Mushroom Nursery

An autonomous environment for enhanced primordia formation and fruit body development of mushroom cultures.

University of Central Florida
Department of Electrical Engineering and Computer Science
EEL 4914, Senior Design 2
Dr. S. M. Richie, Fall 2019

Group 5
John Farriss, CpE
Mardochee Cajuste, CpE
David Booth, CpE

Table of Contents

1. Executive Summary

This document details all the requirements, specifications, and design characteristics for the Mushroom Nursery. This project has closely followed the University of Central Florida College of Electrical Engineering and Computer Science guidelines and the Accreditation Board for Engineering and Technologys' (ABET) requirements for accredited engineering programs.

Technology today impacts every field and can have very specific applications. The Mushroom Nursery targets a niche market in the horticulture realm. Humans have always been known to cultivate things that grow out of the ground. Typically humans rely on farmers to grow and harvest the food, but there is an abundance of current technology that allows humans to grow food themselves in much smaller scales. This has many benefits as the produce grown can be used for more than just consumption. The produce also has scientific and medical applications as well.

Emerging technology gets better and better and keeps raising the bar for future technology. This technology also creates a need for technology that accomplishes much more specific goals. There is an abundance of technology that allows humans to grow a specific type of plant in their house but we noticed that there is a lack of technology for other produce, such as mushrooms. This project has been designed to provide many of the same advantages that plant based technologies provide, but targeted for mushroom growing instead.

The Mushroom Nursery has been designed to fit in small areas in most housing environments. It has also been designed to require very little external support with the only input requirements being an outlet to plug into and manual refilling of the water container. The user interface has been designed with current standards in mind, providing easy-to-use controls to anyone with access to the internet.

Because of the various requirements and component specifications a mixture of various organization and management styles have been utilized. These styles have been scheduled and implemented to allow testing between different components that rely on each other.

2. Project Description

The following section is a detailed description of the project's features and their application, summaries of the research related to the desired components that comprise the project as well as a description of the motivation behind them.

2.1 Motivation

It can be difficult to find certain species of mushrooms in your local supermarket, especially if they are not local to the area or are not in season. Another reason that you may not be able to easily find a certain species of mushroom is that they are too expensive. The Mushroom Nursery aims to provide a solution to these problems. There is an abundant source of technology that allows the average person to grow plants in areas that are typically inhospitable to that species. However, there are very few products that provide this same solution for mushrooms.

Certain species of mushrooms also contain some important properties that are used for medicinal purposes. Scientists need to be able to grow and cultivate these mushrooms in a controlled environment in order to ensure that nothing has been contaminated. With the Mushroom Nursery, scientists would be able to improve their research by optimizing the process of collecting specific mushroom material. The Mushroom Nursery will also provide feedback so that the user can read through the statistics and find out what went right or wrong.

The Mushroom Nursery also aims to provide a hands-off experience for the user, as to eliminate any potential human error in the process. The Nursery will automate daily processes such as hydrating the mushrooms, controlling humidity, and providing fresh air to the environment. The aim is also to reduce the risk of contamination from the external environment, which is a large problem when attempting to grow mushrooms.

2.2 Project Illustration

Figure 0 is a visual rendering of our housing and a mockup of our web app to illustrate the idea to the reader, note though that this should be treated as a purely artistic rendering and that the final product differs slightly.

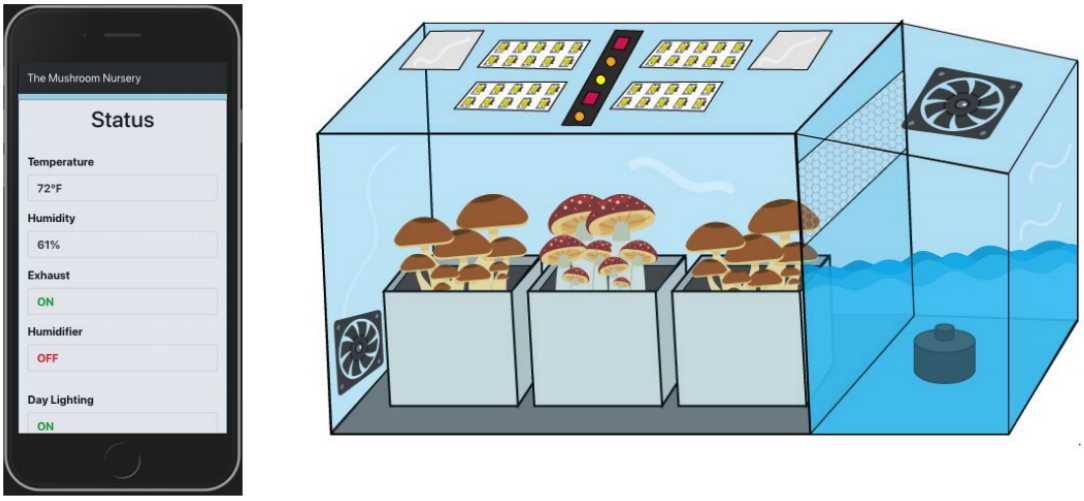

Figure 0: Artistic rendering of Project

2.3 Suitable Operating Conditions

The recommended operating conditions are an important part of the sustainability and operation of the Mushroom Nursery. Placing the Nursery in conditions that are not recommended could put a lot of strain on the system and cause failure in some components. Even if there is no component failure, placing the Nursery in unfavorable conditions could produce subpar results. Because the goal of the Nursery is to provide the most ideal environment for the mushrooms, we highly suggest the recommended operating conditions are taken seriously.

The ideal operating conditions for the Nursery are in a room temperature enclosed area with at least some airflow. Mushrooms tend to grow at temperatures a little higher than room temperature which makes room temperature ideal. This is because, the humidity and mushrooms will produce heat and naturally raise the internal environment beyond that of the external environment. Airflow is a necessary feature of the environment as well, because there will be consistent air exchange between the Nursery and its' external environment.

The surrounding area should be clean which will prevent or greatly reduce the chance of contamination entering the Nursery. The air in the environment can also carry contaminants so it is important to make sure that the air in the room is coming from a suitable environment. Because the Nursery has a lot of electrical components and wiring, the Nursery will need to be placed in a dry environment. All these conditions help to ensure that the Nursery will not break due to component failure.

2.4 Features

This section describes the various features of the nursery, features were derived by identifying needs in the mushroom cultivation process and were then organized by importance through the HOQ method and divided into 3 categories, firstly and deemed most important are the Core Features, second are Advanced Features and the third category of features is titled Stretch Goal Features. The feature descriptions attempt to keep the discussion on a high level without discussing components or technical data as this is reserved for the later sections.

2.4.1 Core Feature Descriptions

This section contains descriptions of the features we deemed most valuable and most important to the project, they are the core set of features that are the cornerstone of this project and its usefulness.

2.4.1.1 Humidity Control

One of the most important parts in cultivating mushrooms is maintaining the proper humidity ranges at the right times of the mushroom growing cycle. Accurate humidity control is essential throughout the stages of mushroom production. During the stages we are considering primarily primordia formation and fruit body development the environment needs to be at a constant, high humidity for the fungi to develop properly. This requires a humidifier and a means of keeping a constant Relative Humidity (RH) between 85-90% in order to prevent the fungus from drying out. There are a number of prebuilt humidifiers available in the market you can buy to grow mushrooms, whether for commercial cultivation or home cultivation, these typically tend to be expensive.

Due to budget constraints, we built our own humidifier system as part of the housing of the nursery, this allowed us to improve control and maintain a consistently high humidity inside the chamber unit. The humidifier device consists of a water chamber, a fan blower and most importantly an ultrasonic ceramic disc atomizer.

The water chamber is used to hold the water that is used to humidify the mushrooms. The fan blower is used to push moisturized air through the chamber. The fan can optionally comprise of a speed controller, having the advantage of controlling the speed of the air flow will allow us to more accurately control air as well as the amount of moisturized air we want to provide across our chamber area. We have the capability of increasing and decreasing the amount of humidified and ambient humidity air.

The ultrasonic atomizer uses a high output ceramic disc that vibrates at an ultrasonic frequency to silently create microscopic water droplets. It helps create a high frequency oscillation just below the water surface. As a result, it creates a very fine fog water droplet that is instantly evaporated into the air.

Methods for raising relative humidity - To maintain the proper level of humidity inside the Nursery, the humidifier will stay constantly on to provide mist or humidity to the chamber. The controller will be constantly reading the current humidity value inside the chamber through the humidity sensor, if the humidity value exceeded the threshold set value, the controller will be able to automatically shut down the humidifier device.

Methods for decreasing relative humidity - The controller will be constantly reading the current humidity value inside the chamber through the humidity sensor, if the humidity value exceeded the threshold set value, the controller will be able to automatically shut down the humidifier device, while continuing to operate the fan to draw less humid ambient air into the nursery, this helps bring the humidity value down or closer to our target value.

2.4.1.2 Lighting

Lighting plays a very important role in the growth and life cycle of mushroom cultivation. Mushrooms use light in many ways, for instance it helps produce the actual mushroom fruit body as well as maintaining the natural day/night cycle. Furthermore, good lighting and a regulated lighting cycle are important factors in fungi growth. As a result, our system provides both qualities, by utilizing an LED light that provides only the desired wavelength output for perfect mushroom development. Another key factor of using LEDs is that, not only are they great for encouraging mushroom growth, they are very energy-efficient, meaning that they require very little power to produce light.

Additionally, the user is able to set a customized lighting schedule that is optimized for the culture inside, once set the system automatically regulates the lights as desired. A manual overwrite option is provided to the user as well, allowing for on demand enhancements to the light cycle. We use an LED strip inside the chamber to ensure that we are providing sufficient lighting to our mushrooms. The lighting features provide the user with the option to manually overwrite the light cycle if he or she finds it necessary to do so.

In addition to the LED lights, we are also employing an ultraviolet (UV) Light, this invisible form of light can safely kill germs, mold, and in some cases even bacteria and viruses. This light protects the mushroom chamber from any virus or bacteria that would otherwise destroy or affect its ability to grow healthy. UV air purifiers are designed to use short-wave ultraviolet light (UV-C light) to inactivate airborne pathogens and microorganisms like mold, bacteria and viruses. The wavelength of UV light is shorter than visible light, but longer than X-rays. Additionally, literature suggests that exposing mushrooms to UV light increases their Vitamin D production [M8].

2.4.1.3 Airflow Control

Airflow control and circulation is very important during various stages of the mushroom life cycle. The mushroom culture will produce and require specific amounts of carbon dioxide during its lifecycle. Since we are primarily focusing on primordia formation and fruit body development during which most fungi actively produce carbon dioxide but also consume large amounts of oxygen one needs to continuously monitor the amount of carbon dioxide present in the chamber and introduce fresh air to increase the amount of oxygen as needed.

One method to increase oxygen inside the grow chamber is through blowing ambient air into it. A fan is assembled in the housing with the ability to draw fresh air into the chamber to achieve this. An important risk factor that we took into consideration is the potential of contaminated air being drawn in by the fan. To ensure that the fan in the chamber draws clean air into the chamber, we are using an air filter to attempt to block small airborne particulate matter that could contain other fungi, bugs, mold, insects and other undesirables from getting into our chamber as these symptoms can affect the growing life of the mushrooms.

As CO2 is more dense than fresh air it will tend to settle on the bottom of the growing chamber and can produce areas of high amounts of CO2 that can compromise the life of the mushroom. To combat this we mounted an exhaust fan inside the chamber to help with airflow circulation. Increased mushroom activity produces large amounts of CO2 inside the chamber and calls for a higher fan speed setting, as a result the fan speed may be varied automatically or can be manually overwritten by the user.

2.4.1.4 Web Application Features

There are a few potential ways to interact with a controller to view and control the current operating conditions of the Nursery. A Web Application allows the user to create a login and have complete access to all elements related to his Nursery. This enables the user to interact with the controls and set the environment to custom settings and also view statistics of how their Nursery is operating.

2.4.1.4.1 User Interface

The user interface provides a clean and easy to use environment in which the user is able to access a personal dashboard and view the conditions under which their Nursery is operating and also view statistics on how the Nursery is performing.

The user interface pulls the statistics of the Nursery from the database and display them to the user in a comprehensive manner. There are a few potential ways in which this is displayed to the user.

Dashboard - Our dashboard as implemented is a complex interface that shows the user all information regarding his Nursery on one webpage. This dashboard contains the controls for the Nursery and all relevant elements on the center of the page. This proved to be very beneficial as it allows the user to see all necessary information on one page.

Home page with detailed tabs - We considered an interface to give the user a home page with very general information on the first page the user sees when they log in. The user interface will then contain a navigation bar where they can navigate to a specific area of the Nursery. For example, if the user navigated to the “Airflow” tab, the webpage would populate with all relevant information regarding airflow. This would include the controls for the fan and the data showing the fan’s performance/impact on the growing mushrooms. The remainder of the page could be filled with information regarding the impact that changing that control would have or some information about how to better optimize the nursery.

Responsive Design - The user interface was created with a mobile responsive design as a top priority. A website is responsive when the content of that website is able to respond and adapt based on the size of the viewing screen.

2.4.1.4.2 Web Stack

The web stack is important in providing features to the user in an indirect manner. By choosing the right web stack we are able to produce a user interface that makes interacting with the Nursery easy and fast. In this section the similarities of the web stacks will not be discussed and the different features that each can offer will be. Each stack will not be able to offer the same features and the offered features of each will need to be compared upon deciding on a stack.

One of the most important features about the stack we choose to develop with is the transmission of information from database to user interface and how that information can be represented. Because we want to display that information in graphs and charts we will need a web stack that allows us to customize the web page in great detail.

The chosen web stack also needs to be considered for another reason that is unordinary of typical web sites. The database will need to collect information from the microcontroller and store that data. This means that the web stacks ability to communicate with a microcontroller is of great importance.

LAMP stack - This stack can be a huge attribute for the project because of its open-source accessibility. There are open source projects out there that have tackled the microcontroller to database communication and being able to view the code from those projects would allow us to learn from them and adapt them to our environment. Also, because of the open source nature of the stack, we will be able to choose from a range of technologies (like operating systems) if it turns out that a previously chosen technology does not support our intended use.

MEAN stack - The MEAN stack provides a good amount of resources in the user interface department since the A stands for AngularJS. The MERN stack which is the MEAN stack just with React.js instead of AngularJS. React and Angular provide a lot of tools for customizing the user interface which will be very useful when creating graphs and charts. It will also make using other javascript libraries very easy since all of the technologies in this stack, except the database, are javascript libraries. Because of the ease of use for other javascript libraries, we will be able to use technologies like chart.js for creating graphs and charts. The stack we chose to develop the application with the MERN stack as described above.

WISA stack - Because softwares like Code Composer Studio, that will serve a big role in testing and possibly production, are best suited for a Windows environment, it may prove to be useful to use a Windows based stack. Because the microcontroller has a good chance of being programmed using Code Composer Studio it may be possible to have the board communicate directly with

the database without the use of a middle man which is a feature that the WISA stack appears to have over the other stacks since the Code Composer Studio website indicates that not all features are supported on a Mac or Linux environment.

Communication with Controller - The web application's main functionality is to communicate with and send commands to the controller on the Nursery. This communication can be done through wifi connectivity or through a bluetooth connection. We implemented the communication through WiFi.

Bluetooth Connection - In a future improvement the user could also be able to control the settings of the Nursery only when they are using a device that is in range to establish a bluetooth connection. This limits the user's ability to interact with the Nursery from long distances, but will provide faster feedback and the user will be able to immediately diagnose any problems.

Communication through an API - The user is able to communicate with the Nursery from any place they have a connection to the internet. This allows the user to fix any problems with the Nursery when they occur from any distance away, which minimizes any negative effect that comes from the Nursery's environment settings being irregular. Communicating with the device from this distance will create complexities in diagnosing when things go wrong. Because the communication network between the phone and the controller is a lot longer and more complex using this method, it will be a more difficult task to pinpoint where the problem lies when something fails.

2.4.1.5 WiFi Connectivity

The controller embedded in the housing of the grow chamber has various wireless communication requirements. The requirements include but are not limited to, sending sensor data for analysis at a later point to a web server, sending notifications about growing chamber conditions to the web server which will handle further delivery, as well as fetching configuration parameters from the web server.

The user will when first setting up the nursery connect the device to the local internet connected WiFi network. To assist in this manner the device will realize that it is not connected to a WiFi network and switch into WiFi credential gathering mode in which it creates an access point with the SSID of "Mushroom Nursery". This access point will not require a password and allow the user to connect to it via their smartphone or computer, once connected the nursery will deliver a web page to the connected smartphone or computer. This web page contains inputs for the WiFi ssid and password the user would like the nursery to connect to, once the user inputs and saves that information the nursery will close

the access point and attempt to connect to the given WiFi network. If not successful the nursery will revert back to WiFi credential gathering mode.

Once successfully connected the nursery will fetch the latest configuration parameters from the web server. Items such as growing profiles and it's various thresholds as well as update interval times are considered part of the configuration parameters downloaded. The nursery in normal operation mode will make periodic check ins with the server to upload sensor data as well as check for configuration updates, it does this by sending GET and POST requests via transport layer security (TLS) protocol to the web server.

If the conditions inside the growing chamber drastically exceed the ideal conditions for an extended amount of time the nursery will send an additional POST request to the web server in order to alert the web server about the condition abnormality. The web server will then log this abnormality and may also deliver a push notification or sms to the end user to notify. This periodic polling technique is implemented to achieve a rudimentary bidirectional communication line between the web server and the mushroom nursery, while not necessarily real-time this type of connection will suffice for the timing requirements of this project, it is important to note however that in a later section bluetooth connectivity is discussed to aim to fulfill similar requirements as the WiFi connectivity, however the goal of bluetooth connectivity is to deliver and communicate directly and in real time with a users smartphone rather than the web server.

2.4.1.6 Watering System

Mushrooms require moisture to produce their fruit. For this reason, they need an environment that has a high humidity to avoid water loss also the soil needs water to maintain its moisture. In other words, water and humidity play a crucial part in the growing process. The final system contains a water chamber atomizer, however we originally researched the option that it may contain a pump, irrigation system, and misting system as well.

Pump System - We explored the use a submersible pump that will submerge in the water tank to supply water to the humidifier as well as the chamber, where water is necessary. This submersible pump is designed for reliability and ultra-quiet operation to provide years of service. It will be anti-corrosive, acid-resistant and durable. A submersible water pump is great for fresh/salt water tank and hydroponic systems.

Misting System - We also researched the option of a misting system feature of emulating rain droplets when fruiting mushrooms inside. It has the ability to raise the humidity inside the chamber without having to operate the humidifier and fan

systems and may stimulate fruit body formation in some mushroom strains. Our system will consist of a high pressure misting system that will be connected to the water source. We will also allow it to be automatically controlled and scheduled as well as manually operated by the user.

Water Tank - We explored the idea of purchasing a 10 Gallon RV water tank made of durable Polyethylene plastic to hold water that we would use for the humidifier, misting system, and the irrigation system. It seemed like we were going to place the Plant Nursery on an elevated surface, most likely on a table, therefore, the water tank reservoir will be placed outside underneath table on a flat surface due to the weight of the water as well as the pump. The refilling of the tank will be done manually. User will need to fill up the water tank when receiving a low water alert from the water level sensor. In the final design we opted for a water chamber instead.

2.4.2 Advanced Feature Descriptions

This section contains descriptions of the features we deemed of medium value and importance to the project and these were part of our backlog of features as we complete the implementation of the core features. We were ultimately able to implement some of these features.

2.4.2.1 Growing Profiles

The growing profiles provide details about the current and previous grows that either the Nursery itself has been involved with, the user has been involved with, or both. The growing profiles provide details on what an ideal environment for a specific species of mushrooms needs. The details can potentially change over time depending on what works and what doesn't. The goal of the growing profile is to provide a dynamic environment that learns and evolves with the new information it receives. There are a few options for controlling the way the information gets collected by the microcontroller. These include user entered information, recording of sensor readings, and recording of controller and sensor interactions. A combination of these could also be useful but we do not want to overwhelm the user with information, some of which may have no relevance to improving the Nursery's performance.

Growing profile based on User's experience - A growing profile based on a user's experience will provide a much broader range of information than a growing profile based on the Nursery's experience. This allows the user to transfer all knowledge collected by the system to applications outside of the Nursery or to a different Nursery. This will also provide a much more ideal environment for testing and experimentation. A problem that this could potentially create is that the information collected is dependent on the Nursery that was collecting this information. In other words, the statistics may not transfer over well

to other environments due to conditions, not recorded by the current set of sensors, that either existed or did not exist in the Nursery.

Growing profile based on the Nursery - Creating a growing profile based on the Nursery itself seems the most practical as all the data that is recorded will be much more relevant. It will be easy to replicate a grow since the environment will be identical. It will also provide much more impactful evidence when experimenting with the conditions and controls. The information recorded will not be as versatile and will not transfer over to other environments as easily.

User entered information - The user will enter in information that is not recorded by the microcontroller. These measurements can include things like height, width, girth, or other measurements that sensors are not recording.

Recording Sensor Readings - The microcontroller will periodically record the current sensor readings and send that information to the database to be stored for later interpretation. The information in this section is limited to the types of sensors that are being used. For example, since there are no visual sensors, there will not be a section for recording height in the sensor readings table.

Recording Sensor-Controller Interactions - Understanding the relationship and communication between the sensors and controllers is a huge part in creating a dynamic and reactive environment. When a sensor indicates that a controller needs to perform an action we will want to record data about that interaction. That data can include response time, time to stable conditions, and overall impact. If the controller that responds to a sensor does not have much of an impact on the condition that the sensor is reading, then we want to be able to visualize the lack of correlation, and see if other controllers do have more of an impact on that condition. Recording the interactions between the sensors and controllers will also be useful in developing bounds within which the controller is required to turn on or off.

2.4.2.2 Account Creation

A user will create an account through the homepage on either the website or the mobile application. Creating an account will require the user to connect the Nursery with the created account. The user will then need to set up the Nursery's environment through the created account. Upon completion of choosing an environment, this will trigger the sensor and controller cycle and will also establish the data collection cycle as well. The user will then be able to access his account and view the performance of the Nursery through the website or phone app.

2.4.2.3 Data Storage

Each user only needs a few kilobytes of storage for each species of mushrooms. The data can be stored in a few ways, none of which will burden the user with too much setup or upkeep. The stored data can be stored using third party cloud storage, stored on the user's computer, or stored on the microcontroller itself.

Cloud storage - Using a service like Amazon Web Services allows the user to be as hands off as possible and will come at a very low cost since the only stored data will be sensor readings for different types of mushrooms. Using cloud storage also makes the data easily accessible for working with since it can be pulled or updated at any time.

Storing on User's Computer - Storing data on the user's computer is a low-cost and easy way to store the data. However, this makes working with the data complicated as the data will only be able to be pulled and updated when the user's computer is on. This would be a great feature for users who are very active in the world of mycology as they may want to protect their information.

Storing on the Microcontroller - Storing data on the microcontroller itself is another low-cost and easy way to store the data. The microcontroller will now be required to perform more operations than just read sensors and manage the environment controllers. Some mathematical formulas will need to be used to keep the data small. An exponential curve is a great way of storing the data on the microcontroller since only two sets of numbers will need to be stored: the weighted average of all the previous readings and the new readings.

2.4.2.4 Grow Data Reports & Graphs

The grow data reports and graphs represent how the Nursery is performing and provide insight into how an operation can be better improved or how an environment can become more ideal. The overall purpose of this feature is to ensure that the Nursery is working properly and providing expected results.

2.4.2.5 Data Collection

The data collection can be split into two sources of data. Functional data and performance based data. Splitting up the data into these two parts will help the user identify the problem if their mushrooms are not growing as expected.

Functional Data - Functional data is the data collected from the sensors. These measurements ensure that both the sensors and controls are working properly. If the data from this field comes back as expected, and the mushrooms are not growing as expected, then the user will be able to determine that the problem is not faulty equipment but rather an error in setting up the environment. Functional data will be collected by the microcontroller periodically. This helps ensure that the Nursery is working over an extended period of time. The data should be collected when a control is not needed, right before a control turns on, and after that control turns on. This is to ensure that the control is directly impacting the condition that the sensor is reading. Functional data will be sent to the database to be processed and displayed on the user interface through graphs and charts.

Performance Data - Performance data is the data collected that indicates the progress of the mushrooms. This data relies on the performance of the sensors and controllers but does not represent their collected data. The methods of collecting this data are discussed in Growing Profiles. Once the data is collected the user will be able to visualize the data on the user interface. Given both the functional and performance data the user should be able to visualize the correlation between the environment conditions and the progress of their mushrooms.

