not suitable. After discussions with the supervisor about what functionality was needed and what was nice to have, it was decided that the feature of each LED being independently addressable was not required and that RGB LEDs were not required. The compromise made was that each individual ring should be addressable in brightness for one specific colour only. This led to a constant current driver called PAM2804AAB010 [14], this IC fit the project needs well as it was a physically small (package TSOT25), has an enable pin which gives the capability to adjust the brightness through varying the duty cycle of a PWM signal generated by the Raspberry Pi 4. Has a feedback pin where a shunt resistor can be used to set the max current allowed to be drawn by the LEDs, giving flexibility in the choice of LEDs so each ring of LEDs does not need to have the same forward current as the only needed change would be in the shunt resistor value. With the final benefit being that the IC requires low number of external components to function, consisting of two 10µF capacitors, one 4.7µH inductor and the shunt resistor. This allows for the design to made compact and has the advantage over the previous version that no SPI communication is needed to control the LEDs, not having SPI allows for a simplified design with less chance of cross talk between traces.

In addition, this design added a fluorescent cyan LED which was added to the board design as one of the stretch goals was Rheinberg Illumination, and thus was implemented here for the potential to use this feature if given the time. In addition, this board has a more complex edge cut geometry, this was done for weight savings as there is a max weight limit of 1.33kg for the CubeSat. Fusion360 was used to create a DXF file to use as the edge cut, as the more complex geometry is not possible to create in KiCAD natively. This design also brought down the physical size from the first iteration of 56mm x 84mm (from mount holes) to 67.1mm x 67.1mm (from mount holes). This design was the proof of concept to go ahead with a 3 LED ring design and provided helpful insight into the optimum placement for components related to the constant current IC. In addition, this board also helped outline that to make this board more compact, the 4.7µH inductors had to be found in a different package along with the P-channel FET used for the reverse polarity protection. The final design is shown in figure 20 as a 3D model.

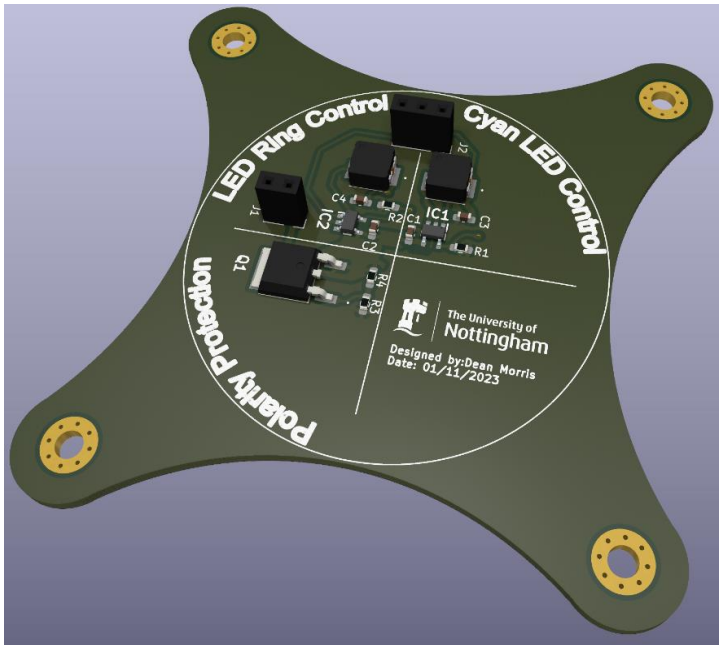

Figure 20: Render of the Second PCB revision utilising the final constant current driver IC.

2.2.1.3 Third and fourth PCB Design for the Illumination System

The third iteration of the PCB was close to the finished design, as the LED driving circuitry never changed and was correct from the third iteration. In this design all the LEDs for all the rings were wired in series, under the false assumption that the supply voltage only had to be the greater than the value of one forward voltage of the LEDs. The reasoning behind this was that LEDs are current driven devices and thus the reasoning was that if there was enough voltage to overcome the forward voltage of the LED then this would be fine as the LEDs are current driven from the constant current driver IC. However, the reason why this is wrong is because for a constant current LED driver to output constant current it needs to be able to vary the voltage, and it needs to be able to vary the voltage from 0 – the sum of all forward voltages in series. This meant that this board design was not going to work as the LED driver has a max input voltage of 6.5V [14], and with red LEDs as an example 18 in series with 2V forward voltage each would require 36V, this exceeds the rating of the driver IC and is also above the 5V limit allowable on the CubeSat. To address this issue the fourth board iteration used series parallel arrangement of LEDs of two in series in each 9 parallel branches. By utilising a series parallel arrangement, it decreases the current draw by a factor of two, however resistors had to be added to the design one per parallel branch. This is because in an ideal situation LEDs with a 2.5V forward voltage drop would be used, as when the total forward voltage drop is equal to the supply voltage then there is no need for current-limiting resistors as the LEDs can't draw more current than their natural forward current at the given 5V supply. This allows for a simpler design that has fewer components and is more efficient as no power is wasted in the form of heat in the resistors. The only reason this was not done is because LEDs with a forward current above 20mA, with 0402 SMD package size and a 2.5V forward voltage could not be found and thus for the fourth board revision along with changing to a series parallel arrangement the current limiting resistors were also added. Besides from electronic configuration the mounting holes were set to be 30mm squared so that the board can be mounted on a Thor Labs standard 30mm cage plate [15]. Both 3D models for the front and rear of the third design are shown in Figure 21 and Figure 22 respectively.

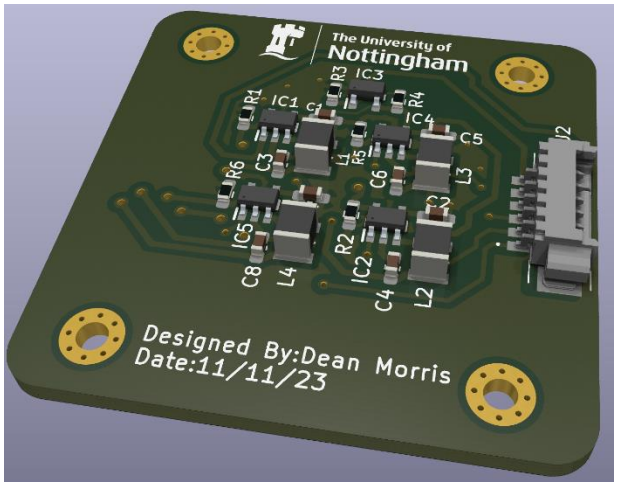

Figure 21: Render of the backside of third board iteration.

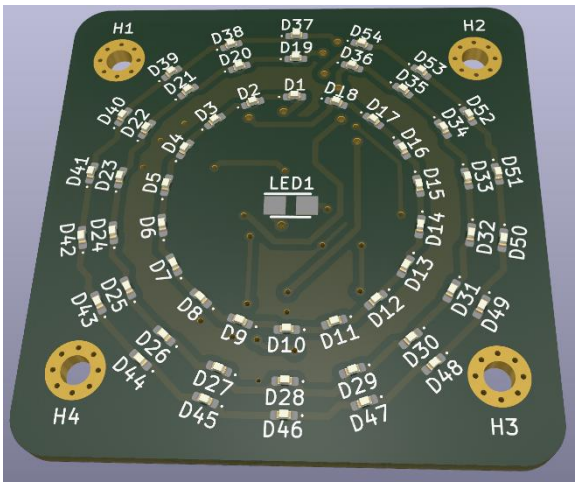

Figure 22: Render of the top side of the third board revision.

2.2.1.4 Final PCB Design for the Illumination System

2.2.1.4.1 Electronic Changes from Previous Iteration

The final iteration of the PCB illumination board addressed one small electronic configuration issue, which was that in the previous board iteration all LED rings were placed in a series parallel configuration, a wrong assumption was made that blue LEDs would not have a forward voltage greater than 2.5V, this is wrong as blue LEDs have a forward voltage in the range of 3V-3.5V this meant the combined forward voltage of each two in series for 9 parallel branches, was greater than 5V (supply voltage) meaning that the supply voltage was not high enough to be able to drive the blue LEDs effectively. The solution to this was just to change the blue LEDs configuration from a series parallel arrangement to a normal parallel arrangement. This has the advantage that the required voltage is now less than the supply voltage as the supply only has to be greater than the forward voltage of one LED. The disadvantage of this however is that the blue LEDs will draw twice the amount of current that they would have otherwise in the series parallel arrangement. Even if blue LEDs could be found with forward voltage less than 6.5V/2 (6.5V max voltage of current driver IC), it would work but would be over the allowable 5V limit for the CubeSat and thus is not a viable option.

2.2.1.4.2 Mechanical Changes from Previous Iteration

In addition to the slight modification to the electronic configuration, considerable time was spent coming up with a design that would allow the PCB to be mounted in multiple configurations including Thor Labs ER6 – cage assembly rods, standard M2.5 bolts, Thor Labs 60mm SM2 lens tube compatible cage plates [16], and Thor Labs 30mm cage plates. Figure 23 shows the important dimensions of the PCB that correspond to the mounting compatibility.

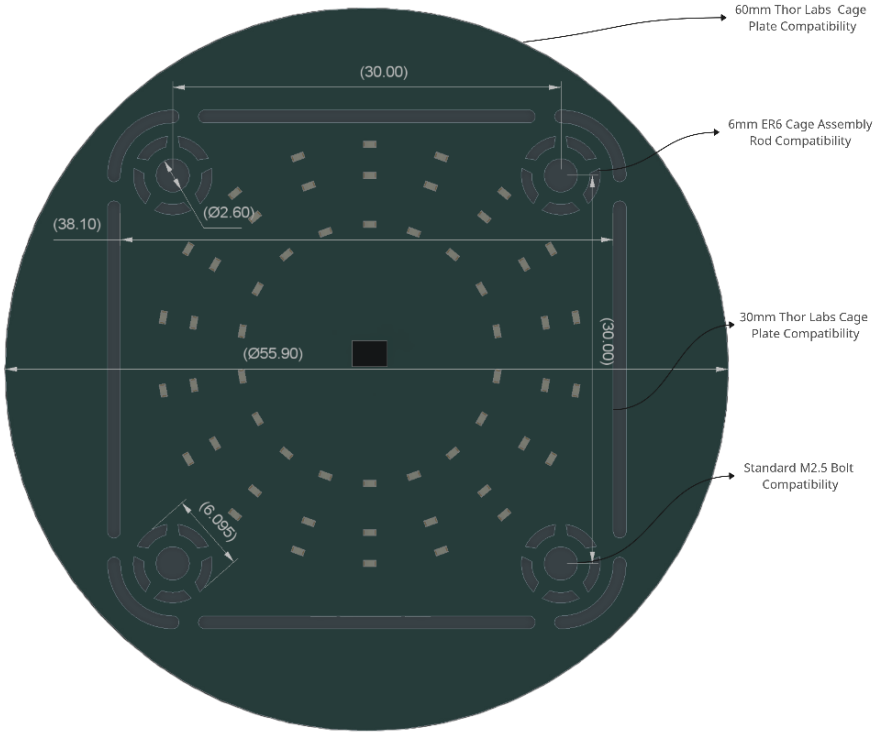

Figure 23: Diagram of the illumination PCB, with dimensions relevant to the mounting compatibility, with each cut labelled to show what it is needed for.

As in previous sections the geometry for the slot/ edge cuts was too complex for KiCAD natively so Fusion360 was used to draw the needed cuts as a DXF file that can be imported into KiCAD as a graphic and set to the edge cut layer. To position the imported edge cut layer accurately the KiCAD “Move Exactly” tool was used which allows an input of the X and Y offset from a given known co-ordinate, this then perfectly lines up the edge cut layer with the PCB design.

The idea to incorporate Thor Labs product compatibility originated from this GitHub repository [17], however this design was simplistic as it only incorporated one LED, is not compatible with Thor Labs 60mm SM2 cage plate, does not have an option for regular mounting with standard M series bolts and the LED driving components are on the same side of the PCB as the LED is, which has the potential to obstruct light coming from the LED at certain angles. Overall, by incorporating all the stated aspects that was not part of the original methodology to into the design allows for a more versatile system, allowing for use in other areas and experiments it was not specifically designed for. This is good practice engineering and is a common theme throughout this thesis, with making systems generalised and scalable for future use in other potential experiments. The front and rear of the final PCB are shown as a 3D model in Figure 24 and 25 respectively.

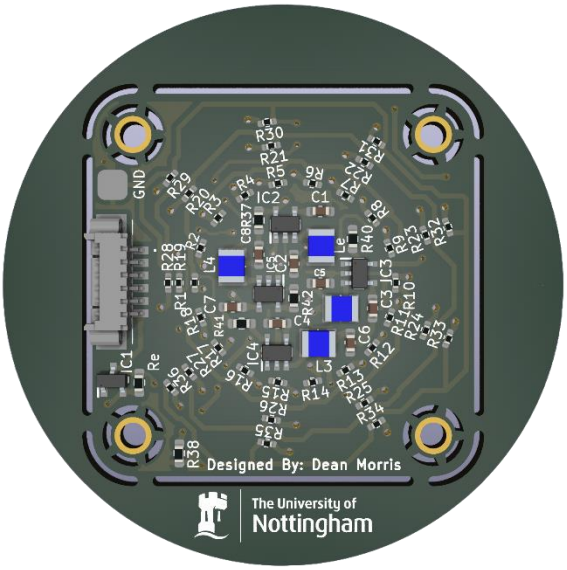

Figure 24: Front side of the PCB, that houses the driving components and connector.

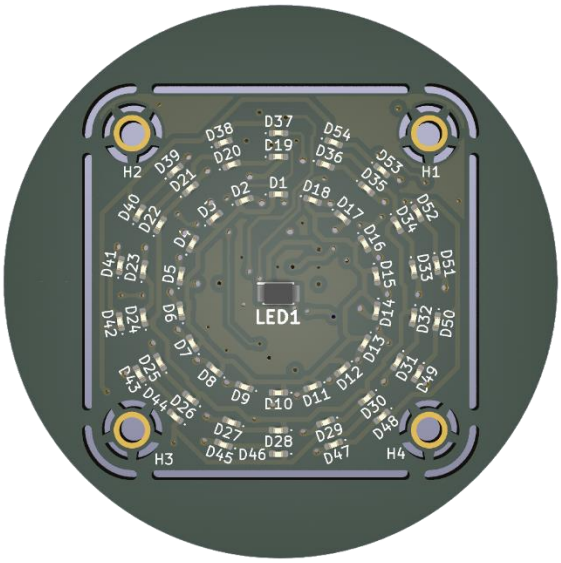

Figure 25: Back side of the PCB that houses the LED rings and central cyan LED.

2.2.2 CAD Development for Microscope Stand

The CAD development for the project had two main iterations, and both utilised a premade metal base that was repurposed from a test tube holder. Although not affective in terms of weight it was required so the stand does not tilt due to the weight of the 3D printed parts. This base would not be required if implemented on the CubeSat as there would be existing mounting points removing the need for a heavy base.

