HomeGrow

Low-Cost Automated System for Growing Produce at Home

Ciara Ward, Sanjana Sastry, and Stephanie Sieben
Team 13
TA: Daniel Ahn

Senior Design Project

Department of Electrical and Computer Engineering
University of Illinois Urbana-Champaign
Spring 2022

Abstract

The goal of this project was to create a low-cost, autonomous, vertical gardening system to assist people in growing organic herbs at home. The device provides the water and light requirements for the plants survival. A moisture sensor manages the watering cycle for the plants, and the lights are set on a schedule optimal for plant growth and nutrition. The user interface provides either an alert that watering will begin in 10 minutes or that the water cycle is being skipped. This prevents user interference with the plants during a watering cycle. The report begins with an introduction about the incentive to work on this problem and the goals set. After presenting the physical design, the paper details the high level requirements for success, and wrap up the introductory section with block diagram and physical design of our model. Following that, subsystems are described in detail, including requirements and verification's for these subsystems. Relevant costs for materials are mentioned, described and calculated. The report concludes with challenges that were faced, what could change, and what could be developed or added if this project were to continue further.

Contents

1 Introduction

1.1 Problem

During the pandemic, growing plants have been both a way to sustain people's positive moods and also elevate their mental well-being. Many people desire to consume fresh and organic produce, but lack the means to get it. Healthy and organic food can be difficult to find, but even if you can find it, it is usually expensive. A great alternative is to grow organic food at home, this also includes an added benefit of relieving stress-induced depression. This however presents issues as many working-class people do not necessarily have the time, knowledge, or means to keep their plants thriving all the time. There are automated plant systems on the market, but they market around hundreds of dollars, even with small plant capacities.

Making fresh food resources expensive only increases the already existing food gap for those with lower incomes and their ability to eat healthy and sustain themselves.

1.2 Solution

Our solution is to create a vertical-farming-based automated system that provides the best growing conditions for a variety of different herbs. The system was designed with LED grow lights and watering schedules to match the needs of the designated plants.

We can place our setup anywhere, without the constraint of locations near windows for the natural light, or to optimize space. There is a built-in lighting system that allows for this flexibility. The lighting and watering is controlled by a microcontroller which contains the details of the needs of several different plants, with a focus on herbs.

1.3 Visual Aid

Figure 1, shown below, contains the labeled components of the design. The soaker hose in our system is made with polyvinyl chloride (PVC) tubing with holes drilled into the middle to allow for water to seep through. The microcontroller controls the water flow, horticulture Light Emitting Diodes(LEDs), and the Liquid Crystal Display(LCD). The figure also depicts the power coming in from the wall and the use of a moisture sensor to prevent over-watering.

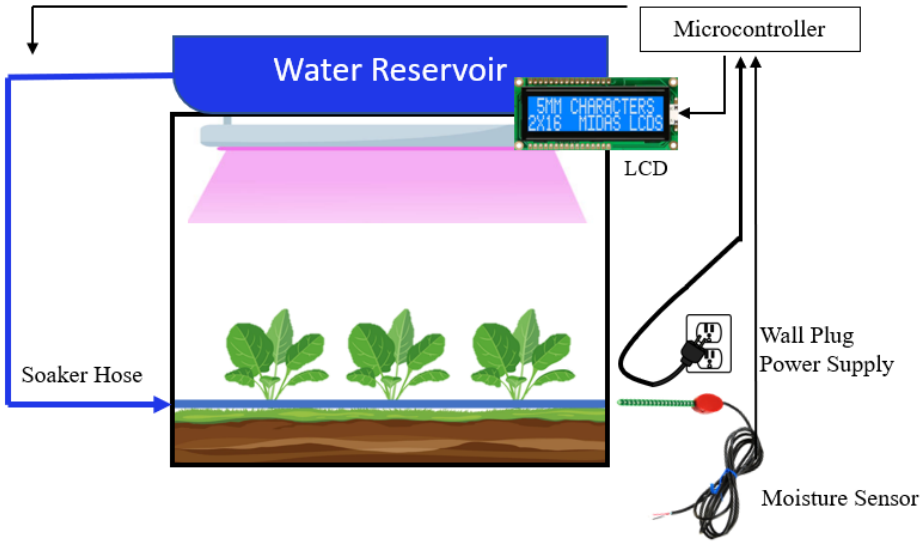

Figure 1: Visual Aid Design

1.4 High-Level Requirements List

- The user interface will notify the user before a watering cycle if a watering cycle is being skipped or alert 10 min before the valve is opened
- Solenoid water valves will release water at a steady flow rate for 15 minutes every 24 hour period, except when the moisture sensor detects moisture levels above 60% saturation [1]
- LED grow lights will turn on when instructed by the microcontroller and will be on for 12 hours and shut off for 12 hours