Model Representation - All collected data will be presented to the user through the use of charts and graphs. There will be different charts and graphs for the functional and performance data. The user will be able to compare data and move charts around on the page in order to study the correlation between two or more sets of data.

2.4.2.6 Automated Alerts

Alerts will be sent to the user when conditions have dropped or risen past the set boundaries. There are a few options for sending alerts.

Email Alerts - The user will receive an email notification when conditions are outside of a given boundary. This option will allow the user to view a full report of the issue and consequences.

Text Alerts - These alerts will be a short description of the issue and consequences since the description needs to fit inside the body of one text message.

Alert on Account - The user will be able to view all alerts on a notification page in the user's account on the website/mobile app. This will enable the user to see if there is a trend in the alerts they receive.

The alerts will also be responsive. If the issue can be solved or at least minimized by activating another control, the alert will have an option to do so, or will at least tell the user what the recommended course of action is.

2.4.3 Stretch Goal Feature Descriptions

This section covers the features that we classified as least important or valuable to the project, this does not imply that they are not important however, they are simply in our estimation not the most crucial to the successful execution of the project goals.

2.4.3.1 Temperature Control

To cool and heat our system, we could use a Peltier device, which is a thermoelectric solid state active heat pump that transfers heat from one side of the device to the other, with consumption of electrical energy, depending on the direction of the current. Thermoelectric cooling (TEC) uses the Peltier effect to create a heat flux at the junction of two different types of materials which is the passage of a current through a junction formed by two dissimilar materials causing a temperature difference. It can be used either for heating or for cooling, we will be using it to do both. The module consists of two sides a hot side and a cold side. The module acts as a heat pump in that it moves heat from the cold side to the hot side. The hot side requires a method of removing that heat for the unit to function properly. The more efficient the means of removing this heat from the hot side, the colder the cold side will operate. TEC helps enhance our cooling ability by creating a temperature differential that can be more easily moved out of

the system. Water-cooling systems can only cool an object to ambient temperatures (room temp), but they still have excess cooling capacity (provided they have sufficient flow-rate and a capable radiator). TEC allows for more of the cooling capacity to be utilized and therefore achieve lesser-than-ambient temperatures. The Peltier works very well as long as you remove the heat from the hot side. After turning on the device, the hot side will heat quickly, the cold side will cool quickly. The primary advantages of using the Peltier cooler over the vapor-compression are its lack of moving parts, circulating liquid, very long life, invulnerability to leaks, small size, and flexible shape these features make this device ideal for our environment.

Methods for Increasing Temperature - The way that we are going to determine when to increase the temperature in the chamber is through the temperature sensor. The controller will read the current value of the temperature in the chamber, if that temperature value exceeded the threshold set value, the controller will automatically trigger the Thermoelectric peltier device into heating configuration and enable it.

Methods for Decreasing Temperature - Just as for the temperature increasing methods, the way that we are going to determine when to decrease the temperature in the chamber is through the temperature sensor. The controller will read the current value of the temperature in the chamber, if that temperature value is below the threshold set value, the controller will automatically trigger the Thermoelectric peltier device into cooling configuration and enable it.

2.4.3.2 Native Mobile Application

The native mobile application will be designed to provide the user with a simple and easy interaction. There are a variety of tools available for designing a mobile application. Most of the applications on the market today are developed using software development kits (SDKs), typically provided by the large scale software device manufacturers. Two examples of these companies, which are the largest producers of operating systems for mobile devices, are Apple (iOS) and Google (Android). There are also a few other tools to build mobile applications without using the SDKs previously mentioned.

2.4.3.2.1 Android SDK

The Android SDK is the front-runner for developing the mobile app for the Nursery. This is because the engineer in charge of testing the application will be using an Android based device. The Android development environment supports creating applications for mobile phones but also for tablets as well. If the user has an Android tablet, they would be able to download the application on their tablet and have complete access to the applications' features. Android is also open source. This will be useful as we do not need any licensing agreements to develop on the entire framework. There are many Application Programmable Interfaces (APIs) levels for Android supported devices to use. The levels dictate which devices are able to operate with a certain version of Android. Because of the forwards compatible nature of the APIs, targeting the level that supports the most devices will allow the user to use a wide range of devices to access the application.

Because the Android SDK is setup to use Eclipse as the Integrated Development Environment (IDE), all development will be done using Eclipse. The language used to develop on this platform is Java and has support from HTML.

The Android SDK is a robust tool for developing applications, and will be very useful as Android users are abundant.

2.4.3.2.2 iOS SDK

Apple provides a native software development kit which allows for direct integration into the device's hardware. Apple only recently released their SDK for iOS devices a little over a decade ago. Because of the hardware integration, applications are optimized for the device and are able to provide better power efficiency. The app that accompanies the Nursery is not intended to be battery intensive, but it is always a plus when an app drains even less battery than originally thought. The iOS SDK, like the Android SDK, allows for development

on more than just mobile phones. There are a wide range of iOS devices, including things like tablets, that a user would be able to download the app on and interact with their Nursery. The iOS SDK is available for download on Apple's website. The Integrated Development Environment that accompanies the SDK is Xcode, which is the dedicated IDE for all Apple devices.

The programming language used to write code for the iOS SDK is called Swift. Swift is an object oriented language similar to languages like Java and Python. Apple also provides software development frameworks that can be used to build more robust applications. One framework that could provide a lot of useful applications to the Nursery application is Cocoa Touch. Cocoa Touch is a User Interface framework designed for building programs to run on an array of Apple devices. Cocoa Touch would provide our applications with much richer graphs and charts. Cocoa Touch would also allow our application to access other applications, which would work really well in designing a middle man service to transfer data between the controller and application.

Because there are two major operating systems that people use on their phones, it would be most efficient to make the separate applications as similar as possible. This would make development easier and would provide users using different devices with a similar experience.

Cross Platform SDKs - There are a few SDKs on the market right now that support multiple operating systems. This would eliminate the need to have two different code bases. Having only one code project solves a lot of issues that would come with having to create one application for two different operating systems with two different toolsets. There will most likely still be issues that only one operating system experiences that the other will not, but this issue is much smaller than having to create two different code projects. This would also make debugging a simpler task because if the bug is happening on multiple operating systems then it would be easy to speculate that the problem lies within the written code.

Appcelerator - Appcelerator Titanium SDK is an open-source framework that we would be able to use to develop native mobile applications on a wide range of platforms. These platforms include but are not limited to iOS, Android, and Windows UWP which are the largest platforms currently on the market. The code used to write these applications using this SDK is JavaScript.

PhoneGap - PhoneGap is similar to Appcelerator in that only one codebase is required for developing across multiple platforms. PhoneGap is an HTML5 framework specifically created to allow developers to build applications across multiple platforms while only requiring code base.

The PhoneGap framework is based on web technologies like HTML, CSS, and JavaScript. Developers will also develop across APIs to interface with the native hardware on the device running the application. PhoneGap provides middle-man software between a JavaScript framework and the underlying hardware on the mobile device to produce applications natively. PhoneGap SDKs also work alongside the SDKs for the other SDKs designed by the phone's operating system manufacturers. This way the programmer does not have to work with the programming languages normally required to develop on specific platforms.

2.4.3.3 CO2 Measurement & Control

CO_2 is a normal product of fungi metabolism and is one of the most important factors in determining high quality and yield of edible fungi. To maximize mushroom growth, it is critical to monitor the level of CO2 (Carbon Dioxide) in your growing chamber. CO2 is measured in ppm and is the major factor for obtaining good quantity, quality, and size of the produced mushrooms. An improper balance of CO2 can induce fruiting body deformities or abnormalities. As a result, we will implement proper CO2 detection and ventilation in order to control the CO2 concentration.

We explored the use of a CO2 sensor to read the level of CO_2 concentration of the air in the chamber. The precise sensor for carbon dioxide measuring uses a built-in non-dispersive infrared element (NDIR), which is responsible for the monitoring of the CO_2 concentrations. The communication with the controller is carried out by a 1-Wire interface. The software sends instructions and commands to the mechanisms for ventilation, after analyzing the data on the CO2 concentration in the chamber and taking into account the rest of the parameters which affects mushroom growth. Based on those readings, we can adjust the level of CO2 in the chamber by introducing fresh air from the ambient environment. Due to the high cost to acquire an accurate CO2 sensor and our budget being constrained we decided to use frequent air exchanges to manage CO2 instead.

2.4.3.4 Bluetooth Connectivity

As previously stated in the WiFi Connectivity section, the nursery has various wireless connectivity needs. As part of those needs the nursery sends sensor data to the web server for logging into a database and to later make available to the end user via web dashboard or mobile app that request this data from the web server. As an additional use case, or in one where the user may not have the nursery connected to WiFi, the user could using the native mobile application connect to the nursery directly via Bluetooth.

The nursery when plugged in and operating will continuously broadcast itself as a bluetooth device that a smartphone or other Bluetooth capable device can pair with. If the user has the native mobile application installed and has their smartphone paired with the nursery via bluetooth he or she will then be able to retrieve real time sensor data directly to the smartphone. This real time data includes items such as temperature inside the growing chamber, humidity inside the growing chamber, CO2 level inside the growing chamber, as well as items such as how long the lights, fan and humidifier have been enabled or disabled.

Besides the previous items the user will also be able to retrieve the current operational status of the fan, lights and humidifier and manually override the operation. This allows for scenarios where the user may want to manually turn the lights on or off or manually want to force more airflow by enabling the fan. Additionally the user will be able to see and change the growing profile that the nursery is currently operating under as well as modify the growing profiles parameters and submit those to the nursery. The nursery will then relay any updates to the webserver at the next update period, if a conflict between updates arises it will always treat the updates made via bluetooth with priority.

2.5 Components Diagram

The diagram in Figure 1 illustrates the components required for the core features as well as some of the components for the lower priority features at a high level. With the controller at the center and sensors as well as connectivity components in purple, the devices such as light, fan and humidifier are displayed in orange while the web server, database and mobile and web application are displayed in blue.

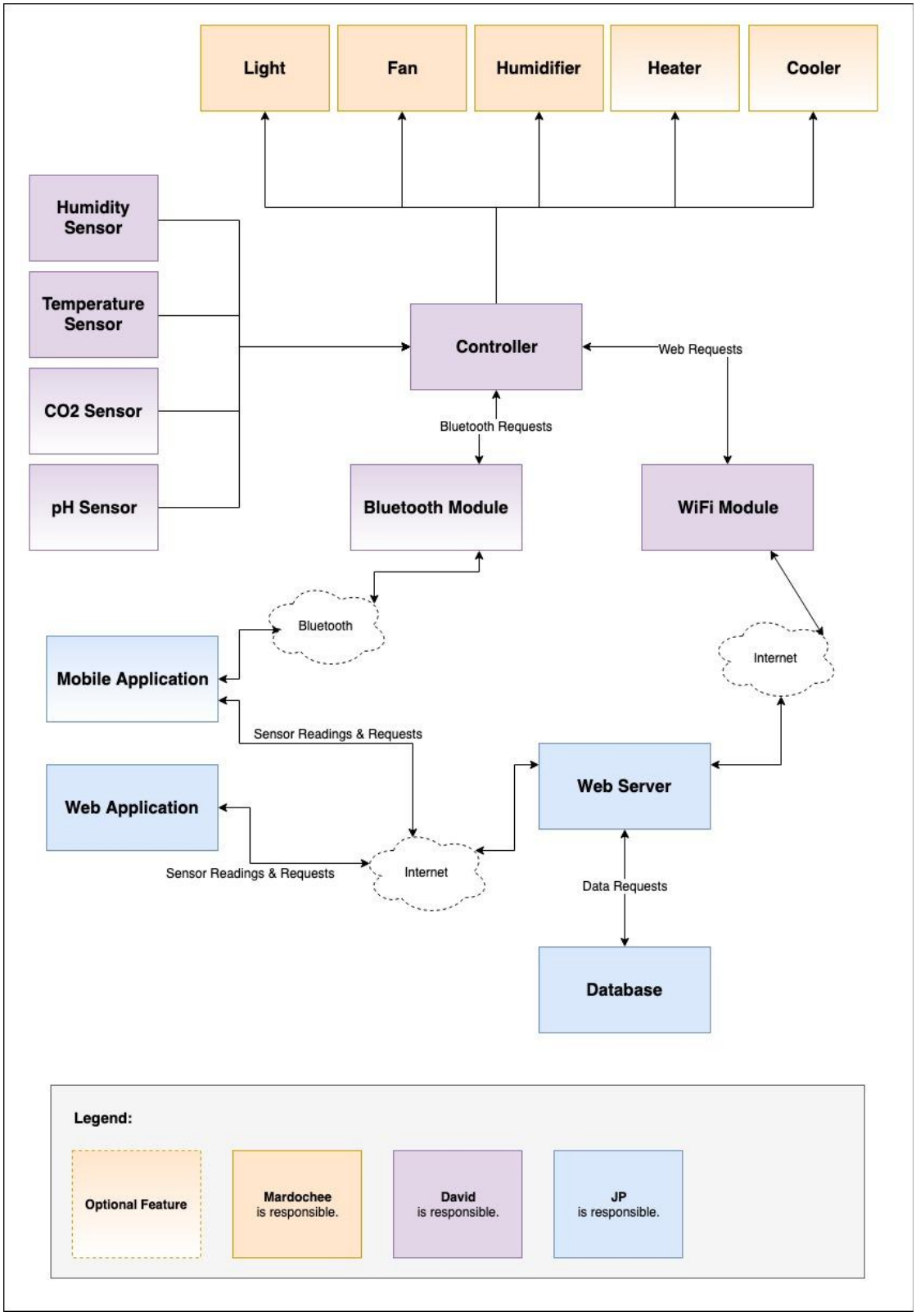

Figure 1: High Level Components diagram depicting major functional components of the project.

2.6 Component Detailed Descriptions

This section describes the components that were chosen, researched or are otherwise relevant to the project in more detail, it may also include helpful diagrams, illustrations and links to educational resources.

2.6.1 Housing

The housing of the components can be divided into two separate parts with a series of tubes connecting them. One part will contain the mushroom mycelium that will produce the mushrooms. This part contains the environment that all the sensors and controllers will be acting upon. The other part will contain the controllers themselves. The two parts will be connected by a series of tubes, so that the output of the controllers are able to enter the environment where the mushrooms are being grown.

Enclosure – The enclosure contains the mushroom mycelium and the sensors. This will be the area where all conditions will need to meet the written criteria. The sensors are able to read data in this area and send necessary information to the controller. The mushroom mycelium will sit on a metal or plastic plate on top of a layer of perlite. The perlite will hold moisture and allow the nursery to easily keep humidity at high levels inside the enclosure. Underneath the perlite layer may be a screen with holes big enough to allow excess water to drain but small enough to keep the perlite from falling through. The excess water will drain onto a pan which will be easily accessible to the user.

Controller Housing - The controller housing will contain all of the components necessary for the system to operate, but are not necessary inside the enclosure. This housing should shield the inner electronics from any foreign materials. The housing should also be setup in a way that overheating does not become an issue. This housing will be connected to the enclosure through a series of tubes that carry the input from this housing into the enclosure.

Summary of Housing Requirements - A summary of the housing requirements described above is placed below to use for easy referencing.

1. Contain all the input from the controllers inside the chamber without leaking out.
2. Prevent electronics from getting wet and/or incurring unnecessary damage.
3. Properly drain excess water from high humidity and allow the user to easily remove the excess water.
4. Expel all current environmental conditions through exhaust ports when necessary.

5. Tubing between the two components should be air tight and prevent any possible leakage.
6. Whole system should be stable and remain still while operating.
7. Circuitry and electronics should be covered and not exposed.

2.6.1.1 Materials

The housing requires a variety of materials that compose different parts of the structure. Materials were carefully considered and chosen to accomplish a specific task. The materials needed to be able to work seamlessly with other chosen materials and be able to withstand any conditions that the Nursery is placed under.

The material that the enclosure is made out of needed to be strong enough to support the tubing and sensors but also see through. The skeleton of the enclosure was designed using polycarbonate. The faces of the enclosure are a see through material capable of holding in moisture, humidity, and able to withhold any temperature that the mushrooms need. There are a few options for the see through material that will be used on the faces of the enclosure.

Polycarbonate sheets - Polycarbonate sheets are a hard and transparent material that is typically used for patio roofs and windows. Polycarbonate is sturdy and will not shatter if dropped. This material would make a good option for the enclosure as it will be able to sustain any of the given conditions.

Polyethylene sheets - Polyethylene sheets are a viable option for the enclosure. They are flexible and will be able to sustain any of the conditions required for the Nursery. Polyethylene sheets are, however, not very transparent and will not allow the user to easily see into the enclosure. An option would be to use polyethylene sheets on three of the four faces and use a different material for the fourth face to allow the user to see in.

Poly(methyl methacrylate) - This material is also known as acrylic, or plexiglass. This thermoplastic material is lightweight, transparent, and shatter resistant. This material would satisfy all necessary conditions for the faces of the enclosure. The decision between plexiglass and polycarbonate sheets will come down to price since both are very similar in composition and performance.

The materials that compose the controller housing should be very similar to a computer case. Thin metal sheets or a plastic casing will be used for the housing body. Silicon tubing will be used to connect the two parts of the system. Silicon tubing is very resistant to high temperatures and will be very useful in the transferring of input conditions.

2.6.1.2 Design

The design was very briefly described above to give an overhead view of how the materials were chosen and how the components will fit inside. This section will illustrate the system.

Figure 2 details one of our ideas for the structure of the enclosure part of the system. The sensor bar runs back and forth (into the page) and will hold all the sensors that read data from the enclosure.

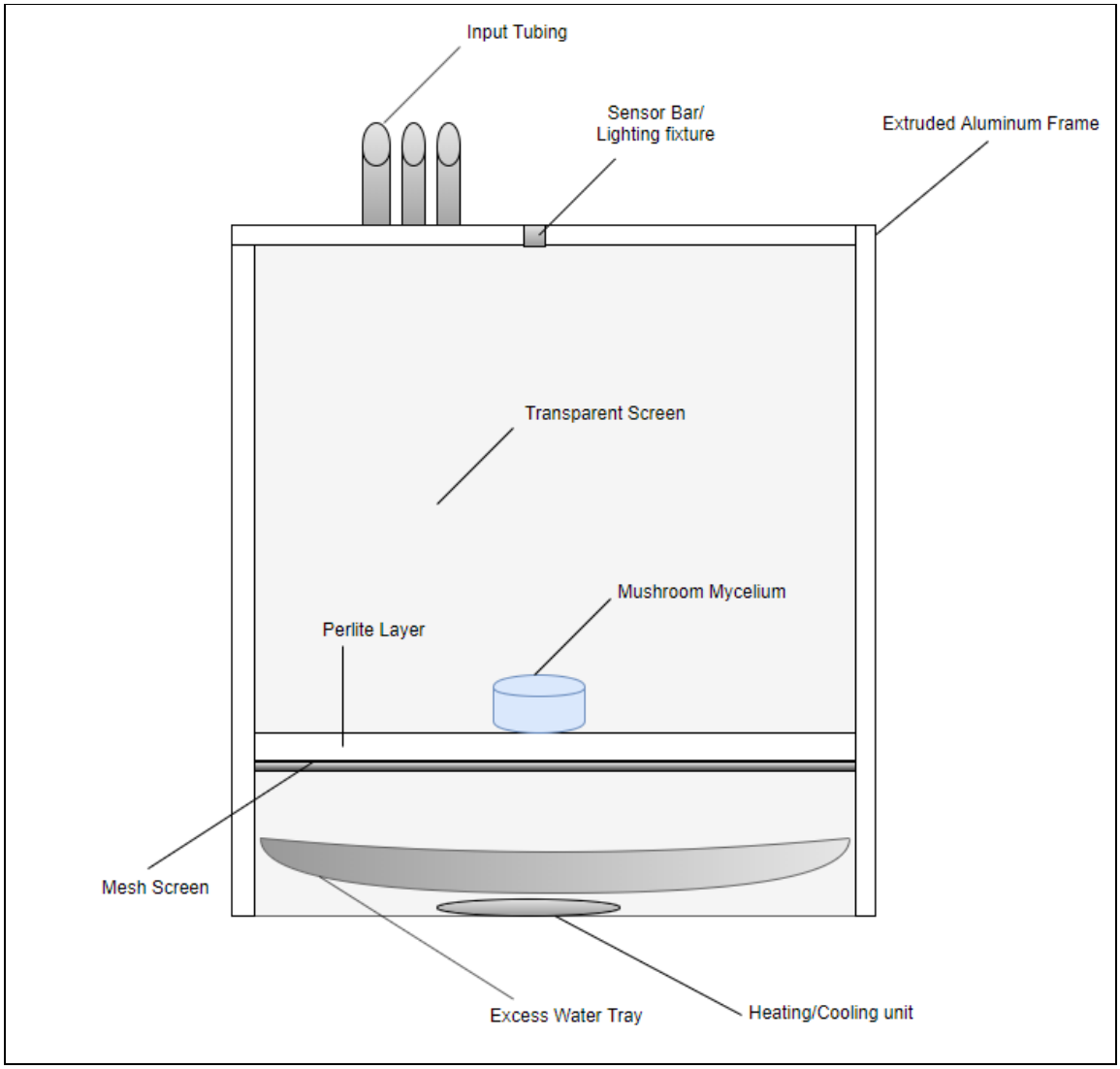

Figure 2: Illustration of the enclosure of the system.

Figure 3 details the structure of the controller housing. The wires in this diagram go to specific components within the enclosure while the tubing is the same tubing that is displayed in Figure 2.

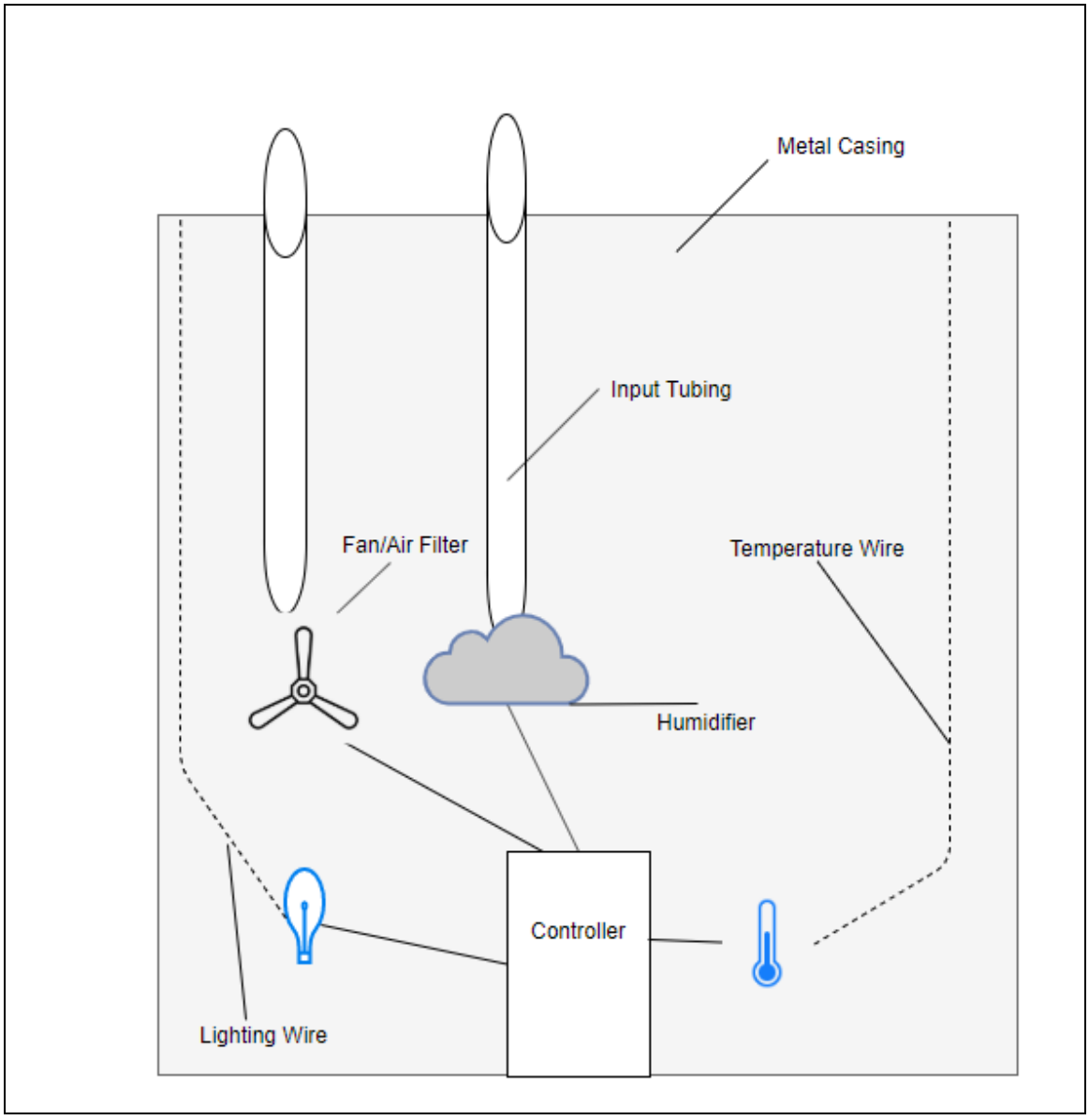

Figure 3: Illustration of Controller Housing

2.6.2 Light

There are a number of lights available in the market to help you grow plants and fungi. For example, HPS (High Pressure Sodium) Grow Lights, HID (High Intensity Discharge) Grow Lights, LED (Light Emitting diode) Lights, and Sulfur Plasma Grow Lights. The HPS Grow Lights have been one of the most common types of indoor growing lights because of their ability of emitting lights at a very high spectrum, 565-700 nanometers (nm) as well as their lifespan around 10,000 hours. The HID Grow Lights offer the brightest light compared to any of the lights mentioned above and they are about 10 times more efficient than traditional incandescent lights.

