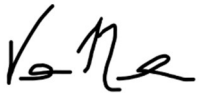
Ohmost Done

Brooke Bonfadini, Sonia Aggarwal, Chloe Tran, CJ Rogers, and Victoria Nilsson,

December 10, 2020

Capstone Design ECE 4440 / ECE4991

Signatures



Statement of Work:

Sonia Aggarwal

My main contribution to this project was building an app, setting up the server with WIFI module connections, and organizing the team. While exploring our options for the WIFI module, an important consideration was whether or not this would be doable within the time and budget constraints. To find out, I set up a meeting with Brad Campbell - a Networks professor at UVA to guide CJ and I through our options and help us come up with a reasonable solution. After this conversation, we knew we needed a mobile application.

I am also the designated scrum master or project lead for the team, which means I set up all the meetings, make the calls about when we should have things done, and keep track of the team progress throughout the project. I have created the Gantt charts detailing our expected timelines and deadlines, both originally and updated. I have also created the team calendar, with recurring deadlines and meetings that the Gantt chart outlines for ease of use. On top of this, I start meetings with our stakeholders (Professor Powell and Professor Barnes) with the 30s elevator pitch for the software side of this project. And lastly, I created all the software test plans with input from CJ. I did this by making sure we had a solid understanding of what was needed to accomplish our goals laid out in the proposal. I paid particular attention to the ADC output values and packet retrieval and as such, a test plan was created for every step of the software development cycle.

Brooke Bonfadini

Within this project, I contributed to numerous aspects of the electrical hardware as well as mechanical components. In the design stages of the project, I found appropriate soil moisture sensors, voltage regulators, and voltage-controlled fuses as well as created a full system diagram for visualization. I assisted in the creation of the Multisim schematic of the header-board created for the microcontroller used in the project as well as created several Ultiboard footprints. Additionally, I was responsible for designing and placing footprints on the PCB in Ultiboard. Once parts were acquired, I tested the water pumps as well as the LEDs on breadboards with a VirtualBench to ensure they worked properly before the PCB came in. After the PCB arrived, I was responsible for all the soldering in the project as well as putting the board together. I also assisted with all hardware testing after parts were placed on the PCB.

As the project progressed, I was responsible for creating the mechanical structure as well as assembling and wiring all the components on the structure. I was present at all lab meetings to help with both hardware and software testing as well as to video the final product. After I gathered video in the lab of the project, I created the final video demonstration using a video editing software.

Victoria Nilsson

My main role for this project was helping with hardware design and testing. During the planning stages, I worked with Chloe and Brooke to come up with a schematic for the hardware components. I also helped with finding components to replace the motor driver for the pumps

after our advisor suggested we take additional precautions to protect our PCB. In terms of the schematic, I laid out the soil moisture and water level sensors as well as the electronic fuses. When it came time to make the PCB, I added the power traces and assisted with making footprints. For the second PCB sendout, I helped redo the footprints for the PCB connectors and the voltage regulator; I also redid some of the traces in order to not entrap the ground of the jack connector.

During the midterm review process, I made the hardware test plan, which was edited after the electronic fuses were added. I tested the water level sensor on the breadboard by replicating the voltage divider network. I also tested the power supply on the breadboard itself and noticed the resistors overheating; after consulting Professor Powell, it was concluded that our fuse layout was incorrect. I worked with Chloe and Brooke to resolder the resistors and test the fuses for the peristaltic pumps and LEDs. The soil moisture and water level sensors were also tested by myself, Chloe, and Brooke.

Catherine "CJ" Rogers

During the course of this project, my main role was to design and complete the firmware for the project. This included determining what types of I/O pins and how many we would need for the project, and then finding a board that matched these requirements. Once our requirements included network communications, I worked with Sonia to find a wifi module that would work seamlessly with the board that was chosen. With the board chosen I made sure to coordinate with the hardware team of Victoria, Booke, and Chloe determine which pins would be used for each of our sensors, LED, and pumps, and took care to communicate back and forth with them anytime one of us would have to make a modification that would affect the other. With this information I was able to design and implement the main loop for the automatic watering system that would monitor and interact with the plants.

Another major part of this role was to ensure the interface between the microcontroller and the server. Sonia and I spent a lot of time researching the best method to do this and found resources that would help us implement the features that we wanted. This included setting up a few meetings with more experienced individuals with whom I discussed expectations and possible routes the project could take within our time frame. I also found example projects that were instrumental in completing the network functionality of the project that I modified to make work in the context that we needed. I also worked with Sonia to test and coordinate the structure of the information that would be sent to her application.

Chloe Tran

My main role on the team was working on the design of the PCB and a supportive role in the design of the mechanical structure. For the PCB, I found the water level sensor, LED lighting, and water pumps for the system and then worked together to integrate them into a Multisim schematic. I worked on the power supply subsection to find a wall transformer suitable for the amount of current that our system was drawing. For the PCB layout, I made footprints for the header board connectors, jack connector and did a portion of the routing. I also fixed any

FreeDFM errors at the times of board sendouts. I worked with Victoria and Brooke to complete hardware testing and helped troubleshoot the electrical problems that arose with the PCB.

For the mechanical structure, I worked with Brooke to design the plant platform. I created a 3D CAD model with specific dimensions which was handed off to Brooke as a guide while physically building it. We also worked together to figure out how the electrical enclosure would attach to the structure and how the wiring would be routed to get to those components located on the structure itself while keeping the electrical parts isolated from water exposure. Lastly, I was responsible for keeping a running bill of materials and ensuring that parts were ordered correctly from Digi-Key.

Table of Contents

Contents

Capstone Design ECE 4440 / ECE4991 1
Signatures 1
Statement of Work: 2
Table of Contents 5
Contents 5
Table of Figures 7
Abstract 9
Background 9
Constraints 11
  Design Constraints 11
  Economic and Cost Constraints 12
External Standards 12
  Tools Employed 13
Ethical, Social, and Economic Concerns 14
  Environmental Impact 14
  Sustainability 14
  Health and Safety 15
  Manufacturability 16
  Ethical Issues 16
  Intellectual Property Issues 17
Detailed Technical Description of Project 18
  Hardware 19
  Mechanical Structure 27
  Firmware 28
  Software 34
Project Timeline 36
Test Plan 38
Final Results 47
Costs 49
Future Work 50

References 52
Appendix 56
Appendix A: Thermal Calculations 56
Appendix B: PCB Layout Layers 57
Appendix C: Microcontroller and WIFI Module Pin Out Sheets 59
Appendix D: Network Testing 60
Appendix E: Firmware Clock Code Listing 61
Appendix F: Analog to Digital Conversion Code Snippets 63
Appendix G: General Purpose Input/Output Code Snippets 65
Appendix H: Server and Application Code 68
Appendix I: Soil Moisture Sensor Calibration Testing 69
Appendix J: Plant Profiles 70
Appendix K: Finances 70

Table of Figures

Figure 1: Main Block Diagram 19
Figure 2: Full System Hierarchical Design Schematic 19
Figure 3: Water Level Sensor Schematic 20
Figure 4: Soil Moisture Sensors Schematic 21
Figure 5: LED Lighting Subsystem Schematic 21
Figure 6: Water Pumps Subsystem Schematic 22
Figure 7: Power Supply Subsystem Schematic 23
Figure 8:PCB Header Board Layout. Ground Plane Visible 25
Figure 9: PCB Header Board Layout. Ground Plane Hidden 25
Figure 10: Populated PCB. Front Side 26
Figure 11: Populated PCB. Back Side 26
Figure 12: Auxiliary View of Structure 27
Figure 13: High Level overview of the Network Communication System 28
Figure 14: Main Project Loop for Auto Watering System 33
Figure 15: Login Screen 34
Figure 16: Basil and Mint Options Screen 35
Figure 17: WIFI Module and Mobile Application communication to Firebase BaaS 36
Figure 18: Parallel and Serial Tasks 36
Figure 19: Original Gantt chart 37
Figure 20: Midterm Review Gantt chart 37
Figure 21: Power Supply Testing Procedure 38
Figure 22: Soil Moisture Testing Procedure 39
Figure 23: Water Level Sensor Testing Procedure 40
Figure 24: LEDs/Pumps Testing Procedure 41
Figure 25: Software Test Plan - Water Level and Soil Moisture Sensors 42
Figure 26: Software Test Plan - LEDs and Water Pump System 43
Figure 27: Software Test Plan - UART 44
Figure 28: Software Test Plan - WIFI Module 44
Figure 29: Software Test Plan - Server 45
Figure 30: Software Test Plan - Mobile Application and Database 46
Figure 31: Completed System 47
Figure 32: NEMA Enclosure Structure 47
Figure 33: Copper Top 57
Figure 34: Copper Bottom 58
Figure 35: Silkscreen Top 58
Figure 36: Microcontroller Pin Sheet 59
Figure 37: WIFI Module Only Pin Sheet 59
Figure 38: Pin Sheet for Project 59

Figure 39: Network Trace for WIFI Module 60
Figure 40: Debugging Terminal for WIFI Module 60
Figure 41: POST Request Template Format 60
Figure 42: createString implementation 61
Figure 43: InitClk function. Initializes the main clock on the controller and enables interrupts. 61
Figure 44: “Get” functions for each fraction of time 62
Figure 45: Timer 2 Interrupt Service Routine for global timer values 62
Figure 46: Initialize the ADC 63
Figure 47: ADC ISR 63
Figure 48: ADC Circular Buffer Averaging Functions 64
Figure 49: GPIO Initialize Functions 65
Figure 50: GPIO Turn On Functions 65
Figure 51: GPIO Turn Off Functions 66
Figure 52: GPIO Toggle Functions 66
Figure 53: GPIO Status Functions 67
Figure 54: GPIO Enable/Disable LED Functions 67
Figure 55: REST API used for Firebase Functions 68
Figure 56: Database Retrieval for Mobile Application 69
Figure 57: Medium Soil Moisture Analog Sensor Value 69
Figure 58: Medium-Wet Soil Moisture Analog Sensor Value 69
Figure 59: Wet Soil Moisture Analog Sensor Value 69
Figure 60: Bill of Materials. Breakdown of System Costs at 1 Unit 70
Figure 61: Bill of Materials. Breakdown of System Costs at 10000 Unit 71
Figure 62: Breakdown of Total Project Cost 71

Abstract

The proposed technology is a semi-autonomous vertical plant management system. The heart of the management system is a microcontroller (MSP) that will monitor I/O inputs and automate three central processes: dispensing water, turning LEDs on/off, and checking the water reservoir level. Soil moisture and water level sensors survey plant growing conditions and report sensory information to the microcontroller. The microcontroller intakes sensory information and determines when to turn on/off the LEDs as well as the peristaltic pumps. I/O pins from the microcontroller, set as outputs, are electronically connected to fuse enable pins. To turn on a particular component, its fuse enable pin is set to high which allows power to flow from the board to the external hardware. Sensory information from the system is sent from the microcontroller to a mobile application via a WIFI module. The mobile application acts as a user interface that allows owners to monitor growing conditions in parallel with the microcontroller; users are also alerted about important system updates, such as low water reservoir level. By automating three central processes of plant upkeep, the responsibility shifts from the plant owner to the semi-autonomous system. Furthermore, displaying sensory information on an app adds an interactive element that will hopefully increase user engagement. Finally, the automated system reduces the stress involved with plant upkeep while away from home.

Background

Vertical farming technology optimizes growing space by stacking crops on top of each other as opposed to having crops grown spread out. In addition, vertical farming systems make use of controlled environment agriculture (CEA) to produce more yield [1]. In CEA, the control factors can include light, soil moisture, temperature, humidity, carbon dioxide levels, and nutrient pH. All these factors can be controlled through the use of environmental sensors and actuators. Since this type of system works well in agriculture, our team saw an opportunity to transition aspects of vertical farming and CEA to a plant management system that takes care of indoor plants instead of crops.

Indoor plants are a popular way to bring greenery inside especially for those living in more urban areas. Having plants inside a space has promising benefits such as air purification, mood improvement, and an increase in creativity [2]. However, having lots of plants requires a lot of preparation and time for the owner. Each plant needs to be individually potted and might need to be placed in different locations since they might have varying lighting and watering requirements. This dispersed location of plants can be space consuming and differing plant care needs can be complex to track.

Therefore, the plant management system combines vertical farming and CEA features. It is a two-tiered vertical structure that condenses indoor plants to a central location and offers personalized watering and lighting control for each type of plant on the two separate tiers. Only two tiers were chosen due to time constraints but more tiers and thus more plant types can be added in the future. Watering and lighting control were chosen since they are the parameters that would be most variable in an indoor environment. Each tier has a plant profile that can be changed depending on the type of plant that the user wants on each tier. A plant profile includes

the soil moisture level that a specific plant type should stay at and how much light per day the plant type should receive.

Many plant management system conceptual designs or prototypes do exist in academic research. The following table summarizes the five systems that are most similar to our team's design:

Academic Paper Reference Number	**[3] Completed Prototype**	**[4] Completed Prototype**	**[5] Proposed Prototype**	**[6] Completed Prototype**	**[7] Completed Prototype**
Indoor or Outdoor Use	Indoor	Indoor	Indoor	Outdoor	Outdoor
Structure	4-tiered vertical shelf plant shelf	Individual potted plant	3D printed smart network of individual flower pots	Automatic irrigation system integrated into an outdoor garden	Automatic irrigation system integrated into an outdoor garden
Sensors	Soil moisture	Soil moisture, light, water reservoir level	Soil moisture, water reservoir level	Soil moisture, light, sonar	Soil moisture, light, water reservoir level, IR, gas, temperature, humidity
Control Parameters	Soil moisture	Soil moisture	Soil moisture	Soil moisture, light	Soil moisture
Automatic or User Control	Notification of low soil moisture is sent to the user who then turns on the watering system through a web browser	Automatic watering system is turned on when soil moisture is below a threshold and sunlight is available	Automatic watering system based on soil moisture threshold	Automatic watering system is turned on when soil moisture is below a threshold value Light bulbs are turned on manually	Automatic watering system is turned for a certain amount of time based on soil moisture and temperature thresholds
Microcontroller?	Yes	Yes	Yes	Yes	Yes
Type of User Interface	Web browser to turn on watering system	SMS sent to user when water reservoir level is low	Phone application	Remote control allows user to activate and deactivate system features	Android application to monitor sensor data, and watering system pumps

Table 1: Summary and Comparison of Similar Plant Management Systems

All similar systems [3] – [7] use a microcontroller as the central control unit, soil moisture as a control parameter, a watering system that is either automatic [4] - [7] or user controlled [3] based on a set soil moisture threshold, and some kind of user interface. Our team's design also uses a microcontroller, a soil moisture sensor, and a phone application user interface. The novelty of our design from projects [4] - [7] is the addition of personalization of the soil moisture and light parameters for each plant type contained in the same system. This is done through the use of plant profiles embedded in the code. Design [3] which is an individual

network of potted plants does allow for light and water threshold personalization for different plant types but it does not include automatic lighting control, instead relying on environmental light and only having automatic watering capability.

Many plant management systems also exist on the commercial side. The products on the market most similar to our design are *iPřiro's* programmable timed watering system and *Click and Grow'*s hydroponic smart garden [8] - [9]. *iPřiro's* product is a system of tubing connected to a water reservoir that turns on at a set time every day [8]. This makes watering an easy task but there is not the ability to personalize watering by plant type or have automatic lighting. *Click and Grow'*s smart garden personalizes plant nutrients based on plant type and has automatic lighting control. It even has a compact, space saving design [9]. However, the smart garden uses hydroponics which is growing plants in water instead of soil. This is because Click and Grow's smart garden is meant to grow crops that are edible and hydroponics is meant to optimize yield [10]. Not all plants are necessarily suited for hydroponic growth therefore our plant management system uses soil in order to have the option to maintain a wide variety of indoor plants including edible ones.

This vertical plant management system employed our team's knowledge gained through various previous coursework. The FUN series (ECE 2630, ECE 2660, ECE 3750) taught us how to carry through a PCB project with a group integrating from signal processing essentials, hardware prototyping and testing, and PCB design. This directly helped some team members be prepared for now doing this process more independently in creating a header board PCB. Power supply basics emphasized in the FUN III (ECE 3750) provided a good knowledge basis in creating a reliable power distribution network to all components of the project. In addition, Embedded Computer Systems (ECE 3430) taught us how to code in C to control attachments to an MSP microcontroller. This was useful as it was exactly what we had to do for our plant management project as we had a microcontroller take in readings from input sensors which then directed the actuation of connected components based on the C code. Lastly, skills gained from Computer Networks (CS 4457) was useful for some team members to connect the microcontroller to a phone application through a WIFI module and have information pass through the channel dependably.

Constraints

Design Constraints

Our design had to include a printed circuit board and an embedded CPU capable of demonstrating real time response since our team is composed of both electrical and computer engineers. This set the initial constraints to some of the factors in our project.

CPU Limitations

We were limited in CPU choice through the number of I/O pins our system needed. We chose the MSP-EXP430F5529LP [11] launchpad because it had enough GPIO pins in order to handle the number of sensors and actuators that were used in our project. This MSP variant also had the necessary pins to enable a WIFI module booster pack [12].

Software Availability

The circuit design and simulation software that is available to UVA students is Multisim [13] and Ultiboard [14] and thus these were the tools used to design the PCB. The microcontroller was programmed with Code Composer Studio [15] which was free with the purchase of the launchpad and also familiar to the team as it was used in the Embedded Computer Systems (ECE 3430) course previously.

Manufacturing Limitations

Advanced Circuits, a PCB manufacturing company, had certain requirements for the PCB designs that were to be manufactured by them [16]. These requirements included:

1. Maximum board size of 60 square inches
2. Minimum 0.005" line/space width
3. Minimum 0.010" hole size
4. Maximum 50 drilled holes per square inch
5. No internal routing
6. No scoring, tab rout, or drilled hole board separations
7. No controlled impedance/dielectric
8. No countersinks/counterbores/castellated holes
9. No cavities/controlled depth

Economic and Cost Constraints

The budget constraint for this project was $500. If all the material needed for this project were to be bought new, this budget would have been exceeded. Money was saved through the use of already existing wood and construction materials owned by Brooke for the physical structure and the use of plants that were bought externally by Chloe to be used for testing purposes. A water level sensor was the largest economic constraint. A more professional and sensitive water level sensor was around $100 [17]. Agreeing that the sensor was too expensive, a $40 plastic sensor meant more for electrical hobbyists was chosen as a tradeoff [18].

External Standards

To ensure this project met important safety and professional standards, external standards such as IEEE, the NEC, NEMA, and IPC were closely followed throughout the design and development stages. Since the system is classified as low voltage and able to be connected to a typical receptacle, no external utility standards were required to be addressed. Additionally, since the project did not employ AC line voltages anywhere other than through the use of a wall transformer, NEMA regulations were addressed in the design of the transformer which was chosen to meet these standards. However, NEMA enclosure recommendations were followed to select an enclosure that would secure the project's circuitry properly [19]. Through the use of NEMA recommendations, it was determined that an enclosure of at least a NEMA 2 rating would be necessary for the project. The National Electric Code (NEC) was also addressed throughout the development of this project. In order to meet NEC standards, the aluminum alloy wiring used throughout the device was selected to be made of AA-8000 series electrical grade

alloy [20]. Additionally, the wiring adheres to ISO standard 10993-1 which states the wiring is able to come in contact with the human body without causing irreversible damage [21]. Since the system includes a man-made water reserve, the following requirements were also taken into account [20]:

(1) 15 Volts (RMS) for AC Sinusoidal
(2) 21.2 Volts Peak for AC Non-Sinusoidal
(3) 30 Volts for Continuous DC
(4) 12.4 Volts Peak for DC that is Interrupted at a Rate of 10 to 200Hz

The designed system stayed well below the voltages required in the standards above. In addition to these voltage requirements, it was also recommended the electrical components be stored off the ground so that in the event of leakage the equipment will avoid the possibility of sitting in water. This led to the NEMA enclosure housing the circuitry as well as the microprocessor being mounted to the side of the structure. During the design of the circuitry and PCB, the following IPC standards were considered [22]:

(1) IPC-CM-770E: Guidelines for Printed Board Component Mounting
(2) IPC/JPCA-2291: Design Guideline for Printed Electronics

The IPC standards allowed for correct spacing of traces and components on the PCB. Additionally, since our PCB consisted of surface mount components, the guidelines on their proper mounting were addressed. The device also incorporates a WIFI module, meaning the device is an intentional radiator and FCC as well as IEEE standards must be addressed. In accordance with FCC standards, the WIFI module operates below 10GHz and the bandwidth ranges from 2.4GHz to 5GHz. Additionally, the field strength within the frequency range described does not exceed 3000 microvolts/meter/MHz at 3 meters for all directions and the field strength is less than 100 microvolts/meter/MHz at 3 meters for all directions for frequencies above the described range [23]. These requirements were met by the WIFI module that was purchased, and the team did not have to address them specifically. As per IEEE 802.11, the clock oscillator is greater than 10KHz which ensures reliable and secure connection and the module operates in either the 2.4GHz or 5GHz band [24]. The 802.11 standard also outlines a structure for data that is transmitted. The structure includes a MAC header, payload, and frame check or Cyclic Redundancy Check (CRC). This was included in the WIFI module as well as included in the code that was written for the microcontroller.

Tools Employed

The tools used to implement this design can be broken down into tools related with hardware, firmware, and software respectively.

Hardware

Multisim [13] was used to lay out the circuit schematic of the header board design through the use of hierarchical block design. Afterwards, the design was forward annotated into Ultiboard [14] which was used to layout the physical PCB itself. Brooke, Victoria, and Chloe all

worked together on this design as the hardware team. Although they all had previous experience with these tools through previous coursework, help from our technical advisor improved our skills in interacting with these tools. After the PCB was designed, a Virtual Bench [25] was used to do hardware testing to ensure that all subsystems were receiving proper levels of voltage and current.

Mechanical

Autodesk Inventor [26] was used to prototype the plant platform mechanical design in 3D. The role of this program was to provide clear dimensions to make constructing the physical structure a more straightforward task.