1.5 Block Diagram

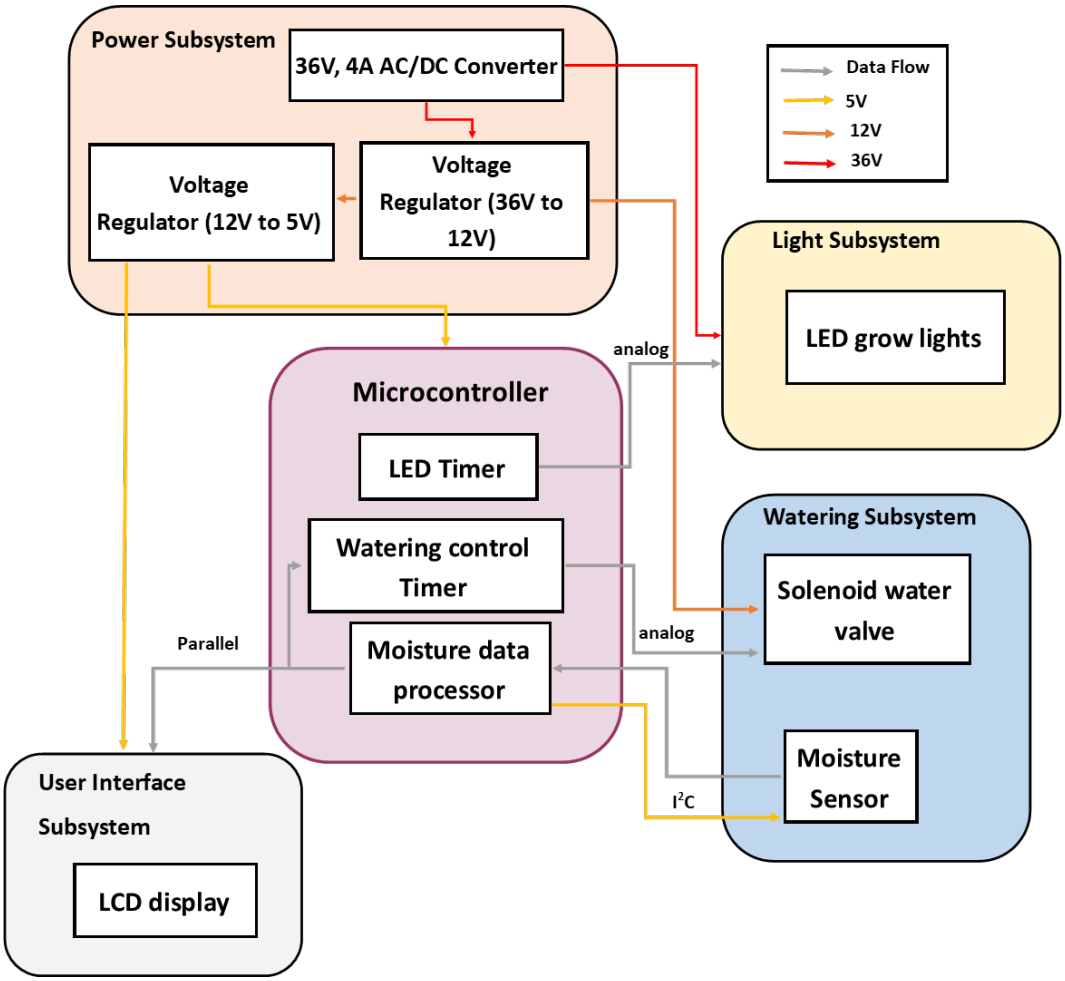

Figure 2: Block Diagram

The block diagram, depicted in Figure 2, provides a visual representation of the power and data flow between each subsystem to create an automated system.