For this project, we are going to use LED grow lights for multiple reasons. First, LED lights have a long lifespan (more than 50,000 hours). In addition, they are extremely energy efficient and consume up to 90% less power than incandescent light bulbs. Furthermore, they do not require the addition of any kinds of reflector, as the light is directly emitted towards the plants and not dispersed like other type of grow lights.

On/Off functionality is available to the users via the mobile/web application. User will be able to choose when to turn on/off the lights in the chamber as well as letting the Nursery automatically control the light cycle inside the chamber.

Table 1 below summarizes the research of the SMD 5050 LED Grow Light by Youhuan, a popular grow light used in the industry, we used this light and it's properties to guide our LED component selection.

Input Voltage	12 Volt DC
Lamp Luminous Flux	1690 lm
Working Lifetime	50000 Hours
Lamp Power	14.4 Watts

Table 1 - Overview of properties of the Youhuan SMD 5050 LED Grow Light

2.6.3 Temperature Sensor

The temperature sensor plays an important role in our project. There are many different types of sensors out there that we can choose from to read the temperature in the chamber. For instance, Negative Temperature Coefficient (NTC) thermistor, Resistance Temperature Detector (RTD), Thermocouple, and Semiconductor-based sensors. For this project we looked at using a High Precision Temperature Sensor known as TMP117 because it has sufficient accuracy and low integration complexity. It outputs temperature readings with a precision of +/- 0.1°C across the temperature range of -20 °C to 150 °C with no calibration. The TMP117 sensor has a 16-bit resolution of 0.0078 °C. The low power consumption of the TMP117 minimizes the impact of self-heating on measurement accuracy. It operates from 1.8 V to 5.5 V with a typical current consumption of 3.5 µA . In addition, RTD sensors are accurate, sensitive, standardized and interchangeable. When a specific high temperature threshold is exceeded, preventative action can be taken by the system to lower the temperature, an example of that would be to turn on the fan or enable the cooling unit. For easy reference the properties we deemed most valuable about this sensor are displayed in table 2. However due to the sensor described in 2.6.4 providing a combination humidity and temperature sensing capability, we ultimately ended up using the RHT03 in our final design.

Name	TMP117 High Accuracy Temperature Sensor
Operating Temperature	-55°C to +150°C
Power Consumption	3.5 µA
Voltage Requirements	1.8 V to 5.5 V
Resolution	16-Bit

Table 2: Overview of the Properties of the TMP117 Temperature Sensor

2.6.4 Humidity Sensor

Our System has a sensor to measure humidity in the chamber. Just like the temperature sensor, having a humidity sensor helps us monitor the level of humidity inside the chamber. When a specific high humidity threshold is exceeded, preventative action can be taken by the system to lower the humidity level, an example of that would be lowering humidifier fan speed or turning off the humidifier unit completely. To read the humidity inside the chamber we researched a variety of components and favor a component by MaxDetect with model number RHT03 (also known as DHT-22), this sensor is a low cost humidity and temperature sensor with a single wire digital interface. The sensor comes

pre-calibrated and doesn't require any additional components. Table 3 below gives an overview of the component parameters for quick reference.

Model	MaxDetect RHT03
Power supply	3.3 - 6V DC
Output signal	Digital signal via MaxDetect 1-wire bus
Operating range	Humidity 0-100% RH Temperature -40~80 Celsius
Accuracy	Humidity: +-2% RH Temperature: +-0.5 Celsius
Current	1 - 1.5mA

Table 3: Overview of MaxDetect RHT03 Humidity and Temperature Sensor

2.6.5 Humidifier

The humidifier is a crucial component of the Nursery as humidity is a key factor in mushroom cultivation, especially during the stages of the mushroom life cycle that we are concerned with. We evaluated both evaporative as well as atomization techniques for humidification and concluded that an atomization technique would be best suited for the project. Specifically using an ultrasonic disc atomizer to generate air that contains a high relative humidity content. The complete humidifier design also includes fans and may include a separate water tank and pump system as depicted in Figure 4.

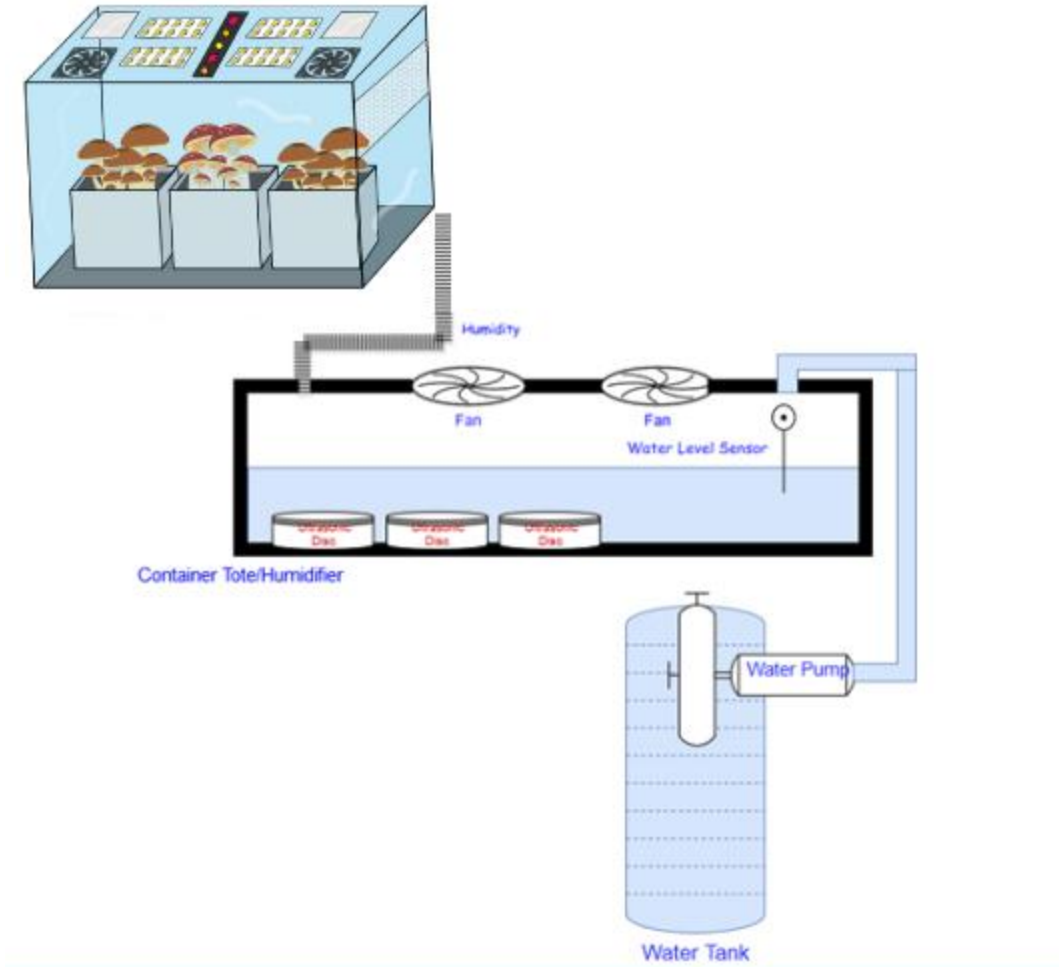

Figure 4: Illustration of Humidifier System Design and connection to Nursery chamber.

Fan - To push the humidified air out of the humidification chamber and into the growing chamber a standard 5-12 Volt DC powered fan will be utilized. These types of fans are typically utilized for CPU and computer related cooling and operate on less than 1 Amp of current. As a reader of this document you are familiar with the properties of these types of fans and thus we removed details of specific fan research for brevity.

Ultrasonic Atomizer - This device uses ultrasonic vibrations to transfer energy into the water that it is submerged in, this creates enough movement to allow water particles to get atomized into the surrounding air at a rapid rate, and hence resulting in air that contains a high relative humidity percentage. We researched a variety of ultrasonic atomizers and concluded that a low voltage DC powered and waterproof device would best fit our needs. The typical parameters of the devices we researched are summarized in table 4.

Operating Voltage	12 - 24 V DC
Current Draw	Up to 1 Amp
Cost	$10 - $15 / unit
Operating Lifetime	3000 - 5000 hours

Table 4 - Summary of typical Ultrasonic Atomizer Operating parameters.

Water Pump - If a water pump would've been included in our final design to provide less frequent need for refilling the Nursery with water, we looked at acquiring a low voltage and relatively low throughput water pump to replenish the atomization chamber with water as needed. These types of water pumps are readily available on the market, are submersible and have a typical operating voltage of 12V with a current draw of up to 3 Amps.

Water Level Sensor - In order to determine when the atomization chamber is running low on water we would implement a water level sensor that will allow the controller to determine the amount of water left in the chamber and if needed take action to refill the atomization chamber. The type of sensors we researched for this project are ones that communicate using a single wire protocol, can be acquired for less than $15 per unit and operate on up to 12V DC with a very low current draw.

2.6.6 Cooling and Heating

To cool and heat our system, we explored the use of a Peltier device, which is a thermoelectric solid state active heat pump that transfers heat from one side of the device to the other, with consumption of electrical energy, depending on the direction of the current. Thermoelectric cooling uses the Peltier effect to create a heat flux at the junction of two different types of materials which is the passage of a current through a junction formed by two dissimilar materials causes a temperature change. It can be used either for heating or for cooling, we will be using it to do both. Table 5 gives an overview of the typical peltier devices that we are considering for the project.

Operating Voltage	12 - 15 V DC
Current Draw	Up to 10 Amp
Maximum achievable temperature difference	50 C
Cost	$10 - $15 / unit

Table 5: Overview of considered Peltier Device Considerations.

2.6.7 Power Control and Power Regulation

The project consists of various components that require power in order to operate, besides just needing power, a variety of components are very sensitive to specific operating voltages in order to function properly as well as not to do any physical damage to the components. All components were selected to require DC voltage, the major components that we are concerned with in this section are the chips housed on the PCB, the sensors and finally the peripheral devices. Peripheral devices includes items such as lights, humidifier, fan and heating/cooling device. Components such as the sensors and chips on the PCB are typically sensitive to voltage fluctuations and hence require regulation of the incoming voltage. The peripheral devices are not as sensitive, mostly since they often times include voltage regulation capabilities themselves, however they need to be switched on and off by our controller. The following sections discuss the research that was done to find appropriate methods for power regulation as well as power switching as well as how those methods are then implemented to power the board and sensors, as well as how they are implemented to switch the peripherals.

Power Regulation Research - When researching how to power the major components of our project we discovered two major factors firstly components like our controller are very sensitive to voltage differences, for example it's ideal operating voltage is 3.3V with an operating voltage range of 2.7V to 3.6V, where at 2.7V the component may experience brownouts and not function properly, and at above 3.6V the controller may experience irreversible physical damage. Hence we require a voltage source that can operate within that 0.9V operational range. And secondly that the type of wall adapters that will power our project are very unregulated in their output voltage and will vary greatly, for example a typical 6V AC to DC wall adapter power supply will often vary in output by as much as 4-5V depending on the load resistance of the device. Hence we will need to run our incoming power through an additional Voltage Regulator that provides much greater precision, prior to passing it on to our controller or the other chips on the board.

Power Switching - Our peripheral devices like lights, fan, humidifier and heating/cooling unit are always in either an on or off state depending on what the controller currently requires from them. The following devices to handle this switching were researched and evaluated: Relays, MOSFETs, Triacs and Thyristors. From the evaluation of our research we concluded that Relays and MOSFETs would be ideal candidates for our application, with relays being ideal for switching loads larger than 5A, and MOSFETs being smaller in size and typically cheaper. Hence we decided to look at both Relay and MOSFET devices as possible candidates for handling the switching of our peripheral devices.

Voltage Regulator Research - The below tables summarize our research regarding Voltage regulators, our main objective was to find a low complexity, cost-effective, accurate regulator that could power our SoC controller module and sensors at 3.3V. We considered a voltage regulator to have low complexity if it requires few or no additional passive components to achieve its function, and considered sufficient cost efficiency to be any regulator that costs less than $1. Since the ESP32 SoC has an ideal operating voltage of 3.3V with a maximum rating of 3.6V it is important that the regulator's output voltage does not vary by more than 10%, however to add an additional safety buffer we set our maximum variation requirement to 5%. The detailed component minimum requirements we were looking for are shown in table 6.

Requirement	Threshold Value	Interpretation
Output Voltage	3.3 V	Fixed, as close to 3.3 V as possible.
Output Current	600 mA	At least, greater is better.
Maximum Input Voltage	6 V	At least, greater is better.
Maximum Cost	$1.00	At most, less is better.
Maximum Output Voltage Variation	± 5%	At most 5% variation, less is better.

Table 6: Minimum requirements of Voltage Regulator

Diodes Inc. AP2112K-3.3TRG1 - This fixed 3.3V voltage regulator by Diodes Incorporated is a SMD component that scored well in our evaluation as it meets all of our minimum requirements. It is also available in other output voltage configurations with may be helpful if we require a 5V line for peripherals. It's specifications are summarized in table 7.

Complexity	Medium. Requires a capacitor on Vin and another capacitor on Vout. Has an enable line with high signal being enable. (Datasheet)
Max Input Voltage	6 V
Max Output Current	600mA
Output Voltage Variation	± 1.5%
Unit Cost & Quantity	$0.47 (30,327 units available)
Vendors	Mouser

Table 7: Summary of the AP2112K Voltage Regulator

STMicroelectronics LD1117 - This fixed 3.3V voltage regulator by STMicroelectronics is available in a variety of packages however we focused on the SOT-223 and SO-8 form factors, it exceeds all our minimum requirements. It is also available in other output voltage configurations with may be helpful if we require a 5V line for peripherals. It's specifications are summarized in table 8.

Complexity	Low. Requires two capacitors one in Vin and one on Vout, does not require enable line. (Datasheet)
Max Input Voltage	15V
Max Output Current	800mA
Output Voltage Variation	± 2% in general, and as low as ±1% at 25C
Unit Cost & Quantity	$0.46 (13,153 units available)
Vendors	Mouser LCSC

Table 8: Summary of the LD1117 Voltage Regulator

Voltage Regulator Research Summary - Both of the components researched satisfy our requirements however the LD1117 by STMicroelectronics scored higher than the AP2112K on the following criteria, first it provides lower implementation complexity as it does not require an additional enable line and second it has higher maximum ratings for input voltage and output current.

Switching Device Research - A variety of similar devices were researched however this research summary lists two components in total, each representing what we considered the average of the typical device available on the marketplace. One component being the typical Relay and one component representing the typical MOSFET device for our application. Our device's power switching requirements were the main guiding factor for coming up with the below minimum requirements. We consider devices to be lights, fan, humidifier and heating/cooling unit as distinct from sensors. We do not anticipate that our sensors require any switching capabilities and if for reduced power consumption measures we do disable them during operation their requirements are less than those of the devices. We estimate that our total current requirements are around 5 Amps, however for additional buffer we are looking for a switching capability of 10 Amps. Additionally we are looking for a component that when not energized disables the device, this is often called Normally Open or abbreviated NO. Table 9 summarizes the requirements.

Requirement	Threshold Value	Interpretation
Minimum Switching Voltage	24V DC	At least, greater is better.
Minimum Switching Current	10 Amp	At least, greater is better.
Normally Open	Yes	Fixed.

Table 9: Minimum Requirements for Switching Device

Omron G5Q-1A4-EL2-HA-DC24 - This PCB Power style miniature relay created by Omron is available in coil ratings from 5V DC to 24V DC. Each relay requires a minimum voltage of 75% of it's coil rating to operate. It comes in a form factor designed to be mounted onto a PCB and generally satisfies our requirements, as can be seen in it's summary in table 10.

Supported Switching Voltages	Packages available in 5V, 9V, 12V, and 24V
Minimum coil voltage	75% of Switching voltage
Switching Current	10 Amp
Normally Open	Yes
Vendor & Datasheet	Mouser (Datasheet)
Cost & Quantity	$1.88 (629 units available)

Table 10: Summary of the Omron G5Q Relay

Fairchild FQP30N06L - This MOSFET by Fairchild is available in Through-hole and SMD packages and meets or exceeds all our minimum requirements. The specifications are summarized in table 11.

Supported Switching Voltages	Up to 60V
Minimum Gate to Source Voltage (Required to turn on the MOSFET and provide sufficient current)	3V provides about 20A to load 5V provides about 30A to load
Maximum Switching Current	30 Amp
Normally Open	Yes
Vendor & Datasheet	Sparkfun (Datasheet)
Cost & Quantity	$0.95 (In Stock, exact count not available)

Table 11: Summary of the Fairchild FQP30N06L MOSFET

Supplying Power to Controller and 3.3V Sensors - The board design will consist of a 3.3V Voltage regulator as described above, a wall adapter with up to 6V DC output will be plugged into the board via a barrel jack plug. The barrel jack plug is connected to the voltage regulator which will feed a steady 3.3V to the controller as well as the Sensors that require 3.3 Volt. The current draw of the controller and sensors is expected to be no more than 600mA which should not warrant for any special temperature or trace width considerations.

Supplying Power to Devices and 3.3V+ Sensors - Since we anticipate some of the devices to have larger power requirements, either a separate power adapter providing up to 24V DC and up to 5 Amp total will be required that will plug into a designated barrel jack, from there the sensors and devices that require more than 3.3V but have a low current draw will be powered through an appropriate Voltage Regulator and controlled via a designated MOSFET. The devices with current requirements of 1 Amp or more will be powered from the same plug and controlled via dedicated Relays, additionally if any power regulation is needed the appropriate voltage regulator will be provided. Due to the larger current flowing through the board we are concerned with temperature rise and a potentially increased circuit board trace width for the lines that will carry this current.

We are also considering a less complex approach that simply provides multiple general use NO Relays or N-channel MOSFETs with screw terminals, this would allow the board to be less customized and more flexible in the type of devices it controls however it would require multiple power adapters, ie. one for each device. In this configuration the device's power adapter positive line would be wired directly to the device, and the device's negative/ground line would be connected to the screw terminal which then connects it to the drain of a MOSFET or the NO connector on the relay. The negative/ground line coming out of the device's power adapter would go into the other port of the screw terminal which then connects it to the source connector of the MOSFET or the common connector of the relay. The controller will then either provide a control signal to the gate of the N-channel MOSFET or the coil of a NO Relay. Note that this approach would not eliminate the need for special temperature rise and trace width considerations. Additionally if a MOSFET is used temperature of the transistor junction should be considered.

MOSFET Transistor Junction Temperature Considerations - To ensure we are appropriately sizing our MOSFETs to meet our power requirements we are using the following calculations to estimate junction temperature inside the MOSFET.

$$T_j = T_a + R_{\theta JA} * P_d$$

Where T_j is the junction temperature, T_a is the ambient temperature, $R_{\theta JA}$ is the junction to ambient thermal resistance of the device and P_d is the power experienced by the device. Using this formula with the assumption of 5 Amps flowing through the Fairchild FQP30N06L at an ambient temperature of 25 C we estimate that the junction temperature is 95 C, this is well within the MOSFETs operating temperature range of -55 to +175 C.

2.6.8 Project Schematic & PCB Design

A big part of the hardware side of the Nursery project was the assembly of micro electronic components onto a board. While this could initially be done with a breadboard, due to the amount of SMD components our project requires the bradboard option is difficult to unusable. Instead we designed a Printed Circuit Board (PCB) that has the required components soldered to it. An important component in the design process is the design software used to create the schematic and PCB board design. This section covers the research summary and evaluation of Schematic & PCB Design software solutions often called Electronic Design Automation (EDA), as well as our schematic of the project.

AutoDesk Eagle - The Eagle design software by AutoDesk seems to be the one most used in the industry and the hobby community, it's been available for over 30 years and seems to be the most complete and feature rich solution, however it did not seem like the most intuitive product and especially it's parts library was not as easy to use as we had hoped. Table 12 contains a summary of our research.

Schematic Design Features	● SPICE Simulator ● Modular Design Component blocks ● Ability to validate design with Electronic Rule Checker
PCB Design Features	● Automatic routing generator ● Ability to validate with Design Rule Checking mechanism (DRC) ● 3d preview of board including models of most components available
Component Library	Manufacturer provided Component Library exists and can be extended through Community created component libraries. However most of the components we would be utilizing were not found in either libraries.
Ease of Use	Medium, seemed to provide a lot of features with a UI that did not feel intuitive to us.
Cost	Free for Students, restricted to 2 layers and max of 80cm^2 area.

Table 12: AutoDesk Eagle Research Summary

EasyEDA - EasyEDA is a schematic and PCB design software that seems to be a recent product, and contains all the features a beginner and professional would use however in terms of user interface it seems to be geared towards beginners or hobbyists. A great feature is that it is a web based software first, with desktop clients available if desired, so sharing the project and team management seems very easy. This EDA software benefits heavily from part manufacturer and community provided component libraries that come with schematic pin mappings and accurate pads and dimensions for PCB design. Our research regarding EasyEDA is summarized in table 13.

Schematic Design Features	● SPICE Simulator ● Modular component blocks ● Ability to open EAGLE files ● Version Control
PCB Design Features	● Auto Routing Tool ● 3d Preview of Board ● Ability to check dimensions of footprints
Component Library	An extensive manufacturer provided component library, including extensive up to date libraries by part manufactures and the community. Overall over 1 million component libraries. Component library has integration with LSC parts catalog. All the components we were researching were available in the component library.
Ease of Use	Easy, we found the software very easy to use and beginner friendly.
Cost	Free full featured commercial use license, only downside of free version is that it contains Ads as well as lacks the ability to create private component libraries.

Table 13: EasyEDA Research Summary

Fritzing - Fritzing is an Open Source EDA developed by the Potsdam University in Germany as well as online contributors, it seems to provide enough featureset to go from schematic to full professional PCB design however it's overall featureset seems slim. It is available as desktop client and has a community provided component library that seems to mostly focus on Arduino and similar type projects. Table 14 contains the summary for the Fritzing EDA.

Schematic Design Features	● Modular component blocks ● Ability to export to a variety of file types ● Has a breadboard schematic designer as well.
PCB Design Features	● Auto Routing Tool ● Live updates when Schematic changes
Component Library	A community driven component library exist but from our research the model quality was very sporadic. Library is relatively slim and mostly Arduino focused.
Ease of Use	Easy, The software seemed easy to use and geared towards beginners and hobbyists.
Cost	Free, Open Source.

Table 14: Fritzing EDA Research Summary

DesignSpark PCB - This EDA is developed by RS Components and integrates with their product catalog nicely, it seems to be a relatively easy to learn tool and does not come with any concernable limitations in the free version. The Schematic design features seemed basic but sufficient and the PCB design features seemed well thought out and feature rich. The software is only available for Windows. Our research summary is found in table 15.

Schematic Design Features	● Modular component blocks ● Ability to export to a variety of file types
PCB Design Features	● Ability to export to most common file types. ● Ability to generate BOM files in a variety of formats. ● Integration with RS parts catalog for pricing and availability of parts.
Component Library	Large manufacturer component library, as well as community library. The majority of the parts we researched were available in this library, however some had inaccurately laid out schematic models which should not affect the results of the PCB design though.
Ease of Use	Medium, features seemed to be geared toward professional users but easy enough to learn. Only supports Windows.
Cost	Free without limitations on design layers, size is restricted to 1m^2.