The first iteration of the microscope stand was printed by on a Fuse Deposition Modelling (FDM) printer. It utilises a rack and pinion system in a spur gear

configuration, this is used to move alter the distance to the lens to achieve optimal focus. The ability to alter the distance between the lens and the specimen holder was needed as there were multiple lens and lens configurations to test all of which have different minimum object distances and thus different optimal focus distances. Also, as part of the design was a universal tool holder with a bolt pattern of 41mm x 10mm with M4 bolts, this allowed for any part printed with this pattern to be attached to the stand.

This design was not used to hold the specimen slide, because it was used to hold the Raspberry Pi HQ camera facing downwards to take images to gauge factors such as magnification, FOV, minimum object distance, vignetting percentage and resolving power. After using this design to perform these tests it was evident that the design could use improvement, for example the gears would get stuck and wouldn't move smoothly along the rack gear. There were two theories why this was happening, the first was because it was printed on a FDM printer which due to the process has rough surface finished resulting in a higher amount of friction. Secondly, was if the gear is not moved perfectly parallel to the rack gear it would also get stuck, which as there is only one gear was easy to be not perfectly straight when moving up and down causing the part to tilt and thus get stuck. The final issue with the design is that the mounting uses brass heat set inserts which work by using a soldering iron to heat up the brass insert and then melt it into the plastic and let it set. The issue found with this was that although they work, they were not accurate due to being difficult to accurately control the final position as when in place and try to remove the soldering iron would cause them to move. This was a major downside as the accuracy achieved from 3D printing becomes meaningless as the final mount positions were a source of error. The CAD designs can be seen in Figure 26 and Figure 27.

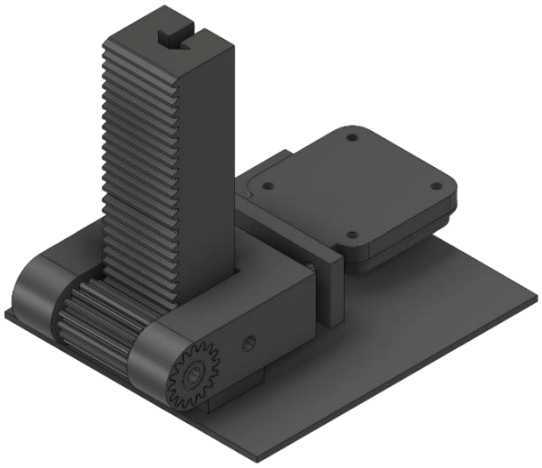

Figure 26: Original stand design from the back.

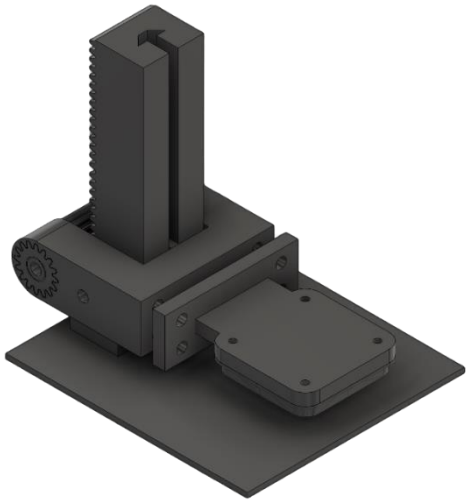

Figure 27: Original stand design from the front.

As a result of the short comings of the previous design, the second iteration aimed at addressing these issues. With the easiest being to overcome the rough surface caused by FDM printing, which was done by using a different 3D printing method called stereolithography also known as resin printing. This technology works in a fundamentally different way to FDM printing as FDM printing will heat up a plastic filament till it melts and then builds up layers of the melted plastic to create the model. Stereolithography works by using an ultraviolet (UV) curable liquid resin and the model is built up by a UV laser which scans across the bed plate in layers, this cures the resin from a liquid to a solid. As a result of the laser curing process resin printing can

be done at a far higher accuracy, with better surface finish which addresses the rough surface finish of FDM printing causing additional friction.

To address the issue with the previous design of the universal tool holder tilting as it moves along the rack gear, an additional gear was placed above the original gear so that there is now two contact points to help prevent the pivoting that occurs where the user tries to move the gear not perfectly parallel to the rack gear as can be seen in Figure 28 and Figure 29. In addition, the tolerance around the key slot was made smaller due to the higher precision available on a resin printer, also contributing to reduce the unwanted tilting.

The final change revolved around removing the need for brass heat set inserts, due to the accuracy issues discussed and because heat set inserts do not work on resin prints as it is a thermosetting plastic meaning it cannot melt after curing, were as FDM printers use thermoplastics such as polylactic acid (PLA). To address these two methods were used, for the universal tool holder, a slot was cut out of the side to allow nuts to be placed in the inside to mate up with the bolts to hold the specimen holder or any tool with the correct bolt pattern to be connected. Secondly, was the use of 3D printed threads, which are used for the M4 grub screws on the side of the universal tool holder.

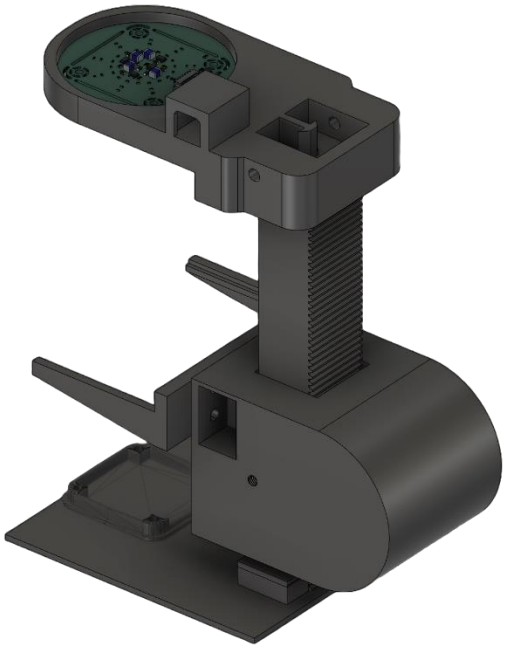

Figure 28: Updated microscope design from the back, with PCB holder and specimen holder.

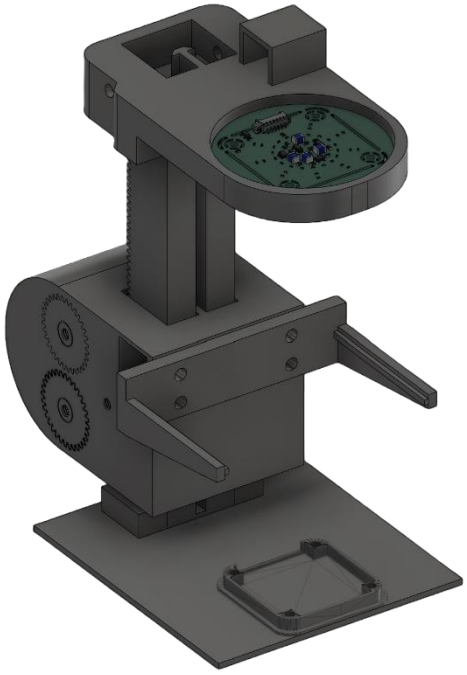

Figure 29: Updated microscope design from the front, with PCB holder and specimen holder.

The benefit of this design other than fixing the issues from the 1st iteration is that it still allows for FDM printed parts to be used in the design, such as for the specimen holder and PCB holder, as there was no reason to have these resin printed, as the additional accuracy is not required. Additionally, resin prints have the disadvantage of being heavier as not all designs can be made with an infill percentage as requires drain holes and is the reason why the resin printed parts in this design are 100% infill, so by being able to use FDM components for the specimen holder and PCB holder reduces the weight of the overall system.

2.2.2 Software Development

The software development was a large proportion of the project and consisted of code for two different systems, one was a GUI application designed to run on Linux and the other was the embedded programming for the Raspberry Pi 4B to control the microscope based on user control from the GUI. The software development for both systems took from November 2023 to February 2024, used two programming languages which were C++ and Python where C++ was the majority and Python was just used for raw image reconstruction and basic post processing and both systems combined were around 3000 lines of code. The GUI designed in a way that allows it to perform well for its intended purpose, but also is designed to be generic so can be used on other systems. Specifically, the GUI can be used to control any Raspberry Pi based imaging system that supports the libcamera command line interface, has an internet connection, and has a corresponding code for the Raspberry Pi to handle data received from the GUI.

2.2.2.1 GUI Development

A GUI was not originally part of the specification for this project and thus not spoke about in the methodology, however it became evident that it was needed as otherwise when the user wants to take an image a libcamera command would have to be entered in the terminal which is bad for testing as takes longer to perform tests and is worse for the user as if without this system it would be difficult for someone who did not develop the system to use the microscope.

To develop the GUI a C++ framework called QT was used paired with QT corresponding IDE Qtcreator. The reason for using the QT C++ framework is due to its ease of use when paired with Qtcreator for creating powerful yet simple to design widget applications. In addition, for the internet messaging protocol the Mosquitto MQTT library [18] was used for its ease of implementation and because it is a C/C++ library. To pair with the Mosquitto MQTT library a QT wrapper for the Mosquitto MQTT library was used called “QMosqMqttClient” by KostiantynBushko [19], this greatly simplified the MQTT integration into the GUI.

As the layout of the GUI was done in the modelling section, the overall design layout of the GUI had been completed and the development included systematically working through each of the widgets to add its implementation. This section comprised of a mass amount of sub-system testing as each widget needs its own implementation, and how it relates to other aspects of the code and thus each section (class/ function) was tested before progressing on, where the overview can be seen in Figure 16.

The diagram in Figure 30 shows a high-level overview of the entire software system, including the Linux GUI, intermediate functions, communication type to the Raspberry Pi and the communication type from the Raspberry Pi to the Linux PC where applicable and finally the corresponding function on the Raspberry Pi to handle the associated Linux PC functions. This was done to break down the complex system into a digestible diagrammatic form which is easier to understand. There are many expected challenges with this section such as, UDP socketing and others which will be discussed more in detail later in this section.

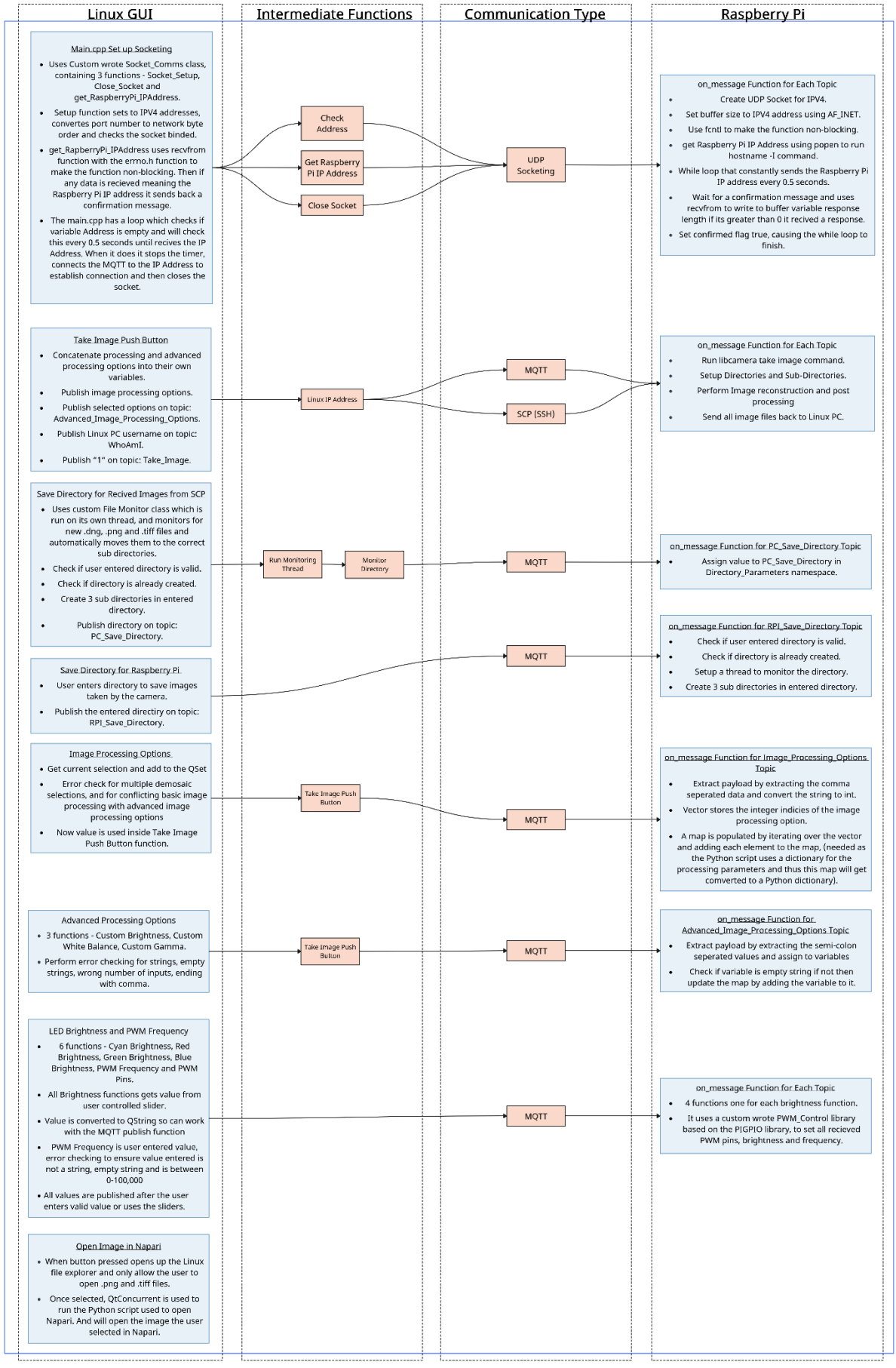

Figure 30: High level overview of the entire software system.

Now that the high-level overview has been shown for the entire software system, individual sub systems can be discussed. As most of the communication between the GUI and the Raspberry Pi happen over MQTT this was the first system that was worked on. As a starting point as the mosquitto MQTT library provides a command line interface a simple test was conducted to see if messages can be sent from the GUI to the Raspberry Pi where the GUI is running the *mosquitto_pub* command on a topic called “Test” and the Raspberry Pi is running the *mosquitto_sub* command to listen on the “Test” topic. This worked successfully and allowed for working on a C/C++ solution. This was simple due to the use of the Qt wrapper for the mosquitto library stated earlier, due to this a simple sample program was made which uses the *client.publishMesage* function of the MQTT wrapper with manually typed in IP addresses to establish communication. Same as before the Raspberry Pi still uses the command line interface to subscribe to the topic but now the publishing is being done via the C++ code. This was successful and meant all the needed MQTT code needed for the GUI side was set up and ready for use in the main GUI project.

2.2.2.1.2 Custom Wrote Classes

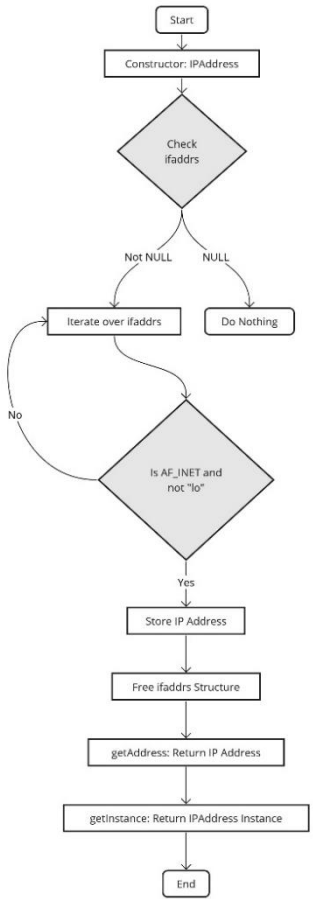

Figure 31: Flowchart showing the operation of the IPAddress class.