1.6 Physical Design

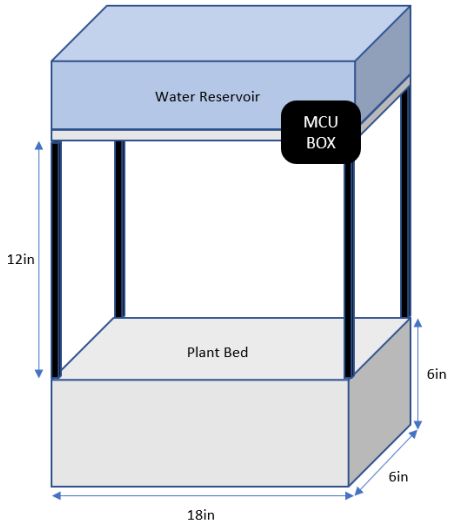

Figure 3: Physical Design Mock Up

Our physical diagram, shown in Figure 3, demonstrates a single tier of our original proposed two tier system. We decided to scale it down from two tiers in order to bring down the cost and increase usability in small spaces. In our design water flows down from the reservoir down to the plant bed via tubes. The moisture sensor in this design will be placed fully submerged into the soil, to have a better idea of the saturation through the soil. The plant bed has designed with a small footprint to be more accessible to those with limited space.

Figure 4: Final Build

The final build of the project was similar to the original proposed design. Figure 3 shows a picture of our final product.

2 Design Process

2.1 Watering Subsystem

2.1.1 Description

In the watering system, there is a water reservoir on the top that utilizes gravitational flow to water each plant bed. From the water reservoir there is a tube or PVC pipe to bring the water down to the soil. This tubing lies on top of the soil in the plant bed and has small holes around it to let the water out into the soil, similar to a soaker hose. There is a solenoid water walve to regulate the flow of water from the reservoir into the PVC tubes. This valve is normally closed and opens when there is voltage supplied. This solenoid valve is connected to our microcontroller with a transistor, where we can control the duration of plant watering. The valve is set to be open for 15 minutes every 24 hours.

2.1.2 Design

One of the most common problems with indoor gardening is over-watering. To avoid this from happening there is a moisture sensor in the soil. This sensor is connected to the microcontroller to tell the system its moisture level and determine if the watering cycle should be skipped. The microcontroller subsystem is programmed such that it waters for 15 minutes a day at the same time every day unless the moisture level detected by the moisture sensor is sufficient, then it will skip that water cycle and water the next day. Underneath the planter box, there is a tray to collect any water run off from the drain holes as to prevent over watering.

The solenoid water valve receives power from the voltage regulator subsystem, the current will run through the solenoid when the transistor receives high input from the microcontroller. The solenoid is connected in parallel with a fly-back diode in order to dissipate the current safely. The moisture sensor is powered by the microcontroller and sends the data back to the microcontroller. Both the transistor and the diode are the components suggested on the datasheet of the water valve. This schematic diagram can be found in Appendix A, Figure 13.

2.1.3 Verification

In order to test if the watering system met one of the requirement laid out in the table of requirements in Appendix C, Table 4, we connected the water valve to the power supply in the lab with 12 V of direct power to see if it opened. We found that the water goes through the valve when the reservoir is halfway full or above because it needs more water pressure in order to release water with the design that the machine shop created.

Table 1: Flow Rate Data

Test #	Duration (min)	Water Collected (oz)	Calculated Flow Rate (oz/min)	Calculated Flow Rate (mL/min)
1	10	4	0.4	11.83
2	15	6.5	0.43	12.72
3	15	6.75	0.45	13.31
4	15	6.75	0.45	13.31
5	15	6.5	0.43	12.72

Table 1, shown above, illustrates the testing that was done on the watering system to determine its flow rate and usability. Here we filled the water reservoir with water and allowed the solenoid valve to remain open for time intervals, measured the collected water and then calculated the corresponding flow rate in both mL/min and oz/min. The water was measured in ounces and using Formula 1, converted to milliliters, and Formula 2 to get the final flow rate in mL/minute.

$$WaterCollected(mL) = Water(oz) * 29.5735 \tag{1}$$

$$WaterFlowRate(mL/min) = \frac{WaterCollected(mL)}{Time(min)} \tag{2}$$

This fulfilled two of the water subsystem requirements, that the valve could remain open for 15 minutes and that 4oz could be collected in 30 minutes or less. The average flow rate for the system ended up being 12.78 ml/min. The graph of these results shown in Appendix D illustrates the consistency of our flow rate measurements.