Firmware

The main tool used in creating the firmware was Code Composer Studio [15]. This tool is an IDE from Texas Instruments that is highly compatible with the MSP430. This makes it easy to develop and debug on the device all from one program. The code was all written in Embedded C. These tools were already familiar to the team from Intro to Embedded and so was an easy pick for developing the firmware. Another tool that was used was Wireshark [27] to validate the data coming to and from the WIFI module. CJ and Sonia had previously used this tool in Computer Networks so were able to use it efficiently to accomplish the desired tasks.

Software

For the user interface (UI) Android Studio [28] was used as the integrated development environment (IDE) for building the mobile application. The entire system was written in Kotlin, compiled in Android Studio and run on a physical device, Samsung Galaxy S10. The application was in direct communication with Firebase [29], a Backend as a Service (BaaS) provider. The Firebase services used were as follows:

- Firebase Cloud Firestore - to store data coming in from the microcontroller
- Firebase Functions - to correctly parse the data coming in from the microcontroller that would then be sent to the mobile application

Firebase was very new to all members of the team and as such there was a slight learning curve, but Sonia and CJ improved their knowledge on networking and HTTP protocol whilst working with this BaaS.

Ethical, Social, and Economic Concerns

Environmental Impact

During the lifetime of this product, several environmental considerations must be addressed.

Sustainability

The structure of the project is made out of treated lumber and waterproof stain. Research shows that these treatments do release a small amount of chemicals into the environment nearest to the lumber containing the treatment. Despite no dangerous biological impacts observed, it is

important to take into consideration the use of appropriate treatment practices to ensure no toxic chemicals are released during manufacturing [30]. Additionally, the use of stains and varnishes to render wood water-proof often results in the emission of formaldehyde which could result in lower air quality indoors, which means coatings for the lumber should be carefully chosen to ensure minimal disruption of air quality [31]. In the development of the project, lumber stain was chosen so that no harmful chemicals would be released, but this should be considered during manufacturing as well as in the development of future technologies similar to this project. In addition, this project uses two types of plastics in its structure. Plastics have been known for quite some time to have negative environmental impacts in both production and disposal. Creation of plastic results in the emission of carbon dioxide, leading to global warming and rise of sea levels, and disposal of plastic results in pollution of land and sea environments [32]. Due to budget constraints, this project uses plastic that is not long-lasting meaning it will result in several iterations of plastic waste throughout its lifetime. In order to mitigate the environmental impacts of the structure design of this product, if the product was to go into mass manufacturing, it would be important that the plastic used was swapped for a sturdier plastic that would need to be replaced less frequently, reducing the amount of plastic waste. In addition, it is essential to choose plastic products that are able to be recycled as well as to ensure when the system is disposed of, the plastic components are sent to a recycling facility instead of a landfill. Additionally, the printed circuit board as well as the microcontroller used in the device are able to be recycled according to the EPA. As long as these components are properly recycled upon disposal, the environmental impact is minimal since many of the pieces of the PCB and microcontroller are able to be reused once they are extracted from the boards [33]. The amount of waste this product produces is also mitigated by choosing sensors that are non-capacitive, meaning they are less likely to corrode due to moisture exposure. This increases the lifetime of the product, and therefore, reduces the number of times the system needs to be replaced. This product also has positive environmental impacts in its functionality. Since the system delivers specialized care to the plants, it reduces water consumption by preventing overwatering. Additionally, since the plants are receiving improved care, less fertilizers may be necessary to sustain plants. Fertilizers contain harmful chemicals that can runoff into streams and rivers, polluting wildlife [34]. This system may reduce the need for these chemicals. Lastly, the product's ability to allow users to maintain plants easier, results in the need for less pre-packaged items to be purchased at stores, reducing packaging waste.

Health and Safety

During the design of this project, consumer health and safety was an important consideration. In order to ensure maximum operating safety, components used in the system were waterproofed, or at the very least, water resistant. For example, the NEMA enclosure ensures no debris or moisture can interfere with the circuitry so there are no chances for shorts or arc flashes. Additionally, all wires were covered with waterproof heat shrink wrap in addition to being rated for chemical as well as water exposure. A strict following of electrical standards described in the *Standards* section, also ensures the product's maximum safety. Measures include adequate handling of ground faults, shorts, and rated equipment that will protect the user in the event of an unexpected electrical malfunction.

Manufacturability

This vertical plant management system is relatively simple to manufacture. All components are currently available from their suppliers. However, an area of concern could be the low availability of the water level sensor and LED lighting if this system were to be manufactured at larger quantities in which case substitute parts should be arranged. As of December 2020, the water level sensor and LED lighting have only 37 and 25 in stock respectively from Digi-Key [18], [35]. All PCB parts are extensively available from Digi-Key so there is no concern on part availability on that front. The PCB requires the soldering of mainly through hole parts but also includes soldering of surface mount electronic fuses. These fuses present the largest challenge in soldering due to their smaller package size. The use of automation to place components on the PCB would improve precision in soldering while also lowering manufacturing costs compared to manual soldering. Automated soldering would also drive down manufacturing time, bringing more systems to completion in a shorter time frame. On the mechanical side, the structure itself that holds all of the system components is made of wooden boxes, plastic bins, and connecting material such as wiring and caulking. The design of the mechanical structure is straightforward which makes it easy to reproduce. This design is discussed further in the Technical Description section.

Ethical Issues

At the surface, this product appears harmless; however, there are ethical considerations to keep in mind. Water conservation is a principal ethical concern. While plants absorb carbon dioxide and emit oxygen, many geographical areas are experiencing water shortages. There is debate as to whether diverting water for this management system will further exacerbate the crisis, even on a very small scale. As mentioned earlier, this system intermingles water and electricity. There is an inherent safety concern for the user. It is imperative the system adheres to relevant safety codes to ensure no pertinent danger. Autonomy is another important ethical concern. Many individuals fear that autonomous technology will take over jobs or repurpose them. Although the autonomous application appears insignificant, future applications could expand to greenhouses as well as plant nurseries after successful proof of concept.

The management system does not inherently favor any one group of people. While it is tailored towards urban residents, individuals of any gender and ethnicity may enjoy this product. Furthermore, the system could reduce the socioeconomic disparity in terms of accessing fresh herbs and produce. Since fresh produce, especially organically grown produce, have large price tags, individuals may not be able to afford them. Ultimately, growing plants at home would eliminate the economic burden and may alleviate the effects of food deserts.

While the microcontroller uses a WIFI module to connect to the telephone, data mining and privacy issues rank low among the list of ethical concerns. App usage will solely require product details and not any personal information. The sensory information obtained revolves around the system performance.

Intellectual Property Issues

[36] presents a patent for a “Method and System for Automated Data Analysis of Soil Moisture”. The technology presented is classified as “the ability to detect and report the actual moisture levels in the soil being irrigated”. The main independent claim is a “method for irrigating a Zone of a planted field, comprising the steps of: irrigating the soil in said Zone; monitoring the soil moisture content at predetermined depths within said Zone; determining a refill point of said soil in said Zone responsive to said monitoring; and second irrigating said soil responsive to said refill point determining.” This is backed up by the dependent claim that “said refill point determining comprises the steps of determining a peak moisture point of said soil at said predetermined depths; identifying a moisture rate change perturbation of said Soil moisture at said predetermined depths; identifying the deepest said predetermined depth at which no said perturbation can be identified; and identifying said refill point responsive to said identified deepest predetermined depth at which no said perturbation can be identified.” In light of these claims, the presented capstone is still patentable because the system built by Ohmost Done does not take into consideration soil depth and how this depth will affect the soil moisture sensor which is used to activate the water dispense system.

[37] presents a patent for a “Soil water potential detector”. The technology presented is classified as a “probe made of an agglomerated material and designed to be inserted in a Solid medium e.g. a Soil in which the humidity level or liquid Saturation can vary.” The main independent claim is a “water detector for determining water availability in a Soil for use in plant cultivation, which comprises: at least one water probe adapted to be introduced a predetermined depth into Said Soil; Said water probe comprising a light impervious material capable of absorbing water and gradually becoming more translucent as water is being absorbed therein; 1O 15 25 8 means associated with Said water probe for detecting light translucency in Said light impervious material following water absorption therein; and Signal means operating in response to Said light translucence detecting means to determine degrees of Said light translucency corresponding to quantities of water in Said Soil and indicate whether or not Said Soil is in need of additional water. ” This is backed up by the dependent claim “which comprises converting Said results to additionally determine the quantity of Said liquid in Said Solid.” Although these claims insist that a water detector is used in the soil in its independent claims, and converting the results of these detectors for notification purposes which closely resembles the capstone project presented, the use of probe becoming translucent as light and water vary, is not part of this project and therefore makes it patentable.

Lastly, [38] presents a “Method and apparatus for optimization of growth of plants". This technology aims to “increase the growth of plants in a greenhouse or other protected area”. One of the independent claims relating to the method is that the measured greenhouse parameters are “light, temperature and carbon dioxide sensing”. Light measurement is a factor in controlling carbon dioxide levels and is not being controlled itself. This is supported with the statement that “regulating with the computer the temperature of the atmosphere and the amount of carbon dioxide in the atmosphere as a function of the measured amount of light at a preselected atmosphere temperature range in the greenhouse…”. An independent claim relating to the apparatus includes the use of “control means including a computer connected to a

microprocessor". The microprocessor acts as a control unit connected to a computer "wherein the computer through the microprocessor changes the temperature by opening the vent", as stated in the dependent claim. The claims asserted depict similar plant management features as our team's design including the use of a central microcontroller and the use of sensors to trigger outputs. However, our team's system detects soil moisture and water level instead of light, temperature and carbon dioxide. Our team's system is also not meant for a larger scale greenhouse use and instead for indoor personal plant management through its compact vertical structure. With the claims presented in this patent, the presented capstone project remains patentable.

Detailed Technical Description of Project

The vertical farming plant management system is an assistive tool to plant owners. As urban development increases, outdoor yards are becoming exceedingly rare. However, indoor vertical planting, which takes advantage of the limited space in urban settings, has emerged as an up-and-coming hobby. Ultimately, individuals who are new to the hobby often fail to water their plants or end up overwatering them. Likewise, during vacations, owners find it difficult to keep their indoor plants alive. As such, the vertical plant management system will help alleviate the burden of owning indoor plants by acting as an autonomous light and water control system.

Two plant profiles exist on the user interface, each consists of average soil moisture levels and amount of artificial light for a particular plant. Given quotas for light and soil moisture, the system behaves semi-autonomously. The microcontroller takes in I/O input from the soil moisture sensors and uses that information to decide when to withhold and dispense water from a reservoir tank. If the microcontroller determines the moisture levels are too low for a particular profile, it enables a fuse pin that turns on a pump and allows water to flow. The microcontroller also monitors the amount of water in the reservoir. Finally, the microcontroller controls the artificial lighting such that the plant receives the allotted amount of light for its individual profile. The sensory information which flows to the microcontroller is transmitted via a WIFI module to an app that acts as the user interface; thus, the user can monitor the plant status along with the microcontroller. The user also receives important system alerts on the app, such as when the reservoir level is low.

Figure 1 outlines the interactions between each of the subsystems, microcontroller, and plant platform. The functionality and circuitry will be further explained in subsequent sections.

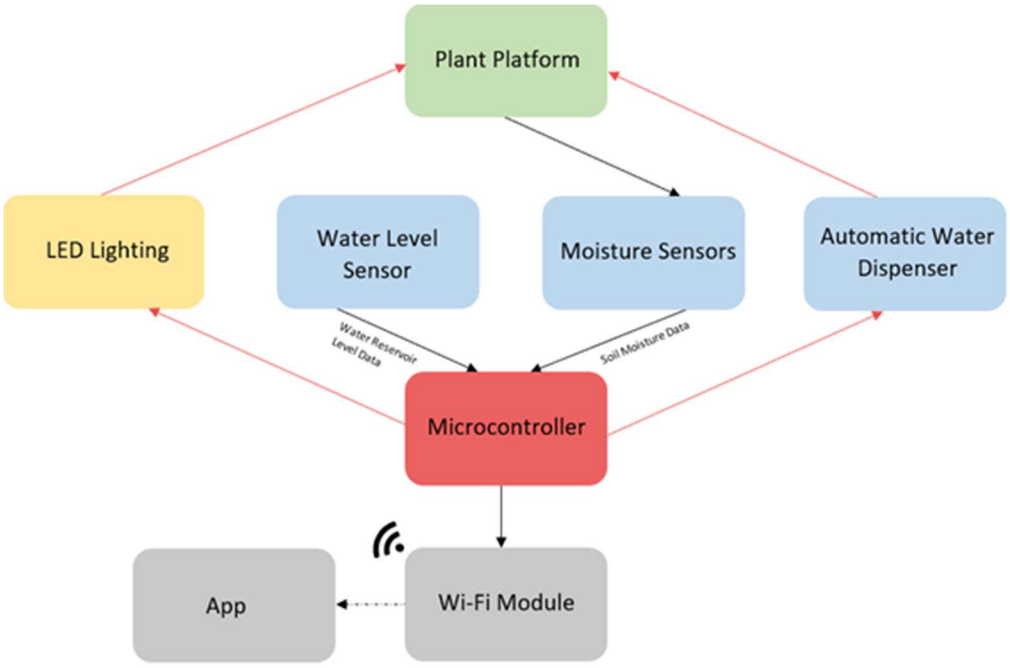


Figure 1: Main Block Diagram

Hardware

Full Schematic

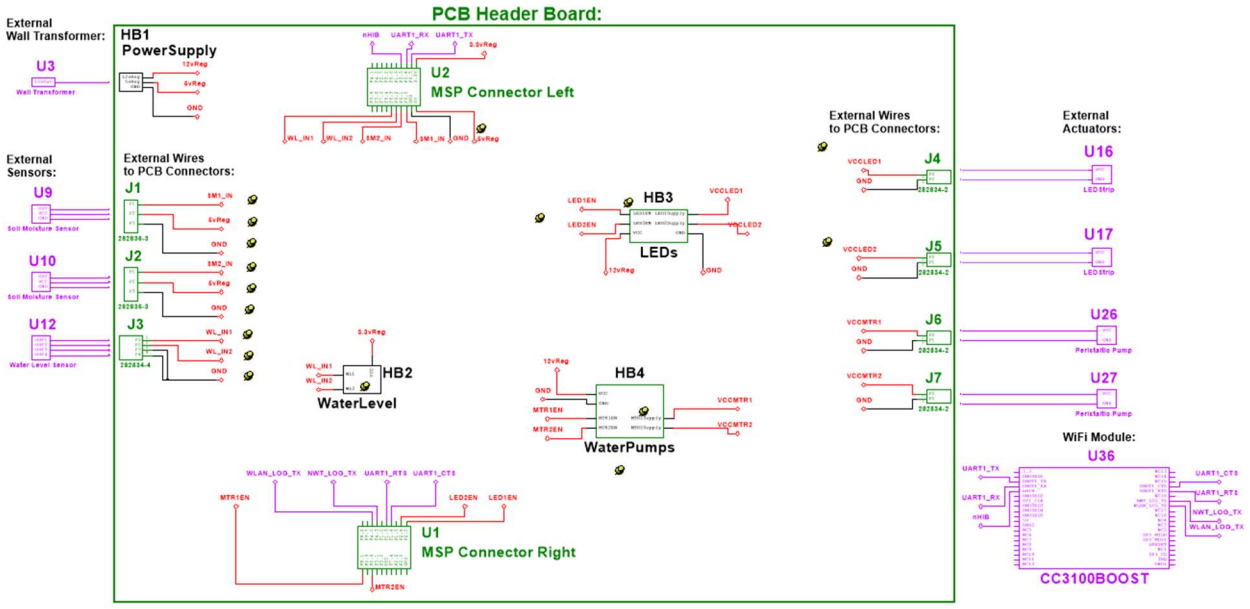


Figure 2: Full System Hierarchical Design Schematic

The final schematic for our electrical design is shown in Figure 2. The subsystems for the plant management system are represented as hierarchical blocks and include the water level sensor [18], the water pumps [47], and LEDs [35]. The individual schematic for each subsystem will be further explained in upcoming sections. The green boxes, entitled J1-J7, represent connectors [44] that bridge external circuitry to the printed circuit board. Components requiring

connectors include the two soil moisture sensors [46], the water level sensor, the two peristaltic pumps, and two neon LED strips. Another important aspect of the schematic are the connections from the microcontroller. The WIFI module [12] sits on top of the MSP430 launchpad [11]; the PCB rests on top of the WIFI module. Header board connectors [45] establish an electrical connection between the MSP430 and the PCB, which allows the microcontroller to set output pins high and read incoming sensory information. Markings on U1 and U2 show the pins on the MSP430. Corresponding connections to sensors, pumps, and LEDs are indicated by hierarchical connectors.

Sensors

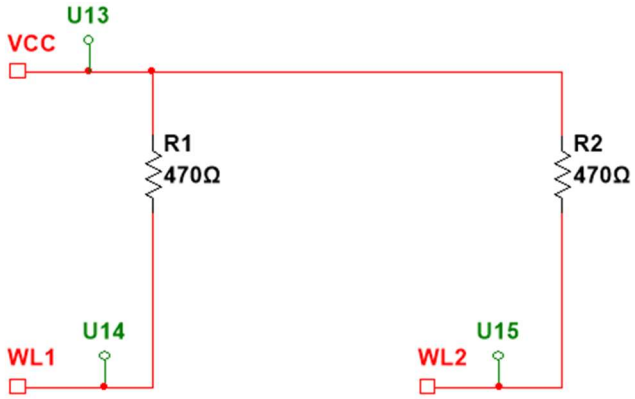


Figure 3: Water Level Sensor Schematic

The water level sensor [18] acts as a potentiometer. There are four pins on the water level sensors. Pins 1&4 represent a reference resistance reaching approximately 1900 Ohms when the water reservoir is empty. Pins 2&3 represent the sensor resistance, which changes as the sensor becomes submerged in water. The reference resistance is permanent regardless of the amount of water in the tank. Ultimately, a full tank will be represented by a low sensor resistance. To make the sensor compatible with the analog to digital converter, the reference and sensor resistances were incorporated into a voltage divider. The VCC was set to 3.3 V from the microcontroller. One voltage divider consisted of a 470 Ohm resistor in series with the reference resistance; the other voltage divider consisted of a 470 Ohm resistor in series with the sensory resistance. The voltage at the node where the two resistors connected was read into the microcontroller after being converted into a digital signal (WL1 and WL2 respectfully). While the voltage at WL1 never changed, the voltage at WL2 would change according to the sensor resistance; a lower resistance resulted in a higher voltage read into the ADC.

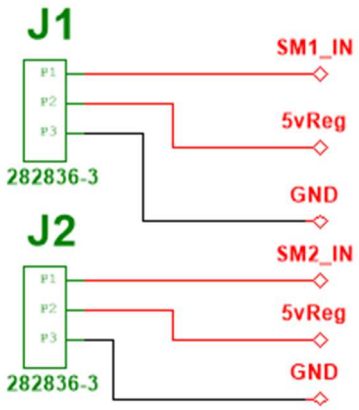


Figure 4: Soil Moisture Sensors Schematic

The two soil moisture sensors [46] both had three wires each. Connectors [44] were used to bridge the external circuitry or each of the three wires to the PCB board. One wire connected to the 5V supply line from the voltage regulator, one connected to ground, and the last remaining wire connected to a pin on the microcontroller set as an input. The capacitive soil moisture sensor outputs a voltage between 0-3 V according to the moisture level of the soil. Dry soil corresponded to a higher voltage compared to wet soil. Similar to the water level sensor, the analog values for both soil moisture sensors were read into the microcontroller after being converted to a digital value via the ADC.

LED Lighting

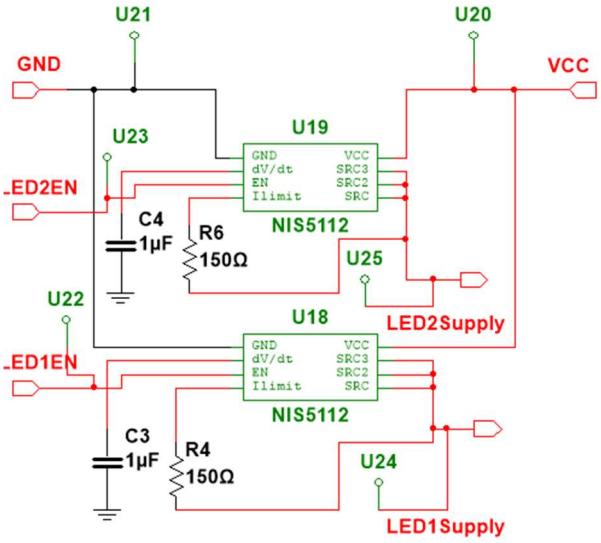


Figure 5: LED Lighting Subsystem Schematic

The LED strips [35] required a 12V supply to its VCC and a ground connection in order to work properly. However, we wanted the microcontroller to dictate when to turn on/off the LEDs, which is where the electronic fuse came in. The electronic fuse [39] connects the 12V

supply line to the source output pins when the fuse is enabled or set to 3.6 V. The source pin leads to a PCB connector where the VCC of the LEDs is attached. Ultimately, the microcontroller can turn on the LEDs by setting an output pin high. The output pin connects to a fuse enable pin; setting an output pin high is analogous to delivering 3.6 V to the pin. The capacitors featured in Figure 6 control the slew rate or the delay from when the fuse is enabled to when the supply voltage appears on the source pins. The capacitors were chosen to be 1 uF in accordance with the datasheet. Another advantage of the electronic fuse is its short protection. The LEDs draw the most current or approximately 0.8 A out of any individual component. If the LEDs start to draw current above the limit set by R4&R6, the fuse will trip. According to the datasheet, a 150 Ohm resistor causes the fuse to trip when the current exceeds 1 A. As such, the limit is above the necessary amount of current draw for the LEDs; however, if the LEDs exceed 1 A, we will likely suspect a short.