Even though the MQTT code is now ready to use, it is currently operating under hard coded IP addresses which is subject to change if for example it is used on a different network or if the devices aren’t assigned a static IP address by the router. Because of this issue a class custom class was wrote called IPAddress.cpp and IPAddress.h which automatically gets the IPV4 address of the GUI but is scalable to work on any Linux based system, the operation of the class is shown in the flowchart in Figure 31.

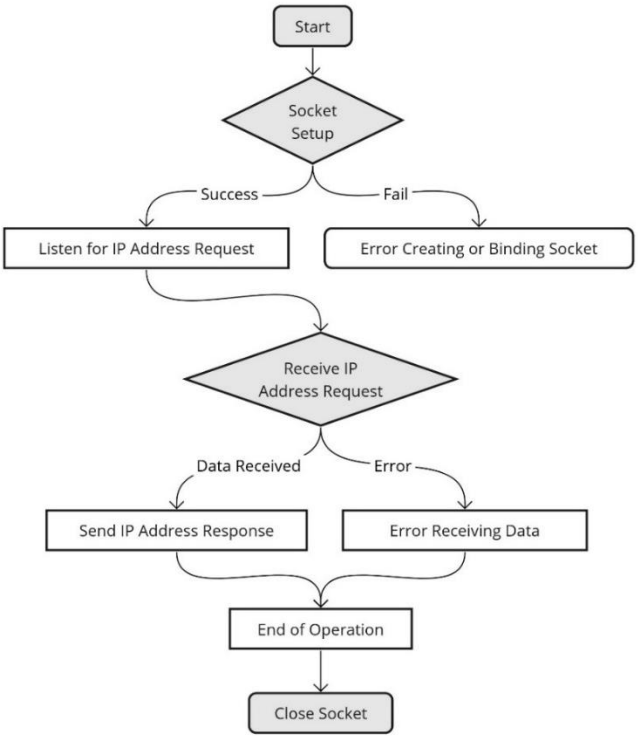

Figure 32: Flowchart for the Socket_Comms class which shows the class functionality.

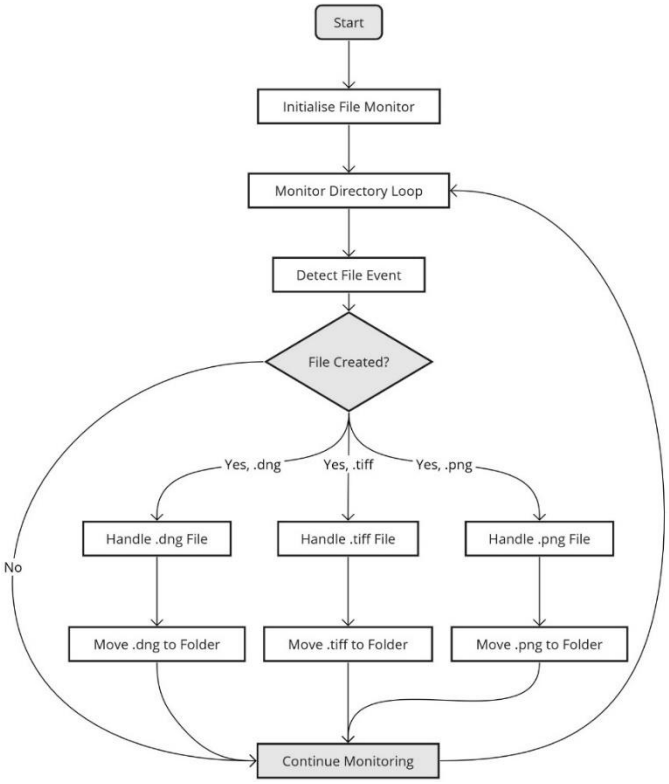

Figure 33: Flowchart showing the operational overview of the File_Monitor class from a high-level perspective.

In addition to MQTT, UDP socketing is also used for communication and was difficult to implement as web socketing is not a straightforward endeavour and took a while to get working. The reason why UDP socketing was needed is because for MQTT communication the two devices (in this case the Linux PC running the GUI and the Raspberry Pi) need to know each other's IP address to establish a connection. Thus, UDP socketing was used to send the Raspberry Pi's IP address to the Linux PC so the MQTT communication can be setup. Figure 32, outlines the function of the custom wrote class to do this in the form of a flowchart.

Another custom class was write called File_Monitor, this class handles the functionality for multiple aspects of file management. Firstly, it monitors the Linux PC save directory which was inputted by the user, it constantly checks to see whether a new file was created, facilitated by the fact it is run in a separate thread and thus doesn't halt or interfere with the rest of the code. If it does detect a new file in the directory, it checks if the file ends in a .dng, .tiff or .png, if the new file is none of these three then it goes back to monitoring the directory. However, if it is one of the three file types then it moves the file to its respective subdirectory, this process is summarised in Figure 33 as a flowchart, the moving of the files to the respective subdirectory wasn't strictly needed however it allows the user to navigate to the files more easily as are already pre-sorted. To be able to test this system independently, a .dng, .tiff and .png were manually placed in the directory the code was monitoring, the system was successful as it moved all files to the respective subdirectories, meaning this class was ready for integration with the full GUI software system.

2.2.2.1.3 Entire Finalised GUI Software Implementation

With all the dependencies of custom wrote classes the main GUI code functionality can be discussed, starting with the “on_Directory_PC_Line_Edit_returnPressed” function. This function is designed such that when the user enters a directory to save the images to, once received from the Raspberry Pi it performs various error checking on the entered directory such as if the directory if the directory starts with the user’s home directory. This verification is crucial as users may input a syntactically valid path that, however, inaccurately references a different user's home directory. For instance, while /home/Dean constitutes a legitimate directory for a user named "Dean," an input such as /home/Test despite being structurally valid would lead to a system crash. This is because the directory, attributed to a non-existent user "Test" in this context, fundamentally does not exist, thereby precipitating a crash. In addition to this check the function also checks for if the path is not absolute and also checks if the path currently exists on the system, if it does then it uses the existing path, otherwise it creates the directory the user inputted and then creates three sub directories one for .dng, .tiff and .png respectively. After all the error checking to ensure the entered directory is valid and the directories have been created the directory path is published via MQTT to the Raspberry Pi on the PC_Save_Directory topic. A more detailed overview of the function is shown in the flowchart in Figure 34.

A very similar function called “on_RPI_Directory_Location_Line_Edit_returnPressed” it provides the directory for the Raspberry Pi to save the images to, but this function has no error checking as it isn’t possible to do error checking on the directory parameters for another device, thus once the user inputs a directory it is just sent via MQTT to the Raspberry Pi.

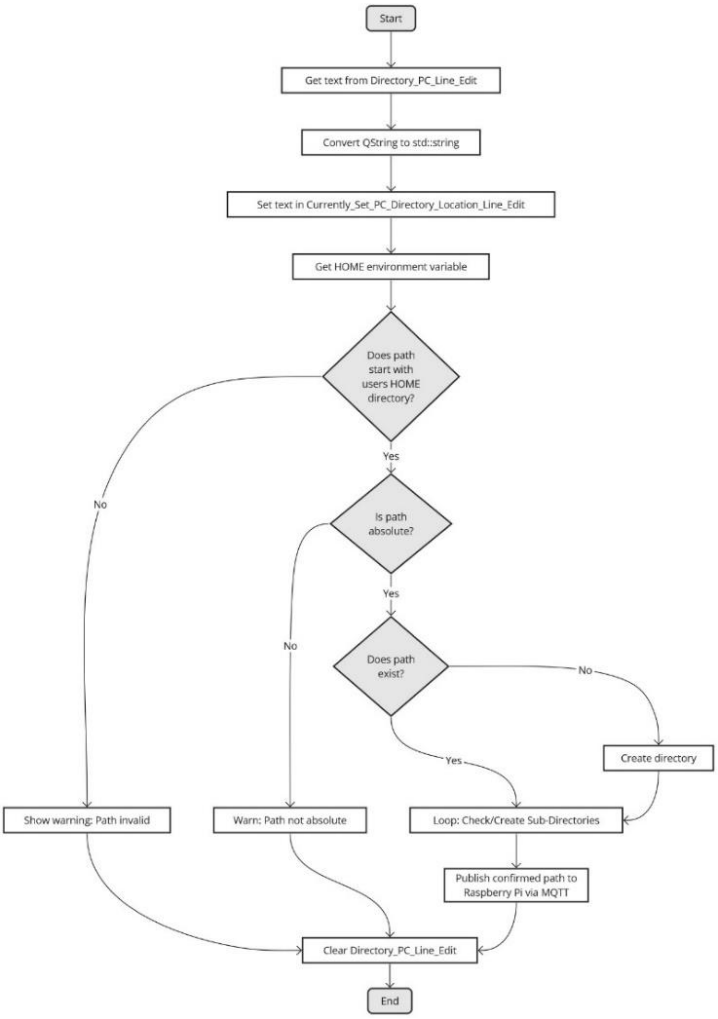

Figure 34: Flowchart showing operation of the "on_Directory_PC_Line_Edit_returnPressed" function.

Moving away from the file directory structure, within the GUI the user has the option to apply different post processing options and demosaic algorithm type (used to reconstruct the image from raw data), to do this in Qt a list widget was used. As it allows the user to select multiple options at once all while being able to keep track of the current selection, the implementation is in a function called "*on_Image_Processing_listWidget_itemSelectionChanged*". Overall, this provides the functionality for the user to select post processing options integrated with appropriate error checking such as not allowing multiple demosaic algorithms to be selected and not allowing the user to enter values for custom brightness or white balance if they already have the automatic brightness or white balance selected, a more detailed overview can be seen in the flowchart in Figure 35.

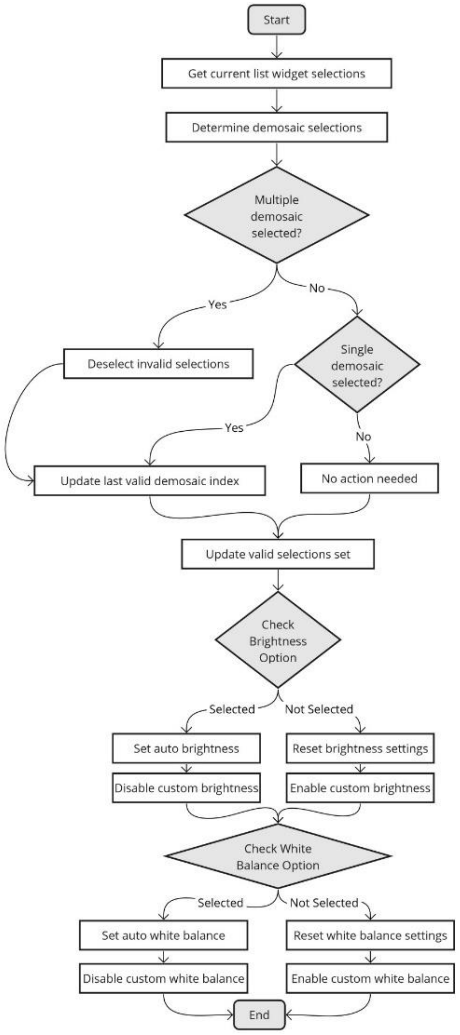

Figure 35: Flowchart showing the operation of the "on_Image_Processing_listWidget_itemSelectionChanged" function.

Following on from the previous function, the following functions expands upon the post processing options, the three functions are for custom gamma, custom white balance, and custom brightness. The options expand upon the Boolean post processing options allowing the user finer control by inputting numerical values. All functions provide error checking on the user input and if all error checks pass then it updates the values to what the user entered, ready to be sent to the Raspberry Pi via MQTT when the press image button is pressed. Further detail can be seen in the flowchart in Figure 36.

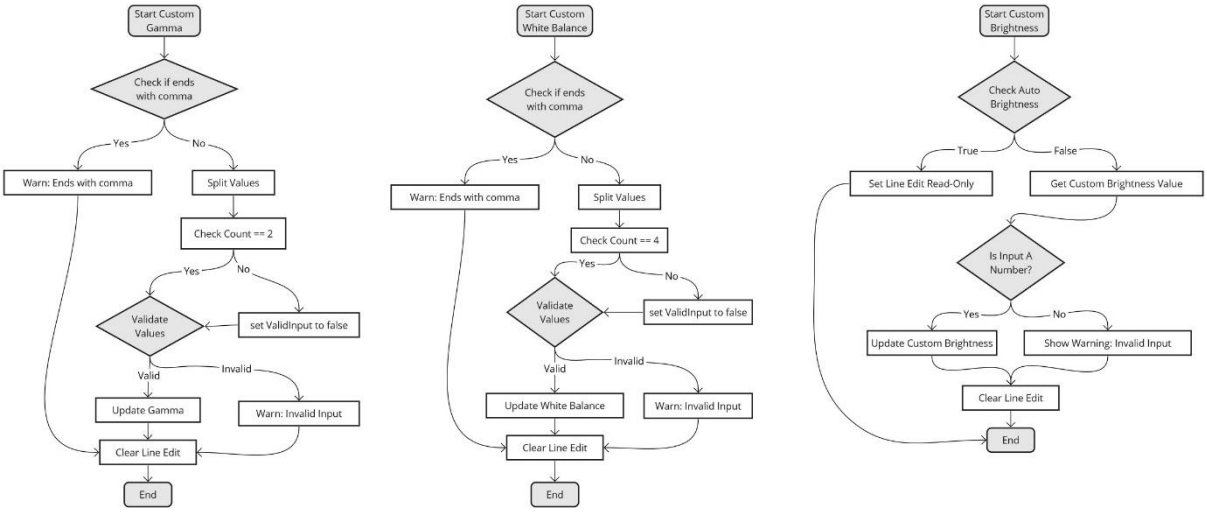

Figure 36: Flowchart showing the operation of the custom gamma, white balance, and brightness functions.

All previously mentioned functions are used to control software aspects such as performing post processing on an image. The following functions differ as they are used to control various aspects of the illumination board through sending commands via MQTT to the Raspberry Pi in order for the Raspberry Pi to control the illumination board. To do this the user must state what PWM pins are being used on the Raspberry Pi which is used for the brightness control of the LEDs. This is both a feature and a limitation as it allows flexibility of the system, but also requires the user to have to know more complicated details of the system than they would have otherwise. If given more time this issue could have been overcome by adding profiles to the GUI software so the user only ever has to do this once and the GUI software would remember these parameters.