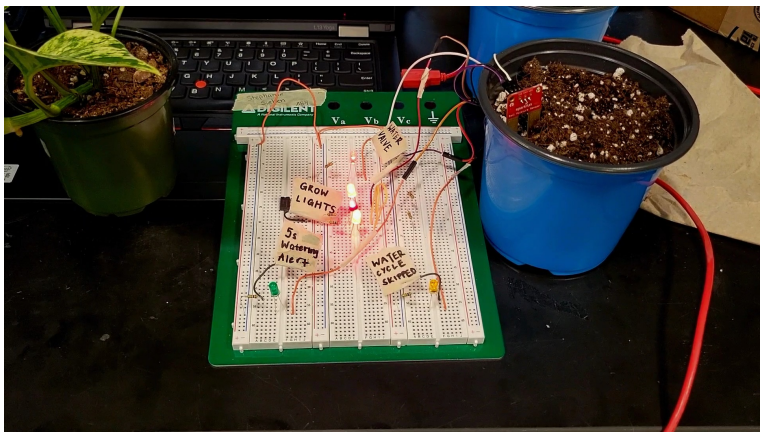

Figure 5: Water Valve Simulation

Figure 5, above, shows the test performed to prove that the water valve opens when the soil moisture is below 500 (60%) saturation.The LED in the middle and back represents the water valve being powered and the three LEDs in the middle represent the grow lights being on during the day.

2.2 Lighting Subsystem

2.2.1 Description

Light is an essential part of plant growth because it is needed for photosynthesis to take place. Mimicking sunlight as close as possible is a key component of our project keeping plants alive. We have selected to use LED's as our light source because it is the easiest way to manipulate wavelengths of light. In our research, we found that the 400-500 nanometers (nm) spectrum, or blue light, is helpful in increasing the quality of the plant. This wavelength promotes stomatal opening which is what allows CO2 to enter the leaves. The red light spectrum of 600-700 nm is also essential for the health of the plants. Red light encourages photosynthesis because it gets absorbed by the chlorophyll pigments. Because of this, it promotes stem and leaf growth. However, if you only give it a red light, then it would be overstretched without having the strong roots it needs. Therefore, we use a balance of both red and blue LED strips to encourage healthy plant growth[2] The LEDs are controlled by the microcontroller. Inside the microcontroller there is a timer that turns them on for 12 hours and off for another 12 hours.

2.2.2 Design

The LEDs are powered by the 28V input from the voltage regulator. The IM02 relay is what allows the microcontroller to switch the LEDs on and off by sending voltage to the relay. This connection can be seen in Figure 14 in Appendix A. The IM02 is rated for 220VDC and 2A which is well over the requirements for our system which are 28VDC and 1.25A.

2.2.3 Verification

To verify that the 28 V listed on the data sheet is the minimum voltage for the lights to turn on, we connected the power supply in the lab directly to the copper pads on the grow lights, the power supply had a max of 25 V, and none of these were able to get the lights to turn on.

We then connected our AC/DC wall plug which gave 36 V and 300 mA to the lights and this turned on the blue LEDs, but caused the red LEDs to flash. We concluded that there was not enough current supplied. We got a replacement wall adapter that supplied the LEDs with 4A, well exceeding the 1.25 Amp minimum, and this allowed for all LEDs to have a steady illumination. The table of requirements for this subsystem can be found in Appendix C, Table 5.

2.3 User Interface Subsystem

2.3.1 Description

This subsystem allows for the user to be notified when the plants will be watered so they can avoid harvesting or replanting during the watering cycle. LED indicators were implemented to inform the user which state it is in, either that the plants will be watered in three minutes or that the watering cycle will be skipped.

2.3.2 Design

As shown in Figures 1 and 2, we had originally designed the user interface to be an LCD screen where we displayed words to tell the user the state of the watering cycle and give an exact time and countdown until the water valve is to be opened. Due to complications in the designing, which is noted in Section 5.1.1 of this document, we decided to use LED indicators instead. In making this decision, we considered both cost of materials and the universal design with avoiding language barriers. Our final design had two LEDs, one indicating low moisture and that the water valve will open in 10 minutes; the other to indicate high moisture and that the watering cycle will be skipped.

2.3.3 Verification

To verify the requirements for this subsystem, which can be found in Appendix C, Table 5, the LEDs were connected to the microcontroller and moisture sensor. Two test conditions were created, one with moist soil and one with dry soil.The test code was ran to ensure that the proper lights turned on if the moisture level was either above or below 60% saturation. We were able to verify that these LED indicators gave the user the correct information. Figure 6, shows the 5 second watering alert in the demo code timing of 24 hours condensed into 6 minutes. Figure 7 below shows the LED indicator on to indicate the moisture level was high and the water cycle is being skipped.