Table 15: DesignSpark PCB Research Summary

Chosen EDA Software - The EDA Software suite we ultimately chose to design our Schematic with is EasyEDA, it provides a great balance of easy to use interface with all the features one requires to develop a professional schematic and PCB design. Besides those features we really liked the fact that it is available as a web based tool, making collaborating across the team very easy and that it's parts library is the highest quality and largest we encountered in our research. If we encounter any unforeseen issues we are able to export EAGLE and GERBER files and would import those into AutoDesk Eagle or DesignSpark PCB as those were our second favorites.

Project Circuit Schematic

Figure 5 illustrates our main board circuit, the schematic was organized by grouping components by their function and connecting them to the controller via net port labels. The schematic was created in EasyEDA and utilizes some components that are different then the ones researched initially however those components that differ vary only slightly and typically were picked due to lower price or larger quantities available.

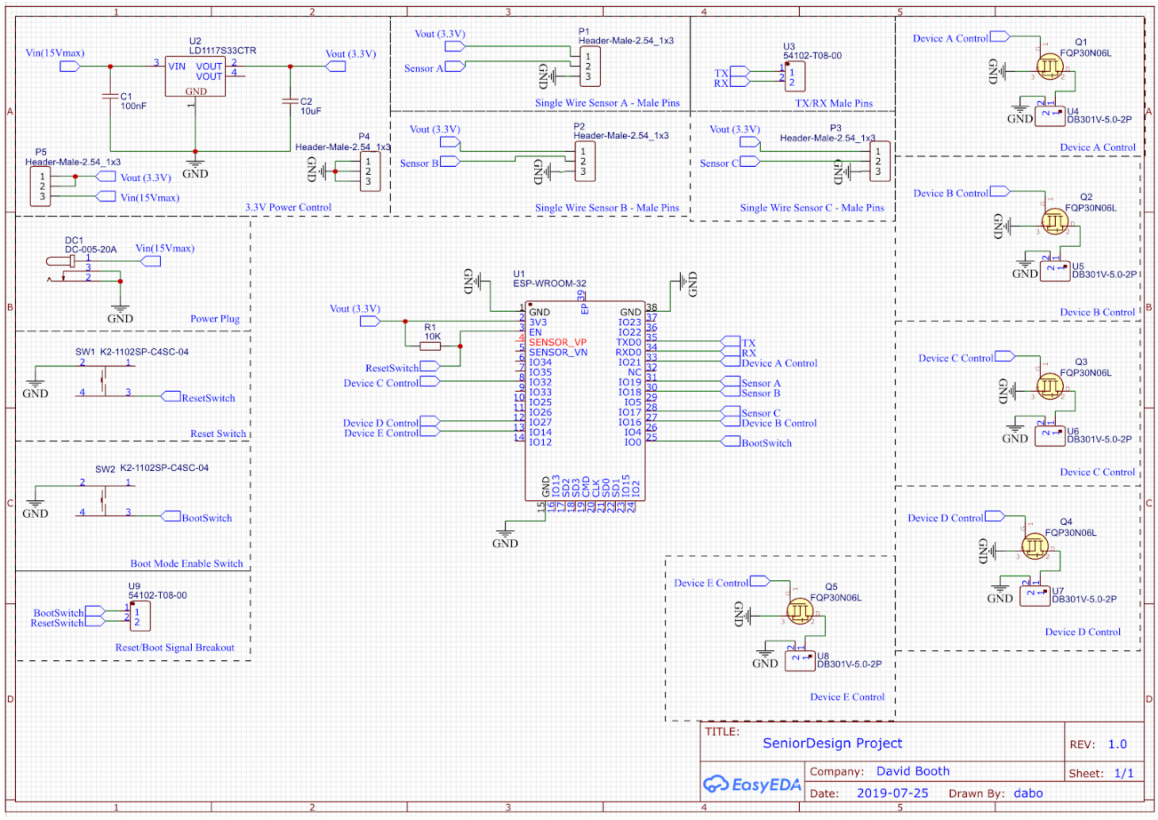

Figure 5: Schematic of Project Circuit Board

2.6.9 Controller Unit

The controller unit, is the heart of the hardware component of this project. It interacts with the various sensors and interprets said inputs to control the operation of other devices such as the humidifier, light and fan. It also is in charge of connecting to the Internet via WiFi and sending & receiving data to and from our web server. This section consists of the research notes associated with the controller, a list of required capabilities expected from the controller, a description of the chosen component and how it satisfies the component requirements, a list of links to important component requirements as well as diagrams, a description of the development boards used for evaluation and development, details about the feasibility of acquiring the chosen component and finally a description of the component will be implemented and utilized in the project.

Component Research - The below components were researched for Controller units, both Microcontroller Units (MCUs) as well as System on Chips (SoCs) that included Wifi modules were researched.

Microchip SAMD21 Cortex-M0+ MCU - Table 16 summarizes our research of the SAMD21 MCU as well as available vendors and unit cost. It is interesting to note that this is the MCU used in the Arduino MKR1010 and MKR1000 which is why it was included in our research.

Feature	Considerations
Processor	SAMD21 Cortex-M0+ 32bit low power ARM processor.
Storage	Internal 256KB Flash
Communication	• UART, SPI, I2C • 10 bit DAC • 38 GPIO Pins
WiFi	Will require external WiFi module.
Development Environment	Moderately easy development due to Arduino support.
Vendors	Mouser
Unit Cost	$3.53

Table 16: Summary of SAMD21 MCU

Microchip ATSAMW25 SoC - Table 17 summarizes our research of the ATSAMW25 System on Chip as well as available vendors and unit cost. It is interesting to note that this is the SoC used in the Arduino MKR1000 which is why it was included in our research.

Feature	Considerations
Processor	SAMD21 Cortex-M0+ 32bit low power ARM processor.
Storage	Internal 256KB Flash
Communication	• UART, SPI, I2C • 10 bit DAC • 15 GPIO Pins
WiFi	802.11 b/g/n/ WiFi (2.4GHz)
Development Environment	Moderately easy development due to Arduino support. Flashing of Arduino firmware most likely difficult.
Vendors	Mouser
Unit Cost	$12.05

Table 17: Summary of ATSAMW25 SoC

Microchip ATmega328P MCU - Table 18 summarizes our research of the ATmega328P MCU as well as available vendors and unit cost. It is interesting to note that this is a very popular MCU that while relatively barebones in it's features still provides great utility.

Feature	Considerations
Processor	8-bit AVR RISC-based processor
Storage	Internal 32KB Flash
Communication	• UART, SPI, I2C • 23 GPIO Lines • 10 bit ADC
WiFi	Will require external wifi module
Development Environment	Easy to setup and develop on, also provides Arduino Support.
Vendors	Mouser
Unit Cost	$1.38

Table 18: Summary of the ATmega328P MCU

Espressif ESP8266 WROOM-02D SoC - Table 19 summarizes our research of the ESP8266 as well as available vendors and unit cost. Specifically we decided to choose the WROOM-02D model as it is a nicely shielded SoC with built in antenna and internal flash memory.

Feature	Considerations
Processor	Xtensa Single-core 32-bit L106 processor
Storage	Internal 2MB or 4MB Flash Memory options are available.
Communication	• 16 GPIO Pins • SPI, I2C, I2S, UART • 10bit ADC
WiFi	• 802.11 b/g/n/ WiFi (2.4GHz) • Onboard Antenna
Development Environment	Espressif proprietary IDE is provided as well as Arduino support is provided by the manufacturer.
Vendors	Mouser (2MB), Mouser (4MB)
Unit Cost	$3.00 (2MB), $3.20 (4MB)

Table 19: Summary of the ESP8266 WROOM-02D SoC

Espressif ESP32 WROOM-32D SoC - Table 20 summarizes our research of the ESP32 as well as available vendors and unit cost. Specifically we decided to choose the WROOM-32D model as it is a nicely shielded SoC with built in antenna and internal flash memory.

Feature	Considerations
Processor	Xtensa dual-core 32-bit LX6 processor
Storage	Internal 4MB and 16MB Flash Memory options are available.
Communication	• 21 GPIO Pins • Multiple SPI, I2C, I2S, UART • 12bit ADC • 2x 8bit DACs • Classic and BLE Bluetooth • Variety of other improved peripheral interfaces and functions.
WiFi	• 802.11 b/g/n/ WiFi (2.4GHz) • Onboard Antenna • 4 Virtual WiFi services that can operate simultaneously
Development Environment	Espressif proprietary IDE is provided as well as Arduino support is provided by the manufacturer.
Vendors	4MB: Mouser, DigiKey, GridConnect 16MB: Mouser, DigiKey, GridConnect
Unit Cost	$3.80 (4MB), $4.50 (16MB)

Table 20: Summary of ESP32 WROOM-32D SoC

Component Requirements - Below is a list that highlights the requirements we have for the controller unit, we used this list to guide our decision making process of the above components.

1. The component should be easily programmed.
2. The component should be easily flashable.
3. The component should either provide built in Wifi connectivity or can easily interact with a WiFi module.
4. The component should have the capability to receive data from and control the components required to achieve the projects features.
5. The component should provide ample flash memory to provide enough room for program space of the current features and possible future features.

The above requirements are derived as follows, the component will require development of custom software to implement the controlling logic required for the project. This software (also referred to as firmware) can be developed using a variety of languages and development environments, the major limiting factor on development environment is the components support for such environments, hence our aim is to find a component that supports a development environment that provides what one would consider a higher level programming language, i.e. provides at least some moderate level of abstraction from the underlying hardware complexities. For this part of the project we will consider programming languages that are C-like or provide object oriented constructs to be sufficiently high level. Besides programming language support we also evaluated community engagement in the development environment and amount of publicly available education resources, the first usually greatly increasing the latter, our aim was to find a development environment with great community support and large collections of publicly available educational resources. And finally we evaluated the tooling provided to aid development both by the manufacturer and community, this includes items such as IDEs, Boardmanagers, flashing scripts, debugging tools and more.

Once the firmware is developed one must find a way to transfer it to the component, various chips provide different mechanisms to achieve this ranging from methods like programming the chip directly over a wire, this is typically referred to as serial programming, to downloading the firmware binary over the air, typically called OTA programming. One can tell that there are a variety of benefits and possible failure points to consider between just the two example methods mentioned above, additionally besides choosing a transfer method the component must typically be made aware that this type of transfer is about to happen, this is typically done by switching the component into a bootloader or flashing mode. For this requirement we evaluated the setup complexity, i.e. number of components and tools involved in switching the component into bootloader mode as well as steps required on the programming computer to establish a connection successfully.

Furthermore we also evaluated reliability, or how successful the previous method is in successfully programming the chip. To ultimately come up with this “ease of flash-ability” score we studied reviews, case studies and other publications of first hand experiences with a variety of different components in both academic and production environments. Once narrowed down we purchased development boards that had the components we wanted to further evaluate and tested the methods we theorized would be able to provide the greatest ease of flash-ability. This is further described in the below section titled *“Development Board Description & Acquisition”*.

Various features of this project require that the component communicates with a web server via the internet. To achieve this the controller component must be able to connect to a WiFi network. In terms of WiFi featureset we considered ability to connect to 2.4GHz and 5GHz bands, ability to support the 802.11 b, g and n protocols as well as support for at least one common security protocol such as WEP. Additionally we evaluated the benefits and drawbacks of a System on Chip (SoC) design where a WiFi module is enclosed directly with the processor and other peripherals in a single chip, to a design where the processor and WiFi modules are independent chips that communicate with each other over wire. We considered SoC designs to be beneficial as it reduced design complexity in both hardware and software of our project. Also note that a more detailed evaluation focusing on WiFi only modules is located in section 2.6.13.

The two core actions of the controller are to measure various environmental parameters via external sensors and then activate or deactivate related devices to maintain said values within a certain range. Hence a core requirement of controller component is the capability to communicate with all sensors and devices. We studied and evaluated this capability mainly across two dimensions, firstly does the controller provider enough physical lines to communicate and control, and secondly does the controller support the communication protocols to communicate with the sensors and devices. To evaluate across the first criteria we calculated an average number of GPIO pins that are required to cover the sensors and devices described in the core, advanced and stretch goal feature sets. We noted that we require 4 data lines, one for each sensor, as well as 5 device control lines, hence a total of 9 GPIO lines however to provide room for some error as well as potential future requirements we aimed for a component with at least 15 GPIO lines. We also noted that since the majority of the devices require a simple enable or disable signal we could if necessary use a shift register to decode a single serial signal into multiple control lines and hence only utilize a single GPIO line, however our aim is to avoid this mechanism for the sake of reduced design complexity. To evaluate the second criteria we wanted to ensure that the component has the capability to communicate via the most common protocols and has the required converting mechanisms built in, we considered the following communication protocols as a minimum requirement

SPI, I2C, I2S, and UART. And we considered a 10 bit resolution analog to digital converter a minimum requirement.

Finally the firmware to operate the controller needs to be stored on flash memory. We evaluated the flash memory requirement across two dimensions, firstly will the flash memory be internal to the chip or will it be connected externally and secondly ensuring that enough storage space is available to house the code for the core, advanced and stretch goal features. For the first criteria we saw benefits with both the internal and external option, the main benefits for internal were reduced complexity of the board design and interaction, and the main benefits for external where ease upgrading if additional space is desired in the future. For the second criteria we estimated that our binary stub size by looking at the file sizes of compiled projects that we deemed to be similar in complexity and architecture. Our estimate came to 3MB of program size, to allow for error in our estimate and future features we set our minimum requirement to 4MB of flash memory but would choose greater options if available.

Chosen Component Description - The ESP32 WROOM-32D System on Chip (SoC) by Espressif, from here on referred to as simply ESP32, was chosen to be the Controller component of this project. The below describes why the component was chosen and evaluates the component's benefits against the component requirements stated above.

1. The ESP32 is SoC that houses a variety of components inside it, among other features it most notably provides processing, wifi, bluetooth and flash memory all in a single unit. This allows the component to deliver great value both in financial terms as well as reduced complexity due to fewer part count in our overall BOM.

2. The ESP32 can be programmed using the Arduino Development Environment, reducing development complexity and providing an abundance of community provided educational resources. Hence satisfying requirement Comp Req. #1.

3. While the component originally either comes with a proprietary firmware provided by Espressif or in some cases no firmware, it is easily flashable to run the Arduino ESP32 Core framework, thus satisfying requirement Comp. Req. #2.

4. The ESP32 provides 802.11 b/g/n/ WiFi on the 2.4GHz band, satisfying requirement Comp. Req. #3.

5. The ESP32 provides 21 general purpose input/output (GPIO) lines, a 12bit resolution analog to digital converter (ADC), two 8 bit digital to analog converters (DACs) and support for various communication protocols such as SPI, I2C, I2S, and UART. The aforementioned features should provide plenty of capability to control and interact with a variety of sensors and other components. (Comp. Req. #4)

6. The ESP32 comes with either 4MB or 16MB of internal flash memory, both options would deliver plenty of program space for implementation of core, advanced and stretch goal features.

Component Resources - The following section contains tables, diagrams and links to detailed resources describing the component and it's operation as it relates to its implementation in this project. Additionally Figure 6 can be found in the appendix depicting the ESP32 pin layout that was utilized in the schematic creation.

Recommended Operating Conditions of the ESP32 - Table 21 displays the minimum, ideal and maximum recommended operating conditions. Specifically a power value below the minimum specified vale may cause the component to enter a brown-out condition where power values above the recommended maximum values may cause physical damage to the component.

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power Supply Voltage	2.7	3.3	3.6	V
Ivdd	Current delivered by external power supply	0.5	N/A	N/A	A
T	Operating Temperature	-40	N/A	85	Celsius

Table 21: Recommended operating conditions of the ESP32

Education and Reference Resources for the ESP32 - Table 22 below lists links to educational and reference resources for the general ESP32 MCU as well as the WROOM-32D SoC module that we decided to utilize. Additionally helpful similar components like breakout boards containing the module are included.

Description	Link
ESP32-WROOM-32D Datasheet for both 4MB and 16MB variations.	https://www.mouser.com/datasheet/2/891/esp32-wroom-32d_esp32-wroom-32u_datasheet_en-1365844.pdf
ESP32 Generic Datasheet	https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
Arduino ESP32 Core	https://github.com/espressif/arduino-esp32
Sparkfun ESP32 Breakout Board	https://www.sparkfun.com/products/14689
Graphical Datasheet of ESP32 Breakout Board	https://cdn.sparkfun.com/assets/learn_tutorials/8/5/2/ESP32ThingPlus_GraphicalDatasheet.pdf
Arduino Software Setup Guide for ESP32	https://learn.sparkfun.com/tutorials/esp32-thing-plus-hookup-guide?_ga=2.45451684.1462509115.1562192986-1370538347.1559912484#software-setup
Serial Protocol Definition of the ESP32 Bootloader	https://github.com/espressif/esptool/wiki/Serial-Protocol
Entering the ESP32 Bootloader	https://github.com/espressif/esptool/#entering-the-bootloader
ESP32 Arduino Core Boardmanager Instructions	https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md
ESP32 Bootloader Message Codes	https://github.com/espressif/esp-idf/blob/release/v3.0/components/esp32/include/rom/rtc.h#L80

Table 22: Educational and Reference Resources for the ESP32

Overview of Hardware Specifications of the ESP32 - The information in the table 23 was summarized from the ESP32-WROOM-32D datasheet, and gives a great overview of the hardware specifications and included peripherals. It is important to note that the Integrated Flash Memory is listed as 4MB, however version of 8MB and 16MB are available as well.

Categories	Items	Specifications
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 µs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4MB, 8MB and 16MB available
	Operating voltage/Power supply	2.7 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ +85 °C

Table 23: Overview of ESP32 Hardware Features and Parameters

Development Board Description & Acquisition - For component evaluation as well as development and prototyping a development board, also known as breakout board for both the ESP32 and ESP8226 were purchased.

The development board for the ESP32 was ordered from sparkfun.com for $20.95. This board includes the Espressif ESP32-WROOM-32D mounted on a PCB, however along with the controller the board also includes a variety of convenient peripherals designed to ease development, some of the notable ones include:

- USB Programming Interface, including a USB microB connector wired to a USB to serial converter which in turn is wired to the appropriate ports on the ESP32.
- Easily accessible GPIO pins.
- A Reset Button
- A general purpose button pre-wired to GPIO 0
- A 4 pin JST connector designed to power the board from a LiPo battery.
- A Qwiic connector, a proprietary system designed by Sparkfun designed to easily establish I2C communication.
- A general purpose LED, connected to GPIO 13
- A LiPo battery charging circuit
- A voltage regulator, takes the input voltage from USB, LiPo or the VIN pin and regulates to 3.3V

The development board for the ESP8226 was also ordered from sparkfun.com at a price of $17.95, this board has the Espressif ESP8226EX mounted on it. It is important to note that this is not the exact same chip that is being considered for the final product, the component considered for the final product is the ESP8226-WROOM-02D. The major notable differences are that the WROOM-02D version of the chip includes internal flash memory and provides a built in antenna as well as protective shielding. The development board provides the following convenient features:

- 512KB of external flash memory, wired appropriately to be used by the ESP8226.
- USB Programming interface, consisting of a USB microB connector, connected to a USB to serial converter, which is wired to the appropriate pins on the controller.
- Easily accessible GPIO pins.
- A On/Off Switch
- A general purpose LED connected to GPIO 5
- A u.fl antenna connector
- A voltage regulator that either takes the USB voltage or VIN pin as inputs and regulates to 3.3V to be used by the controller.

ESP32 Acquisition - The ESP32 SoC will be purchased from any of the vendors listed in the above Component Research table, depending on availability the 16MB version is preferred, however none of the functionality or feature set should be affected if the 4MB version is chosen instead. At the time of writing, the stock counts for this part are collected in Table 24.

	Mouser	**DigiKey**	**GridConnect**
4MB	3,082	9,830	In Stock (Exact part count not available)
16MB	0 (2 Week Lead-Time)	1,192	In Stock (Exact part count not available)

Table 24: Vendor availability for ESP32 SoC

Implementation Description - This following sections describe the processes involved in implementing the controller, it's features and how it interacts with its neighbouring components. The section consists of the following topics: flashing the chip, programming the chip, a high level components diagram indicating the interactions of the controller and finally a high level software components description and class diagram.

Flashing the Chip - The ESP32 comes from the factory with a proprietary firmware installed, this firmware provides support for interaction through Hayes commands (also known as AT commands) over the UART pins of the chip. If the module where to be used as a wifi module only by an additional processor then this basic AT command set usually suffices. However since this project will utilize the ESP32 both as main processor and WiFi a more sophisticated development environment is needed. Specifically the Arduino development environment was chosen for this project as it provides a high level language C++ derived language, as well as great community support and educational resources. Luckily Espressif provides an Arduino Core framework designed for the ESP32 under the GPL v2.1 license, this framework implements almost all of the Arduino standard libraries with only a few exceptions, most notably it is currently missing support for the analogWrite() method, however workarounds are provided.

In order to install, also known as flash, the chip with our custom firmware one must reset the chip into bootloader mode and upload the new firmware via the serial pins of the chip. Specifically in order to enable bootloader mode one must interact with the strapping pins of the chip, the following pins are the designated strapping pins GPIO0, GPIO2, GPIO5, MTDI, MTDO.

Strapping Pins - This section describes the Bootloader related pins and their configurations to enter various booting modes on the ESP32. We have summarized the strapping pin configuration that are needed by our project in table 25.

Voltage of Internal Flash Memory			
Pin	Default	3.3V	1.8V
MTDI	Pull Down	0	1
Booting Mode Selection			
Pin	Default	SPI Boot	Download Boot
GPIO 0	Pull Up	1	0
GPIO 2	Pull down	Don’t care	0
Debug Log Print Verbosity			
Pin	Default	Verbose	Silent
MTDO	Pull up	1	0

Table 25: ESP32 Strapping Pin Configurations

If GPIO0 and GPIO2 are held low during reset, the chip will enter flashing mode also known as serial bootloader mode which allows it to be ready to download the new firmware into its flash. Additionally during this reset if the MTDI pin is driven high the chip will assign 1.8V to power the flash memory, this pin is by default connected to a pull down resistor and if low will use the default 3.3V. By default the bootloader will send boot messages to aid the programmer, these messages contain reason codes that are explained further in M27, by driving the MTD0 pin low one can silence these boot messages.

In order to set the appropriate signals one must derive a reset circuitry that holds GPIO0 low during reset of the chip, since GPIO2 is connected to a pull down resistor it is low by default and hence does not need to be explicitly held low. A simple manual implementation of this is to install a button that pulls GPIO0 low. With the chip in bootloader mode it waits for a UART Serial connection to begin downloading the new firmware, to do this one must connect a computer to the serial lines of the chip. A common method, and the one we used for this project, is to use USB to connect to the computer and then utilize a USB to serial converter on the PCB to translate the signals from the computer to the serial lines on the chip. Espressif provides a python script called *esptool.py* that

contains various functions to aid in uploading firmware to the chip once it is in bootloader mode.

Programming the chip - The software that controls the chip, also called firmware, was developed using the Arduino IDE. The languages supported by the Arduino IDE and the ones utilized to develop the software for this project are C and C++, it is interesting to note that the Arduino IDE uses a slightly different code structure compared to the traditional C or C++ specification. Besides an IDE the Arduino Project provides a variety of libraries that simplify peripheral interactions, for example our project will utilize the WiFi library to find and connect to a WiFi network as well as send requests to a web server. Additionally through the Arduino BoardManager definition provided by Espressif we were able to upload the program through the flashing scenario described above directly from within the Arduino IDE.

Controller Interactions - This section explains the interactions the controller unit engages in, they are grouped into 3 categories: Sensor Readings (green), Device Control (yellow), Internet Interactions (blue). Figure 7 gives a visual representation of the categories.

