Project ID: DRA 2001

Internet-Connected Biomonitoring Device

A Major Qualifying Project Report submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the degree of Bachelor of Science

Submitted by:

Shruti Bhatia, Biomedical Engineering/Electrical and Computer Engineering

Diana Galvez, Biomedical Engineering

Kathleen Peter, Biomedical Engineering

May 18th, 2020

Submitted to:

Professor Dirk Albrecht, Ph.D., Advisor
Department of Biomedical Engineering

Professor Stephen Bitar, Ph.D., Advisor
Department of Electrical and Computer Engineering

Abstract

The goal of this project was to design a low cost, easy to use internet-connected biomonitoring system which fits in existing incubators and is capable of remote culture monitoring with video and data logged automatically. The final design used a Raspberry Pi microcontroller, USB camera, LED array, custom-created plastic housing, and temperature/humidity sensor. In addition to live video streaming, the device captures still images to create time-lapse videos. The device costs $200 to produce and was made customizable and easily reproducible through the creation of a step-by-step building guide which includes alternative design component suggestions. The reproducibility of the device was tested by creating multiple systems from the step-by-step guide. These systems were used to capture time-lapse videos with durations ranging from 1.5 hours to 5+ days and magnifications from 10x to 250x. The temperature/humidity sensor and pH sensing capabilities were verified against gold standard technologies. The sensor data is logged automatically to a cloud server and overlaid on the images using open source python code.

Acknowledgements

We wish to acknowledge the assistance given to us by our main advisor Professor Dirk Albrecht as well as our secondary advisor Professor Stephen Bitar. Additionally, we would like to thank Professor Raymond Page for his assistance with our project design, and all those who helped with our planning, testing, and validation of our design including Eric Larsen, Rob Kirch, Professor Vidali, Professor Glenn Gaudette, and Professor Ambady.

Table of Contents

Table of Figures

Table of Tables

Authorship Table

Section	Author(s)
Abstract	Kathleen Peter
Introduction	Kathleen Peter
Literature Review	
Cell Culture	Diana Galvez
C. elegans Culture	Diana Galvez
Principles of Image Gathering	Shruti Bhatia
Microscopy	Shruti Bhatia & Kathleen Peter
Existing Device Research	Kathleen Peter
Internet Connectivity and IoT Devices	Kathleen Peter
Project Strategy	
Initial Client Statement	Kathleen Peter
Technical Design Requirements	Shruti Bhatia
Design Requirements (Standards and Regulations)	Diana Galvez
Revised Client Statement	Kathleen Peter
Management Approach	Kathleen Peter
Design Process	
Needs Analysis	Shruti Bhatia
Concept Maps	Shruti Bhatia
Alternative Designs	
Raspberry Pi Model	Shruti Bhatia
Remote Monitoring	Diana Galvez
Image Gathering Source	Kathleen Peter
Light Source	Shruti Bhatia

Housings	Kathleen Peter
Temperature and Humidity Sensor	Shruti Bhatia
pH Sensing	Shruti Bhatia
Data Logging	Shruti Bhatia
Final Design Selection	
Raspberry Pi Model	Shruti Bhatia
Remote Monitoring	Kathleen Peter & Diana Galvez
Image Gathering Source	Kathleen Peter
Light Source	Shruti Bhatia
Housings	Kathleen Peter
Temperature and Humidity Sensor	Shruti Bhatia
pH Sensing	Shruti Bhatia
Data Logging	Kathleen Peter
Design Verification	
Preliminary Testing	Kathleen Peter
Verification	
Experimental Results	All
Sensor Results	Shruti Bhatia
Data Logging Results	Kathleen Peter
Live Sample Tracking	Diana Galvez
Final Design and Validation	
Meeting Objectives	Diana Galvez
Reproducibility	Kathleen Peter
Experimental Methods	Shruti Bhatia
Standards	Diana Galvez
Global Impact	Kathleen Peter

Discussion	All
Conclusion and Recommendations	All

Chapter 1 - Introduction

Oftentimes in scientific research it is pertinent to use cell or organism cultures as model systems. In order to monitor dynamic biological processes, time-lapse microscopy is often employed. Time-lapse microscopy (TLM) is a technique in which a sequence of microscopic images are captured at regular time intervals [1]. These images are often stitched together to form time-lapse videos which show the dynamics of the culture over a set period of time. TLM allows for the long-term observation of changes in the behavior of cells and organisms in culture [1]. In contrast to endpoint assays where the most common method of capturing images is to take the culture plate out of the incubator at several points during the culturing process and observe and capture images using a microscope and camera, TLM generally uses continuous monitoring [2].

Ideally, cultures should be monitored remotely and continuously while remaining in the incubator. Current endpoint assay methods have a strong potential to interfere with the culture being analyzed, as the plates are being removed from the desired environment, potentially altering the results of a study. Continuous monitoring allows for more consistent monitoring and decreases the potential for interference with test conditions and supplies more effective information. More importantly, continuous monitoring reports how the biologic process occurs, not just the end result like endpoint assays. The importance of this can clearly be seen by looking at two cultures commonly used as model systems.

Cell culture has been used for model systems, toxicity testing, cancer research, virology, cell-based manufacturing, genetic engineering and therapy, and drug screening and development since its advent in the early 1960s [3]. As part of acting as a model system, the basic biology, biochemistry, and interactions of cells within culture are studied. In order to fully model the behavior of cells in culture the cells cannot be fixed in place, and therefore grow and migrate while interacting with other cells and biological agents [3]. Additionally, the successful culturing of cells in a laboratory requires very specific conditions. In order to maintain cell health, cells need specific CO_2, humidity, O_2, and pH levels [4]. These conditions, the overall state of the cells, and the behaviors of the cells can be monitored via imaging.

Another organism commonly used for model systems is *Caenorhabditis elegans* (*C. elegans*). *C. elegans* is a soil nematode which was first introduced as a model organism for development and neurobiology in 1974 [5]. In recent decades *C. elegans* has been shown to be a valuable model for a wide variety of biological processes and underlying mechanisms of human disease [5]. Examples of discoveries made using *C. elegans* include seminal discoveries about programmed cell death, and RNA interference, which is gene silencing by double-stranded RNA [6]. Due to the fact that *C.elegans* are whole living organisms, their behavior in culture is even more dynamic than the behavior of cells. *C. elegans* culturing requires very tight monitoring of O_2 levels, temperature, and humidity among others [7]. Imaging of these plates is required to be able to analyze the behavior of the nematodes [7].

There are several devices currently on the market that monitor cell health, function, and growth via video recording or TLM while the cells stay in the incubator, allowing for minimal interference with the culture [8]. However, these devices are insufficient as they are large, expensive, require wired connections, or aren't customizable to allow for various experiments. The large size and high cost of these devices makes them prohibitive for use in most labs. The goal of this project was to evaluate the current solutions available for continuous culture monitoring and prototype a new device capable of remotely and inexpensively monitoring cell or organism cultures. In order to create project objectives, we analyzed the current on-market gold standard devices as well as the devices created by other research labs. From our analysis of the strengths and weaknesses of each existing device we determined the basic needs of the system, and ways in which to improve upon the market standard. These objectives consisted of; capability of sending photos and video through the internet to the user along with data retrieved from optional sensors, low-cost, easily reproducible through use of a step-by-step instruction guide, small size, and versatility so that it may be applicable for use in various settings.

Once our objectives were determined, we researched the potential components that would be necessary to create a device capable of achieving these objectives. The main components we deemed necessary were a microcontroller, light source, image gathering source, remote monitoring software, and a housing. We also identified sensors for pH, humidity, and temperature as optional components. For each of these components we came up with alternative design options which were tested and compared to determine the final design. Since one of the objectives of this project was versatility, many of the components had multiple alternative design options which could function well as part of the final device and our selections were made based on our specific applications. Once a final design was selected, we completed validation testing to ensure that the device could function for our desired applications.

In chapter 2 we will present our background research in the form of a literature review. After that, our next chapters will go further in depth into the specifics of our project approach as outlined above. After our literature review, we present the project strategy which details our client statement, design requirements, and management approach. We then move into our design process focusing on our needs analysis, conceptual designs, and a detailed analysis of our alternative designs and final design selection. The fifth chapter goes into the specifics of our design testing and verification including protocols to test the optional sensors. We further move into the effects of our device on the world and the meaning of our results, finally closing with our concluding statements and recommendations of future work.

Chapter 2 - Literature Review

2.1 Cell Culture

Cell cultures have been a crucial primary source for investigations and studies in the medical and biomedical field. Cultures performed in a controlled lab environment serve as a model system to conduct studies outside of a living organism for a wide variety of possible goals and applications. For example, cell-lines have been used to test different types of drug delivery mechanisms, such as nanoparticle-based drug delivery systems. This *in vitro* alternative solves the ethical issue of performing tests in *in vivo* models while still maintaining the ability to assess the system's functionality, efficacy, and biological responses [9]. Furthermore, more advanced 3D cell culture models can be used to study different types of cancers as well as drug development [10]. There are two types of cell cultures that can be used: (1) primary cells and (2) cell lines [11]. Primary Cells come directly from the patient or living organisms and have a limited time span of about 40 to 60 cell divisions [11]. On the other hand, cell lines are engineered or mutated so that they go beyond the 40 to 60 division mark and continue to proliferate indefinitely.

By controlling and optimizing the culture environment, we can maintain a healthy and steady cell culture. Generally, the goal is to produce an environment that mimics the corresponding *in vivo* conditions based on the study being conducted. This is important because the cell culture will be able to yield results and measurements that accurately represent the *in vivo* behavior of the system in question. Some of the common controlled factors used in cell culture include pH, CO_2, and O_2 levels as well as temperature of incubation and percent humidity. The optimal pH value for cells is between 7.2 and 7.4 [11]. Phenol red serves as a common pH indicator within the cell culture media, however, there are other alternatives to pH regulators and indicators as well. Based on phenol red as the indicator, a yellow colored media indicates a lower pH than the optimal values and a bright pink media color indicates a higher pH than the optimal values. For cell survival, a 5% margin of CO_2 is used to stabilize the pH levels [11]. Generally, cell cultures are prepared and held under atmospheric conditions (about 20% O_2) [11]. However, that does not accurately represent physiological levels of O_2 (about 5%). This is due to the fact that the natural level of O_2 in the atmosphere is about 20% and it is hard to reduce and control low levels of oxygen under normal conditions. The optimal temperature for a cell culture depends on the types of cell origin. For mammalian cells, the temperature for optimal growth and proliferation rate is about the same as body temperature, which is 37 degrees Celsius [11]. Moreover, humidity for all cell cultures should be maintained between 95 to 100% to prevent the cell culture media from evaporating [11].

Some of the current techniques for monitoring cell culture involve plating cells in a Petri dish or plate, adding media and keeping track of the criteria for a healthy culture environment within an enclosed and sterile incubator. To monitor the culture's activity, the plate has to be taken out of the incubator and imaged under a microscope. This may result in loss of some of those optimal conditions we discussed above.

2.2 *C. elegans* Culture

Another widely popular model organism for studies in the biological and medical field is the nematode *Caenorhabditis elegans*. Because of their free-living conditions as well as their rapid growth and reproduction, *C. elegans* are often chosen to conduct many studies [7]. *C. elegans* can be obtained from the Caenorhabditis Genetics Center, or CGC. They are fed by an *E. coli* strain and Nematode Growth Medium (NGM) [7]. One of their most attractive and useful features are the fact that they mature and reach adulthood in about three days, are able to reproduce at an accelerated rate, and are transparent and easy to look at under a microscope. These characteristics are ideal to study their development, reproduction, and aging patterns.

C. elegans are periodically transferred to new dishes for feeding and standard development cycle purposes. Transfer frequency depends on the type of *C. elegans*, the temperature in which it is incubated, and the goal of the study being conducted. They can be transferred to a new plate by simple chunking or moving one by one from plate to plate with a sterilized scalpel or "worm picker" respectively. In order to maintain a healthy and optimized nematode culture, some conditions must be controlled, such as temperature, frequency of subcultures, and oxygen levels [7]. These parameters will vary depending on the application, genotype, and environment in which the plates are stocked and cultured in, meaning *C. elegans* tolerate a wider and not so strict array of parameters as compared to cell cultures. For example, cultures maintained at around 25 degrees Celsius will reach starvation sooner than an identical culture maintained at around 16 degrees Celsius [7]. Additionally, plates kept at around 11 degrees Celsius can survive several months without plate transfers [7]. In general, *C. elegans* can survive without being fed for several months.

However, it is most ideal to conduct studies using well-fed worms because the lack of food as well as the temperature and the frequency of passaging all change their behavior [12].

2.3 Principles of Image Gathering

Light is defined as an electromagnetic wave composed of photons. Images are created via light waves refracting through a lens [13]. Refraction is the deflection of light rays due to them passing from one medium to another. Because different mediums have different material properties, this changes their velocity, and therefore the direction of light rays. Determining how these rays are diffracted can be determined by Snell's Law [14]. Snell's Law is based on Fermat's principle, which states that light travels in the shortest path that takes the least time and can be seen below in **Figure 1**.

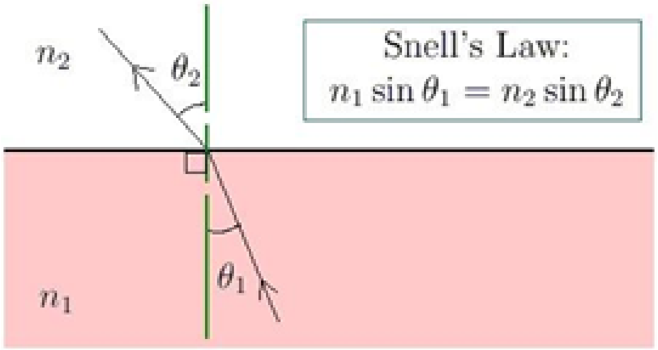


Figure 1: Snell's Law [15]

Figure 1 above shows a light ray passing through mediums n_1 and n_2. The diffraction angle, **θ**, depends on the ratio between refractive indices n_1 and n_2 of the two materials [15]. Snell's Law is applicable to image capture since light passes from the air to the lens, and back to the air. By understanding Snell's Law, we are able to predict where the image will be produced relative to the lens. Below in **Figure 2** is a diagram depicting how Snell's Law works in lenses to capture an image.

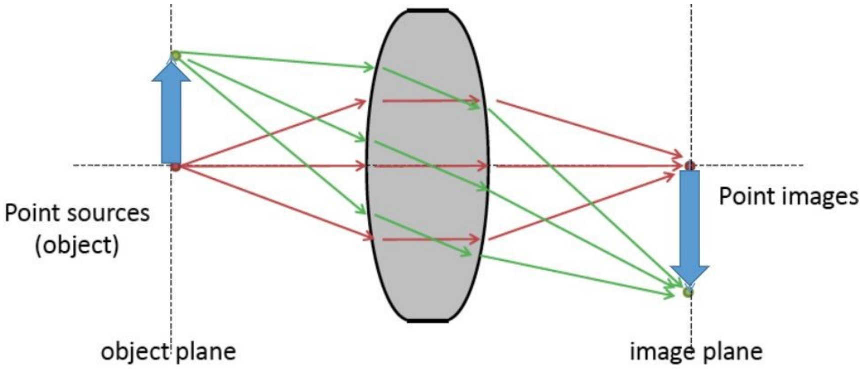


Figure 2: Image Created due to Refraction of Light Through a Lens [14]

As shown by **Figure 2** above, light rays are reflected from the object of the image, causing them to bounce back into the lens. Then, they are refracted twice due to the change in medium into the lens and then into the air. The point at which these rays meet outside the lens is where an image is created. However, the image is created upside down since the distance between the optical axis and the point at which the rays focus is in the downward direction. In order to see the image created, an opaque image plane must be placed at the point where the rays converge so the light rays can be absorbed. The point at which the optical axis originates from is determined by the center of the lens [14].

Another important parameter of a lens to consider when creating an imaging system is the focal length of the lens. Focal length is the distance from the center of the lens at which parallel rays are brought to a focus, assuming the object of the image is infinitely far away. If the image plane is placed slightly within the focal length of the lens, the image will be blurry since the light rays are not focused yet. Focal length of a lens can be determined using the following equation, known as the Lens Maker's Equation, where f is the focal length, n is the refractive index of the medium, and R_1 and R_2 are lens thickness of each side of the lens with respect to the center of the lens [14]:

$$(1)\ 1/f\ =\ (n-1)(\frac{1}{R1}-\frac{1}{R2})$$

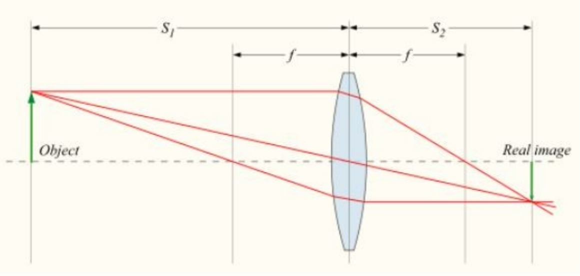
As we will see in the coming section, focal length helps the user determine the magnification of the image. Therefore, lens makers will often manipulate R_1 and R_2 to meet the client's specifications for magnification and focal length.

2.4 Microscopy

Microscopy is the field of using microscopes to view samples and objects that cannot be seen with the naked eye [16]. Microscopy is important because it allows for the viewing of cell culture and small organism experiments which are too small to be seen with the naked eye. Microscopy is essentially done by magnifying a captured image. Therefore, we need to understand the properties of image gathering and magnification to determine the parameters that affect the quality of microscopy.

2.3.1 Application of Image Gathering in Microscopy

We can use the property of focal length to magnify an image using the Thin Lens Formula. In **Figure 3** below, f is the focal length of the lens, and S_1 and S_2 are the distances of the object and image to the center of the lens, respectively.



$$1/f = 1/S_1 + 1/S_2 \text{ (Thin lens formula)}$$

$$\text{Magnification} = S_2 / S_1$$

Figure 3: Thin Lens Formula [17]

The Thin Lens Formula is derived from a combination of Snell's Law and the Lens Maker's Equation. The Thin Lens Formula allows us to determine the distance the object source should be from the lens in order to get our desired magnification for the image [17]. This is extremely important in microscopy since, as mentioned earlier, microscopy magnifies an image in order for us to see microscopic objects such as cells and small organisms. Despite the fact that the object source and the image plane are not at the focal length, the image is still clear due to the proportionality of the Thin Lens Formula. **Figure 4** below is a simplified version of how lenses work together to create a magnified image. However, there are likely more than two lenses in a given microscope, as increasing the number of lenses can help to focus light and therefore provide a clearer image.

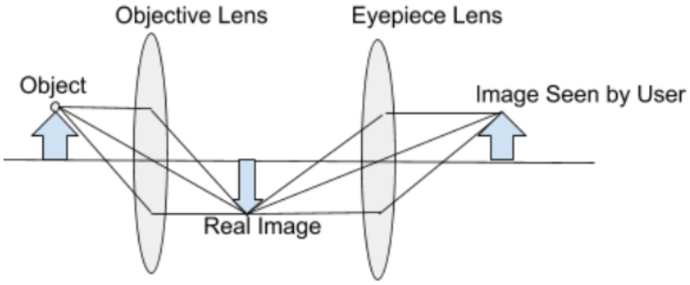


Figure 4: Image Production via Microscope Lenses

Despite there being a general alignment of lenses to create an image, there are many ways to transmit light through the object, which changes the type of microscopy, and therefore the image quality. One reason for this is because the density and material properties of objects vary, meaning they have their own refractive index. This affects the direction of light reflected off the object, and consequently the resulting focused image. There are three main types of light transmitted microscopy using visible light: bright-field, dark-field, and phase contrast. Additional forms of microscopy include differential interference contrast, Hoffman modulation contrast, and fluorescence. These forms were not considered for this paper as they use more complicated means to achieve higher contrast levels which would be difficult to replicate on this scale.

2.3.2 Brightfield Microscopy

Brightfield Microscopy is the easiest method of light transmitted microscopy. In brightfield, light is transmitted straight through the sample. The image contrast is created by the absorbance of the transmitted light in denser areas of the sample. Because many biological samples are transparent, a lot of brightfield images do not show a lot of contrast. Therefore, scientists stain the samples to produce contrast proportional to the density of the sample or to bind to specific biological components [17]. An example of the comparison between unstained and stained samples can be seen below in **Figure 5**. Staining of samples is not feasible for the applications of this device though as staining cannot be done on live cell or organism imaging.

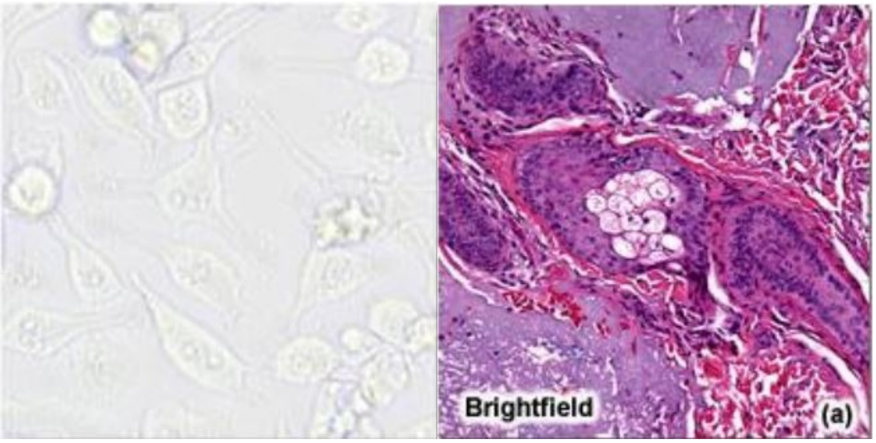


Figure 5: Not Stained (Left) vs. Stained (Right) Brightfield Image [17]

2.3.3 Darkfield Microscopy

In darkfield microscopy, the light is condensed before hitting the sample so that all of the light waves are concentrated into the object. Light is condensed using an annulus, which has holes that only allows some waves to pass through, increasing the light density through those holes. The bright, straight-on light hits the specimen at an angle, meaning it’s reflected away from the objective lens after passing through the specimen. The indirect light waves are diffracted by the specimen due to the different diffraction indexes within the specimen’s structure. Because these are indirect light waves, they are less bright due to having less photons and having a lower photon density. Therefore, there is more space between each photon, making diffracted waves very sensitive to changes within the specimen. The contrast in the image is caused by scattering light from the sample. Because of the sensitivity, samples do not need to be stained to create contrast. Then, the diffracted light rays pass through the objective lens and are magnified to create the image as shown below in **Figure 6** [17].

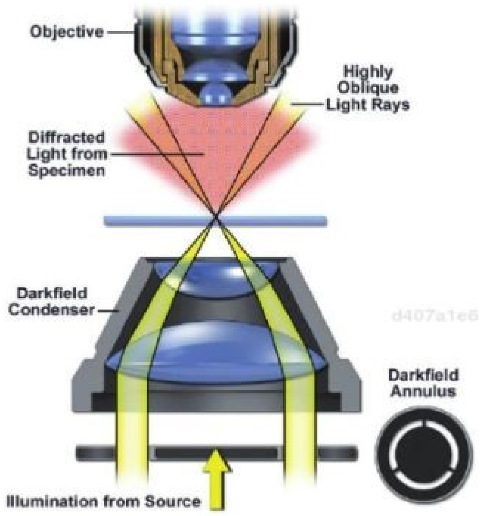


Figure 6: Dark Field Microscopy [17]

2.3.4 Phase Contrast Microscopy

Phase contrast gets its name because it takes the phase shift caused by a specimen and converts it into an amplitude, creating contrast. This is done by comparing the diffracted waves to surrounding bright light as shown below in **Figure 7**.

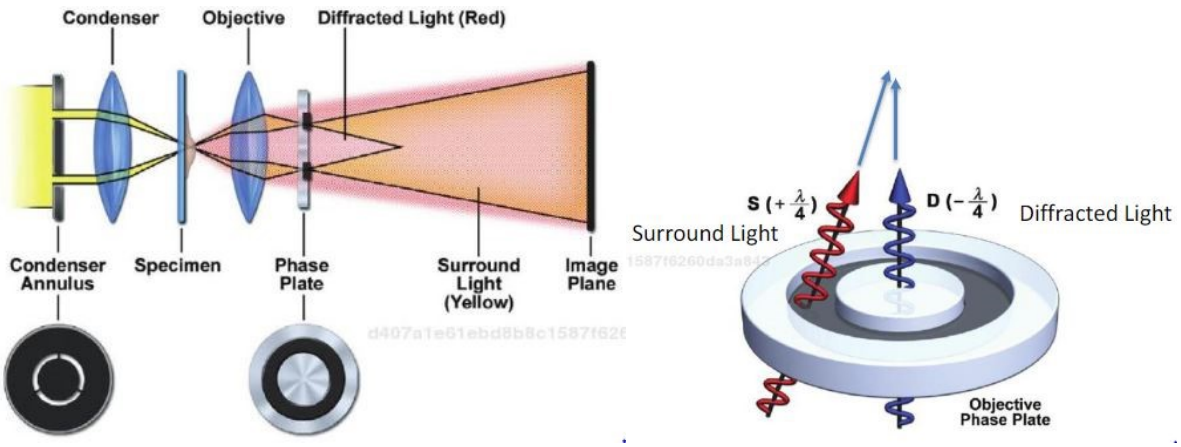


Figure 7: Phase Contrast and Phase Plate Hardware [17]

The waves are condensed using an annulus, as explained in section 2.3.3. The condensed and diffracted waves then hit the specimen and then pass through the objective, unlike in darkfield microscopy where only the diffracted waves pass through the objective lens. The condensed light is very dense and is therefore not as sensitive to changes in the specimen. Both diffracted light and condensed light go through the phase plate, where the condensed light is now considered to be surrounding light. In order to assure the surrounding light can pass through the phase plate, the focal length of the objective lens has to be calculated very precisely using the Thin Lens Formula. With both the surrounding light and diffracted light passing through the phase plate, we can compare the difference in light intensity, creating contrast [17].

2.3.5 Digital Image Sensors and USB Microscopes

Being able to see images due to the refraction and reflection of light waves through lenses is very useful to our society. However, in our digital age, being able to replicate these results on a screen aids with further development of the image to the user's desire. This has led to the increased use of digital image sensors, especially in the form of USB microscopes.

Digital cameras take images via an image sensor, which primarily uses complementary metal-oxide-semiconductor (CMOS) technology. A CMOS sensor has photodetectors that correspond to each pixel in the image. These photodetectors capture an analog voltage signal from the energy of the photons and the frequency at which the photons are hitting the sensor. This signal gets amplified and converted into a digital signal relating to the brightness of the photons using a digital to analog converter. To get color data, an RGBG Bayer filter is layered over the array of photodetectors to mimic the human eye, and a computation program produces the full-color image. The job of a lens in a USB microscope is to focus the light onto the sensor to produce a crisp image. The more photodetectors the CMOS has, the more pixels the image can be divided into, which directly correlates to the number of megapixels (MP) a camera has. For example, an 8 MP sensor has 8 million photodetectors [18].

Digital magnification manipulates the image it's capturing to make it larger. When the image size is increased, the size of each individual pixel is also increased, which causes the image to look pixelated. To combat this, viewing software adds more pixels into the image and interpolates their corresponding voltage based on the photon voltage the CMOS' photodetectors are reading. Therefore, it is imperative that the USB microscope's CMOS sensor has a high resolution, aka a high number of photodetectors; so that when the image gets larger, it can be divided into more pixels [19].

USB microscopes have become increasingly more popular as they are significantly cheaper than tabletop microscopes. Therefore, most USB microscopes use brightfield microscopy since it's the simplest and cheapest option which requires the least amount of hardware. Focus and magnification in a tabletop microscope is adjusted via moving the sample's platform. However, in

USB microscopes, magnification is changed using a digital camera within the microscope [19]. While the low cost of these devices is certainly an advantage, they do have certain limitations as compared to their large-scale counterparts. USB microscopes are able to capture an image with magnification sufficient to observe cells (< 10 μm), but compound microscopes can reach diffraction limits of ~300 nm. Additionally, USB microscopes generally have fewer lenses which leads to more aberrations, and the lenses are often made of plastic which tends to have a worse image quality than glass lenses. However, the cost to effectiveness of these USB microscopes is still relatively high, and therefore is desirable for increasing the access of microscopy.

2.4 Existing Device Research

2.4.1 Commercial Devices

Our project aims to create a device similar to devices on the market but with specific differences to increase the accessibility of such devices. These devices are what we will be referring to as the gold standard for our application. There are other monitoring devices on the market which do not use microscopes and instead use more indirect monitoring methods, but these devices do not gather the specific information we wanted as part of our design. The gold standard continuous monitoring devices which use microscopes fall into two major categories. The first category, which has been around longer, is a device which consists of a standard high-end phase contrast microscope with a built-in incubator component. One main example of such a device is the Zeiss live cell imaging device [20]. As you can see in **Figure 8** below, the device creates an incubator-like environment using a plexiglass chamber around the microscope stage and attached O_2, CO_2, and temperature modules [20].

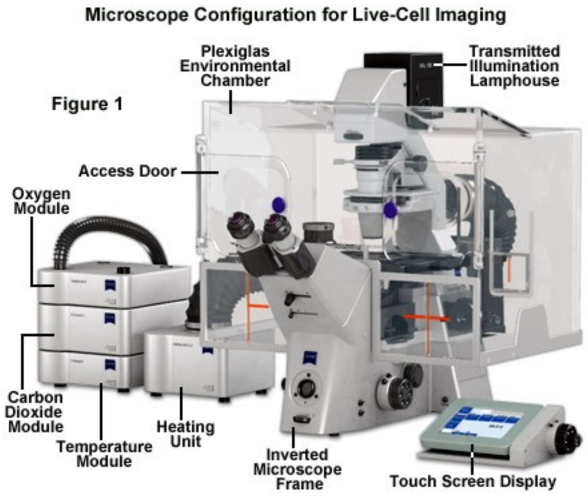

Microscope Configuration for Live-Cell Imaging
Figure 1
Plexiglas Environmental Chamber
Transmitted Illumination Lamphouse
Access Door
Oxygen Module
Carbon Dioxide Module
Temperature Module
Heating Unit
Inverted Microscope Frame
Touch Screen Display

Figure 8: Zeiss Live Cell Imaging System [20]

An additional gold standard device which includes its own incubation is the Viva View FL Incubator Microscope [21]. This device is an inverted microscope built into a high-tech incubator [21]. **Figure 9** below shows the outside of the device which looks similar to a standard incubator and has an attached computer module. The device is capable of both fluorescence and phase contrast microscopy [21]. However, this device can only image samples in 35 mm diameter glass bottom plates.

Figure 9: Viva View FL Incubator Microscope [21]

The other category of devices capable of continuous monitoring are small microscope systems which fit inside standard incubators. An example of these devices is the CytoSMART Lux2 shown below in **Figure 10**. This device is capable of bright field microscopy only and has a fixed 10x objective [8]. The system has an open design to allow the imaging of any transparent culture vessel [8].

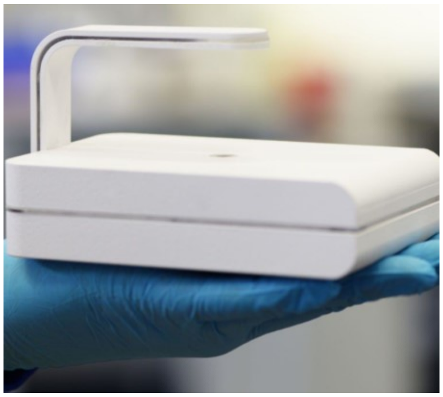


Figure 10: CytoSMART Lux2 [8]

With these three gold standard devices identified, we completed an analysis comparing the pros and cons of each device to find a comparable gold standard to our goal device. This comparison can be seen in **Table 1** below.

	Zeiss Live Cell Imaging	Viva View FL	CytoSMART Lux2
Pros	● High quality microscope ● Differential phase contrast compatible ● Environmental control	● High quality microscope ● Differential phase contrast compatible ● Environmental control	● Compatible with standard incubators ● Usable with any transparent vessel
Cons	● Not compatible with standard incubators ● Low quality incubation ● Only 1 sample at a time	● Not compatible with standard incubators ● Exclusive 40x objective ● Can only image 35mm diameter glass bottom plates	● Bright field only ● Fixed 10x objective ● Fixed height difference between sample and objective ● No environmental sensing

Table 1: Current Devices Pros and Cons

From this analysis we chose the CytoSMART Lux2 as a comparable gold standard for comparison to our device. This system has a small size with dimensions of 13.3 x 9.0 x 10.0 cm (L x W x H) and costs roughly $2500 to purchase [8]. Based on the pros and cons for this device, we focused on making a device which had more potential for customization than the CytoSMART device and which could be recreated for a lower cost.

2.4.2 Low-cost Non-commercial/Research Lab Devices

Other labs have also begun to investigate low-cost mobile microscopy as an alternative to these expensive gold-standard devices. Many of these devices are built using easily accessible and low-cost components such as smartphone cameras, microcontrollers, and digital cameras. In many cases these labs augment the camera with added lenses. Some techniques used by other labs to develop low-cost small size devices include the use of an LED Array. This allows various kinds of multi-contrast images such as bright field, dark field, and differential phase contrast (DPC) to be acquired through the use of various illumination patterns of the light source [22]. In this paper, a Raspberry Pi personal computer controls the LED array illumination patterns and the images are captured through the use of an image sensor [22]. The center part of the led array being lit equates to bright field, dark field can be achieved by illuminating LEDs around the edges of the array such that the numerical aperture (NA) of the illumination is beyond the NA of the objective, and DPC is computed as the normalized difference between two intensity images taken with opposite illumination angles [22]. The difference in DPC is related to the object’s phase gradient along the axis of symmetry [22]. The size defining the transition from bright field to dark field is dependent upon the numerical aperture of the objective [22]. This Raspberry Pi based microscope consists of a 10× (NA, 0.25) objective lens, a 150 mm focal length tube lens, a raspberry pi camera module camera and an 8 x 8 Red Blue Green LED array measuring 5.3 cm x 5.3 cm and can be seen below in **Figure 11** [22].