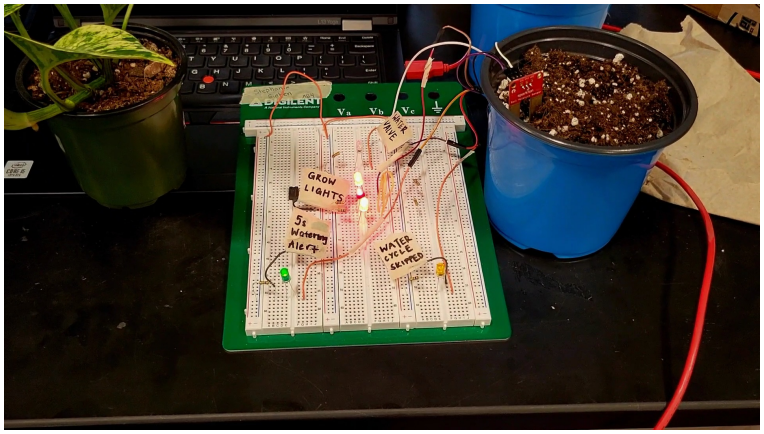

Figure 6: Water Cycle Alert

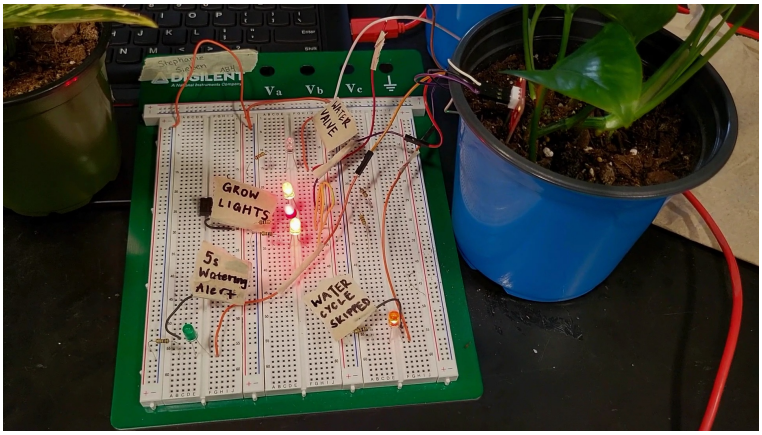

Figure 7: Water Cycle Skipped

2.4 Power Subsystem

2.4.1 Description

In the interest of usability of the device, it is powered by a standard wall outlet. This power subsystem provides power to all the other subsystems in the design. Standard power output from wall outlet is 120V in the United States. The adapter that takes the 120V of AC power and covert it into 36V of DC Power. This is used to power the grow lights. An LM3480 voltage regulator steps down the voltage to 12V which is needed to open the solenoid water valve. This 12V will serve as the input of the next regulator.The LM1117-5, which is a 5 volt regulator that is simulated below in Figure 8. The 5V that it outputs is used to power both the microcontroller and the User Interface System.

2.4.2 Design

Each of the voltage regulators in the system are hooked up according to their respective datasheet, as seen in Figure 8 below. During the hardware design phase, three voltage regulators were implemented because the lights were rated to 28 V. However, during the testing phase it was found that the first regulator to 28V was not working.The lights were then tested with 36 V and they worked, so we did not work to address this issue as it made no difference in functionality. The design of each voltage regular can be found in Appendix A, Figure 15.