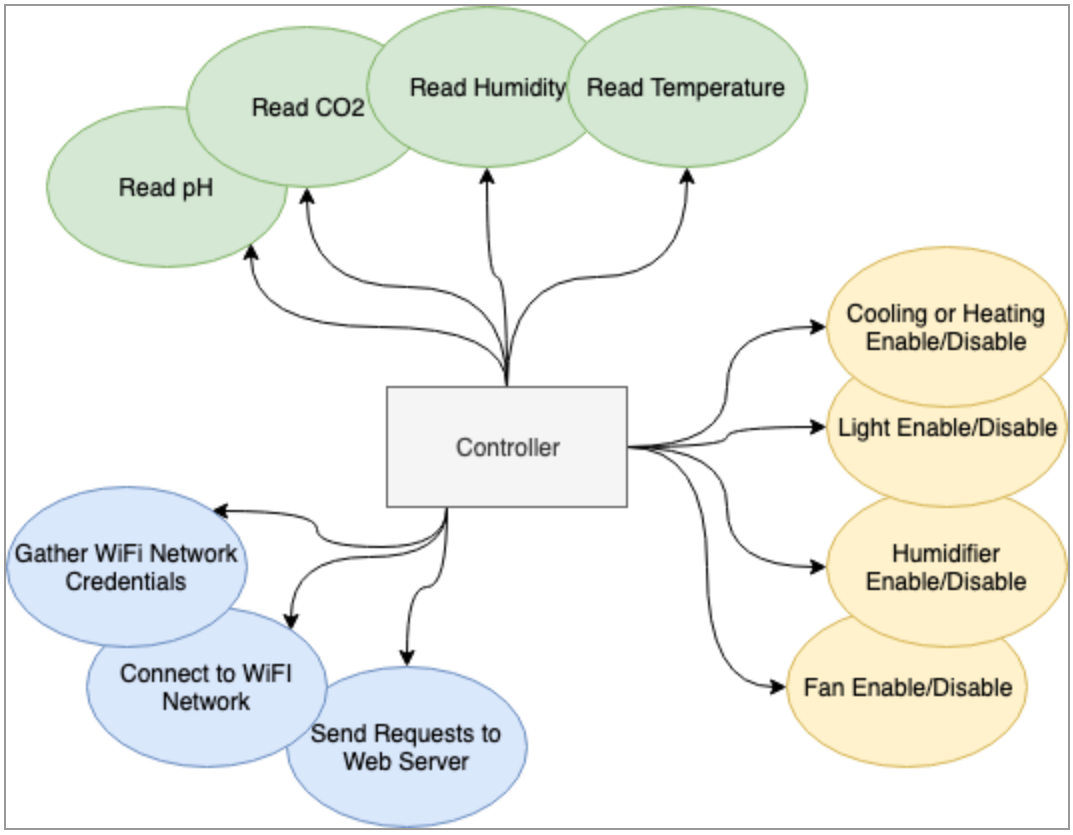

Figure 7: Overview of Controller interactions

Sensor Readings involve reading and interpreting digital or analog inputs on one of the GPIO lines, and then based on the given reading triggering the desired action or inaction, specifically the reading and interpretation logic for Temperature, Humidity, CO2 and pH sensors are required. Note also that we may combine the Temperature and Humidity reading into a single sensor if parts availability allows for it.

Software/Firmware Components - This section contains a high level description of the software architecture that will operate on the chip. Functionality is organized in into dedicated classes/libraries that share a similar purpose. The major sections are Sensor Classes (green), Device Classes (yellow), Internet Class (blue) and Controller Class (purple). Their purpose and functionality is described below.

Sensor Classes - The Sensor Classes encapsulate all functionality related to Sensors, shared functionality that is common across all sensors is abstracted out into a SensorBase Class and can be overridden as necessary by the inheriting Sensor Classes if needed. Figure 8 displays the Class Diagram for the Sensors.

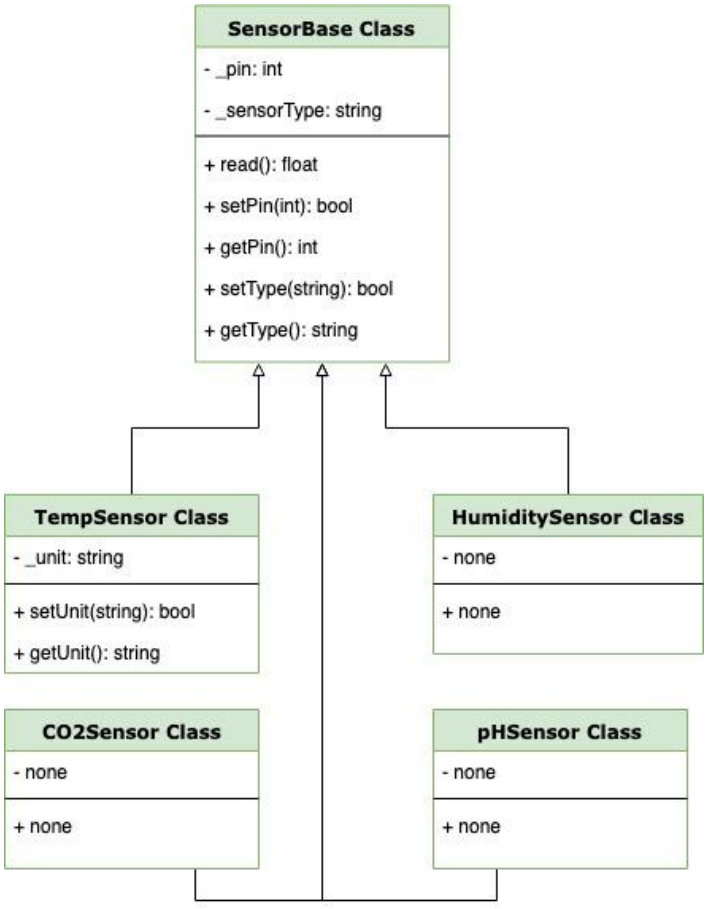

Figure 8: Class Diagram of Sensor Classes

The common functionality and members that are abstracted into the SensorBase Class are as follows:

- ***_pin:*** A private integer defining the GPIO pin to be used to communicate with the sensor.
- ***_sensorType:*** A private string defining the type of sensor this is.
- ***read():*** A public method that reads the value from the sensor and returns it. This method may be overridden by the inheriting Sensor Classes to include sensor specific conversions of the readings.
- ***setPin():*** A public method that sets the pin to be used to communicate with the sensor, this method will return a boolean indicating if the given pin is valid and is now being used.
- ***getPin():*** A public method that returns the pin integer currently in use by the sensor instance.
- ***setType():*** A public method that accepts a string input to declare the type of sensor for this instance, this method will return a boolean indicating that the given type is valid and is now being used.
- ***getType():*** A public method returning the type of this sensor instance.

Additionally the TempSensor Class defines the following members and methods:

- ***_unit:*** A private string that defines which unit the temperature is being returned in. Valid units are Fahrenheit and Celsius.
- ***setUnit():*** A public method that accepts a string to define in which unit the measured value returned by the read() method should be in. A boolean is returned to indicate if the given unit is valid and now being used.
- ***getUnit():*** A public method that returns the unit that is being used by this instance of the Temperature Sensor Class.

Device Classes - The Device Classes encapsulate all functionality related to Devices, shared functionality that is common across all devices is abstracted out into a DeviceBase Class and can be overridden as necessary by the inheriting Device Classes if needed, the class diagram is depicted in Figure 9.

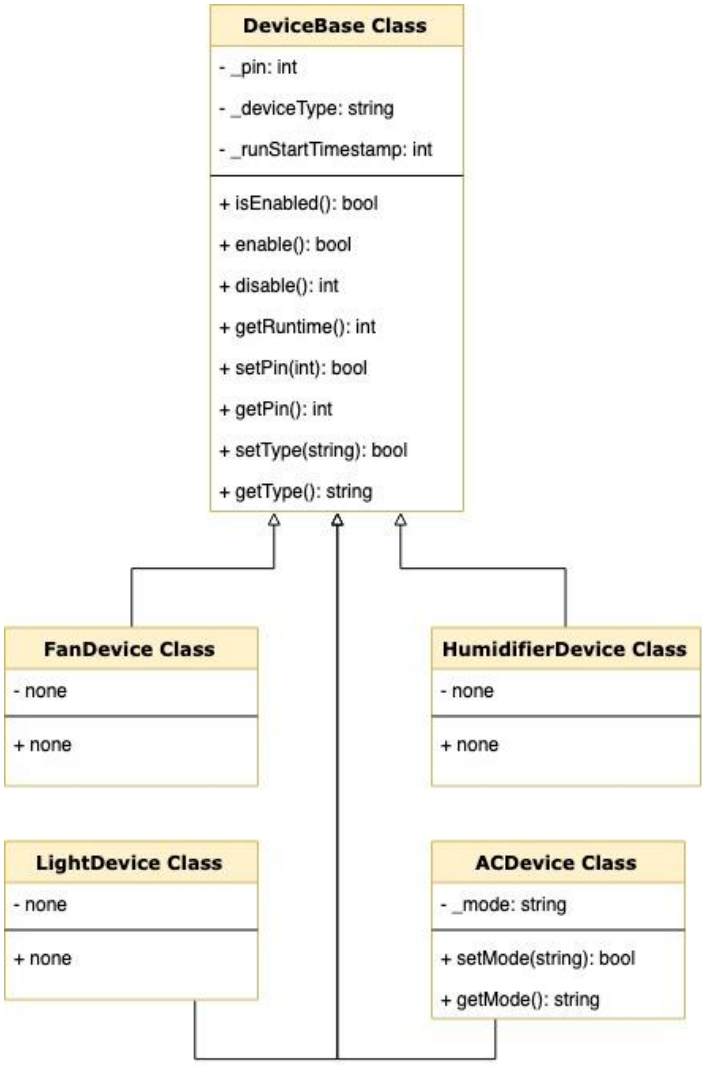

Figure 9: Class Diagram depicting Device Classes.

The common functionality and members that are abstracted into the DeviceBase Class are as follows:

- ***_pin:*** A private integer defining the GPIO pin to be used to communicate with the device.
- ***_deviceType:*** A private string defining the type of device this is.
- ***_runStartTimeStamp:*** A private integer holding a unix timestamp of when the device was enabled. This value will later be used when the device is being disabled or when the getCurrentRuntime() is being called to determine for how long the device was enabled.
- ***isEnabled()***: A public method returning a boolean that indicates if the device is currently enabled or not.
- ***enable():*** A public method used to enable the device, it returns a boolean indicating that the device is enabled successfully, it may contain logic that prevents the device from enabling in which case it would return false.
- ***disable():*** A public method used to disable the device, it returns an integer which is the amount of time in seconds the device was enabled for.
- ***getRuntime():*** A public method that returns an integer which is the amount of time in seconds the device has been enabled for, if the device is currently disabled it will return 0.
- ***setPin():*** A public method that sets the pin to be used to communicate with the device, this method will return a boolean indicating if the given pin is valid and is now being used.
- ***getPin():*** A public method that returns the pin integer currently in use by the device instance.
- ***setType():*** A public method that accepts a string input to declare the type of device for this instance, this method will return a boolean indicating that the given type is valid and is now being used.
- ***getType():*** A public method returning the type of this device instance.

Additionally the ACDevice Class defines the following members and methods:

- ***_mode:*** A private string defining the operating mode of this ACDevice instance. Valid modes are either heating or cooling.
- ***setMode():*** A public method that accepts a string which defines the operating mode of this ACDevice instance. It returns a boolean indicating that the passed in operating mode is valid and is now being used by the device.
- ***getMode():*** A public method that returns the current operating mode of this ACDevice instance.

Internet Class - The Internet Class encapsulates all internet related functionality, mainly it abstracts wifi related activities and web server interaction functionality, it's class diagram is shown in Figure 10. During initialization of this class it will check the EEPROM for any cached WiFi Credentials and if so set them immediately to the private member variables.

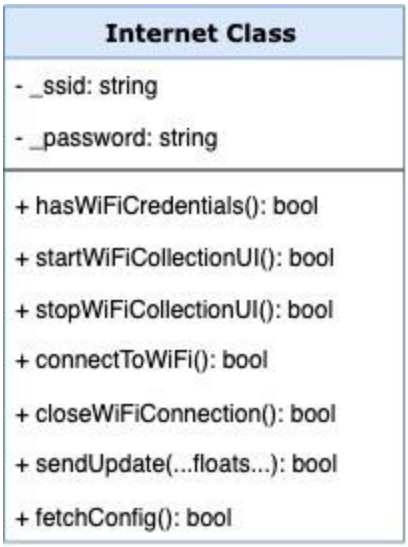

Figure 10: Class Diagram for Internet Class

The Internet Class consists of the following members and methods:

- ***_ssid:*** A private string containing the ssid of the wifi network to connect to.
- **_password:** A private string containing the password of the wifi network to connect to.
- ***hasWiFiCredentials():*** A public method that checks if the Internet Class instance has an ssid and pw, it will return false if either of the strings are empty. It is also important to note that during initialization of this class it will fetch previously stored WiFi credentials from EEPROM, so this method can be used to check of their existence.
- ***startWiFiCollectionUI():*** A public method that starts a WiFi access point and hosts a web server containing a User Interface with WiFi ssid and password inputs. It returns a boolean indicating that the AP was successfully started.
- ***stopWiFiCollectionUI():*** A public method that stops the WiFi access point, and returns a boolean that indicates that the access point was successfully terminated.
- ***connectToWiFi():*** A public method that attempts to establish a WiFi connection to the internally set ssid. If the connection process times out, the connection could not be established, or hasWiFiCredentials() returns false, then the method will return false and hence not establish a WiFi connection.

- ***closeWiFiConnection():*** A public method that attempts to terminate the WiFi connection, it will return a boolean indicating if the termination was successful or not.
- ***sendUpdates():*** A public method that accepts a series of floating point and integer inputs, reflecting sensor values, device runtimes and device enable/disable statuses. It submits those parameters as a web request to the web server and returns a boolean indicating the success or failure of the request.
- ***fetchConfig():*** A public method that makes a web request to the web server to query for any configuration updates that the user wishes to make. It returns a boolean indicating success or failure of the request.

Controller Class - The controller class houses the functionality and logic that binds the rest of the classes together to make an operational controller, it is depicted in Figure 11. Therefore it handles interactions such as reading the temperature periodically and enabling heating or cooling based on the sensor results, periodically checking for configuration updates from the server and applying them, determining if the WiFi collection user interface should be shown as well as many other controlling related functions.

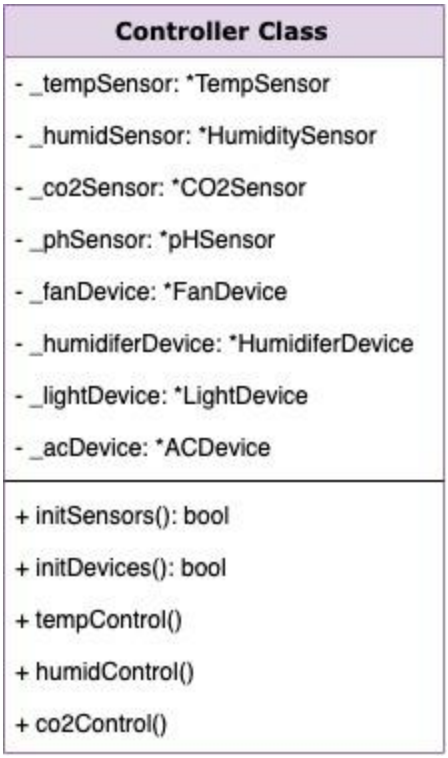

Figure 11: Class Diagram for Controller Class

The members and methods of the Controller Class are as follows:

- ***_tempSensor:*** A private pointer to the temperature sensor instance used by this controller.
- ***_humidSensor:*** A private pointer to the humidity sensor instance used by this controller.
- ***_co2Sensor:*** A private pointer to the CO2 sensor instance used by this controller.
- ***_phSensor:*** A private pointer to the pH sensor instance used by this controller.
- ***_fanDevice:*** A private pointer to the fan device instance used by this controller.
- ***_humidiferDevice:*** A private pointer to the humidifier device instance used by this controller.
- ***_lightDevice:*** A private pointer to the light device instance used by this controller.
- ***_acDevice:*** A private pointer to the AC device instance used by this controller.
- ***initSensors():*** A public method that initiates the Sensor instances and sets the respective pointers to them on the controller class. It returns a boolean indicating if the process was successful or not.
- ***initDevices():*** A public method that initiates the Device instances and sets the respective pointers to them on the controller class. It returns a boolean indicating if the process was successful or not.
- ***tempControl():*** A public method that initiates the temperature control logic, in which a temperature reading is taken and a decision is made to set the AC Device mode to either heating or cooling and to either enable or disable the AC Device and Fan Device.
- ***humidControl():*** A public method that initiates the humidity control logic, in which a humidity reading is taken and a decision is made to either enable or disable the humidifier and fan device.
- ***co2Control():*** A public method that initiates the co2 control logic, in which a co2 reading is taken and a decision is made to either enable or disable the fan device.

2.6.10 WiFi Module

The following components were researched and evaluated in order to provide insight into standalone WiFi modules that the controller component could interact with, if a controller component that does not include a WiFi module inside it's chip is chosen. This research is summarized in table 26.

Component	Feature Considerations	Unit Cost	Vendors
Microchip ATWINC1500	• Supports 2.4GHz Band • 802.11 b/g/n Support • Serial Interface supporting SPI • UART for debug only • WEB, WPA, WPA2 Security Protocols	$9.41	Mouser
ESP8266 (utilized as stand alone WiFi Module)	• Supports 2.4GHz Band • 802.11 b/g/n Support • Simple AT commands to control WiFi operation over UART • WEB, WPA, WPA2, TKIP and AES Security Protocols • IPv4, TCP, UDP, and HTTP network protocols supported	$3.00	Mouser Mouser
ESP32 (utilized as stand alone WiFi Module)	• Supports 2.4GHz Band • 802.11 b/g/n Support • WEB, WPA, WPA2, WPA2-Enterprise, WPS, TKIP and AES Security Protocols • SoftAP support • 4 virtual WiFi interfaces that can be used simultaneously. • Simple AT commands to control WiFi operation over UART • IPv4, TCP, UDP, and HTTP network protocols supported	$3.80	Mouser DigiKey GridConnect

Table 26: Summary of WiFi module research.

Chosen Component Description - While all 3 components would be sufficient for our connectivity requirements, the ESP32 seems to provide the most featureset and the most up to date features while only costing $3.80. Especially it's capability for simultaneously operable virtual WiFi interfaces would allow us to offer simplified configuration flows for the user via SoftAP as well as the option to connect the web or mobile application directly via the software enabled access point to the nursery, without having to disconnect the nursery from its internet connection.

2.6.11 Web Application Software Framework

After consideration of different methods of communicating with the controller unit, a user will use a website to login and interact with the Nursery. The database is comprised of statistics from the user's active grows.

Component Research - Research regarding the different web stacks and applications that were evaluated can be found in sections 2.4.1.4 and 2.4.3.2.

Component Requirements

1. The Component should allow the user to control the settings on the Nursery through a device that can access web pages through an internet connection.
2. The Component will be able to receive statistics indirectly from the controller and create graphs and charts based on those statistics.
3. The Component should allow the user to view statistics on how the Nursery is performing within a given set of boundaries.
4. The Component should allow the user to view statistics on the performance of the growing mushrooms.
5. The Component should have the ability to send and receive data in order to accomplish it's given tasks.
6. The Component should have the ability to send/receive notifications when conditions in the Nursery are not ideal or to the users' specifications.
7. The created User Account needs to take in as an input a specific identification number from the Nursery.
8. The Component will not allow unauthorized access to a Nursery that is not associated with the logged in account.
9. The Component needs to create a report of any errors that occur within the user interface.
10. The Component will allow the user to opt in or out to any of the notification methods.
11. The Component must display everything to the user in a clear and understandable manner.
12. The Component must be responsive and react accordingly to a user's commands.

13. The Component will use cookies and allow the user to stay logged in for a given amount of time on the same device.
14. The Component will contain an instructional dialog window that educates the user on how to navigate the features.

Chosen Component Description - This Component was decided upon because of the ability for the user to interact with the Nursery from anywhere with an internet connection and a device that supports interacting with web pages.

This Component will be designed using a modified MEAN stack. The modification will be that the A, which stands for AngularJS, will be replaced with React which is commonly known as the MERN stack. This will provide the user with a very clean and easy to use interface. The user will be able to view statistics and interact with controls with simple touches of buttons and will not have to write any hardware code to interact with the controller unit.

Because of the open source nature of the technologies, the only costs will be for hosting services. The hosting services will be acquired through account setup with a UCF knights email account. The open source technologies used to build this component can be acquired as downloads from the owners of the respective technologies. These services include hosting for the database component and the API component. The costs associated with the web application framework can be viewed in Table 27.

Component	Feature Description	Unit Cost	Vendor
Amazon RDS	Database hosting	Free year with student account	Amazon Web Services
Amazon API Gateway	API hosting	Free year as new AWS customer	Amazon Web Services

Table 27: Hosting Service cost association.

Implementation Description - The main focus of the software component is to provide a clean and easy to use interface for the user**.** The application contains four different web pages which are the main menu/login screen, the account setup screen, the control page, and the statistics display page. The control page is dynamic and the user will choose from three view options in order to operate the controls. The setup of this application can be seen in Figure 12.

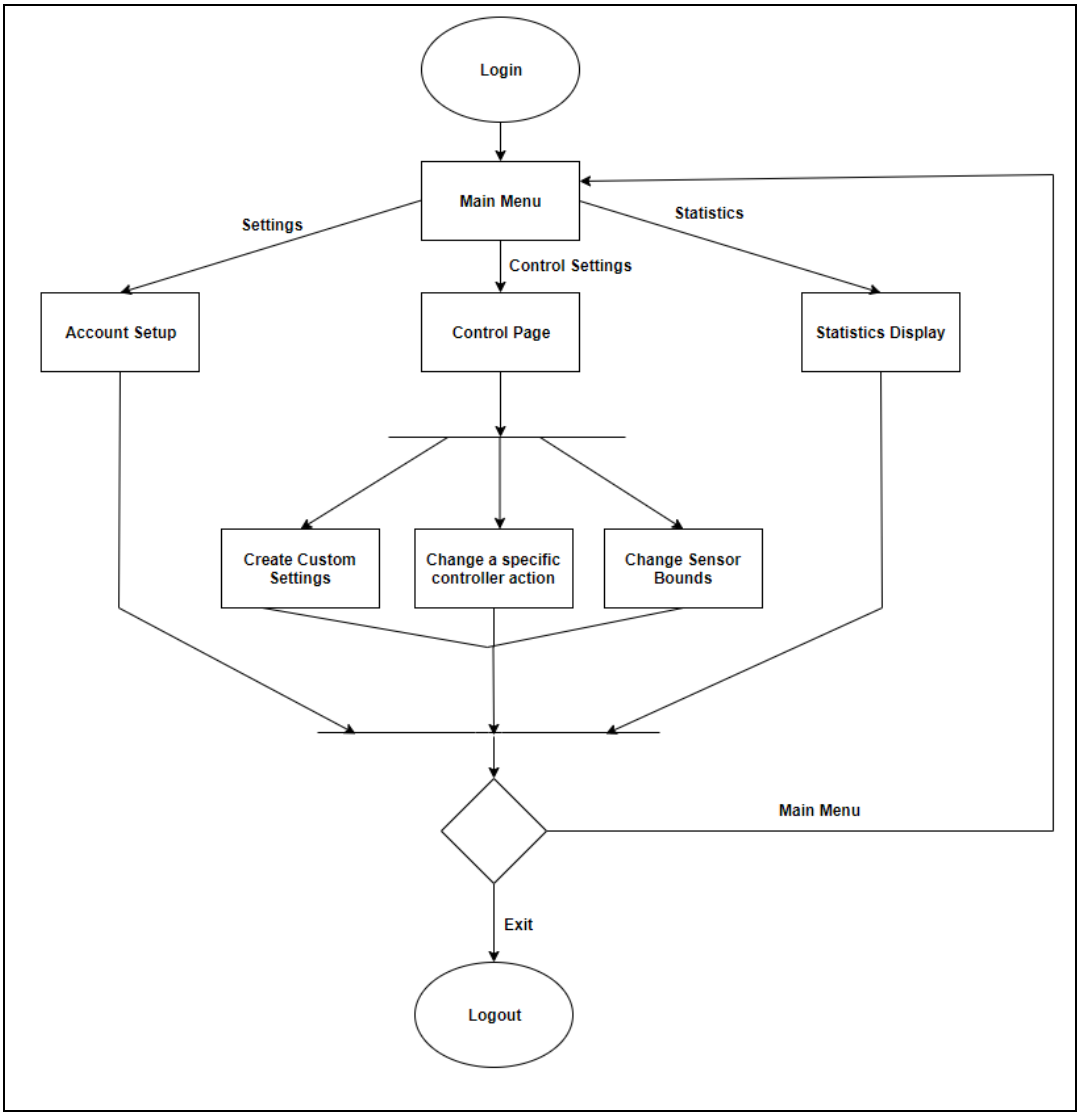

Figure 12: User Interface Flow Chart

Management and Organization - The management style of the software portion is one of the most crucial aspects of the development process. Because of all the moving parts it is easy for components of the code to be written without the communication for another component in mind. Because of these complexities, this project will be using industry-level software development practices. The management areas for this portion of the project will be the software development life cycle, configuration management, and effective coding standards. Each of these areas will have strict guidelines that the developer must follow to create effective code that will produce the desired result.