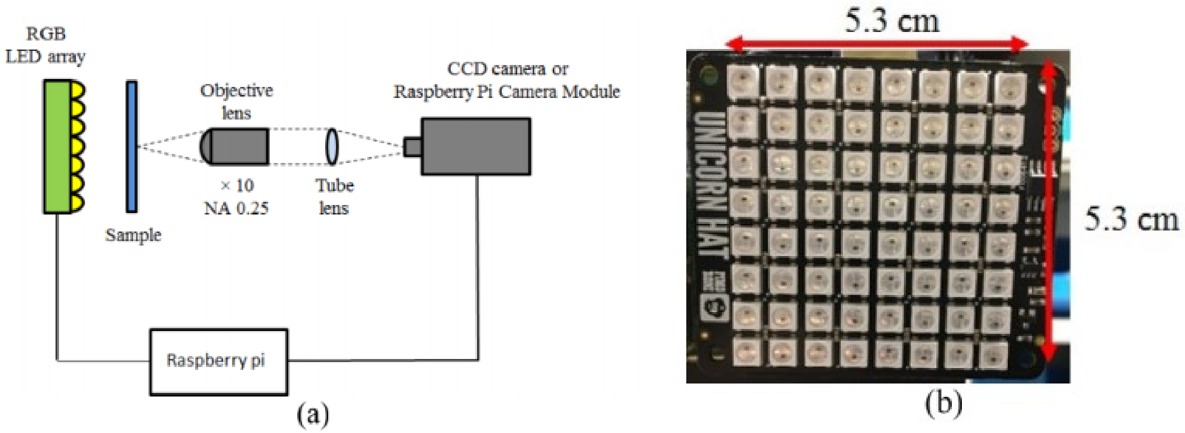


Figure 11: (a) Schematic for a Raspberry-pi-based LED array microscope and (b) picture of a RGB matrix LED array [22]

An alternative solution which has been studied is low-cost, sub-micron resolution, wide-field computational microscopy. In conventional imaging the number of pixels in the detector array constitutes a hard limit on the space-bandwidth product (SBP) and so therefore an increased field of view can only be achieved through a reduced spatial resolution [23]. This paper investigates the use of a low-cost, wide-field, high-resolution Fourier-ptychographic microscope (FPM) made of 3D printed opto-mechanics and a Raspberry Pi for data acquisition as can be seen in **Figure 12** [23].

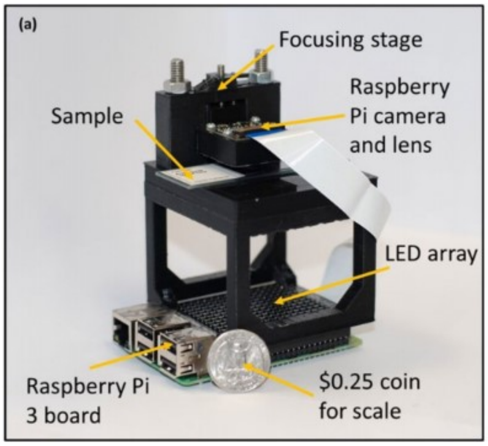


Figure 12: Raspberry Pi 3 single-board computer board based experimental setup next to a quarter US dollar for scale [23]

In this application, high SBP images are constructed from multiple low-resolution images captured in time sequence using oblique illumination angles [23]. Stitching of the images in the frequency domain is implemented using an iterative phase-retrieval algorithm to recover high-resolution amplitude and phase of the sample image as well as aberrations due to the objective [23]. These solutions have some of the aspects of the gold-standard devices, but do not achieve it all in one system. Additionally, both of these systems are designed to be used with fixed sample imaging rather than live sample imaging.

2.5 Internet Connectivity and IoT Devices

The Internet of Things (IoT) is defined as the interconnected network of internet-based devices that has grown over time since more and more devices have been created which connect to the internet [24]. IoT device examples cover many different technological categories including ATMs, smart watches, smart baby monitors, laptops, cell phones, microcontrollers, and devices such as the Amazon Echo or Google Home [24]. The nature of a remote monitoring device necessitates a connection to the internet to allow device to device wireless communication meaning that such a device would fall under the IoT umbrella.

IoT devices use server protocols to fetch and send information through the internet. For the purposes of this project we will be focusing on HTTP. HTTP is the foundation of any data exchange on the internet and is a client-server protocol which means that requests for information are initiated by the recipient [25]. Most of the time the recipient is a web browser page which sends its requests to the server [25]. Once the request is sent it passes through multiple layers to reach the server which sends the HTML document which represents the page [25]. HTTP does not directly control the connection as this occurs at the transport layer and HTTP resides at the application layer of the network, so it relies on the Transmission Control Protocol (TCP) which lets two hosts connect and exchange data streams [25].

Every internet connected device is considered a host, and each host has layers that allow the device to interface with the internet and other devices via the internet. The layers of a host include the Access Layer, Transmission Layer, Internet Layer, Linked Layer, and PHY layer, in order of how closely they interact with the internet. The purpose of the Access Layer is to connect data with the specified application, the Transmission Layer ensure reliable data delivery, the Internet layer facilitates routing/communicity across data networks, the Linked Layer negotiates access to the medium the data is being sent across in the presence of other hosts, and the PHY layer converts the data into an electromagnetic with a specific frequency, such that it can be sent wirelessly. If a device is connected to the internet, it has an IP address on the internet and externet side. Externet is the network that is interfaced between the modem and the service provider. Modems are given an IP address accessible by the service provider, as well as a local IP address that interfaces with the devices connected to it. This is necessary such that the modem can pass the externet connection to private devices, without sharing their IP addresses to the public. The network between a modem and its connected devices is therefore called the internet. The modem also assigns IP addresses to the devices connected. These IP addresses are typically dynamic, unless specified otherwise by the network service provider. The IP address of a device is stored in its Internet Layer, and devices can interface with other devices using their IP addresses, given they are on the same network.

Chapter 3 - Project Strategy

3.1 Initial Client Statement

The initial client statement that we were given by our advisor was as follows: design a compact, inexpensive internet-enabled biosensing and video monitoring system. The video and data should be logged automatically and accessible remotely via the internet.

3.2 Technical Design Requirements

Overall, we had several design objectives based on our initial client statement which are shown in the tree in **Figure 13** below:

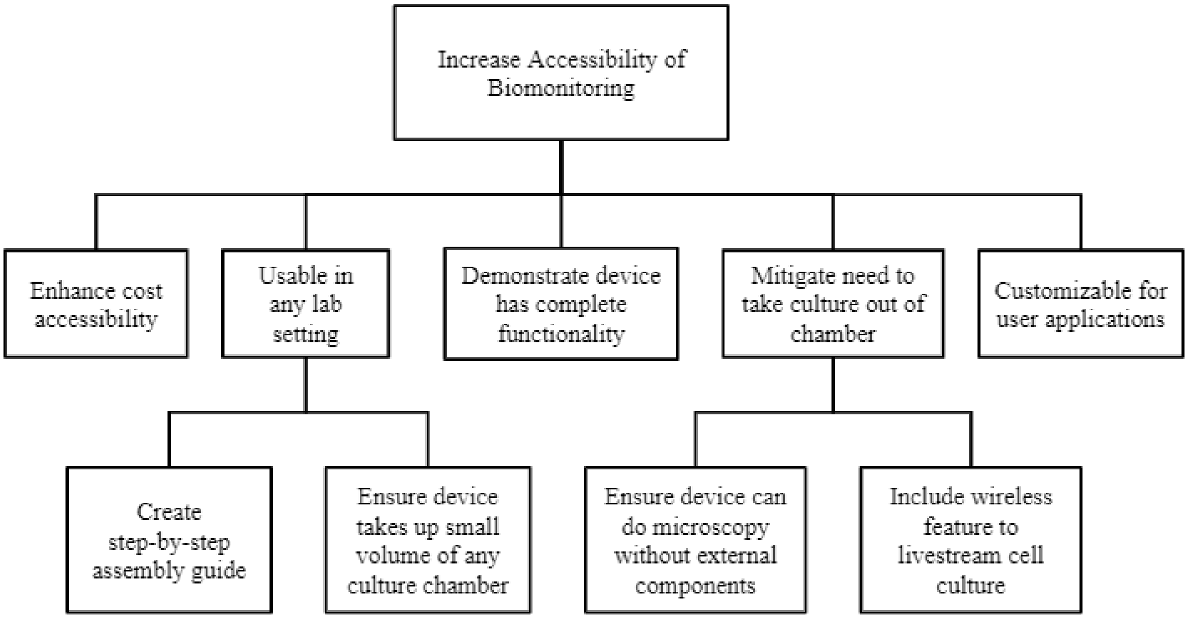

Increase Accessibility of Biomonitoring
Enhance cost accessibility
Usable in any lab setting
Demonstrate device has complete functionality
Mitigate need to take culture out of chamber
Customizable for user applications
Create step-by-step assembly guide
Ensure device takes up small volume of any culture chamber
Ensure device can do microscopy without external components
Include wireless feature to livestream cell culture

Figure 13: Objectives Chart

In addition to these design objectives, the device should not directly interact with the culture as to prevent negative impact on the cells/organisms being cultured. Cells and organisms can be easily influenced by toxicity in proximity or direct contact. Therefore, it must be ensured that all potentially toxic materials are kept several inches away from the culturing plate. Additionally, there are non-contact possible sources of contamination or alteration to the culture, such as heat or illumination generated from the device in proximity. Therefore, these possible additional issues must be accounted for and addressed in order to ensure a safe environment for the culture.

However, in order to know how to apply the design objectives towards creating a device, the parameters for the application for the device must be understood first. These are shown below in **Table 2**.

	C. elegans plate	*C. elegans* microfluidic	Cell Culture
Field of view	35 or 60mm plates	30 x 30mm 16 x 16mm	1.4 x 1mm
Required resolution	50 pixels/mm	50 pixels/mm	1200 pixels/mm
Required frame interval	2-3 frames/sec	2-3 frames/sec	4 frames/hour
Required time duration for recording	Few minutes for locomotory state, 12 for sleep/long-term behavior	2-3 hours, 12 for sleep	5 days

Table 2: Application Needs

The table above was constructed based on the needs of our advisor for *C. elegans* culturing due to their own experimentation. As mentioned in the background, *C. elegans* age quickly, where they reach adulthood in 3 days. Therefore, the frame interval is 2-3 per second and duration of recording must also be 2-3 hours. On the other hand, cells demonstrate changes very slowly, so it is more reasonable to record them at a rate of 4 frames/hour, with experiments lasting about 5 days. Additionally, a nominal field of view to observe cell count/migration in a cell culture experiment is 1.4 mm x 1 mm. In order to calculate the resolution for cell culture, we must know the resolution of the image sensor used and the size of the cell being studied. Therefore, we hypothesized using an image sensor with a resolution of 1600 x 1200, since it provides a relatively clear picture to the human eye, while ensuring the device remains as affordable as possible. The average mammalian cell is between 10-100 µm in diameter. Assuming a 10x smaller pixel size than cell size and the aforementioned resolution, this allowed us to calculate a resolution of approximately 1200 pixels/mm both directions.

3.3 Design Requirements (Standards and Regulations)

As one of our deliverables, we developed a step by step guide on how to recreate our device for educational purposes. In order for our device to be reproduced successfully and in compliance with the regulations, we had to establish a set of standards to keep in mind. Below are our standards categorized by different aspects of our device.

3.3.1 Computer and Electronic Equipment

Underwriters Laboratories published a set of standards for Information Technology Safety that apply to our device under the UL 60950-1 set of standards. These standards cover protection from potential hazards, physical requirements, and wiring and connection to networks.

3.3.2 Software Coding

The ISO 9000 standard series are standards to document quality management and assurance of the system in order to be able to maintain a properly functioning system. These standards can be applied to the software aspect of the device as well as the system as a unit.

3.3.3 Wifi

The Institute of Electrical and Electronics Engineers has developed a set of standards for the usage of WI-FI in multiple frequencies under the IEEE 802.11 standard series. These standards are applicable for devices that connect and have access to the internet without using wires. Some of the frequencies included within these standards are 2.4, 5 and 60 GHz frequency bands.

3.3.4 Cell/C. elegans Culture

The ISO 20387 standard discusses requirements to ensure quality and competence of biological materials and the data collection related to them. These standards are applicable to microorganisms used for research and development.

3.4 Revised Client Statement

A low cost, easy to use device which will fit in already existing incubators and be capable of remotely monitoring cell/*C. elegans* cultures through the use of live video capabilities. The device needs to be able to connect to WiFi and use web-based monitoring software to control image capture. Additionally, the device should be equipped with a lighting apparatus which allows for viewing of the samples through contrast, and an image sensor capable of viewing small organisms and cells. A secondary goal is to include sensors for humidity and temperature, and image segmentation capabilities to determine pH.

3.5 Management Approach

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)
A Term Goals BME 4300				
Complete Draft of Chapter 1	9/12	9/16	0	4
Complete Outline of Chapter 2	9/12	9/16	0	4
Presentation #3	9/16	9/20	4	4
Complete Draft of Chapter 3	9/16	9/25	4	9
Complete Draft of Chapter 4	9/23	10/2	11	9
Final Draft of Chapters 1, 3, & 4	10/2	10/8	20	6
Presentation #4	9/23	10/3	11	10
A Term Goals MQP				
Research based off of revised project goal and directions	9/13	9/23	1	10
Create a list of design objectives and user needs for the device	9/13	9/23	1	10
Begin designing the device parameters based on goal, research, and design requirements	9/23	10/3	11	10
Prototype designs	10/3	10/10	21	7
Begin working with the Raspberry Pi	9/26	10/10	14	14
B Term Goals				
Experiment with image capture options	10/22	12/7	40	46
Connect Raspberry Pi to Internet	10/22	12/7	40	46

Write paper chapters	10/22	12/13	40	52
Design basic housing options	11/27	11/30	76	3
CAD model housings	12/13	1/15	92	33
Upload Motioneye to Raspberry Pi	12/13	1/15	92	33
C Term Goals				
Test Light Source Options	1/15	1/29	125	14
Work out bugs in MotionEye code	1/15	1/29	125	14
Test image source, light source, and Pi together	1/29	2/12	139	14
Assemble device with housing and test device with cells and *C. elegans*	2/12	2/26	153	14
Writing paper	1/15	3/6	125	51
Work on incubator-proofing of device	2/12	3/6	153	23
D Term Goals				
Add Sensors to Device				
Determine which sensors to use	3/30	4/1	200	2
Write pseudo code for sensors	4/2	4/7	203	5
Determine how the sensors will be attached to the device	3/30	4/1	200	2
Order Sensors	3/30	4/1	200	2
Attach sensors to device	4/8	4/9	209	1
Integrate IP Display into system	4/12	4/16	213	4
Test sensors by simulating conditions	4/8	4/18	209	10
Validation				
Run Experiments	4/1	5/1	202	30
Paper and Poster				
Edit Presentation	4/1	5/1	202	30
Work on Paper	3/25	5/11	195	47
Presentation Practice Days	5/1	5/4	232	3

Figure 14: Gantt Chart

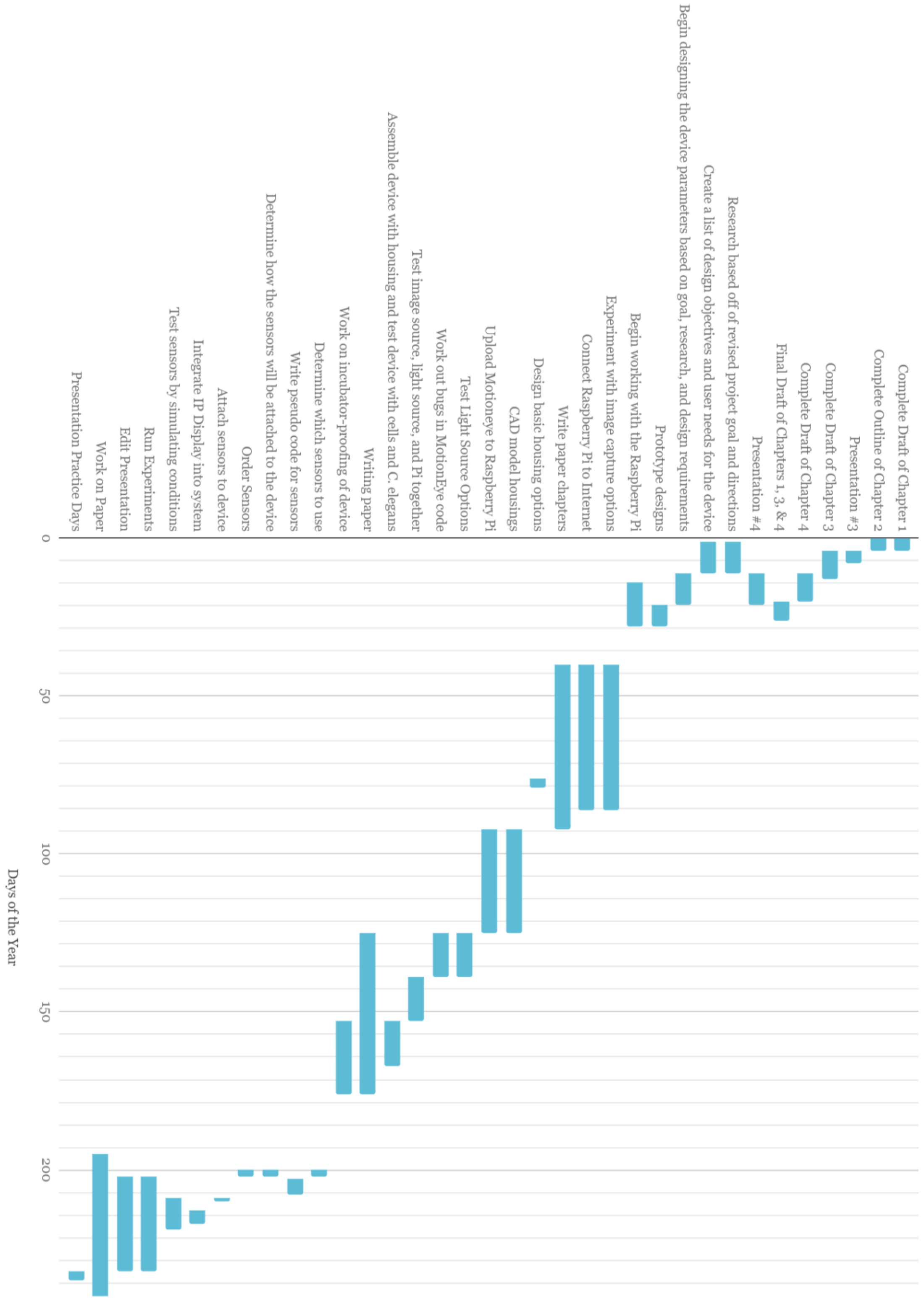


Figure 15: Full Year Calendar

This Gantt chart and calendar show our plan for the course of our four term project. Much of the preliminary device design testing was accomplished in B term and then functional testing and reworking of the preliminary designs occurred in C term. The majority of the work in B and C terms is shown as occupying the same time on the calendar as work was done concurrently by different members of the team at each time. In D term we completed verification testing of the device and finalized our paper and presentation.

Financially, we had a $750 budget for this project. We aimed to make this device cost effective for late high school and early college. In order to test as many alternative design options as possible, and to create multiple systems to complete verification testing in D term we had to purchase many components. Overall, we spent $642.04 on supplies for the project. The full breakdown of expenses can be seen below in **Table 3**.

Item	Cost	Amount	Total
Case base	$5.00	3	$15.00
case lid	$3.00	3	$9.00
Pi 3B+	$35.00	4	$140.00
Unicorn Hat 8 x 8 LED Array	$29.95	1	$29.95
Housings	$0.00	1	$0.00
Sugru	$16.99	1	$16.99
screws	$5.45	1	$5.45
nuts	$8.85	1	$8.85
Micro SD cards 3 pack: 32 GB	$12.39	1	$12.39
Regular SD cards 3 pack: 32 GB	$11.99	1	$11.99
Power cords	$8.99	3	$26.97
keyboard and HDMI	$20.00	1	$20.00
Temperature + Humidity Sensor	$7.00	2	$14.00
USB microscope (white)	$39.56	1	$39.56
Arducam (5 MP)	$9.49	1	$9.49
Raspberry pi camera(2.1)	$29.95	2	$59.90
SD card: 8GB	$5.97	1	$5.97
USB microscope (black)	$79.95	1	$79.95
SD card: 32 GB	$7.99	1	$7.99
Raspberry pi zero	$14.00	2	$28.00
Raspberry pi zero case	$5.95	2	$11.90
autofocus cam	$14.99	1	$14.99
White LED Backlight Module	$2.95	2	$5.90
Pi Camera Board Case with 1/4" Tripod Mount	$2.95	1	$2.95
Unicorn Hat 16 x 16 array	$34.95	1	$34.95
Mini Display	$14.95	2	$29.90
		Total spent:	$642.04

Table 3: Expenses

While we spent $642.04 overall, the actual cost of one system is much less. The breakdown of the estimated cost to replicate this device can be seen below in **Table 4**. Note that the cost for the housing in this table differs from the cost of our housing as we had access to a 3D printer where the housing could be printed for free. Additionally, there may be additional shipping costs associated with the purchase of each of these components which we did not have to pay because of the large scale of the orders in which our components were purchased. Sensor costs were not included in the singular device cost as they are an optional addition. The final estimated cost of this device is $191.22 which falls below our initial goal of $200.

Item	Cost	Amount	Total
Case base	$5.00	1	$5.00
case lid	$3.00	1	$3.00
Pi 3B+	$35.00	1	$35.00
Unicorn 8 x 8 LED Array	$29.95	1	$29.95
Housing	$30.00	1	$30.00
Sugru	$16.99	1	$16.99
Screws	$5.45	1	$5.45
Wing nuts	$8.85	1	$8.85
SD card (32 GB)	$7.99	1	$7.99
USB Microscope	$40.00	1	$40.00
Power cord	$8.99	1	$8.99
		Total:	$191.22

Table 4: Single Device Cost

Chapter 4 - Design Process

4.1 Needs Analysis

Given our revised client statement, we divided our design requirements into our primary and secondary goals, primary defining our devices' needs and secondary defining our devices' wants.

4.1.1 Needs

Based on the client's application needs specified in Table 2, **Table 5** below demonstrates the ideal physical design parameters of our device to ensure maximum compatibility and efficiency:

Design Requirement	Corresponding Parameter	Ideal Value
Small	Volume Occupied by Device	10 cm x 10 cm x 10 cm
Sufficient Frame Size	Maximum Viewing Area	30 mm x 30 mm
Light Source setup for Microscopy	Direction of Light	Around sample for Dark Field Microscopy
Functions in High Humidity	Sealant Around Device	Withstands 90 - 100% Humidity
Clarity of Image	Resolution	1600 x 1200 pixels
Processing Ability	Minimum RAM	512 MB
Sufficient Data Storage	Minimum Memory Storage	32 - 128 GB depending on experiment type
Streaming Ability	Data Transmission Rate	25 mbps

Table 5: Ideal Device Parameters

Values for the device and field of view size were given by our original problem statement from our advisor. Direction of light was determined by background research from our advisor's lab, where it was discovered that *C. elegans*, our choice of organism to monitor, are clearly imaged when light is diffused by placing it very far away from the sample or placing a privacy film to cloud the light [26]. This results in a similar effect as dark field microscopy (DFM), since dispersing the light leads to less dense light and less photons per light ray, allowing for sensitivity to change in the sample. It is also mentioned in Section 2.4.2 that other labs have used both dark field microscopy and simulated phase contrast techniques with optical tricks to clearly image cells [22]. An evaluation of these methods will be discussed in Section 4.2.

Based on the preliminary photos taken of *C. elegans* taken with the highest resolution imaging sensor and interface of our device, where the field of view was 3 mm x 3 mm, the maximum memory taken up by each photo was 1.1 MB. Given we want to take 4 frames per hour for 24 hours for 5 days, that leads to the maximum memory occupied to be about 528 MB. However, other sensing processes, which will be discussed further in chapter 4, require a picture to be taken at least every minute, which results in 7.92 GB being consumed. For *C. elegans*, change is observed via pictures taken 2 frames per second (fps). Given *C. elegans* experiments last up to 12 hours overnight, this results in 86.4 GB being consumed assuming each picture takes up 1.1 MB. Additionally, 20 GB of leeway is allotted to download libraries for necessary programs for the interface, sensors, and light source.

Our microprocessor must perform the following minimum functions: controlling the light source, taking an image at a specified interval, and sending the image to an online interface. Through preliminary testing with the Raspberry Pi Zero W, it was determined that 512 MB is enough RAM to perform the minimum functions of our device given the images were taken and saved at a rate of 2 fps. Reasoning for using a Raspberry Pi as our microcontroller is described in section 4.2.

To determine streaming ability and clarity of image, current streaming services such as Hulu and Netflix were analyzed, and it was determined that their highest streaming quality in 1080p requires a maximum of 25 mbps [27]. However, the speed of the signal is primarily dependent on the wifi source and is therefore, not in the control of our device.

In order to ensure the electronic parts are not corroded due to high humidity in the chamber, the device must be protected by a sealant. The products our team researched to seal off our device from the humid environment in the chamber will be discussed in section 4.2.

4.1.2 Wants

Our secondary goal is derived from our wants as defined by **Table 6** below.

Aspect	Range
Temperature Monitoring	0.0 - 50.0 C ± 0.1 C
pH Monitoring	6.0 - 8.0 ± 0.1

Table 6: Wants Analysis

In addition to the aspects of our secondary goal in the table above, our group would also like to illustrate a step-by-step assembly guide to allow for further accessibility of this device, as it would give other educational labs the opportunity to perform low-cost wireless microscopy.

The parameters characterized in the above table were determined from the ideal parameters for cell and organism culture as explained in Sections 2.1 and 2.2 and a tolerancing amount. In order for sensors to effectively determine if an aspect is within ideal conditions, it must be able to withstand and monitor in a range larger than the ideal range.

4.1.3 Technical Constraints

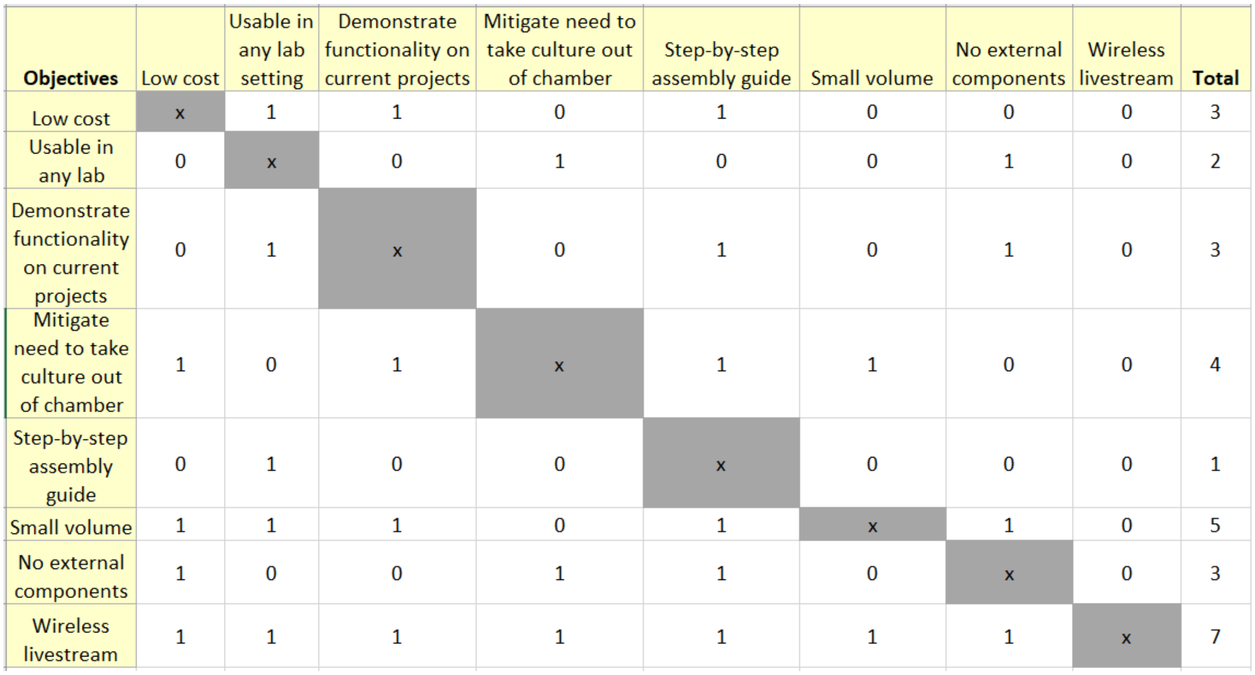
Along with our device needs and wants, it is important to consider the feasibility of our device by analyzing the corresponding constraints. The constraints are listed in **Table 7** below:

Constraint	Bounds
Withstand High Humidity	Up to 100%
Withstand High Temperature	Up to 37C
Range of Wireless Signal	Must be able to pass through incubator
Data Transmission Rate	Must allow for clear image
Memory Storage	32 - 128 GB depending on experiment
Power Consumption	Must be powered up to 5 days

Table 7: Device Operation Constraints

The aspects above were determined as constraints as they can adversely affect the functionality of the monitoring system. Humidity and temperature are known throughout the device industry to impact functionality as heat adds energy to the air which interferes with the energy flowing through the circuit, the electrons [28]. Humidity adds water to the air which allows for conductivity of charges through the air which could also impact the electrical system [29]. Range of the wireless signal and data transmission rate are constraints due to our concern that the signal cannot permeate through the incubator due to an incubator's insulation. Data transmission rate and range of the wireless signal can affect the livestream video received by the observer of the culture. If the data transmission rate is not high enough, the receiver will receive different pixels of the image at different times, which is when people often see lag in the picture. Similarly, if the range of the wireless signal of the computer is not high enough, then it will restrict the rate at which certain pixels of the image are sent or the signal may not be receivable at all, meaning there would be no interface able to retrieve the video. The parameters for humidity and temperature constraints were determined due to parameters of cell culture defined in Section 2.1. Calculation of the constraint for memory storage is described in Section 4.1.1.

Below can be found a pairwise comparison of our project objectives ranking them according to importance. In this comparison a 1 indicates that the objective in the row is more important than the objective in the column and a 0 indicates that the objective in the column is more important than the objective in the row. The total column represents the sum of the numbers for each row, allowing the objectives to be ranked.



Objectives	Low cost	Usable in any lab setting	Demonstrate functionality on current projects	Mitigate need to take culture out of chamber	Step-by-step assembly guide	Small volume	No external components	Wireless livestream	**Total**
Low cost	x	1	1	0	1	0	0	0	3
Usable in any lab	0	x	0	1	0	0	1	0	2
Demonstrate functionality on current projects	0	1	x	0	1	0	1	0	3
Mitigate need to take culture out of chamber	1	0	1	x	1	1	0	0	4
Step-by-step assembly guide	0	1	0	0	x	0	0	0	1
Small volume	1	1	1	0	1	x	1	0	5
No external components	1	0	0	1	1	0	x	0	3
Wireless livestream	1	1	1	1	1	1	1	x	7

Figure 16: Pairwise Comparison of Objectives

As can be seen in **Figure 16**, our highest ranked objective is the wireless livestream, with small volume being our next highest. These objectives being ranked highly shows how important it is for us to consider our design constraints highly when designing our final device. Multiple of our objectives are ranked evenly in importance, as they were all essential in achieving our overarching goal.

4.2 Concept Maps / Conceptual Designs, Design Concept Prototyping / Modeling, Feasibility Studies

Our initial conceptual design has a basic block diagram of a computing system with an LED array and optical microscope with image sensing capabilities.

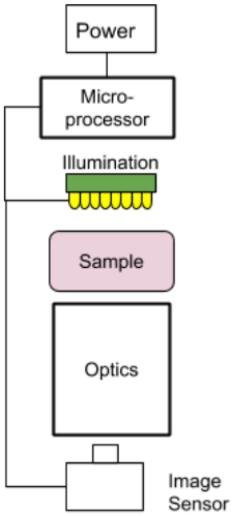


Figure 17: Initial Design Block Diagram

The main components of our initial design as shown in **Figure 17** are a microprocessor acting as control of the device, an image sensor and optics for image capture, and illumination to create image contrast. Depending on the image sensor chosen, the optics may be built in, such as in a USB microscope. The device can be powered using a battery pack or a wall outlet, depending on the environment for the experiment. This initial block diagram leaves room for alterations in the specific components based on alternative design testing.

A description of the interconnected features of our device design is shown below in **Figure 18**.

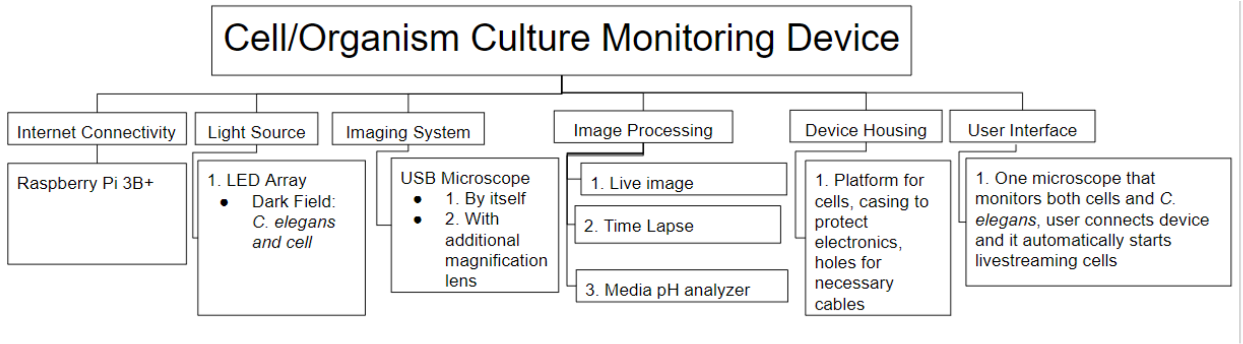

Cell/Organism Culture Monitoring Device
Internet Connectivity
Raspberry Pi 3B+
Light Source
1. LED Array
• Dark Field: C. elegans and cell
Imaging System
USB Microscope
• 1. By itself
• 2. With additional magnification lens
Image Processing
1. Live image
2. Time Lapse
3. Media pH analyzer
Device Housing
1. Platform for cells, casing to protect electronics, holes for necessary cables
User Interface
1. One microscope that monitors both cells and C. elegans, user connects device and it automatically starts livestreaming cells

Figure 18: Design Process Block Diagram

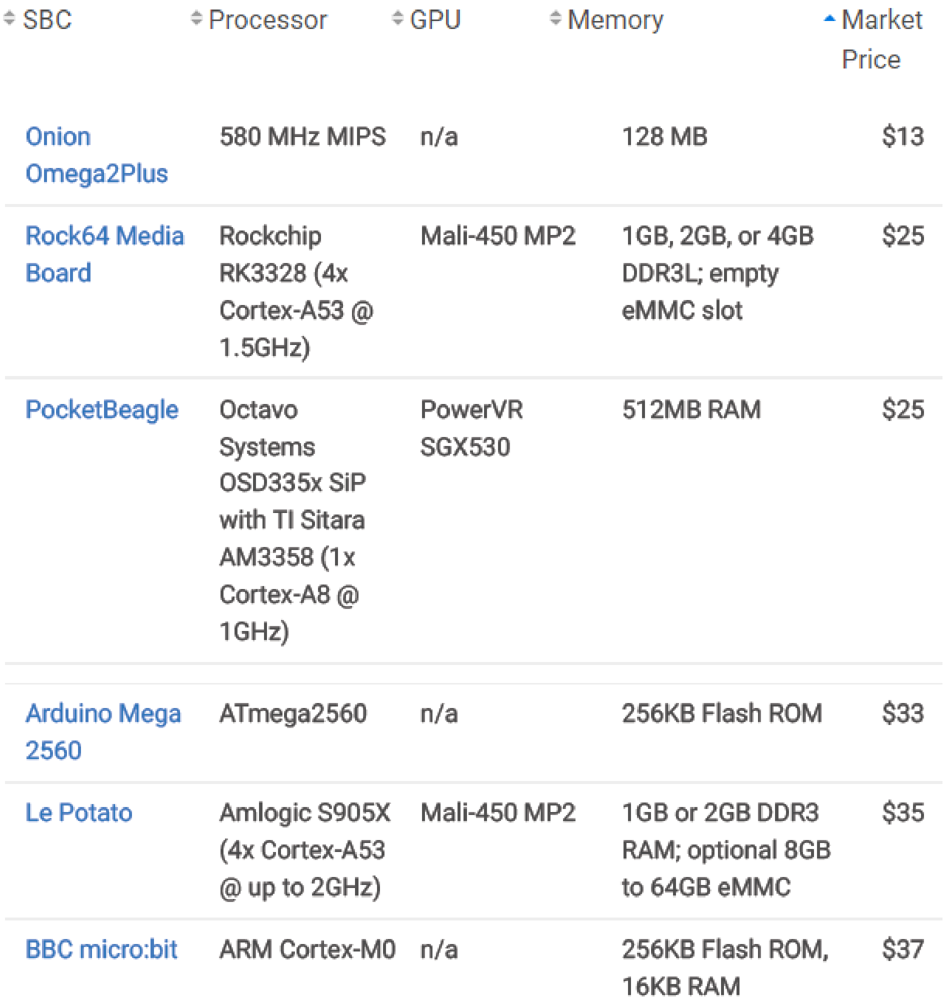
4.2.1 Microprocessor Feasibility Studies

Although our advisor recommended using a Raspberry Pi for the microprocessor, technical specifications of several other brands were also researched in preliminary studies to ensure we picked the best microprocessor for our design. The requirements for our microprocessor were chosen based on our design goals shown in Figure 11 and our design parameters in Table 5. They are as follows:

1. Must be able to send and receive wireless via the internet without external components
2. Must have at least 512 MB of RAM
3. Must be less than $50
4. Must have ports for camera connectivity

Figure 19 is a table from All3DP that lists alternative microprocessors considered outside of the Raspberry Pi family under $50. For each of these microprocessors with a memory of 512 MB RAM or greater, further research was done on the device's website to check if they were capable of transmitting and receiving data over the internet. Out of this list, only the Rock64 Media Board, the PocketBeagle, and the Le Potato had a RAM of 512 MB or greater. Upon further research on the company websites for each of these processors, it was discovered that none of them are capable of communicating wirelessly via the internet without a USB dongle.