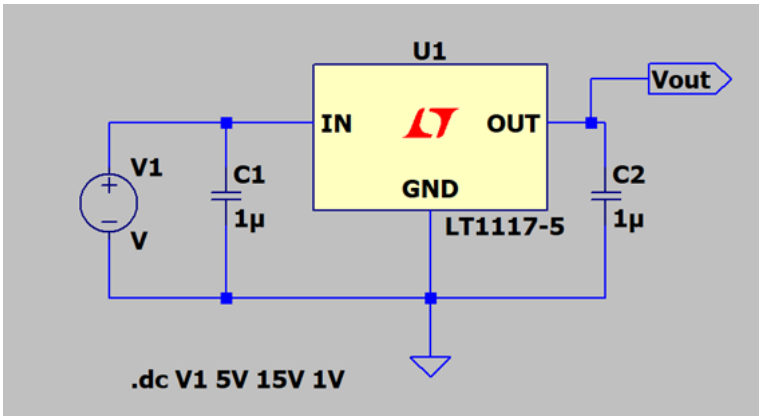

Figure 8: LTSPice Schematic

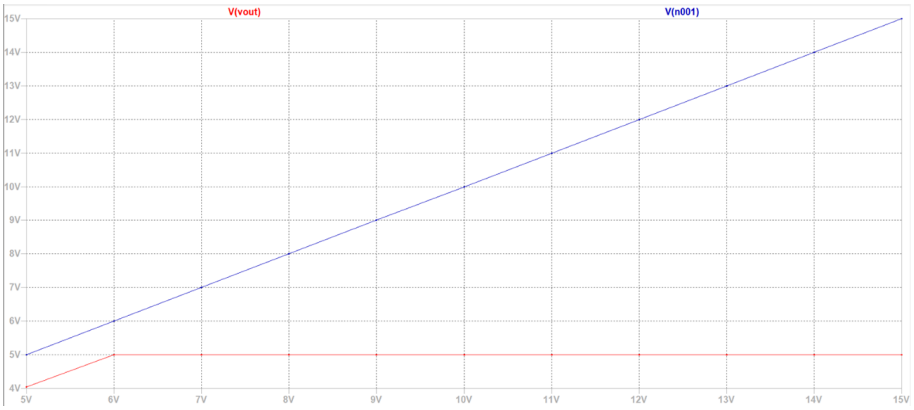

Figure 9: DC Sweep from LTSpice

The simulation in Figure 9 proved that the fixed regulators will output the desired voltage even if the input voltage is larger than the input that was planned for in the design.

2.4.3 Verification

As stated in the design section, the 36V regulating to 28V did not work as we intended. There was not access to a voltage supply that went above 25V so we tested with the wall adapter AC/DC converter that had been ordered that provided 36V. The datasheet of the grow lights did not specify whether the 28 V was the minimum or maximum voltage needed. We supplied the lights directly from the power supply with 25 V and they did not turn on. So we made an informed decision that the voltage rating was the minimum and that it could be powered with 36 V without damaging the LEDs. We then tested this by directly soldering the barrel adapter from the wall and plugging it into the lights. This was able to power the lights, but caused the red LEDs to flash.

After reviewing the specifications on both the wall adapter and the grow lights, we determined that we ordered an adapter that supplied 300mA instead of the 1.25A that was

needed for the lights. Instead of trying to amplify the current, we ordered a 36V and 4A wall adapter which powered the lights and the LEDs were one consistent brightness.

We kept our printed circuit board (PCB), shown in Appendix B, Figure 17, the same because when testing with the wall plug hooked up and probing with the multi-meter. Figure 10 shows the output of the 12V regulator and Figure 11 shows the output of the 5V regulator. Both of these have an output within the range of tolerance set in the requirements and verification table which can be found in Appendix C: Table 5.

Figure 10: Digital Multimeter Output

Figure 11: Digital Multimeter Output

2.5 Microcontroller

2.5.1 Description

The microcontroller is the central processing element of the project and tells the other subsystems what to do. It receives and interprets data from all input systems and sensors. The microcontroller receives power from the power subsystem and it sends out the voltage to the other three subsystems when necessary. There is code that will send out voltage for 12 hours for the lights to be on, then wait 12 hours and repeat. Similarly, the microcontroller will send current to open the water valves, keep it open for a set amount of time before stopping the current and closing the valve. This will occur at set times unless the signals from

the moisture sensor tells it not to. The moisture sensor sends data to the microcontroller to tell it how much moisture is in the soil. The microcontroller will turn on the moisture sensor when it needs to read the data, and turn it off at other times as to not have electricity and water for long periods of time. The microcontroller will send data to the LCD screen to give the user information about the watering schedule.

2.5.2 Design

When designing the microcontroller subsystem we identified which pins were needed and what ports should be avoided. Port A on the chip consisted of basic I/O pins that were all usable for our design. Port B had the serial peripheral interface(SPI) pins needed to use the USBasp Programmer that the lab has available to use. The schematic diagram to show these connections can be found in Appendix A, Figure 16.

2.5.3 Verification

After attempting multiple tool configurations to program the microcontroller. We were unsuccessful in this verification found in the requirement and verification's in Appendix C, Table 8. This shortcoming is further discussed in Section 5.1.2.