The software development life cycle is the overall process of planning, creating, and testing software systems. Using this guideline we aim to produce effective software that accomplishes all set goals and do so according to a given timeline. The lifecycle can be split up into phases and these phases can be broken down

into tasks. These tasks include things like software design, reviewing requirements, prototyping, testing, and deployment. The software development life cycle provides an organization to the order and length of the phases that encompass these tasks. A key feature of the software development life cycle is the availability to change. This project will be using short development cycles that have flexible schedules to be able to allow for changes in the project that impact the development of the software.

The specific software development life cycle strategy that this team will be utilizing is the Agile Model. Because of the many moving parts and potentially changing requirements, this specific model seemed to fit our project best. This project will also not be built around a long-term plan which is another reason that the Agile model fit so well with our project. Because of the focus on teamwork within the development teams who could be working on the same code pieces, this model also fits well with our team dynamic. We will only have one or two members working on the web and user interface software for the entirety of the project. A small team coupled with constantly changing requirements creates a complex nature to the project that can be navigated really well using the Agile methods. Through the Agile development style, code will be in constant production, review, and testing in order to encourage fast development of software that meets all given requirements.

Because there will be a lot of different code files and packages, a well maintained configuration management system is necessary. All pieces of software will be maintained using a GitHub server. GitHub also serves a purpose in version control, so if anything breaks we can always revert to an earlier version. Documentation describing the code or any processes related to software development will also be stored on this GitHub server. This will allow us to easily manage the software storage and be able to easily diagnose and solve any problems related to the software. Merging tools like Beyond Compare will be used to ensure that things go smoothly during the merging process in order to prevent any time being taken away from the development.

Because this code will be worked on by more than one person, it is important to create consistent coding standards within the project. These standards will help ensure that the person reading or writing the code knows exactly what that code is supposed to do. Comments will be used extensively in the code to detail the name of the project file, the version of that file, and the date that the code was last updated. This tactic is to help create clear communication lines between the team. If the file worked before it was last updated then it will be easy to know who changed the code and what they changed that caused the code to break. This team will be using Camel case for naming of variables and will be using tabs. These simple rules can have a big effect on making the code look clean and organized. Variable names should also represent what that variable is used for, and ambiguous variable names should be avoided. All functions and methods

created should have detailed comments that describe what is happening within the body of that procedure.

2.6.12 Web Server

Since we will be using a MERN stack, the web server was created using Node.js. Node.js allows users to create a web server and web page in a singular environment. Node.js has similar learning curves to other web server applications such as Apache HTTP server. Apache HTTP and similar server applications require understanding of PHP, whereas, Node.js is written completely in JavaScript. This is useful because every other component of the web stack is also written in JavaScript. The team members in the group who created the software for the web application have a good understanding of JavaScript, so this makes the development phase a little clearer than if the server were to be written in PHP. Node.js also has a non-blocking asynchronous design which will make input and output between the microcontroller and user interface much more reliable. This will also allow multiple users to use the software if/when the project needs to scale up.

The server is required to interact with the rest of the system. The overhead view of this process can be described in simple and single task oriented procedures. The procedures will be detailed below.

Receiving a request from the application and translating that request into appropriate variables is the first step needed to be carried out by the server. The request will be time stamped and the identification number of the requesting device will be stored in variables. The server will use this identification number to look up the corresponding user associated with that device. The server receives a post request for sensor values.

The data from the request containing the sensor data will be stored in the database. The data will be compared against the boundaries that are set for the sensors. If the sensor data is outside the range for the given data then instructions will be sent to the controller unit detailing what needs to occur to have the current condition operate within bounds.

The application always store the latest request in order to compare against new requests. If a new request comes in and requests for conditions that are not within bounds then the server will send instructions to the application to initiate a diagnosing process. This process checks to see if there is a change in mushroom species within the Nursery. If all conditions are the same as the previous request, then the user will be sent a notification that contains an override button allowing the user to set the conditions outside of the bounds.

The server receives these requests periodically as the checks for the sensor values will be done every so often by the controller. Only when the user is trying to set a condition outside of given bounds or if conditions are outside a set of given bounds, will the user be sent a notification from the server.

2.6.13 Database

To keep consistent with the web stack we are using, this project uses a non-relational database as opposed to a relational database. A non-relational database keeps data in collection of JSON documents which will make working with data on the front end very simple. Specifically, this project will be using MongoDB. MongoDB allows for quick implementation and does not have a large learning curve when it comes to working with it's API. MongoDB is also compatible with Node.js and we will be able to use helper modules created using JavaScript to access information.

The database is broken down into tables. MongoDB does not store data in tables but instead uses documents since it is a non-relational database. The database will store a document for each user of the system. The database will take in user information that includes name, email, password, and phone number. The email and phone number are taken because this is where the notifications will be sent if the user opts in for those forms of notifications. The user will be assigned a device ID which corresponds to their Nursery. The device ID will serve many purposes in identification, especially when it comes to sensor data. Accordingly, the device ID will also be a property of each sensor.

Also stored in the database, are standard settings for certain mushroom species. A mushroom species variable will be assigned to a User. This way if the user changes settings accidentally or goes to far experimenting, they can always set the Nursery back to recommended settings for that species of mushrooms.

The documents that store the mushroom species settings, are statically written and entered into the database. Careful research will be performed on the ideal conditions for the mushroom species. The settings will be based on this research. Figure 13 depicts an entity relationship diagram detailing the structure and connections between each component.

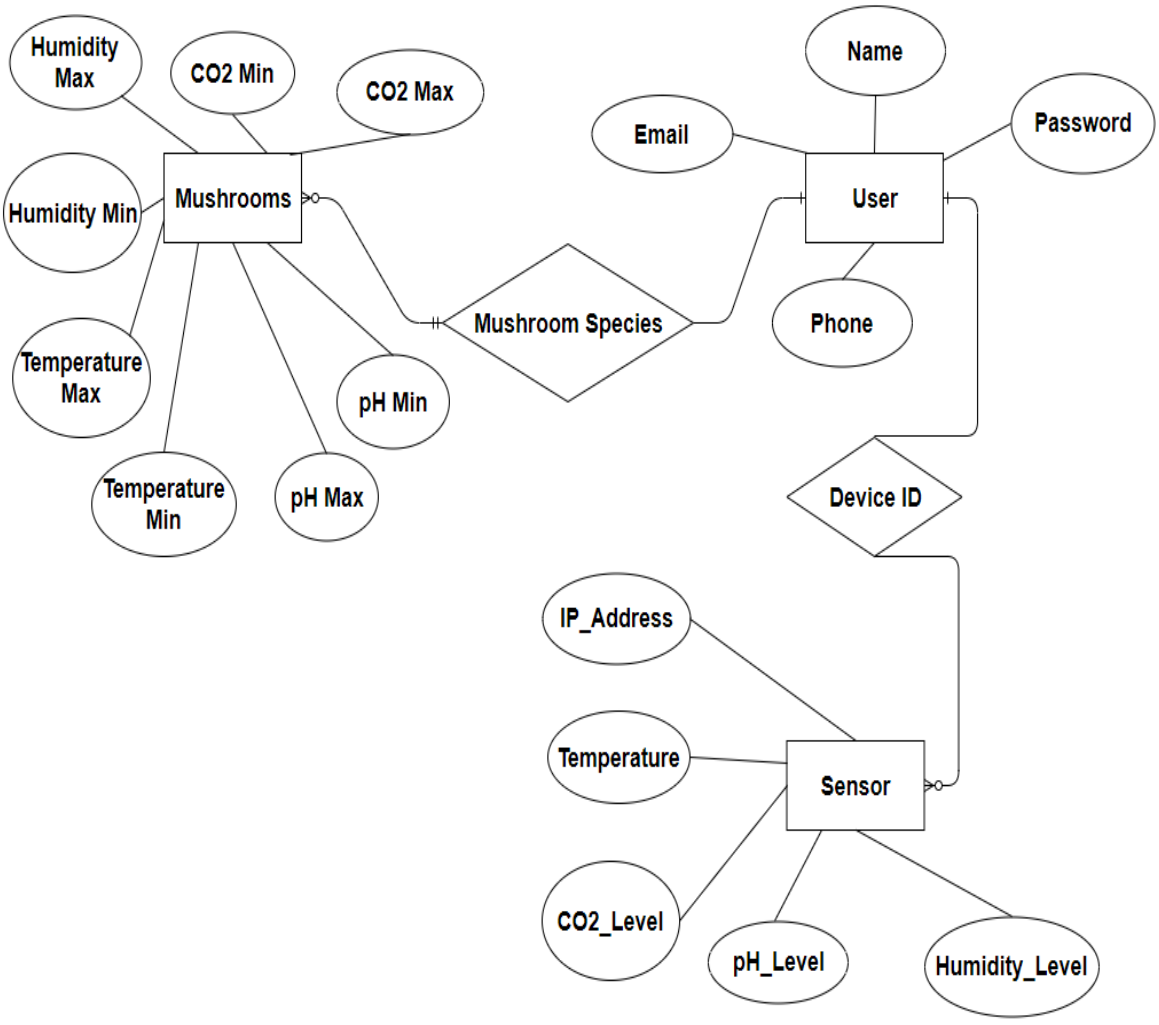

Figure 13: Mushroom Nursery Entity-Relationship Data Model

2.6.14 pH Level Sensor

To measure the pH Level of the soil of the mushroom, we explored the use of a Vernier pH sensor. This sensor is a simple device that can be used to measure the acidity and basicity of liquids. This device is suitable to be used in a multitude of applications. For instance, pH monitoring in home aquariums, pH in mushroom growing, and analysis of water quality in water. The pH sensor probe is a sealed, gel-filled, type with a hard outer epoxy body that can function in a temperature range of 5 to 80° C. Additionally, these sensors can test in the typical pH range of 0-14 and each have an accuracy of +/- 0.2 pH units. It is easier to clean than a bulb, and the flat shape allows for smaller sample size. The research of this sensor is summarized in Table 28.

Temperature range	5 to 80°C (41 to 176°F)
Range	0–14 pH
Accuracy	+/- 0.2 pH units
Resolution	0.0025 pH units - 0.005 pH units
Response time	90% of final reading in 1 second

Table 28: Research Summary of the Vernier pH Sensor.

2.6.15 Native Mobile Application

The native mobile application could've been written in support of the web application. The native mobile application would need to provide a simplified version of the interface represented on the web component.

The application could be built using the Android SDK. This means that users with an Apple device will only be able to use the web portion of the user interface. This option was chosen because the testing phase will require the use of an Android phone since that is the make of the tester's mobile device.

Android SDK - This package was chosen to design the mobile application because it is the most evolved SDK for designing Android apps. Using this SDK will require the use of Eclipse Integrated Development Environment. The language that will be used to develop this application is Java.

The main purpose of a native mobile application is to enhance the user experience. This means that the most important part of the application is the user interface. Because of this we will be using React Native to develop the user interface.

React Native - React native is a JavaScript framework that can be used to write natively rendered mobile applications. React Native is targeted for developing on Android and iOS devices so there should not be any issues with compatibility.

React Native is based on React, the JavaScript library we will be using to develop the front end of the web application. Because these two technologies are so closely related, designing a similar user interface will not be a difficult task. It will also help in providing the user with a similar environment when switching between the two applications.

Because React Native renders its host platform's standard rendering APIs, it will not take a large performance hit like other cross platform application

development tools. Typically mobile applications for cross platform purposes are written using the standard combination of HTML, CSS, and JavaScript which typically render using webviews. Because of this method, the applications have a web feel to them and do not have the sleek mobile application feel. Developing using React Native will allow the user to have a similar experience to the web application without the mobile application feeling like a web interface.

Because React Native is just JavaScript, there is not much of a learning curve associated with the tool. This provides a lot of potential crossover features between the web application and the mobile application. This will also help in the development phase as React Native supports a lot of debugging tools and error reporting that can help us to find flaws within our code.

The point of the mobile application is not to replace the web application. The mobile application is meant to expedite the process of controlling settings on the nursery. Because the application is on the user's phone they will be allowed to easily enter the app, make a change, and close the app without having to go to a web browser and login.

2.6.16 CO2 Sensor

To maximize mushroom growth, it is critical to monitor the level of CO2 (Carbon Dioxide) in your growing chamber. CO2 is measured in ppm and is a major factor for obtaining good quantity, quality, and size of the produced mushrooms. An improper balance of CO2 can induce fruiting body deformities or abnormalities. Table 29 summarizes the research results of the typical CO2 Sensor we looked to implement in this project. As we decided not to implement a CO2 Sensor we summarized the typical properties below.

Operating Voltage	3.3V - 5V DC
Current Draw	Less than 500mA
Interface	Single Wire

Table 29: Summary of CO2 Sensor Properties

2.6.17 Plugs

The following sections cover plugs required to power and or interact with the various components of this project.

Controller Power and Programming Interface - This section covers research plug choices to power the PCB board including controller and other chips mounted on the board as well as plug requirements for programming the controller. The requirements for the plug(s) are as follows:

1. Easily accessible and easy to plug in/out.
2. Ability to plug in an of the shelf AC to DC power adapter.
3. Ability to connect to a computer for flashing.
4. Mountable on the PCB.

The requirements are derived as follows, we expect the main power plug and the programming plug to be connected and disconnected rather often, hence it needs to be easily accessible and easily to connect and disconnect. Since the board will be powered of DC voltage we will require an external AC to DC converter, being able to use an off the shelf wall-wart style power adapter would be ideal as they are plentiful, cheap and mostly reliable (further discussion about reliability and mechanisms used to regulate inconsistencies in the supply are discussed in section 2.6.10). Since we will need to program/flash the chip with our custom firmware we will require a connector to connect a computer to the chip, in this research we are evaluating the options of using a seperate power and programming connector or combining both functions into a single connector. The benefits of a single connector are reduced part count, and better space efficiency, however it comes at the cost of increased design complexity and depending on the type of connector chosen additional converter chips may be required on the PCB as well. Finally the plug or plugs chosen should be mountable on the PCB for a sturdy connection.

Table 30 depicts the results of the following plugs that were evaluated, all can be acquired at an average unit cost of about $1.

Component	Feature Considerations	Vendors
Barrel	• Supported by majority of AC wall adapters. • Cheap • Various mounting options for PCB. • Would require separate programming plug.	Mouser DigiKey Sparkfun LCSC
Molex	• Various mounting options for PCB. • Not very common plug for tradition AC wall adapter power supplies. • Would allow for programming if multiple lines are used. • Cheap	Sparkfun DigiKey
JST	• Various mounting options for PCB. • Common in battery powered devices. • Would allow for programming if multiple lines are used.	Sparkfun DigiKey
Pin Header	• Would allow for powering and programming. • Not common for powering. • Would allow for USB programming with external USB to serial converter. • Cheap	Male: Sparkfun Female: Sparkfun

Table 30: Plug Research Summary

Table 31 summarizes the USB plug research, all the USB plugs evaluated can also be acquired for an average unit cost of about $1.

Component	Feature Considerations	Vendors
USB A	• Would allow for powering and programming. • Various dc wall adapters with USB A female ports exist. • Would allow for powering from computer. • Would require a USB to Serial converter on PCB for programming. • Relatively large compared to size of PCB. • Four pins	Male: Sparkfun Female: Sparkfun
USB B	• Same considerations as USB A • Even bulkier than USB A	Sparkfun
USB Mini	• Same considerations as USB A • Much smaller in size	Sparkfun
USB Micro	• Same considerations as USB A • Much smaller in size. • Additional fifth pin. • Cheap	DigiKey

Table 31: USB Plug Research Summary

Chosen Plug - The chosen plug choice for powering the board and it's non peripheral components is a barrel jack style connector, the simplicity of the connector as well as the sturdy connection it creates when plugged in and it's compatibility with such a wide array of available ac to dc wall plug adapters makes it the ideal choice for this project and outperforms the other plug choices researched.

Since we are powering the board through a barrel jack style connector, a separate dedicated plug for programming the chip is required. This programming connector is wired to the TX and RX pins of the controller, and an additional push button to GPIO0 was added to the board to enable programming mode on the controller. We chose to use male pin header connectors for the TX and RX lines for simplicity. In this configuration we are able to utilize an external USB to serial converter that plugs into our computer via USB interface and then plugs to our TX and RX programming lines via female pin headers.

Power & Programming Plug Acquisition - The Barrel Jack Power Connector can be purchased from any of the vendors listed in the below table, a wide variety of vendors sell this style of connector so no shortage of units was expected. Since these types of connectors are available from a wide variety of manufactures we did not limit ourselves to a specific manufacturer and model but instead chose connectors from different manufacturers that meet our minimum requirements, and focused more on availability and price when querying the vendors.
Our minimum requirement for the power plug is that it is rated up to 24V DC and 2.5 Amps. Table 32 shows the part counts available at time of writing.

	Mouser	**DigiKey**	**Sparkfun**
5.5mm Barrel Jack adapter	43,314	120,277	In Stock (Exact part count not available)
Male Pin Header	46,692	217,144	In Stock (Exact part count not available)

Table 32: Barrel Jack and Pin Header Availability.

Sensor Plugs - This section covers plugs required to power and read measurements from the sensors. The requirements as they pertain to the plugs for each sensor are as follows:

1. Moderately easy to plug in and out.
2. Sturdy connection once plugged in.
3. Mountable on the PCB.
4. Multiple wire support.

Similarly to the power and programming plugs we want the female plug connectors to be mountable onto the PCB, and while we do want them to be easily plugged in and out the emphasis rather goes to a sturdy connection once they are plugged in since in the traditional application they should not be plugged in and out very often. Additionally the connectors need to support multiple wires, the typical sensor has at least 3 wires, one for ground, power supply and data.

2.7 Requirements Specification

This section covers the formal requirement specifications of the project, they are grouped by priority into three sections, first Core Feature Requirements containing the requirements that are most important, second Advanced Feature Requirements containing the features with a medium priority and finally Stretch Goal Feature Requirements containing features of low importance.

2.7.1 Core Feature Requirements

Table 33 lists core feature requirements, including the requirement identifier and statement, reason and any additional notes that assist in interpreting the requirement. This section is additionally split into two tables the first covering the first 7 core feature requirements and the second, table 34, covering the last 10 core feature requirements.

#	Requirement	Reason
CF1	The system will be able to read the temperature inside the grow area.	In order to upload the temperature to the server and in order to regulate temperature as described in SGF1 and SGF2.
CF2	The system will be able to log sensor readings and device actions such as lights ran for X minutes, humidified for Y minutes.	In order to further analyze this information at a later time.
CF3	The controller will be able to connect to a Wifi network.	In order to submit data to the web server.
CF4	The system will be able to read the relative humidity inside the nursery	In order to upload the value to the server and later regulate humidity.
CF5	The system will be able to engage/disengage a fan. The fan will provide an air filter.	In order to reduce contamination and to regulate air flow, humidity and temperature.
CF6	The system will include a controller unit, that is installed near or on the case.	In order to control the devices and sensors interacting with the grow box and to interact with the server.
CF7	The system will include a web server, that is hosted remotely.	In order to provide an interface for web or mobile applications to fetch data from and in order to provide an interface to receive data from the grow box.

Table 33: The top 7 Core Feature Requirements.

Core Feature Requirements 8 through 18 - Table 34 contains the continuation of the core feature requirements, starting with CF8 and ending with CF18.

#	Requirement	Reason
CF8	The system will include a database, that is hosted remotely.	In order to reliably store data received from the controller, user related and other data.
CF9	The system will include a web application.	In order to allow a user to access and modify the grow information.
CF10	The system will include a light.	In order to provide light when required.
CF11	The system will have means for automatically turning on and off the light described in CF10.	In order to allow the automation of the light cycle.
CF12	The system will have means for increasing humidity inside the grow area.	In order to achieve optimally high humidity inside the grow area.
CF13	The web server shall require authenticated access prior to accepting or releasing any data.	In order to prevent malicious or unapproved access.
CF14	The web server shall expose an application programming interface (API).	In order to allow the web/mobile applications and controller to engage with the server.
CF15	The web application shall be mobile responsive.	In order to allow it to be easily used on desktop and mobile devices.
CF16	The web application shall allow user to view current sensor readings.	In order to allow user to check condition of the grow environment remotely.
CF17	The web application shall allow user to modify grow environment configuration.	In order to allow for manual optimization of environment.
CF18	The web application shall allow user to manually enable/disable light, humidifier and fan.	In order to allow for manual optimization of environment.

Table 34: Core Feature Requirements number 8 through 18.

2.7.2 Advanced Feature Requirements

Table 35 lists the Advanced feature requirements, including the requirement identifier and statement, reason and any additional notes that assist in interpreting the requirement.

#	Requirement	Reason
AF1	The system shall be able to represent logged data in a report style format, including graphs and charts.	In order to easily represent and visualize the logged data.
AF2	The system will provide a set of preconfigured growing environment configurations that the user can apply via the web interface.	In order to make it easy for the user to grow a variety of different fungi or other organisms without having to modify the growing parameters directly.
AF3	The system will provide the ability to initiate and end a grow run.	In order collect, analyze and present data specific to a grow run.
AF4	The system will notify the user on their web or mobile application when conditions inside the nursery exceed a notification threshold.	In order to alert the user to check on the nursery.

Table 35: List of Advanced Feature Requirements

2.7.3 Stretch Goal Feature Requirements

Table 36 below lists the Stretch Goal feature requirements, including the requirement identifier and statement, reason and any additional notes that assist in interpreting the requirement.

#	Requirement	Reason
SGF1	The system will be able to cool the inside of the grow area.	In order to maintain ideal growing conditions.
SGF2	The system will be able to heat the inside of the grow area.	In order to maintain ideal growing conditions.
SGF3	The system will be able to measure the pH level of the spawn substrate.	In order to upload this information to the web server. Note that pH levels of the substrate may or may not have a major impact on growth as long as they are in a normal range. See M1.
SGF4	The system will include a mobile application.	In order to allow a user to access and modify the grow information.
SGF5	The system will measure CO2 levels inside the nursery and upload the readings to the web server.	In order to monitor and manage CO2 levels inside the grow chamber. Note that CO2 levels are an important factor for fruit body development and vary between species.
SGF6	The system will provide the ability to connect to the mobile application via Bluetooth and bidirectionally transfer data and configuration parameters between the two.	In order to allow for a non WiFi based method of controlling and interacting with the nursery.

Table 36: List of Stretch Goal Feature Requirements

2.8 House of Quality Illustration

Figure 14 visualizes our feature importance evaluation in a House of Quality (HOQ) layout.

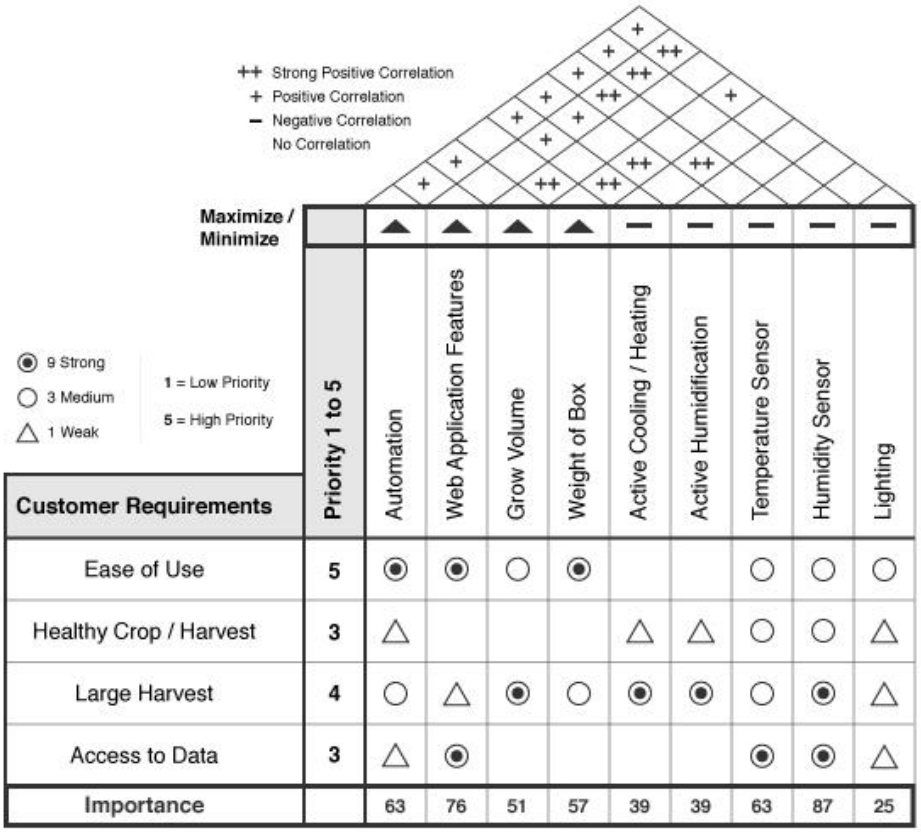

Figure 14: House of Quality Illustration

2.9 Realistic Design Constraints

Realistic Design Constraints focus on the constraints that originate from external sources. These constraints are predetermined by the environment in which the Nursery is being built and careful consideration must be taken in order to create a product that operates within these bounds.