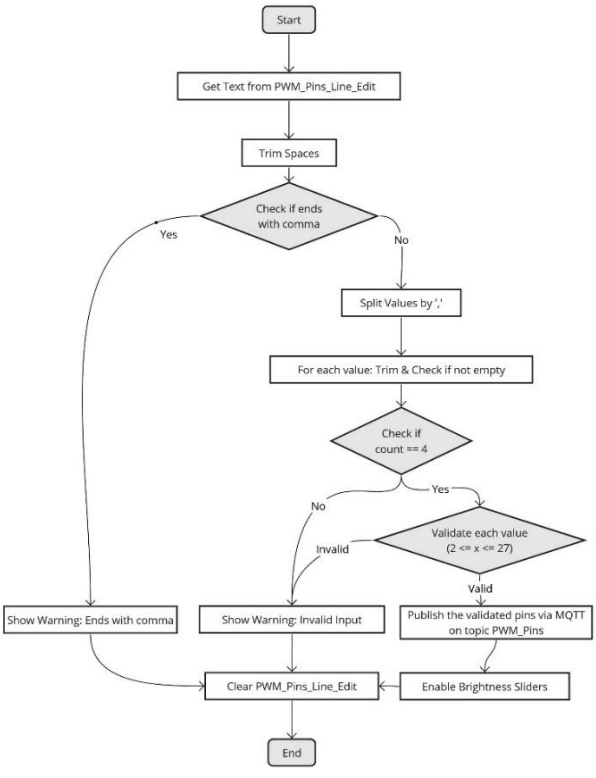

Figure 37: Flowchart showing the operation of the on_PMM_Pins_Line_Edit_returnedPressed function.

The function shown in Figure 37, takes the user inputted PWM values and performs a variety of error checks, such as if user input ends in a comma, if the count is equal to 4 and if the values entered are between 2 and 27 as these are the only valid pin numbers for the Raspberry Pi. Once validated the user inputted pin values are sent to the Raspberry Pi via MQTT.

The last needed function to initialise the PWM before the brightness can be controlled is setting the PWM frequency, the range for this is 0-100kHz as that is more than enough bandwidth for this application and is what is hard coded in the custom PWM library I wrote. The main purpose of this function is to take the users input, perform error checking and if the error checking comes back as a valid entry, then the value is published via MQTT on topic PWM_Frequency, and is received by the Raspberry Pi. The flowchart for this function is shown in Figure 38.

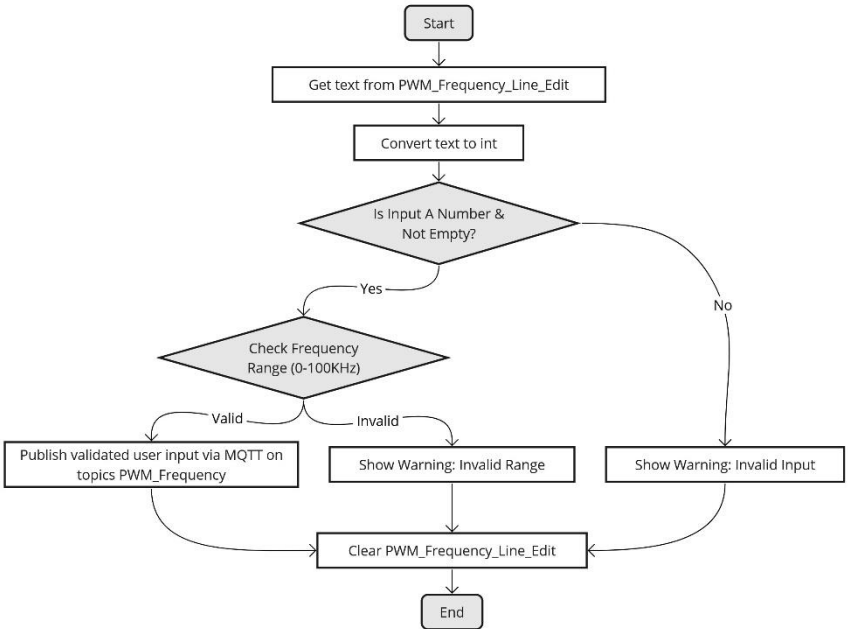

Figure 38: Flowchart showing the operation of the on_PWM_Frequency_Line_Edit_returnPressed function.

With all the functions developed needed to initialise the PWM, the control for the LED brightness of the different colour LEDs on the illumination PCB can be developed. It does this by the user entering a percentage value for the brightness, which is sent to the Raspberry Pi via MQTT, where the Raspberry Pi will handle the hardware control. This adjusts the brightness as the constant current IC used on the PCB has an enable pin, which when PWM modulated the brightness can be controlled through the duty cycle of the PWM, as this determines how long the LEDs are on per PWM time period. Figure 39 outlines the functionality for the cyan brightness function however the other colours functions are identical with different variable names and MQTT publish topics and thus only one flowchart is provided here to explain this.

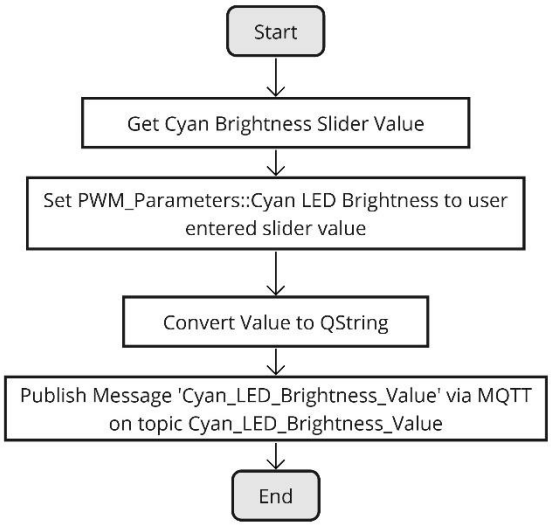

Figure 39: Flowchart showing the operation of the on_Cyan_Brightness_horizontalSlider_sliderReleased function.

2.2.2.2 Raspberry Pi Software Development

The Raspberry Pi acts as the interface for the Linux GUI app to interact with various hardware such as the illumination board and the camera. In addition to acting as the interface between the GUI and the hardware it also performs various software processing such as reconstruction of images from raw data captured from the camera into useable file formats (.tiff and .png) and performing the post processing the user selects from the Linux GUI along with other more minor tasks that will be spoke about more in detail later in this section.

Once implemented in the CubeSat there will be no Linux GUI and thus won't have to handle any internet messaging protocols such as UDP socketing and MQTT to establish communication with the Linux GUI. Instead, it acts as the main processor and will automatically loop through various processing options at set intervals to capture raw images, process them into viewable formats and store them on the Raspberry Pi's SD card.

2.2.2.2.1 Custom Written Classes for the Raspberry Pi

Due to the scalable way the code for the Linux GUI was written, there are only two independent custom classes for the Raspberry Pi that were not used in the Linux GUI, as the IPAddress class and the File_Monitor class were also used in the Raspberry Pi software but have already been explain in section 2.2.2.1. And a custom wrote python script which was compiled into C through the use of Cython and then into a .SO library file so the Python program can be called and used from inside the main CPP file.

The first custom wrote class called "DateTime" gets the current date and time or when the function was called. This was needed as when an image is taken on the Raspberry Pi, it is saved in the user selected directory, so the DateTime class is used to append the current date and time of when the image was taken, which means if multiple images are taken then the previous image won't be overwritten with the new image. In addition, it also improves the users ability to find the wanted files as the user can search for a time an image was taken in the file explorer and find the needed files easily, with this also being simplified as the raw and processed images each have their own sub-directory, where the sub directories are created in the same way as done for the Linux GUI software. Figure 40 shows the two functions in the class in the form of a flowchart. To ensure this system was working a simple sample code was used separate from the main code, where both member functions were used to assign the current date and time to a string variable, and this was printed to the screen with the cout command. This was a success as it printed the correct time to the screen down to the second.

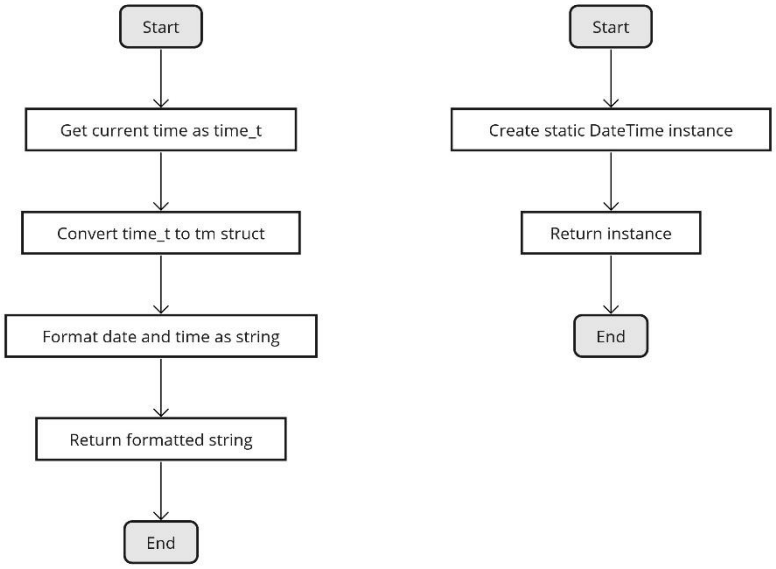

Figure 40: Flowchart showing the operation of both functions in the DateTime class (getDateTime and getInstance) where getInstance is used to simplify the implementation in the main code as don't have to explicitly declare an object to use the getDateTime member function.

The final custom class for the Raspberry Pi, is more akin to a library, as it provides a custom wrote wrapper around the PIGPIO library to simplify the PWM initialisation and PWM parameters needed to control the brightness of the LEDs. The library states the functions in terms of LED parameters such as brightness for each colour, as this makes the main code much more readable as is clear that the code is controlling a certain LED colours brightness rather than an arbitrary line of code which changes the duty cycle of a stated GPIO pin.

In addition, the library was written in a scalable way by providing get and set member functions for all parameters such as LED brightness (PWM duty cycle), PWM pin and PWM Frequency and an additional function to set the brightness of all LEDs at once. This allows for potential future improvements in the redundancy of the system, as if it is detected that one or multiple of the LED rings are not working as expected it would be possible to have a handle error function which could go through and check all of the currently set values and make changes to the system based off some pre-defined logic. Also, if redundant wiring is provided for the PWM connection to the illumination PCB by having more connected PWM lines than needed, then this error function could also change the GPIO pin controlling the PWM parameters for each LED ring.

For the set PWM pins and brightness there are four copies of each function but for the different colour LEDs so only one is shown per function type, and get functions are not shown as they just return the value requested. Shown in Figure 41 is the operation of the "set_Red_PWM_Pin" function, it checks if the pin is valid, by ensuring the GPIO number is between 2-27, and sets the pin if valid or outputs an error for invalid GPIO number.

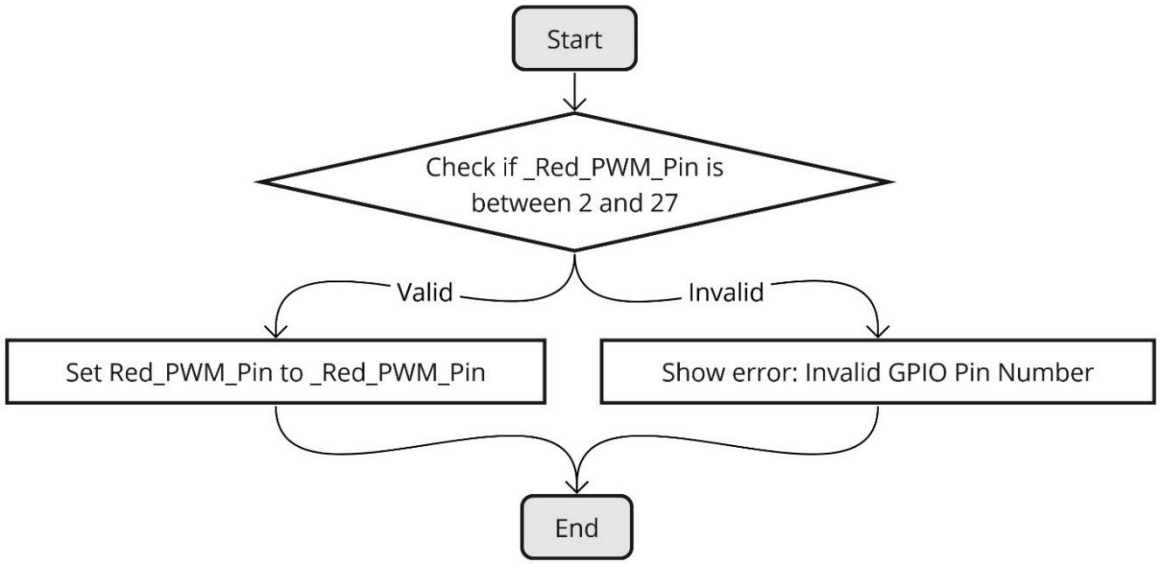

Figure 41: Flowchart showing the operation of the set_Red_PWM_Pin function.

Once the PWM pins have been set, the PWM frequency also needs to be set the function for this is called set_PWM_Frequency and provides simple error checking that the inputted frequency is between 0-100KHz and will set the frequency for all LED colour channels in this function if the error check is valid, if not then an error is outputted fir invalid PWM Frequency, as shown in the flowchart in Figure 42.

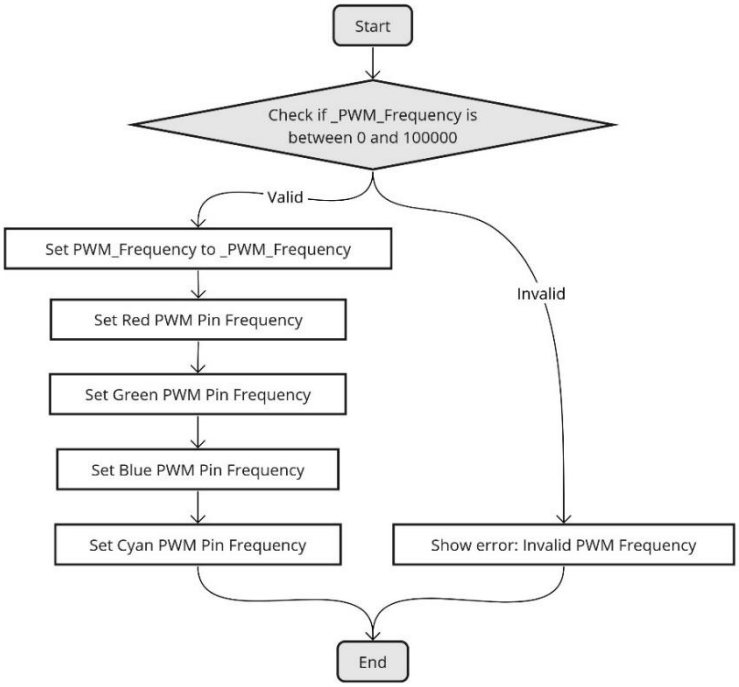

Figure 42: Flowchart showing the operation of the set_PWM_Frequency function.

Now with the initialisation done, the intended purpose is to be able to control the brightness of the LEDs which is done by adjusting the duty cycle, so once the user has inputted the value from the GUI and is received by the Raspberry Pi it performs an error check to ensure its between 0-100 if it is then it sets the brightness by setting the PWM duty cycle through the underlying PIGPIO library function calls. If the error check was invalid, then the error invalid brightness value is shown. The flowchart fir the function is shown in Figure 43.