Using a USB dongle for internet connectivity was not something we wanted to include in our device. This is because we wanted the user to have maximum opportunity for customization, which meant leaving as many ports available as possible. Additionally, setting up the dongle and downloading its supported packages if there are any would take up additional memory storage of the device and time for the user. Since it has to be purchased separately, this would also raise the cost of the device. It also has the potential to be lost given it is an external component, increasing the cost the user must invest in the device in the future. Therefore, we concluded that using a device with a USB dongle as opposed to having built in WiFi would be inadvisable given there are products on the market that have built in WiFi connectivity, such as the Raspberry Pi. Therefore, none of these alternative options were chosen as they were unable to meet our design parameters.

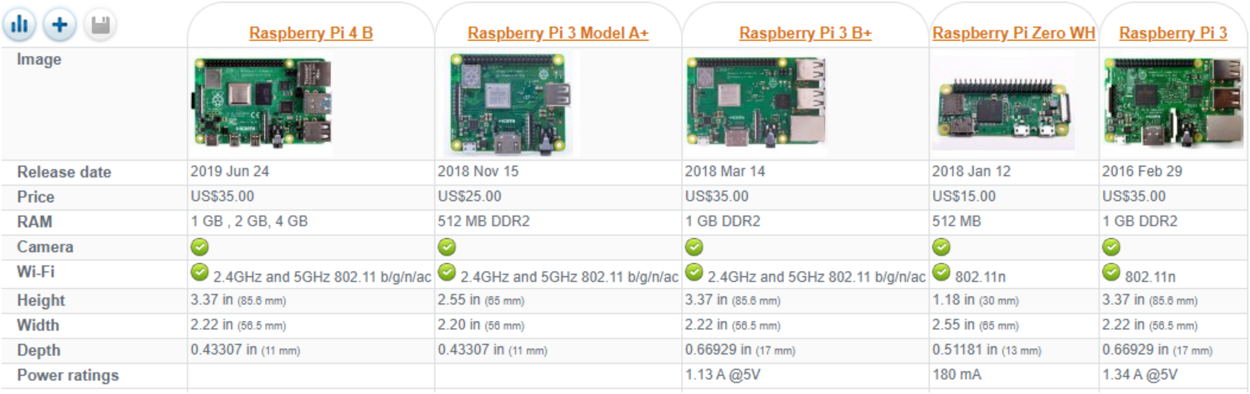


SBC	Processor	GPU	Memory	Market Price
Onion Omega2Plus	580 MHz MIPS	n/a	128 MB	$13
Rock64 Media Board	Rockchip RK3328 (4x Cortex-A53 @ 1.5GHz)	Mali-450 MP2	1GB, 2GB, or 4GB DDR3L; empty eMMC slot	$25
PocketBeagle	Octavo Systems OSD335x SiP with TI Sitara AM3358 (1x Cortex-A8 @ 1GHz)	PowerVR SGX530	512MB RAM	$25
Arduino Mega 2560	ATmega2560	n/a	256KB Flash ROM	$33
Le Potato	Amlogic S905X (4x Cortex-A53 @ up to 2GHz)	Mali-450 MP2	1GB or 2GB DDR3 RAM; optional 8GB to 64GB eMMC	$35
BBC micro:bit	ARM Cortex-M0	n/a	256KB Flash ROM, 16KB RAM	$37

Figure 19: Microcontroller Specifications Comparison Chart [30]

Preliminary research was done on various models of Raspberry Pi to determine which should be ordered for our alternative and final designs. The chart below in **Figure 20** shows Raspberry Pi Models compared due to their specifications. The criteria for choosing a particular Raspberry Pi were the same as mentioned previously for the microprocessors. The Raspberry Pi 3, and 4 were eliminated immediately, as the 3 was discontinued, the 4 had the largest dimensions. Out of the Raspberry Pi 3 A+, 3 B+, and Zero WH, we further narrowed our choices to the

Raspberry Pi 3B+ and Zero WH, since the Zero WH has the same capabilities as the 3A+, along with being smaller and cheaper, and the 3B+ had more RAM. The 3B+ and Zero WH will be compared in Section 4.3.



	Raspberry Pi 4 B	Raspberry Pi 3 Model A+	Raspberry Pi 3 B+	Raspberry Pi Zero WH	Raspberry Pi 3
Image					
Release date	2019 Jun 24	2018 Nov 15	2018 Mar 14	2018 Jan 12	2016 Feb 29
Price	US$35.00	US$25.00	US$35.00	US$15.00	US$35.00
RAM	1 GB , 2 GB, 4 GB	512 MB DDR2	1 GB DDR2	512 MB	1 GB DDR2
Camera	✓	✓	✓	✓	✓
Wi-Fi	✓ 2.4GHz and 5GHz 802.11 b/g/n/ac	✓ 2.4GHz and 5GHz 802.11 b/g/n/ac	✓ 2.4GHz and 5GHz 802.11 b/g/n/ac	✓ 802.11n	✓ 802.11n
Height	3.37 in (85.6 mm)	2.55 in (65 mm)	3.37 in (85.6 mm)	1.18 in (30 mm)	3.37 in (85.6 mm)
Width	2.22 in (56.5 mm)	2.20 in (56 mm)	2.22 in (56.5 mm)	2.55 in (65 mm)	2.22 in (56.5 mm)
Depth	0.43307 in (11 mm)	0.43307 in (11 mm)	0.66929 in (17 mm)	0.51181 in (13 mm)	0.66929 in (17 mm)
Power ratings			1.13 A @5V	180 mA	1.34 A @5V

Figure 20: Raspberry Pi Specifications Comparison Chart [31]

4.2.2 Lighting Feasibility Studies

In our background research, we determined that imaging cells in a low-cost system without staining is possible through both dark field microscopy and DPC. In order to create DPC, different areas beneath the sample must be lit at different times at wavelengths corresponding to red (660 nm), green (550 nm), and blue light (450 nm). All of these parameters need to be varying in differential phase contrast in order for the computer to receive all angles and absorbances of the sample to process them into one image. In order to simulate dark field microscopy, an LED array is lit around the sample, resulting in angled light hitting the sample [22]. The method of dark field microscopy was pursued over DPC as it required a simpler light configuration and code, while resulting in a clear image of cells, as shown in Watanabe's study and **Figure 21** below.

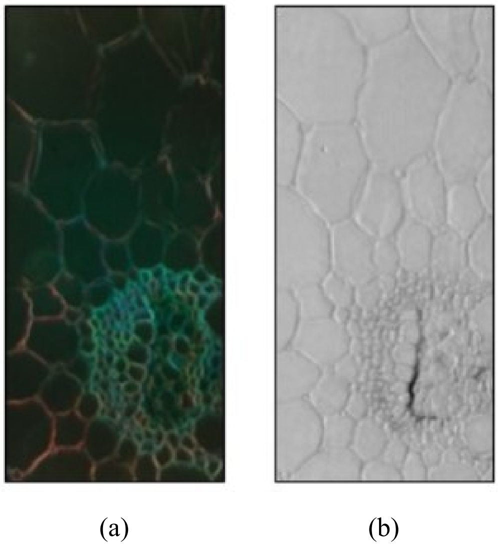


Figure 21: (a) darkfield, and (b) left-right DPC images of corn cells [22]

Also, our advisor's research shows that *C. elegans*' behavior changes with the presence of blue light [32], meaning it's possible that other colored light could affect the behavior of cell culture or other small organisms, which could skew the results of the user's experiment. Because we decided to pursue dark field microscopy, we considered using an external light source that lights up around the edges by design, such as the Ty Lite in **Figure 22** below:

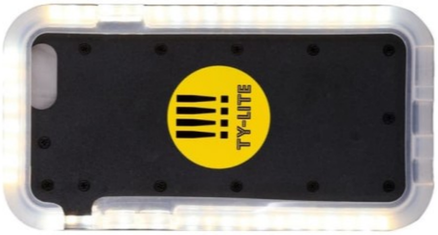


Figure 22: Possible External Light Source

However, since we wanted to experiment with different types of microscopy with our lighting source and ensure the user had the most opportunity to customize the device, we thought it made sense to choose a lighting source where the lights' position can be changed to the need of the user.

4.2.3 Humidity Feasibility Studies

To decide a feasible sealant for the device to prevent interference with humidity, online recordings of experiments were observed to understand how humidity can affect the device's ability to conduct current, as well as innovative materials that can waterproof consumer products

[33, 34]. Through these videos, we searched for a consumer available spray that allowed the user to coat an item with the hydrophobic substance such that water would not stick to the surface of the electronic components, but many results were for industrial use only, and therefore would not meet our accessibility objective of the device.

4.2.4 pH Reading Feasibility Studies

We considered two options to determine pH of the sample: measuring the absorbance of the sample's solution given a light with a known wavelength is transmitted or determining the pH via detecting the color of the sample. As mentioned in Section 2.1, an indicator is added to the solution of the sample, which changes the color of the solution depending on its pH. In common cell culture practices, the person conducting the experiment uses the color of the solution to determine when to change the media of the sample. Given this procedure, we theorized that using a color sensor or image processing program that extracts the RGB value of the sample would allow us to draw a correlation between pH and RGB values such that the user can take a picture of the sample and extract a pH based on its RGB value.

Out of these two options we eliminated measuring absorbance of the sample, due to the presence of cells possibly interfering with the absorbance. Also, phenol red experiences the most absorption when the light passed through the sample has a wavelength about 425 nm for pHs from 4 - 7 and about 560 nm for pHs 8 - 10 [35], corresponding to purple and yellow light, respectively [36]. As mentioned earlier in this section, shining colored light at the sample could affect the behavior of cell culture or other small organisms, which would go against our design objectives for this component.

Feasibility studies were also done to determine the most effective method to extract RGB values. Color sensors within our budget typically operate via shining a light on the sample, and then reading the RGB values of the light that is reflected off the sample using a photodiode. However, given the samples in our device would be transparent, this method seemed flawed for ensuring we would get the true RGB value of the sample, since the light from the color sensor would pass through the sample as opposed to bouncing off of it, meaning not all of the sample's color could be picked up by the photodiode. Also, because our design involves placing a light across the sample to allow for microscopy, it can possibly interfere with the light picked up by the photodiode that determines RGB value.

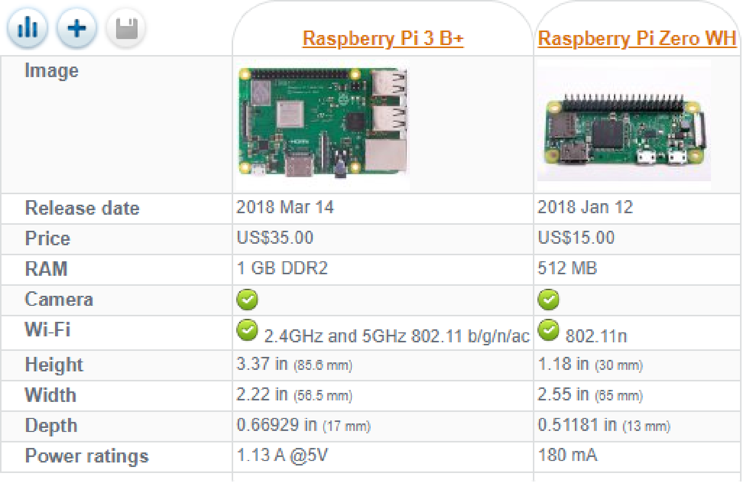
4.2.5 Housing

This device was designed to include a 3D printed housing structure to create the necessary support for the components of the device. We originally believed that with the use of a commercially available USB microscope we would have less need for this structure. In the case of the Raspberry Pi camera, we knew we needed this structure to hold the camera and objective lens. We realized however, that in both cases we would need the housing to hold the Raspberry Pi and LED array, and the sample container.

4.3 Alternative Designs

4.3.1 Raspberry Pi Model

As mentioned in Section 4.2, we narrowed our options for a microprocessor down to the Raspberry Pi Zero WH and the Raspberry Pi 3 B+. **Figure 23** below describes the specifications for both the Raspberry Pi 3B+ and Zero WH. As explained in section 4.1.1, 512 MB was determined to be a sufficient amount of RAM to complete the minimum functions required of our device through preliminary testing. However, when installing the required libraries and packages needed to set up the device, the Raspberry Pi Zero WH was significantly slower than the Raspberry Pi 3B+, causing visible lag in the desktop of the Raspberry Pi's OS.



	Raspberry Pi 3 B+	Raspberry Pi Zero WH
Image		
Release date	2018 Mar 14	2018 Jan 12
Price	US$35.00	US$15.00
RAM	1 GB DDR2	512 MB
Camera	✓	✓
Wi-Fi	✓ 2.4GHz and 5GHz 802.11 b/g/n/ac	✓ 802.11n
Height	3.37 in (85.6 mm)	1.18 in (30 mm)
Width	2.22 in (56.5 mm)	2.55 in (65 mm)
Depth	0.66929 in (17 mm)	0.51181 in (13 mm)
Power ratings	1.13 A @5V	180 mA

Figure 23: Raspberry Pi 3B+ and Zero WH Comparison Chart [31]

It is also worth noting that the power rating for the Raspberry Pi 3B+ is about 10 times higher than that of the Zero WH. Knowing the current ratings for the two microcontrollers allowed us to calculate the necessary amount of current needed to power the Raspberry Pi over the course of several days. As mentioned in Section 4.1.1, the maximum amount of time for a cell culture experiment is 5 days. Therefore, the Raspberry Pi 3B+ would need a battery pack with a current rating of approximately 135,600 mAh or power rating of 678,000 mWh, assuming the experiment is in a chamber without outlets. Given a battery pack with this capacity is not commercially accessible, it is recommended that if the 3B+ is used that the power cable plugs into an outlet outside the chamber with the Raspberry Pi inside the chamber. The Raspberry Pi Zero WH would need a battery pack with a current rating of approximately 21,600 mAh, which is commercially available to the average consumer for around $30.

When comparing the dimensions between the two microcontrollers, we noticed that the two had a similar width and depth. However, the height of the Raspberry Pi 3B+ is a little more than double that of the Zero WH. Regardless, the height and width dimensions of the Raspberry Pi 3B+ fit within the desired dimensions of our product as described in Section 4.1.1.

4.3.2 Remote Monitoring

A key aspect of this project is the ability to remotely monitor the culture. This means that the device needs to be capable of sending data wirelessly to the user. Multiple options were evaluated to complete this aspect. The first option evaluated was the MotionEye Operating System (OS) for the Raspberry Pi. This OS is made of open source code and can be downloaded from the internet onto any Raspberry Pi for free. This option would be accessible to anyone and could work on any Raspberry Pi model. Alternatively, we also considered downloading a version of Raspbian, a common RasPi operating system, that was compatible with Motioneye's independent software and use the combination of the two instead of MotioneyeOS. The Motioneye software has the same functionality as the Motioneye OS, but it allows the Pi to function as more than just the remote data communication tool. The Motioneye software is compatible with several Raspbian versions such as the newest versions of Buster and Stretch. Lastly, we considered the possibility of writing our own code from scratch to be able to communicate data wirelessly. This option allows for more customization of the software's capabilities. However, it takes a long time to develop. For example, we would have to create code for features like taking photos at preset intervals for a specific period of time, sending photos and sensor data to a cloud server and overlaying this information on the captured images, allowing the user to control and monitor the camera view of the system in real time as well as compile all the images of an experiment into a time lapse video and extract valuable data from them (tables, graphs and more).

In order to compare these options, the team planned to compare the performance of MotionEye OS with the performance of MotionEye software on Buster OS. When attempting to download the MotionEye OS on the Raspberry Pi, it was discovered that the OS was incompatible with the Worcester Polytechnic Institute wireless service and could not be downloaded or started while the Pi was connected to the Institute's wireless network. The team worked with the Institute's IT and Network Operations department to find the root of the issue for several days. It was found that because of how the University's Wireless network works as well as how it is secured, the motioneye configuration timeout was set to occur before it had received all the information for the network to authenticate the device. WPI uses WPA-Enterprise to secure the wireless Wi-Fi network, which requires each user to present log in credentials in order to authenticate and allow the user to connect to the network. Usually, most universities, companies, and institutions have a network with extra levels of security that home networks do not have. Therefore, it is expected that this type of issue will not happen on a regular home network due to a less structured network security.

4.3.3 Image Gathering Source

The next source of alternative considerations in our design was the source of image gathering. Ideally, one image gathering source would be capable of being used for both the cell and *C. elegans* cultures.

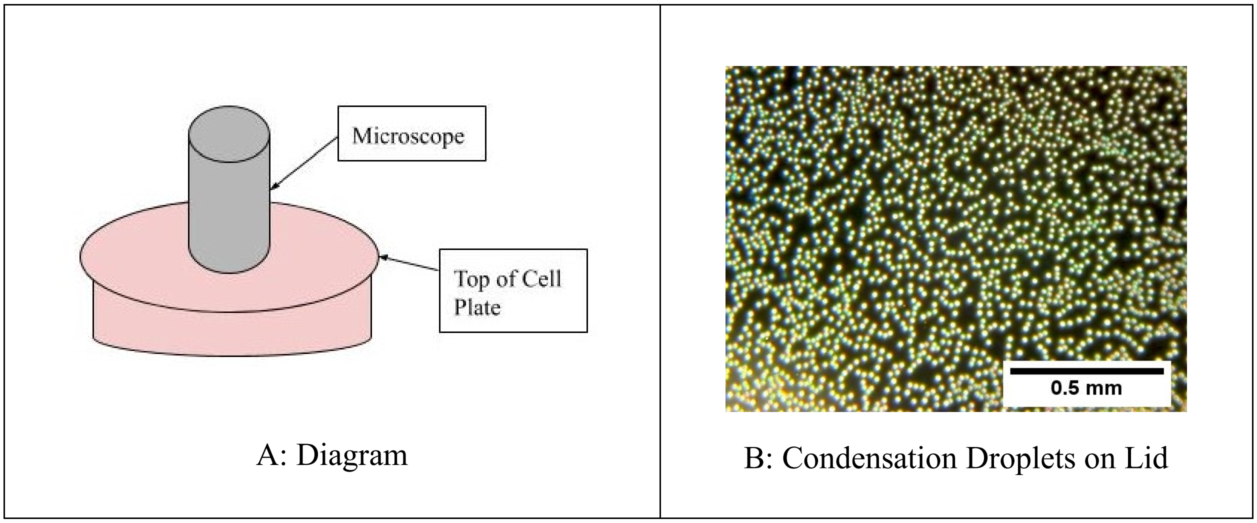
4.3.3.1 USB Microscopes

We investigated the use of a USB microscope for viewing cells in the lab. We tested two different USB microscopes to determine which one would work better. The specs of these two microscopes are included in **Table 8** below.

Specifications	**USB Microscopes**	
	Microscope 1 [37] 	Microscope 2 [38]
Purchase Source	Adafruit	Monoprice
Manufacturer	MicroView	Monoprice
Model Number	Product ID: 636	Model 11613
Max Magnification	220x	250x
Size	110 mm (L) x 33 mm (D)	80 mm (L) x 30 mm (D)
Image Sensor	2MP	2MP
Cost	$80	$40

Table 8: USB Microscope Specifications

The first test run on USB Microscope 1 was conducted by plugging the microscope into one of our computers and turning it on to try and view a sample of 3T3 cells in culture. When we attempted to view the cells from above with the microscope, we found that the focal length of the microscope was not long enough to be able to view the cells and focus on them as the field of view was only reaching the depth of the culture plate lid. This is demonstrated in **Figure 24** below which shows condensation bubbles on the underside of the culture plate lid and a diagram of the setup used to capture this image.



A: Diagram

B: Condensation Droplets on Lid

Figure 24: First USB Microscope Image Attempt

Our next move was to attempt to image a hemocytometer, a small device used for cell counting, in order to determine if the microscopes were capable of seeing the small sizes required for viewing cells. The success of this test is demonstrated in **Figures 25 and 26** below where we were able to view and capture an image of the smallest subset of squares on a hemocytometer. These squares measure an area of 0.0025 mm^2 or volume of 0.25 nL. The entire image in **Figure 25** represents a field of view of approximately 2 mm x 2 mm and in **Figure 25** a field of view of approximately 1 mm x 1.5 mm. The microscope which captured **Figure 26** had a superior image quality of the two microscopes tested when viewed at the default pixel resolution of each microscope. This microscope was therefore chosen to continue with the preliminary USB microscope tests, however both microscopes were used for versions of the final design testing.

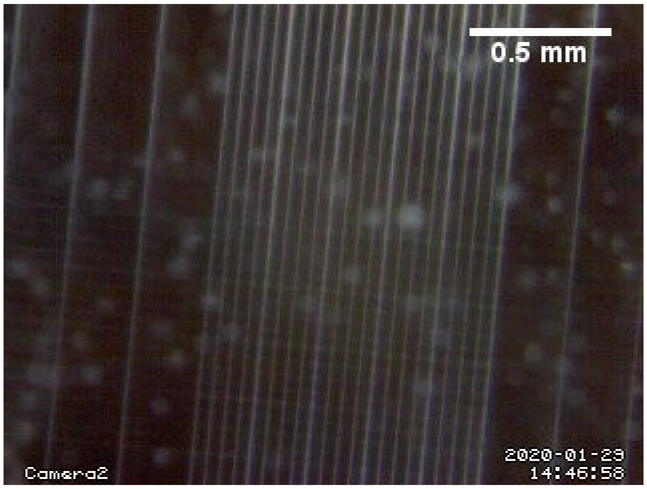


Figure 25: Hemocytometer Imaged with USB microscope 1

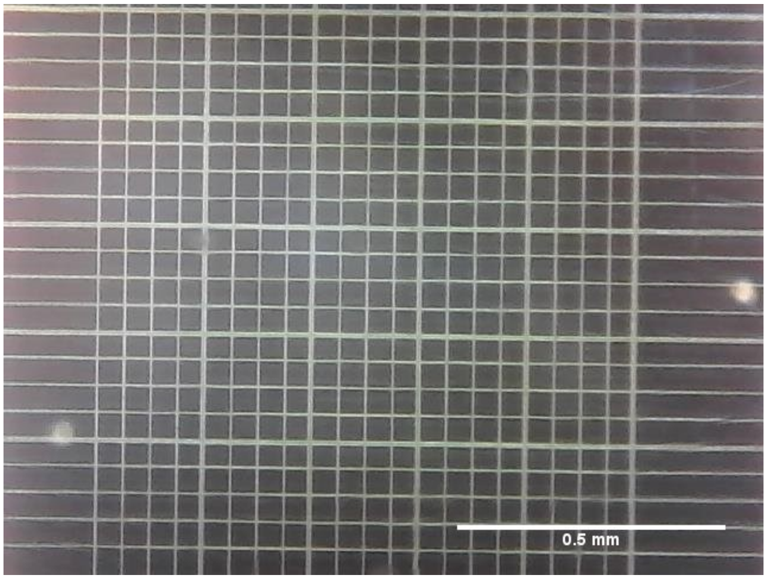
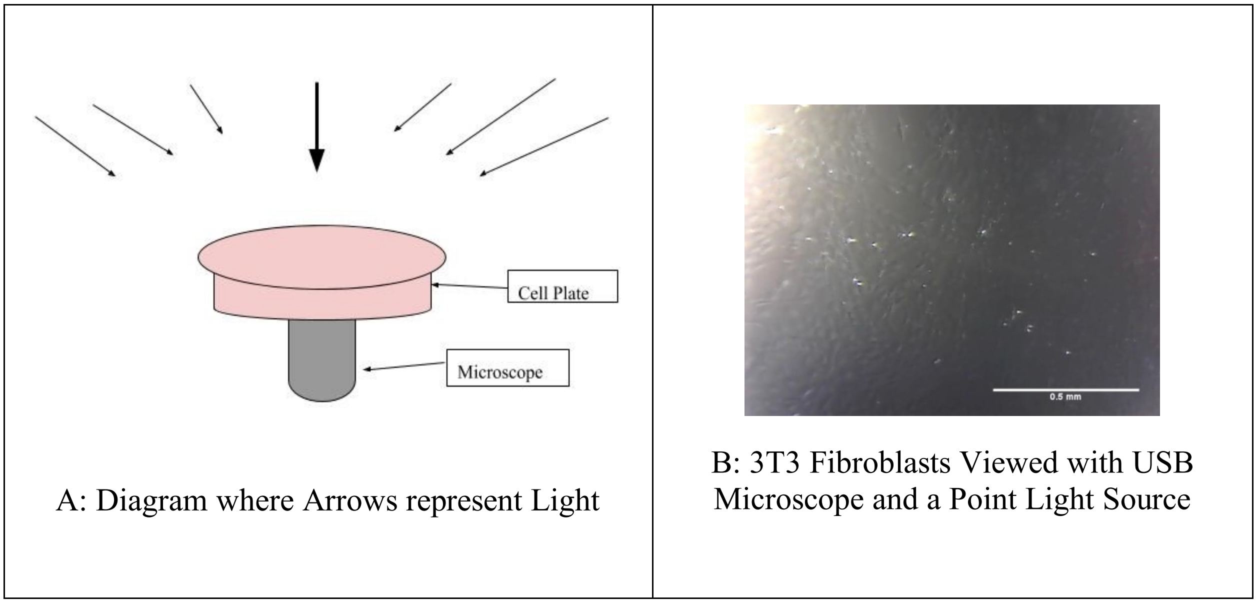


Figure 26: Hemocytometer Imaged with USB microscope 2

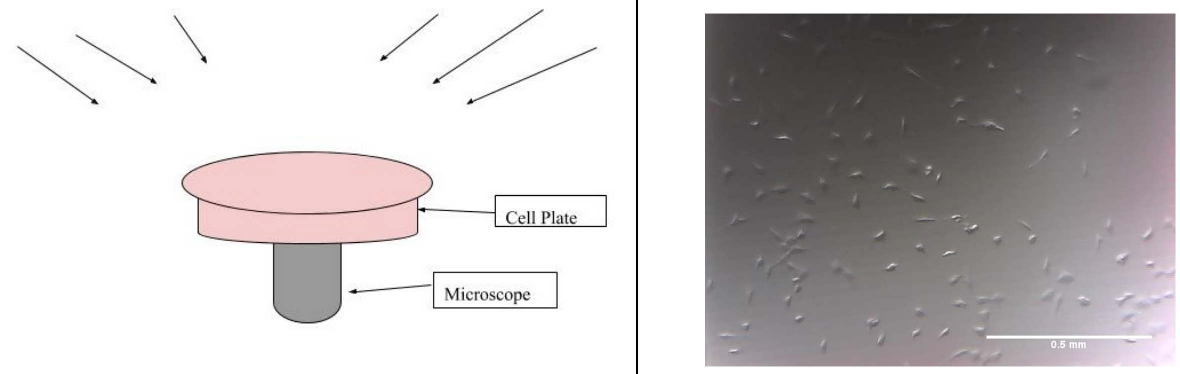
Once we knew that the microscope could image at a high enough magnification to view particles the size of cells, we attempted to view the cells from beneath the plate in order to put them closer to the image sensor. In doing this, we were able to view and focus on attached 3T3 cells in culture with the microscope. These images were taken at the full magnification the microscope offers which is 250x magnification. Examples of this are shown in **Figures 27 and 28** below. **Figure 27** was captured with a direct point light source above the plate where **Figure 28** used only ambient room light.



A: Diagram where Arrows represent Light

B: 3T3 Fibroblasts Viewed with USB Microscope and a Point Light Source

Figure 27: Dark Image of cell plate



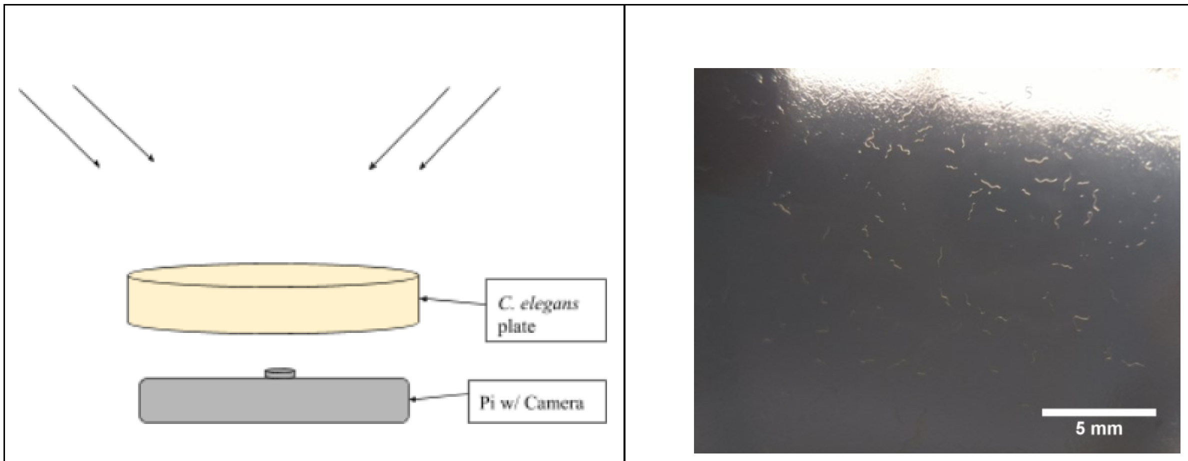
A: Diagram where Arrows represent Light

B:3T3 Fibroblasts Viewed with USB Microscope

Figure 28: Clear Image of cell plate

4.3.3.2 Raspberry Pi Camera

As an alternative to the USB microscopes, we also investigated the use of an 8MP Raspberry Pi V2.1 camera. The tests run with the Raspberry Pi camera were run on a sample of *C. elegans* in culture as the camera alone does not have the magnification necessary to view cells. The plate was imaged from below at a distance of half an inch. These nematodes are visible in **Figure 29** below. The initial images are rather dark as they were captured with only ambient room lighting which does not create the necessary contrast to properly view the nematodes.



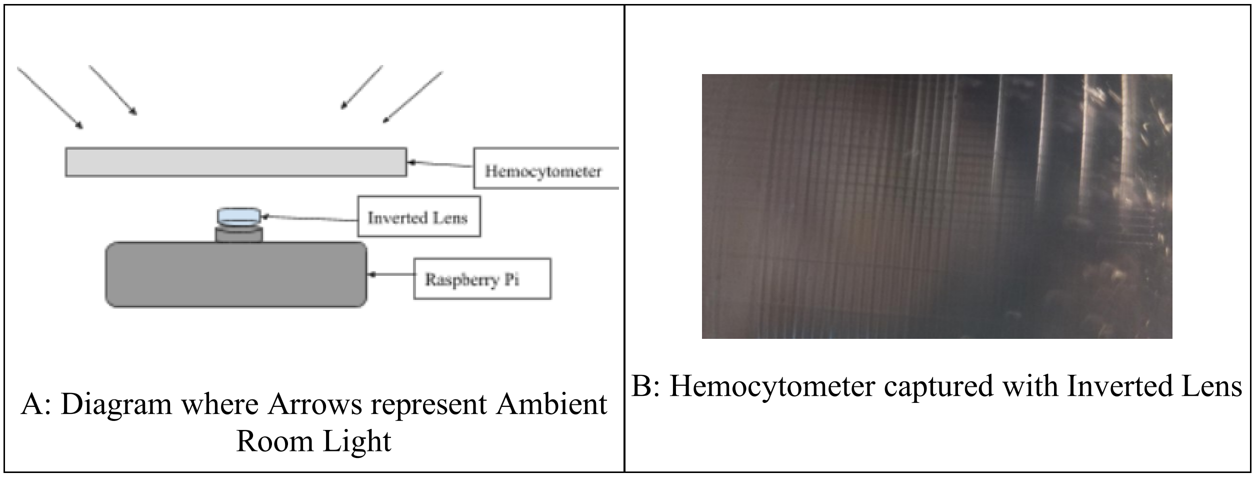
A: Diagram where Arrows represent Light

B: *C. elegans* on an Agar Plate Imaged with Pi Camera

Figure 29: Nematodes Imaged with Raspberry Pi Camera

4.3.3.3 Raspberry Pi Camera with Inverted Lens or Objective

Since the Raspberry Pi V2.1 camera alone cannot reach a magnification high enough to visualize cells, we also looked at the use of a Raspberry Pi V2.1 camera with a second Raspberry Pi camera lens inverted in front of it. This idea was based on previous research that showed that inverting an identical version of the lens will give the image the resolution and magnification of the original image sensor [39]. These images have what appears to be a similar magnification to the USB microscopes (roughly 220-250x) but are shaky due to the team members having to hold the second lens while capturing the image. Multiple attempts were made to use the inverted lens, and even with more stable positioning, the images were of a lower quality than those with the microscopes. The image in **Figure 30** appears to represent approximately 1.5 mm x 2 mm overall. The smallest squares visible possess the same dimensions as those in Figures 25 and 26.