2.9.1 Economic Constraints

One of the main goals of this project is to provide a reasonably priced method of growing mushrooms. This goal provides limits on the materials used to build the Nursery. In order to decrease the market price of the final product our team must actively strive to minimize the costs associated with production. With these goals in mind, a lot of research has been conducted regarding the cost of raw materials in order to produce a product that is not only affordable, but also functional.

The objective of this product is not to supply the user with a piece of the necessary tools to grow mushrooms on their own. Instead we aim to wrap up all tools in one all inclusive package so that the user does not have to plan to spend an ambiguous amount of money on third party tools. Our search for raw materials has been fruitful and being able to produce a complete product as described above is reasonable.

Another economic constraint that needs to be taken into consideration is operating costs. The Nursery does not supply its own power and water nor is it capable of automated regular maintenance. The user will have to plug the Nursery into an outlet in their chosen operating environment and periodically fill the water tank. The user may also want to clean the Nursery in between grows as to prevent contamination.

While operating costs are a burden the user alone must bear, our team has taken these costs into consideration while designing the product. The Nursery will not be power hungry and will be capable of operating at very low power levels. Tap water is a very cheap resource and the user will only need to fill the tank every so often. One advantage of the system being automated is that the Nursery will only use exactly what is necessary, nothing more and nothing less. This ensures that operating costs are consistent and kept at a minimal level.

Maintenance costs are another aspect of economic constraints that are relatively cheap. The user will be able to clean the Nursery with standard household cleaning products. The Nursery aims to provide the user with a hands-off approach as to reduce the risk of contamination and user error. The user will only be required to clean the Nursery as they see fit, however, recommended maintenance standards will be provided to the user.

2.9.2 Environmental Constraints

Because of the nature of the project, the Nursery is bounded by many environmental conditions. One of the biggest environmental constraints is temperature. The ideal environment for the Nursery is indoors where the temperature is controlled. Room temperature is ideal for the Nursery as the Nursery will be able to raise or lower temperature to levels that are ideal for mushrooms much more easily. For example, if a user were to set up a nursery outside in Alaska during the winter, the Nursery would be required to drastically increase the inside temperature and sustain that environment. Not only would this be costly, but the Nursery would be much more susceptible to component failure. This does, however, work vice versa as well. If a user sets up the Nursery in an environment that is already ideal for the species of mushrooms being grown, then the Nursery does not need to manipulate the inside conditions very much, and the user will experience much cheaper operating costs.

Outside contaminants also produce a potential environmental problem. If the Nursery is placed in an environment where contaminants are abundant then the mushrooms will be greatly impacted. The air filter is only able to accomplish so much, and many contaminants produce spores that are microscopic and can easily bypass the filtration system. Not only does the Nursery itself need to be clean, but the surrounding environment must be clean as well.

2.9.3 Social Constraints

Mycology and growing mushrooms has many applications. Growing mushrooms is a hobby with many enthusiasts. We want to provide enthusiasts and hobbyists with a system that is not only affordable but aesthetic as well. Word of mouth will be the most effective means of advertisement. We want to construct the Nursery so that users will want to show it off to their fellow hobbyists.

Mycology has many scientific applications as well. While aesthetics is a factor in this area as well, reliability is much more important. If unreliable components are the cause for failed experiments we can not expect scientists and engineers to recommend this product to their counterparts. Reputation and creating a positive impression on the scientific community is of the utmost importance when it comes to creating a product that sells.

2.9.4 Political and Ethical Constraints

While our team will take pride in seeing how consumers use this product in various ways, we recognize that there are some potentially illegal or immoral ways that this product can be used. We do not promote nor do we condone using this product for applications that are in direct violation of the laws that the user must operate within. While we do not endorse wrongfully using this system, it is the responsibility of the user to ensure that they are using this product within the boundaries of the law.

2.9.5 Health and Safety Constraints

Because the intended use of this system is to grow various fungi species, some potentially hazardous conditions could be created. There are many contaminants and mold species that require the same environments as mushrooms. If the user does not keep a close eye on the growth of their mushrooms and periodically check for contamination then the Nursery could potentially become a breeding ground for the contamination. We highly advise that the users learn to handle contamination properly and do not attempt to interact with the contamination closely. Many mushroom hobbyists use the method of smelling their mycelium to test for contaminants but this is a bad idea, as the contamination could enter the user's body and cause some serious health concerns.

The user must also develop a thorough understanding of the mushroom species that they are attempting to grow. Just because the user grew the mushroom in a clean and controlled environment does not mean that any species of mushroom is now safe to eat. We recommend that if the user is new to the hobby of mushroom growing, that they start with one of our recommended species.

2.9.6 Manufacturability and Sustainability Constraints

While we used some raw materials to construct this system, we did not manufacture anything that requires the use of industry level machinery. The machinery that is present in the UCF Engineering labs was used to manufacture any parts that may be required. While using industry level machinery to manufacture various parts would be useful, no one on the team had access to such technology.

Just like any physical system the Nursery will undergo normal wear and tear. We expect this to be minimum as there will not be a lot of stress placed on the components. Regardless, the components should be easy to remove and replace if a component were to fail beyond a point that it would not repairable.

3. Similar Projects Research

The following four projects were identified as being similar to ours and researched for similarities and differences.

3.1 Plant Automated Sustainable System

The Plant Autonomous Sustainable System was a project created by previous UCF Senior Design students. It is an autonomous system that facilitates plant growth by controlling different factors in the plant's environment changes and use little human interaction in the growing process. The primary benefit that the Plant Automated Sustainable System (PASS) offers is sustainability for the end-user. The device is analogous to a control system; it regulates the current ambient condition of the plants environment and takes action to maintain the current state of the plant when factors in the environment change. In addition, the device consists of an integrated feedback network to sense environmental conditions such as light, moisture, temperature, the alkalinity of soil and other factors that govern growth. This information is sent to a microprocessor that may activate a sub-system to compensate for the ambient change. Such autonomy reduces the need for human intervention while minimizing its own carbon footprint by utilizing natural resources, such as solar energy for power. The system contains a reservoir for and an irrigation system to water the plant. Because of its inherent autonomy, the system possesses a significant feature which similar devices often lack, which is its adaptable functionality – the user may simply press a few buttons to alter the growing conditions for different plants. This is accomplished through an onboard database that contains the growing specifications for different plants. The user may wirelessly receive status updates and alerts regarding the condition of the plant via a local area network.

3.2 Automated Plant Growth System

The automated plant growth system was also a project created by previous UCF Senior Design students. The purpose of this project was to automatically measure and regulate the growth characteristics of plants while requiring minimal to no user interaction. Some of the controllable features of this project include the regulation of nutrient concentration and pH level. The system also gives feedback of CO2 level, humidity, temperature and liquid level of our mixture. A valve and pump system is used for water regulation and draining, which includes a liquid level pressure sensor and a feeding pump. An automated lighting level adjustment is also integrated into the project. A wireless interface hosted by a web-server offers a web-based GUI for system settings and viewing of environmental conditions. Most of these controls for the project are based on the data that is taken from the sensors that were selected.

For the sensors in the project, the students used specific meters that detect the level of that given solution. Most of these sensors have analog outputs and must be adjusted to a proper interval for the A/D converters so that microcontrollers can do the calculations and output different information to the web with the wireless controller which is displayed on the web page online. The frame is a rigid wood structure because of the strength and low cost that it gives. A web interface allows the user to access the system easily from anywhere on the planet. This allows the user the ability to monitor almost all elements and regulate certain portions of the simulated environmental conditions of a specific plant in the system.

3.3 Smart Plant Pot

This project was also created by UCF students. It was designed for botanical-based researchers, this is a transportable monitoring environment for a plant's growth and life cycle. Various sensors track important metrics and records the data. To date, the few smart plant growth environments available tend to provide immediate singular data, not data trends. This project's enclosure will provide safety to the plant.

3.4 Autobott

This project was also created by UCF students. The AutoBott is a fully automated indoor hydroponics smart garden that is reliable and fruitful. It can grow almost any plant, including vegetables, fruits, flowers, and even trees, when they are in their beginning stages of growth. This system was designed with the motivation of targeting people who live in urban areas and do not have the luxury of having an outdoor garden, as well as, people who live in bad weather and climate conditions. These people will be able to use and control their AutoBott to grow whatever they choose. The AutoBott is made up of two subsystems, hydroponics and the enclosed environment. Both systems are fully automated in which they control the climate inside the enclosure and the water quality that supplies nutrients to the plants. AutoBott is not only efficient and useful but is also able to be simply plugged into a normal house outlet, which makes it convenient and easy for the user to place in their homes.

They used various sensors that detect light, air, and water, and will automatically adjust the environment to the appropriate levels to ensure an ideal environment for the plants inside. The system is enclosed in a cabinet sized container, allowing it to be moved to different locations. The system does monitor power consumption. Lights consist of low power LED grow lights that are timed based on the plant and plant cycle. The number of lights depends on the number of plants and the intensity of light desired. Air quality is tested using multiple sensors including: temperature, humidity, and CO_2. Adjustments to the environment are automated, viewable from a web/mobile app and adjusted if the

user wishes. With the lights in the system on, temperatures are expected to increase. To bring the temperatures back to desired levels, warm air will be cycled out and fans will turn on providing a breeze. Humidity levels will be adjusted according to the plants needs and a humidifier will provide more air moisture for the plants. As the temperatures and humidity levels adjust, CO_2 levels will also change. A CO_2 sensor will detect whether the environment needs adjusted CO_2 levels. A CO_2 tank will then pump small amounts of CO_2 until the desired level is reached.

4. Prototype

The following section details the methods we used to create prototypes for the system and subsystems so that our team developed optimal solutions to all the parts of the Nursery.

4.1 Web Application Prototype

In order to ensure the best possible outcome with the web application, prototypes of the application will be created. Prototyping is a method of creating preliminary models of the application so that we can see how far away we are from the final product with each prototype created.

Prototyping in software development is similar to prototyping in other fields, like mechanical engineering or manufacturing, but is more robust and allows for more flexible prototypes. In software, a prototype does not need to represent all aspects of the final product and can be used to represent certain specifications of the final product. This is extremely beneficial, as we will be able to test the software part by part before testing the software as a whole. We will also be able to produce stronger estimates for deadlines and milestones as more prototypes are developed. Prototypes in software also have the benefit of becoming the final product. This means that instead of discarding the prototype, we can simply modify it so that it meets all of the final products standards and requirements.

Prototyping process - The prototyping process, for the software section, is made up of a series of steps that should be followed closely.

Step 1 - Identify Basic Requirements - In this step we will determine the basic requirements which should give an intuitive description of what the software is attempting to accomplish. These requirements should include the input and output information.

Step 2 - Develop Initial Prototype - In this step we will develop the initial prototype which should give a view of what the user interface will look like.

Step 3 - Review - As a team we will review the prototype and, using the team's feedback, create a list of potential additions or changes.

Step 4 - Revise and enhance the prototype - Using the feedback acquired from step 3 we will make the necessary changes to the prototype. After completion, we will return to step 3 until there are no more changes.

Step 5 - Write the final requirements - Once all the preliminary requirements have been accomplished we are able to write the final requirements for that system or subsystem.

Prototyping Dimensions - Like most software systems our project can be broken down into a front-end and a back-end. In terms of prototyping these are known as the horizontal prototype (front-end) and the vertical prototype (back-end).

Horizontal Prototype - The user interface prototype is commonly known as the horizontal prototype. This prototype should give a broad view of the system that the prototype is meant to represent. This prototype focuses less on low-level system functionality and more on user interaction. This kind of prototype should be developed before any vertical prototypes because it details how the user will interact with the system.

The horizontal prototype will be useful for the project because it will help develop estimates regarding cost, effort, and time. The prototype will also help to confirm or deny any user interface requirements or scope specifications.

Vertical Prototype - Vertical prototypes are used to give an overhead view of a single subsystem or function. This will help our team obtain detailed requirements for the function that we are prototyping for.

The vertical prototype will be useful for multiple reasons. It will help us design a database that does not obtain unnecessary information. It will provide us with information regarding system interface needs. We will be able to narrow down requirements that are initially too vague or too broad.

Variations of Prototypes - There are many variants of prototypes and before prototypes are created our team will take careful consideration in choosing the correct type for the project.

Throwaway Prototype - Throwaway prototype, also known as close-ended prototyping or rapid prototyping, is the development of a prototype that will eventually be discarded. A simple working model of the system being prototyped is created to demonstrate what the requirements could look like when implemented into the final system.

This method of prototyping focuses on speed. Throwaway prototyping involves creating a working model of the various parts of the system during the early stages of development. This method is typically informal and focus is not put on code readability or coding conventions. Instead, the focus is placed on developing a working model. This is useful as we are still able to see what works and why it works, while still being able to develop and re-examine requirements. After the prototype is completed and all necessary information and requirements are extracted, the prototype is to be discarded.

The tactic of developing a prototype quickly has a lot of benefits. Creating prototypes quickly means that, as a team, we will be able to develop requirements and get feedback quickly as well. We will then be able to refine requirements in the early stages of development so that we have very descriptive and meaningful requirements in the later stages. Another benefit is that throwaway prototyping is extremely cost effective. This is because prototypes are developed in the early stages and therefore not much will need to be redone in the later stages of development. Small changes added to large projects that are far in their development phase can prove to be costly and time consuming.

Testing is another aspect of software development that needs to be taken into consideration when choosing an appropriate prototyping method. Throwaway prototyping will allow us to create interfaces that can be tested early on. Testing early will allow us to identify problems early, therefore, greatly reducing the threat of the system surprising us with bugs in the late stages of development.

This prototyping method can benefit our team because we are working on a limited budget and timeline. Being able to develop prototypes and create requirements in the early stages of development will allow us more time and a more flexible budget in other areas of the project that require more attention.

Evolutionary Prototype - Another popular prototyping method that is very different from throwaway prototyping is evolutionary prototyping. Evolutionary prototyping is also known as breadboard prototyping. The idea behind evolutionary prototyping is to initially create a robust prototype in a manner that follows syntax rules and coding conventions and constantly refine it until it meets all the requirements of the final product. This will save time and money as the system will be completely built once all the requirements and specifications have been created. Also, there will be no need to rebuild the system once the prototype has reached the end of its' life.

Even though we intend to use the prototype as the final product, we acknowledge, during the early stages of development, that we do not yet understand all the requirements and specifications of the final product. The prototype and requirements will be developed hand in hand.

Because the prototype evolves with the project, it will handle more than just one subsystem at a time. The prototype may start with one subsystem, but will eventually consist of multiple subsystems. Being composed of many moving parts the prototype will allow us to see how all the subsystems interact and will show us any problems in the communications between parts.

While there are many benefits to using this variant of prototype, this may make testing a little complicated. While we are unit testing, we will have to do our best to isolate the subsystem that we are testing. Other subsystems could get in the way and make this a challenge.

Evolutionary prototypes, however, do have an advantage over throwaway prototypes in that they will become the final system. This means that during development they are treated as functional systems and will be used to test other parts of the project, giving us a head start on the integration of the final system and the hardware.

Evolutionary prototypes will still be built piece by piece, and construction of one subsystem will not begin until the previous subsystem has been built and tested. Creating a system of many unfinished subsystem is messy and can cause a lot of confusion, ultimately resulting in failing software. The goal is to develop each subsystem and make sure it works perfectly before considering it to be a functional part of the system as a whole.

Incremental prototype - Incremental prototypes are a combination throwaway prototypes and evolutionary prototypes. Like throwaway prototypes, subsystem prototypes are developed separately. Like evolutionary prototypes, these subsystems will eventually become the final system. This is done by merging the subsystems into one system once each subsystem has been built and successfully tested.

While this variant of prototyping is very efficient on time, a gamble is taken because testing of how the parts interact is done at the very end of the development process. If for some reason the subsystems do not interact properly, quick solutions will be required to fix the issue. This could potentially reduce the quality of the software being developed.

Extreme Prototyping - Extreme prototyping is a prototype variant that is typically used for developing web applications. Extreme prototyping breaks down the development into three phases.

Phase 1 - Static Prototype Phase - During this phase of development only the static webpage will be developed. This phase includes development using JavaScript, CSS, and HTML on the front-end of the application.

Phase 2 - Dynamic (Extreme) Prototype Phase - The screens will be fully functional and operations will use a simulated services layer to represent how the application will work once the final service layer is implemented.

Phase 3 - Service Implementation Phase - This phase includes connecting with an API and services that can communicate with the database to retrieve real functional data.

Combination of variants - This application will use a mix of the previously described prototyping methods. Because the software in this application will be used to test the controller and sensors, prototypes will start on both the front-end and back-end and meet in the middle at the last stages of development. The front-end will be designed using throwaway prototyping as this allows us to develop the optimal user experience. The back-end system will use evolutionary prototyping to ensure that the code does not break with all the different calls to the controller for various sensor data.

4.2 Hardware Prototyping

This section covers the three major physical processes undertaken when designing the hardware and it's various prototypes. These major processes consist of designing and testing circuit sub systems via a breadboard, testing sub systems including firmware via development boards, and finally designing and assembling the printed circuit board. Throughout we were often required to use similar components in prototyping if exact components were not available due to factors such as time, cost and general availability.

Subsystem Prototyping via Breadboards - While developing the various subsystems of the project circuit we often verified our designs via circuit simulation software and by assembling the subsystem in question on a breadboard. Since our PCB design makes use of various SMD components, which are typically too small and too difficult to work with when breadboarding we would attempt to locate either the same component in a through hole package or if unable to, locate a similar component in a through hole package. When looking for a similar but not identical component we tried to ensure that the parameters affecting our test are identical or nearly identical to the ones of our desired components. Some of the subsystems we evaluated with this method were the programming circuit, the power regulation circuit, the sensor connection circuits and the device control circuits.

Subsystem Prototyping via Development Boards - Some of the crucial components to our project like the controller unit are simply only available in package sizes that are too small or too difficult to assemble via a breadboard or to solder by hand, and due to their proprietary and unique operations we did not

want to locate a similar component as the risk of incompatibility is too high. In that case we looked for development and breakout boards that as their name implies break out all the pins of the SMD component in question into a breadboardable or solderable way. For example, the majority of the development boards we encountered for the ESP32 contained either male or female pin headers for all the GPIO ports in question. In the case of the ESP32 this allowed us to test some of the subsystems that operate with it like the programming circuit and powering circuitry as well as the firmware operating the controller.

PCB Fabrication and Assembly - To develop the printed circuit board we used an electronic design software called EasyEDA it is further described in the PCB section of this document. Using the schematic and board layout designs from EasyEDA one has to now find a fabrication house, we researched the following fabrication houses and summarized some of their noteworthy differentiators. It is also important to note that some fabricators also offer assembly service of the board and its components. The summary of the vendors researched for PCB fabrication is in table 37.

Fabricator	Build Time Estimation	Offers Assembly?	PCB Cost Estimation
JLCPCB.com	1-2 Days	Not currently, but assembly coming soon.	$20-$30 / unit
PCBWay.com	24 Hours	Yes, at about $30 assembly cost.	$5 - $15 / unit
Oshpark.com	5 Business Days to 3-4 Weeks	No	$5 / square inch for 2 layer boards.
ProtoExpress.com	5 Days (including Assembly)	Yes	Not available.
Wellpcb.com	20 Days (including Assembly)	Yes	Not available.
Allpcb.com	7 Days (including assembly)	Yes	With assembly about $38 per unit. Minimum order of 5 units.
4pcb.com	1 Day (including assembly)	Yes	$33/unit with student discount.

Table 37: Summary of PCB Fabrication and Assembly Research

5. Quality Control

The following sections describe our research and process of Quality Control processes as well as our Software and Hardware testing strategy.

5.1 Web Application Software Testing

Testing the software is an important process of determining quality and getting a feel for how much more time and resources are needed in the development phase. Testing is also necessary for finding errors or other defects that prevent the software from producing the targeted results. The ultimate goal of testing is to determine whether or not the software is ready to be applied to the system.

The goal of testing is not just to make sure that the software works. Through testing, we will be looking for bugs and attempting to break the code in order to ensure that the code is ready. There are very specific results that we will be looking for when testing the software. These results include:

- Meets the requirements that the software was built upon
- Meets any requirements that came after the initial design
- Accomplishes any required functions within an acceptable time frame
- Consistently Successful
- Operates sufficiently within the intended environment

Because we used an Agile approach, the code was tested very frequently, through release iterations. This helped reduce the risk of any problems with the code in future integrations. During each sprint, the software was tested to make sure it is suitable to enter the system. Larger subsystems will be tested once enough pieces have been integrated. The entire system will be tested once all of the necessary pieces have been integrated.

While the primary purpose of software testing is to find errors in code that need to be corrected there are other purposes for testing the software. Testing also extends to finding any conditions under which the application does not support. The code will need to be executed under various environments and conditions to ensure that it supports anything that a user attempt. For example, the system may be willing to accept a certain set of inputs, but what happens if a user gives an input that is not in that set. The system will need to recognize that the input is invalid and react accordingly. A collection of combinatorial tests will need to be developed that give the system a random series of acceptable and unacceptable inputs.

Testing Approaches - Different approaches for testing the application will be detailed below. A multitude of approaches will be used, rather than one universal approach, as this will test the code from multiple angles and provide more reliable results.

Static Testing - Static testing refers to reviewing the raw code to ensure that there are no obvious errors. When writing code it is always useful to go back and review the code you have written to catch any mistakes that may not be apparent when running the code. There can be many reasons that a program does not operate as intended and, sometimes, the programmer may look for logic errors when the only problem is a missing semicolon.

Dynamic Testing - Dynamic testing covers a broad range of testing and takes place when running the program. Dynamic testing can be used to test specific methods or functions. This form of testing also uses passive testing, where we will run the code and not interact with the application to ensure that the base of the program runs without error.

Box Testing - Testing methods are typically done using the box testing method. This method is usually divided into white-box testing and black-box testing. These approaches provide different perspectives when testing code and designing test cases. There is also a gray-box testing method which is a combination of the two box testing methods.

White-box Testing - White-box testing, also known as clear box testing or glass box testing, is used to verify the internal structure of a program. Test cases are designed using the source code and any method used to design the source code. This method allows us to see how the code should operate under ideal conditions. With this testing method we give the code a certain input and know exactly what output to expect. Just like testing nodes in an electrical circuit we can view the outputs through the path of the software to determine exactly where the code breaks. White-box testing is typically done at the unit-testing level, which will be discussed in the levels of testing section below. This method is designed to uncover problems in the code itself and is not used to find missing requirements.

White-box testing uses different techniques to test the code. The code can be tested against different APIs to ensure that the service layers are producing expected results. The code can be tested against the entire system to ensure that parts do not interfere with each other even if they are working fine independently. We could also test the testing strategies themselves by introducing faults to test the error handling code paths.

Black-box Testing - Black box testing is also known as functional testing. With this method the testers of the software only have knowledge of what the software

is supposed to do. The testers do not see the source code in this scenario. The benefit of using this method of testing is that it provides the tester with an objective point of view. The tester can mark down what works and what does not work without having a biased point of view based on what they think is feasible given knowledge of the source code. This will allow the tester to view the website like any other website when experimenting with the different features of the website. This will also be beneficial to our team, as any member can test the application, not just the programmers.

One of the most common methods of black-box testing is specification-based testing where the tester tests the software against applicable requirements. This will allow the tester to use thorough test cases which can be marked with a pass or fail upon experimentation. The test cases used for this method will be designed with the specifications and requirements as the primary focus. These test cases are designed to find out whether the application does what it is supposed to do.