It should be noted that we initially started with a very complex schematic. We used a transistor network to turn on tri-colored LEDs. However, for the sake of simplicity and short protection, we swapped out the tri-colored LEDs for a single-color LED strip and the transistor network for the electric fuse.

Water Pumps

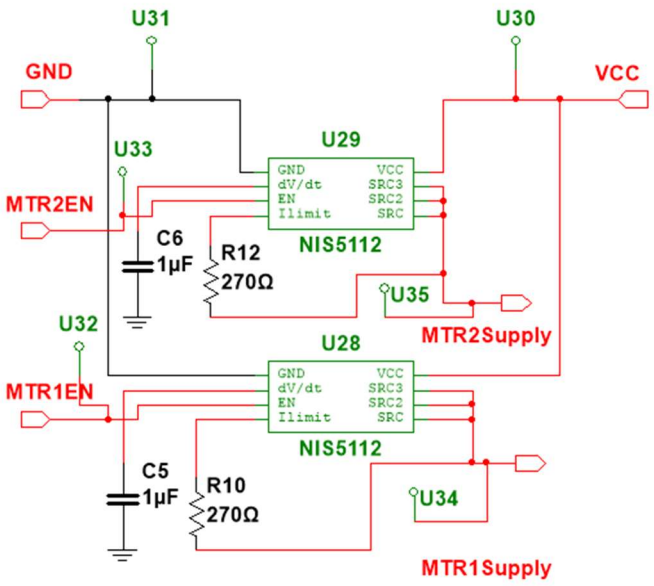


Figure 6: Water Pumps Subsystem Schematic

The same electronic fuse used for the LEDs were also used for the two peristaltic pumps [47]. In order to run, the pumps required a VCC of 12V and a ground connection. Ultimately, the water pump fuses were configured in the exact same way as the LEDs, excluding the values of the resistances R10&R12. The current limit for the fuse was set lower by choosing higher resistances for R10&R12; this is because the peristaltic pumps do not require as much current draw as the LEDs [35]. At 270 Ohms, the current limit in which the fuse will trip is about 0.5 A.

The water pumps only draw 0.3 A each; as such, the fuse's short protection will not interfere with component functionality.

It is important to note that the original circuitry for the water pumps was quite complex. The water pumps were controlled by a motor driver, which required a separate power supply and intensive setup. However, to make debugging easier and include short protection, the motor driver was swapped out for the electronic fuse.

Power Supply

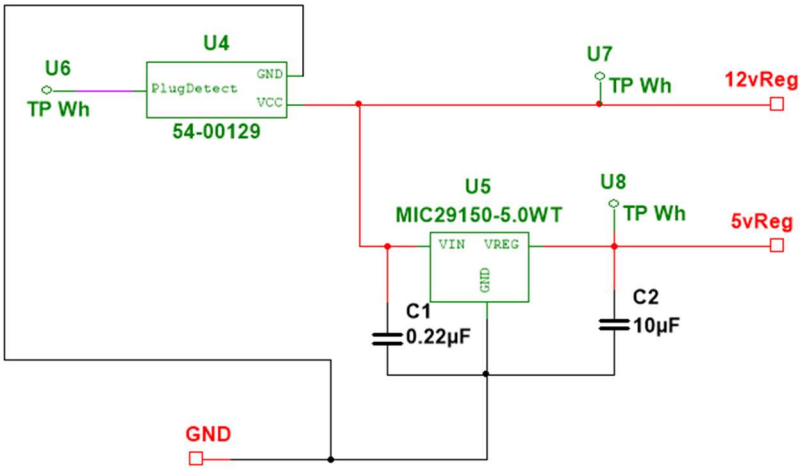


Figure 7: Power Supply Subsystem Schematic

For the residential application of the plant management system, a continuous power supply from the wall was preferential to a battery. In order to convert the three-phased AC power into DC, a wall transformer was used. A 12V supply was needed for the LEDs and water pumps, a 5V supply was needed for the microcontroller and soil moisture sensors, and a 3.3 V supply was needed for the voltage divider network. Ultimately, we choose a 12V regulated wall transformer [40] that outputs 4 amps of current. If all the components were operating at the same time, the total current draw would be less than 4 A. A jack connected [41] was used to connect the wall transformer to the PCB. The 12V supply line was wired directly to the VCC of each electronic fuse. A 5V voltage regulator [43] with a current limit of 1.5 A was used to step down to the 12V supply from the wall transformer to 5V. The voltage regulator circuitry includes two bypass capacitors that short at AC signals to ground; as such, circulating AC signals do not interfere with the DC output. The 5V regulated supply was used to power the microcontroller, WIFI module, and the two soil moisture sensors. Due to the amount of current drawn from these components, it was necessary to attach a heat sink to the voltage regulator. The heat sink [42] captured excess heat that would harm the inner circuitry of the voltage regulator. For thermal calculations for the heat sink attached to the voltage regulator, refer to Appendix A. The 3.3 V supply from the microcontroller was configured as the VCC for the voltage divider networks.

PCB Layout

The PCB layout is seen in Figures 8-9. The main components on the PCB include PCB board connectors (one for each subsystem), the four electronic fuses, the two voltage dividers for the water level sensor, and the header board connectors. The PCB layout was constantly referenced during the testing phase of the project. For example, the first soil moisture sensor, referred to as J1, had its ground connection on the first PCB connector pin; however, the second soil moisture sensor, referred to as J2, had its ground connection on the third PCB connector pin. Matching up the wires from the external sensors to the proper connection on the PCB became a time intensive process. When it came time to test the enable pins with the software, the layout showed which microcontroller pin corresponded to a particular fuse enable pin. Another important aspect of the PCB was the component footprints. To ensure a proper connection between the microcontroller and PCB, the dimensions of the header boards were checked numerous times. The header board footprints were designed according to the datasheet dimensions. In the first PCB layout, the thru-holes were designed for circular pins; however, after getting the component and learning the pins were square, we redesigned the diameter of the thru-holes according to the hypotenuse. We encountered a second issue pertaining to footprints for the four pin PCB connector; ultimately, we used the default footprint in the Multisim database but ordered the wrong component. Since we needed to make modifications to the header board footprints, we decided to amend the PCB connector according to the component we bought. Power traces, which include the 12V supply line, 5V supply line, 3.3V supply line, and ground, were made the thickest. Special care was taken not to circulate the wall transformer ground with traces. The yellow/green traces are on the copper top layer while the red traces are on the copper bottom layer. A ground plane was inserted; thru-holes with a pink cross indicate ground connections.

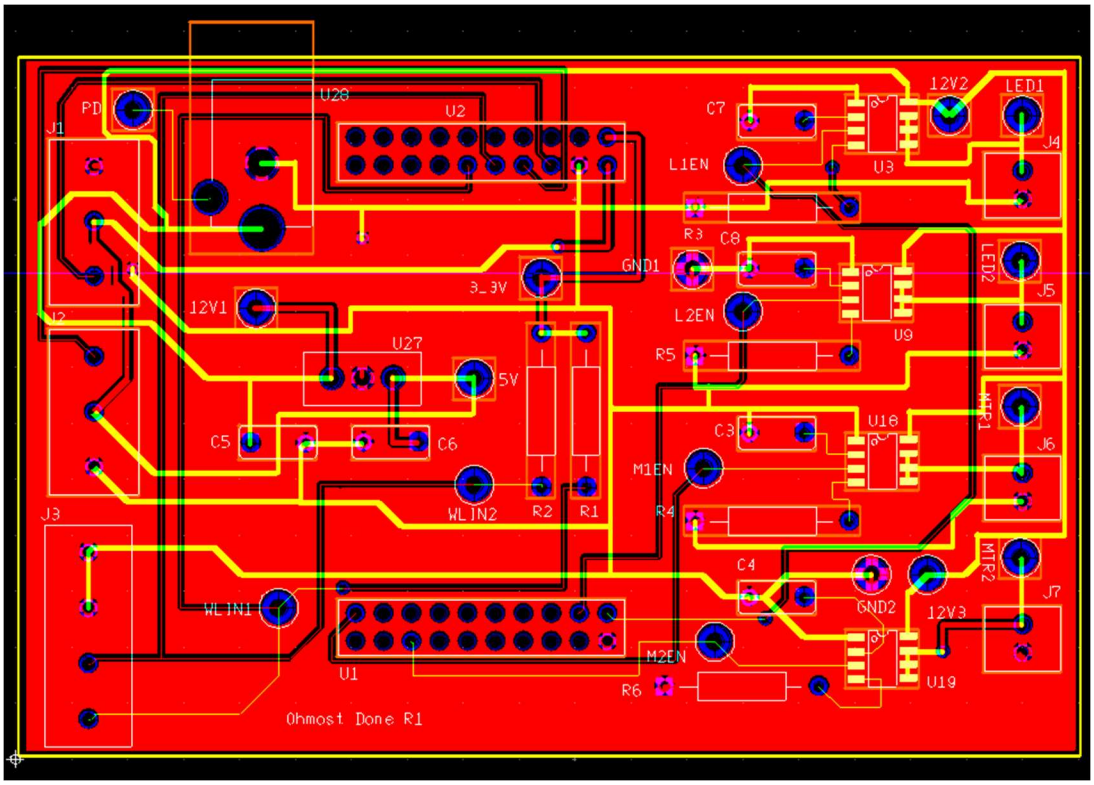


Figure 8:PCB Header Board Layout. Ground Plane Visible

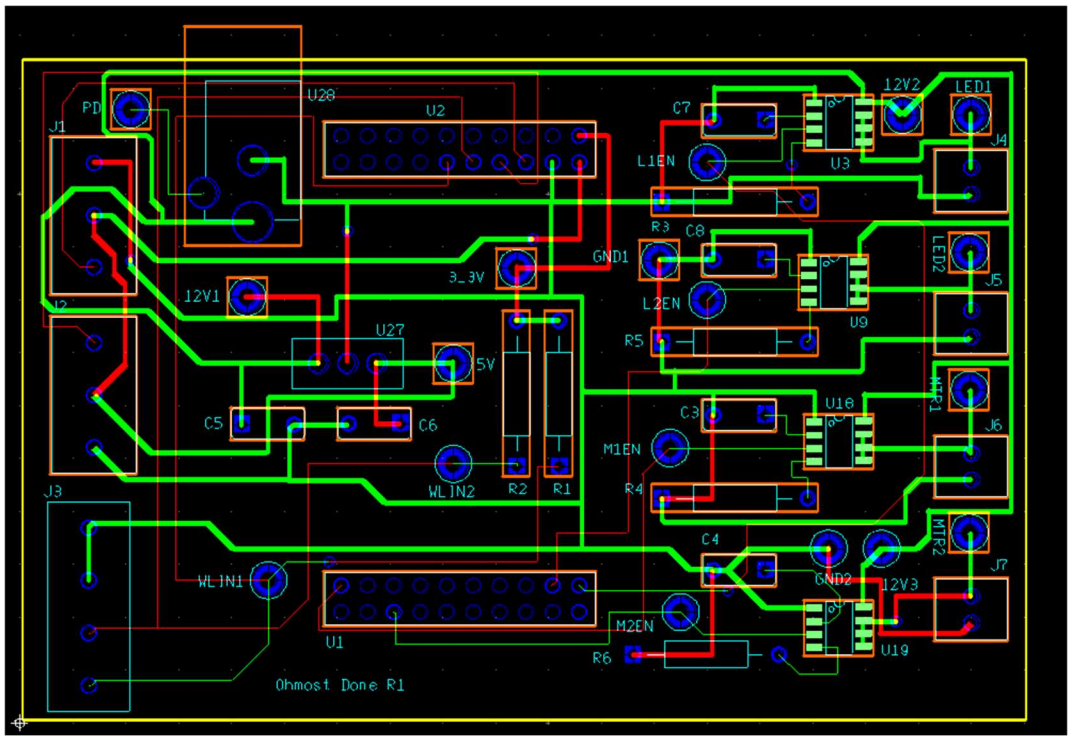


Figure 9: PCB Header Board Layout. Ground Plane Hidden

Importantly, the resistor for each fuse has a ground connection. In the Multisim schematic, the ground connection is replaced with a source pin connection. This modification was made after our second PCB board sendout. While testing the second board, we noticed the resistors becoming extremely hot within seconds of plugging in the wall transformer. Ultimately, we had misinterpreted the datasheet; the ground connections should have been a source pin connection. Although we were able to recorrect this error in the final schematic, we had to solder one end of each resistor to the top of the fuse pin. The main modifications between our first and second PCB layout were adjusting the dimensions of footprints and moving copper bottom traces to not entrap the ground of the jack connector.

Figures 10-11 show the PCB board populated with components. The only solder mask components were the fuses seen to the right of Figure 10. The jack connector is located on the upper left. A heat sink slid over the voltage regulator; however, it is not featured in Figure 10. The header board connectors are best seen in Figure 11.

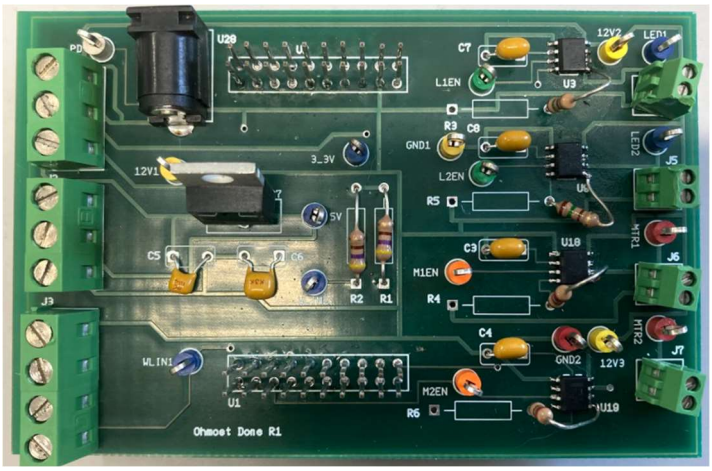


Figure 10: Populated PCB. Front Side

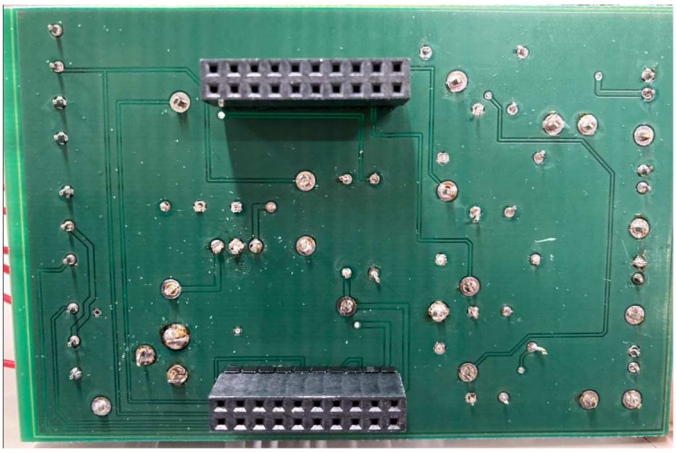


Figure 11: Populated PCB. Back Side

Mechanical Structure

Due to the nature of the project, a mechanical structure was required for housing the plants as well as electrical components of the system. The main structure of the system was designed to resemble a shelf. It was constructed using treated wood and was designed according to the dimensions laid out in Figure 12.

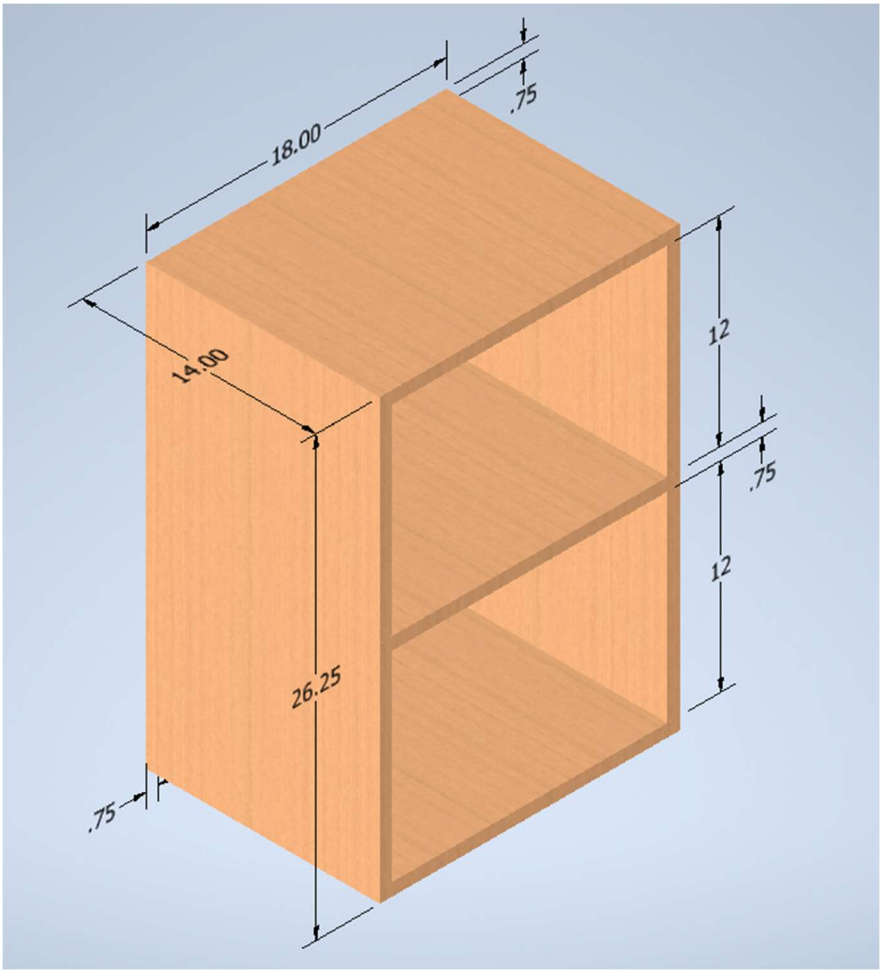


Figure 12: Auxiliary View of Structure

The wood used to build the structure was then painted with a weather resistant stain to limit damage caused by water, soil, and light exposure. The structure was assembled using three stainless steel screws at each intersection. Two holes were drilled on one side of the structure to allow wires to be run from the inner shelves to the PCB. Industrial construction glue was then used to attach the LEDs to the tops of each shelf section in the shape of an inverted "U." Shallow plastic tubs were then attached to each shelf to be used as flood trays. Next, a larger tub was acquired for the water reservoir, and two small holes were drilled into the side of the tub to allow the plastic tubing from the pumps to enter the reservoir. These holes were drilled to be 4mm wide and were waterproofed using underwater caulk. This reservoir was secured to the top of the

structure using the industrial glue. A NEMA enclosure and connectors were also purchased to ensure proper mechanical protection [48]. Two ¾" holes as well as one ½" hole was drilled into the side of the enclosure and the connectors were attached. Wires from the soil moisture sensors and water level sensor were run through the drill holes in the structure to the NEMA connectors and attached to the PCB. The wires from the LEDs and pumps were run to the NEMA connectors on the opposite side of the enclosure and attached to the PCB. The power supply cord was run through the largest NEMA connector and plugged into the power jack on the PCB. All wires and connections were covered with waterproof heat shrink wrap.

Firmware

The code for the firmware is best outlined in separate parts since there were two separate devices that are included in this section (the MSP and the WIFI Module) that have vastly different operations, but are within the same code base. These parts will be called the "Controller" for the Microcontroller and its main tasks, the "WIFI Module", and "Common" for any setup and configuration that applies to them both.

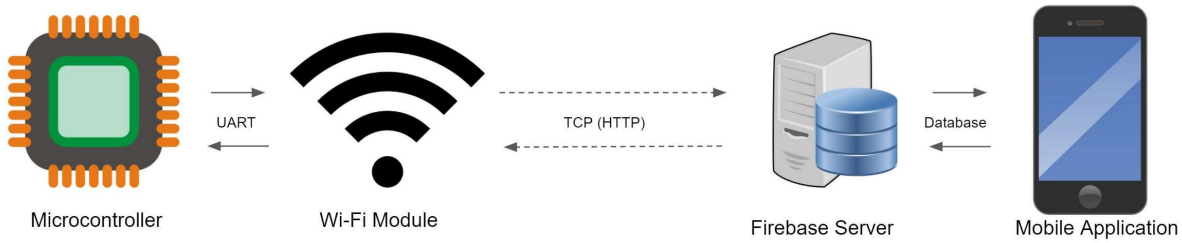


Figure 13: High Level overview of the Network Communication System

The microcontroller used for the project was an MSP-EXP430F5529LP [11] due to its compatibility with the CC3100BOOST [12] WIFI Module. The two components are shown on the left as the Firmware section of the Network Communication System. These two were chosen together since combined they would be able to do everything that the team needed. The alternative choice was the MSP-EXP430FR5969 49]. At the time of purchasing the board this version of the MSP430 did not have adequate capabilities for what our project required, so the F5529 was chosen instead. The MSP430F5529 and the CC3100BOOST combination of boards also came with a vast library of example projects. Two of the example projects were used as a base (specifically the "getting-started-with-wlan-station" [50] and "tcp-socket" [51] projects) which were modified to fit into the wider goals of the project.

1. Common

System Clock and Timers

The main clock for the system runs at the maximum possible for the microcontroller at 25MHz. This main clock is then used to create two slower timers. One is used for the UART communication for terminal debugging, and the other is used as a timer for the automated systems. The UART clock is set to accommodate for a Baud rate of either 9600 or 115200. The main clock is divided by the baud rate to supply the

compare capture registers with the proper values to count to. The other clock is set to 1Hz which is used as a 1 second timer incrementing a global variable that is used in other sections of the project. The comparison value here is 32700 with the input as the ACLK. This second clock is used for the main functions including all timers on the automated watering system. The timer utilizes the compare capture register ISR to increment a counter every second. After each 60 second interval the ISR will increment a minute counter, and after 60 minutes, will increment an hour counter. Each of these counters has its own "get" method to retrieve the value from another section of the project, but is only used in main. Since the project, under the normal use case, will operate on the multiple hour scale, being able to measure larger scale time increments without having to worry about overflow was important. The alternative was to have each operation run on its own timer, but this option was chosen instead due to its reliability, simplicity, and consistency. All important sections of code related to the initialization and utilization of clocks can be seen in Appendix E.