A: Diagram where Arrows represent Ambient Room Light

B: Hemocytometer captured with Inverted Lens

Figure 30: Image of Hemocytometer taken with Inverted Lens on Raspberry Pi Camera

We additionally considered using an objective lens in place of the inverted lens to achieve the magnification necessary to visualize cells [22]. We attempted to test this with the use of borrowed objective lenses of varying magnifications but were not able to line them up properly to be able to capture an image. Due to the difficulty of using and integrating this technique, the idea was therefore eliminated as an alternative design option but remains as a potential avenue of future work.

4.3.3.4 Autofocus Raspberry Pi Camera

In addition to the Raspberry Pi V2.1 camera, we also looked at a 5MP Raspberry Pi Autofocus camera. This camera would allow the user to change the camera focus from afar rather than having to manually change it. In theory, this would be very useful to this device in regard to limiting the amount of times the incubator needs to be opened and the sample disturbed. However, this camera was difficult to use and would require the same inverted lens set up in order to reach a magnification high enough for cells. This camera would likely be useful for work with macroscopic studies. While the team was able to get the autofocus camera to function eventually, we decided to move it to future work as when we needed to move into final design testing it did not work yet.

Overall, the functional image gathering sources found from early tests were both of the USB microscopes, and the Raspberry Pi V2.1 camera (for macroscopic work).

4.3.4 Light Source

In order to properly light the samples while inside an incubator, the device has to have a built-in light source opposite of the image gathering source. As mentioned in Section 4.1.1 and Section 4.2, we determined that the best method of microscopy for our device would be darkfield, meaning the light source would have to illuminate the area around the sample. This need comes from our background research of existing lab devices similar to ours in Section 2.4.2, where we were inspired to use a programmable LED array [22]. Additionally, we took into consideration our advisor's lab's work, where they imaged *C. elegans* with a rectangular flat light and a blocking filter to diffuse the light [26]. Therefore, the options we considered using were an LED array, with a varying number of LEDs, and a flat light. **Table 9** below shows the different light sources, along with some relevant specifications.

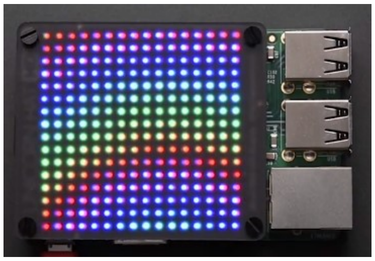
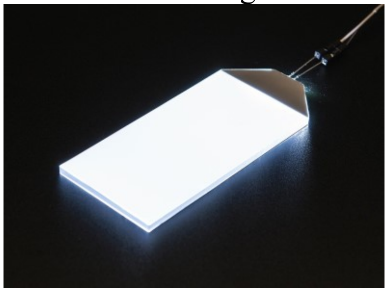
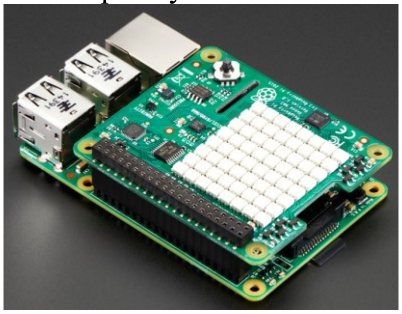
Light Source	Size	Connection	Price
8x8 Pimoroni UnicornHat LED Array 	65mm x 56mm x 6.5mm	GPIO	$29.95
16x16 Pimoroni UnicornHat LED Array 	65.0mm x 56.0mm x 7.9mm	GPIO	$34.95
White LED Backlight Module 	45mm x 86mm	GPIO	$2.95
Raspberry Pi SenseHat 	65.1mm x 56.6mm x 13.9mm (whole structure, not just light source)	GPIO	$39.95

Table 9: Light Source Options [40, 41, 42, 43]

All of the lighting options above connect to the GPIO header of the Raspberry Pi. The White LED Backlight Module needs to be plugged into two GPIO pins to operate, one pin being ground and the other being power. Also, the white LED backlight module can be plugged into a clock pin to be configured to turn on when a picture is taken, allowing the user to save power. A tutorial for this is in the Step-by-Step Assembly Guide in Appendix A. Overall, the white LED backlight module allows the user for the most customization, as it leaves the maximum number of GPIO pins unused. The 8x8 LED uses three pins, one for data, one for ground, and one for power. However, it sits on all 40 pins of the Raspberry Pi's header, and therefore, must be wired differently than the manufacturer's recommendation if use of the other GPIO pins is required. The method for appropriate wire configuration is also shown in the Step-by-Step Guide in Appendix A. The 16x16 LED Array uses the most GPIO pins, proving the least customization of the GPIO pins for other uses. However, the smaller LEDs within the same size as the 8x8 LED Array allows for further customization of lighting direction and configuration. Using a programmable LED array also allows the user to potentially try the DPC method described in Section 4.2, as the lights' colors can be controlled, as well as timing.

The Pi Sense Hat also has a programmable LED array along with temperature, humidity, and pressure capabilities. These extra capabilities are beneficial as it allows for further compactness of the device. However, because the LED array does not take up the whole structure like the 8x8 and 16x16 UnicornHat does, there is a concern that this LED array would not light the whole sample.

4.3.5 Housings

As part of our design alternatives, we designed multiple housing ideas to hold the components of the device. Each different imaging technique could require a different housing set up to function. The initial sketches of these housings are shown below in **Figure 31**.

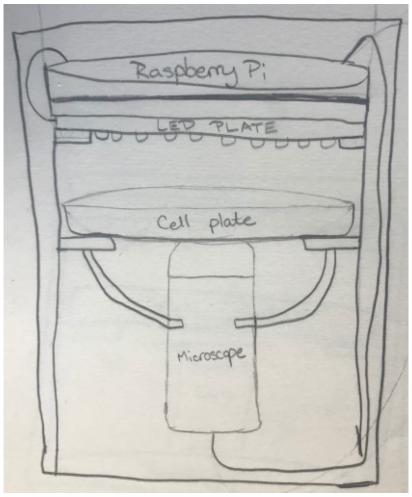
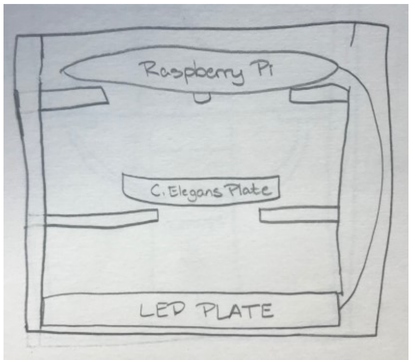
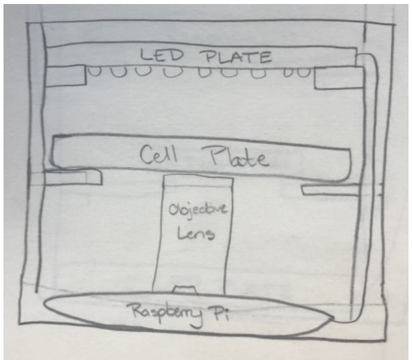
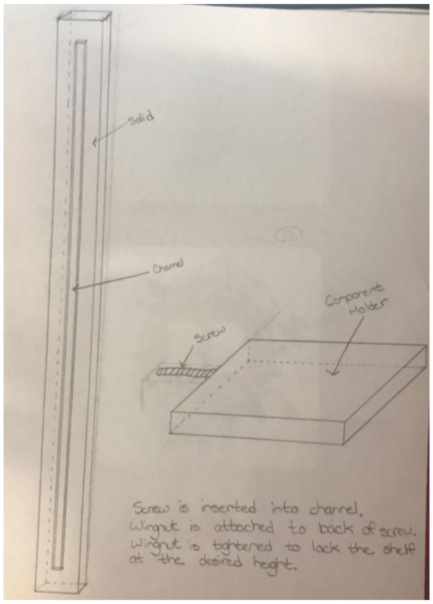
		
A: Housing for USB microscope	B: Housing for *C. elegans* with Raspberry Pi camera	C: Housing for Raspberry Pi with Objective lens

Figure 31: Alternative Housing Designs

After further consideration of these housing designs, another alternative was designed which would be more versatile than the previous design options. This housing option, shown below in **Figure 32**, consists of a main pole or rod with a channel cut into it, and an assortment of "shelves" with screw attachments that can be inserted through the channel of the rod. Wing nuts would be used to secure the shelf in place once the screw is through the channel.



Screw is inserted into channel.
Wingnut is attached to back of screw.
Wingnut is tightened to lock the shelf at the desired height.

Figure 32: Isometric Sketch of Hollow Rod Housing Design

4.3.6 Temperature and Humidity Sensor

We considered two types of temperature sensors for our device, the DS18B20 and the AM2302/DHT22. **Table 10** below compares the sensors' specifications.

Sensor	GPIO Pins	Voltage Rail	External Components	Accuracy	Size
DS18B20	3: Data, power, and ground	3.3 or 5 V	4.7 kOhm resistor	-10 C to 85 C: ±0.5 C -55 C to -10 C and 85 C to 125 C: ±2 C	Cable: 4 mm diameter 91 mm long Sensor: 6 mm diameter 30 mm long
AM2302/DHT22	3 : Data, power, and ground	3.3 or 5 V	4.7 kOhm/10 kOhm resistor	-40 C to 80 C: ±0.5 C	15.3 mm x 7.8 mm x 25.3 mm

Table 10: Temperature Sensors' Specifications [44, 45]

The data pin can be configured for almost any pin on the Raspberry Pi header for both sensors. There are tutorials online for the software setup of both sensors. However, the setup for the AM2302/DHT22 is simpler as it only requires libraries and repositories to be downloaded and installed, where the code to receive temperature readings is within the downloaded software. This differs from the DS18B20, as software setup for this sensor requires the user to write their own code; although there are also many straightforward tutorials online. Also, the length of the DS18B20 was significantly more than that of the AM2302/DHT22 due to the fact that it has a cable component. Therefore, we were concerned this would impact the overall size of our device, as one of our design goals was to ensure our device was compact, as mentioned in Section 4.1.1.

Additionally, the DHT22/AM2302 has a humidity sensor, while the DS18B20 doesn't, meaning if the user chooses the DS18B20, they must find an alternate humidity sensor. However, the DS18B20 is waterproof, meaning its longevity in a humid incubator will be longer than that of the DHT22/AM2302.

We found in research that the family of DHT22/AM2302 sensors was the only humidity sensor commercially available to the average consumer with Raspberry Pi compatibility. Therefore, no alternatives for this sensor were found.

4.3.7 pH Sensing

After comparing methods of pH sensing in Section 4.2, it was determined that the best method was to analyze RGB values via an image processing program. The options considered for this were the MATLAB Image Processing Toolbox, and Open Computer Vision (OpenCV) supported by python. The documentation for MATLAB's image processing toolbox is very thorough with each iteration of MATLAB in comparison to OpenCV. There are multiple functions in MATLAB that allow the user to perfectly filter an image before it's analyzed, such as using a Gaussian Blur, filling holes, edge detection, edge segmentation, and eroding the edges of an image. However, OpenCV does not have direct functions for all of the indicated filtering processes, requiring the user to do more software development for the desired result. Also, when MATLAB and Image Processing Toolbox is downloaded for the Raspberry Pi, all dependent packages are automatically downloaded by the install, whereas for OpenCV, the dependent packages must be downloaded manually, making the installation process longer for the user. Also, the install instructions for MATLAB are very clearly documented, whereas the instructions for the OpenCV install on their website is not straightforward for first-time users, meaning outside tutorials need to be used, and many are incomplete and therefore lead to a failure in the install.

OpenCV is advantageous over MATLAB for this project though, in that it doesn't require a license, meaning it's no additional cost for the user for installation. Additionally, software maintenance for a MATLAB license expires after a year, meaning it would have to be renewed every year. For educational purposes, if the user's school does not provide campus-wide licenses, a permanent license that allows for unlimited renewal is $500. If the user is in primary or secondary school, the license price $100 [46].

4.3.8 Data Logging

Assuming the temperature sensor used is the AM2302/DHT22, we had a few options for data logging. The AM2302 library from Adafruit came with a Python script that could be configured to send data to a google spreadsheet in a .csv format. However, given experiments are recording temperature, humidity, and pH every minute, and sometimes more often depending on the application, this would result in at least 7200 data points, making it difficult for the user to parse through and determine which data they would like to visually represent.

Another option we considered for data logging was using ThingSpeak. ThingSpeak is an online interface supported by MathWorks that allows the user to send data to their channel to make it viewable to the public or custom members if desired by the user. They also allow the user to customize the resulting graph in several ways, such as limiting the number of data points to show, which days of the data to show, adjust the minimum and maximum values on the y axis, and modify axis and chart titles. The benefit of this is it automatically graphs the data for the user, allowing it to be visible in real time while data is still being taken. However, since this is a program supported by MathWorks, it requires a MATLAB license to unlock all features, with its high costs, as mentioned in Section 4.3.8 above.

4.4 Final Design Selection

4.4.1 Raspberry Pi

As described in 4.3.1, the Raspberry Pi Zero WH and 3B+ specifications were compared. For a list of the specifications of the two microprocessors refer to Section 4.3.1. It was found that the Zero WH could accomplish the tasks given it has 512 MB of RAM compared to the 3B+ having 1 GB of RAM. However, the install process took much longer on the Zero WH due to having less RAM. The Zero WH is also smaller and $20 cheaper than the 3B+. Also, the Raspberry Pi Zero WH can be reasonably powered with a commercially available battery pack, whereas the Raspberry Pi 3B+ would need to be plugged in outside the incubator for long time lapse experiments.

Despite consuming more power and being bigger than the Pi Zero WH, it was determined that the Raspberry Pi 3 B+ was the correct choice for the device based on its RAM, which allowed for ease of use during setup. Additionally, the fact that the 3B+ was more expensive and larger than the Zero WH was not deemed important, since it was still within our budget for a microprocessor, as outlined in Section 4.2, and also its size was still within our needs described in Section 4.1.1.

4.4.2 Remote Monitoring

Each of the alternative design options for wireless communication and automated features was examined in the following evaluation table and compared to determine the best choice for the final design. In the table below, “not customizable” means that in order to change certain aspects of the user interface, additional python or shell scripts have to be created and run. In other words, the base code of the program itself is not easily customizable.

Wireless Communication	Pros	Cons
MotionEye OS	• Easy to use • Open source • Built in user interface	• Not compatible with WPI wifi • not customizable • Pi OS is devoted only to wirelessly communicating data collection
MotionEye Software	• Easy to use • Open source • Built in user interface • Compatible with downloaded some versions of Raspbian OS (flexible)	• sensor data reporting more complicated
Custom Software (Our Own)	• customizable • could build in sensor data reporting feature	• time consuming to create due to the number of features to develop from scratch • need to build user interface

Table 11: Remote Monitoring Evaluation Table

Based on the findings in **Table 11** it was decided to move forward with the MotionEye software downloaded with Buster OS as it was the most accessible, and easiest to use of the options.

4.4.3 Image Gathering Source

Based on the photos taken with each image gathering device which can be seen in the alternative design section, a matrix of pros and cons of each image gathering device was made. This can be seen below in **Table 12**. Each of the USB microscopes have the con of the potential for focus drift. This means that over time it is possible that the focus of the microscope would change slightly causing the sample to drift out of focus in the images. The autofocus camera has no focus drift as a pro because while the focus still has the potential to change, the nature of the autofocus camera allows it to be refocused remotely without disturbing the sample.

Image Gathering Source	Pros	Cons
USB Microscope 1 (Microview)	• Easy to use • Inexpensive • Clear image of hemocytometer • Easy to create a housing for	• Potential for focus drift • Large size • short focal length • not customizable • poorer focus on hemocytometer
USB Microscope 2 (Monoprice)	• Easy to use • Inexpensive • Clear image of hemocytometer • Clear image of cells • Easy to create a housing for	• Potential for focus drift • Large size • short focal length • not customizable
Raspberry Pi Camera V2.1	• Easy to use • Inexpensive • Clear image of *C. elegans* • built into raspberry pi • easier to use GPIO with	• low magnification • short focal length • difficult to use with a Pi based light source
Raspberry Pi Camera V2.1 with inverted lens	• Inexpensive • easier to use GPIO with	• low image quality • short focal length • difficult to use with a Pi based light source • difficult to create housing for
5 MP Autofocus Raspberry Pi Camera	• no focus drift • easier to use GPIO with • adjustable focal length	• low image quality • difficult to use with a Pi based light source • difficult to create housing for

Table 12: Image Gathering Source Evaluation Table

Based on this evaluation table it was decided that the best option to move forward with was USB microscope 2. While the autofocus camera appeared to be a strong option, in reality it was difficult to use, and the team could not get it to work before the final design testing phase of the project. This option is discussed further in the recommendations chapter of this paper.

4.4.4 Light Source

The alternative light source options were theoretically evaluated and the pros and cons of each were compiled in **Table 13** below.

Light Source Option	Pros	Cons
Pimoroni UnicornHat 8x8 LED Array	• easy to use • powered by Pi • controlled through code on Pi • inexpensive • accessible code	• small amount of LEDs • has to be attached to Pi • creates white light out of RGB • no additional sensors
pi SenseHat LED Array	• allows use of other sensors on same board • powered by Pi • controlled through code on Pi • inexpensive	• small array • smaller LEDs • has to be attached to Pi • creates white light out of RGB • more expensive
Pimoroni UnicornHat HD 16x16 LED Array	• larger array • easy to use • powered by Pi • controlled through code on Pi • accessible code	• has to be attached to Pi • creates white light out of RGB • more expensive • no additional sensors
White LED Backlight Module	• white light • powered by Pi • inexpensive • no additional software required	• no control over individual LEDs • no control over color of light • no additional sensors

Table 13: Light Source Evaluation Table

In addition to the features listed above, **Table 9** in Section 4.3.4 describes several other features, such as wire connectivity type, price and size. Based on size, the Pi Sense Hat was eliminated, as the LEDs were a section of the structure, as opposed to being the whole structure like the UnicornHats. Because of this, we were concerned the light would not cover the whole area of the sample. The White Backlight LED Module was also taken out of consideration, as the individual LEDs cannot be programmed to change the direction or color of light. However, the direction of light could be modified by diffusing the light via a privacy film [26]. This leaves the 8x8 and 16x16 UnicornHat LED array, where we ultimately chose the 8x8 LED array due to price. Despite being cheaper, and therefore having less and larger LEDs, we thought the 8x8 LED array could be easily programmed to test out lighting configurations for different types of microscopy, which are further explained in Section 5.1.

4.4.5 Housing

Once the design alternatives for the device's housing were created it was necessary to complete an evaluation to determine which design would be selected as the final device housing. **Table 14** shows the pros and cons of each design alternative.

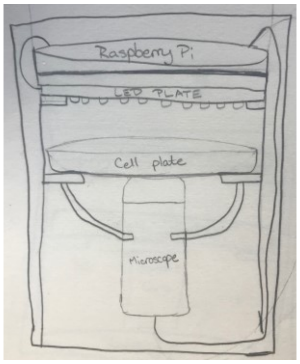
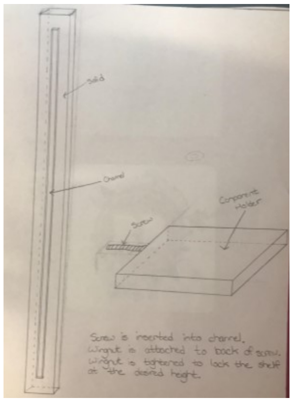
Design	Pros	Cons
Rigid structure, cubic shape with built in shelves  	• Easy shape to design • Stable structure • Easily reproducible • Can be made from cheap/accessible materials	• Set distance between components • Only functional for one application per structure • Requires more material than other design • Takes up more space
Hollow stand with removable shelves  Screw is inserted into channel. Wingnut is attached to back of screw. Wingnut is tightened to lock the shelf at the desired height.	• Easy shape to design • Stable structure • Easily reproducible • Can be made from cheap/accessible materials • Easily customizable • Functional for more varied applications • Easy to construct and alter	• More pieces to potentially misplace • More complicated to assemble than the single piece design • Requires use of metal screws and nuts

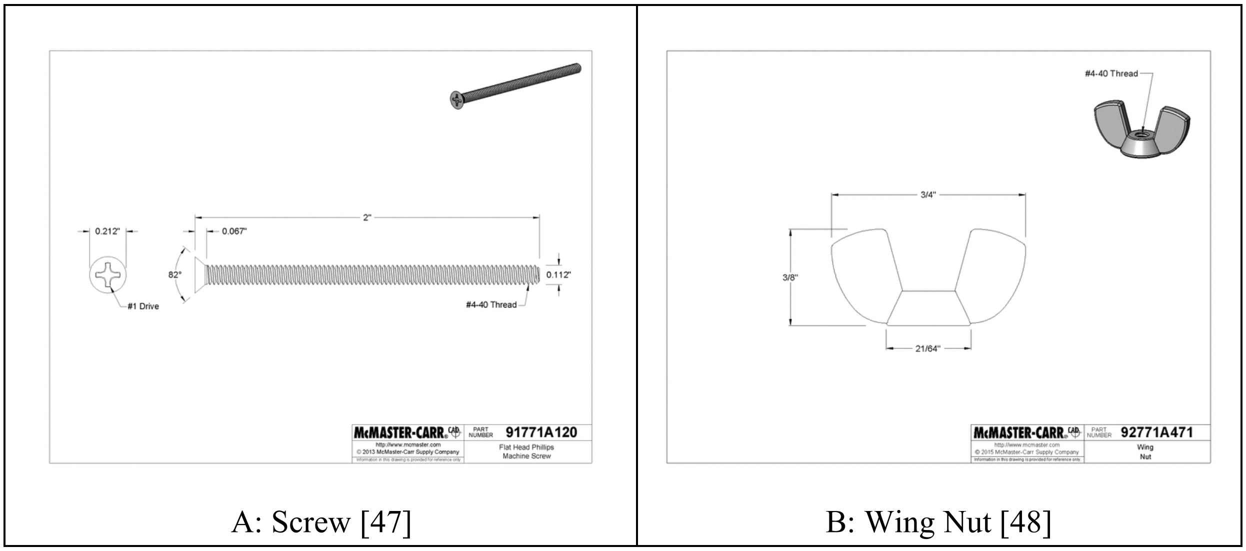
Table 14: Housing Design Evaluation Table

These pros and cons led to the selection of the hollow rod housing design. After selection of this design, it was realized that by adding a second nut to the design mobility could be achieved in the x direction as well as the z direction. Additionally, in moving forward with this design it was necessary to evaluate whether the screw element should be built into the shelf, or a separate piece. The pros and cons of each option were evaluated as shown in **Table 15** below.

Design Element	Pros	Cons
Built in screw with separate nut	• Screw can't be misplaced • Easier to assemble • Stronger attachment to shelf	• Plastic screw element is easier to break • Can snap off shelf if not stored properly • Difficult to design screw for correct functionality • More difficult to properly size
Metal screw and nut assembly (not built into shelf)	• Easier to design (don't have to CAD the screw) • Easier to fabricate • Metal screw is stronger • Assures correct screw size and threading for standard nuts	• Screw can be misplaced • Need to specify length and size of screw in handbook

Table 15: Screw Evaluation Table

Through this evaluation, it was deemed better to use a separate metal screw and nut assembly rather than a built-in screw. 3D models of the housing components were made using a Computer-aided Design (CAD) software called SolidWorks. In order to design these models, measurements were taken of each component to ensure an accurate fit. Additionally, different options for screws were compared. It was found that the longest fully threaded screws available were 2 inches long. These selected screws were flat-head, phillips, stainless steel screws with a small head diameter and large number of threads. These features allow the screws to easily and unobtrusively fit into the component shelves and be tightened to exactly the right length as needed. Wing nuts which would fit with the screw size were also chosen at this time. Brass wing nuts were chosen because they are corrosion resistant in wet environments. Schematics of the chosen screws and wing nuts are in **Figure 33** from the McMaster-Carr website [47, 48].

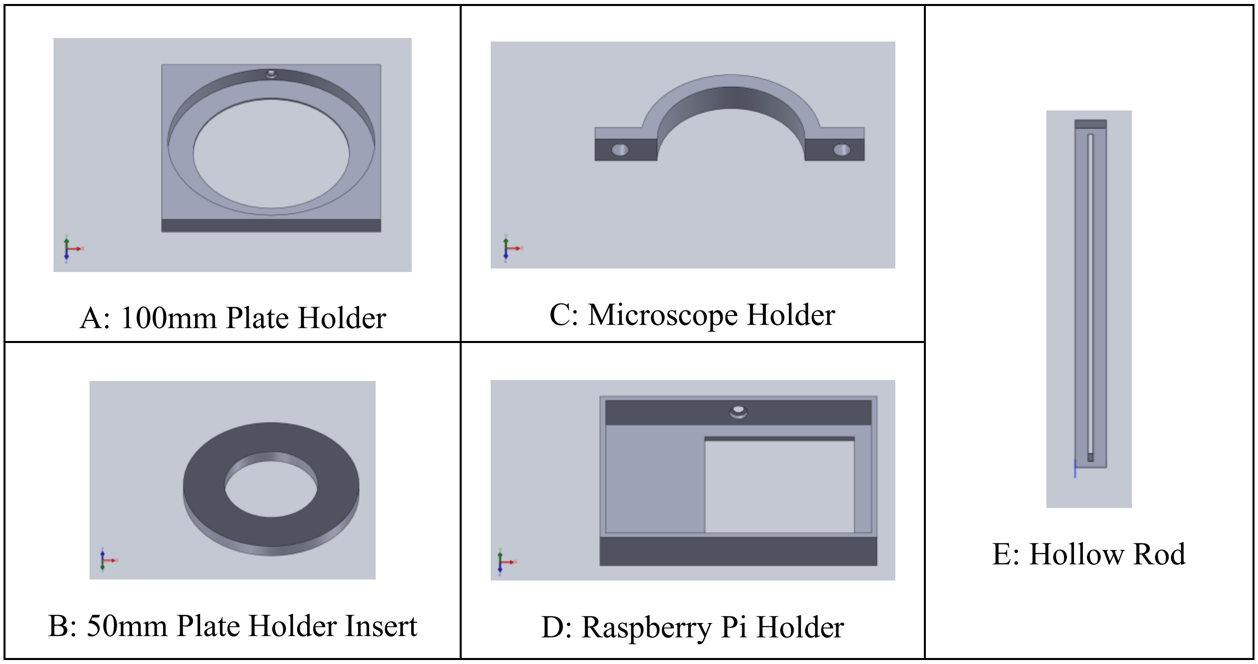

0.212"
0.067"
2"
82°
0.112"
#1 Drive
#4-40 Thread
McMASTER-CARR
PART NUMBER
91771A120
http://www.mcmaster.com
© 2013 McMaster-Carr Supply Company
Information in this drawing is provided for reference only.
Flat Head Phillips Machine Screw
#4-40 Thread
3/4"
3/8"
21/64"
McMASTER-CARR
PART NUMBER
92771A471
http://www.mcmaster.com
© 2015 McMaster-Carr Supply Company
Information in this drawing is provided for reference only.
Wing Nut

A: Screw [47]

B: Wing Nut [48]

Figure 33: Diagrams of Housing Screws and Wing Nuts

After the screws were chosen and the measurements of components were taken, the initial CAD designs were created. For the plate holder components, it was deemed beneficial to make one plate holder for 100mm plates and an additional converter which would rest inside that component making the opening smaller. The first iteration of each component can be seen below in **Figure 34.**



A: 100mm Plate Holder

C: Microscope Holder

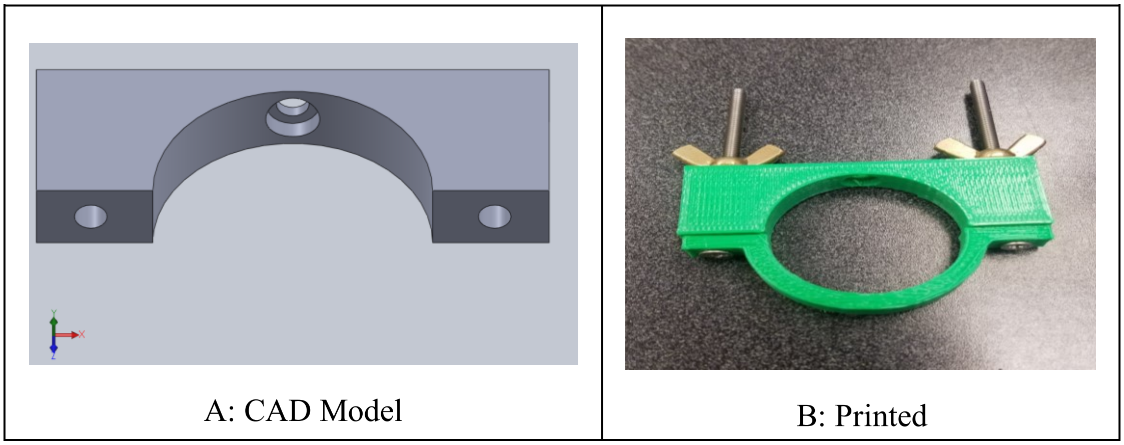
E: Hollow Rod

B: 50mm Plate Holder Insert

D: Raspberry Pi Holder

Figure 34: First Iteration CAD Models

After these initial CAD designs were made, an expert on Worcester Polytechnic Institute 3D printing was consulted. Unfortunately, the printers on campus were not capable of printing the rod as it was currently designed due to its height of roughly 31 cm. In order to work around this the design was cut in half and pegs were created so that the two halves of the rod could snap together after printing to reach the full rod height. After this design alteration, all of the components were printed. After the initial print of each component it was found that the 100mm plate holder, and 50 mm converter initial designs were adequate, and became the final designs for those components. The rest of the component holders needed continued prototyping. The microscope holders fit around the microscope but did not have a way to attach to the rod. Subsequently, a new design was created which would allow it to attach to the rod. The final microscope holder design entailed one print of the initial design attached to one of the secondary designs through attached screws. Both the CAD model and printed design can be seen below in **Figure 35**.



A: CAD Model

B: Printed

Figure 35: Final Microscope Holder

The initial design for the Raspberry Pi holder had a couple of flaws. The wall of the holder was not thick enough to accommodate the hole for the screw with enough strength. Additionally, the opening for the LED array was not positioned correctly and overlapped partly with a row of the array. The holder was redesigned to fix these issues, and the final CAD design is shown below in **Figure 36**.

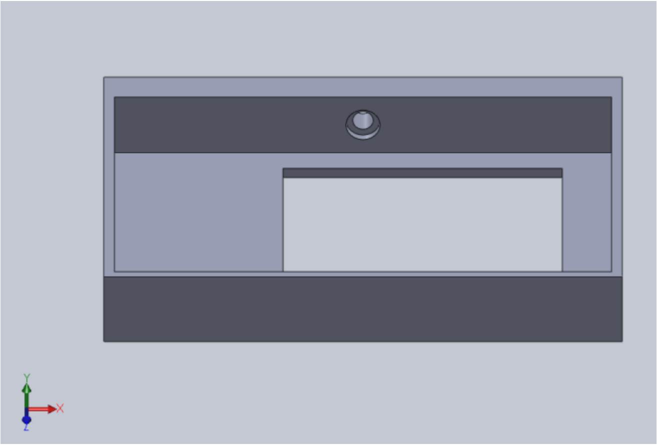
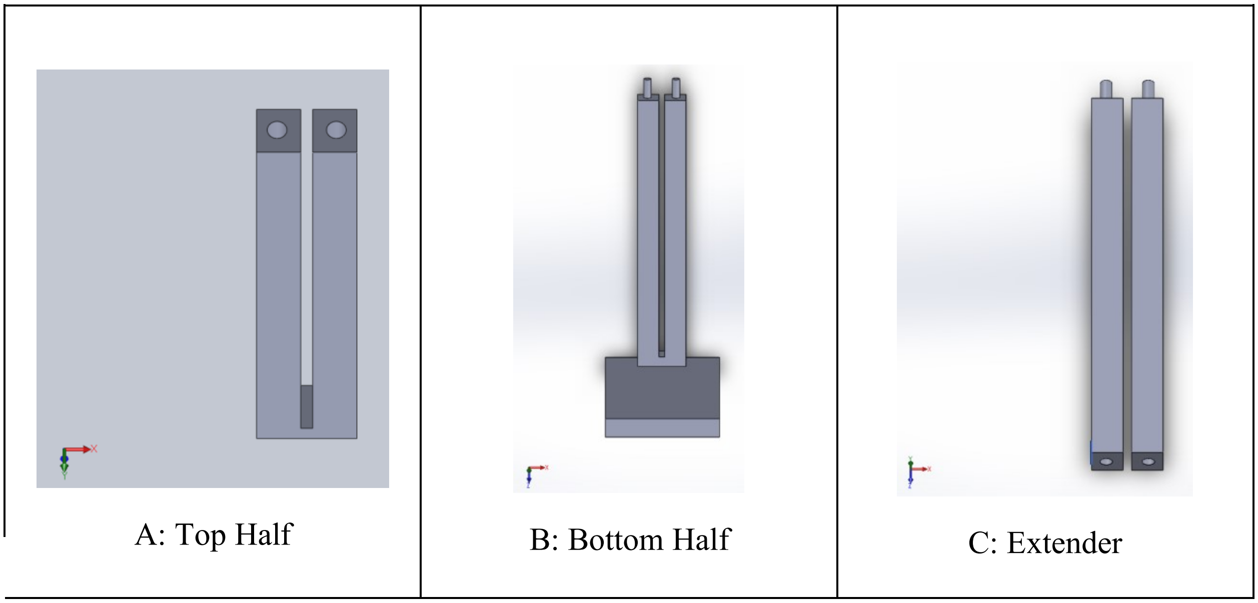


Figure 36: Final Raspberry Pi Holder

The initial rod design proved to be too thick when printed to allow for the proper alignment of the component shelves. Additionally, the channel was too wide so that the wing nuts would also slide partly into the channel. It was also discovered that in order to maintain the balance of the housing, a larger base was needed to counterbalance the weight of the shelves. As part of this redesign, additional extenders were designed for the rod so that the height can be customized as needed. These design changes are reflected below in **Figure 37**.



A: Top Half

B: Bottom Half

C: Extender

Figure 37: Final Hollow Rod CAD Model

After finalizing these designs, the team also decided to design a plate holder for 6-well plates in order to run additional experiments with the device. The CAD design of this component can be seen below in **Figure 38**.