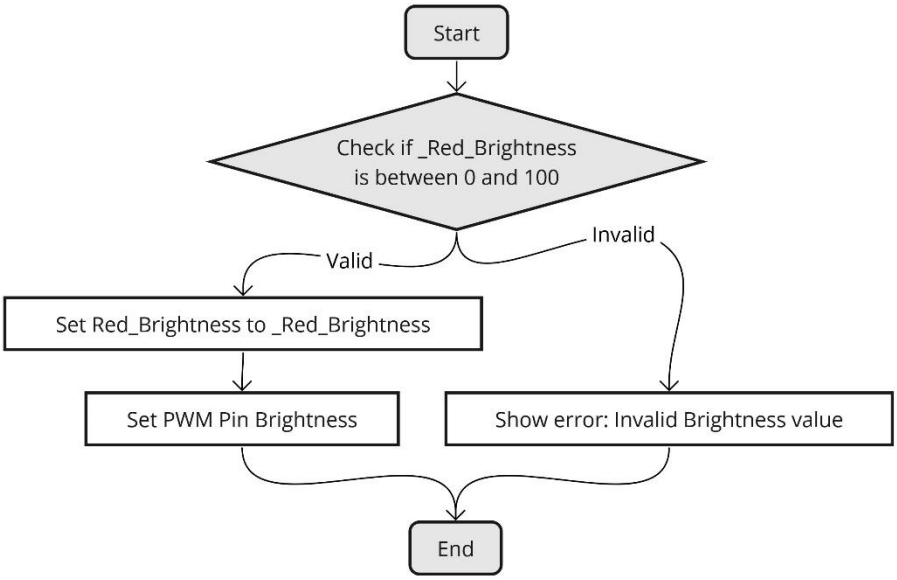

Figure 43: Flowchart showing the operation of the set_Red_Brightness function.

2.2.2.2.2 Python Program to Perform Raw Image Reconstruction and Image Processing

As originally defined in the methodology in the project proposal the original plan was to use OpenCV for the image processing however as mentioned before in section 2.1.1 OpenCV only supports a maximum bit depth of 8 bit for the image quality, where the Raspberry Pi HQ Camera can take 12-bit depth images. Thus, by using OpenCV meant sacrificing on image quality and as a result was not viable.

The developed Python code uses rawpy PIL and ImageIO as the main libraries, where rawpy is used for the raw image reconstruction by using demosaic algorithms and other post processing and PIL is used to convert the .tiff generated by rawpy into a .png file format. There are three Python functions and are all highly abstracted from the user, as the user must input parameters in the GUI which are sent to the Raspberry Pi via MQTT and that is all that is required from the user, Figure 44 outlines the functionality of the entire image_processing Python program that contains three functions (*process_image, _process_image_c and convert_options*).

The process_image function is the function that is called from the C++ function through the .SO library, it passes the filename and a dict to the function by calling the convert_options function, where the dict contains the user inputted processing parameters such as the type of demosaic algorithm, and post processing options, the C++ program can pass a dictionary as the C++ code uses a map which is converted to a dict by using the Python.h library for C++. Once the parameters are received the filename is encoded with utf-8 encoding, (it was needed as otherwise an error would occur because of the differences between C++ char* and Python strings), and the *_process_image_c* function gets called.

The _process_image_c function is where the rawpy, PIL and imageio implementations are contained, and processes the file from the filename parameter passed to the function.

Finally, the convert_options function converts all the values in the dictionary to the appropriate type for use in the rawpy function raw.postprocess.

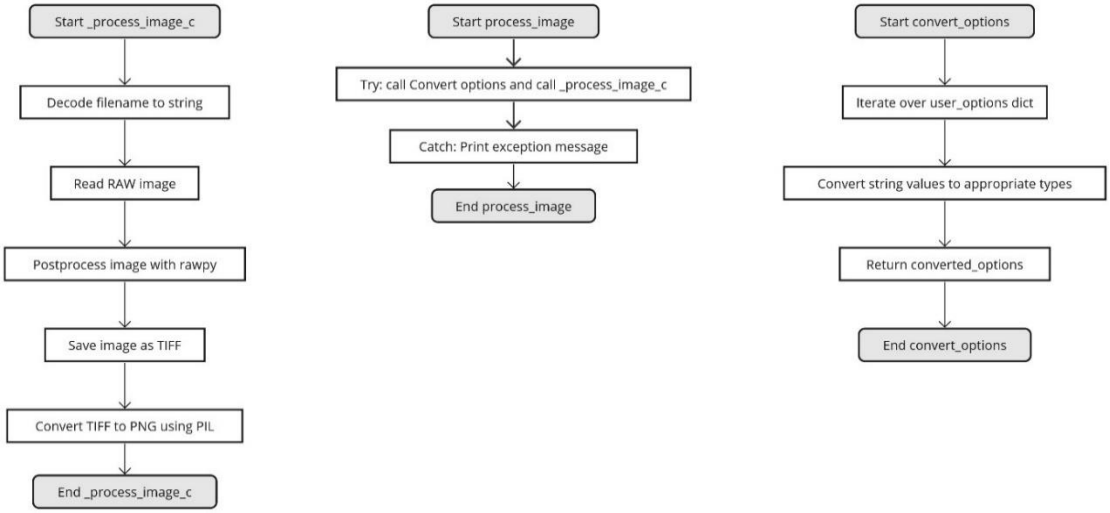

Figure 44: Flowcharts outlining the operation of the process_image, _process_image_c and convert_options Python functions.

Figure 45 shows the C++ function for *Raw_Image_Reconstruction*, this function uses the Python.h library to allow for the previously developed Python function which deals with the raw image reconstruction and post processing options, to be called from a C++ function. What the function does is it checks if python is initialised and if not will initialise it, then the function tries to import the image_processing.so library which was created from compiling the Python code into C using Cython. Finally, after checking if process_image is callable it makes an empty dictionary and populates it with the processing options from the map function, prepares the Python function for execution from C++ and calls the function.

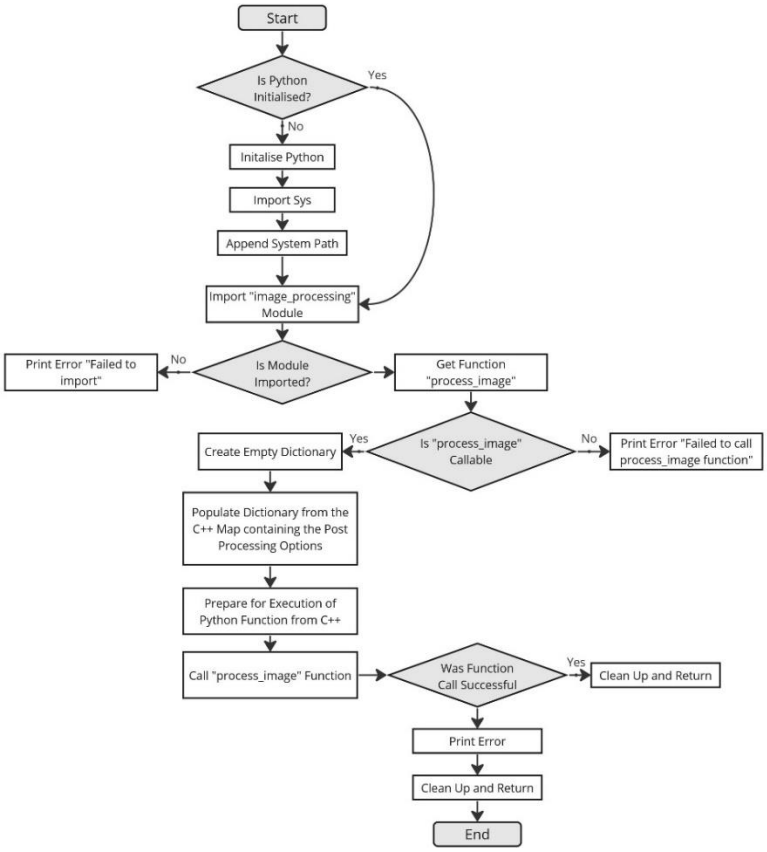

Figure 45: Flowchart showing the logic of the Raw_Image_Reconstruction function.

2.3 Final System Testing and Validation

2.3.1 Specification Validation

Specification Point	Pass/Fail/Partial
Custom PCB for the illumination system with 3 LED rings (red, green, and blue) with central cyan LED.	Partial
Choose components for the optical setup including lenses and camera.	Pass
Develop C++ code to run automatically on Raspberry Pi 4B to take images, post process and store to SD card.	Fail
Design of custom microscope enclosure with ability to manually adjust specimen height.	Pass
Final system has ability to take brightfield and darkfield images using custom illumination PCB in condenser free setup.	Pass
Integrate sub-sections including PCB illumination system, optical system and software design.	Partial
Stretch goal to design high level easy to use GUI to control microscope wirelessly using MQTT.	Pass
Stretch goal to have processed and non-processed images sent back to PC running the GUI automatically.	Pass
Stretch goal to provide a force analysis of the custom enclosure and PCB design to withstand rocket launch forces	Fail
Cost for project less than £250	Fail

Table 14: Summarising the specification points with an associated pass/fail or partial criteria.

The custom PCB was only a partial success as it was functional enough to have one ring working to take images with, however due to the small components used and solder paste without a stencil it is likely the lack of functionality comes with my assembly of the board and may have shorts from solder paste underneath components which is not visible, if professionally manufactured and populated it is expected the design would work as intended. This impacted the project as it meant to switch between modalities the PCB illuminator had to be physically set to different vertical heights, where as if the PCB had full functionality it would not have to be moved as each ring of the PCB is responsible for a different modality, so by simply turning one ring off and turning another on would have allowed for switching between the modalities.

The final partially met specification point is the integration of all sub-sections, this is due to the PCB not having all the functionality as stated above. Even though all of the code worked to control the brightness, set PWM pins and set PWM frequency as this couldn't be used to control the PCB due to the hardware issues meant this all could not be fully integrated.

The only technical failure of the project that was not a stretch goal was the C++ code to automatically run on the Raspberry Pi 4B was not done. This was because the lens modelling took a long time to complete and was not part of the original risk mitigation or Gantt chart as it was not expected for the values in the datasheet would be inaccurate. Also, as the development of the GUI and corresponding Raspberry Pi code to facilitate this took a long time to develop, however most of the code developed on the Raspberry Pi side would be directly applicable in a code to take images and post

process automatically as the specification set out. This didn't have a large impact on the project in terms of getting results and testing the microscope however it did affect the aim of the final system as the goal for project was to implement the microscope on a CubeSat and thus requires an automated imaging system. Due to this not being developed this would not be possible with the currently developed code for the GUI and thus would require the development of code to facilitate this.

Overall cost for the project was not kept within the £250 budget but was went over with permission, the reason for not meeting the limit was the cost of the lens kit from RobotShop [8] with a cost of £127.65 stated in Table 15. However, when considering only two of the lenses were used in the final version this brings the cost of the kit down to £51.06 and makes the total for the project £283.81 which is closer to the set budget.

Item	Cost
Electronic Components for PCB	£104.50
Raspberry Pi 4B	£53.23
Raspberry Pi HQ Camera	£48.16
Lens Kit	£127.65
PCBs	£26.86
Total Cost	£360.40

Table 15: Summarising the cost of each item or components used in the project.

Finally, the force analysis of the microscope enclosure and PCB was not conducted, not due to any technical difficulty but because of time restraints due to additional sections that were not in the original Gantt chart such as the lens modelling and length of time required to develop the GUI.

2.3.1.1 Results

As part of the final system testing the illumination PCB was used independently of the developed imaging setup with the Raspberry Pi and HQ camera. Instead, the images were taken on a Cairn Research OpenFrame microscope [20] to allow for validation of the illumination PCB to take bright field, dark field, and phase contrast images a diagram of the modalities is shown in Figure 46 where brightfield that isn't shown the light beam can either focus inside or outside of the phase ring.

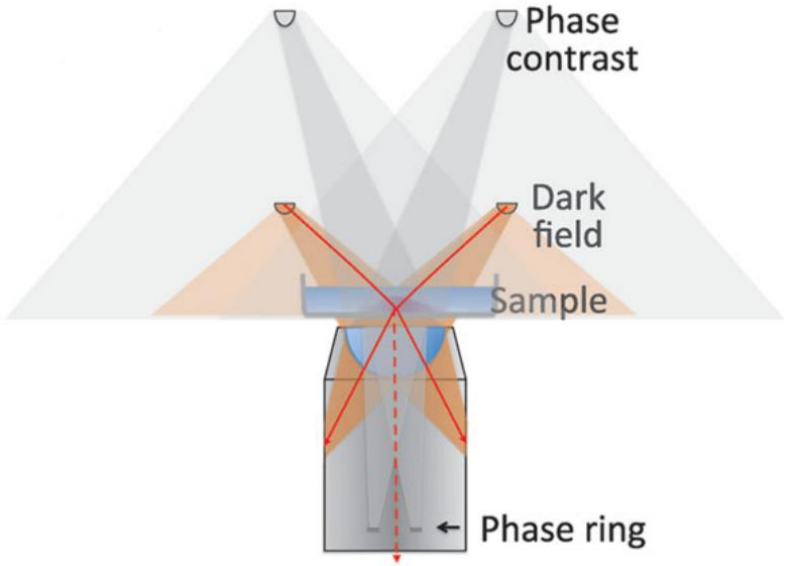

Figure 46: Diagram showing the modalities, except brightfield where the light would pass straight from the illuminator into the lens.

Figure 47 shows the results of imaging Buccal Epithelial cells for all three modalities, where it is shown that both bright field and dark field produce a similar level of detail in terms of the visible features of the cells in the image, however the dark field image has better contrast due to the dark background, caused by the light from the illuminator entering the objective at an extreme angle such that it does not transit the objective [5]. Finally, phase contrast performed the best out of the three modalities, however this modality won't be possible on the developed system with the Raspberry Pi HQ camera with the currently used lenses. This is because for the phase contrast modality the lens requires a phase ring, which only come in lenses that cost more than the budget for this project, thus the only phase contrast images taken are taken on the Cairn Research OpenFrame microscope, using my custom illuminator aligned for the three modalities by condenser-free vertical adjustment [5].