UART & SPI

The communication protocols between the microcontroller and the WIFI Module as well as the communication between the boards and the debugging terminal, were all library functions that were included in the base code for the wlan station. Nothing of this code was altered and the implementation in the project was done according to the standard methods for both protocols. While initially considered a larger part of the project these two aspects only came into play when designing the PCB by ensuring that no pins associated with either function were used by other aspects of the project. Instead of designing our own versions of these methods, the library functions were used to ease the process of testing and alleviate concern that having mismatched protocols caused.

2. Controller

ADC

All four sensor inputs to the controller are analog signals that require the use of an ADC. These four inputs are two soil moisture sensors [46] and a water level sensor [18] with two outputs. The MSP430 used for the project has an ADC12 that was set up to do these conversions. The settings used were extended sampling timer, and repeated sequence. Since the system will be constantly running and monitoring the system the ADCs should be running periodically the entire time the system is operational. Each of the four channels used it added to the mux that the ADC will switch between as it samples. Each sample is added directly to the system memory where it will be retrieved later. After it has finished sampling all four of the channels, the ISR for the ADC is triggered. This will move the data from memory into a circular buffer, one for each of the four inputs. The ADC will continue this process until power is disabled or a system fault occurs. When the main program wants to access the data from the buffers it will call an average function that will average the data currently in the buffer and then return it. The goal here is to eliminate the erroneous data that may come from issues with the sensors.

Since the water level sensor has both a reference and a variable measurement, it uses another function to get the actual value of the sensor. This function returns an error code if the variable value is greater than the reference, or returns the level of the water as a percentage of "full". Full in this case means that the reservoir is filled to a capacity that was deemed as the maximum that the container could safely hold without worrying about spilling over the top. Code snippets for this portion of the project can be seen in Appendix F.

GPIO

All of the outputs of the system are basic GPIO signals. The two pumps and the two LED strips are all active high. The associated pins are enabled in the firmware and then set to 0 initially. Each of the LEDs are associated with an enumerated type that is used in the relevant functions. The functions for the pumps follow the same structure but use a separate enumerated type. Each type of output has an init, turn on, turn off, toggle, and status function. The only difference between the two sets of functions is the LED's additional enable and disable functions. These were created specifically to satisfy one of the main design constraints that were originally outlined. While it ended up not being as large of an issue in the final design, originally there was a need to make sure that the LED and Pump for a single plant were never active at the same time. The idea was to turn the LED off when the pump needed to run in order to water the plant. This task was accomplished with the enable and disable functions. The standard turn on and off functions will not maintain state. So, while the LED could be turned off, there would be no way to know if it needed to get turned back on. The enable and disable functions solve this problem. When the disable function is called, the function will return the current state of the disabled LED. Then when the enable is called, that value is then passed in and will reset the LED to the state it was before being disabled. The function works by using an XOR property where the value is XOR'd by the same value twice. Check Appendix G for the specific implementation of these functions.

3. WIFI Module

TCP Client Mode

The basis of the TCP communication was created from the two TI example projects mentioned "getting-started-with-wlan-station" [50] and "tcp-socket" [51]. The combination of these two projects completed all of the essential functionality for connecting to a wireless network and to a remote server. In order to make these two projects work in conjunction with one another the only things that needed to be changed were the network credentials, the main function used, destination IP address, and the port number used for communication. When first being tested the goal was only to make sure that the module could ping and handshake with the destination server. This was all handled by the main file code supplied by the "tcp-socket" project. It sets up the CC3100 to its default state, starts up the device, connects to an access point to acquire an IP address, and creates a connection to a TCP server by handshaking with it. The main issue

that was discovered from this process was that the file was assuming that the data was going two ways, sending and receiving. While this idea was initially considered, the concern was that while the WIFI module was waiting to receive data back from the server, if the data never came the system would hang and the automatic system would not be able to function. This was the main reason the communication between the module and the server was designed to be one way. This led to the two-way communication feature being removed from the system entirely in favor of only one way, from module to server. Even without this feature the main file was able to connect to a wireless network and ping a server. The network used for the project was associated with CJ's computer so that the network traffic to and from the device could be monitored and so that the credentials would not have to be updated every time the system moved locations during testing. The "tcp-socket" code was transferred into a separate header and C file so that they could be called in the main loop being designed for the project. This led to the alteration of a few functions. The most important change was to the original main function. It was renamed to a connectToServer function that would run the original code, but instead of looping forever when an issue occurred, it would return an error status of -1. The connectToServer function also took two parameters which are important for the HTTP protocol for the data and to what field the data belonged.

HTTP Protocol

Due to the nature of the Firebase server that the team is using, the information being sent to the server needs to be in an HTTP packet format. The code from the example projects is only able to create a connection with TCP so all the formatting for a HTTP packet is done dynamically by a set of functions created for this purpose. Since the connection to the server is only one-way, all the data sent to the server is done so in a POST request, part of the REST API. Using this API is the only way that the server will accept the data that comes to it from the WIFI module. By utilizing Postman, the team determined that the fields required for sending the data were type (POST), destination, version of the HTTP protocol (1.1), host name, content-type (JSON), content length, user agent (MSP430), and the data. The formatting for the POST request can be seen in Appendix D. The data itself is going to be in a JSON format. This is explained more in the *Software* section since interpreting the data for the application will occur on the server end of the system. Due to the complexity of the HTTP data formatting, to ensure that no issues occur during runtime, the majority of the formatting is handled statically. A macro is declared that is the template for the data when it is going to be sent. Then the data is dynamically inserted into the template whenever the main program needs to send data to the server.

Transmitting Data

To actually ensure that the data is properly formatted when it is sent to the server, a function called *createString* is used. This function will take as parameters the data and a tag to identify the data. The maximum for the data is four digits and the maximum for the tag is three characters. Since the data from the ADC can only range between 0 and 4096,

this constraint will not be an issue. The 4-digit limit was also created because of this factor. Each Input analog signal has a predetermined tag. SM(M/B) for the soil moisture sensors, and WL(N/D) for the water level sensor. These specific tags will be explained with the main loop code. Each time that the main loop needs to send data, the *connectToServer* function is called with the data and tag. Once the module has been connected to the access point successfully, it will attempt to initiate the TCP handshake with the remote server. Before the handshake occurs, the *createString* function will be called to format the date to be sent if the handshake is successful. It does this by using *sprintf* to fill a temporary buffer with the integer data, then replaces the specific characters in the template string with the data and tag before returning the buffer to the function that called it. See the implementation in Appendix D. This buffer of data is what is sent to the server before the connection is terminated and the watering system moves forward as normal.

Main Autonomous System Loop

The main loop of the program starts off by initializing the system clocks, disabling the watchdog timer, initializing the ADC and GPIOs, configuring the UART and SPI features, and setting values for the system presets. The system presets include the max time lights are one, max time lights are off, and the interval for measuring the soil moisture. Each of the two plants has one set of max lights measures as well as a variable that will hold the last time that they were changed, and one that will hold the current state of that LED. Once the system is initialized, the system will enter an infinite loop. This loop monitors the system and will act accordingly when the timers indicate that a certain action should be performed.

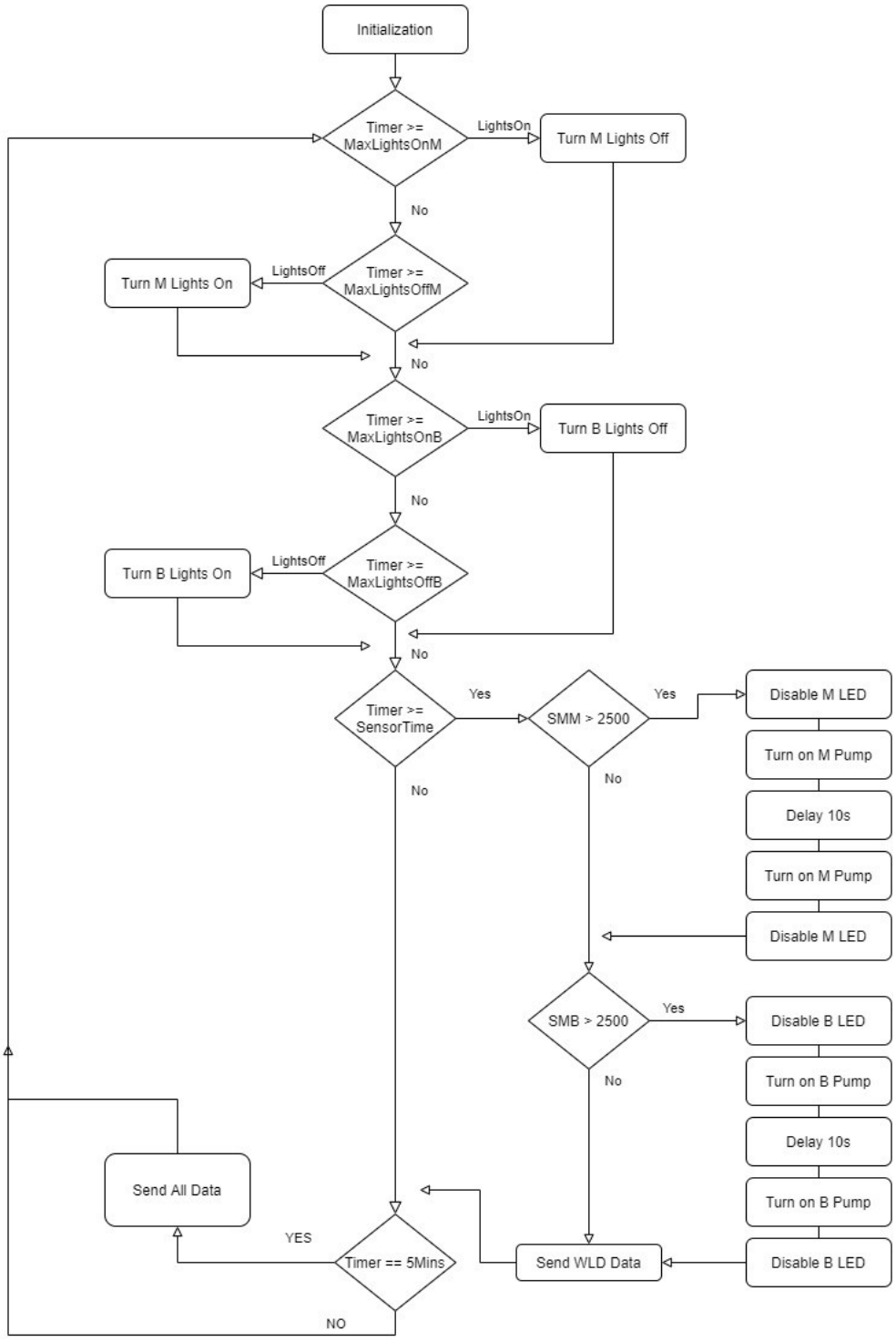


Figure 14: Main Project Loop for Auto Watering System

The loop starts by checking the current system time and setting it to a variable that will be used throughout the rest of this iteration of the loop. It will first check if the first plant (Mint in the case of the demo) needs to have its lights turned on or off based on how long it has been in its current state. If it has been in that state for the time outlined by the max time on or off, then it

will toggle the lights and mirror that change on the state variable. The system will then check for the same cases on the other plant (Basil for our demo). Once it has checked if the lights need to be toggled it will then check to see if the plants need to be watered. If the average of soil moisture for either of the plants is too low it will water that plant. It does this by disabling the associated LED, turning on the pump, waiting for 10 seconds, turning the pump off, enabling the LED and then indicating that a pump has run by adding 1 to a counter of pumps run. If neither pump runs then this value will be 0, if one runs the value will be 1, and if two run the value will be 2. If the value is greater than one after both sensors have been checked, the system will then get a measurement of the water level and send a message to the server with the tag WLD to indicate that the value occurred after water was dispensed from the reservoir. After this process completes, or if it doesn't happen in this iteration of the loop, the system then checks to see if it should send data to the server. If it needs to send data it will then send three messages. The first will be the data with the tag "WLN" for water level normal. The second is the SMM data for the mint's soil moisture. Finally, it will send data with the tag "SMB" for basil. At this point the loop will then repeat forever.

Software

Mobile Application

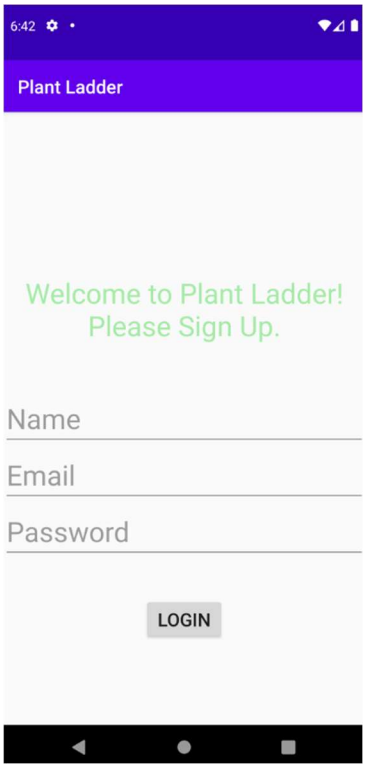


Figure 15: Login Screen

The mobile application was used to display the user interface (UI), for android users only. Once the application is opened, a screen for login is displayed as shown in Figure 15. This is not Google login, but this would have been better for security purposes. This was not implemented here because this was a prototype and security were not a large consideration for this version of

the product. The next screen, after login, shows two buttons: Basil and Mint. These buttons, when clicked, will display the information sent from the microcontroller. See Figure 16 for reference. This could have been user mutable, but this design choice was because the networking involved in communicating from the mobile application to the microcontroller is complex and with time constraints, not feasible.

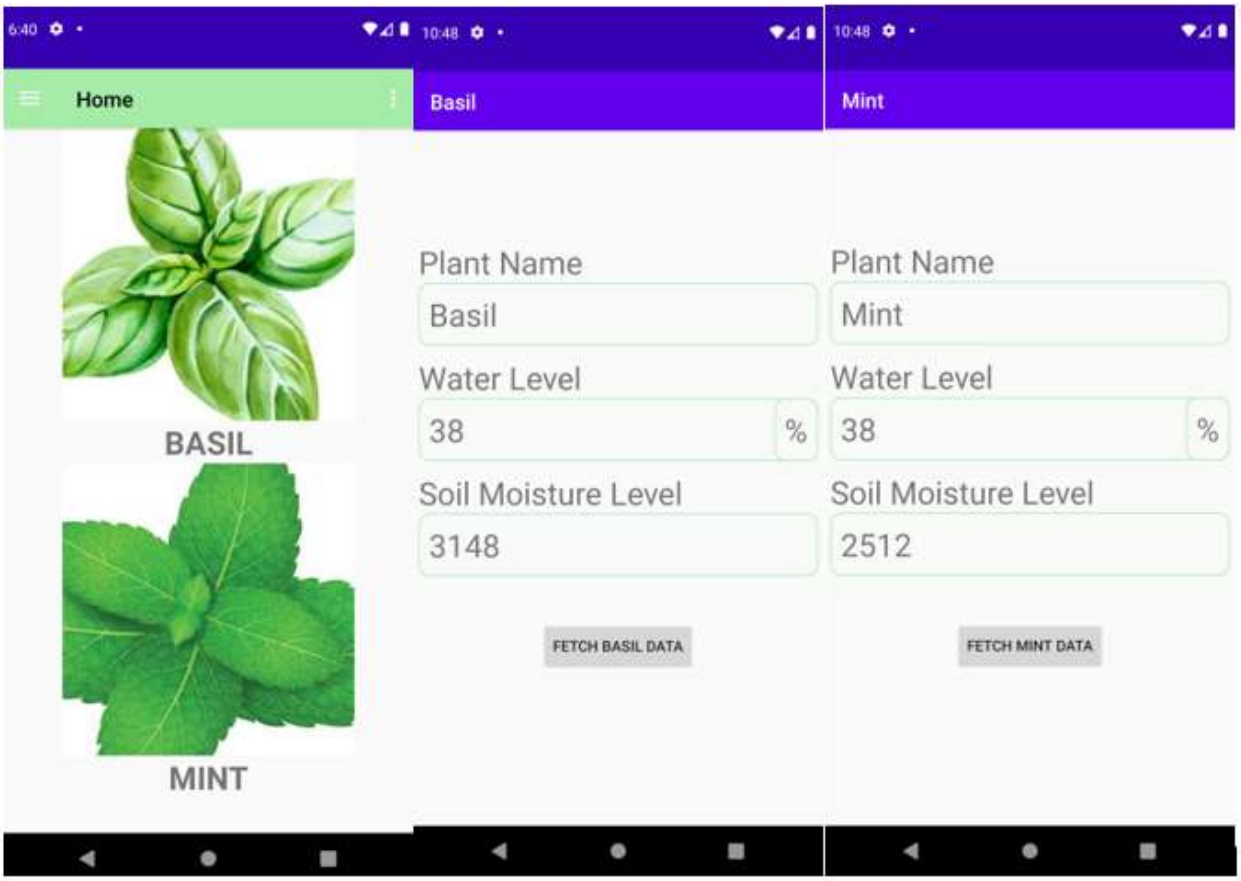


Figure 16: Basil and Mint Options Screen

This application retrieves this information as shown in Figure 17. The WIFI module sends data regarding the soil moisture sensors and water level sensor as explained in the Firmware Section. In order for this data to be parsed correctly, it is necessary to create a Firebase Function that will create GET and POST requests, in HTTP formatting. The code to do this is done in JSON using Visual Studio Code as is shown in Appendix 55. This new Firebase function will be deployed so that the Microcontroller's WIFI module can communicate with the database, Cloud Firestore.

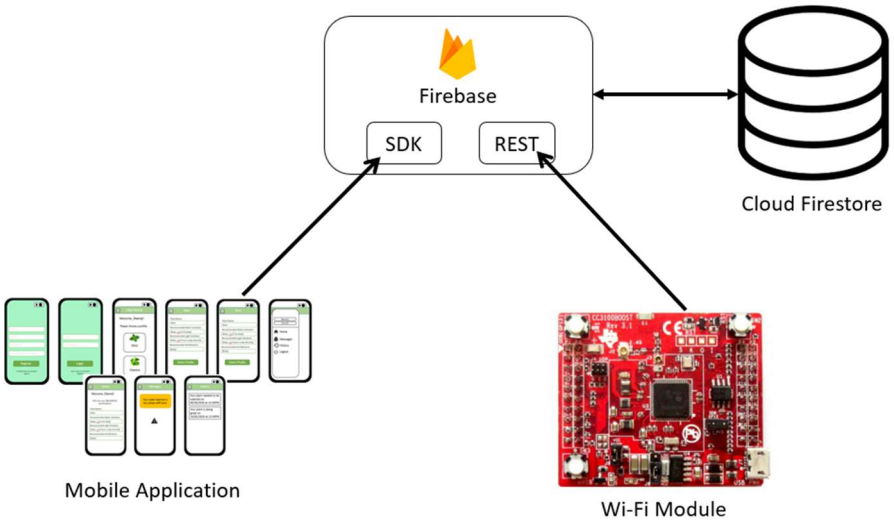


Figure 17: WIFI Module and Mobile Application communication to Firebase BaaS

The mobile application will then retrieve data from the Cloud Firestore and convert the entries to integers so that they can be used for comparison. This retrieval process is shown in a code snippet in Appendix 56.

Project Timeline

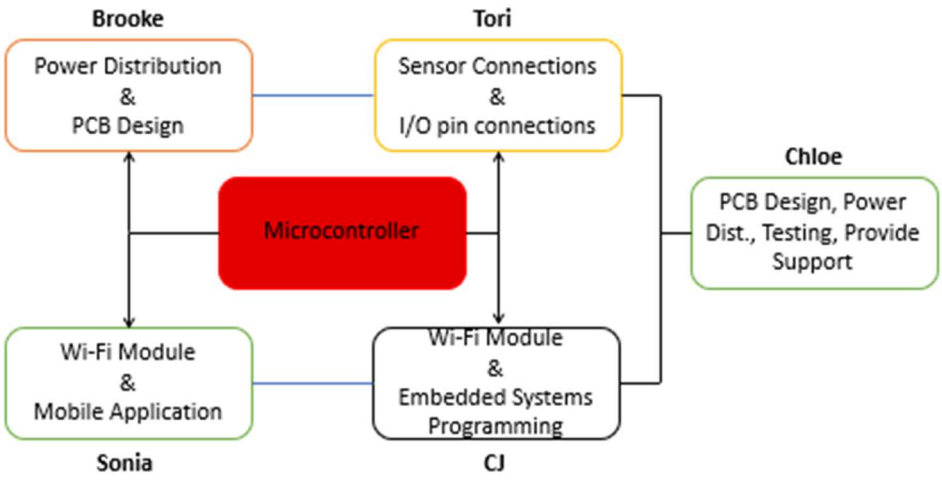


Figure 18: Parallel and Serial Tasks

The responsibilities of each team member are outlined as follows. Sonia worked on the mobile application development as well as enabling the WIFI connection between the microcontroller and the application. CJ was responsible for the embedded systems programming including connecting the sensor data to the microcontroller. Tori assisted with the sensor-microcontroller connection since this is an extensive task as well as ensure correct connection to I/O pins. Brooke built the mechanical structure as well as assisted in power distribution and PCB

design. Chloe assisted with PCB design, power distribution, testing the equipment, and providing overall support in areas needed.