2.6 Design Alternatives

Throughout this project we came across several design issues and inconsistencies between the intended output and actual output of our product. One design issue that we came across towards the end of our build process was the integration of our LCD (Liquid Crystal Display). As described previously we had intended this to provide a countdown until the next watering cycle or notify if the watering cycle is being skipped due to soil over saturation. This became an issue when the LCD was essentially unresponsive to inputs. The display was supplied with 5V and tested with a simple "Hello World" code to determine if it was functioning, but it did not respond and would cause the development board to disconnect from the laptop. When tested in the lab it overheated and was completely unusable. To remedy this issue we created a new User Interface that consists of two different colored Light Emitting Diode (LED). The moisture sensors will take an average of readings 10 minutes before the set watering time and if the average is less than 500 (60%) the first LED will illuminate 10 minutes before the watering begins and stay until it starts watering. If the average is greater than 500 (60%) the second LED will illuminate 10 minutes before the intended watering time, indicating the water cycle is being skipped. This proved to to fulfill the initial intention of the LCD but at a cheaper cost and simpler process.

We had another issue with the program-ability of our microcontroller. We had used Arduino and Microchip Studio to try and program our microcontroller but they were unable to flash our code. Upon further research we discovered that the Arduino IDE did not support the ATmega325PA that we had implemented into the design. We switched to programming the ports in Microchip Studio which is known to support this particular microcontroller. With the provided SPI programmer from the course staff, we were unable to build the right tools in order to make everything compatible. To resolve this we created a temporary fix of using a development board to demonstrate the validity and usability of our code.

3 Cost & Schedule

3.1 Cost Analysis

Labor Cost: The average salary of a person with a Bachelors of Science in Electrical Engineering from Illinois is approximately 79,000[3]. Given there are approximately 2080 working hours in a year, the average hourly rate is about $38/hr.
Student labor hours: $10 weeks * 8 hr/week * \$38 * 2.5 * 3 people = \$22,800$
Machine shop labor hours: $\$50/hour * 5 hours = \$250$

Table 2: Parts Breakdown

Part	**Price**	**Quantity**	**Final Cost**
Microcontroller(ATMEGA325)	$5.18	1	$5.18
LED grow light	$9.31	1	$9.31
12 volt regulator	$1.78	1	$1.78
Relay(IM04)	$4.41	1	$4.41
5 volt regulator	$1.64	1	$1.64
flyback diode	$0.41	1	$0.41
TIP120 Transistor	$0.75	1	$0.75
BJT TRANS NPN 20V 0.3A	$0.46	1	$0.46
Plastic Water Solenoid Valve	$6.95	1	$6.95
Moisture Sensor	$6.95	1	$6.95
PCB Board	$0.33	3	$1
Wall Adapter	$6.99	1	$6.99
Planter box	$6.38	1	$6.38

Total raw material cost = $86.47

3.2 Schedule

Below, in Table 3, is the timeline of the project and which team member was in charge of each component.

Table 3: Project Schedule

Deadline	**Task**	**Who is taking lead**
2/24/22	Design Document	Team
2/27/22	Finalize parts list	Stephanie
3/1/22	Design Review	Team
2/29/22	PCB Board basic mock-up	Stephanie
3/2/22	Order parts	Sanjana
3/5/22	Discuss design and dimensions of watering system	Ciara
3/8/22	PCB Audit	Stephanie
3/9/22	Design voltage regulator circuit	Ciara
3/10/22	Write light control code	Sanjana
3/11/22	Turn materials into machine shop	Team
3/24/22	Test soil sensor on current plants and Arduino module	Stephanie
3/25/22	Test LCD Display	Sanjana
3/30/22	Write watering system code	Stephanie&Sanjana
4/4/22	Program mcu and test that it can run simultaneously	Team
4/8/22	Assemble prototype	Team
4/11/22	Run tests on prototype and record videos	Team
4/18/22	Mock Demo	Team
4/25/22	Final Demo	Team
5/4/22	Final Paper	Team

4 Ethics and Safety

4.1 Ethical Considerations

The IEEE Code of Ethics [4] Section 1.2 discusses the importance of ethical conduct "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems". This is relevant to our project as we are creating an intelligent watering system with the purpose of aiding people obtain higher quality and healthier foods.

Another ethical issue that we had to resolve was that of privacy. We had considered making our device app or web compatible for the user to see the health or state of the plant while away. This would mean the data could be hacked or viewed by outsiders who are not intended to see this private data. To prevent this we decided to keep our device offline and keep communications local to the microcontroller only.