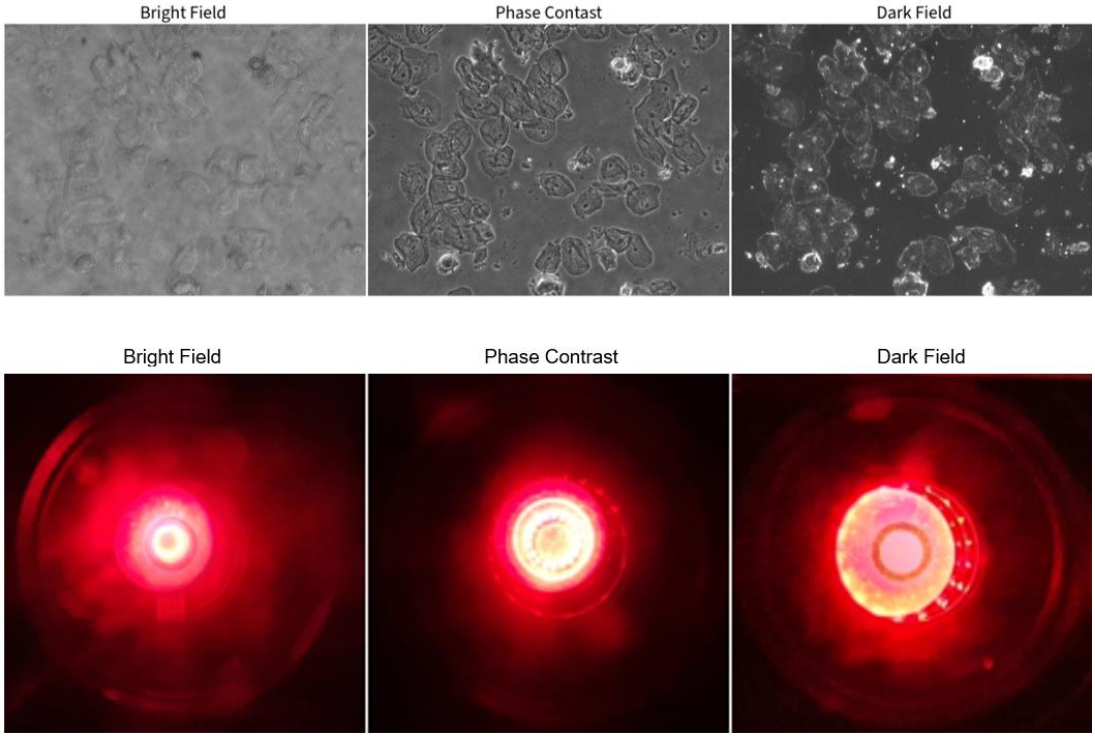

Figure 47: Images for all three modalities of Buccal Epithelial cells, while showing the alignment of the LED ring relative to the lens.

In addition to the Buccal Epithelial cells imaged on the Carne Research OpenFrame microscope, the *C. elegans* were also imaged using my custom illuminator, for all three modalities as shown in Figure 48. These images also meet the specification of µm resolution as shown by the scale bar, however this was as expected due to being taken on the Cairn Research OpenFrame microscope with a 10X objective and thus no need for a quantitative analysis. However, it did allow for verification of the custom illuminator to take the images in a condenser free setup.

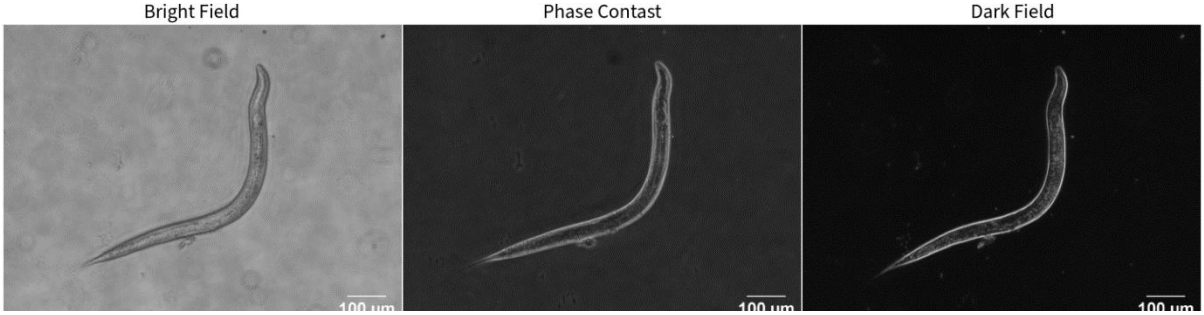

Figure 48: Image of a C. elegan taken in the Cairn Research OpenFrame microscope with a 10X objective for all 3 modalities.

With the illumination PCB verified to be working and can produce images for all of the needed modalities in a condenser free configuration, the system was integrated onto the developed optical setup with the Raspberry Pi and HQ camera with the M02/M01 lens pair used for infinite conjugate imaging. Shown in Figure 49 is the testing setup for all of experiments including the imaging of the *Diatoms*, *C. elegans,* and Buccal Epithelial cells. As stated in section 2.2.2 the microscope stand comprises of a universal tool holder, which is setup with a specimen holder set to a hight such that the specimen slide is 8mm above the camera lens, as this is the minimum object distance as found from the modelling results in section 2.1.5.3.

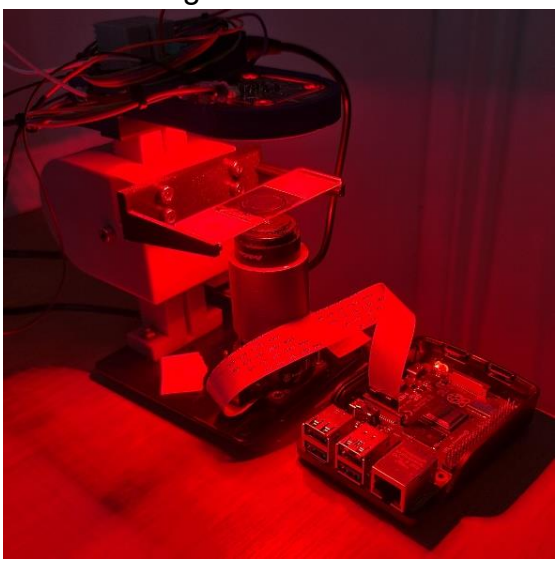

Figure 49: Testing setup for imaging Diatoms, C. elegans and Buccal Epithelial cells using the developed microscope stand, custom PCB illuminator and optical setup based on the output of the merit function from section 2.2.5.6.2.

Shown in Figure 50 is both the brightfield and darkfield modality when imaging *Diatoms where a Diatom* is silicified algae [21]. As expected, the darkfield image contains more detail due to the contrast enhancement it provides. It does this as the light from the PCB illuminator is at such an oblique angle that only the light that interacts with the *Diatoms* is able to enter the objective, resulting in bright features from the *Diatoms* on a dark background. Utilising this contrast enhancement is especially evident for imaging *Diatoms* due to their transparent silica cell wall [21], as brightfield imaging tends to struggle with transparent specimens that are unstained as the light from the illuminator passes through the specimen into the objective resulting in a bright background and thus harder to see finer details of the specimen. For reference the scale bar for the images was calibrated from the 500µm increment ruler from figure 7 for the M02/M01 lens pair in section 2.1.5.3. The brightfield image was taken with 1/333 seconds exposure time and darkfield was taken with 1/17 seconds exposure, the reason why the exposure time for brightfield was so low was because the PCB didn't fully meet specification and thus could not control the brightness, thus without lowering the exposure time drastically the images were being overexposed. Also, as part of the test conditions all images were taken in a dark room with no external light source and due to PCB hardware issues discussed to instead of switching between LED rings, darkfield condition is met by moving the illuminator vertically until the camera displays a black background, and converly brightfield condition is when the background is bright.

To quantitively demonstrate the performance of the developed microscope system, the red profile plot of an edge transition of a *Diatom* was taken for both the brightfield and darkfield images and normalised as shown in Figure 50, where the specific Diatom used for the analysis is circled in blue. The red profile plot for the brightfield image had a rise coefficient of -10.47, where the negative is because in bright field the specimen is dark and the background is bright thus the curve is opposite to what has been shown in the section 2.1.5.6. and the red profile plot for the darkfield image has a rise coefficient of 12.94. When compared to the normalised red profile plot from section 2.1.5.6 when imaging the Samsung Galaxy Tab S6 Lite tablet which had a rise coefficient of 13.72, shows that the system performed closely to the modelled performance, and the discrepancy is likely due to the larger amount of red light when imaging the *Diatoms* that what was present from the light output of the tablet, as the background will be brighter than the column channels in-between the pixels of the tablet.

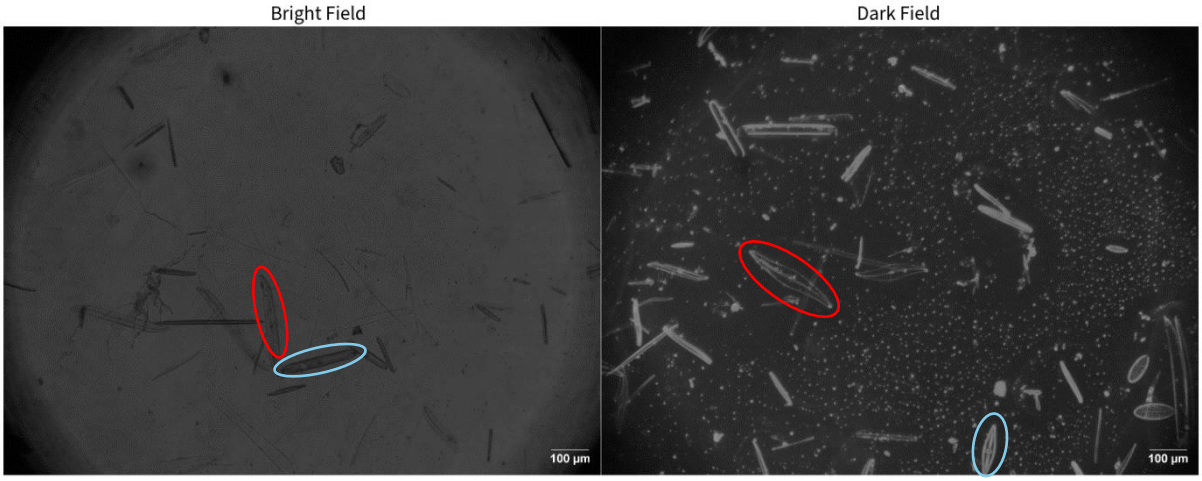

Figure 50: Image shows both brightfield and darkfield modalities when imaging Diatoms on the developed microscope setup, including custom stand, illumination PCB and lens configuration.

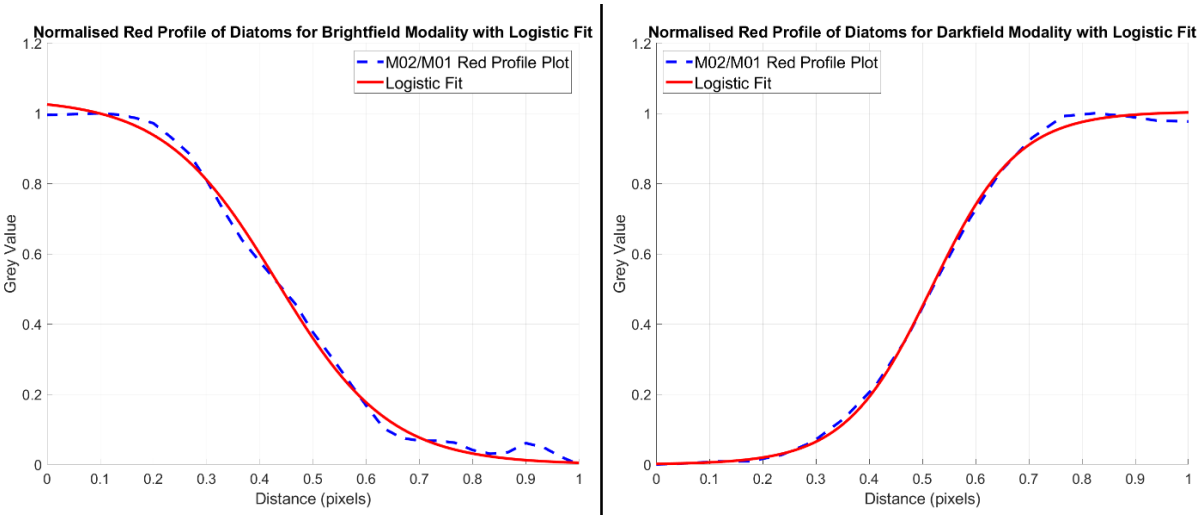

Figure 51: Two graphs showing the normalised red profile plot for both brightfield and darkfield modalities for the images in Figure 50.

Also shown by the profile plot is a mostly linear rise meaning that chromatic aberration is low in the images, which is also as expected as from the modelling section blue light the colour band that was focusing to a different plane, thus as this imaging is just using red light and no blue light is present results in little to no chromatic aberrations present.

To compare the measured resolving power to the modelled resolving power from section 2.1.5.4.1, after setting the scale in ImageJ for the M02/M01 lens pair the finest detail was measured where the specific *Diatoms* for the analysis are circled in red. From this the width of the cuticle was measured and came out as 2.73µm for darkfield and 2.86 for brightfield and which when compared to the red channel modelled resolution for the M02/M01 lens pair of 2.94µm it proves that the modelling conducted was accurate as the small amount of is within margin of error.

Although *Diatoms* are of similar size to *C. elegans* and thus are a good representation, it was not part of the specification to image *Diatoms* thus shown in Figure 52 is the brightfield and darkfield images of *C. elegans* under the same test conditions as for the *Diatoms* with brightfield exposure time of 1/333 seconds and darkfield exposure time of 1/17 seconds. Although it can be seen from Figure 52 that this meets the specification of imaging *C. elegans* the level of detail is clearly inferior to the images of the *Diatoms* in both modalities, this is because the *C. elegans* have degraded over time. As these specimens were the same specimens in Figure 50 which was taken on 28/03/2024 and the images taken on my custom microscope setup was taken on the 07/05/2024, which is the reason for the inferior images compared to the *Diatoms.* For this reason, the above analysis for the rise coefficient was not done as would only show hindered results for no fault of the hardware or imaging system and is just because of degraded samples, but the *Diatoms* serve as a valid comparable alternative.

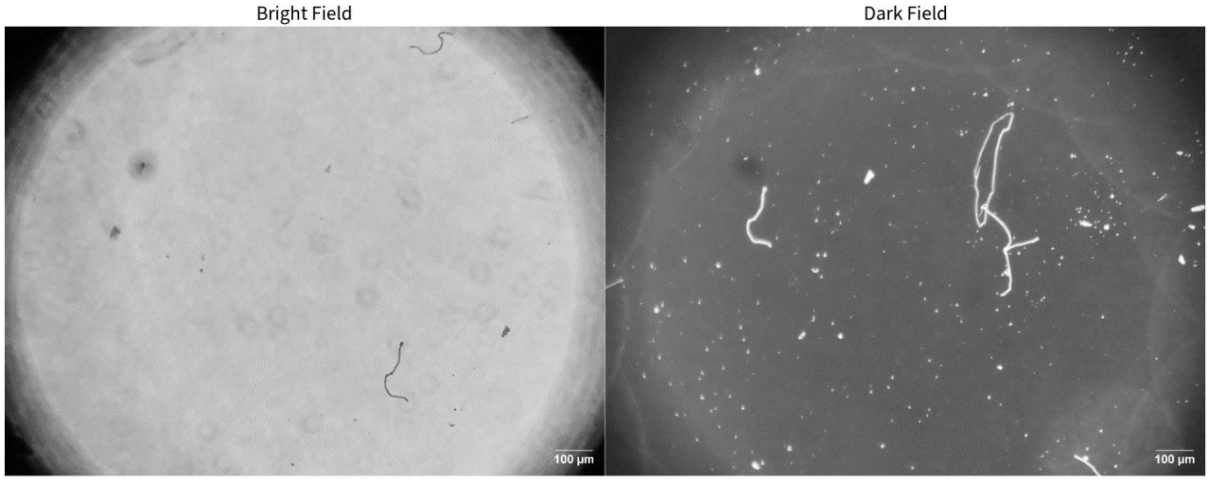

Figure 52: Brightfield and darkfield images of C. elegans taken on the custom microscope setup.