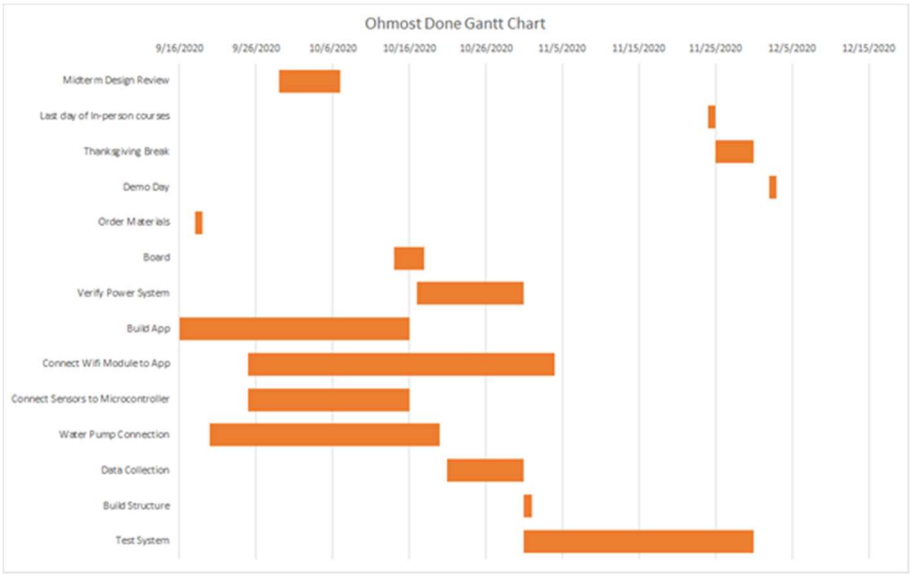


Figure 19: Original Gantt chart

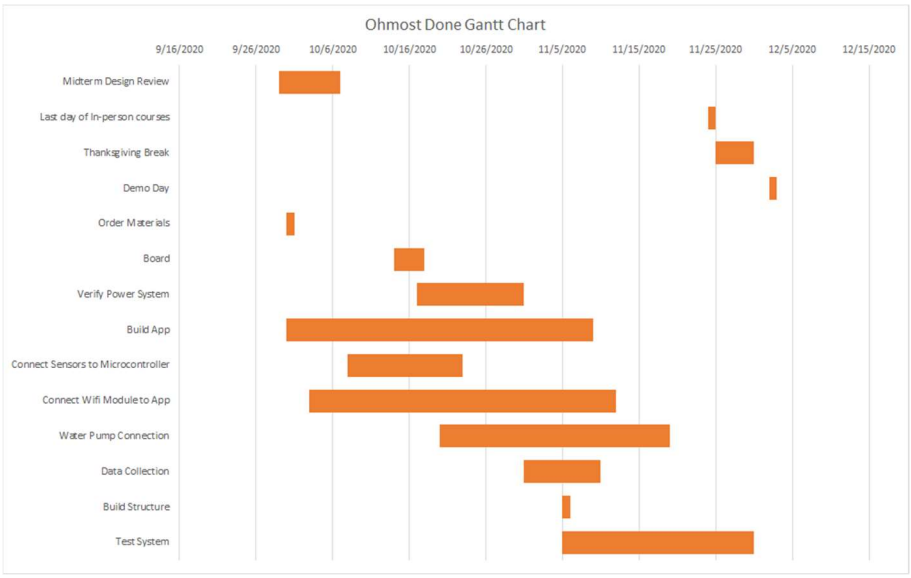


Figure 20: Midterm Review Gantt chart

Excel was used to create a Gantt Chart to map out the timeline of the project. The initial timeline was modified during the Midterm Design Review. Both timelines can be seen in Figures 19 and 20 with key dates at the top so the team would remain cognizant of the Midterm Design Review, the last of in person classes, Thanksgiving Break, and Demo Day. Each large category of tasks, such as the PCB, Network protocols, Software UI, and Mechanical structure, were parallelized, while each subtask within these categories was serial, relying on successful past steps to continue. In particular, it was important to frontload ordering parts and designing the

PCBs, since these had the longest turnaround times and the rest of the system relied on these tasks being completed. Secondly, having the software UI and firmware done while this was being processed ensured all team members had an active role at all times. At the Midterm Design Review, the team realized the dates for sendouts and reviews were later than anticipated, so the updated Gantt chart reflects this change. Finally, system integration and testing were predicted to be time consuming, so all task deadlines were pushed up, with the team aiming to have a fully working system and complete demo by Thanksgiving break.

Test Plan

Figures 21-24 show our group's Hardware test plan.

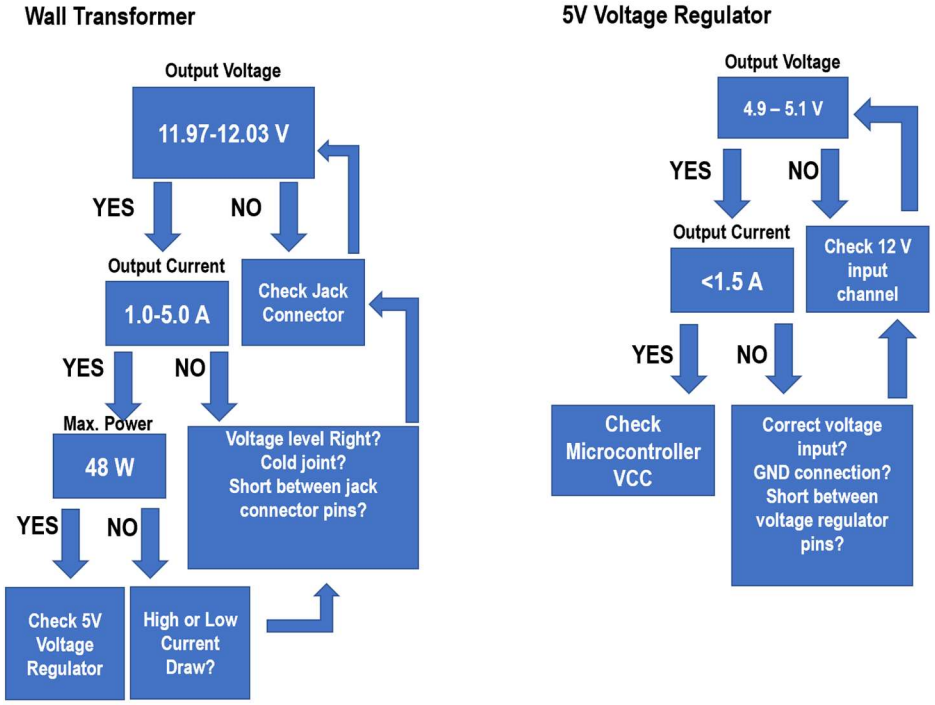


Figure 21: Power Supply Testing Procedure

The performance of the system was entirely contingent upon the power supply working. As such, the power supply was the first subsystem tested. There were two main power systems: the regulated 12V from the wall transformer [40] and the 5V voltage regulator [43]. Since the voltage regulator required 12V input from the wall transformer, the wall transformer system was checked first. The wall transformer supplied a voltage within the correct range as specified by Figure 21. Configuring the ammeter to check the current draw was extremely difficult; as such, after talking with our technical advisor, our group decided to reference the datasheet for the amount of current being drawn. After verifying the wall transformer, the voltage regulator was tested. While 5V appeared at the regulator's output, it was again very difficult to configure the ammeter. Furthermore, after recalculating each subsystem's current draw, it was necessary to swap out the device for one with a higher current limit. Finally, the current draw from the regulator gave rise to concerns about the device's temperature. A heat sink [42] was ordered and attached to the regulator to prevent overheating the device. The measured input and output

voltages were in the desired range and the regulator did not overheat. Ultimately, we concluded the power supply was working properly.

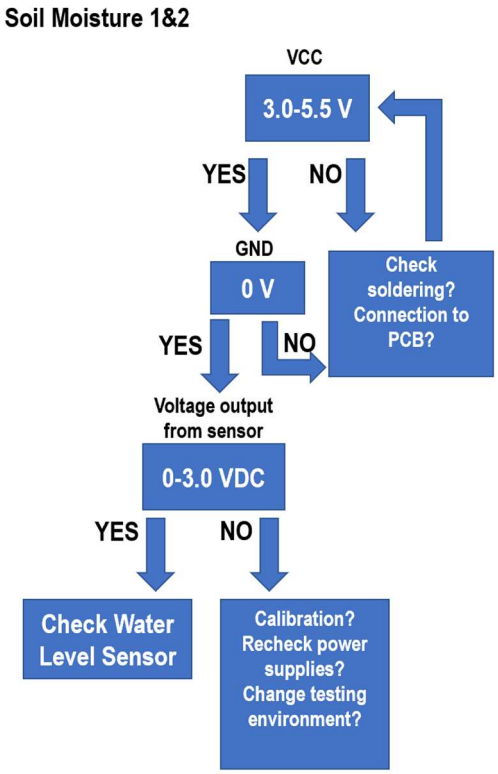


Figure 22: Soil Moisture Testing Procedure

The soil moisture sensors [46] required 3-5V supplied to VCC as well as a ground connection to function properly. The multimeter showed a VCC voltage of 5V and GND voltage of 0V. After checking the power supply, the sensor performance was assessed. Ultimately, the sensor outputs an analog voltage between 0-3 V. High voltages corresponded to dry soil, and low voltages corresponded to wet soil. The analog signal from the sensor did not noticeably change unless the soil was either very wet or very dry. The logic for the system was altered to account for this abnormality. However, since the sensor produced analog voltages in the specified range and distinguished between wet/dry soil conditions, our group concluded that the soil moisture sensors were working.

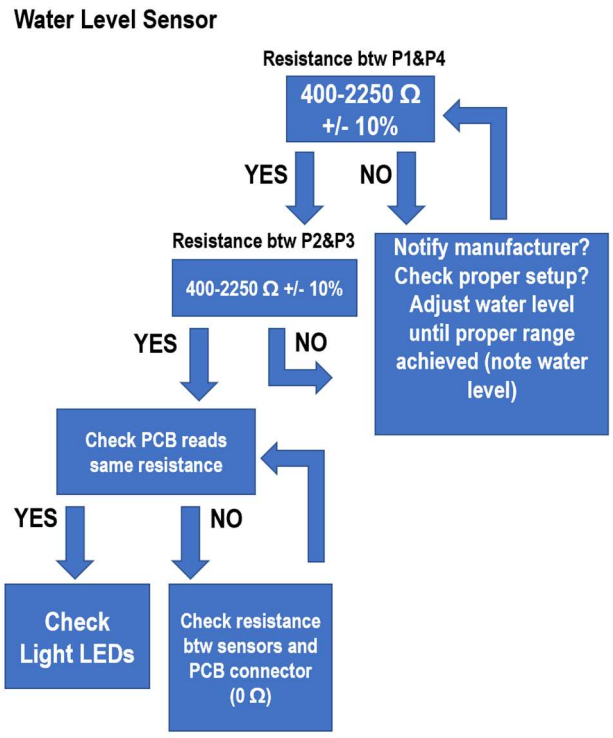


Figure 23: Water Level Sensor Testing Procedure

Since the water level sensor [18] acts similar to a potentiometer, it does not require power. In order to test the water level sensor, we oriented it vertically and flushed the back of the sensor to the water reservoir. We measured the resistance across pins 1&4, which was the reference resistance, and across pins 2&3, which was the sensor resistance, when the tank was empty. We added water to the reservoir and saw the sensor resistance decrease while the reference resistance stayed the same. Ultimately, the water level sensor worked properly.

The last system components to be tested were the two LEDs [35] and two peristaltic pumps [47]. An output pin from the microcontroller led to the enable pin of an electronic fuse [39]. Once enabled, the fuse supplied 12 V to the VCC of each component. The test plan for the four components consisted of checking the functionality of the fuse.

Figure 24 shows the test plan for the LEDs/Pumps.

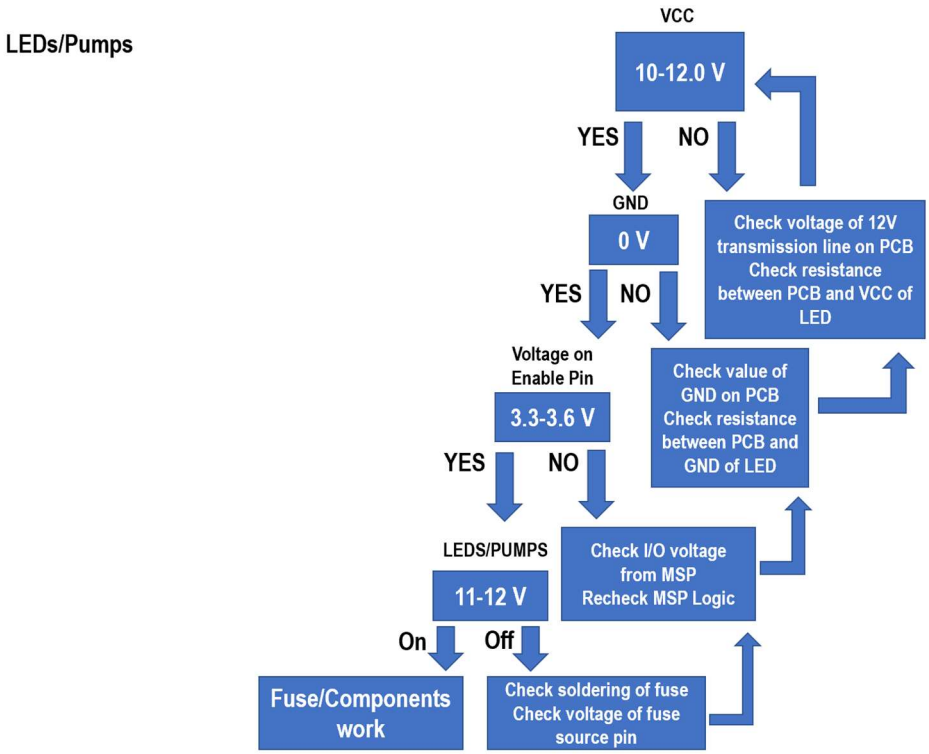


Figure 24: LEDs/Pumps Testing Procedure

The first step was to ensure the proper voltage was being sent to the fuse. The supply voltage came from the wall transformer; however, seconds after plugging the transformer in, resistors connected to fuses started to overheat. Ultimately, we had misinterpreted the datasheet and connected each resistor to ground rather than to the source pins on the fuse. After breaking the ground connection and resoldering one end of the resistor to a source pin, the problem went away. We rechecked the supply voltage, and it read 12V for all four fuses. Next, we set an output pin from the microcontroller high, which sent 3.6 V to an enable pin for the respective fuse. If the LED/pump did not turn on after 3.6 V appeared on the fuse's enable pin, the soldering of the resistor to the source pin was rechecked. An incomplete connection between the resistor and source pin as well as floating grounds were our two most common causes of error. First, we made sure both of the pumps worked and then checked the LEDs.

There were no major changes in our design from our initial testing. In our first PCB, we had messed up a couple footprints for thru-hole components, which delayed testing. Our second and final PCB was almost identical to the first design; however, the footprints were fixed in the second version. If we had a third PCB send out, we would have changed the ground connection of resistors to the source pins.

Figures 25 - 30 show our group's Software Test Plan:

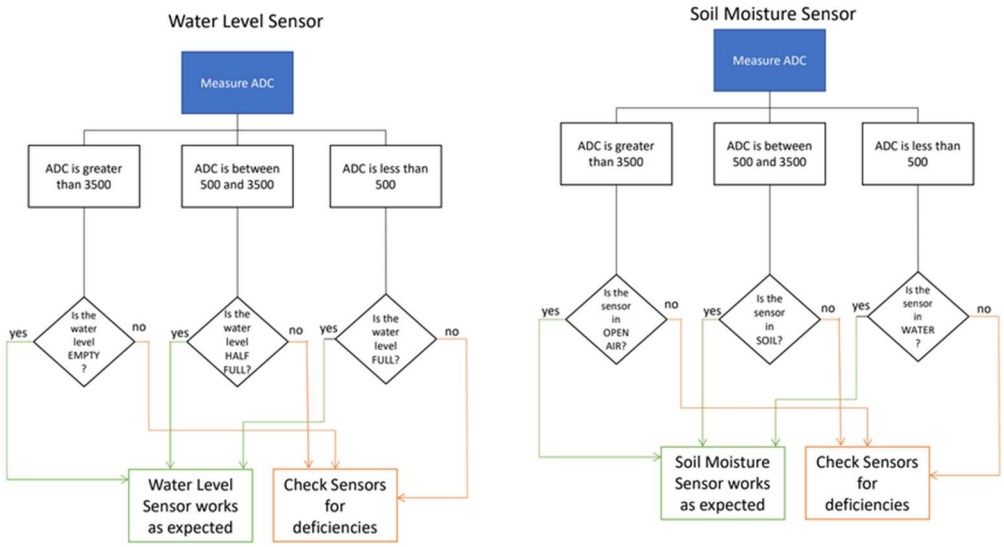


Figure 25: Software Test Plan - Water Level and Soil Moisture Sensors

Since the sensors for the water level and soil moisture worked similarly from a code perspective, as such the output from the sensors was the only information we needed to make sure was correct. This step was mostly designed for confirmation and to help determine what thresholds would need to be used in the final version of the firmware. The numbers for the Soil Moisture Sensors ended up being a tighter range then we were expecting, ranging between about 3200 and 1500. The lower range only means that the system would not be as sensitive as originally anticipated, but not a barrier to completion of the project. The Water Level sensor on the other hand had more inconsistencies. When testing the levels it would take a long time to settle and would often jump up or down by 20%. Due to budget constraints and time, the sensor was not changed out since its inconsistent inaccuracies did not negatively affect the system as a whole.

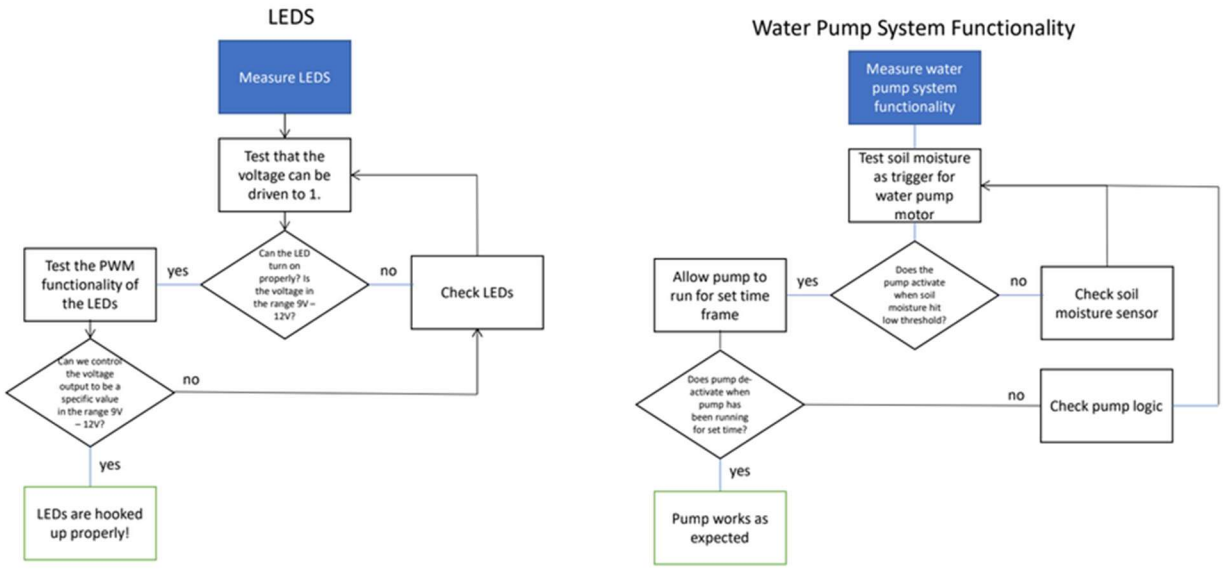


Figure 26: Software Test Plan - LEDs and Water Pump System

The first part of testing any of the firmware related to the GPIO pins was to make sure that they were working properly in the debugger. Once the GPIO pins were initialized and tested as outputs the LEDs were connected. Since they required a higher voltage power source than the MSP could give the board was used as a middle man. In the original testing, one LED worked without issue and the other ran into some issues. In order to solve the issue, the flowchart in Figure 26 was followed. Since the LED did not turn on, the voltage at the enable pin from the MSP was tested and was showing the correct voltage. The team then determined that the connection between the LED and the board was not strong and was able to adjust it so that the LED would turn on. While testing the pumps, the enable was working as expected with each pump turning on at the correct time, but later it was discovered that they were going to be pumping water in the wrong direction so the pins had to be flipped. The time in which the pumps were set to run also took a few adjustments until the team determined the best time was 10 seconds. Testing both of these in junction also meant making sure that only the pump or the LED was on, and not both at the same time.

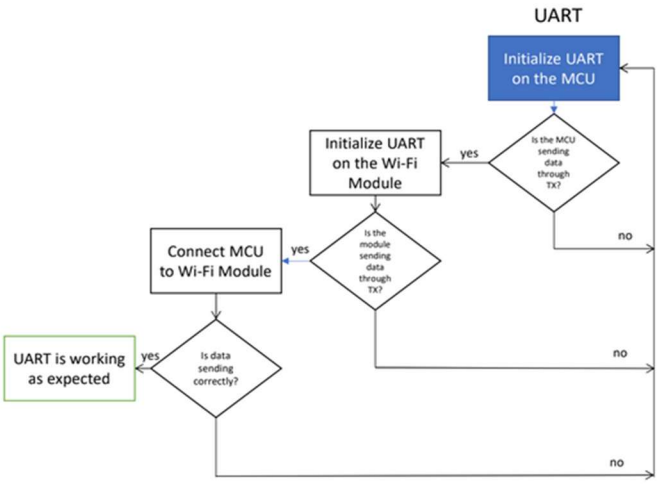


Figure 27: Software Test Plan - UART

Due to the way that the WIFI module code was completed, the communication between the MSP and the WIFI module was largely handled by the library code given by the CC3100. The two devices were able to seamlessly interact with each other with little change needed from the team.