Black-box testing is not limited to testing the user interface. Component interface testing is a variation of black-box testing where data values are checked at different units within the application. There are many layers and programming languages that the data will be using and making sure that the data stays consistent is high priority. If, for example, the javascript parses the read sensor values incorrectly, then the user will see incorrect sensor values and assume that something is wrong with their nursery. This is why we want to make sure that the communication between the layers of the application is fluid and consistent. We will be able to read these values using various consoles and log files, such as, the developer tools on browsers.

Visual testing is another form of black-box testing that is very useful. The goal of this method is to be able to reproduce software failures so that testers are able to show the engineers exactly what the problem is. This will be very useful when testing the user interface. Visual testing greatly improves communication and reduces the risk of misinterpretation since the problem can be presented to the engineer, instead of being described.

Gray-box testing - Gray-box testing is a combination of white-box and black-box testing. The tester will still have inside knowledge of the code structure and algorithms used, but will conduct tests at the user level. Sometimes without being able to see the source code, fixes can be very ambiguous and a solution isn’t obvious. Because gray-box testing allows the tester to view the code, the tester can constantly make changes and rerun the code to get an understanding of what is wrong. The tester will also make better software choices when writing fixes to the code, since they will know what tools and methods were used to write the source code.

Levels of Testing - Testing occurs at different levels of the software development life cycle and should occur frequently to ensure that code works with any new technology introduced to the system. There are typically three levels of testing software. These are unit testing, integration testing, and system testing. These levels of testing occur at strategic points in the software development life cycle as to get optimal results from the system and reduce the risk for any potential errors or bugs.

Unit testing - Unit testing is a form of testing where specific sections of code are tested independently of the system as a whole. These types of tests help the developers while they work on the code, to ensure they are headed in the right direction as they develop a unit of code. Unit testing does not only occur at the completion of the development of the unit but rather throughout the lifecycle of that unit. Just like the system as a whole, the unit will be tested incrementally before the entire subsystem is tested. Unit testing does not verify the functionality of software but rather ensures that the unit being tested will make a suitable building block for the final system.

Unit testing aims to increase the quality of the software that will eventually be a part of the system. Unit testing is also helpful because the developer will begin to notice patterns in software faults between the different units. Unit testing ensures that the units being developed provide a strong base for the software system to be built on.

Integration testing - Integration testing aims to verify the interfaces between the different software units. This form of testing attempts to ensure that the different pieces fit together and that code from one unit does not negatively impact the performance of another unit. Integration testing also aims to test the code written to interface the components. This code could be considered a unit itself but is still tested within this level. Larger groups of code are created as more and more test cases are passed until the software works as a whole.

System testing - System testing is the final method of testing before we will consider the system to be acceptable. Final test cases will be written using methods described above to ensure that the code works fluidly and consistently. An example of a system test for this application would involve a user creating an account, logging in, and creating a Grow profile. The user will then attempt to operate the Nursery using the user interface and should receive feedback in the form of statistics as described in the Features section. If the user is able to accomplish all these tasks without error then the test case will be passed.

5.2 Hardware Testing

This section covers our hardware testing process and the environments that are utilized for testing. Specifically this section is broken into a discussion of unit testing of the individual subsystems, testing the fabricated and assembled PCB and finally a description of the environments that will be or have been utilized for testing.

Unit testing of subsystems - To ensure correct operation and behavior of our design as well as entire project and to minimize the amount of time needed to troubleshoot and isolate a specific problem, we planned to initially test the individual subsystems of our hardware. This method allowed us to divide the amount of complexity into individual units and should benefit from quicker troubleshooting and remission of potential problems. The individual subsystems were either assembled onto a breadboard if the components required weren't able to be found in breadboard friendly formats or a similar breadboard friendly component may be acquired in its place, additionally a combination of the two previous methods may be used for subsystems that are made up of a variety of components. If we were unable to find any breadboard friendly components that are identical or very similar to the component in question we either designed a breakout or acquire a breakout board for that component. If neither of the previous options are a possibility we attempted to at least model the subsystem in question with a circuit simulation software such as MultiSim Live.

Testing of the assembled PCB - was divided into two stages, firstly the peripherals were assembled one by one and attached to the PCB and again tested for operation individually, as it may be difficult to fully check if a subsystem meets it’s feature requirements due to a dependency of related subsystems, we determined success in an ad-hoc fashion until the tester is satisfied that this subsystem is functioning properly. And secondly the entire project was assembled and mounted as desired to test for complete function of the PCB and the connected peripherals. Success was determined by checking that all the feature requirements are able to be executed as described by the requirement.

Testing Environments - We used the following environments to conduct our tests in. Initially our individual home labs provided enough equipment for testing, however once additionally specialized equipment or materials were required we utilized the labs provided by UCF. Specifically we envisioned more frequent visits to the UCF Labs once more in depth troubleshooting was required that may only be carried out with specialized equipment. Once functionality is established we operated the Nursery in a variety of typical households as this would be the target user environment.

6. Related Standards

This section identifies and briefly discusses standards, protocols and best practices that are related to the project and most importantly how they relate to the project. We refered to all three types as Standards during this section, however the reader should understand that some of them may not be as rigid as the term Standard implies. While this is not a comprehensive list, as with most technology systems there are standards, protocols and best practices on almost every level, it covers the documents that are most significant and impactful to the project.

6.1 Software Standards

The software standards identified and described here relate to the software stack utilized by the project, where Javascript is the language utilized by the web application and server of this project and C++ is the language utilized by the controller unit of the project. JSON and Rest API are standards that both the web server, web application and controller unit interact with and are affected by.

Rest API - This is an architectural design style describing the communication between web servers and clients, it was first published in 2000 in a dissertation by Roy Fielding. The term REST is an acronym for Representational State Transfer and the standard focuses on interoperability between web services by requiring stateless and standardized operations. More information regarding the specifics of the standard can be found in Roy Fielding's dissertation [S9] and the related wikipedia entry [S5]. This standard relates to our project as it guides the design of our web server's application programming interface with which the web application and the controller unit will frequently interact.

JSON - This data interchange format provides formatting rules for a text based and relatively lightweight mechanism to create portable and structured data. It is frequently used due to its language independence even though it originally emerged from the ECMAScript (also known as Javascript) programming language standard. The JSON acronym stands for JavaScript Object Notation and further details can be found in the Internet Engineering Task Force's RFC #8259 [S4]. Our project requires that the web server interchanges data with the web application and controller unit, hence the inclusion of this standard.

C++ - This programming language definition is well known as an extension to the C programming language, it has many general purpose applications due to it comprising of object oriented, procedural and functional aspects. The language is formally standardized by the International Organization for Standardization (ISO) with the current standard being C++17. More information can be found in the

official ISO C++17 standard [S10] as well as the wikipedia entry for the language [S11]. Our controller unit will be programmed using the Arduino Environment which primarily utilizes C++, it is important to note however that the Arduino IDe has a pre-compile stage that provides additional syntactic features.

Javascript - This programming language is mostly known as being the scripting language used by web sites however in recent times it is also very frequently found in non browser environments. The language is formally standardized by the ECMA International Standards Organization with the current standard being ECMAScript 2018. It proved itself as highly productive multi-paradigm dynamic language that supports functional, imperative and object oriented architectural styles. More information can be found in the ECMAScript 2018 standard [S12] as well as the Wikipedia entry for Javascript [S13]. Our project utilizes Javascript as the primary development language on the web application as well as web server, additionally the native mobile application itself may be developed using a hybrid strategy which also utilizes javascript.

6.2 Hardware Standards

This section identifies and describes standards relating to the hardware and hardware interactions required by this project. The interactions of the hardware are both wired, i.e. physical connection to another device, and wireless i.e. the controller unit connecting to the internet via a wireless router.

Asynchronous Serial Communication - This communication standard allows for communication between two devices without requiring a common clock signal between them, instead it achieves synchronization by including start and stop signals before and after transmission of data respectively. This also allows the protocol to be configurable to a variety of transmissions speeds. More information can be found in the Wikipedia entry for Asynchronous Serial Communication [S14] and the Wikipedia entry for UART [S15]. Our project's control unit uses asynchronous serial communication through a universal asynchronous receiver-transmitter module also called UART, to download it's operating firmware from a computer.

Single Wire Protocols - Our project employs multiple sensors, from a communication perspective these sensors delivery data in a very simple format. They either provide an analog signal that is linearly proportional to the sensed value, as in the case of the TMP36 temperature sensor the analog value can easily be converted to temperature using a scale factor of 10 mV/C, or the sensors provide the data over a 1-wire protocol. Specifically the 1-wire bus protocol utilized by our Humidity Sensor RHT03 is a proprietary one by the manufacturer but in general these protocols send a certain number of data bits followed by a checksum that can be utilized to confirm the integrity of the data. More general information for 1-wire bus protocols can be found at the Wikipedia

entry [S16], while the specific 1-wire bus protocol for our Humidity sensor can be found in it's datasheet [S17].

IEEE 802.11 - This extremely popular wireless computer networking standard is utilized by the majority of all common at home wireless networks, it defines both media access control and physical layer protocols and supports 2.4 GHz and 5GHz bands. It has multiple amendments that append or approve it's uses these are typically titled by adding a letter behind the 802.11 number. More information can be found at the formal standard definition maintained by IEEE [S1] as well as the Wikipedia Article about IEEE 802.11 [S18]. This standard relates to our project as our controller unit provides a built in WiFi chip that utilizes the standard to connect to a nearby access point.

6.3 Regulatory Standards

This section describes the current regulatory compliance requirements by the FCC, mainly the requirements are standardized in Title 47 of the Code of Federal Regulations (CFR), this code regulates the radio spectrum and since devices with high speed clocks may produce unintentional radio interference the FCC regulates those devices.

FCC CFR 47 Part 15 Testing Requirements - The FCC requires all manufacturers to submit their device for testing, this testing is often described as cumbersome and expensive. The details can further be read up on in Part 15 of Title 47 [S19]. On important item to point out is that in subsection 23 of part 15 a Home built device exemption [S20] is described, specifically it authorizes an individual to be exempt from FCC testing if the device is built for personal use and in quantities of five or less. Our project qualifies for this exemption. Additionally there are additional exemptions pointed out in subsection 103 of title part 15 [S21] that lists a variety of exemptions, for example it includes an exemption for devices that are exclusively used as industrial or commercial equipment as well as digital devices that consume less than 6nW of power. The full list can be found in subsection 103 [S21].

7. Administrative Items

The following section covers administrative concerns such as the team members responsibilities, budget and financing concerns, the bills of materials for the major sections of the project as well as milestone table.

7.1 Team Member Responsibilities

Table 38 gives an overview of each team members responsibilities as it relates to the features of the project as well as the creation of the pertaining documentation.

David	**JP**	**Mardochee**
Cover Page Table of Contents Project Illustration WiFi Connectivity Bluetooth Connectivity Components Diagram Power Control and Regulation Project Schematic & PCB Design Plugs Controller Unit WiFi Module Requirement Specifications House of Quality Illustration Hardware Prototyping Hardware Testing Related Standards Appendix Formatting	Executive Summary Project Description Motivation Suitable Operating Conditions Web Application Features Growing Profiles Grow Data Reports & Graphs Automated Alerts Native Mobile Application Housing Web application Software Framework Web Server Database Native Mobile Application Realistic Design Constraints Web Application Software Prototyping Web Application Software Testing	Humidity Control Lighting Airflow Control Watering System Temperature Control pH Level Control CO2 Level Control Humidity Sensor/Humidifier Cooling/Heating Unit Fan/Air Filter pH Level Sensor CO2 Sensor Similar Projects Research Administrative Items Project Summary

Table 38: Team Member Responsibilities

7.2 Budget & Financing

The following section lists items purchased in table 39 and Bill of Materials for the Project board in table 40.

SparkFun ESP8266 Thing	$17.95
SparkFun Thing Plus - ESP32	$20.95
Break Away Headers	$1.50
ESP32 DEVKIT C Version 4	$14.95
PCB Components	$18.41
Fans	$15
Ultrasonic Atomizer	$6.99
Hosting	$22.00

Peltier Devices	$19.95
CO2 Sensor	$49.00
pH Sensor	$19.95
Humidity & Temp Sensor	$9.95
LED Lights	$25.98
Housing Materials & Tools	$160.50
Printing & Assembly of PCBs	$123.40
Total:	**$526.48**

Table 39: List of Items Purchased

PCB BOM - The bill of materials shown in table 40, includes the components that are soldered onto the PCB, the total of the components comes to $7.41

Name	Symbol	#	Mfg Part	Mfg	Sup.	Sup. Part	Unit	Total
ESP-WROOM-32	U1	1	ESP-WROOM-32	ESPRESSIF	LCSC	C95209	$3.19	$3.19
LD1117S33CTR	U2	1	LD1117S33CTR	STMicroelectronics	LCSC	C35879	$0.14	$0.14
100nF	C1	1	CL10B104KB8NNNC	SAMSUNG	LCSC	C1591	$0.01	$0.01
10uF	C2	1	CL21A106KPFNNNE	SAMSUNG	LCSC	C17024	$0.01	$0.01
DC-005-20A	DC1	1	DC-005-20A	HRO	LCSC	C136744	$0.09	$0.09
10K	R1	1	RC0603JR-0710KL	YAGEO	LCSC	C99198	$0.00	$0.00
K2-1102SP-C4SC-04	SW1,SW2	2	K2-1102SP-C4SC-04	HRO	LCSC	C127509	$0.04	$0.08
54102-T08-00	U3	1	54102-T08-00	Amphenol ICC	LCSC	C213181	$0.02	$0.02
Header-Male-2.54_1x3	P1,P2,P3	3	Header2.54mm 1*3P	BOOMELE	LCSC	C49257	$0.01	$0.03
DB301V-5.0-2P	U4,U5,U6	3	DB301V-5.0-2P	DIBO	LCSC	C395882	$0.06	$0.18
FQP30N06L	Q1,Q2,Q3	3	FQP30N06L	ON Semiconductor	Mouser	512-FQP30N06L	$1.22	$3.66

Table 40: Bill of Materials for PCB

7.3 Milestones Table

The following two tables table 41 and table 42 illustrate the timeline for Senior Design 1 and Senior Design 2 milestones.

Objective	Duration (Days)	Start	Finish
Summer Senior Design I	80	5/14/2019	8/2/2019
Project Idea for Senior Design	1	5/17/2019	5/17/2019
Meeting to determine Project Idea	1	5/21/2019	5/21/2019
Senior Design Bootcamp	1	5/24/2019	5/24/2019
Design Research Begin	21	6/3/2019	6/23/2019
Split up tasks and begin writing documentation	1	6/24/2019	6/24/2019
Divide and Conquer Version 1.0 Documentation	7	5/20/2019	5/27/2019
Divide and Conquer Version 2.0 Documentation	7	6/1/2019	6/7/2019
Table of Contents Assignment	3	6/11/2019	6/14/2019
10 Pages per person	5	7/1/2019	7/7/2019
50% Draft of Senior Design I (Due)	0	7/7/2019	7/7/2019
10 Additional Pages per person	7	7/14/2019	7/21/2019
75% Draft of Senior Design I (Due)	0	7/21/2019	7/21/2019
10 Additional Pages per person (Total 30 pages per person)	14	7/22/2019	8/2/2019
Project Due	0	8/2/2019	8/2/2019

Table 41: Senior Design 1 Milestones

Objective	Duration (Days)	Start	Finish
Senior Design II	106	8/26/2019	12/10/2019
Determine the Part to Order	7	8/26/2019	9/2/2019
Test Parts Individually	5	9/2/2019	9/6/2019
Begin Project Construction	10	9/7/2019	9/17/2019
Construction Complete	1	9/18/2019	9/18/2019
Insert Mushroom into the Nursery/Testing	2	9/19/2019	9/20/2019
Presentation	1	11/22/2019	11/22/2019

Table 42: Senior Design 2 Milestones

8.0 Project Summary and Conclusions

The purpose or goal of this project was to build a nursery system that can be used to automatically monitor, control, and grow mushrooms in an ideal environment. Additionally one of the biggest problems farmers face when attempting to cultivate mushrooms is external environment contamination. Our system is designed to help fight and reduce mushroom contamination.

The system has been carefully designed not to only grow mushrooms for consumption, but can also be used for scientific and medical purposes as there are certain species of mushrooms that contain important properties used in various scientific and medical purposes. In addition, this project has been designed to require very little physical interaction with the end-user in order to prevent or eliminate any potential human errors in the process. Through the use of the controller system, different components such as, sensors, lights, fan, and humidifier will able to interact with one another to provide feedback and help keep an ideal environment for the mushrooms. Also through the web based application, the user will be able to login into the application to view the current state of the chamber as well as to change and adjust settings accordingly.

In conclusion, The Senior Design I and II project has served as a vital and unforgettable experience for each and every one of us in our educational journey. It has provided us with the opportunity to work as a team for almost an entire year. We have learned a lot about new technology and how we can

incorporate these new technology into other projects and further improve the Mushroom Nursery. Most importantly, it has provided us with the skill set to adhere and follow the University of Central Florida College of Electrical Engineering and Computer Science guidelines and the Accreditation Board for Engineering and Technology (ABET) requirements for accredited engineering programs. In addition, this project has provided us with experience in research, proper documentation, and in the overall field of Computer & Electrical Engineering.

9.0 Appendix

9.1 Additional Figures

This section contains figures that were deemed important to be included as references for the reader but were not included in the main body of the document for brevity.

Figure 6 shows the Pin layout of the ESP32 WROOM modules.

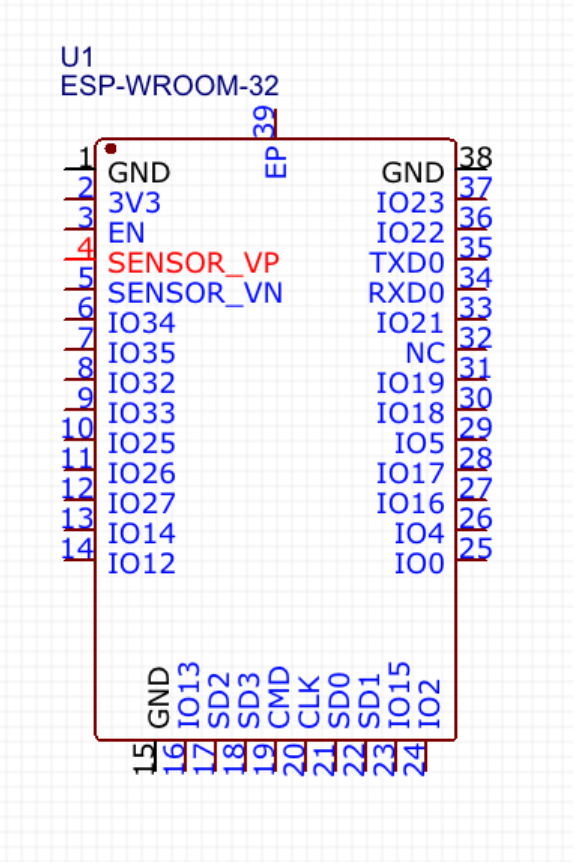

Figure 6: ESP32 Pin Layout

9.2 List of Tables

The following lists all tables used in the main body of the paper.

9.3 List of Figures

The following lists all figures used in the main body of this paper.

9.4 Reference Materials

The following tables lists noteworthy reference materials that were either directly cited in the paper or used as important educational resources.

#	Reference Name	Link
M1	Do PH Levels matter?	https://mycotopia.net/topic/89009-do-ph-levels-matter/
M2	IEEE 802.11 Wiki	https://en.wikipedia.org/wiki/IEEE_802.11
M3	WiFi Wiki	https://en.wikipedia.org/wiki/Wi-Fi
M4	Environment for Mushroom Growth	https://homeguides.sfgate.com/environment-mushroom-growth-28551.html
M5	Relative Humidity Wiki	https://en.wikipedia.org/wiki/Relative_humidity
M6	Application Programming Interface (API) Wiki	https://en.wikipedia.org/wiki/Application_programming_interface
M7	Responsive Web Design	https://en.wikipedia.org/wiki/Responsive_web_design
M8	UV light increases Vitamin D production in mushrooms	http://sciencenordic.com/uv-light-turns-mushrooms-vitamin-d-bombs
M9	Arduino MKR 1010	https://store.arduino.cc/usa/mkr-wifi-10
M10	Arduino MKR 1000	https://store.arduino.cc/usa/arduino-mkr1000
M11	How to choose a microcontroller	https://learn.adafruit.com/how-to-choose-a-microcontroller/wifi-boards
M12	Arduino Uno Rev3 SMD	https://store.arduino.cc/usa/arduino-uno-smd-rev3
M13	ESP32 vs ESP8266 – Pros and Cons	https://makeradvisor.com/esp32-vs-esp8266/
M14	Sparkfun ESP8266 Thing Breakout Board	https://www.sparkfun.com/products/13804
M15	Sparkfun ESP32 Thing Breakout Board	https://www.sparkfun.com/products/14689

#	Reference Name	Link
M16	ESP8266 Wiki	https://en.wikipedia.org/wiki/ESP8266
M17	ESP8266 Model comparison	http://esp8266.net/
M18	ESP8266 WROOM-32D Datasheet	https://www.mouser.com/datasheet/2/891/esp32-wroom-32d_esp32-wroom-32u_datasheet_en-1365844.pdf
M19	QWIIC	https://www.sparkfun.com/qwiic
M20	Description of System on Chip (SoC)	https://en.wikipedia.org/wiki/System_on_a_chip
M21	Arduino ESP32 Core Framework	https://github.com/espressif/arduino-esp32
M22	Graphical Datasheet of ESP32 Breakout Board	https://cdn.sparkfun.com/assets/learn_tutorials/8/5/2/ESP32ThingPlus_GraphicalDatasheet.pdf
M23	ESP32 Generic Datasheet	https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
M24	ESP32 Bootmode Selection	https://github.com/espressif/esptool/wiki/ESP32-Boot-Mode-Selection
M25	ESP32 Core Framework License	https://github.com/espressif/arduino-esp32/blob/master/LICENSE.md
M26	Hayes / AT Command Set Wiki	https://en.wikipedia.org/wiki/Hayes_command_set
M27	ESP32 Bootloader Message Codes	https://github.com/espressif/esp-idf/blob/release/v3.0/components/esp32/include/rom/rtc.h#L80
M28	ESP32 Serial UART Bootloader Description	https://github.com/espressif/esptool/wiki/Serial-Protocol
M29	8bit Shift Register	https://www.sparkfun.com/products/13699

9.5 Cited Standards

The following is a table covering additional resources for the standards and protocols mentioned in the paper.

#	Name	Link
S1	IEEE 802.11	https://ieeexplore.ieee.org/document/7786995
S2	ARM v7 Architecture	https://static.docs.arm.com/ddi0403/ed/DDI0403E_d_armv7m_arm.pdf
S4	JSON	https://tools.ietf.org/html/rfc8259
S5	REST API	https://en.wikipedia.org/wiki/Representational_state_transfer
S6	I2C	https://en.wikipedia.org/wiki/I%C2%B2C
S7	SPI	https://en.wikipedia.org/wiki/Serial_Peripheral_Interface
S8	Arduino Wifi101 Library	https://www.arduino.cc/en/Reference/WiFi101
S9	REST Dissertation by Roy Fliedling	https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
S10	C++17	https://isocpp.org/std/the-standard
S11	C++ Wiki	https://en.wikipedia.org/wiki/C%2B%2B
S12	EcmaScript 2018	http://ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
S13	JavaScript Wiki	https://en.wikipedia.org/wiki/JavaScript
S14	Asynchronous Serial Communication Wiki	https://en.wikipedia.org/wiki/Asynchronous_serial_communication
S15	UART Wiki	https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
S16	1-Wire Bus Protocol	https://en.wikipedia.org/wiki/1-Wire
S17	MaxDetect 1 Wire Bus Protocol	https://cdn.sparkfun.com/datasheets/Sensors/Weather/RHT03.pdf

#	Name	Link
S18	IEEE 802.11 Wiki	https://en.wikipedia.org/wiki/IEEE_802.11
S19	FCC CFR Title 47 Part 15	https://www.govinfo.gov/content/pkg/CFR-2010-title47-vol1/pdf/CFR-2010-title47-vol1-part15.pdf
S20	FCC CFR Title 47 Part 15.23 - Home built devices	https://www.govinfo.gov/content/pkg/CFR-2010-title47-vol1/pdf/CFR-2010-title47-vol1-sec15-23.pdf
S21	FCC CFR Title 47 Part 15.103 - Exempted devices	https://www.govinfo.gov/content/pkg/CFR-2010-title47-vol1/pdf/CFR-2010-title47-vol1-sec15-103.pdf

9.6 Copyrights

All visual supportive elements used in this document were created by the group members hence no explicit Copyright permission requests were required.