The final specimen imaged on the custom microscope setup was Buccal Epithelial cells shown in Figure 53, both modalities performed well however darkfield provides more detail of internal structures of the cell than the brightfield image produced, where the brightfield was took with an exposure time of 1/333 seconds and the darkfield exposure time was 1/17 seconds. The same analysis was performed for the Buccal Epithelial cells that was done for the *Diatoms* and for brightfield the rise coefficient was -9.85 and for darkfield the rise coefficient was 11.78. Again, further proving the validity of the images taken on the developed microscope across different specimen types.

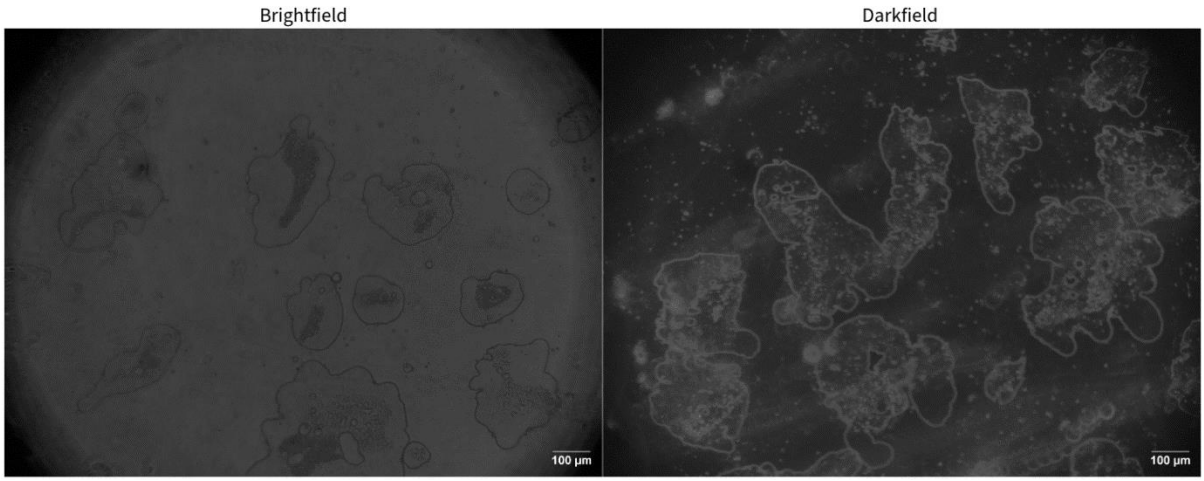

Figure 53: Two images of Buccal Epithelial cells for brightfield and darkfield modalities.

3.0 Conclusion

3.1 Consideration of System Within the Wider Context

3.1.1 Future PCB Design Improvements

Further development of the system would encompass multiple factors. Firstly, the illumination PCB developed for the project would need some small changes by adjusting the reverse polarity protection circuit, and the PCBs would be manufactured as populated due to the small nature of 0402 components used, which resulted in being extremely difficult to solder by hand and most likely contributed to the lack of functionality in the PCB caused by shorts underneath components caused by the use of solder paste.

In addition to making small changes and getting the boards professionally populated, a redesign of the PCB could be done to use larger addressable LEDs such as the WS2812C-2020 [22]. This would simplify the board design as addressable LEDs come with a built-in controller allowing for daisy chain connections and control via a one wire interface allowing for control of colour and brightness of each LED. As well as allowing for a simplified design it also adds functionality as the PCB that was developed for the project has control of three LED rings of set colour and only entire rings are brightness controllable. The only disadvantage of this method is the size of 0402 LEDs currently used are 1mm x 0.5mm and the 2020 package in the WS2812C-2020 are 2mm x 2mm meaning that less LEDs can be fit on the board, however as the control for the LEDs are done via a one wire communication not as many LED driving components are needed as only one driver would be needed to supply enough power rather than with current configuration which has one driver per ring to achieve brightness control.

The final work that would be needed to done is prepping the PCB for space applications, which mainly consists of via filling to prevent out gassing which can be done by the PCB manufacturer for an additional charge of $49.50 or £39.61 [23] and applying a conformal coating to prevent environmental damage to components on the PCB. Another consideration for PCBs for space applications is the phenomenon of Tin whiskers [24]. This is a process that single crystal filaments emerge from the solder under conditions such as temperature fluctuations and mechanical strain [24], where vacuum and thermal cycling accelerate whisker growth. To prevent / take steps to mitigate this involves using unconventional solder, such as leaded solder and by using conformal coating [24] such as Arathane 5750 by Silmid for £831.19 [25]. Overall, the future PCB design changes would not take significant time, even if a new design was done with the WS2812C-2020 LEDs, and the steps required for preparing the PCB for space application would not take long as the PCB manufacturer can perform the via filling and can request a leaded solder to help mitigate metal whisker formation. Finally applying a conformal coating would not take long and could be outsourced to a professional company such as SCH Technologies [26]. However, the time taken for this could take longer if the PCB were to meet NASA standards as it would have to meet the following standards [27]: MIL-STD-55110, MIL-PRF-31032, IPC A-600, IPC-6012, IPC-6013, IPC-6015, IPC-6018, IPC-6012DS and others.

3.1.2 Future Microscope System Development work

In terms of the future development for the microscope system, the main required work would be to shrink the design down to fit inside the CubeSat 10cm^3 limit. To do this the vertical height would have to be reduced, as because of the budget for the project non-ideal lenses had to be made to work, resulting in just the camera and lens arrangement taking up 80mm vertical height, which would leave little to no room for the illuminator PCB. The main way this would be overcome is by using a higher end professional lens with parameters as close as possible to the best lens selection based on the output from the merit function in section 2.2.5.6.

A lens that could be used in the future development work is a triplet lens from Thor Labs, specifically TRS064-010-A [28]. This would be an optimal choice for lens selection as a triplet lens comes with unity magnification, which based off the size of the *C. elegans* is enough to resolve sufficient detail. In addition, due to the construction of a triplet lens having 3 optical components, where the first is crown glass, then BK7 glass then crown glass again, this has a benefit of cancelling out the aberrations due to the two crown glasses being identical but mirrored over the horizontal axis, effectively eliminating the issues with chromatic aberration. The triplet lens also has the benefit of a 10mm focal length, thus an optimal solution would have the lens placed 10mm away from the camera imaging sensor and the specimen 10mm from the front of the lens, given that the thickness of the lens gives a total size of 24.7mm from the camera sensor to the specimen. This reduces the height of the lens configurations by almost a third. This would not take much time to implement as all that would be needed is a custom adapter for the lens to mount to the camera which could be 3D printed on a resin printer for extra accuracy.

In terms of additional improvements to the lens configuration, if using the best lens configuration from the merit function output, then this drawback of chromatic aberration can be used to provide imaging at multiple focal planes without the need for physically moving the lens configurations [29]. This works as chromatic aberration results in a certain colour being focused to a different plane than others and thus can use this chromatic aberration to image on multiple planes via different colour channels based off where the lens focal plane is for that colour channel. As this adds additional features and does not require expensive lenses to achieve, results in high applicability for use in education as the entire microscope system is low cost and easy to assemble and thus can be used to teach basic optical principles and give experience to younger people in constructing and setting up the microscope. In addition, as the data from this thesis will be open sourced provides additional benefit for an educational setting as there is no licence fees and none of the files or data will be deprecated, allowing for a stable teaching platform for a long period of time.

Future improvements also would include upgrading to stepper motors to adjust the height of the specimen or adjust the height of the camera to allow for finding the focus point much more easily than the manual method currently implemented. This would increase the use case for in education and make the microscope accessible to a more diverse range of students as will be easier to operate allowing students to focus on the concepts of microscopy rather than very accurately adjusting focus by hand. Expanding on having stepper motors, could allow for implementation of auto focus to abstract away the focusing from the user entirely. Multiple methods exist for this, the

simplest being contrast detection, which works by looking at the gradient of pixel pairs for the x and y direction, the height is then changed and ran again, and the value is checked, higher values correspond to higher focus. Thus, then an iterative approach is used and will converge on an optimal focus distance, the processing time for this can be substantial and thus other methods are more efficient such as Sobel operation [30]. Sobel operation works principally the same as contrast detection but is more efficient as it reduces the amount of data and will remove unneeded information without compromising on the structural properties of the image [31]. Software libraries already exist for implementing this such as MATLAB can be used to generate an independent C++ library [32] to perform the calculations. This expands on the previous point of expanding diversity and inclusion by making the system easier to use allowing for a wider range of abilities to participate in using the system, reinforced by the fact that all data for the design will be open source allowing for a stable teaching platform for a significant period of time as no licences are needed and nothing will get deprecated.

3.1.3 Current State of the Project

The current state of the project fully addresses the following specification points:

- Design of a custom PCB with 3 LED rings of red, green, and blue with central cyan LED.
- Components for the optical setup have been selected including the lenses and camera module.
- Design of microscope stand/ enclosure with manual height adjustment for the specimen positioning relative to the camera lens.
- All systems including the software, camera and optics setup, microscope stand, and PCB illuminator have all been integrated into one system.

- ➢ In terms of the stretch goals set out for the project a graphical user interface was successfully developed with MQTT integration for wireless control of the microscope system.
- ➢ The final stretch goal that was achieved was implementation of a communication system to send post processed and non-processed images taken from the microscope back to the Linux PC which is controlling the microscope.

The following specifications points are the ones that were fully met:

- Components for optical setup were selected included lenses and camera.
- Design of microscope stand/ enclosure with manual height adjustment to position specimen relative to the camera lens.
- Stretch goal of developing high level GUI with MQTT to wirelessly control the microscope.
- Stretch goal to have images that were taken using the GUI to send back processed and non-processed images back to the Linux PC running the GUI.

The following specification points were the only one which was partially met:

- Although the illumination PCB was designed there was functionality issues such that only one of the LED rings works at a time and only operates at full

brightness. The reason for the lack of functionality is most likely caused by the soldering of the components to the board, as the size of most of the components are 0402, and as solder paste was used without a stencil its likely there are shorts underneath components are not visible. This is reinforced as during population of components each LED ring was soldered then test before the next ring is done, while doing this the first ring and central cyan LED were working fully and when populating the rest is when functionality loss occurred, even though the 3 rings and driving components are practically identical to each other.

- Integrating all the sub-system was not fully done due to the hardware functionality limitations of the PCB as everything worked individually such as the code to control the PCB but as the PCB had hardware issues the software wasn't able to fully integrate due to this lack of functionality.

Finally, the following specifications points are the ones that were not met:

- A force analysis of the microscope stand/ enclosure with the PCB design was not conducted. This was because of the unexpected amount of time that had to be spent categorising the lenses used because of how inaccurate the datasheet was that came with it.
- The final specification point that was not met was the development of a code to run on the Raspberry Pi 4B, to automatically take images, perform post processing and store the output to the SD card. The reason why this specification point was not met was because significant time was spent developing the code for the Raspberry Pi to facilitate the GUI. However, most of the code developed for the Raspberry Pi to facilitate the GUI is applicable and reusable for a future code that would be developed to achieve this, as most of the time consuming challenges such as Python integration to C++ programs and development of the rawpy Python code have already been figured out, thus wouldn't not take significant time to implement into a system to run autonomously.

3.1.4 Reflection on Management

Overall, my self-management of the project was good but with room for improvement. Where the improvement mainly lies in thinking more critically about how the components of the project interact in a more detailed way. One example of this is in the progress pro-forma from 20/11/2023 and the initial Gantt chart from the proposal, the CAD design for the microscope was intended to be finished in week 7 with an error bar for week 8. The issue with this is that the first PCB design was set to finish at the same time with the 1-week error bar, this meant it would be obvious that this can't be achieved as the PCB design is required to do the CAD work as the final PCB dimensions need to be known. Thus, my project management skills have been developed over the course of the project to think more critically about each component of a design, how they relate, and prioritisation of tasks such that future tasks are not waiting on previous tasks to be completed.

In addition to prioritisation, my project management could have benefited from a more in-depth analysis of the needed components for a successful project. Mainly demonstrated by the lack of the GUI in the original project specification, as if a more in-depth analysis was conducted at the start of the project about how the user interacts with the system, it would have been evident that a GUI was required. As shown in the revised Gantt chart in the appendix this was not a small task and ultimately resulted in other tasks being delayed, which could have been prevented.

Most of the times the project fell behind according to the Gantt chart and required actions to set the project back on track was because of delays in one task causing delays in subsequent tasks. For example, in progress pro-forma from 20/11/2023, as the PCB design went through multiple iterations due to slight errors in the design, meant that other sections such as microscope CAD development or testing of microscope system could not be done, and as each PCB iteration meant waiting for the boards to arrive. It meant the action required to get the project back on track was to prioritise the design change for the PCB so that other tasks can be completed while waiting for the boards to arrive, and once arrived and verified correct then the other tasks that depended on the PCB design can resume such as the CAD design for the microscope and microscope testing.

The risk mitigation strategy was only somewhat affective as it primarily focused on risks that were outside my control such as component shipment delays, 3D printing failures, all of which are valid but don't address some of the obvious risks. As an example, the PCB design took five different iterations for a correct design with multiple orders for PCBs, due to errors in the design, this resulted in not being able to test the PCBs until week 24-26 instead of the intended weeks 9-10. Thus, what was learnt from this was that the risk management section should have included risk mitigations for factors such as errors in designs, accidents in testing such as short circuiting a PCB on accident and other factors that I was in control of.

4.0 References

[1] “Raspberry Pi Documentation - Camera.” Accessed: Mar. 10, 2024. [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html

[2] A. E. Ronca, K. A. Souza, and R. C. Mains, “TRANSLATIONAL CELL and ANIMAL RESEARCH in SPACE 1965-2011,” 2015.

[3] C. H. Lai, C. Y. Chou, L. Y. Ch’ang, C. S. Liu, and W. C. Lin, “Identification of Novel Human Genes Evolutionarily Conserved in Caenorhabditis elegans by Comparative Proteomics,” *Genome Res*, vol. 10, no. 5, pp. 703–713, May 2000, doi: 10.1101/GR.10.5.703.

[4] T. C. elegans S. Consortium*, “Genome sequence of the nematode C. elegans: A platform for investigating biology,” *Science (1979)*, vol. 282, no. 5396, pp. 2012–2018, Dec. 1998, doi: 10.1126/SCIENCE.282.5396.2012/SUPPL_FILE/C-ELEGANS.XHTML.

[5] “Condenser-free contrast methods for transmitted-light microscopy,” *J Microsc*, vol. 257, pp. 8–22, 2015, doi: 10.1111/jmi.12181.

[6] W. Keil, L. M. Kutscher, S. Shaham, and E. D. Siggia, “Long-Term High-Resolution Imaging of Developing C. elegans Larvae with Microfluidics In Brief,” *Dev Cell*, vol. 40, pp. 202–214, 2017, doi: 10.1016/j.devcel.2016.11.022.

[7] W. Lan, “CubeSat Design Specification Rev. 13 The CubeSat Program, Cal Poly SLO CubeSat Design Specification (CDS) REV 13 Document Classification X Public Domain ITAR Controlled Internal Only”.