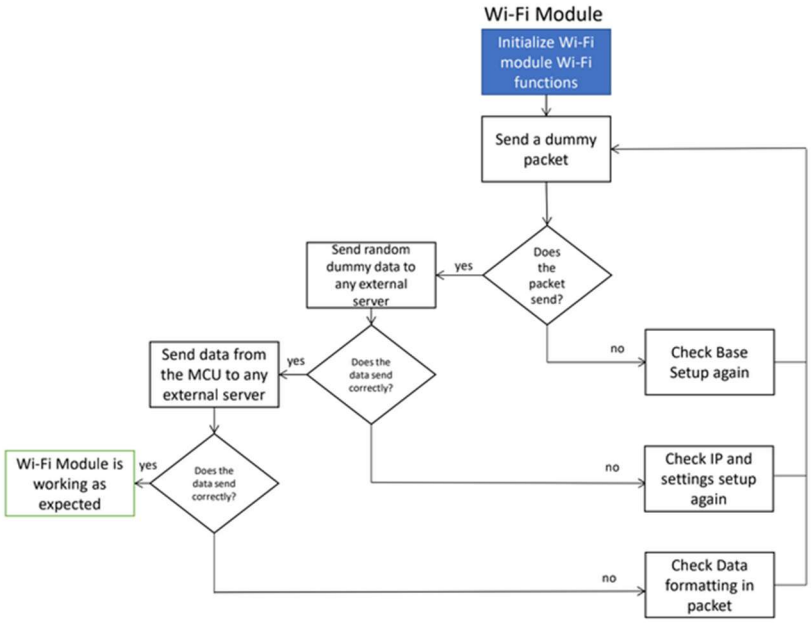


Figure 28: Software Test Plan - WIFI Module

The first steps that were taken in order to test the WIFI module were to first check if the device could connect to the internet. Once the device was able to connect, the group then tested if it could connect to a server. This was first tested by sending a default message to see if the TCP handshake was occurring. The debugging system built into the code was able to show that the handshake was attempting to occur but failing. After determining the correct server IP address and port number the handshake was able to occur but the data sent was malformed for the server to register the connection. With the handshake properly occurring the HTTP protocols needed to be configured for the server and the WIFI module to properly communicate. Postman was used to determine the formatting and fields needed for the data. With this information, dummy data was able to be sent to the server. From there the functions were altered so that the data sent to the server could be set dynamically. Once the server was able to properly receive this data, the WIFI module code was fully tested.

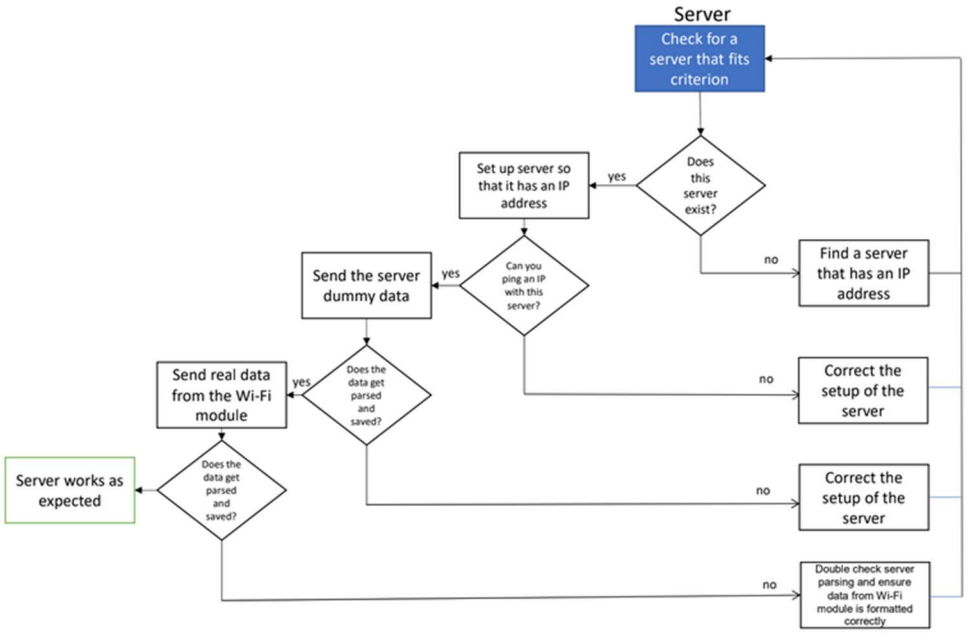


Figure 29: Software Test Plan - Server

The server needed to be tested incrementally, to make sure each step of the server setup was done correctly. Firebase was the server that was used, as previously mentioned, so upon deployment, the server inherently had an IP address. Then, the task of sending dummy data from the microcontroller was necessary and tested, and finally, real data from the microcontroller to the server was tested.

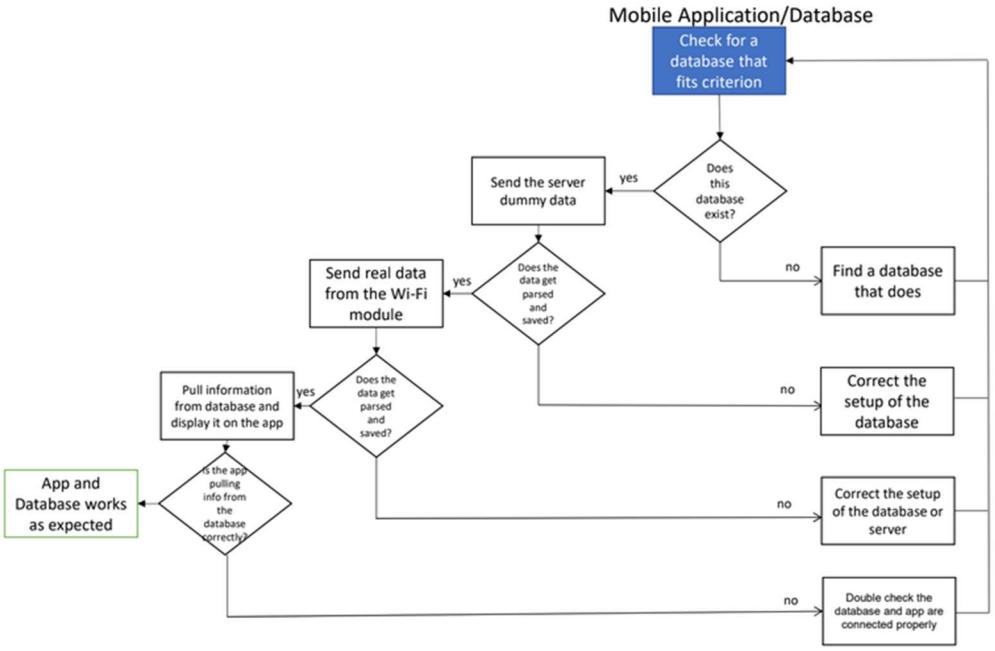


Figure 30: Software Test Plan - Mobile Application and Database

The database used was within Firebase, specifically Cloud Firestore. The database was then sent dummy data from both Postman [53] - a service to test HTTP Requests (GET and POST) and then from the microcontroller. Once dummy data was confirmed on both steads, real data was sent from the microcontroller. It was easy to pinpoint if the error was from the microcontroller or the database based on whether or not dummy data was sent correctly. Then, the mobile application needed to display the data so the app pulled from the database each time anything was stored - both dummy and real data.

Final Results

Figure 31: Completed System

The final system constructed is depicted in Figure 31. It consists of two wooden tiers, a water reservoir placed on top, and a NEMA enclosure attached to the side. Each tier has its individual soil moisture sensor, water pump, LED lighting, and flood tray. A water level sensor is attached on the inside of the water reservoir. Mint is placed on the top tier while basil is placed on the bottom tier.

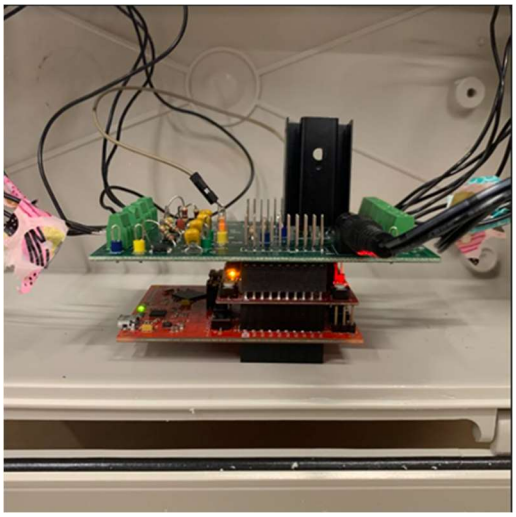


Figure 32: NEMA Enclosure Structure

The electrical control center hidden by the NEMA enclosure is shown in Figure 32. All external components are routed through cord grips secured in holes that are drilled into the

enclosure. Colored tape was used to distinguish the meaning of the all-black wires from each other. The configuration of the stacked integrated circuit boards from bottom to top is, microcontroller → WIFI module → PCB header board. The original plan was to have the WIFI module be situated on the top of the stack but the addition of a heat sink required us to switch the order. The functionality of the system remained the same even with this change.

Demo Video Details

The demo video aims to demonstrate all the features of the system in a short time span. A longer time lapse video was not possible due to the COVID-19 restrictions placed on the laboratory where the project was being worked on. Therefore, instead of determining the functionality of the system based on plant profiles for the mint and basil plant types, the functionality of the system was instead based on if the following features were met:

1. The LEDs are controlled individually on each tier.
2. The water pumps do not turn on when the soil moisture value is below the threshold on both tiers. This indicates that the soil on both tiers is too wet.
3. The water pumps turn on when the soil moisture value is above the threshold on both tiers. This indicates that the soil on both tiers is too dry. This is simulated in the demo video by placing both soil moisture sensors in air.
4. The correct water pump turns on when one tier's soil moisture threshold is surpassed while the other tier's soil moisture threshold is not surpassed. This was simulated in the demo video by purposely wetting one tier's soil moisture level so that it was below the threshold.
5. The user gets a notification on the phone application that the water reservoir needs to be refilled when the water level sensor threshold is reached.
6. Historical empirical data is displayed on the phone application in a graph format.

For features 2-4, the soil moisture threshold of 2800 was chosen based on what the measured soil moisture of the mint and basil plants were at the time of testing. This threshold is a digital value corresponding to an analog voltage. The conversion formula followed by the analog to digital convertor embedded in the microcontroller is $AnalogVoltage = 3.3\ (\frac{DigitalValue}{4096})$. Thus, the 2800 digital value corresponds to an analog voltage of 2.26V. To get a better reference of the range of analog values that the soil moisture sensors are calibrated to, refer to Appendix I. Additionally, refer to Appendix J for the plant profiles of mint and basil. The thresholds defined in the profiles will replace the threshold values used for testing so the system can attain its intended purpose which is long-time, semi-autonomous, and personally curated plant management.

Grading Rubric

With the fulfillment of features 1-6 as described and depicted in the demo video, this places our team's grade in the A range as defined by the grading rubric below. The plant platform fully meets the necessary physical requirements, the sensors are accurate in detecting real time soil moisture and water reservoir level values, the water pumps and LED lighting

respond accurately to defined threshold values, and the user interface is capable of user notification and data storage and display.

Points	Plant Platform	Sensors Subsection	Controls Subsection	User Interface
3	Platform is two-tiered and has a water reservoir on top with two pumps attached	Accurate soil moisture and water reservoir level detection	Accurate control of water pumps and light based on plant profile and sensor data	App receives notification of low water reservoir level and stores historical empirical data
2	Platform is two-tiered and has a water reservoir on top with one pump attached	Soil moisture detection and water reservoir level detection	Decently accurate control of water pumps or light based on plant profile and sensor data	App stores historical empirical data
1	Platform is one-tiered and has one pump attached to a water reservoir	Soil moisture detection and no water reservoir level detection	Slightly inaccurate control of water pumps and no control of light based on plant profile and sensor data	App exists but does not receive any data
0	Platform is one-tiered	No soil moisture or water reservoir level detection	No control of water pumps or light	App does not exist

Points	Grade
10-12	A
7-9	B
4-6	C
0-3	D

Costs

The direct cost of this vertical plant management system is $417.44. This includes all electrical components and mechanical components that we needed to purchase directly from external suppliers such as Digi-Key and McMaster-Carr. This cost does not include mechanical costs associated with materials we already had available such as lumber and caulking. The total cost of the project including extra materials is available in Appendix K in Figure 62. The table below outlines the expenses of this system if it were to be manufactured at both 1-unit and 10000-unit quantities based only on the direct cost:

	1 Unit Cost Per System	10000 Unit Cost Per System
PCB	$33.00	$2.66
PCB Parts	$51.50	$40.32
Sensors and Actuators	$133.71	$133.71
Microcontroller	$15.59	$15.59
WIFI Module	$23.99	$23.99
Mechanical Parts	$159.65	$155.46
Total Cost Per System	$417.44	$371.73

Table 2: System Cost at 1 Unit and 10000 Unit

The sensors and actuators and the mechanical materials are the most expensive parts of the system and are unfortunately not available at bulk price from their distributors. The microcontroller and WIFI module are also not available at a bulk price. However, the PCB cost is reduced significantly at 10000 quantity at only $2.66 per PCB as quoted personally by Advanced Circuits, thus still reducing the unit price per systema good amount. Most of the PCB parts are also available at bulk which reduces the final 10000-unit cost estimate per system to $371.73, a difference of $45.71. To further reduce the cost of the system, it might be recommended to find replacements for parts from other manufacturers that have pricing for larger quantities. To note, the bulk prices used to calculate the estimate are from what was available on Digi-Key which ranged in quantity from around 25 units-1000 units, therefore the 10000-unit estimate is not extremely precise. The costs not included in are the manufacturing process costs. Since the system is relatively simple to manufacture only requiring a PCB with through hole and surface mount parts and also a non-complex mechanical structure, the use of automation would allow for the total costs to not increase by a large amount.

Future Work

On the software side of the system, the phone application is able to display sensor information and give user alerts when the water level on the reservoir is low. This is a one-way communication channel from the microcontroller to the phone application. Due to time constraints of the semester, two-way communication was delegated as a low priority task. The design of the user choice of plant profiles was instead hard-coded into the firmware. In the future, sending information from the phone application to the microcontroller should be implemented to raise the degree of usability that the system has for the intended user group of indoor plant owners.

The team also recommends that other types of sensors be integrated into the system. The current system has soil moisture sensors, water level sensors, and light control but not light

sensing. The LED lights are turned on based on a plant type's ideal light exposure level per day. This does not take into account the intensity of light that a plant needs. Some plants prefer indirect sunlight while others thrive more in direct sunlight. A dimming function on the LED lighting could be added to mimic this effect in the plant management system. A light sensor could take in the ambient light from the environment and adjust the dimming accordingly. There are also other parameters in controlled environment agriculture (CEA) that could be considered. For instance, a plant's health also depends on the temperature, humidity, and carbon dioxide levels [1]. Sensors that indicate to the user when thresholds of these parameters are not ideal would optimize the maintenance of a plant even further and having these additional sensors trigger their own actuator such as a temperature sensor turning on an internal system heater or cooler would bring the system more towards autonomy. With more sensor options, the user could have a choice on the types of sensors that they are most interested in for the type of plant maintenance they would like to enact. These choices could be selected through the phone application interface.

The team also would want to expand the project to be able to manage a wider range of plant types. Currently, the system is a two-tiered platform capable of maintaining two types of plants corresponding to two sets of sensors, water pumps, and LED lighting. Since the system has a vertical design, it is seemingly simple to scale this project up. Each additional tier would require another set of all the listed components which means that the PCB design would need to just be broadened with each succession adding to the cost. However, more components from additional tiers and more sensor types would require more current draw from the wall transformer. Thus, there would need to be a redesign of the power supply system to adapt to a new maximum current draw. Another consideration is that the maximum power dissipation would increase which would affect the operating temperature of the voltage regulator. If that temperature surpasses the sinking capability of the current heat sink, a new heat sink design would need to be implemented. It is pertinent to do the full calculations for the heat sink to ensure proper thermal design based on the manufacturer's datasheet. Without this, the voltage regulator would malfunction or burn up easily.

Overall, the team recommends a more user-friendly and user-flexible design. This can be achieved through the implementation of a two-way communication channel on the phone application user interface and a robust redesign of the power supply system that is capable of handling a large number of attached components. This allows users an easier way to have a higher degree of control in which sensors they would want to have in their vertical plant management system and the amount of plant types that they would want to manage.

References

[1] K. Nemali, “Controlled Environment Agriculture.” Purdue University, Accessed: Sep. 07, 2020. [Online]. Available: https://www.purdue.edu/hla/sites/cea/wp-content/uploads/sites/15/2017/04/Controlled-Environment-Agriculture.pdf.

[2] Ujunwa Nwachukwu, “Health Benefits of Indoor Plants.” Silverhill Institute of Environmental Research and Conservation , Accessed: Sep. 07, 2020. [Online]. Available: https://www.silverhillinstitute.com/pdf/Health_Benefits_of_Indoor_Plant_2013.pdf.

[3] M. I. H. bin Ismail and N. M. Thamrin, “IoT implementation for indoor vertical farming watering system,” in 2017 International Conference on Electrical, Electronics and System Engineering (ICEESE), Nov. 2017, pp. 89–94, doi: 10.1109/ICEESE.2017.8298388.

[4] E. Mackensen, J. Klose, A. Rombach, and A. Spitznagel, “Energy autonomous automation of Smart Home applications using the example of a wireless Indoor Smart Gardening system,” in 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), Aug. 2019, pp. 1087–1092, doi: 10.1109/COASE.2019.8843000.

[5] Md. A. Muhtasim, S. Ramisa Fariha, and A. M. Ornab, “Smart Garden Automated and Real Time Plant Watering and Lighting System with Security Features,” in 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Sep. 2018, pp. 676–679, doi: 10.1109/GUCON.2018.8675077.

[6] “Automated Expedient Watering System For Small Plants And Acquaintance About Deficit In Water Supply,” ResearchGate. https://www.researchgate.net/publication/312653566_Automated_Expedient_Watering_System_For_Small_Plants_And_Acquaintance_About_Deficit_In_Water_Supply (accessed Sep. 07, 2020).

[7] B. R. Sandhya, M. Pallavi, and C. Chandrashekar, “IoT Based Smart Home Garden Watering System Using Raspberry Pi 3.” International Journal of Innovative Research in Science, Engineering and Technology, Jul. 12, 2017, Accessed: Sep. 07, 2020. [Online]. Available: http://www.ijirset.com/upload/2017/iccstar/20_O010.pdf.

[8] “iPřiro Houseplants Automatic Watering Sy- Buy Online in El Salvador at Desertcart.” https://elsalvador.desertcart.com/products/171931293-i-priro-houseplants-automatic-watering-system-automated-watering-device-with-30-day-programmable-timer-and-5-v-usb-charging-cable-for-15-potted-plants-indoor (accessed Sep. 07, 2020).

[9] “The Smart Garden 27,” Click & Grow. https://www.clickandgrow.com/products/smart-garden-27-home-garden (accessed Sep. 07, 2020).

[10] “Hydroponics: A Better Way to Grow Food (U.S. National Park Service).” https://www.nps.gov/articles/hydroponics.htm (accessed Sep. 07, 2020).

[11] “MSP-EXP430F5529LP Development Kit ,” *Texas Instruments*. https://www.ti.com/tool/MSP-EXP430F5529LP (accessed Nov. 27, 2020).

[12] “CC3100BOOST Evaluation Board,” *Texas Instruments*. https://www.ti.com/tool/CC3100BOOST (accessed Nov. 27, 2020).

[13] “What is Multisim?,” NI. https://www.ni.com/en-us/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-multisim.html (accessed Nov. 27, 2020).

[14] “Ultiboard,” NI. https://www.ni.com/en-us/shop/software/products/ultiboard.html (accessed Nov. 27, 2020).

[15] “CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) ,” *Texas Instruments*. https://www.ti.com/tool/CCSTUDIO (accessed Nov. 30, 2020).

[16] “2 Layer & 4 Layer PCB Special Pricing Options,” Advanced Circuits. https://www.4pcb.com/pcb-prototype-2-4-layer-boards-specials.html (accessed Nov. 27, 2020).

[17] “Water Level Sensors,” Vegetronix. https://vegetronix.com/Products/AquaPlumb/ (accessed Nov. 27, 2020).

[18] “464,” Digi-Key. https://www.digikey.com/en/products/detail/adafruit-industries-llc/464/8558263?s=N4IgjCBcoLQdIDGUAuAnArgUwDQgPZQDa4ArAEwAcIAugL517nEgAsAbK7XUA (accessed Dec. 04, 2020).

[19] “NEMA Enclosure Ratings.” [Online]. Available: https://www.nemaenclosures.com/enclosure-ratings/nema-rated-enclosures.html. [Accessed: 09-Dec-2020].

[20] NFPA 70, National Electric Code. National Fire Protection Agency, 2020.

[21] Center for Devices and Radiological Health, “Use of ISO 10993-1, Biological evaluation of medical devices - Part 1,” U.S. Food and Drug Administration, 2020. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/use-international-standard-iso-10993-1-biological-evaluation-medical-devices-part-1-evaluation-and. [Accessed: 09-Dec-2020].

[22]“IPC Standards,” IPC International, Inc., 16-Nov-2020. [Online]. Available: https://www.ipc.org/ipc-standards. [Accessed: 09-Dec-2020].

[23] "Electronic Code of Federal Regulations (eCFR)", *Electronic Code of Federal Regulations (eCFR)*, 2020. [Online]. Available: https://www.ecfr.gov/cgi-bin/text-idx?SID=548c069c8161a207d347bdc60ad98fdc&mc=true&node=pt47.1.15&rgn=div5#_top. [Accessed: 07- Sep- 2020].

[24] B. Ogilvie, Clock Solutions for WIFI. Pericom Semiconductor, 2020.

[25] “What Is VirtualBench?,” NI. https://www.ni.com/en-us/shop/electronic-test-instrumentation/virtualbench/what-is-virtualbench.html (accessed Nov. 27, 2020).

[26] “Inventor,” *Autodesk*. https://www.autodesk.com/products/inventor/overview?plc=INVPROSA&term=1-YEAR&support=ADVANCED&quantity=1.