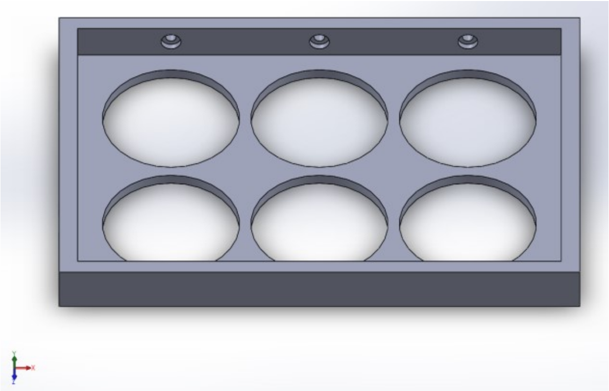


Figure 38: 6-Well Plate Holder

Once all of these components were designed, printed, and redesigned to come up with the final CAD models of each component the full housing was printed on a MakerBot 3D printer using PLA plastic in colors available in the lab. The printed components were assembled along with the screws, nuts, and additional components including the raspberry pi and microscope to create the final base device design (without added sensors). This printed design can be seen below in **Figure 39**.

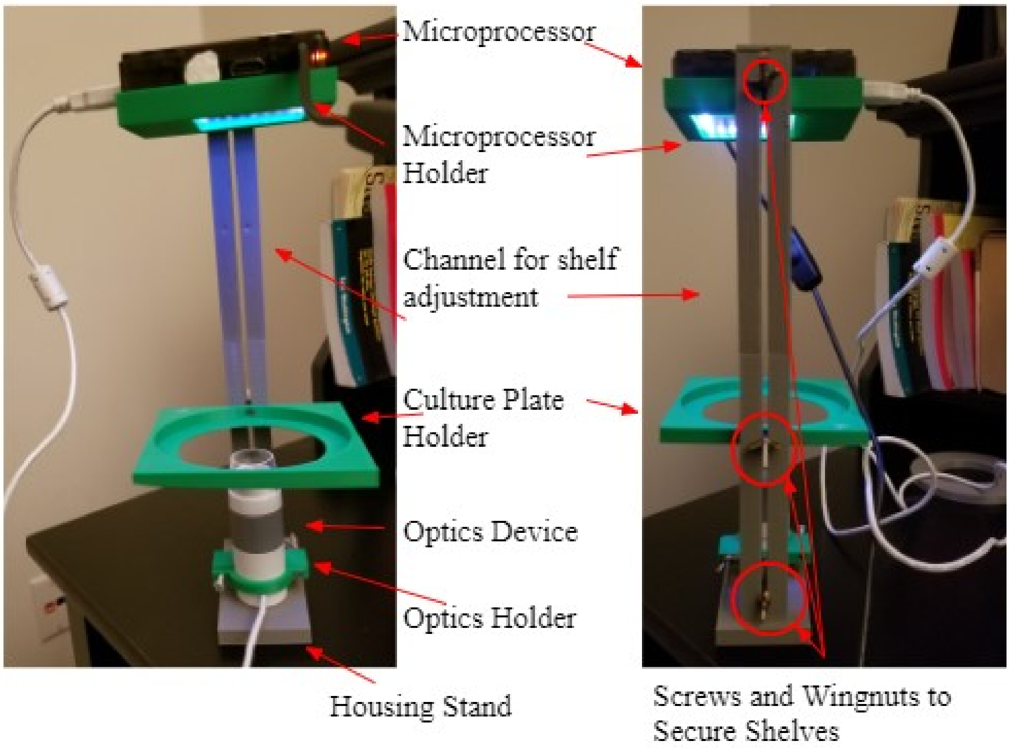
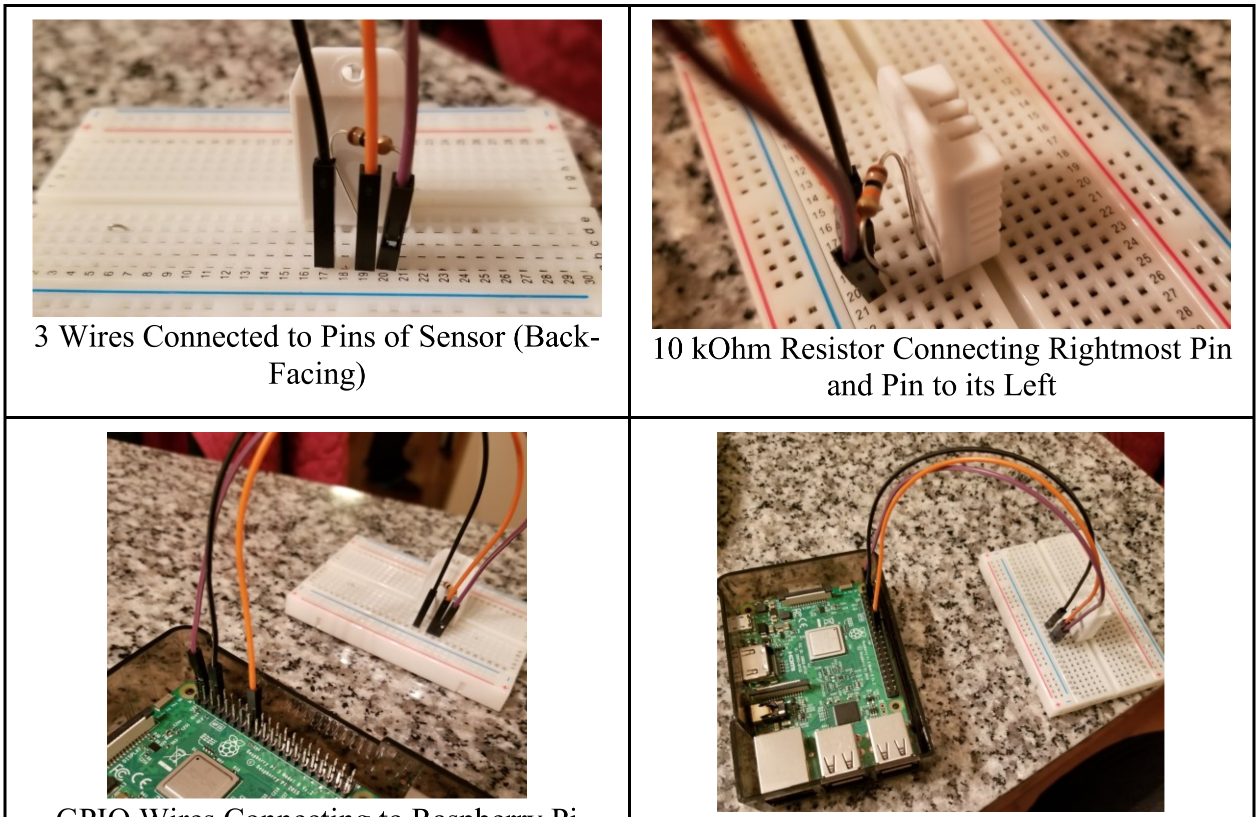


Figure 39: Final Printed Device Front and Back Views

4.4.6 Temperature and Humidity Sensor

The team chose to use the DHT22/AM2302 for its compact size, ease of use for the user, and additional humidity sensing feature, as mentioned in Section 4.3.6. It's accuracy was comparable to that of the DS18B20, as shown in **Table 10** in Section 4.3.6. A breadboard is required to set up this sensor to ensure it can sit upright within the chamber and read the ambient temperature. After the temperature sensor is inserted in the breadboard, a wire connecting the rightmost pin when facing the back of the sensor should be connected to the 5V rail of the Raspberry Pi, which is header pin two. A 10 kOHm/4.7 kOhm resistor then connects the rightmost pin of the temperature sensor and the pin to its left. The 2nd pin from the right of the sensor is connected via a wire to any data pin on the Raspberry Pi the user desires, as this can be configured within the temperature reading program. We connected our wire to GPIO pin 23, also known as header pin 16. The 3rd pin of the temperature sensor is not connected to anything. The 4th pin of the temperature sensor connects to a ground pin on the Raspberry Pi, which is on header pin 6. **Figure 40** below shows the configuration.



3 Wires Connected to Pins of Sensor (Back-Facing)

10 kOhm Resistor Connecting Rightmost Pin and Pin to its Left

GPIO Wires Connecting to Raspberry Pi GPIO Pins

Overall Configuration

Figure 40: Temperature Sensor Wire Configuration

After that, we installed the necessary packages and code repository, as that code allows the user to receive temperature and humidity readings from the sensor. In order to view the temperature and humidity readings from the sensor, we navigated to the corresponding directory where the code was stored and ran the temperature script in the Raspberry Pi terminal. After ensuring our sensor was reading values, we put our hand over the sensor to see if it would pick up the change in humidity and temperature caused by the change in the environment. We observed that the temperature and humidity did change significantly when a hand was placed over the sensor, meaning it was sensing changes in the environment.

Another feature we wanted to include, along with data logging of the temperature and humidity, is the temperature and humidity of the environment of the sample being overlaid onto the interface, allowing the user to visually track the changes in the environment during an experiment. This entailed us writing a shell script that ran our temperature and humidity reading script every time the overlay script was called. The overlay data script is run every minute via a cron job, which is a command sent to the Raspberry Pi every desired time interval to run a specific script. The fastest a script can update with a cron job is every minute, which is how often our temperature and humidity overlay updates. A full tutorial on how the user can accomplish this is in the Step-by-Step Assembly Guide in Appendix A.

4.4.7 pH Sensing

The method chosen to determine pH was to use the OpenCV library supported by python, as it was more cost efficient in comparison to using MATLAB. However, because the method of image processing was chosen in feasibility studies, pH can only be read and overlaid onto the interface when taking pictures of the sample, which would occur during a time lapse. For this reason, despite only needing a picture every 15 minutes in cell culture, a picture must be taken every minute to ensure the pH is being analyzed and updated every minute.

The python script imports the OpenCV library, Numpy, os and glob. Using the os and glob import, the program iterates through all the pictures in the specified folder to find the most recent picture. However, the limitation with this is that the time lapses are separated by folder according to date. Therefore, if there is a time lapse that spans several days, the file path will need to be updated every day. Further development can be done to program this or configure motioneye such that time lapses are not grouped by date.

The most recent picture is then imported into the script converted to an RGB format. This is just to ensure our pictures are in color, although they are expected to be taken in color anyways. After that, the RGB value of each pixel is taken by looping through all the pixels in the image using a nested for loop and analyzing the RGB value of each pixel, resulting in a 2D array. The elements of the outer array correspond to each pixel, and the elements of the inner array correspond to the B, G, and R value of the corresponding pixel. The values in this array are averaged using numpy.average along the vertical axis, resulting in a 3-element array where the elements are B, G, and R. If the elements were averaged along the horizontal axis, this would result in an array n number of pixels long, where each element is the averaged RGB value for that pixel. A correlation between pH and RGB was attempted in order for the system to import a picture and read the pH value based on its RGB value. An explanation of the experimental procedure to create this correlation is in Section 6.3 and the results and explanation are in Sections 5.2.2.1 and Section 7, respectively.

4.4.8 Data Logging

Ultimately, it was decided that the ThingSpeak automatic data logging website would be used to log data from the device sensors as it was relatively easy to create code to run continuously, and the website automatically creates graphs of the collected data. Additionally, the sensor data is reported on the images taken along with the timestamp so that the data at a given time can be directly correlated to the image being taken. While this website is not ideal for all uses since access to a free MathWorks account has limitations, but since the addition of sensors and data logging is not integral to the function of the basic device, this could be customized by an end user if they did not have access to the necessary features.

Chapter 5: Design Verification

5.1 Preliminary Testing

Once a final design was selected and built for this device, it was necessary to test the functionality of the components together. The first test which we ran was a test of the USB Microscope and Raspberry Pi 3 B+ with MotionEye software together. We needed to assure that the Raspberry Pi was capable of detecting the image from the USB microscope, and communicating it through the MotionEye software. The USB microscope was plugged into the Raspberry Pi and positioned above a hemocytometer. The IP address of the Raspberry Pi was found, and a laptop was used to go to the URL for the MotionEye user interface. A live image of the hemocytometer was able to be seen from the USB Microscope, and a still image was captured. This image is below in **Figure 41**.

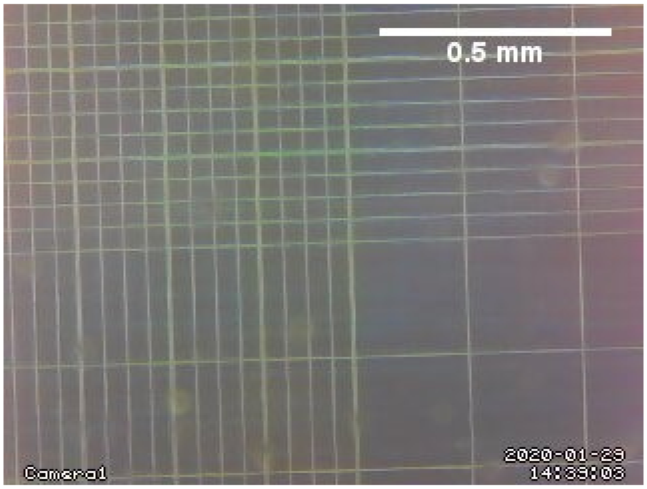


Figure 41: Image Taken of Hemocytometer with USB Microscope and Pi 3 B+ with MotionEye Software

The next test was a test on the lighting source, the Pimoroni UnicornHat 8x8 LED array, with the Raspberry Pi 3 B+ and USB microscope system. In order to test the LED array, it was attached to the Pi, and the USB camera was used to capture an image. A hair was used as a sample for the image to focus on. **Figure 42** below shows the hair under the LED with all LEDs turned on.

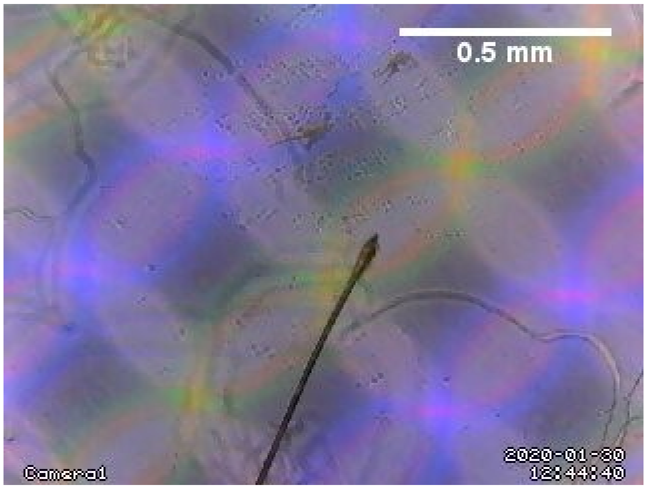


Figure 42: Hair Viewed Using USB Microscope with All LEDs Activated

Based on this test we determined that having the full LED array turned on (bright field microscopy) led to the color makeup of the white light (the RGB leds) being visible in the image. More testing of bright field microscopy with the LED is required with a different sample to determine how visible the colors are when the magnification is greater and distance between the image gathering source and LED array is greater. Dark field microscopy was tested next which consisted of just a ring of LEDs being turned on. This test was much more successful at showing the detail of the hair sample and is closer to what the LED array will be used for as a part of culture experiments. This can be seen below in **Figure 43**.

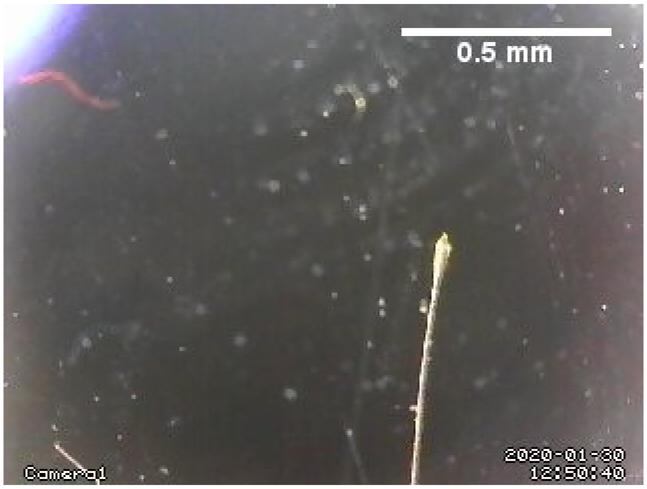


Figure 43: Hair Viewed Using USB Microscope with Ring of LEDs Activated

An additional test was run to determine if the device was capable of capturing images at set intervals (determined in the MotionEye User Interface) and create a time lapse video. This was the first time lapse taken on our device with the LED array, and Raspberry Pi 3B+. The objective of this time lapse was to test if all the components work together to clearly image small organisms. *C. elegans* were used as an example organism, and a time lapse was completed using the device to see how the organisms would react to rapidly changing light colors. We also wanted to verify that changing the interval at which pictures are taken would work by observing the change in seconds with each picture within the time lapse. At the beginning of the time lapse, photos were set to be taken every 2 seconds. One quarter of the way through, the interval was changed to take pictures every second. For the last third of the time lapse, the interval was set to take a picture every 3 seconds. The color of the lights varied due to a preprogrammed script that came with the 8x8 UnicornHat library. The overall time lapse was taken across 2 minutes and 13 seconds, and the pictures from each second of the experiment were compiled at a rate of 20 fps, resulting in a 3 second time lapse. The device was able to successfully capture images and create a downloadable time lapse video. An example image can be seen below in **Figure 44** and the video can be viewed at https://youtu.be/tKdHw6qSjQs.

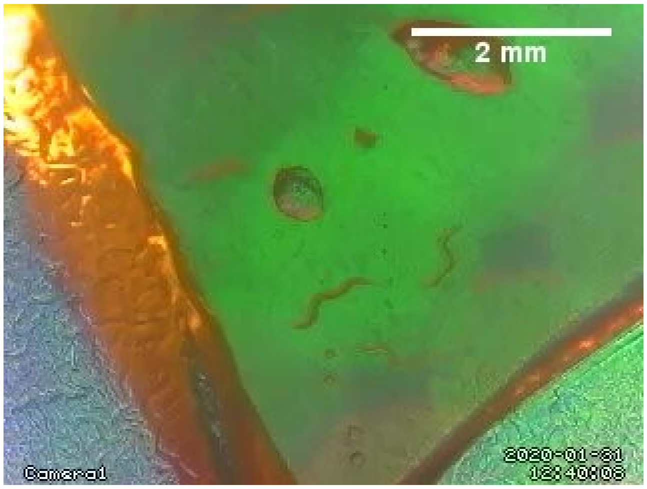


Figure 44: C. elegans Under Multi-colored Light

Once it was determined that all of the components functioned together, we tested the ability of the device to function within an incubator. This consisted of testing the ability of the wireless signal to leave the incubator, as well as assuring that the device is not affected by the humidity and temperature of the incubator. The Raspberry Pi 3 B+ was put into an incubator with a power supply and the door was closed. In order to test the wireless capabilities a laptop outside the incubator attempted to log into the MotionEye User Interface. The laptop was able to login and see an image, meaning that the signal was able to exit the incubator. However, additional testing to determine signal strength from the closest access point was not done due to not having access to the labs in D-term. The command sudo iwconfig wlan0 or sudo iwlist wlan0 scan can provide the user with information such as signal strength in magnitude (not dB), bitrate, package delivery protocol, ESSID, and frequency band of the wireless signal [49]. With this wifi test successful it was important to assure that the device would not be affected by long term exposure to the conditions inside the incubator. Since the Raspberry Pi 3 B+ is fundamentally a circuit board, it was always planned to have it inside a case. This case would protect the majority of the Pi from the humidity within the incubator which was the greatest concern. The only areas of weakness for the case were the openings for the Raspberry Pi ports. In order to prevent any water entering these ports, the unnecessary openings were covered in a material, sold under the trade name Sugru, which hardens into water-proof rubber after 24 hours. The remaining ports can be sealed after the device is set up for each new experiment, and the material removed at the end if necessary.

5.2 Verification

5.2.1 Experimental Results

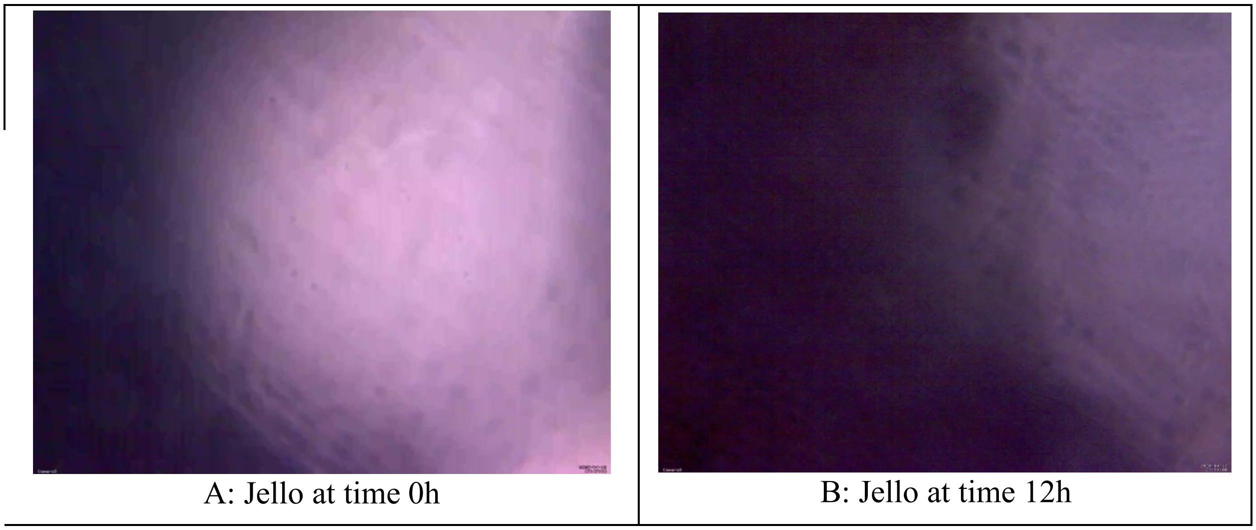
After assuring that the entire device works and can be used in an incubator the next step is to validate the ability of the device to work for the intended applications. A list of experiments to potentially run with the device were compiled by the team as shown in **Table 16** below. There were two broad categories of experiments we found feasible, time lapse experiments which are interesting and closer to what the device would be used for at a high school level, and time lapse of more cutting-edge college level experiments. Time lapse ideas in the no column were eliminated for safety concerns, and ease of access of materials concerns.

Feasible For Home	Feasible In Lab	No
● Ice freezing ● Swab some bacteria on agar and watch it grow ● Food molding (e.g. strawberries) ● Out window time lapse ● Flower blooming ● *C. elegans* ● Ice melting ● Ants eating food	● Cell attachment in different materials with different hydrophilicities (different contact angles) ● Decellularization of spinach leaves with fluid flow ● C2C12 Mouse myoblast differentiation ● Chemotaxis with 3T3 cells	● Water evaporating ● Reacting metal ● Flower staining ● Leaf drying up (from a flower somewhere) ● Bean sprouts growing ● Attach wires to power supply and put other end into water to watch it oxidize ● Put metal in saltwater and watch it oxidize

Table 16: Potential Experiments

In terms of the first category planned experiments include ice freezing, food molding, flower staining, and bacterial growth on agarose. Unfortunately, due to the campus closure related to the Coronavirus Outbreak of 2020, the cutting-edge experiments were not able to be performed as the laboratories couldn’t be accessed. In order to still complete verification testing, each member of the team had an individual system in order to test the device from home. Two members had Pi 3B+s and one had a Pi Zero. All of the Raspberry Pis were equipped with Motioneye. One member with a 3B+ had USB Microscope 2 and the printed housing model. The other 3B+ system had a Raspberry Pi V2.1 camera and the Zero had USB Microscope 1.

The first experiment run with the Pi Zero system was an attempt to watch bacteria grow on gelatin. Store brand jello was placed in one well of a 6-well plate and allowed to set. Once the jello was set bacteria was acquired from a cat's mouth and swabbed onto the jello. This was monitored with the device for 12 hours, taking an image every 60s and a time lapse was created at a 40 frame/s rate. The device was successfully able to monitor the sample for the full length of time and the time lapse was easily created using MotionEye. Unfortunately, no growth can be seen in the time lapse but there is movement which is believed to come from the jello melting over the course of the experiment. A sample image of the jello is shown below in **Figure 45** and the video can be viewed at https://youtu.be/lmnZLTp-rMQ.

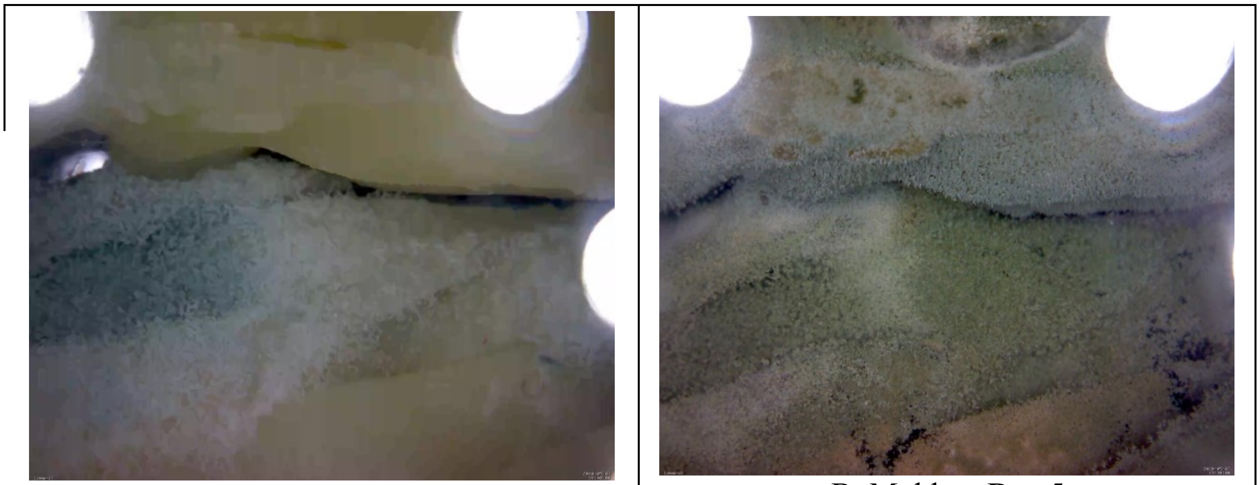


A: Jello at time 0h

B: Jello at time 12h

Figure 45: Picture of Jello Melting Taken with Pi Zero and USB Microscope 1

The second experiment run with the Pi Zero system was a multiday mold growth experiment. Cheese which had grown mold spots was placed in one well of a 6-well plate and the system was set up to take a photo every 5 minutes. The system recorded the mold growth for just under 5 days. These images were compiled into a single time lapse with a rate of 30 frames/s. An example image of the moldy cheese can be seen below in **Figure 46** and the video can be viewed at https://youtu.be/lFgpFkF1l2U.

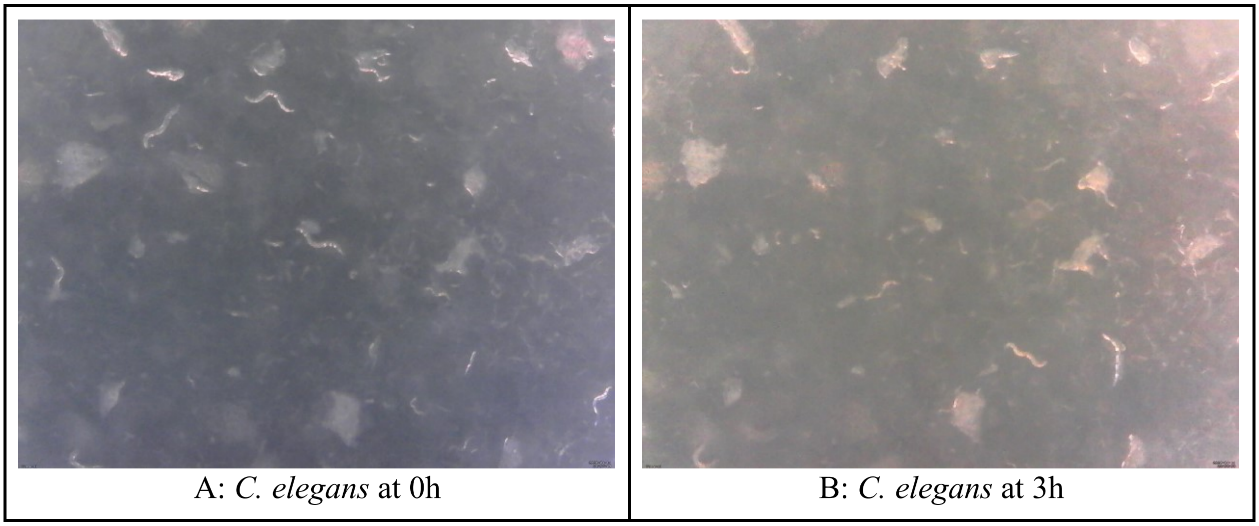


A: Mold on Day 1

B: Mold on Day 5

Figure 46: Mold on Cheese Viewed with USB Microscope 1 and Pi Zero

The other experiment we ran with the whole system involving *C. elegans* was a focus drift experiment. A time lapse of *C. elegans* was taken across 3 hours and an image was taken every second. The time lapse compiled images from every 5 seconds at a rate of 20 fps. The purpose of this experiment was to determine whether the presence of focus drift was a concern for our system. The results of this experiment are explained in Section 5.2.x.

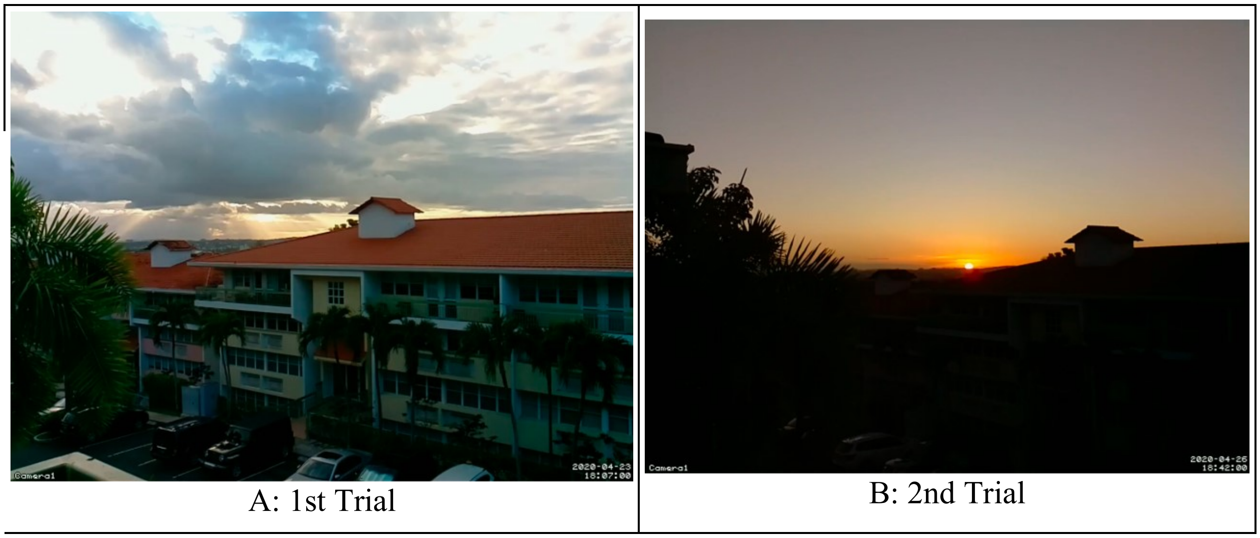


A: *C. elegans* at 0h

B: *C. elegans* at 3h

Figure 47: C. elegans Movement over 3 Hours

The next experiment consisted of taking a time lapse video of the new from a balcony. This particular experiment was done on two separate occasions. In both occasions, the device used consisted of a Raspberry Pi 3b+ microprocessor and a Raspberry Pi camera. For the first trial of the experiment, the device was set to take pictures at 1 frame per minute and left to run for about 3.5 hours. The images were compiled into a time lapse video of 10 frames per second. For the second trial of these experiments, the device was set to take pictures at 2 frames per minute for about 1.5 hours. Both trials successfully recorded the progression of the last few hours of natural sunlight and the sunset. Example images of what the time lapse videos for these experiments showed can be seen in **Figure 48** below and the videos can be viewed at and the time lapse videos can be viewed at https://www.youtube.com/watch?v=cFQoCEsq_4I and https://www.youtube.com/watch?v=oJF5BBZKzks respectively:

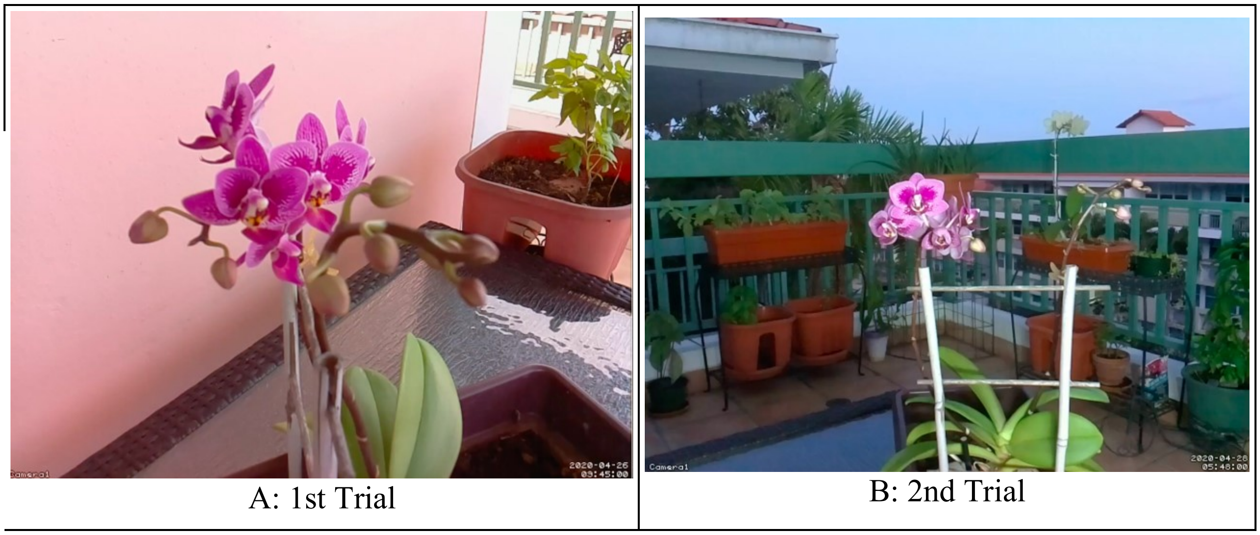


A: 1st Trial

B: 2nd Trial

Figure 48: Example Frames of Two Trials Performed for Balcony View Experiment

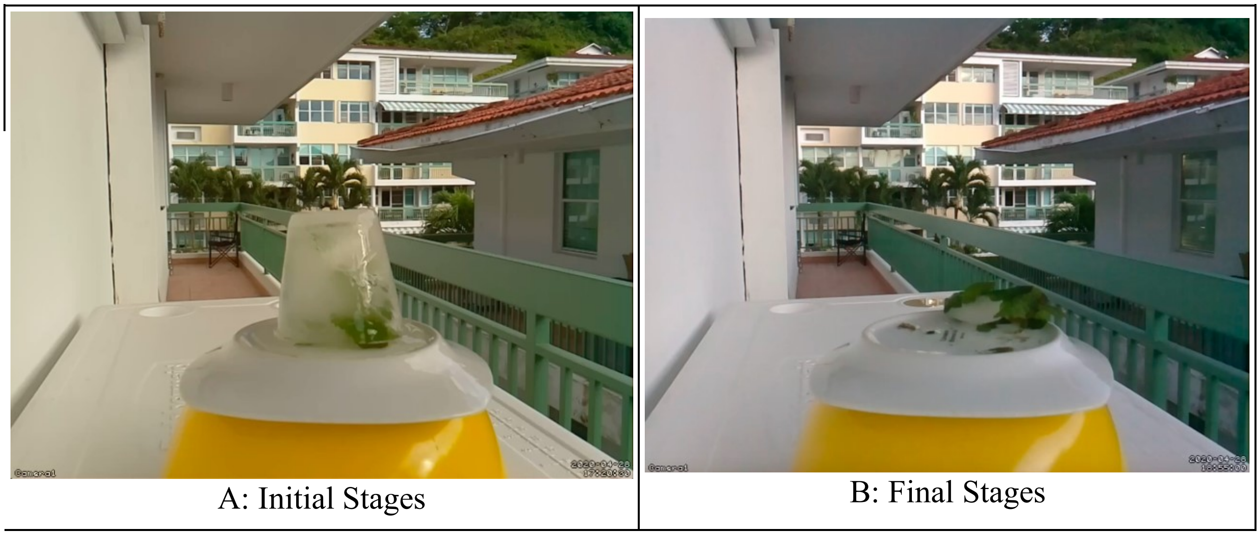
We also conducted a flower experiment in which we attempted to capture a flower's blooming process. This experiment was also done on two separate occasions. The time lapses for this experiment were taken using a Raspberry Pi 3B+ microprocessor and a Raspberry Pi camera. For the first trail, the device was set to take a picture every 3 minutes for 2 and a half days (about 60 hours). For the second trial, the device was set to take pictures every 3 minutes for almost two days (about 40 hours). Both trials showed the meticulous process of a flower blooming. However, a lot of the process is lost in the first trial due to lack of illumination during night-time, so a lightbulb was installed for the second trial to be able to capture the entire process. Example images of what the time lapse videos for these experiments showed can be seen in **Figure 49** below and the videos can be viewed at https://www.youtube.com/watch?v=xY2RAhfInh8 and https://www.youtube.com/watch?v=9Vdx0hvJOzA respectively:


A: 1st Trial

B: 2nd Trial

Figure 49: Example Frames of Two Trials Performed for Flower Blooming Experiment

An ice melting time lapse was gathered as well. For this experiment a Raspberry Pi 3b+ and a Raspberry Pi camera were used to gather the images. The device was set to take pictures at 2 frames per second for about 2 hours. This time lapse showed an ice melting naturally due to the difference in temperature between the ice block and the outside temperature (which was about 85 degrees Celsius that day). Example images of the experiment can be found in **Figure 50** below and the video can be viewed at https://www.youtube.com/watch?v=3yuzEA8iRbw:


A: Initial Stages

B: Final Stages

Figure 50: Example Frames of Ice Melting Experiment

We also conducted an experiment recording a spider's activity for several hours. For this experiment, an indoor spider was collected and put inside an improvised chamber. The device used to record this experiment consisted of a raspberry Pi Zero microprocessor and a Raspberry Pi autofocus camera. An LED array was used as backlight and the device was set to record images for about two and a half hours. We were able to successfully perform a quantitative analysis from the images and data recorded from this experiment. Example images of the time lapse video for this experiment is shown in **Figure 51** below and the video can be viewed at https://www.youtube.com/watch?v=ngtG7XjpiVc :

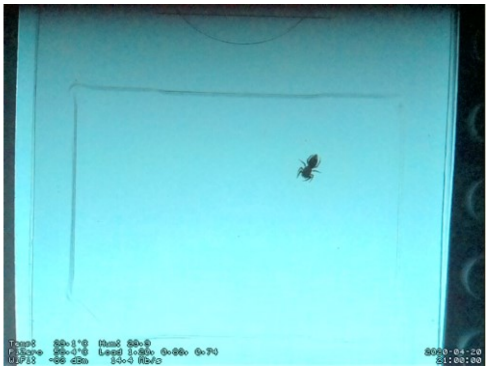


Figure 51: Example Frame of Spider Activity Experiment

5.2.2 Sensor Results

5.2.2.1 Temperature and Humidity Results

Three experiments were conducted to validate our choice to use the AM2302/DHT22 as our temperature and humidity sensor. The objective of our first experiment was to test the precision of the AM2302/DHT22 in comparison with an FDA approved medical device, the Cat. 16811-100 Digital Thermometer by Allegiance, a Cardinal Health Company. We blew a hairdryer with the same configuration over both our temperature sensor and our thermometer at a specified distance and observed the change in time versus temperature. We did this to measure the R^2 value to determine the precision of both the thermometer and our sensor. The results are shown in **Figures 52 - 57** below:

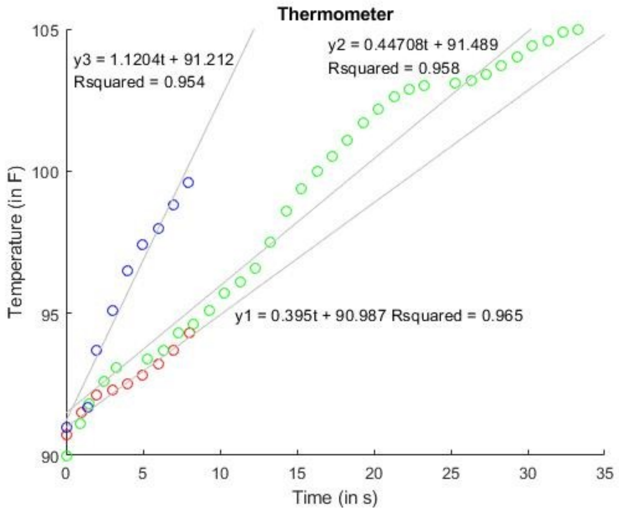


Figure 52: Thermometer Temperature Graph vs. Time

Figure 52 above plots the temperature vs. time for each trial, along with the computed R^2 value and linear regression equation. The three trials were plotted separately because plotting all the data together was potentially skewing the R^2 value due to human error of the experiment. The procedure and possible areas for error will be described in Section 6.3.

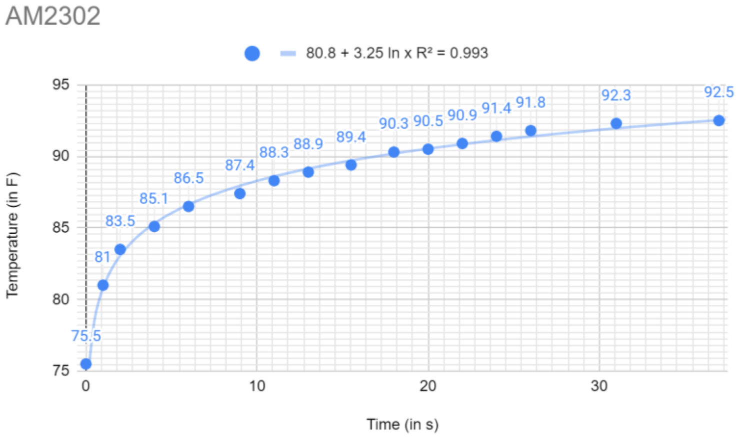


Figure 53: Trial 1 AM2302 Temperature vs. Time

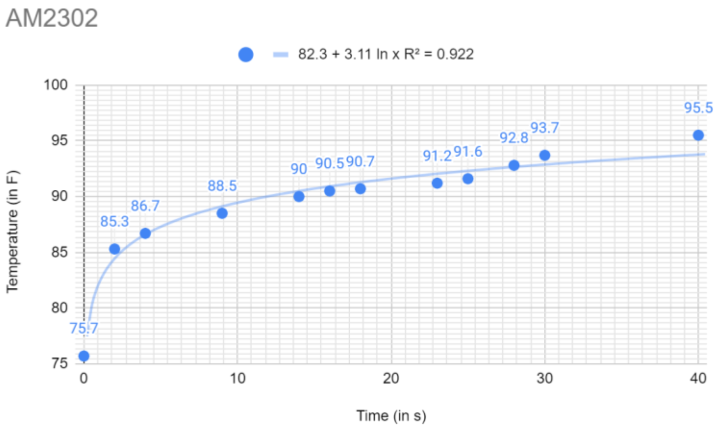


Figure 54: Trial 2 AM2302 Temperature vs. Time

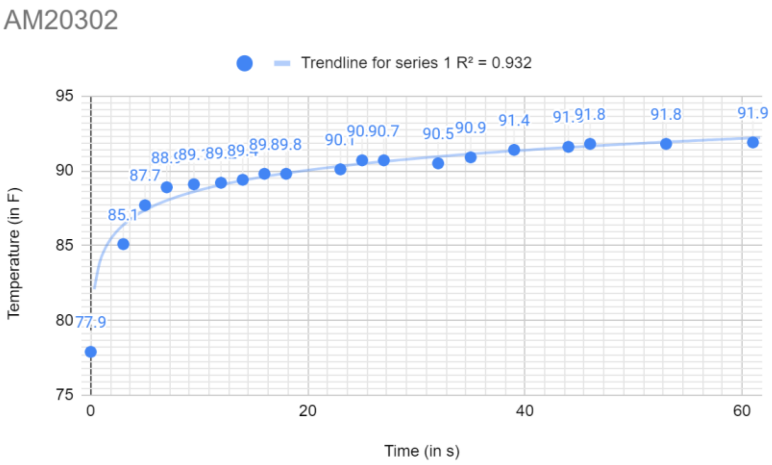


Figure 55: Trial 3 AM2302 Temperature vs. Time

In **Figure 53 - 55** above, the trials are plotted separately, once again to ensure human error would not affect the R^2 value of the sensor.

Another experiment we ran to better control the ambient temperature and humidity being read by the AM2302/DHT22 is we put our system in the oven and set the oven to 150 F. Ideally, we would have set our oven to 100 F, but it was unable to produce a controlled temperature that low. The objective of this experiment was to verify the AM2302/DHT22 can accurately read the ambient temperature, given that we are able to control the temperature of the environment it is in. The following graphs in **Figure 56 and Figure 57** from the experiment were collected using ThingSpeak.

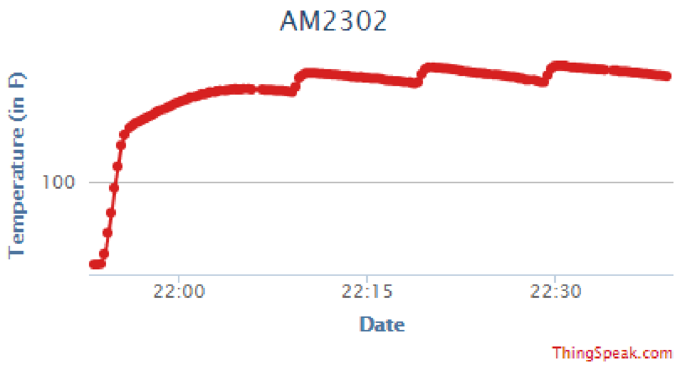


Figure 56: AM2302 Temperature vs. Time in Oven

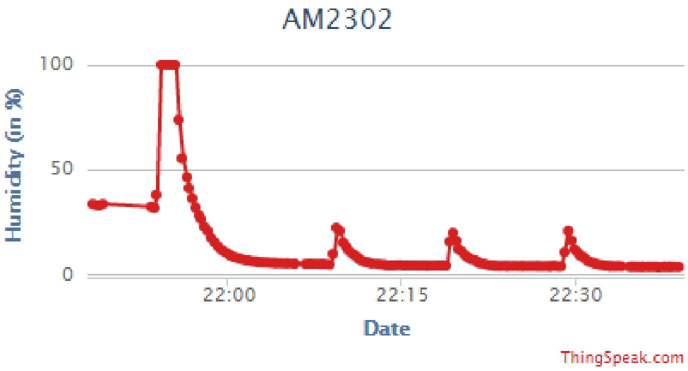
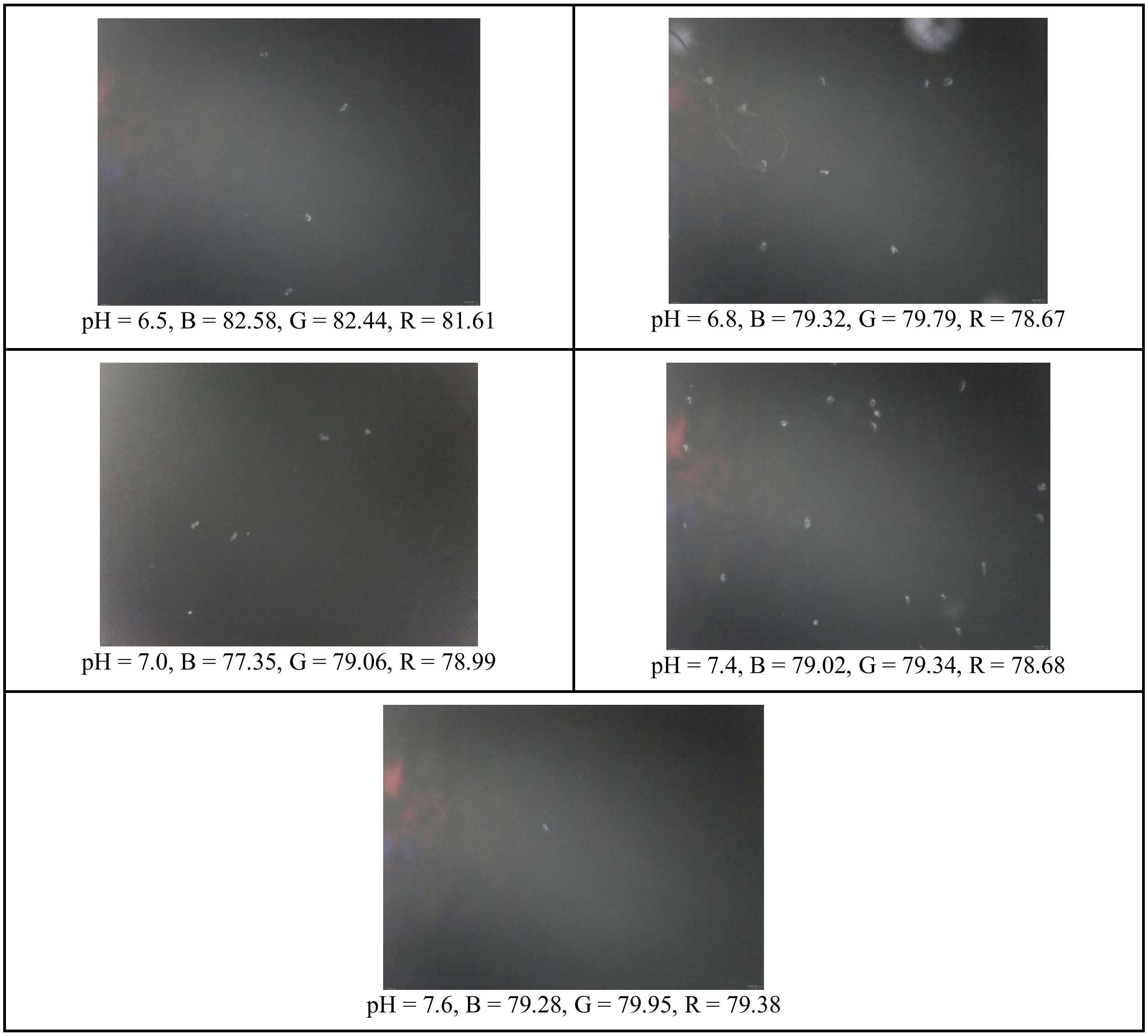


Figure 57: AM2302 %Humidity vs. Time

According to the Samsung website, which is the brand of oven, which was used, calibrated consumer ovens can vary in temperature ±25 F, which is in accordance with manufacturing standards [50]. Since the temperature was set to 150 F, the temperature sensor could have measured readings varying from 125 F - 175 F. The oven indicated it was done preheating the oven 5 minutes after the experiment started, at 21:58. This is important to note since the temperatures succeeding this time should be within the tolerance of the oven.

5.2.2.2 pH Sensing

Attempts to calibrate the RGB values to specific pHs were conducted to create a correlation between the two. This was done with the expectation that the color of a solution would change and its RGB values could be detected via image processing, as described in Section 4.4.7. **Figure 58** below shows images of solutions at different pHs taken with our system.



pH = 6.5, B = 82.58, G = 82.44, R = 81.61

pH = 6.8, B = 79.32, G = 79.79, R = 78.67

pH = 7.0, B = 77.35, G = 79.06, R = 78.99

pH = 7.4, B = 79.02, G = 79.34, R = 78.68

pH = 7.6, B = 79.28, G = 79.95, R = 79.38

Figure 58: RGB values of Solutions taken at 250x Magnification (rounded to the nearest hundredth)

It is clear that the images taken do not vary in color to the naked eye, and the program detected little variation in RGB values. This is likely due to a lack of contrast in the image. There are several possibilities for this, which will be discussed in Chapter 7.

5.2.3 Data Logging

Once ThingSpeak was chosen as the data logging software for this project, the next step was to undergo the process of creating instructions for its use and testing its functionality. The process of using ThingSpeak for Raspberry Pi data logging was to create a script in Python on the Pi which would run continuously and send the data captured with the attached sensors to the cloud log. Our advisor uses ThingSpeak in his lab and was therefore able to give us basic instructions on how to use it which we adapted for our purposes. These adapted instructions for ThingSpeak use are included as part of our Step-by-Step assembly guide found in Appendix A. Once the code was created and set up to capture the data from a sensor, the functionality of ThingSpeak was tested by attaching a sensor to the Raspberry Pi and recording its output for multiple days. This test was run on the Raspberry Pi Zero system using a DHT11 Temperature and Humidity Sensor. The sensor ran continuously for 5 days. A sample of the data recorded by ThingSpeak can be seen below in **Figure 59**.

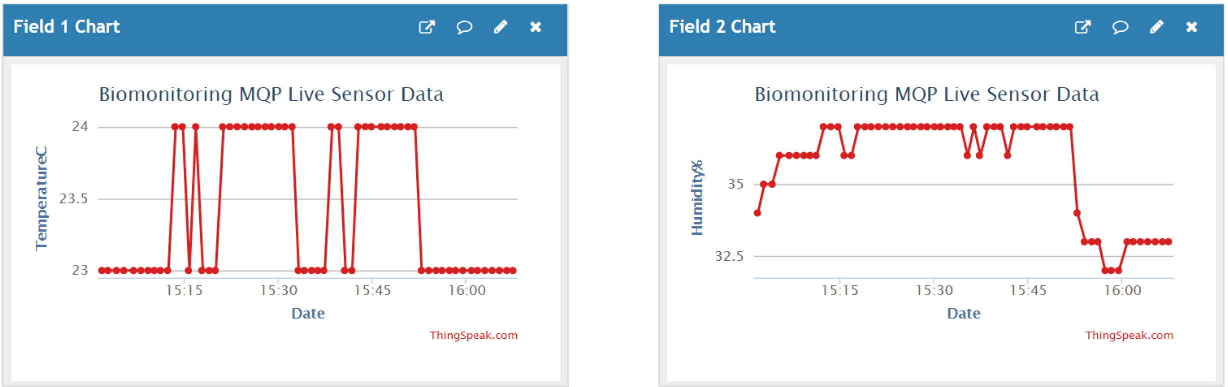


Figure 59: DHT11 Temperature and Humidity Data Collected with ThingSpeak

Since the system is meant to run in an environment where it cannot be connected to a monitor while running, an SSH connection was used to access the Pi's command window remotely from a laptop to run the code. Overall, the use of ThingSpeak was deemed effective for sensor data logging.

5.2.4 Live Sample Tracking

As a monitoring device potentially used for data analysis and observation, it is important that the device is able to obtain significant numerical data from the compiled videos or images. Quantification of the size, position and movement of an object within an image sequence can produce useful information for many studies and applications in the medical and biomedical engineering field. In order to assess this feature, we compiled a time lapse video of a spider inside an enclosed space to track and quantify its movement over time (**Figure 60**). In this case, the type of data obtained can help us understand important information about animal behavior. For example, patterns of activity relating to time of the day, patterns of movement in certain motion or direction to accomplish different goals, etc.

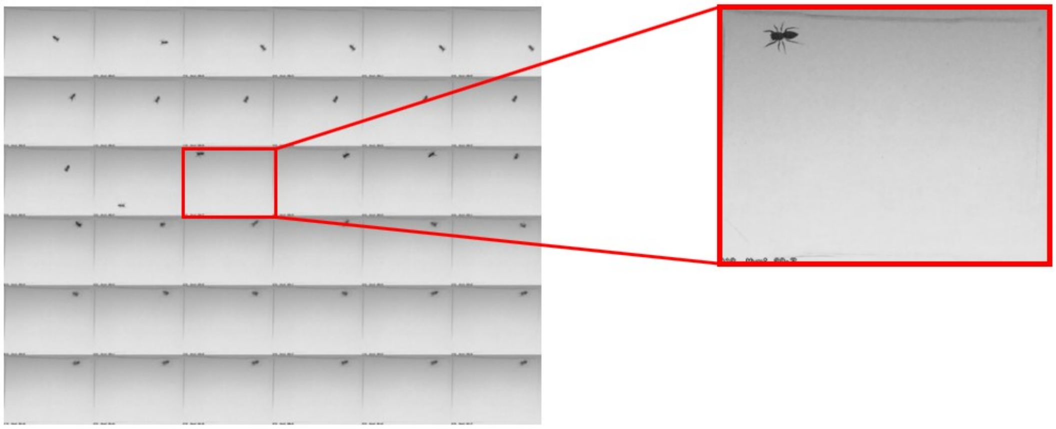


Figure 60: Image Sequence of the Spider Movement Over Time

For this particular experiment, an indoor spider was collected and kept inside a 32x24mm SD card case as a type of cage or chamber. The device used to record the time lapse consisted of a Raspberry Pi Zero microprocessor and an autofocus Raspberry Pi camera. An LED backlight was placed underneath the case to create a bright background effect. The autofocus camera was placed about 6cm above the case. Motioneye was accessed via IP address and used to set automatic and continuous image collection at 1 frame per minute. To quantify, visualize and track the spider's activity, we used the ImageJ program to perform image processing and segmentation. By processing and adjusting the image sequence, we were able to simplify the data by eliminating the unnecessary information within them and highlighting the important elements. Similarly, we used image segmentation techniques to translate the processed image sequence to quantitative data. First, we cropped out the border pixels that were not holding any important information (**Figure 61.A to 61.B**); this means the sections of the images that the object of interest (the spider) did not appear in. Then, we converted the color images to a grayscale (**Figure 61.B to 61.C**). Following this, we reduced the background information by separating the background from the object of interest. This was done by generating a MAX Z projection of the image and subtracting it from the original image (**Figure 61.C to 61.D**). Then, we set a threshold so that the lighter pixels were easily distinguished from the darker pixels (**Figure 61.E**) . The best threshold is the one that best outlines and fills the object of interest in red. Lastly, to eliminate noise and for further specificity of the object, we set the program to consider particles with an area of 2 to 30mm^2 and circularity of 0 to 1 and replace the object with a representative ellipse in its place (**Figure 61.F**). Lastly, a scale was created and calibrated based on known lengths identified in the image with their pixel equivalents calculated by the program.

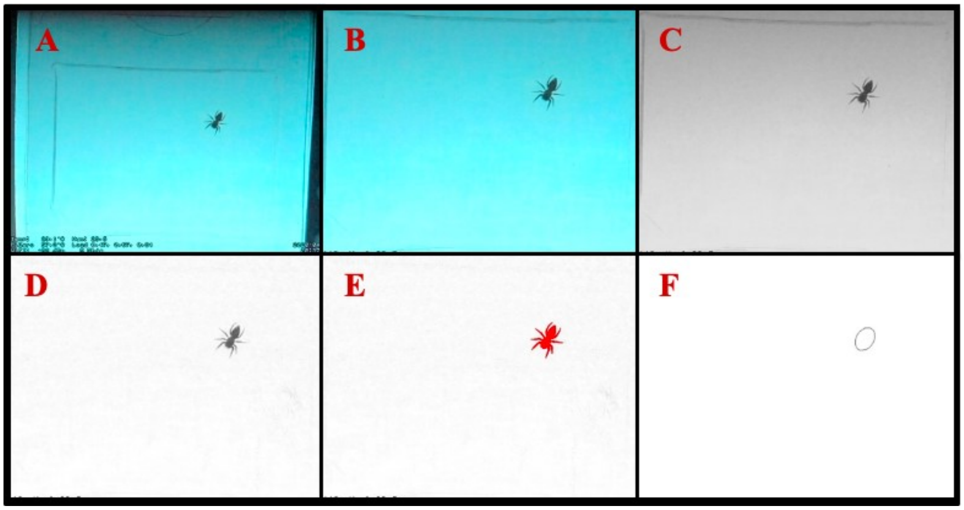
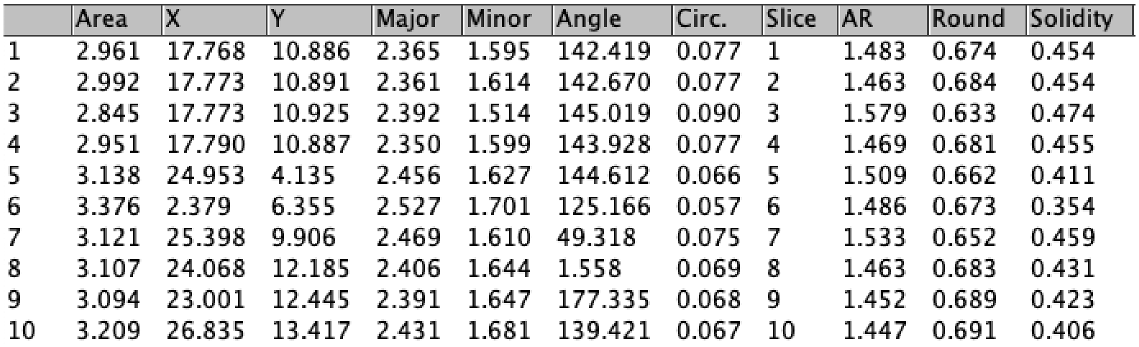


Figure 61: Image Processing and Segmentation Process

After setting the appropriate parameters discussed above, a table with data resulting from the image sequence analysis was generated to display the frame number, area, position in the x axis, position in the y axis, angle and other useful information. The spider's displacement was calculated from the x and y values using the distance formula and an excel sheet. Additionally, the spider's displacement and orientation were graphed over time.



	Area	X	Y	Major	Minor	Angle	Circ.	Slice	AR	Round	Solidity
1	2.961	17.768	10.886	2.365	1.595	142.419	0.077	1	1.483	0.674	0.454
2	2.992	17.773	10.891	2.361	1.614	142.670	0.077	2	1.463	0.684	0.454
3	2.845	17.773	10.925	2.392	1.514	145.019	0.090	3	1.579	0.633	0.474
4	2.951	17.790	10.887	2.350	1.599	143.928	0.077	4	1.469	0.681	0.455
5	3.138	24.953	4.135	2.456	1.627	144.612	0.066	5	1.509	0.662	0.411
6	3.376	2.379	6.355	2.527	1.701	125.166	0.057	6	1.486	0.673	0.354
7	3.121	25.398	9.906	2.469	1.610	49.318	0.075	7	1.533	0.652	0.459
8	3.107	24.068	12.185	2.406	1.644	1.558	0.069	8	1.463	0.683	0.431
9	3.094	23.001	12.445	2.391	1.647	177.335	0.068	9	1.452	0.689	0.423
10	3.209	26.835	13.417	2.431	1.681	139.421	0.067	10	1.447	0.691	0.406

Table 17: Snippet of the Data Table Generated Using ImageJ

The initial time lapse by itself only offered a visual representation of the spider's activity. However, combining the video with the data and the graphs generated by the data allows for a clearer and more specific interpretation of the animal's behavior. For example, a change in displacement and orientation during the same time points in both graphs might represent exploration. No significant change in displacement or orientation might represent rest or feeding. A change in orientation but no change in displacement might represent turning movements, and constant changes in orientation but small changes in displacement over a sustained period of time might represent web making. Higher activity at specific timepoints in a day might represent the animal's tendency to be more active at certain times or conditions. The pattern for the spider's behavior during the time lapse video can be observed in the figures below:

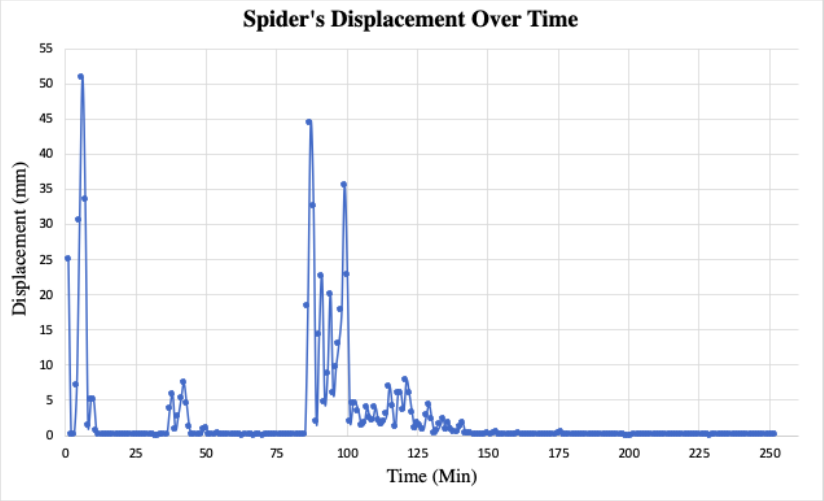


Figure 62: Graph of the Spider's Displacement Over Time

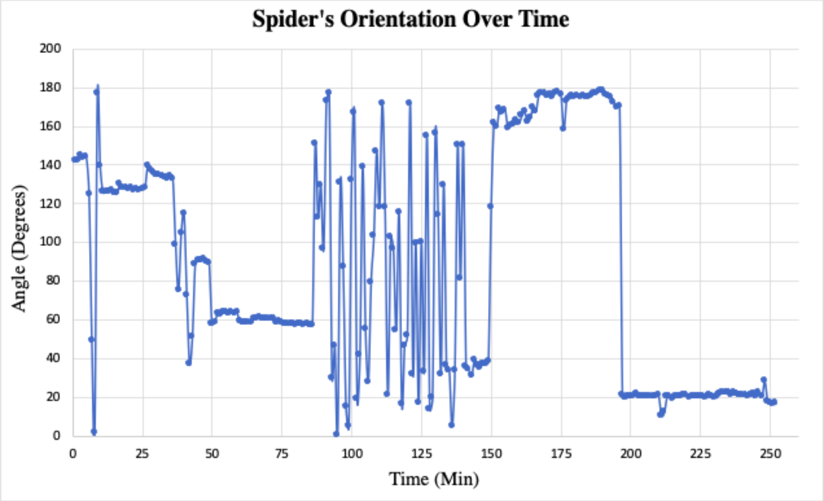


Figure 63: Graph of the Spider's Orientation Over Time

We can observe from the graphs that from minutes 0 to 10, there is high displacement with changes in angular orientation, meaning the spider was most likely roaming around the enclosed space. From minutes 10 to 80, there is smaller displacement and some angular differences, meaning the spider was still roaming around but not as much as before. From minutes 80 to 100, we can see high displacement and angles again, meaning the spider went back to roaming all around, probably looking for the best place to start a web. From minutes 100 to 140, there is frequent angular change and small displacement, meaning the spider was most likely building a web. By looking at the time lapse video around minutes 100 to 140, we can confirm that the spider was indeed building a web. And lastly, from minutes 150 to 250, there is almost no displacement and a couple of sudden angular changes, meaning the spider was probably resting and shifted positions a couple of times. This process demonstrates that we can successfully perform quantitative tracking of moving objects and organisms on the videos and images collected using our device. This assessment can be applied to cell and cell culture-related studies as well as many other biological organism monitoring applications. In this particular experiment, we were able to capture basic spider behavior including web building and we were able to extract quantitative data such as angular change, displacement and time duration through that process.

Chapter 6: Final Design And Validation

6.1 Meeting Objectives

The main goal for this project was to create a device capable of remotely monitoring cell/organism cultures that would be more accessible than current gold standard devices. As we have previously discussed, our design objectives included allowing the user to have remote control and access to data, enhancing the accessibility of the device, making the device customizable and adaptable to any lab setting and application, demonstrating full functionality of the device through experiments and minimizing the need to take the device out of the controlled-environment chamber. These objectives were achieved by creating a small and inexpensive device that uses open source code for remote monitoring. To achieve remote access and control of our device, we used Motioneye's software interface, which allows the user to access the system from anywhere in the world just by knowing the device IP address. Through the software interface, the user can access a livestream camera view with the corresponding overlaid sensor data. The user also has the option to take still images and videos, compile time lapses, and download files. In order to make the device more accessible to a wider spectrum of potential users, we designed a device using inexpensive parts and components which ended up costing about $200 per unit. This price is significantly lower than the price of gold standard devices which can cost anywhere from $2,000 to $200,000. Additionally, we made our device easily reproducible by making a step-by-step guide with customizable options to fit every lab setting, need and application. Each member of the team was able to successfully create a device from scratch and customize it using the step-by-step guide based on the experiments we each performed. We were able to test the functionality of the device by gathering different time lapses with overlaid sensor data and using them to perform a quantitative analysis of the experiments. Based on these time lapse experiments, we determined that the device is capable of capturing images as frequent as 2 frames per second, which is enough to capture C. elegans movement and activity. Additionally, we also determined that the device can continuously record data if it is powered by an outlet. However, the longest recording time if powered by a battery pack will depend on the battery pack used. Lastly, we designed our device to be compact and comfortable in size when compared to gold standard devices. Our final design is 31cm tall, 11.2 cm wide and 11.2 cm long.