[8] “ArduCam CS Lens Kit for HQ Camera 14-65deg (5pcs) - RobotShop.” Accessed: Mar. 17, 2024. [Online]. Available: https://uk.robotshop.com/products/arducam-cs-lens-kit-hq-camera-14-65deg-5pcs

[9] “Understanding Microscopes and Objectives | Edmund Optics.” Accessed: Apr. 29, 2024. [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-notes/microscopy/understanding-microscopes-and-objectives/

[10] “What is an Infinity-corrected Optical System? | Learn about Microscope | Olympus.” Accessed: Mar. 17, 2024. [Online]. Available: https://www.olympus-ims.com/en/microscope/terms/feature15/

[11] D. G. H. Andrews, “A new method for measuring the size of nematodes using image processing,” *Biol Methods Protoc*, vol. 4, no. 1, Jan. 2019, doi: 10.1093/BIOMETHODS/BPZ020.

[12] “Galaxy Tab S6 Lite (SM-P610) Gray 64 GB | Samsung Jordan.” Accessed: Apr. 20, 2024. [Online]. Available: https://www.samsung.com/levant/tablets/galaxy-tab-s/galaxy-tab-s6-lite-10-4-inch-gray-64gb-wi-fi-sm-p610nzaamid/#specs

[13] “TLC5947 24-Channel, 12-Bit PWM LED Driver With Internal Oscillator datasheet (Rev. B) | Enhanced Reader.”

[14] D. Incorporated, “1A STEP-DOWN CONSTANT CURRENT, HIGH EFFICIENCY LED DRIVER,” 2015, Accessed: Apr. 29, 2024. [Online]. Available: www.diodes.com

[15] “Standard 30 mm Cage Plates.” Accessed: Mar. 13, 2024. [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=2273

[16] “Thorlabs - LCP34T/M 60 mm Cage Plate, SM2 Threads, 0.9" Thick, M4 Tap (Two SM2RR Retaining Rings Included).” Accessed: Apr. 26, 2024. [Online]. Available: https://www.thorlabs.com/thorproduct.cfm?partnumber=LCP34T/M

[17] “GitHub - Llamero/30mm_cage_plate_LED_MCPCB: LED mount that is compatible with the Thorlabs 30 mm cage plate.” Accessed: Apr. 26, 2024. [Online]. Available: https://github.com/Llamero/30mm_cage_plate_LED_MCPCB

[18] “Eclipse Mosquitto.” Accessed: Apr. 26, 2024. [Online]. Available: https://mosquitto.org/

[19] “GitHub - KostiantynBushko/QMosqMqttClient: Qt wrapper for mosquitto mqtt client.” Accessed: Mar. 19, 2024. [Online]. Available: https://github.com/KostiantynBushko/QMosqMqttClient

[20] “open-source modular microscope platform.” Accessed: Apr. 26, 2024. [Online]. Available: https://www.cairn-research.co.uk/product/openframe-microscope/

[21] P. H. Lambrev, P. Akhtar, and H. S. Tan, “Insights into the mechanisms and dynamics of energy transfer in plant light-harvesting complexes from two-dimensional electronic spectroscopy,” *Biochim Biophys Acta Bioenerg*, vol. 1861, no. 4, Apr. 2020, doi: 10.1016/j.bbabio.2019.07.005.

[22] “WS2812C-2020 Intelligent control LED integrated light source Mechanical Dimensions”, Accessed: Apr. 30, 2024. [Online]. Available: http://www.world-semi.com

[23] “PCB Prototype & PCB Fabrication Manufacturer - JLCPCB.” Accessed: Apr. 30, 2024. [Online]. Available: https://cart.jlcpcb.com/quote?orderType=1&stencilLayer=2&stencilWidth=100&stencilLength=100&stencilCounts=5

[24] Z. WANG and S. WANG, “Formation and evolution mechanism of metal whiskers in extreme aerospace environments: A review,” *Chinese Journal of Aeronautics*, vol. 36, no. 9, pp. 1–13, Sep. 2023, doi: 10.1016/J.CJA.2023.07.007.

[25] “Huntsman Arathane 5750 A/B (LV) Urethane Conformal Coating 1USQ Kit | Silmid.” Accessed: Apr. 30, 2024. [Online]. Available: https://www.silmid.com/adhesives/polyurethane-adhesives/araldite-arathane-5750a-b-1usq-kit/

[26] “Subcontract Coating Services: Conformal Coatings, Nano-Coating, Parylene.” Accessed: Apr. 30, 2024. [Online]. Available: https://www.conformalcoating.co.uk/coating-services/

[27] “Printed Circuit Boards.” Accessed: Apr. 30, 2024. [Online]. Available: https://nepp.nasa.gov/index.cfm/27505

[28] “Thorlabs - TRS064-010-A Unmounted Ø1/4" Steinheil Achromatic Triplet, f = 10 mm, ARC: 400 - 700 nm.” Accessed: May 08, 2024. [Online]. Available: https://www.thorlabs.com/thorproduct.cfm?partnumber=TRS064-010-A&utm_source=meetoptics&utm_medium=referral&utm_campaign=product-page&utm_content=TRS064-010-A

[29] M. Weinigel, A. L. Kellner, and J. H. Price, “Exploration of Chromatic Aberration for Multiplanar Imaging: Proof of Concept with Implications for Fast, Efficient Autofocus,” 2009, doi: 10.1002/cyto.a.20788.

[30] “Sobel Edge Detection - an overview | ScienceDirect Topics.” Accessed: May 01, 2024. [Online]. Available: https://www.sciencedirect.com/topics/engineering/sobel-edge-detection

[31] “Edge Detection.” Accessed May 04 2024 {online} Available: https://www.cs.auckland.ac.nz/compsci373s1c/PatricesLectures/Edge%20detection-Sobel_2up.pdf

[32] “Generate Code for Sobel Edge Detection That Uses Half-Precision Data Type - MATLAB & Simulink - MathWorks United Kingdom.” Accessed: May 01, 2024. [Online]. Available: https://uk.mathworks.com/help/coder/ug/edge-detection-with-sobel-method-in-half-precision.html

5.0 Appendix - Project Review Pro-forma

<table>
<tr><th colspan="2">Review Pro-forma – 23/10/2023</th></tr>
<tr><td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Start research into phase contrast microscopy.
Start research into CubeSat specification.
Start the design for microscope parts.
Order Raspberry Pi 4B and Camera.</td><td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
Started research into phase contrast microscopy.
Started research into CubeSat specification.
Researched into potential component options for the PCB illuminator.</td></tr>
<tr><td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]
The research sections are on track and am in the middle of developing an understanding for both the microscopy and what is required for the CubeSat specification. What isnt to plan so far is the raspberry Pi and camera have not been ordered yet as time has been spent on researching for PCB components and still need to research further into what camera is going to be most suitable.
To get back on track the Raspberry Pi 4B needs to be ordered and a comparison of available cameras compatible with the Raspberry Pi need to be done to choose the optimal camera module.
No additional resources or support is required.</td></tr>
</table>

<table>
<tr><th colspan="2">Review Pro-forma – 20/11/2023</th></tr>
<tr><td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Researched CubeSat Specifications and Microscopy research.
CAD Design for microscope prototype.
Design or source microscope parts.
1st PCB schematic and PCB Design.
Ordered Raspberry Pi and Components.
Ordered Raspberry Pi 4B and camera.</td><td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
Finished research for CubeSat.
Still researching microscopy as such a large topic.
Ordered Raspberry Pi and Components
Designed schematic and PCB.
Some software designed to control PCB (Custom PWM Library).
Software developed to control camera.
Software developed to take raw images from camera and reconstruct for full 12bit resolution of camera. (rawpy library used instead of OpenCV).
Progress made in developing GUI for ease of testing purpose, uses MQTT internet messaging protocol so can wirelessly control camera
PCBs ordered.</td></tr>
<tr><td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]

I am ahead in some areas such as, software to control PCB, software for camera control, software to develop raw images into .TIFF and made good progress for a stretch goal of a GUI to control the microscope for testing purposes, and PCBs have been ordered. However, I am off track in some areas such as my time plan set out to do the CAD design for the microscope prototype, which has not been done, as after starting the project I realised I need to have completed other tasks before this is possible such as the PCB design. And I originally made a mistake with the PCB and had to do it again so waiting until this was finalised seemed logical. To get back on track for this once I verify PCB design is correct and works and dimensions don’t need to change then I can start the CAD design. Not finishing the CAD design does affect other tasks such as testing the microscope system, but luckily CAD design doesn’t take that long to design or print. Finally, the camera has not been ordered as the comparison of available Raspberry Pi compatible camera modules has not been conducted yet. Thus, to be back on track the camera comparison analysis needs to be completed and finalise the PCB design so that the microscope CAD work can be done.
No additional support or resources are required.</td></tr>
</table>

<table>
<tr><th colspan="2">Review Pro-forma – 29/01/2024</th></tr>
<tr><td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Software development for Raspberry Pi 4B.
CAD design for the Raspberry Pi HQ camera holder.
Started development of final PCB Design.</td><td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
Software development for Raspberry Pi started before the last review meeting as required for testing GUI, and has been developed further, (expanded to allow for testing of GUI).
The CAD design for the Raspberry Pi HQ Camera is complete.
Progress is on track according to Gantt chart for development of final PCB.
Not in the Gantt chart as for GUI stretch goal. Managed to compile the Python code for rawpy into C++ using Cython to allow execution of Python function from C++ code.
Ordered lens kit.</td></tr>
<tr><td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]
From the last proforma review to now everything is on track according to the Gantt chart.
From the last proforma the 1st PCB design test was suppose to have been done, it still hasn't been done as after the arrival of the boards errors were noticed with the design that meant they had to be slightly redesigned and reordered.
Finally, from the previous proforma the microscope CAD design still hasn't been done as the final decision on lenses hasn't been made, however a step towards the lens selection has been done as a lens selection kit has been ordered to try a selection for the optimal lens.
To get back on track a lens needs to be picked for the camera and then the CAD design can be done for the microscope. In addition, once PCBs arrive check over them and test them, once verified it works and the dimensions are correct the CAD for the microscope holder can be done.
No additional resources or support is required.</td></tr>
</table>

<table>
<tr><th colspan="2">Review Pro-forma – 26/02/2024</th></tr>
<tr><td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Finished design for final PCB.
Software to control custom PCB.
Test microscope with Raspberry Pi camera.
Final CAD design for the microscope.</td><td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
Finished all software development for the GUI apart from some of the controls for the PCB as don’t have the PCB finalised.
Also finished the corresponding Raspberry Pi code apart from the controls for the PCB.
Started performing lens characterisation tests due to inaccurate data sheet specifications, such as magnification, minimum object distance and FOV.
Some CAD design started for the microscope, designed with adapter to be multi-functional and is being used to hold camera facing downwards to take images for magnification tests.
Finished PCB that wasn’t finished from last proforma and ordered them.</td></tr>
<tr><td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]
Even though the design was finished for what was thought would be the final PCB, due to an error I thought that the forward voltage of blue LEDs were comparable to red and green, and thus were wired in a series parallel arrangement, however as blue LEDs have forward voltage above 2.5V meant I did not have the voltage from a 5V supply to run them properly. Thus, this will have to be fixed and PCBs ordered again.
It was not possible to test the microscope with the Raspberry Pi HQ camera properly as the stand has not been fully designed and the illumination PCB had to be slightly redesigned. Even though not finished the CAD design for the microscope has been started and when finished will just need 3D printing.
To get back on track the changes to the PCBs need to be made as soon as possible so they can be reordered, this had the consequence of not being able to test the microscope as the CAD can’t be done to hold the PCB and the illuminator PCB is needed for testing.
No additional support or resources are required.</td></tr>
</table>

<table>
<tr><th colspan="2">Review Pro-forma – 25/03/2024</th></tr>
<tr><td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Final CAD work for microscope.
Stress analysis of PCBs.
Software to control custom PCB.
Combine OpenCV and software for PCB.</td><td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
The final CAD work for the microscope is finished, it was completely redesigned from the first iteration and was resin printed for higher accuracy for the gears.
Finished the software to control the custom PCB by adding the needed components to the GUI and setting up new MQTT topics for them.
Software all combined together so the GUI can control user post processing options and can control the software for the PCB.
Performed many more lens characterisation tests and have finished all of them including tests for infinite conjugate imaging such as the same tests for finite and vignetting, light through put chromatic aberration tests and resolving power tests.
The new PCBs came through and have all been soldered up.</td></tr>
<tr><td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]
Overall on track according to the Gantt chart for all but the stress analysis as it has not been conducted for the PCBs, this isnt because of any technical difficult but because the lens characterisation took a long time to perform and were not included in the original time plan as it was not foreseen that the lens parameters from the manufacturer would be inaccurate and thus the time to perform all these tests were not accounted for in the original Gantt chart. No additional support or resources are required.
Even though I am off track on the force analysis of the PCB this is a stretch goal of mine, and given the available time left due to parts taking a long time that were unforeseen such as all of the lens characterisation, means that the force analysis wont get completed on time.</td></tr>
</table>

5.1 Reflective Gantt Chart

Key: Time Took | = Milestone | Holidays | Exam Period | Supervisor Meeting

Project Overview Points	Index	Dependencies	Hours Spent
Design and Research Phase			
Phase Contrast Microsopy Research	1	N/A	18
CubeSat Specification Research	2	N/A	10
CAD Design for Microscope Prototype	3	N/A	7
Research Component Options for PCB	4	N/A	4
1st schematic/PCB design	5	N/A	8
Christmas Holidays	6	N/A	N/A
Exam Period	7	N/A	N/A
Easter Holidays	8	N/A	N/A
Exam Period	9	N/A	N/A
Software to Control Pi4B Camera	10	N/A	10
GUI Development	11	N/A	50
Corresponding Raspberry Pi Code for GUI	12	11	30
Raw Image Reconstruction with Rawpy	13	N/A	10
Custom PWM Library Wrote	14	N/A	7.5
CAD for Raspberry Pi Camera Holder	15	N/A	2
Schematic Design for Final PCB	16	5	2
PCB Layout Design for Final PCB	17	17	14
Project Defence	18	1,2	1
CAD Design for Microscope Hardware	19	15	10
Software to Control Custom PCB	20	16	15
Combine all Software	21	10,11,12,13,14,20	30
Order Raspberry Pi 4B/ and Camera	22	N/A	0.5
Order Components	23	N/A	0.5
Lens Tests, Mgnification, FOV, Obj Distance	24	N/A	15
Infintie Conjugate Tests, Vignetting, Chromatic Aberration, Resolving Power	25	N/A	30
Testing Phase			
Test Final PCB design	26	16,17	15
Test microscope with Raspberry Pi Camera	27	10,19	6
Test Final microscope Version	28	19,21, 24,25	15
Document Phase			
Project Overview	29	N/A	20
Adding Content to Thesis Report	30	26,27,28	55
Thesis 1st Draft	31	30	N/A
Thesis Report	32	30,31	N/A
Thesis Defence Preperation	33	32	10
Thesis Defence	34	33	1
Project Presentation Preperation	35	32	8
Project Presentation	36	35	1
Supervisor Progress Meetings			
Meeting 1	37	N/A	0.5
Meeting 2	38	N/A	0.5
Meeting 3	39	N/A	0.5
Meeting 4	40	N/A	0.5
Meeting 5	41	N/A	0.5