[27] “Wireshark” *Wireshark · Go Deep.* [Online]. Available: https://www.wireshark.org/. [Accessed: 08-Dec-2020].

[28] “Android Studio and SDK tools: Android Developers,” *Android Developers*. [Online]. Available: https://developer.android.com/studio. [Accessed: 08-Dec-2020].

[29] “Firebase.” *Google*. [Online]. Available: https://firebase.google.com/. [Accessed: 08-Dec-2020].

[30] S. Lebow and M. Tippie, "Guide for minimizing the effect of preservative-treated wood on sensitive environments", *United States Department of Agriculture,* 2001. .

[31] A. Agnihori, V. Gokhale and D. Joshi, "Analysis of Threat from Formaldehyde Exposure in Interior Environment of Urban Built Stock", *Ijrar.com,* 2019. [Online]. Available: http://ijrar.com/upload_issue/ijrar_issue_20543399.pdf. [Accessed: 06- Sep- 2020].

[32] A. Okunola A, O. Kehinde I, A. Oluwaseun and A. Olufiropo E, "Public and Environmental Health Effects of Plastic Wastes Disposal: A Review", *Journal of Toxicology and Risk Assessment,* vol. 5, no. 2, 2019. Available: 10.23937/2572-4061.1510021.

[33] EPA, “Printed Circuit Board Recycling Methods,” *EPA - Documents*, 2012. [Online]. Available: https://www.epa.gov/sites/production/files/2014-05/documents/handout-10-circuitboards.pdf. [Accessed: 08-Dec-2020].

[34] “How Do Fertilizers Affect the Environment? You'll Be Stunned to Know,” *Help Save Nature*, 21-Jun-2011. [Online]. Available: https://helpsavenature.com/how-do-fertilizers-affect-environment. [Accessed: 09-Dec-2020].

[35] “3865,” *Digi-Key*. https://www.digikey.com/en/products/detail/adafruit-industries-llc/3865/9555465 (accessed Dec. 04, 2020).

[36] T. McCoid, “METHOD AND SYSTEM FOR AUTOMATED DATA ANALYSIS OF SOIL| MOISTURE,” *US Patent*, 19-Jul-2016.

[37] J. Caron and J. Bouedreau, “SOIL WATER POTENTIAL DETECTOR,” *US Patent*, 28-Feb-2006.

[38] W. Carlson, “METHOD AND APPARATUS FOR OPTIMIZATION OF GROWTH OF PLANTS,” *US Patent*, 11-Feb-1986.

[39] “NIS5112D2R2G,” Digi-Key. https://www.digikey.com/en/products/detail/on-semiconductor/NIS5112D2R2G/1484386 (accessed Dec. 08, 2020).

[40] “L6R48-120,” Digi-Key. https://www.digikey.com/product-detail/en/tri-mag-llc/L6R48-120400/364-1285-ND/7682648 (accessed Dec. 08, 2020).

[41] “54-00129,” Digi-Key. https://www.digikey.com/product-detail/en/tensility-international-corp/54-00129/839-1512-ND/9685438 (accessed Dec. 08, 2020).

[42] “574902B00000G,” Digi-Key. https://www.digikey.com/en/products/detail/aavid-thermal-division-of-boyd-corporation/574902B00000G/1625571 (accessed Dec. 08, 2020).

[43] “MIC29150-5.0WT,” Digi-Key. https://www.digikey.com/en/products/detail/microchip-technology/MIC29150-5.0WT/771575 (accessed Dec. 08, 2020).

[44] “282834-2,” Digi-Key. https://www.digikey.com/en/products/detail/te-connectivity-amp-connectors/282834-2/1150135 (accessed Dec. 08, 2020).

[45] “ESQ-110-14-T-D,” Digi-Key. https://www.digikey.com/product-detail/en/samtec-inc/ESQ-110-14-T-D/SAM9309-ND/6678046 (accessed Dec. 08, 2020).

[46] “SEN0193,” Digi-Key. https://www.digikey.com/product-detail/en/dfrobot/SEN0193/1738-1184-ND/6588605 (accessed Dec. 08, 2020).

[47] “1150,” Digi-Key. https://www.digikey.com/product-detail/en/adafruit-industries-llc/1150/1528-1404-ND/5638299 (accessed Dec. 08, 2020).

[48] “NBF-32018 Bud Industries: Boxes, Enclosures, Racks,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/bud-industries/NBF-32018/2328538. [Accessed: 09-Dec-2020].

[49] Texas Instruments. “MSP-EXP430FR5969 Development Kit | TI.Com.” Ti.com. https://www.ti.com/tool/MSP-EXP430FR5969 (Accessed September 14, 2020).

[50] Texas Instruments “CC3100 Getting Started With WLAN Station - Texas Instruments Wiki.” https://processors.wiki.ti.com/index.php/CC3100_Getting_Started_with_WLAN_Station (accessed 13 October 2020).

[51] Texas Instruments. “CC3100 TCP Socket Application - Texas Instruments Wiki.” http://processors.wiki.ti.com/index.php/CC31xx_TCP_Socket_Application (accessed 13 October 2020).

[52] “Visual Studio Code - Code Editing. Redefined,” *RSS*, 14-Apr-2016. [Online]. Available: https://code.visualstudio.com/. [Accessed: 08-Dec-2020].

[53] “The Collaboration Platform for API Development,” *Postman*. [Online]. Available:

[54] “Growing Mint Inside: Information On Planting Mint Indoors,” *Gardening Know How*. https://www.gardeningknowhow.com/edible/herbs/mint/growing-mint-indoors.htm (accessed Dec. 07, 2020).

[55] O. F. Almanac, “Basil,” *Old Farmer’s Almanac*. https://www.almanac.com/plant/basil (accessed Dec. 07, 2020).

Appendix

Appendix A: Thermal Calculations

The datasheet for the voltage regulator used in the power supply subsystem gave guideline thermal equations used to determine the need for a heat sink [42]. To use the equations, the output current of the voltage regulator is needed. Table 3 below shows how the output current is the sum of the currents drawn from various components that are powered from the voltage regulator.

Part	# of Devices	Current [mA]	Total Current [mA]
Soil Moisture Sensor	2	5	10
Water Level Sensor	1	13	13
MSP	1	17.2	17.2
WiFi Module	1	325.69	325.69
		Total Current Draw Needed [mA]	365.89

Table 3: Summary of Current Needed from Voltage Regulator Powered Components

The total output current is therefore 0.36589A. This current value as well as a 12V input voltage from the wall transformer (V_{in}), a 5V output voltage (V_{reg}), an ambient temperature (T_A) of 20°C, and a package thermal resistance of 56°C/W (θ_{JA}) are used in the following calculations.

$$Power\ Dissipation = (V_{in} - V_{reg}) \cdot I_{out} - (V_{in} \cdot 0.0016A)$$

$$= (12 - 5) \cdot 0.36589 - (12 \cdot 0.0016) = \mathbf{2.54W}$$

$$Junction\ Temperature = (\theta_{JA} \cdot Power\ Dissipation) + T_A = (56 \cdot 2.54) + 20$$

$$= \mathbf{162.24^{\circ}C}$$

Since the junction temperature for the voltage regulator's expected power dissipation is greater than the maximum allowable temperature of 125°C (T_{JMAX}), a heat sink is needed. The following calculations find the thermal resistance that is needed for a heat sink with the expected power dissipation (θ_{SA}). In the calculations, the given value of the package thermal resistance (θ_{JC}) is 2 for the TO-220 package.

$$Power\ Dissipation\ =\ I_{out}\,(1.01(V_{in}\ -\ V_{reg}\,))\ =0.36589\ (1.01(12-5))\ =\ 2.587W$$

$$\theta_{SA}\ =\ \frac{T_{JMAX}\ -\ T_A}{Power\ Dissipation}\ -\ (\theta_{JC}\ +\ 2)\ =\frac{125\ -\ 20}{2.587}\ -\ (2+\ 2)\ =36.59°C/W$$

Thus, a heat sink with a thermal resistance of 36.59°C/W is needed. The heat sink chosen has a thermal resistance of 16°C/W which provides a higher level of thermal protection [42].

Appendix B: PCB Layout Layers

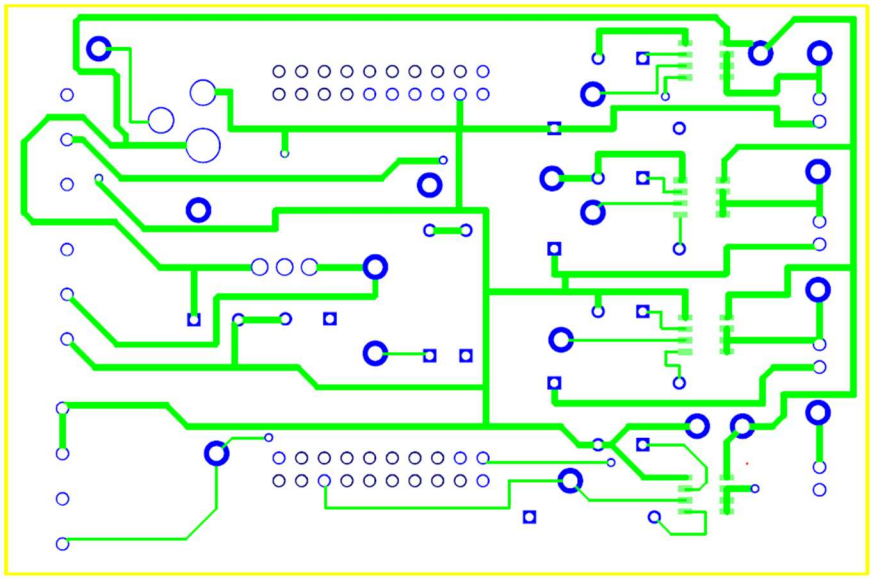


Figure 33: Copper Top

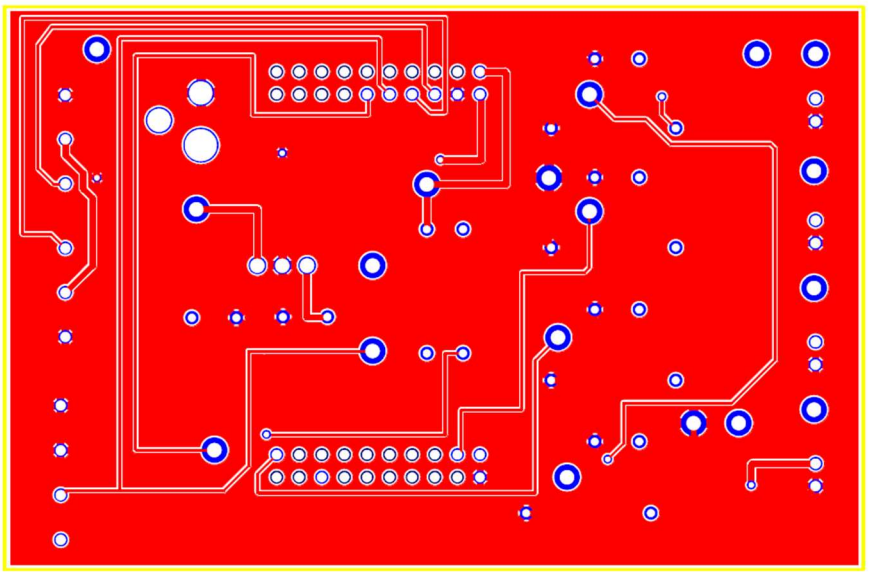


Figure 34: Copper Bottom

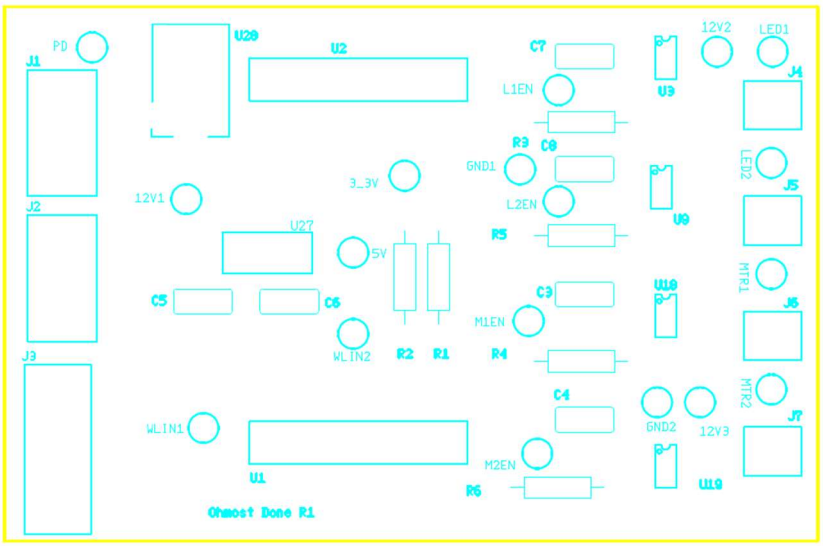


Figure 35: Silkscreen Top

Appendix C: Microcontroller and WIFI Module Pin Out Sheets

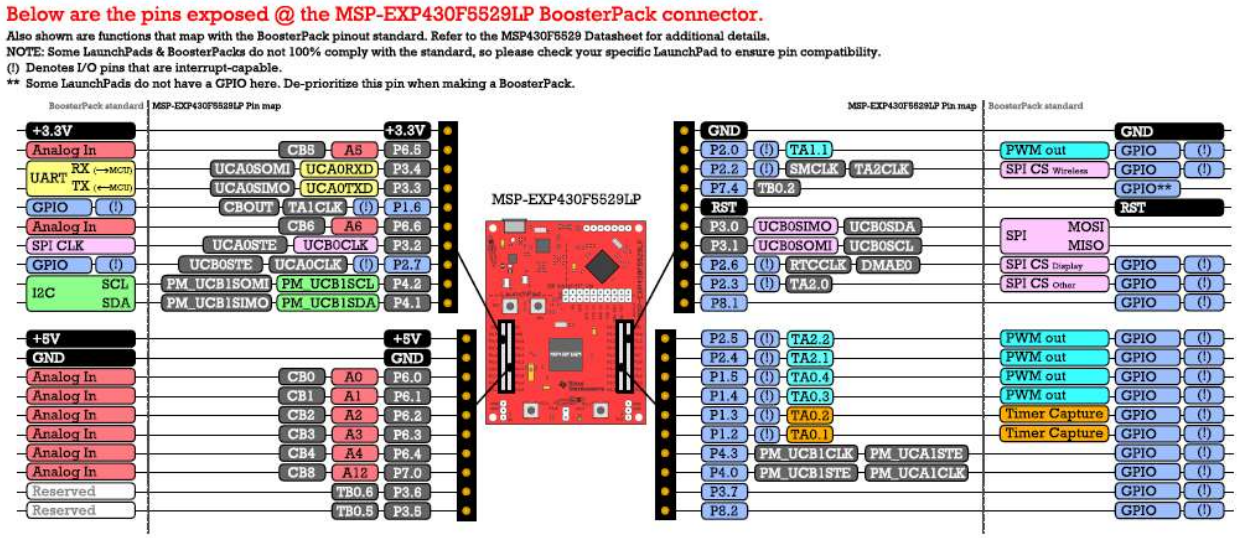
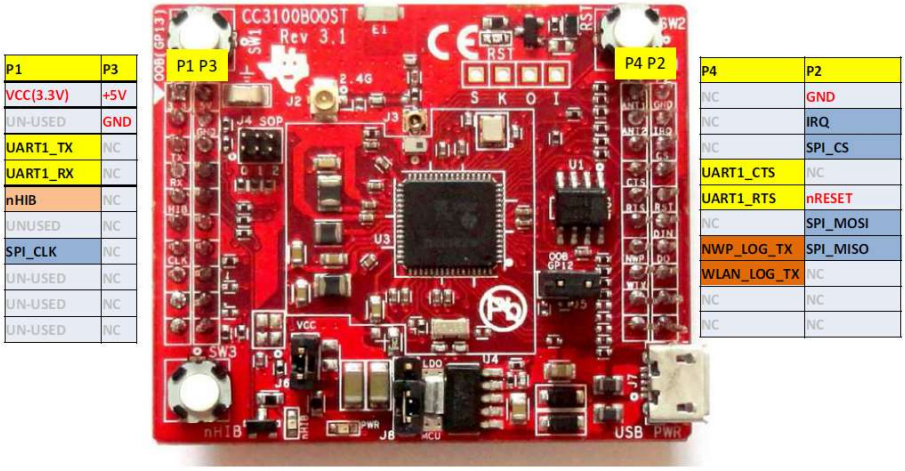


Figure 36: Microcontroller Pin Sheet



P1	P3
VCC(3.3V)	+5V
UN-USED	GND
UART1_TX	NC
UART1_RX	NC
nHIB	NC
UNUSED	NC
SPI_CLK	NC
UN-USED	NC
UN-USED	NC
UN-USED	NC

P4	P2
NC	GND
NC	IRQ
NC	SPI_CS
UART1_CTS	NC
UART1_RTS	nRESET
NC	SPI_MOSI
NWP_LOG_TX	SPI_MISO
WLAN_LOG_TX	NC
NC	NC
NC	NC

Figure 37: WIFI Module Only Pin Sheet

Identifier	Use	Identifier	Use		Identifier	Use	Identifier	Use		Modules:	# of Pins needed	
3.3V		5V			P2.5	LED 1 (J4)	GND			LED 1	1	OUT
P6.5		GND			P2.4	LED 2 (J5)	P2.0			LED 2	1	OUT
P3.4	UART1	P6.0	Moisture 1 ADC (J1 Mint)		P1.5		P2.2			Moisture 1	1	IN
P3.3	UART1	P6.1	Moisture 2 ADC (J2 Basil)		P1.4	UART1	P7.4			Moisture 2	1	IN
P1.6	nHIB	P6.2	Water Level ADC		P1.3	UART1	RST			Open Water 1	1	OUT
P6.6		P6.3	Water Level ADC		P1.2		P3.0			Open Water 2	1	OUT
P3.2		P6.4			P4.3	NWP_LOG_TX	P3.1			Water Level	2	IN
P2.7		P7.0			P4.0	WLAN_LOG_TX	P2.6	Water Logic 2		Wifi Module	UART	IN/OUT
P4.2		P3.6	Reserved		P3.7		P2.3					
P4.1		P3.5	Reserved		P8.2	REASSIGNED	P8.1	Water Logic 1				

Figure 38: Pin Sheet for Project

Appendix D: Network Testing

26164	556.601417	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=906 Win=65535 Len=0
26165	556.601417	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=1087 Win=65535 Len=0
26166	556.601417	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=1268 Win=65535 Len=0
26168	556.611403	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26169	556.611678	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26170	556.611953	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26171	556.612682	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26172	556.613790	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26173	556.615054	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26174	556.615378	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26175	556.615756	192.168.1.37	216.239.36.54	HTTP	235 POST /user HTTP/1.1 , JavaScript Object Notation (application/json)Continuation
26176	556.615940	192.168.1.37	216.239.36.54	TCP	54 62407 → 80 [FIN, ACK] Seq=2716 Ack=1 Win=33580 Len=0
26177	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=1449 Win=65535 Len=0
26178	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=1630 Win=65535 Len=0
26179	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=1811 Win=65535 Len=0
26180	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=1992 Win=65535 Len=0
26181	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=2173 Win=65535 Len=0
26182	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=2354 Win=65535 Len=0
26183	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=2535 Win=65535 Len=0
26184	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=2716 Win=65535 Len=0
26185	556.641833	216.239.36.54	192.168.1.37	TCP	56 80 → 62407 [ACK] Seq=1 Ack=2717 Win=65535 Len=0
26197	557.175675	216.239.36.54	192.168.1.37	HTTP	305 HTTP/1.1 201 Created

Figure 39: Network Trace for WIFI Module

```
Getting started with station application - Version 1.3.0
**************************************************************
Device is configured in default state
Device started as STATION
Connection established w/ AP and IP is acquired
Pinging...!
Device successfully connected to the LAN
Device successfully connected to the internet
```

Figure 40: Debugging Terminal for WIFI Module

```
POST /user HTTP/1.1
Host: us-central1-ohmost-done.cloudfunctions.net
Content-Type: application/json
Content-Length: 18
User-Agent: MSP430F5529/0.5.1

{"SMMData":"0000"}
```

Figure 41: POST Request Template Format

```
void createString(char * buffer, int data1, char * field){
    char string1[5] = "00000";

    sprintf(string1, "%d", data1);

    int nullchar;

    for (nullchar = 0; nullchar < 5; nullchar ++){
        if (string1[nullchar] == '\0'){
            break;
        }
    }

    int place = 171;
    int loc;
    for (loc = nullchar-1; loc >= 0; loc--){
        buffer[place] = string1[loc];
        place--;
    }

    if (place != 168){
        while (place >= 168){
            buffer[place] = '0';
            place--;
        }
    }

    buffer[158] = field[0];
    buffer[159] = field[1];
    buffer[160] = field[2];

}
```

Figure 42: createString implementation

Appendix E: Firmware Clock Code Listing

```
void initClk()
{
    /* Setup XT1 and XT2 */
    XT1_XT2_PORT_SEL |= XT1_ENABLE + XT2_ENABLE;

    /* Set Vcore to accomodate for max. allowed system speed */
    SetVCore(3);
    /* Use 32.768kHz XTAL as reference */
    LFXT_Start(XT1DRIVE_0);

    /* Set system clock to max (25MHz) */
    Init_FLL_Settle(25000, 762);
    SFRIFG1 = 0;
    SFRIE1 |= OFIE;

    /* Globally enable interrupts */
    __enable_interrupt();
}
```

Figure 43: InitClk function. Initializes the main clock on the controller and enables interrupts.

```
unsigned long platform_get_time_in_secs()
{
    return sec_tick;
}

unsigned long platform_get_time_in_mins()
{
    return min_tick;
}


unsigned long platform_get_time_in_hrs()
{
    return hour_tick;
}
```