6.2 Reproducibility

In order to make this project easily reproducible, and accessible to use in real world settings we created a step by step guide to recreate the final design of the device and this guide can be found in Appendix A. The guide has subsections for the creation of the physical device, the software setup, how to create time lapses, optional features, and alternative design options for customization. Within the hardware assembly instructions is included the overview of how to put the microprocessor unit in its case and attach it to the SD card, power cord, and external mouse and keyboard as well as detailed instructions for how to print and assemble the device housing using the included 3D CAD models. These CAD models were made accessible for free online using Thingiverse so that all an end user who wants to recreate this device exactly has to do is download the STL files from the provided links and then 3D print the housing components and assemble. Links to the screws and wingnuts used for the housing are also included in the guide along with a full description and part numbers to ensure the purchase of the correct items. The software setup instructions include open source code and detailed instructions for the user to; download and set up the appropriate operating system on the Pi, connect to WiFi networks, download and install the MotionEye software, enable port forwarding, and how-to setup SSH which allows remote access to the Pi’s command terminal. The time lapse setup section of this guide details how to capture still images at set time points and download these images as a time lapse using the MotionEye software. The optional additions section details how to set up temperature/humidity sensors, pH sensing, data overlay on an image within MotionEye, data logging using cloud storage, and the setup for use of a mini display. The guide also includes a section on troubleshooting wifi connectivity, and a list of additional parts that could be used in place of specific components to allow customization of the device. In order to verify the ease of reproduction of this device each team member used the step by step guide to successfully build a biomonitoring device.

6.3 Experimental Methods

Each component of this device had to be tested to verify it completed our desired operations before testing all the chosen components together. We first tried to connect our Raspberry Pi with MotioneyeOS to WPI WiFi. A Raspberry Pi Zero WH was being used due to it being in the original problem statement given by our advisor. When meeting with WPI's NetOps, to connect the Raspberry Pi Zero WH with MotioneyeOS, it was discovered that MotioneyeOS could not connect to the network due to the timeout, as mentioned in Section 4.3.2. NetOps then created a set of instructions to connect a Raspberry Pi with Raspbian OS, which we used to connect our Pi Zero WH to WPI-Wireless. We then tried downloading and installing Motioneye on our Pi Zero WH, which was moving very slowly, as mentioned in Section 4.3.1. We decided to try downloading it on our 3B+, and the install process went much faster and successfully connected to the internet.

After that, all of the potential imaging sources were evaluated by imaging a hemocytometer as shown in Section 4.3.3. Because we hadn't decided on a lighting source yet, we used a pointed light source from our phones, which caused a very bright light in the corner of a picture, but diffused light amongst the rest of the picture, allowing us to observe enough contrast to differentiate the quality between imaging sources. In preliminary testing of the image sensors, pictures were captured using the manufacturer's recommended software. Once we had chosen an imaging source, we captured an image of the hemocytometer using the Motioneye interface, as shown in Section 5.1.

From seeing the effects of using a pointed light source and doing background research, the team was inspired to use a light source that allows for the position of the lighting to be controlled via programming, such that we could test the best lighting configuration. After purchasing a light source based, preliminary photos and time lapses were taken with multiple biological specimens, such as *C. elegans* and a hair, as shown in **Figures 39 and 41** in Section 5.1. The original lighting configuration tested was with all of the LEDs in the array on, where we saw a disco-light effect, which was likely due to the fact that the white light from the LEDs is made up of a red, green, and blue LED. Since this reflection impedes the user's ability to view the sample, we tried the dark field lighting method used by Watanabe [22] that we found in our background research and reimaged the hair, as shown in **Figure 42** in Section 5.1.

To test the precision of the AM2302/DHT22, we blew a hairdryer over Cat. 16811-100 Digital Thermometer by Allegiance, a Cardinal Health Company and the AM2302/DHT22. The heat settings on the hairdryer were the same for all trials, where it was set to low with a cool blast. The cool blast was set since the thermometer's temperature sensing range was less than the nominal heat of the hairdryer. The hairdryer was placed two inches away from both temperature sensing devices and it was made sure that the air was directly hitting the front of each device. However, it was possible for error to be introduced since the thermometer and the hairdryer were being held as opposed to being clamped to a structure, due to lack of supplies. The hairdryer was especially difficult to hold steadily since it took extra pressure to press the cool blast button. Temperature readings were taken by running the temperature script that came with the DHT22/AM2302 library in the terminal. The script was run as often as possible by the user. It was not possible to loop through this script automatically for the duration of the experiment as it would run out of RAM after 8 consecutive readings. Using ThingSpeak for this experiment was not possible since the base subscription for ThingSpeak updates the graph every 15 seconds, and the thermometer readings changed every second, meaning we wanted our AM2302/DHT22 sensor readings to be at a similar interval. After that, the data points of the temperature vs. time reading were plotted for both the thermometer and the AM2302/DHT22. Each trial was plotted separately as to not skew the R^2 value of each trial, which indicates precision. The higher an R^2 value of a curve, the stronger the correlation is, which indicates precision because it means the data points taken are following a clear pattern. The regression for both sensors was compared to temperature vs. time data of other temperature sensors to ensure the correlation was legitimate.

To test accuracy of the AM2302/DHT22, the sensor was placed in the oven on a tray, along with a battery pack, and the Raspberry Pi 3B+. The oven was set to 150 F since that was its lowest possible setting. For this experiment, data logging with ThingSpeak was set up to automatically graph the change in humidity and temperature. Using ThingSpeak, as opposed to manually logging data as described in the previous experiment, was possible since we were not comparing the AM2302/DHT22 to any other temperature sensor. So, we did not have to take another sensor's update rate into consideration. The experiment was run long enough to observe 3 peaks in temperature and humidity. The time at which the oven added more heat to the environment was observed to occur right before the humidity and temperature spiked. All the temperature readings after the oven was done preheating were checked to be within the tolerance of the oven's temperature. During this experiment an attempt was made to take a time lapse of the Cat 16811-100 Digital Thermometer, as used in the hair dryer experiment. This would have added further validity to the accuracy to our AM2302 sensor since it would be compared to an FDA approved medical device. However, the microscope that was reading the digital readout on the thermometer failed due to not being suitable for temperatures greater than 100 F. Additionally, the thermometer's readout range was significantly less than 150 F, tapping out at around 108 F, meaning the thermometer was not the best temperature sensor to compare the AM2302/DHT22 with during an experiment in the oven. Therefore, this experiment was reattempted with an analog temperature sensor that provides readouts greater than 150 F and a Raspberry Pi V2.1 Camera. However, this experiment was not able to be completed as the analog temperature sensor had to be far enough away from the camera in order to record the changing temperature, and the tools to do this were not available. It was theorized that the thermometer could be taped to the oven door, but heat resistant tape was not available. It was also theorized that the thermometer could be hung from a shelf, but there was a concern that the string would burn. Ideally, a configuration would've been found if we had access to the labs on campus, which was not possible due to COVID-19. Also, the Raspberry Pi V2.1 Camera could not be focused in the oven due to a lack of light, despite the oven's light being turned on.

For pH development we had to test each portion of the code. The RGB value of one pixel of an image taken with our system was analyzed and printed to the terminal to ensure RGB values could be analyzed. We then wrote a nested for loop around this command to retrieve the RGB values of the pixels in a 2 x 2 area, and then 10 x 10. As mentioned in Section 4.4.7, this results in a 2D array, where the elements in the outer array correspond to the number of pixels, and the elements in the inner array correspond to B, G, and R values. We then expanded this for loop to analyze the RGB values of the height and width of the image, resulting in a large 2D array, since every pixel of the image was being analyzed. We then tested averaging the RGB values using np.average with axis = 1 to ensure the output was a 3-element array, where each element was a B, G, or R value. An explanation of the averaging axis of this command is in Section 4.4.7. The next step was testing to make sure the program was pulling the most recent picture within the specified data folder in the camera configuration folder. We took a picture with our microscope that was completely black and loaded it into the program using a glob iterator, as mentioned in Section 4.4.7, and analyzed the RGB value. We then downloaded that picture to our desktop and analyzed its RGB value using Paint software to ensure the RGB values matched up. This test also proved to us that the program was detecting RGB values for different pictures. To test the picture was being taken from the most recent folder, we ran this test again, except with two glob iterators, one for file path within the camera configuration folder and one for picture within the most recent file path. We ran this code with the expectation that the RGB values received would be the same as that of the previous test, where we only retrieved the RGB values of the most recent picture in a known file path.

To calibrate the correlation between RGB and pH, solutions with a pH of 6.5, 6.8, 7.0, 7.4, and 7.6 were prepared. These solutions were prepared by adding baking soda to water, since baking soda's most active ingredient is $NaCO_3$, sodium bicarbonate. To measure the pH of these solutions, pH paper sticks were used that change color depending on the pH of the solution. The legend to determine what the corresponding pH was for each color of the paper stick came with the container of pH sticks. Originally, each solution was imaged using the dark field microscopy method, as described in Section 5.1. After seeing a lack of contrast to determine the color of solution, bright field microscopy was also attempted to see if that was a better technique to create contrast in the solution.

6.4 Standards

For this project, we outlined a set of standards in order for our device to be reproduced successfully and in compliance within regulations while still being able to ensure a quality product. The standards we highlighted are based on different aspects of our device. For example, the UL 60950-1 set of standards regulates the usage of computer and electronic equipment appropriately. The ISO 9000 regulates the usage and development of software coding. The IEEE 802.11 standard series regulates the wifi usage and the ISO 20387 standard regulates the appropriate usage of microorganisms for studies and data collection.

6.5 Global Impact

6.5.1 Economics

The low cost of producing this device would directly affect the economics of labs that are looking to use remote monitoring devices. The current gold standard devices which remotely monitor cultures while inside an incubator cost around $2,000 [8]. This means that these devices are inaccessible for many labs and programs. Our device is only around $200 to make which means that it can be accessed by low-resource programs and it makes the device more highly reproducible.

6.5.2 Environmental Impact

The biggest environmental impact that this project has is the fact that the housing is made out of plastic. Plastic was chosen for this project because it is cheap and accessible. However, the use of plastic for these purposes could potentially increase the amount of plastic entering landfills and therefore the environment. Specifically, because of the nature of the housing design, the process of 3D printing requires scaffold material to be printed to support them. Once the pieces are done printing this scaffold material is immediately thrown away. Since the device itself is durable and reusable however, there is ultimately not much waste produced per device.

6.5.3 Societal Influence

This project could potentially make fun science experiments using time lapse more accessible to high school and early college level students even in low resource environments. This socially influences the ordinary person by opening more opportunities for students to get an interest in higher level science, and potentially pursue careers in science or engineering. Additionally, this device can allow more labs access to time-lapse microscopy and continuous monitoring for experiments at a low cost which has the potential to further the knowledge gained from research which could positively affect society as a whole.

6.5.4 Political Ramifications

Likely, this project would not have any effect on the culture of other countries, or on the global market. The largest effect this may have would be making science more accessible in low resources areas of other countries as well. Additionally, this project may affect the market for the gold standard devices though it seems unlikely. This project could also potentially affect global politics through the increase of knowledge on science as it applies to public health.

6.5.5 Ethical Concerns

This project does not have any negative ethical concerns, and on the contrary brings up a discussion of the ethics of its gold standard devices. The high cost and lack of accessibility of these devices means that they are only available to the wealthy. This brings up the ethics of having such a device which could advance biomedical research be available to such a low number of researchers. Our project opens this technology up to more researchers so that the positive effects of its use can be more widely experienced.

6.5.6 Health and Safety Issues

The only health and safety issues related to this device come from the fact that it involves electricity and electronic components which if mishandled could lead to potential harm to the user. Additionally, the small parts such as the screws and wingnuts should be kept away from small children as they could pose a choking hazard.

6.5.7 Manufacturability

This project could be very easily reproduced. Attached as an appendix to this paper is a step by step guide on how to create this device including links to access the open source code used and the CAD models made of the housing design. This guide allows anyone to recreate the device and gives tips on how they can modify it for potential different desired applications. It is easily reproducible at a large scale as well because of the low cost of the materials, and relative ease of manufacture and assembly of the parts.

6.5.8 Sustainability

This project requires the use of electricity and is made of electronic and plastic parts. To make this project more sustainable, renewable energy could be used to power the device, and all electronic and plastic pieces should be recycled properly. For example, there are solar battery packs that can be used to power the device instead of using electricity from an outlet.

Chapter 7: Discussion

The preliminary testing of this device showed that each of the components could function in harmony with each other in order to capture images. This means that the main functional components of the device were working as expected so that we could move onto the next stage of verification by testing the device on actual experiments to collect data. The colored *C. elegans* time lapse was the first time lapse we took with small organisms with the whole device to verify all the components integrated well with each other. These preliminary tests cannot be taken to mean that the device definitively works as the housing was not used to hold the components in these experiments, and the tests were only run for a very short amount of time. However, these results were very promising in showing that the device could meet the main objectives of ensuring the device can do microscopy without external components, including a wireless feature to livestream culture, and demonstrating functionality.

The additional verification experiments run on the complete system each showed that the system could meet various objectives of the design goal. Overall, the creation of a step-by-step assembly guide meant the completion of an objective, and by using this guide to create separate systems for each group member to run experiments with, we met the objectives of ensuring use in any setting and customization for user applications.

The results of the jello experiment show that the device has complete functionality in regard to being able to capture a time lapse over the course of multiple hours. The cheese mold growth experiment was able to show more functionality in terms of an increased experiment duration, and a higher level of magnification which was required to see the mold well. This experiment's results also showed that the USB microscope did not have any noticeable focus drift over the course of an almost five-day long experiment. The plain *C. elegans* time lapse was taken to determine the presence of focus drift with the USB microscope. The sunset balcony view experiments were able to demonstrate the device can work under high temperature conditions and is able to record for prolonged hours under said conditions. The flower blooming experiments show the device will continuously record images for as long as needed if the device is connected to a power outlet and a steady internet connection. These experiments were run for about three consecutive days. The ice melting experiment shows the device can be set to take pictures at a framerate fast enough to capture the fast-changing stages of ice melting. The spider activity experiment shows the device is able to record data that can be used for a quantitative analysis. The live sample tracking analysis we performed using the spider activity experiment video demonstrates that we can successfully quantify the position, shape and movement of objects within our time lapse videos. This assessment can be very useful when monitoring cells or C. elegans in order to track movement or behavior over a period of time. All of these experiments show the versatility of our device designs by showing that both the Pi 3B+ and the Pi Zero WH work, that both of the USB microscopes and the Pi camera worked. Additionally, these experiments show that images can be compiled into time lapses both within MotionEye, and by downloading the images and compiling them externally. Additional testing with the Pi Zero device showed that a 2-fps frame capture rate could be achieved on the Pi as long as it could maintain a streaming rate above 2 fps. This streaming rate was unreliable though and showed that the Pi Zero has issues with high frame rate speeds. Due to the unreliable nature of this rate, a full experiment was not able to be run showing this 2-fps frame capture rate.

A summary of the time lapse videos and a link to them are shown in **Table 18** and **Table 19** respectively.

Ex. #	Sample	Device	Lighting	Framerate	Duration	Total Frames	Observations
1	Colored *C. elegans*	Pi 3B+ + USB microscope	8x8 UnicornHat LED Array	1 frame every second	2 mins	120 frames	Whole system is able to take time lapse videos with clear image
2	Bacteria growth on gelatin	Pi Zero + USB microscope	USB Microscope light	1 frame per minute	12 hours	720 frames	Melting of jello
3	Mold growth on cheese	Pi Zero + USB microscope	USB Microscope Light	1 frame every 5 minutes	120 hours	1440 frames	Spread of mold on pieces of cheese
4	White Light *C. elegans*	Pi 3B+ + USB microscope	8x8 UnicornHat LED Array	1 frame every second	3 hrs.	10,800 frames	No focus drift over 3 hours
5	Balcony view #1	Pi 3B+ + Pi camera	Natural light	1 frame per minute	3.5 hours	210 frames	Progression of sunset

6	Balcony view #2	Pi 3B+ + Pi camera	Natural light	2 frames per minute	1.5 hours	180 frames	Progression of sunset
7	Flower Blooming #1	Pi 3B+ + Pi camera	Natural light	1 frame every 3 mins	60 hours	1,200 frames	Progression of a flower blooming
8	Flower Blooming #2	Pi 3B+ + Pi camera	Natural light + lightbulb at night	1 frame every 3 mins	40.8 hours	816 frames	Progression of a flower blooming
9	Ice melting	Pi 3B+ + Pi camera	Natural light	2 frames per minute	2 hours	240 frames	Ice melting
10	Spider activity	Pi Zero + Pi autofocus camera	LED array backlight	1 frame per minute	2.5 hours	152 frames	Spider web building

Table 18: Summary of Time-Lapse Experiments

Experiment #	**Link to the time lapse videos**
1	https://www.youtube.com/watch?v=tKdHw6qSjQs
2	https://www.youtube.com/watch?v=lmnZLTp-rMQ
3	https://www.youtube.com/watch?v=lFgpFkF1l2U
4	https://www.youtube.com/watch?v=YDtAWWe8TzY
5	https://www.youtube.com/watch?v=cFQoCEsq_4I
6	https://www.youtube.com/watch?v=oJF5BBZKzks
7	https://www.youtube.com/watch?v=xY2RAhfInh8
8	https://www.youtube.com/watch?v=9Vdx0hvJOzA
9	https://www.youtube.com/watch?v=3yuzEA8iRbw
10	https://www.youtube.com/watch?v=ngtG7XjpiVc

Table 19: Link to the time lapse videos

The results of our data logging experiment show that the device is able to wirelessly communicate data to an external cloud server so that it can be recorded and logged successfully. Additionally, these results gave important information about the feasibility of using SSH to run long term command programs on the Raspberry Pi module.

Looking at the results from the first temperature experiment we were able to determine that each trial had high R^2 values within a 10% confidence interval for both the thermometer and the temperature sensor, as shown in **Figures 45 - 48** in Section 5.2.2.1. However, they do follow different regressions, as the thermometer follows a linear regression, whereas the AM2302/DHT22 follows an exponential with the equation $y = A(1-e^{kt})$, where A is the asymptote the regression approaches and k is the rate of change. This is likely because the thermometer uses a different sensing technology than the AM2302/DHT22, or the raw sensor data from the thermometer is digitally processed to output readings linearly.

The results from the second temperature experiment tell us whether the temperature sensor was accurately reading the temperature in the oven. As mentioned in Section 5.2.2.1, the temperatures read by the AM2302/DHT22 were expected to be in the range of the oven's tolerance, 125 F - 175 F, after the oven had finished preheating around 21:58. Data points at 21:58:17 and 21:58:42 read temperatures of 123.98 and 124.52, respectively, meaning these two temperature readings were outside the tolerance of the oven. It is likely this occurred due to a lag in the sensor readings in comparison to its environment. However, starting at 21:58:52, the temperature sensor was reading values from 125.06 F - 139.82 F, meaning the rest of the values were within the oven's tolerance.

During pH calibration, it was discovered that not enough contrast could be made using our dark field setup to determine the color of the solution. This is possibly due to the fact that the camera automatically adjusts to remove hues of the image. This is hypothesized because during calibration, it was observed that the hue of the image on the interface changed from being a light purple upon initial illumination of the device to black and white after a second had passed. Therefore, the white balance was adjusted within the motioneye interface, such that no auto white balancing would occur, and no change was seen. Another reason a lack of contrast could have occurred is because the image sensor was only magnified to look at the solution in a culture plate during calibration, meaning it had no other objects in view to compare the color of the solution to.

We also tested a bright field setup to see if this would change due to more light being transmitted through the sample. However, the bright field configuration showed even less contrast, as the sample appeared as if it were just a clear solution. Adjustments to white balance were made again, such that no auto white balance would occur, and no change was seen. Recommendations on future testing and development are written in Section 8.2.

Overall, we were able to create a device that costs about $200 which is significantly cheaper than the gold standard devices that can cost up to $200,000. This device is able to communicate wirelessly with a remote monitoring software, and capture images without requiring components to exit an enclosed environment. Through MotionEye's user interface, the user can remotely access data and control of the system just by connecting to it using the device IP address. Through the use of a USB microscope this device can reach magnification levels of 10x to 250x the original sample size. Using the built-in features of MotionEye, the user can set specific time intervals for image capture and compile these images into time lapse videos. Additionally, temperature and humidity can be read, logged, and graphed on a ThingSpeak channel. The temperature and humidity can also be overlaid on the Motioneye interface so that the user can visually see environmental changes during the experiment. Because pH is determined via image processing, it requires an image to be taken in, as mentioned in Section 4.4.7. Therefore, it can only run when pictures are being taken in interval snapshots to be compiled into a time lapse.

There are some limitations to the device we have created, but these did not affect the device's ability to achieve the project objectives and can be addressed as part of future work. As mentioned in the previous paragraph, reading and overlaying the pH is only possible when taking images for a time lapse, meaning a user is restricted to only taking time lapses if they would like to observe pH. Additionally, because the time lapses are grouped by date, if a multiple-day time lapse is being run, the file path must also be changed to ensure the image processing program is pulling the most current picture in the time lapse Also, as mentioned in Section 4.3.1, the current needed to power the Raspberry Pi 3B+ for the maximum number of days would be a battery pack with 135,600 mAh, which is not commercially available for consumers. While the alternative design choice of the Pi Zero would allow for use of a commercially available battery pack, the Zero has limitations which were discovered through our experimentation which showed issues with large images, and inability to operate with high frame rates. Therefore, with the current design, a power wire would need to exit the incubator to be plugged into a wall outlet. Another limitation of the current device is the focus control. Currently, with the use of an USB microscope, the microscope must be focused manually before the experiment is begun. Our long-term experiments showed that there was no noticeable focus drift over the course of five days, but it could become an issue once the device is inside an incubator, or if the experiment is longer than five days. If focus drift were to occur, the user would have to manually refocus the microscope which would potentially require interrupting the experiment. An additional limitation of the current device design is that the housing is made of PLA plastic and is most accessible to those with 3D printers. There are services that will 3D print based off of designs which users could utilize to get the housing pieces if they do not have a 3D printer but depending on the service this could make the device slightly more expensive to create. Additionally, we were unable to test the effects of prolonged high humidity on the plastic housing. The housing also poses a similar limitation to the USB microscope in the fact that the component holders need to be manually positioned so if something were to become misaligned during an experiment the user would have to manually adjust. Another limitation of this device is the fact that in order to monitor multiple samples at once, such as in a 6-well plate, multiple image gathering sources are required. This can be achieved using multiple hollow rods on the same base, and multiple image gathering sources plugged into the same microcontroller. This increases the bulk of the device however and is cumbersome to the monitoring of more than two wells simultaneously. If multiple samples are to be studied in separate plates however, this device is ideal as its small size means that many can fit in an incubator at once. In terms of software interface, downloading the outlined software used for our device might be challenging for individuals with limited coding experience. Despite providing a step-by-step guide that describes how to go through the process in detail, sometimes the user has to troubleshoot some steps in order to get it right, which might not be easy without code experience.

Chapter 8: Conclusion and Recommendations

8.1 Conclusion

Overall, the results of our testing show that the final device design meets all of the project objectives. The final device shows full functionality and is capable of remotely monitoring a sample without external components and wirelessly captures images with overlaid data which is logged in a cloud server mitigating the need to take the sample out of the experimental environment. Additionally, this device is customizable, can have dimensions as small as 15 cm x 11cm x 11cm, and a step by step guide was made to show how to recreate the device making it ideal for use in any setting. This device meets all of these objectives and can be made for the low cost of $150-$200 dollars which represents a one-hundred-fold decrease in cost as compared to the gold standard. This project represents the creation of an effective biomonitoring device which can be made widely accessible for use in classrooms and laboratories all around the world without the issue of high cost or requiring a high-level understanding of coding or CAD modeling as the code and models are all open source.

8.2 Recommendations

An important next step for this project would be to test our device in a real laboratory environment in which the device can monitor organisms over a sustained period of time. An appropriate assessment would be monitoring a cell culture inside an incubator and evaluating the device performance in terms of image quality, sensor readouts and resistance to the incubator's environment on a long-term basis. Additionally, we recommend adding and testing a broader selection of sensors to compliment the compiled data. Some useful sensors to monitor biological organisms would be light, oxygen and carbon dioxide sensors. We also recommend developing an automated housing component with motorized movement that is able to withstand high humidity on a long-term basis. Automating and motorizing the device housing would even further mitigate the need to take the device out of the chamber for any adjustments needed. In order to address additional limitations of our device such as the potential for focus drift, we recommend that future work look at the integration of the 5MP Autofocus camera into the device. Future testing to create enough contrast to generate pH values from RGB values can be done by using another camera with a lower magnification such that contrast can be viewed between the solution and the sides of the plate. Also, a white paper can be placed opposite to the image sensor on top of the culture plate to create contrast between the color of the solution and the white background. Also, because of lack of lab access due to COVID-19, this calibration could not be done with cells, whereas the presence of cells could help determine contrast between the color of the solution. If color cannot be determined, despite creating further contrast between the solution and other items in the photo, it is recommended that the image be translated into grayscale in the image processing program, and the intensity of the greyscale is analyzed, as phenol red causes a change of the color of solution ranging from yellow to magenta as the solution becomes more basic. Therefore, on a grayscale this would be perceived as less gray to more gray, meaning the intensity of the grayscale could be correlated to pH. It is also recommended that the auto white balance for the pH program is turned off by modifying the camera configuration files, as opposed to through the interface. This can be attempted by adding "mmalcam_control_params -awb greyworld" (without quotation marks) to "Extra Motion Options" in the motioneye.conf file. Future development is also required to ensure the file path in the program doesn't need to be changed daily due to time lapses being separated by date. This can be attempted by changing rollover to manual in the motioneye.conf file by changing ffmpeg_timelapse_mode.

8.3 Personal Reflection

A wide range of engineering and scientific course materials were found to be incredibly useful and were directly applied in this project. Many of the principles used in our preliminary design phase were taught in Biomedical Engineering Design (BME 3300) and MQP Capstone Design (BME 4300). In completing our background research for this project, we drew on information learned in multiple BME and ECE courses including Cellular Engineering Lab (BME 3813), Biomedical Signals, Instruments, and Measurements (BME 2210), Biomedical Imaging (BME 4201), and Introduction to Communications and Networks (ECE 2305). Concepts from Electromagnetic Theory (ECE 2112) were used to hypothesize any interference between the wireless signal and the incubator structure. In order to determine important design requirements for our microprocessor, concepts from Embedded Computing in Engineering Design (ECE 2049) were referred to. When it came time to finalize the design elements and create our custom housing, the team heavily relied upon the use of Computer Aided Design Software. The knowledge of how to use this software came from a team member's experiences in high school. The process of 3D printing the components was learned both from this team member's experience and working with Rob Kirch to train us on the BME department machines. Intro to Material Science (ES 2001) was helpful in determining the necessary properties of the materials used in creation of the housing. In regard to the coding work necessary to install and run the various programs used in the creation and operation of this device Computer Science classes such as Intro to Program Design (CS 1101) and Introduction to Programming for Non-Majors (CS 1004) were useful. Biomedical Signals, Instruments, and Measurements (BME 2210) and Biomedical Sensors Laboratory (BME 3012) were also helpful for the sensing portion of this project.

References

1. J. L. Collins, B. van Knippenberg, K. Ding, and A. V. Kofman, “Time-Lapse Microscopy,”.
2. Straube, Tamara, & Claudia. (2016, March 24). How to do a Proper Cell Culture Quick Check. Retrieved October 10, 2019, from https://www.leica-microsystems.com/science-lab/how-to-do-a-proper-cell-culture-quick-check/.
3. Ryan, J. (2008). Introduction to Animal Cell Culture. *Corning Incorporated*.
4. Invitrogen. (n.d.). Cell Culture Basics. *Cell Culture Basics*.
5. Freshney, R. I. (2006). Basic Principles of Cell Culture. *Culture of Cells for Tissue Engineering.*
6. H. Nicholas, “Animals in research: C. elegans (roundworm),” The Conversation, 28-Feb-2020. [Online]. Available: https://theconversation.com/animals-in-research-c-elegans-roundworm-14163. [Accessed: 07-Mar-2020].
7. History of research on C. elegans and other free-living nematodes as model organisms. (2017). *WormBook*, 1–84. doi: 10.1895/wormbook.1.181.1
8. “CytoSMART Lux2,” *CytoSMART*. [Online]. Available: https://www.cytosmart.com/products/lux2. [Accessed: 12-May-2020].
9. Garcia, E., Shinde, R., Martinez, S., Kaushik, A., Chand, H. S., Nair, M., & Jayant, R. D. (2018, October 12). Cell-Line-Based Studies of Nanotechnology Drug-Delivery Systems: A Brief Review. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780128140338000126?via=ihub
10. Lovitt, C. J., Shelper, T. B., & Avery, V. M. (2014). Advanced cell culture techniques for cancer drug discovery. *Biology*, *3*(2), 345–367. https://doi.org/10.3390/biology3020345
11. Ambady, S. Cell Culture Procedures and Protocols (2010) (PDF)
12. Quantitative Neutotechnology Lab, WPI. Retreived May 14,2020, from https://wp.wpi.edu/qntl/resources/c-elegans/getting-started-with-c-elegans/
13. “Refraction.” [Online]. Available: https://www.merriam-webster.com/dictionary/refraction. [Accessed: 07-Mar-2020].
14. K. Lee, ‘Optics Review’, Worcester Polytechnic Institute, 2019.
15. Admin, “Snell's Law Formula: Definition and Examples,” 27-Dec-2018. [Online]. Available: https://byjus.com/snells-law-formula/. [Accessed: 07-Mar-2020].
16. “What is Microscopy?,” *The University of Edinburgh*, 27-Sep-1970. [Online]. Available: https://www.ed.ac.uk/clinical-sciences/edinburgh-imaging/for-patients-study-participants/tell-me-more-about-my-scan/what-is-microscopy. [Accessed: 07-Mar-2020].
17. K. Lee, ‘Light Microscopy’, Worcester Polytechnic Institute, 2019.
18. T. Schiesser, “Know Your Smartphone: A Guide to Camera Hardware,” 29-Jul-2014. [Online]. Available: https://www.techspot.com/guides/850-smartphone-camera-hardware/. [Accessed: 07-Mar-2020].
19. S. Taylor and C. Wilborn, “What are Digital Microscopes?,” 06-Mar-2020. [Online]. Available: https://www.wisegeek.com/what-are-digital-microscopes.htm. [Accessed: 07-Mar-2020].
20. “Education in Microscopy and Digital Imaging,” ZEISS Microscopy Online Campus | Live-Cell Imaging | Imaging Systems. [Online]. Available: http://zeiss-campus.magnet.fsu.edu/articles/livecellimaging/imagingsystems.html. [Accessed: 12-May-2020].

21. “Viva View FL Incubator Microscope,” Viva View FL | Incubator Microscope | Olympus Life Science. [Online]. Available: https://www.olympus-lifescience.com/en/microscopes/inverted/incubator/#!cms[focus]=cmsContent583. [Accessed: 12-May-2020].
22. Watanabe, W., Maruyama, R., Shimada, T., Yokoe, R., & Arimoto, H. (2018). Implementation of a Raspberry-Pi-based LED array microscope for multi-contrast images. *Biomedical Imaging and Sensing Conference*. doi: 10.1117/12.2319076
23. Aidukas T., Eckert R., Harvey A.R., Waller L., Konda P. C. (2018). Low-cost, sub-micron resolution, wide-field computational microscopy using opensource hardware. *Nature Scientific Reports*.
24. M. O’Neill, “Insecurity by Design: Todays IoT Device Security Problem,” Engineering, vol. 2, no. 1, pp. 48–49, 2016.
25. “An overview of HTTP,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. [Accessed: 18-May-2020].
26. Burnett, K., Edsinger, E., & Albrecht, D. R. (2018). Rapid and gentle hydrogel encapsulation of living organisms enables long-term microscopy over multiple hours. *Communications Biology*, *1*(1). doi: 10.1038/s42003-018-0079-6
27. Gonzalez, B. (2019, September 2). How Fast Should My Internet Be for Netflix and Other Similar Services? Retrieved October 10, 2019, from https://www.lifewire.com/internet-speed-requirements-for-movie-viewing-1847401.
28. Effect of heat on electronic devices | Technical Information | Apiste Corporation. (n.d.). Retrieved October 10, 2019, from http://www.apiste-global.com/enc/technology_enc/detail/id=1262
29. Effect of humidity on electronic devices | Technical Information | Apiste Corporation. (n.d.). Retrieved October 10, 2019, from http://www.apiste-global.com/enc/technology_enc/detail/id=1263.
30. 2019 Best Single Board Computers (Raspberry Pi Alternatives). (2019, October 10). Retrieved October 10, 2019, from https://all3dp.com/1/single-board-computer-raspberry-pi-alternative/.
31. RaspberryPI models comparison: Comparison tables. (2019, October 10). Retrieved October 10, 2019, from http://socialcompare.com/en/comparison/raspberrypi-models-comparison.
32. J. Gong, Y. Yuan, A. Ward, L. Kang, B. Zhang, Z. Wu, J. Peng, Z. Feng, J. Liu, and X. S. Xu, “The C. elegans Taste Receptor Homolog LITE-1 Is a Photoreceptor,” Cell, vol. 168, no. 1-2, p. 325, 2017.
33. GreatScott, “How to Waterproof Electronics”. 23-Dec.-2017.
34. U. Inc. International, “The SECOND Official Ultra-Ever Dry Video - Superhydrophobic coating - Repels almost any liquid!”. 23-Dec.-2017.
35. G. Davis, “Phenol Red - pH Indicator,” *Phenol Red - pH Indicator*, 01-Apr-2020. [Online]. Available: https://cran.r-project.org/web/packages/colorSpec/vignettes/phenolred.html. [Accessed: 18-May-2020].
36. A. Z. Jones, “The Visible Light Spectrum Contains the Colors We See,” *ThoughtCo*, 14-Feb-2020. [Online]. Available: https://www.thoughtco.com/the-visible-light-spectrum-2699036. [Accessed: 18-May-2020].

37. Adafruit Industries, “USB Microscope - 5MP interpolated 220x magnification / 8 LEDs,” adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/636. [Accessed: 18-May-2020].
38. S. H, D. M, V. P, Martha, D. L, and J. R, “Monoprice 60x, 250x Digital Microscope with Suction Cup Stand and Observation Pad reviews summary,” Monoprice, Inc., 02-Jan-2019. [Online]. Available: https://www.monoprice.com/Product?p_id=11613&fbclid=IwAR1x_qYWsgl7RHBh3jGaT7Gdtni7tuXp8kni06Amb-UoCF_ZeL4RLDoQADk. [Accessed: 18-May-2020].
39. Switz N.A., D’Ambrosio M.V., Fletcher D. A. (2014). Low-Cost Mobile Phone Microscopy with a Reversed Mobile Phone Camera Lens. PloS one. 9. e95330. 10.1371/journal.pone.0095330.
40. A. Industries, “Pimoroni Unicorn Hat - 8x8 RGB LED Shield for Raspberry Pi A+/B+,” *adafruit industries blog RSS*, 2015. [Online]. Available: https://www.adafruit.com/product/2288. [Accessed: 18-May-2020].
41. Adafruit Industries, “Pimoroni Unicorn HAT HD,” adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/3580. [Accessed: 18-May-2020].
42. Adafruit Industries, “White LED Backlight Module - Large 45mm x 86mm,” adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/1621. [Accessed: 16-May-2020].
43. Adafruit Industries, “Raspberry Pi Sense HAT,” adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/2738?src=raspberrypi. [Accessed: 18-May-2020].
44. https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf
45. https://datasheets.maximintegrated.com/en/ds/DS18S20.pdf
46. “Pricing and Licensing,” MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=edu. [Accessed: 18-May-2020].
47. “Carr,” McMaster. [Online]. Available: https://www.mcmaster.com/91771a120. [Accessed: 18-May-2020].
48. “Carr,” McMaster. [Online]. Available: https://www.mcmaster.com/92771a471. [Accessed: 18-May-2020].
49. S. Sonavane, “WiFi Tools for Raspberry Pi,” Raspberry Pi Adventures, 09-Sep-2012. [Online]. Available: https://rasspberrypi.wordpress.com/2012/09/09/wifi-tools-for-raspberry-pi/. [Accessed: 18-May-2020].
50. “How Can I Test My Oven Temperature For Accuracy?: Samsung Support AFRICA_EN,” Samsung africa_en, 20-Apr-2018. [Online]. Available: https://www.samsung.com/africa_en/support/home-appliances/how-can-i-test-my-oven-temperature-for-accuracy/. [Accessed: 18-May-2020].