Figure 44: “Get” functions for each fraction of time

```
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=TIMER2_A0_VECTOR, TIMER2_A1_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(USCI_A0_VECTOR)))
#endif
void TIMER_A2_ISR(void)
{
    sec_tick++;
    //enableADCinterrupt();
    if (sec_tick % 60 == 0){
        min_tick++;
        if (min_tick % 60 == 60){
            hour_tick ++;
            min_tick = 0;
        }
        sec_tick = 0;
    }

    TA2CTL &= ~TAIFG;
}
```

Figure 45: Timer 2 Interrupt Service Routine for global timer values

Appendix F: Analog to Digital Conversion Code Snippets

```
void initADC()
{
    P6SEL = 0x0F;                                 // Enable A/D channel inputs
    ADC12CTL0 = ADC12ON + ADC12MSC + ADC12SHT0_8; // Turn on ADC12, extend sampling time
                                                  // to avoid overflow of results
    ADC12CTL1 = ADC12SHP + ADC12CONSEQ_3;         // Use sampling timer, repeated sequence
    ADC12MCTL0 = ADC12INCH_0;                     // ref+=AVcc, channel = A0
    ADC12MCTL1 = ADC12INCH_1;                     // ref+=AVcc, channel = A1
    ADC12MCTL2 = ADC12INCH_2;                     // ref+=AVcc, channel = A2
    ADC12MCTL3 = ADC12INCH_3 + ADC12EOS;          // ref+=AVcc, channel = A3, end seq.
    ADC12IE = 0x08;                               // Enable ADC12IFG.3
    ADC12CTL0 |= ADC12ENC;                        // Enable conversions
    ADC12CTL0 |= ADC12SC;                         // Start convn - software trigger
}
```

Figure 46: Initialize the ADC

```
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(ADC12_VECTOR))) ADC12ISR (void)
#else
#error Compiler not supported!
#endif
{
    static unsigned int index = 0;

    switch(__even_in_range(ADC12IV,34))
    {
        case 0: break;                          // Vector  0:  No interrupt
        case 2: break;// Vector  2:  ADC overflow
        case 4: break;// Vector  4:  ADC timing overflow
        case 6: break;// Vector  6:  ADC12IFG0
        case 8: break;// Vector  8:  ADC12IFG1
        case 10: break;// Vector 10:  ADC12IFG2
        case 12:// Vector 12:  ADC12IFG3
        A0results[index] = ADC12MEM0;// Move A0 results, IFG is cleared
        A1results[index] = ADC12MEM1;// Move A1 results, IFG is cleared
        A2results[index] = ADC12MEM2;// Move A2 results, IFG is cleared
        A3results[index] = ADC12MEM3;// Move A3 results, IFG is cleared
        index++;// Increment results index, modulo; Set Breakpoint1 here

        if (index == 8)
        {
            (index = 0);
        }

        //disableADCinterrupt();
        case 14: break;// Vector 14:  ADC12IFG4
        case 16: break;// Vector 16:  ADC12IFG5
        case 18: break;// Vector 18:  ADC12IFG6
        case 20: break;// Vector 20:  ADC12IFG7
        case 22: break;// Vector 22:  ADC12IFG8
        case 24: break;// Vector 24:  ADC12IFG9
        case 26: break;// Vector 26:  ADC12IFG10
        case 28: break;// Vector 28:  ADC12IFG11
        case 30: break;// Vector 30:  ADC12IFG12
        case 32: break;// Vector 32:  ADC12IFG13
        case 34: break;// Vector 34:  ADC12IFG14
        default: break;
    }
}
```

Figure 47: ADC ISR

```c
int SMB_Avg()
{
    int sum = 0;
    int ind;
    for (ind = 0; ind < 8; ind++)
    {
        sum += A0results[ind];
    }
    return (sum >> 3);
}

int SMM_Avg()
{
    int sum = 0;
    int ind;
    for (ind = 0; ind < 8; ind++)
    {
        sum += A1results[ind];
    }
    return (sum >> 3);
}
```

```c
int WL_Avg()
{

    int sum = 0;
    int ind;
    for (ind = 0; ind < 8; ind++)
    {
        sum += A2results[ind];
    }
    return (sum >> 3);
}

int WL_Ref()
{

    int sum = 0;
    int ind;
    for (ind = 0; ind < 8; ind++)
    {
        sum += A3results[ind];
    }
    return (sum >> 3);
}
```

```c
int WL_Level(int ref, int avg)
{
    if (avg > 3330){
        return 1111;
    }

    int dif = 3330 - avg;

    int perc = dif / 20;

    return perc;
}
```

Figure 48: ADC Circular Buffer Averaging Functions

Appendix G: General Purpose Input/Output Code Snippets

```
void initLEDs()
{
    P2OUT &= ~BIT4;
    P2OUT &= ~BIT5;
    P2DIR |= BIT4;
    P2DIR |= BIT5;
    P2OUT &= ~BIT4;
    P2OUT &= ~BIT5;
}
```

```
void initPumps()
{
    P2OUT &= ~BIT6;
    P8OUT &= ~BIT2;

    P2DIR |= BIT6;
    P8DIR |= BIT1;

    P2OUT &= ~BIT6;
    P8OUT &= ~BIT1;
}
```

Figure 49: GPIO Initialize Functions

```
void turnLedOn(char ledNum)
{
    switch (ledNum)
    {
    case LED1:
        P2OUT |= BIT4;
        break;
    case LED2:
        P2OUT |= BIT5;
        break;
    }
}
```

```
void turnPumpOn(char pumpNum)
{
    switch (pumpNum)
    {
    case PUMP1:
        P2OUT |= BIT6;
        break;
    case PUMP2:
        P8OUT |= BIT1;
        break;
    }
}
```

Figure 50: GPIO Turn On Functions

```
void turnLedOff(char ledNum)
{
    switch (ledNum)
    {
    case LED1:
        P2OUT &= ~BIT4;
        break;
    case LED2:
        P2OUT &= ~BIT5;
        break;
    }
}
```

```
void turnPumpOff(char pumpNum)
{
    switch (pumpNum)
    {
    case PUMP1:
        P2OUT &= ~BIT6;
        break;
    case PUMP2:
        P8OUT &= ~BIT1;
        break;
    }
}
```

Figure 51: GPIO Turn Off Functions

```
void toggleLed(char ledNum)
{
    switch (ledNum)
    {
    case LED1:
        P2OUT ^= BIT4;
        break;
    case LED2:
        P2OUT ^= BIT5;
        break;
    }

}
```

```
void togglePump(char pumpNum)
{
    switch (pumpNum)
    {
    case PUMP1:
        P2OUT ^= BIT6;
        break;
    case PUMP2:
        P8OUT ^= BIT1;
        break;
    }

}
```

Figure 52: GPIO Toggle Functions

```c
unsigned char GetLEDStatus()
{
    unsigned char status = 0;

    if (P2OUT & BIT4)
        status |= (1 << 0);
    if (P2OUT & BIT5)
        status |= (1 << 1);

    return status;
}
```

```c
unsigned char GetPumpStatus()
{
    unsigned char status = 0;

    if (P2OUT & BIT6)
        status |= (1 << 0);
    if (P8OUT & BIT2)
        status |= (1 << 1);

    return status;
}
```

Figure 53: GPIO Status Functions

```c
int DisableLED(char LEDNum)
{
    int value = 0;
    switch (LEDNum)
    {
    case LED1:
        value = P2OUT & BIT4;
        P2OUT ^= value;
        break;
    case LED2:
        value = P2OUT & BIT5;
        P2OUT ^= value;
        break;
    }
    return value;
}

void EnableLED(char LEDNum, int value)
{
    P2OUT ^= value;
}
```

Figure 54: GPIO Enable/Disable LED Functions

Appendix H: Server and Application Code

```
app.get("/", async (req, res) => {
  const snapshot = await admin.firestore().collection("users").get();

  let users = [];
  snapshot.forEach((doc) => {
    let id = doc.id;
    let data = doc.data();

    users.push({ id, ...data });
  });

  res.status(200).send(JSON.stringify(users));
});

app.get("/:id", async (req, res) => {
    const snapshot = await admin.firestore().collection('users').doc(req.params.id).get();

    const userId = snapshot.id;
    const userData = snapshot.data();

    res.status(200).send(JSON.stringify({id: userId, ...userData}));
})

app.post("/", async (req, res) => {
  const user = req.body;

  await admin.firestore().collection("users").add(user);

  res.status(201).send();
});
```

Figure 55: REST API used for Firebase Functions

```
class BasilActivity : AppCompatActivity() {

    private val mDocRef : DocumentReference  = FirebaseFirestore.getInstance().document( documentPath: "users/HwduazgxMgLGkByWNw8u");
    private val waterRef : Query  = FirebaseFirestore.getInstance().collection( collectionPath: "users").whereNotEqualTo( field: "WLDData",  value: "")
//    private val list : MutableList<String> = mutableListOf<String>("SM1Data", "SM2Data", "WLData")
    private val soilRef : Query  = FirebaseFirestore.getInstance().collection( collectionPath: "users").whereNotEqualTo( field: "SMBData",  value: "")

    private lateinit var basilWaterTextData : TextView
    private lateinit var basilSoilTextData : TextView
    private lateinit var basilFetchData : Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        var view = setContentView(R.layout.activity_basil)
        basilWaterTextData = findViewById(R.id.text_basil_water_level);
        basilSoilTextData = findViewById(R.id.text_basil_soil_moisture_level);
        basilFetchData = findViewById(R.id.button_basil)

        basilFetchData.setOnClickListener { view ->
            fetchWaterData(view)
            fetchSoilMoistureData(view)

        }
    }

    fun fetchWaterData(view : View){
        waterRef.get().addOnSuccessListener { docs ->
            for(doc in docs){
                // Grab the most up to date content
                var x = doc.data.values.toString().removeSurrounding( prefix: "[",  suffix: "]").toInt()/2
                basilWaterTextData.text = x.toString()
            }
```

Figure 56: Database Retrieval for Mobile Application

Appendix I: Soil Moisture Sensor Calibration Testing

Figure 57: Medium Soil Moisture Analog Sensor Value

Figure 58: Medium-Wet Soil Moisture Analog Sensor Value

Figure 59: Wet Soil Moisture Analog Sensor Value

Appendix J: Plant Profiles

Mint [54]

Soil moisture threshold: 2.265V or 2811

Lighting: 7 hours of light per day

Basil [55]

Soil moisture threshold: 2.265V or 2811

Lighting: 6 hours of light per day

Appendix K: Finances

RefDes	Description	Vendor	Manufacturer	Manufacturer Part No.	Vendor Part No.	1 Unit Price	Quantity Needed	Total Price
J4,J5,J6,J7	2 Terminal Wire Connector	DigiKey	TE Connectivity	282834-2	A98333-ND	1.25	4	5.00
J1,J2	3 Terminal Wire Connector	DigiKey	TE Connectivity	282836-3	A98077-ND	1.24	2	2.48
J3	4 Terminal Wire Connector	DigiKey	TE Connectivity	282836-4	A98078-ND	1.42	1	1.42
U1,U2	Header Board Connector	DigiKey	Samtec Inc.	ESQ-110-14-T-D	SAM9309-ND	2.53	2	5.06
U3	Wall Transformer	DigiKey	Tri-Mag, LLC	L6R48-120	364-1285-ND	16.87	1	16.87
U4	Barrel Jack Connector	DigiKey	Tensility International	54-00129	839-1512-ND	0.93	1	0.93
U5	Voltage Regulator	DigiKey	Microchip Technology	MIC29150-5.0WT	576-1106-ND	3.51	1	3.51
N/A	Clip-On Heat Sink	DigiKey	Aavid	574902B00000G	574902B00000G-ND	2.05	1	2.05
U18,U19,U28,U29	Electronic Fuse	DigiKey	ON Semiconductor	NIS5112D2R2G	NIS5112D2R2GOSCT-ND	2.92	4	11.68
C3,C4,C5,C6	Capacitor, 1uF	DigiKey	KEMET	C333C105K5R5TA	399-13915-ND	0.43	4	1.72
C2	Capacitor, 10uF	DigiKey	KEMET	C322C106K3R5TA	399-13968-ND	0.78	1	0.78
U12	Water Level Sensor	DigiKey	Adafruit Industries LLC	464	1528-2561-ND	39.95	1	39.95
U9,U10	Soil Moisture Sensor	DigiKey	DFRobot	1738-1184-ND	1738-1184-ND	7.98	2	15.96
U26,U27	Peristaltic Liquid Pump	DigiKey	Adafruit Industries LLC	1150	1528-1404-ND	24.95	2	49.90
U16,U17	LED Strip Lighting	DigiKey	Adafruit Industries LLC	3865	1528-2657-ND	13.95	2	27.90
N/A	Microcontroller	DigiKey	Texas Instruments	MSP-EXP430F5529LP	296-36506-ND	15.59	1	15.59
U36	WIFI Module	DigiKey	Texas Instruments	CC3100BOOST	296-37769-ND	23.99	1	23.99
N/A	Wiring	DigiKey	Alpha Wire	3051/1 BK005	A3051B-100-ND	40.41	1	40.41
N/A	NEMA Enclosure	DigiKey	Bud Industries	NBF-32018	377-1766-ND	25.40	1	25.40
N/A	Plastic Bins	Walmart	Sterilte	N/A	N/A	3.56	5	17.80
N/A	Silicone Tubing	McMaster-Carr	N/A	N/A	5054K326	11.90	1	11.90
N/A	0.19"-0.25" Cord Grip	McMaster-Carr	N/A	N/A	7529K173	9.80	1	9.80
N/A	0.25"-0.31" Cord Grip	McMaster-Carr	N/A	N/A	7529K102	9.50	2	19.00
N/A	Locknut & O-Ring for 1/2 Cord Grip	McMaster-Carr	N/A	N/A	7466K37	7.28	1	7.28
N/A	Locknut & O-Ring for 1/4 Cord Grip	McMaster-Carr	N/A	N/A	7466K35	1.65	2	3.30
N/A	1" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	7861K48	7.68	1	7.68
N/A	0.253" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	73115K71	8.70	1	8.70
N/A	0.16" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	6855K14	8.38	1	8.38
N/A	PCB	Advanced Circuits	N/A	N/A	N/A	33.00	1	33.00
							Total System Cost	417.44

Figure 60: Bill of Materials. Breakdown of System Costs at 1 Unit

RefDes	Description	Vendor	Manufacturer	Manufacturer Part No.	Vendor Part No.	10000 Unit Price	Quantity Needed	Total Price
J4,J5,J6,J7	2 Terminal Wire Connector	DigiKey	TE Connectivity	282834-2	A98333-ND	0.49	4	1.96
J1,J2	3 Terminal Wire Connector	DigiKey	TE Connectivity	282836-3	A98077-ND	0.49	2	0.98
J3	4 Terminal Wire Connector	DigiKey	TE Connectivity	282836-4	A98078-ND	0.65	1	0.65
U1,U2	Header Board Connector	DigiKey	Samtec Inc.	ESQ-110-14-T-D	SAM9309-ND	1.51	2	3.02
U3	Wall Transformer	DigiKey	Tri-Mag, LLC	L6R48-120	364-1285-ND	16.87	1	16.87
U4	Barrel Jack Connector	DigiKey	Tensility International	54-00129	839-1512-ND	0.46	1	0.46
U5	Voltage Regulator	DigiKey	Microchip Technology	MIC29150-5.0WT	576-1106-ND	2.66	1	2.66
N/A	Clip-On Heat Sink	DigiKey	Aavid	574902B00000G	574902B00000G-ND	1.37	1	1.37
U18,U19,U28,U29	Electronic Fuse	DigiKey	ON Semiconductor	NIS5112D2R2G	NIS5112D2R2GOSCT-ND	2.92	4	11.68
C3,C4,C5,C6	Capacitor, 1uF	DigiKey	KEMET	C333C105K5R5TA	399-13915-ND	0.11	4	0.44
C2	Capacitor, 10uF	DigiKey	KEMET	C322C106K3R5TA	399-13968-ND	0.23	1	0.23
U12	Water Level Sensor	DigiKey	Adafruit Industries LLC	464	1528-2561-ND	39.95	1	39.95
U9,U10	Soil Moisture Sensor	DigiKey	DFRobot	1738-1184-ND	1738-1184-ND	7.98	2	15.96
U26,U27	Peristaltic Liquid Pump	DigiKey	Adafruit Industries LLC	1150	1528-1404-ND	24.95	2	49.90
U16,U17	LED Strip Lighting	DigiKey	Adafruit Industries LLC	3865	1528-2657-ND	13.95	2	27.90
N/A	Microcontroller	DigiKey	Texas Instruments	MSP-EXP430F5529LP	296-36506-ND	15.59	1	15.59
U36	WIFI Module	DigiKey	Texas Instruments	CC3100BOOST	296-37769-ND	23.99	1	23.99
N/A	Wiring	DigiKey	Alpha Wire	3051/1 BK005	A3051B-100-ND	40.41	1	40.41
N/A	NEMA Enclosure	DigiKey	Bud Industries	NBF-32018	377-1766-ND	21.21	1	21.21
N/A	Plastic Bins	Walmart	Sterilte	N/A	N/A	3.56	5	17.80
N/A	Silicone Tubing	McMaster-Carr	N/A	N/A	5054K326	11.90	1	11.90
N/A	0.19"-0.25" Cord Grip	McMaster-Carr	N/A	N/A	7529K173	9.80	1	9.80
N/A	0.25"-0.31" Cord Grip	McMaster-Carr	N/A	N/A	7529K102	9.50	2	19.00
N/A	Locknut & O-Ring for 1/2 Cord Grip	McMaster-Carr	N/A	N/A	7466K37	7.28	1	7.28
N/A	Locknut & O-Ring for 1/4 Cord Grip	McMaster-Carr	N/A	N/A	7466K35	1.65	2	3.30
N/A	1" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	7861K48	7.68	1	7.68
N/A	0.253" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	73115K71	8.70	1	8.70
N/A	0.16" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	6855K14	8.38	1	8.38
N/A	PCB	Advanced Circuits	N/A	N/A	N/A	2.66	1	2.66
							Total System Cost	371.73

Figure 61: Bill of Materials. Breakdown of System Costs at 10000 Unit

RefDes	Description	Vendor	Manufacturer	Manufacturer Part No.	Vendor Part No.	Unit Price	Quantity Purchased	Total Price
J4,J5,J6,J7	2 Terminal Wire Connector	DigiKey	TE Connectivity	282834-2	A98333-ND	1.25	4	5.00
J1,J2	3 Terminal Wire Connector	DigiKey	TE Connectivity	282836-3	A98077-ND	1.24	2	2.48
J3	4 Terminal Wire Connector	DigiKey	TE Connectivity	282836-4	A98078-ND	1.42	1	1.42
U1,U2	Header Board Connector	DigiKey	Samtec Inc.	ESQ-110-14-T-D	SAM9309-ND	2.53	2	5.06
U3	Wall Transformer	DigiKey	Tri-Mag, LLC	L6R48-120	364-1285-ND	16.87	1	16.87
U4	Barrel Jack Connector	DigiKey	Tensility International	54-00129	839-1512-ND	0.93	1	0.93
U5	Voltage Regulator	DigiKey	Microchip Technology	MIC29150-5.0WT	576-1106-ND	3.51	2	7.02
N/A	Clip-On Heat Sink	DigiKey	Aavid	574902B00000G	574902B00000G-ND	2.05	2	4.10
U18,U19,U28,U29	Electronic Fuse	DigiKey	ON Semiconductor	NIS5112D2R2G	NIS5112D2R2GOSCT-ND	2.92	8	23.36
C3,C4,C5,C6	Capacitor, 1uF	DigiKey	KEMET	C333C105K5R5TA	399-13915-ND	0.43	8	3.44
C2	Capacitor, 10uF	DigiKey	KEMET	C322C106K3R5TA	399-13968-ND	0.78	2	1.56
U12	Water Level Sensor	DigiKey	Adafruit Industries LLC	464	1528-2561-ND	39.95	1	39.95
U9,U10	Soil Moisture Sensor	DigiKey	DFRobot	1738-1184-ND	1738-1184-ND	7.98	2	15.96
U26,U27	Peristaltic Liquid Pump	DigiKey	Adafruit Industries LLC	1150	1528-1404-ND	24.95	2	49.90
U16,U17	LED Strip Lighting	DigiKey	Adafruit Industries LLC	3865	1528-2657-ND	13.95	2	27.90
N/A	Microcontroller	DigiKey	Texas Instruments	MSP-EXP430F5529LP	296-36506-ND	15.59	1	15.59
U36	WIFI Module	DigiKey	Texas Instruments	CC3100BOOST	296-37769-ND	23.99	1	23.99
N/A	Wiring	DigiKey	Alpha Wire	3051/1 BK005	A3051B-100-ND	40.41	1	40.41
N/A	NEMA Enclosure	DigiKey	Bud Industries	NBF-32018	377-1766-ND	25.40	1	25.40
N/A	Plastic Bins	Walmart	Sterilte	N/A	N/A	3.56	5	17.80
N/A	Silicone Tubing	McMaster-Carr	N/A	N/A	5054K326	11.90	1	11.90
N/A	0.19"-0.25" Cord Grip	McMaster-Carr	N/A	N/A	7529K173	9.80	1	9.80
N/A	0.25"-0.31" Cord Grip	McMaster-Carr	N/A	N/A	7529K102	9.50	2	19.00
N/A	Locknut and O-Ring for 1/2 Cord Grip	McMaster-Carr	N/A	N/A	7466K37	7.28	1	7.28
N/A	Locknut and O-Ring for 1/4 Cord Grip	McMaster-Carr	N/A	N/A	7466K35	1.65	2	3.30
N/A	1" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	7861K48	7.68	1	7.68
N/A	0.253" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	73115K71	8.70	1	8.70
N/A	0.16" Heat Shrink Wrap	McMaster-Carr	N/A	N/A	6855K14	8.38	4	33.52
N/A	PCB	Advanced Circuits	N/A	N/A	N/A	33.00	2	66.00
							Total Project Cost	495.32

Figure 62: Breakdown of Total Project Cost