Appendix A: How To Build Your Own Internet-Connected Biomonitoring Device

1.0 Materials:

1.1 Set Up and Programming

- Monitor
- USB Keyboard
- USB Mouse
- Wifi
- HDMI cord
- Micro SD card (16GB+)
- Micro SD to SD card adapter
- Micro USB Power cord (preferably with "on/off" switch)

1.2 Computer Unit

- Raspberry Pi 3B+
- Raspberry Pi 3B+ case
  - Top in smoke gray - Product 2244 from Adafruit
  - Bottom in smoke gray - Product 2256 from Adafruit

1.3 Monitoring Device

- CAD housing pieces
- Brass wing nuts - Part # 92771A471 from McMaster-Carr
- 2" Long Stainless Steel Phillips Flat Head Screws - Part # 91771A120 from McMaster-Carr
- USB Microscope
- UnicornHat 8x8 LED Array
  - 3 GPIO wires are necessary to connect the LED Array to the Pi

1.4 Optional

- Temperature and Humidity Sensor (DHT22/AM2302)
  - GPIO wires are necessary to connect the sensor to the Pi
- Adafruit Mini PiTFT 1.14 Display
- HDMI to VGA adapter in case monitor doesn't have HDMI port

2.0 Hardware Assembly Instructions:

2.1 Imaging Unit

1. Put Raspberry Pi inside its case
2. Insert micro SD card with metal pads facing the board
3. Plug power cord into PWR IN port
4. Plug HDMI cord in indicated port to connect it to the monitor

5. Plug in the mouse and keyboard in USB ports

2.2 Housing

1. Component holders and Rod:
    a. Download STL files from links found below
        i. Raspberry Pi 3B+ Holder:
            1. https://www.thingiverse.com/thing:4252999
        ii. Microscope Holder:
            1. https://www.thingiverse.com/thing:4253013
            2. https://www.thingiverse.com/thing:4253018
        iii. 100 mm Plate Holder:
            1. https://www.thingiverse.com/thing:4253022
        iv. 50 mm Plate Converter:
            1. https://www.thingiverse.com/thing:4253027
        v. 6-well Plate Holder:
            1. https://www.thingiverse.com/thing:4253047
        vi. Rod:
            1. https://www.thingiverse.com/thing:4253063
    b. If you need to edit any of the components the original Solidworks files can all be found here:
        i. https://www.thingiverse.com/thing:4253075/files
    c. Once you have downloaded the STL files they are ready to be printed. The parts can be printed in basic PLA plastic. In terms of printing you have 2 options:
        i. If you own or have access to a 3D printer such as a MakerBot you can print these files yourselves.
        ii. If you do not have access to a 3D printer, you can have the parts printed through a service such as can be found in this article: https://www.allthat3d.com/best-3d-service-online/
    d. If neither of these options for acquiring the housing elements work for you, the housing can be recreated out of other materials/objects found around the house as necessary.
2. Screws and Nuts:
    a. The screws which are linked in the supply section of this document are the exact ones which the STL files have been designed to fit.
        i. The nuts which are linked in the supply section of this document are the nuts we recommend to pair with the screws which are linked in the supply section of this document.
    b. If you plan to use other screws the CAD files will need to be edited according to the diameter of the screw head and tail of the new screws.
        i. Additionally, you will need to find new wingnuts to work with your new screws.

3. Assembly:
    a. The two halves of the rod are designed to be printed separately and then snapped together. If there is wiggle room between the peg and hole, the rod can be glued together.
        i. If the standard rod height is not long enough for your purposes, a rod extender is included in the files which can be used to add extra height.
    b. Once the rod is assembled, the additional component holders can be attached using the screw and wingnuts.
        i. Each component holder can be attached at the desired height by feeding the screw through the hole in the holder and then the hole in the rod.
            1. The wingnut is screwed on to the back side of the rod to hold the “shelf” in place and can be loosened to adjust the height of said “shelf”.
        ii. For the Microscope holder, the same screws and wingnuts are used to attach the two halves of the holder to each other around the microscope.
            1. In order to line the microscope up to the hole in the plate holders and view the sample,two additional wingnuts can be used to adjust the microscope position within the x-y plane.
4. Tips:
    a. If the middle where the pieces of the rod meet expands and the channel is too wide you can squeeze the sides while tightening the nut in place to ensure that the shelf won’t fall
    b. The shelves can be placed at various heights, but if using a usb microscope it is recommended to place it as close to the base as possible for maximum balance of the device

2.3 Lighting Setup without Temperature + Humidity Sensors

1. Place UnicornHat 8x8 LED Array on Raspberry Pi header
    a. Make sure pins are properly aligned to the openings on the LED array
    b. Push down to ensure none of the header pins can be seen
2. Snap the lid of the case onto the Raspberry Pi case
    a. Make sure the notches on the lid are aligned with the notches on the case

3.0 Software Set Up Instructions:

3.1 Operating System

For the purposes of this device, download Buster with Desktop and Recommended Software onto the SD card

1. Insert the SD card into your computer
2. Go to https://www.balena.io/etcher/ and download the BalenaEtcher application to your computer by clicking on the download button for your corresponding OS (i.e macOS, Windows 10, ect

3. Go to https://www.raspberrypi.org/downloads/raspbian/ and download “Raspbian Buster with Desktop and Recommended Software” as a ZIP file to your computer
4. Expand the downloaded Buster zipped file (By just double clicking on it on macOS). It should be an .img file after expansion.

   **(this step is optional, the file can be selected as a zip file from the BalenaEtcher application)
5. Select the newly expanded (.img) file from the BalenaEtcher application. More specifically, from where it says “Select Image”
6. Make sure the drive selected is the drive where the SD card reader is (Apple SD card reader Media for macOS)
7. Click on “Flash!”
8. Eject the SD card when done if it didn’t do it automatically
9. To double check what version/release of Raspbian you downloaded, you can insert the SD card in the Raspberry Pi, get it started, open terminal and type
   cat /etc/os-release

3.2 Internet Connectivity - Home Network

1. Insert Micro SD card into the Raspberry Pi
2. Connect the Pi to a power source, an HDMI monitor, a USB keyboard and a USB mouse.
3. Turn the Pi on
4. Complete initial configuration about time zone and language
5. Create and confirm new Password for Raspberry Pi
6. The Pi will automatically look for available WiFis, select and enter the password for the desired WiFi connection
7. Restart the Raspberry Pi

3.3 Internet Connectivity - WPI Wireless Network

These steps were based on the WPIHub page for connecting a Raspberry Pi to the WPI Wireless network connection: https://its.wpi.edu/article/361/connect-to-wpi-wireless-using-a-raspberry-pi-with-gui

1. First, register both Ethernet (eth0) and Wireless (wlan0) in the WPI network Registration here using your own computer: https://netreg.wpi.edu/
   a. Log in with WPI credentials
   b. Register the device’s Ethernet and Wireless
   c. To find their respective MAC Addresses, type
      i. > ifconfig
      ii. in the Terminal
2. Now, connect the Raspberry Pi via ethernet
3. Run the following commands on Terminal
      i. > sudo apt update
      ii. > sudo apt install network-manager-gnome

        iii. > sudo systemctl enable NetworkManager
        iv. > sudo systemctl disable dhcpcd
4. Reboot the Raspberry Pi and log back in
5. There are now two icons on the status bar for the network. Hover over the rightmost icon (a tool tip will display about dhcpcd). Right click and remove the icon from the panel.
6. Disconnect the ethernet
7. Using the other Network Manager icon, connect to WPI-Open
8. Visit the Wireless Connection page: https://guest.wpi.edu/guest/start.php?_browser=1
9. Select the "Setup My WPI Wifi"
10. Select "Unknown" from the dropdown menu
11. Select "Sign IN" and use your WPI credentials to log in
12. Once logged in, create a new certificate
    a. Enter Wireless MAC address including the colons and click "create
    b. create a new password for the certificate
    c. click "submit" and a certificate will be generated. The certificate will be called something like "username@wpi.edu.p12"
13. Download the "WPI NetOps Wireless CA" certificate
14. Go back to the Network Manager icon and connect to WPI-Wireless by selecting the following
    a. Authentication: TLS
    b. Domain: (leave blank)
    c. Identity: yourusername@wpi.edu
    d. user certificate: username@wpi.edu.p12 (the one you just created)
    e. CA certificate: "WPI NetOps Wireless CA"
        i. 3905b39901f24ad1293bee10ec9a67703b61834e.cer
    f. Private Key: username@wpi.edu.p12 file should auto-populate
    g. Private Key Password: use the password created for the certificate
15. click "save"

3.4 MotionEye

These steps were based on the GitHub page for installing motioneye on Raspbian:
https://github.com/ccrisan/motioneye/wiki/Install-On-Raspbian

1. Turn the Raspberry Pi on
2. Open Terminal
3. Enable the Raspberry Pi CSI Camera Module by typing

```
> sudo raspi-config
```

    Select "5 Interfacing Options", then "P1 Camera", then "yes". Select "Finish" and the "Yes" for reboot

4. Run an update:

```
> sudo apt-get update
```

5. Install ffmpeg and other motion dependencies:

```
> sudo apt-get install ffmpeg libmariadb3 libpq5 libmicrohttpd12
```
Type “y” and press enter

6. Install motion:
   ```
   > sudo wget https://github.com/Motion-
   Project/motion/releases/download/release-4.2.2/pi_buster_motion_4.2.2-
   1_armhf.deb
   > sudo dpkg -i pi_buster_motion_4.2.2-1_armhf.deb
   ```
7. Install the dependencies from repositories:
   ```
   > sudo apt-get install python-pip python-dev libssl-dev    libcurl4-openssl-dev
   libjpeg-dev libz-dev
   ```
   Type “y” and press enter
8. Install motioneye, which will automatically pull Python dependencies (tornado, jinja2, pillow and pycurl)
   ```
       > sudo pip install motioneye
   ```
9. prepare the configuration directory:
   ```
       > sudo mkdir -p /etc/motioneye
   > sudo cp /usr/local/share/motioneye/extra/motioneye.conf.sample
   /etc/motioneye/motioneye.conf
   ```
10. Prepare the media directory:
    ```
        > sudo mkdir -p /var/lib/motioneye
    ```
11. Add an init script, configure it to run at startup and start the MotionEye server:
    ```
    > sudo cp /usr/local/share/motioneye/extra/motioneye.systemd-unit-local
    /etc/systemd/system/motioneye.service
        > sudo systemctl daemon-reload
        > sudo systemctl enable motioneye
        > sudo systemctl start motioneye
    ```
12. To upgrade to the newest version of MotionEye server:
    ```
    >  sudo pip install motioneye --upgrade
    > sudo systemctl restart motioneye
    ```
13. To get the IP address, type
    ```
        > hostname -I
    ```
14. Go to any browser type the following URL
    ```
        > http://(your IP address):8765/
    ```
    For example: If your IP address is 10.0.0.18, your URL should look like this http://10.0.0.18:8765/
15. Use “admin” and no password as the initial credentials when asked for them
16. Add and prompt yout camera based on the type of camera or imaging device being used

3.6 UnicornHat LED Array

1. Enter following line of code into the terminal:
   ```
   >curl https://get.pimoroni.com/unicornhat | bash
   ```

2. Press y to continue
3. During installation, you'll be asked if you want to install Flask for Unicorn Paint. Answer no
4. Once the install is done, the script will ask if you want to download example code, answer yes.
5. Type the following command into the terminal to navigate to the code directory using the following terminal command:
    a. cd ~/Pimoroni/unicornhat/examples/
6. To verify the scripts work with your UnicornHat, type the following command into the terminal
    a. sudo python ./simple.py
7. Use the following command to edit the simple.py lighting script
    a. >sudo nano simple.py
8. Change your code accordingly to set up configurations for dark field and bright field lighting:
    a.

```
#!/usr/bin/env python
import time
import unicornhat as unicorn

print("""Simple

Turns pixels on for dark field microscopy

If you're using a Unicorn HAT and only half the screen lights up,
edit this example and  change 'unicorn.AUTO' to 'unicorn.HAT' below.
""")

unicorn.set_layout(unicorn.HAT)
unicorn.rotation(0)
unicorn.brightness(0.5)
width,height=unicorn.get_shape()

## for bright field
""" for y in range(height):
   for x in range(width):
      unicorn.set_pixel(x,y,225,225,225)
      unicorn.show() """
```

```
for y in range(height):
    unicorn.set_pixel(0,y,255,225,225)
    unicorn.set_pixel(7,y,225,225,225)
    unicorn.show()

for x in range(width):
    unicorn.set_pixel(x,0,225,255,225)
    unicorn.set_pixel(7,y,225,225,225)
    unicorn.show()

#change for time duration of experiment in seconds
time.sleep(1800)
```

9. Press Ctrl+X to exit, Y to save, and then enter to confirm file name
10. Go back to the home directory by using the following terminal command
    a. >cd ~/

3.5 Port Forwarding:

This is used if the wifi your Pi is connected to is a network (such as home networks) which doesn't allow outside access to IPs. If using WPI wireless or a wifi with similar security this wouldn't be necessary.

1. Open a browser on your computer.
2. Log into your wifi router using either 192.168.1.1 or 192.168.0.1
    a. More instructions about logging into your router can be found on your router or through your internet service provider
3. Once logged into your router you should be able to find a section of information labelled " Port Forwarding". Within this section you will make a new "rule".
    a. Destination port: this is the local IP address, like **192.168.1.xxx**, and forward port number **8765** for motioneye
    b. Entry port: pick a number-- security measures suggest using a different port number. For example, **5678**
    c. Protocol: Default TCP/UDP
    d. Click Add or Save
4. Now, you find your external IP address
    a. You can use a web browser to whatismyip.com or similar. Should say something like: "My Public IPv4 is: 188.60.149.109"
5. You now have an external IP to use to view MotionEye on a different network.
6. In order to do this you have to go to the new IP for example **188.60.149.109**, with the port number you created, for example **5678**. This would read as **188.60.149.109:5678**.
7. This should allow you to connect to MotionEye and view your sample from wherever you are.

3.6 SSH Setup

1. Connect your Pi to a monitor
2. Go to the raspberry pi symbol in the upper left corner
    a. Open the preferences menu
    b. Select configuration
    c. Select interfaces
    d. Enable SSH
3. Now you will have to download PuTTY on the computer you wish to use to access the Pi
4. Once PuTTY is downloaded you can open it and put in the IP address of your Pi and hit open
5. Once PuTTY opens a command window you can log into your Pi using the hostname and password you created during setup

4.0 Time Lapse Set Up

4.1 Still Images Configuration

1. Open MotionEye in web browser
2. log in
3. Select the desired camera to take the pictures for the time lapse
4. Set image quality to preferred setting
    a. e.g. "100%"
5. Set capture mode to "interval snapshots"
6. Select the appropriate time interval between each picture taken
    a. e.g. "30" secs per snapshot
7. Click on the orange button at the top of the menu that says "Apply"
8. Let device take pictures for however much long as needed

4.2 Time Lapse

1. Open MotionEye in web browser
2. Log in
3. Select the image icon on the top right corner of the camera view in MotionEye
4. Select time lapse

5.0 Optional Additions

5.1 Light Wiring with Temperature + Humidity Sensor

1. Plug in male end of 3 GPIO wires to Raspberry Pi Header pins 2, 6, and 16 (GPIO 23)
2. Thread these wires through the top slit in the Raspberry Pi 3B+ case's lid
    a. Make sure the shape of the lid fits the orientation of the case
3. Take another 3 GPIO wires and cut off the female end using wire cutters
4. Strip this end to ¼ "

5. Plug the male end of these GPIO wires into Raspberry Pi Header Pins 4, 12 (GPIO 18), and 14
6. Insert the stripped end of these wires through the underside of the 3 holes at the bottom of the LED Array
    a. Pin 4 → leftmost hole, pin 12 → middle hold, pin 14 → rightmost hole
    b. These holes should be facing the USB and ethernet ports on the Raspberry Pi
7. Solder each of the wires in place by placing solder on the LED-side of the holes on the LED array
8. Coil GPIO wires under LED array such that they fit in the case
9. Secure coiled wires with ziptie
10. Lower case lid onto threaded wires and secure case lid to case
    a. You should hear it snap in place
11. Plug in temperature sensor on breadboard
12. Connect rightmost pin and pin to its left facing the back of the sensor with a 4.7 kOhm or 10 kOhm resistor
13. Connect Raspberry Pi Header pin 2 → rightmost temperature sensor pin, Raspberry Pi Header pin 6 → leftmost temperature sensor pin, Raspberry Pi Header pin 16 → 2nd from rightmost temperature sensor pin

5.2 Temperature + Humidity Sensor Software

1. Enter the following commands into the terminal to update packages:
    ```
    >sudo apt-get update
    >sudo apt-get install build-essential python-dev python-openssl git
    ```

2. Install the directories containing the temperature and humidity readings code by typing the following commands into the terminal
    ```
    >git clone https://github.com/adafruit/Adafruit_Python_DHT.git && cd
    Adafruit_Python_DHT
    >sudo python setup.py install
    ```

3. Navigate to the directory by typing the following command into the terminal:
    a. `>cd ~/Adafruit_Python_DHT/examples`
4. Test you can get readouts by entering the following command into the terminal:
    a. `>sudo ./AdafruitDHT.py 2302 23`
5. Edit the temperature by typing the following command into the terminal:
    a. `> sudo nano AdafruitDHT.py`
6. Change your code accordingly to remove the need for user input. This is so the temperature can be read without the user needing to prompt the script:
    ```
    #!/usr/bin/python
    # Copyright (c) 2014 Adafruit Industries
    # Author: Tony DiCola
    ```

```
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE
# SOFTWARE.
import sys
import time
import Adafruit_DHT


# Parse command line parameters.
#sensor_args = { '11': Adafruit_DHT.DHT11,
#               '22': Adafruit_DHT.DHT22,
#               '2302': Adafruit_DHT.AM2302 }
#if len(sys.argv) == 3 and sys.argv[1] in sensor_args:
#sensor_args[sys.argv[1]]
#sys.argv[2]

#else:
#    print('Usage: sudo ./Adafruit_DHT.py [11|22|2302] <GPIO pin number>')
#    print('Example: sudo ./Adafruit_DHT.py 2302 4 - Read from an AM2302
connected to GPIO pin #4')
#    sys.exit(1)
sensor = Adafruit_DHT.AM2302
pin = 23
# Try to grab a sensor reading.  Use the read_retry method which will retry up
# to 15 times to get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

# Un-comment the line below to convert the temperature to Fahrenheit.
temperature = temperature * 9/5.0 + 32

# Note that sometimes you won't get a reading and
# the results will be null (because Linux can't
# guarantee the timing of calls to read the sensor).
# If this happens try again!

if humidity is not None and temperature is not None:
      print("Temp={0:0.1f}F;Hum={1:0.1f};".format(temperature, humidity))
else:
      print('Failed to get reading. Try again!')
      sys.exit(1)
```

7. Press CTRL+X, Y, and then enter to save the file under its original name
8. Make the script executable by typing

   ```
   > chmod +x ~/Adafruit_Python_DHT/examples/AdafruitDHT.py
   ```

9. Go back to the home directory by using the following terminal command
    a. >cd ~/

5.3 OpenCV Installation and RGB Reading Code for pH Sensing

1. Update the pi using the following command in the terminal:

   ```
   >sudo apt update
   >sudo apt upgrade
   ```

2. Install dependent packages by running the following commands into the terminal:

   ```
   >sudo apt install cmake build-essential pkg-config git
   >sudo apt install libjpeg-dev libtiff-dev libjasper-dev libpng-dev libwebp-dev
   libopenexr-dev
   >sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
   libxvidcore-dev libx264-dev libdc1394-22-dev libgstreamer-plugins-base1.0-dev
   libgstreamer1.0-dev
   >sudo apt install libgtk-3-dev libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
   ```

```
>sudo apt install libatlas-base-dev liblapacke-dev gfortran
>sudo apt install libhdf5-dev libhdf5-103
>sudo apt install python3-dev python3-pip python3-numpy
```

3. Modify the swap file configuration by typing the following command into the terminal:
   a. >sudo nano /etc/dphys-swapfile
4. In this file find CONF_SWAPSIZE=*your number* and take note of this number
   a. Change it to CONF_SWAPSIZE=2048
5. Restart the swap file by typing the following command into the terminal:
   a. >sudo systemctl restart dphys-swapfile
6. Clone the opencv repositories to your Pi by typing the following commands into the terminal:

   ```
   >git clone https://github.com/opencv/opencv.git
   >git clone https://github.com/opencv/opencv_contrib.git
   ```

7. Create and navigate to the directory where the OpenCV will be installed by typing the following commands into the terminal

   ```
   >mkdir ~/opencv/build
   >cd ~/opencv/build
   ```

8. Prepare the build files to start the OpenCV install by typing the following command into the terminal

   ```
   >cmake -D CMAKE_BUILD_TYPE=RELEASE \
      -D CMAKE_INSTALL_PREFIX=/usr/local \
      -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
      -D ENABLE_NEON=ON \
      -D ENABLE_VFPV3=ON \
      -D BUILD_TESTS=OFF \
      -D INSTALL_PYTHON_EXAMPLES=OFF \
      -D OPENCV_ENABLE_NONFREE=ON \
      -D CMAKE_SHARED_LINKER_FLAGS=-latomic \
   ```

9. Compile the build files by typing this command into the terminal:

   ```
   >make -j$(nproc)
   ```

10. Install OpenCV by typing the following command into the terminal:

    ```
    >sudo make install
    ```

11. Link the dependent libraries to the install file:

    ```
    >sudo ldconfig
    ```

12. Change the swap file size back to its original size by typing the following command into the terminal:

    ```
    >sudo nano /etc/dphys-swapfile
    ```

13. In this file find CONF_SWAPSIZE=2048 and change it to CONF_SWAPSIZE=*your original number*
14. Restart the swap file by typing the following command into the terminal:

a. >sudo systemctl restart dphys-swapfile

15. Test OpenCV is installed by typing the following commands in the terminal:

```
>python3
>import cv2
>cv2.__version__
```

16. Your result from these commands should be a version number, such as ‘4.1.2’
17. Navigate back to the home directory by typing the following command into the terminal:

```
>cd ~/
```

18. Create a directory for pH reading code and navigate to it by typing the following commands into the terminal:

```
>Mkdir ~/pHreading
>cd ~/pHreading
```

19. Create a script to read pH based on RGB values by typing the following command into the terminal

```
>sudo nano RGB2pH.py
```

20. Edit the script to reflect the following:

a. Note: this script has to be calibrated with cells or reconfigured for further contrast. See recommendations (Section 8.2) and Discussion (Section 7) for more information

```
import cv2
import numpy as np
import os
import glob


LatestFile = max(glob.iglob('/var/lib/motioneye/Camera1/2020-05-18/*jpg'),
key=os.path.basename)


img = cv2.imread('LatestFile',cv2.IMREAD_COLOR)
height = img.shape[0]
width = img.shape[1]

allBGR = []
for x in range(height):
    for y in range(width):
        curBGR = img[x,y]
        allBGR.append(curBGR)

b,g,r = np.mean(allBGR, axis = 0)
```

```
print(b)
print(g)
print(r)
```

5.3 Data Overlay on Images

1. Ambient Temperature
   a. Create a new file
      ```
      > sudo nano /usr/local/bin/overlaytemperature
      ```
   b. Type the contents below in the new file created
      ```
      > CAM_CONFIG_FILE="/etc/motioneye/camera-1.conf"
      >CAM_NAME="
      >if [ -f $CAM_CONFIG_FILE ]; then
          > CAM_NAME=`grep text_left $CAM_CONFIG_FILE | cut -f2 -d' '`
      >fi

      >TEMP=`/home/pi/Adafruit_Python_DHT/examples/AdafruitDHT.py | cut -f1,2 -d';'`
      >curl "http://localhost:7999/1/config/set?text_left=$TEMP\n$CAM_NAME"
      ```
2. Save under original file name by hitting CTRL+X, Y, and enter

5.4 Data Logging/Storage

Once the desired sensor is attached to the Pi you can create code to run the data logging automatically (this requires access to a MathWorks Account)

1. Go to ThingSpeak.com
2. Log in using your MathWorks Account credentials
3. Go to the Channels tab and create a new channel
4. Set up two fields named Temperature and Humidity and name the Channel as you wish
5. Save the channel
6. Go to the sharing tab and turn on “Share channel view with everyone” if you would like the data to be visible without logging in
7. Go to the API keys tab and find your Write API key for the next steps

These steps can either be completed while the Pi is plugged into a monitor or by using SSH

8. Once you have access to a command window on the Pi you must create a file storage location by typing
   ```
   >mkdir datalogging
   ```
9. You can now switch into this folder by typing
   ```
   >cd ~/datalogging
   ```
10. To create your program type
    ```
    >sudo nano cloudlog.py
    ```
11. This will open a file named cloudlog.py which you can now type this code into

```
        >import sys
        >import urllib2
        >import Adafruit_DHT
        >from time import sleep
        >
        >myAPI = '8JDYC1LUM8FO80XW'
        >
        >baseURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPI
        >
        >def AM2302_data():
                sensor = Adafruit_DHT.AM2302
                pin = 23
                humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
                temperature = temperature *9/5.0 + 32
                return humidity, temperature


        >while True:
        >       try:
        >               humidity, temperature = AM2302_data
        >
        >               # If Reading is valid
        >               if isinstance(humidity, float) and isinstance(temperature, float):
        >
        >                       # Format to two decimal places (optional)
        >                       tempC = '%.2f' % temperature
        >                       humidity = '%.2f' % humidity
        >
        >                       # Send the data to thingspeak
        >                       with urllib2.urlopen(baseURL + '&field1=%s&field2=%s' %
(tempC, humidity)) as response:
        >                               html = response.read()
        >                       #print conn.read()
        >
        >                       # Close the connection
        >                       #conn.close()
        >               else:
        >                       print('Error')
        >
        >               sleep(10)
        >               # Build in a delay between loggings. For ~every minute, use a delay
of 60s.
        >               # Note that this may not be exact if the other commands take time.
You can use
        >               # a more precise timing system, like monitoring a system timer.
        >       except:
        >               print('Data Logged')
        >       sleep(60)
```

a. Within this code you should replace the string in myAPI with the Write API Key of the channel you set up on ThingSpeak

12. Save you file by hitting ctrl x to exit and y to save
13. You can now start this program by typing

    >sudo python ./cloudlog.py

14. If you wish to end the program running you can hit ctrl c
15. You can go to your ThingSpeak channel to view your data logged on graphs in real time

5.5 Mini Display Setup

The screen will be set up to display the following stats about your Raspberry Pi:

IP address
CPU load
Disk space
CPU temperature

Based on: https://learn.adafruit.com/adafruit-mini-pitft-135x240-color-tft-add-on-for-raspberry-pi?view=all

Python setup

1. Install Adafruit_Blinka library
    a. run necessary updates:
        > sudo apt-get update
        > sudo apt-get upgrade
        > reboot
        > sudo pip3 install --upgrade setup tools
    b. Enable I2C
        https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c
        > sudo apt-get install -y python-smbus
        > sudo apt-get install -y i2c-tools
        > sudo raspi-config
            Go to Interfacing options
            the I2C
            Enable
            Yes
            Yes
            Finish
        >sudo reboot
    c. Enable SPI
        https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-spi
        > sudo raspi-config
            Interfacing Options
            SPI
            Yes
            Yes
            Finish
    d. Enable second SPI
        > sudo nano /boot/config.txt
        add the following line at the bottom
        Reboot

e. Install Python Libraries and Adafruit Blinka

   ```
   > pip3 install RPI.GPIO
   > pip3 install adafruit-blinka
   ```

2. Run the following commands at the Terminal

   ```
   > sudo pip3 install adafruit-circuitpython-rgb-display
   > sudo pip3 install --upgrade --force-reinstall spidev
   > sudo apt-get install ttf-dejavu
   > sudo apt-get install python3-pil
   > sudo apt-get install python3-numpy
   ```

3. OPTIONAL: If so desired, go to the link below to download the “Quickstart Button Test” demo to make sure it works

   https://learn.adafruit.com/adafruit-mini-pitft-135x240-color-tft-add-on-for-raspberry-pi?view=all

   then, run the following command

   ```
   > sudo python3 rgb_display_minipitfttest.py
   ```

   Once its running, push the top button to get a red display, the bottom button to get a blue display, and both buttons to get a green display

Stats Display

1. Go to the link below to download and save the “stats.py” code in your home directory on your raspberry Pi

   https://learn.adafruit.com/adafruit-mini-pitft-135x240-color-tft-add-on-for-raspberry-pi?view=all

2. Run the script by typing

   ```
   > python3 stats.py
   ```

Running Stats on Boot

1. Run the following command

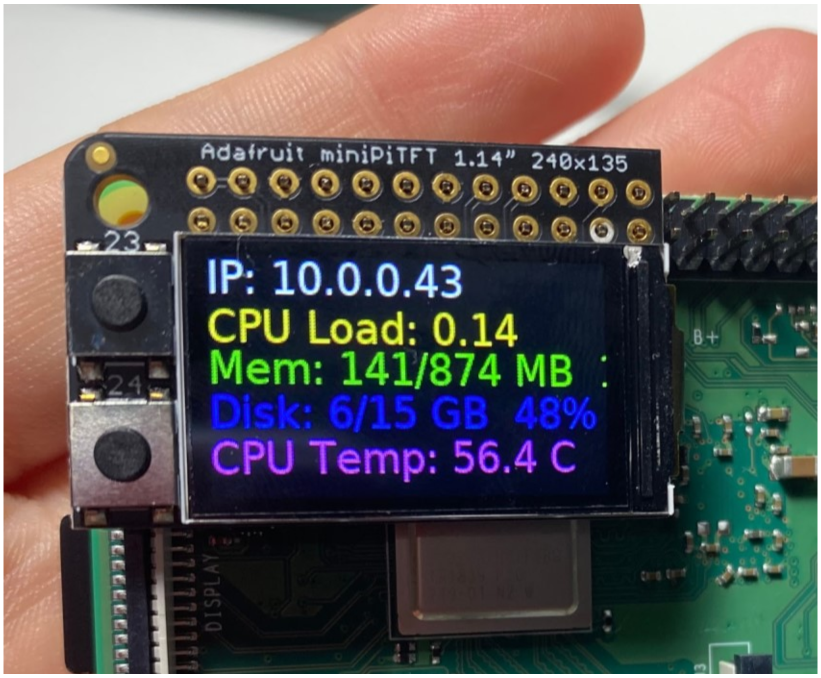
   ```
   > sudo nano /etc/rc.local
   ```

2. Add the following line own its own command line right before the command line that says “exit 0”

   ```
   > sudo python3 /home/pi/stats.py &
   ```

3. Save (CTRL O) and Exit (CTRL X)
4. Reboot to make sure it works

It Should look like this:



```
IP: 10.0.0.43
CPU Load: 0.14
Mem: 141/874 MB
Disk: 6/15 GB  48%
CPU Temp: 56.4 C
```

If the screen is taken off, there is no need to reconfigure, just plug it back on and it should work.

6.0 Troubleshooting

6.1 Internet Connection Troubleshooting

1. Computer Parameters Needed:
    a. wpa_supplicant.conf in wpa_supplicant folder
    b. Network name (ssid)
    c. Network password
    d. Whether it's a hidden network or not
2. Troubleshooting Steps:
    a. If the folder wpa_supplicant folder is missing, you must create it and create the wpa_supplicant.conf file within it by pressing the new folder and new file icon in the upper left hand corner
    b. Type sudo raspi-config to enable wireless networking
    c. Open in wpa_supplicant.conf in nano by typing
    ```
    sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
    ```
    d. Ensure that the following lines of code are at the top of the file:
    ```
    ctrl_interface=DIR=/var/run/wpa_supplicant
    GROUP=netdev
    update_config=1
    country=<Insert country code here>
    ```
    e. Go to the bottom of the file and add the following:
    ```
    network={
        ssid="Network_Name"
        psk="networkPassword"
    ```

```
}
```

f. If the network is a hidden network, add `scan_ssid=1` to the network definition in the brackets

g. If the network has no password, add `key_mgmt=NONE` to the network definition in the brackets

6.2 Timelapse not Compiling

This is possibly due to the lack of RAM in the system to compile a time lapse. In this case, MobaXterm can be downloaded to interface with the Pi and access its files

1. Download MobaXterm https://mobaxterm.mobatek.net/download.html and follow the instructions
2. Once install is complete start MobaXterm
3. Select "Start Local Terminal"
4. Type ssh pi@*your IP address*, assuming SSH is enabled
   a. A file system should appear in a sidebar on the right
5. In the file system, click the button labelled "Parent Directory" twice to navigate to the overarching directory
   a. "Parent Directory" label will appear after hovering over the button
   b. File path name after one click should be /home/, file path name after two clicks should be /
6. In the overarching directory double click on var, then bin, then motioneye, then the camera name according to your Motioneye UI, and then date of timelapse
7. Select all the files in the folder by pressing the topmost file, shift, and then the bottom most file
8. Select the "Download" button, which is to the immediate right of the "Parent Directory" button
   a. Once again, the label of the button will show up when hovering above it

7.0 Alternative Supply Options*:

- Microprocessor
  - Raspberry Pi 3B (and corresponding case and power cord)
  - Raspberry Pi 3A+ (and corresponding case and power cord)
  - Raspberry Pi 4B (and corresponding case and power cord)
  - Raspberry Pi Zero WH (and corresponding case and power cord)
- Image Gathering Source
  - Raspberry Pi V2.1 Camera (macroscopic view only)

- For Raspberry Pi Zero WH:
  - Raspberry Pi Zero v1.3 Camera Cable
- 5MP Raspberry Pi Autofocus Camera (macroscopic view only)
- Light Source
  - UnicornHat 16 x 16 LED Array

*If any alternative design components are selected, the housing may need to be altered to work with the new components