Finally working in a group, it is important to pay mind to the IEEE Code of Ethics Section 2.2, where we commit "to treat all persons fairly and with respect, and to not engage in discrimination...". These values will help to promote a healthy and conducive work environment.

4.2 Safety Considerations

Section 1.1 of the IEEE code of Ethics [4] Section 1.1 indicates that it is important "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment". This is relevant and important to our project as we are creating a system that incorporates water and electricity for proper function. In order to eliminate any issues of this sort the design is created such that the water and electrical components will be kept on different sides of the product.The design also includes a pipe watering method instead of an overhead watering so that the lights and the water would avoid contact with each other. However, the water release valve needs to be connected to power. In this case, we will keep the wires and water double-insulated to ensure the safety of the user. In addition, the solenoid valve is designed for water usage, and should not pose any issues either. When applicable, we are also adhering to IEEE Standards like 833-2005[5], on the Recommended Practice for the Protection of Electric Equipment from Water.

5 Conclusion

5.1 Accomplishments

Despite the roadblocks we faced, which is discussed in Section 5.1, we accomplished many of the goals for the design. We set out to design an autonomous system and this was achieved by the water and light subsystems functioning without any input from the user, aside from filling the water reservoir. Our final product remained low cost, at around $85, to make and would be an even lower cost if it were to be mass produced. The User Interface was designed to be more universal than originally intended. Eliminating the LCD took away the need to reprogram for a different language, and is now a matter of changing the labeling on the manufactured product.

We were also able to fully integrate our code together. Each subsystem code worked well together to create the automated system. This was initially a concern because each member of the team is an electrical engineering major with limited coding experience. With software being a weakness for us, producing a functioning code that allowed for the intended automation is something we are proud of. Additionally, we stayed diligent, on top of our schedule, and prepared for each due date and presentation. Our organizational skills allowed for us to capitalize on non-technical points and present ourselves and the project professionally and competently.

We successfully designed a product to help minimize the food gap and help people to achieve a sustainable lifestyle. This to us was the main focus as food scarcity and access to healthy foods is an issue that effects many people and their overall quality of life. Proving that a system like this can be made at such a low cost is our greatest accomplishment throughout this semester.

5.2 Challenges

5.2.1 User Interface

We had originally intended to use an LCD as the user interface for our project, and ordered a standard 16 x 2 LCD. To make sure the LCD was functional, we wired it to an Arduino and attempted to run simple code through it.

Upon compiling the code, the Arduino shut off with a 5V input. Further research suggested that there could be a short in the wiring, but double-checking it confirmed that it was not the most probable issue. Another effort we made to troubleshoot this problem was altering the current on the potentiometer that was used while wiring up the LCD, but this did not fix it either. When switching the power input from 5V to 3.3V, we managed to get the LCD backlight to turn on and flicker, but it did not display anything. Our next option was to bring the LCD into the lab to test it with a separate power supply, thinking that perhaps the Arduino was being overloaded. In the lab, we delivered a little over 5V to the LCD in the hopes that something would turn on, but ended up overheating the display and rendering it unusable. In the end, it is surmised that the LCD was a faulty piece.

5.2.2 Microcontroller

We ordered an ATmega325PA to use in our design because it had sufficient I/O pins. Since ATmega was a familiar name, we made the assumption that our chosen microcontroller could program with the Arduino IDE similarly to the ATmega328. We wrote the

arduino code and were ready to program, but after searching for the correct tools and board managers we hit a dead end, with most contributors mentioning that the ATmega325 was not compatible with the Arduido IDE.

Since Arduino was not an option to for our microcontroller unit (MCU), we then downloaded Microchip Studio, which is closely related to Atmel studio, and designed to program Atmel MCUs. After understanding how to translate our code from programming individual pins into programming ports, we learned how to set the input and output pins within a single port and perform bit-wise operation to set these pins to a high or low value. We were able to do this with successful builds of the code, but were ultimately unable to create the right set up for the board to get recognized for loading the program. We suspect that the generic SPI programmer was not compatible with either the program or the microcontroller.

5.3 Takeaways

We learned a few technical skill throughout this project as well. One of the skills was hardware design. During the course of our electrical engineering curriculum, we did a lot with calculating circuits and learning all the elements, but we had never gotten the chance to put these into practice and design our own project. This took a lot of reading datasheets and putting together knowledge of what we had learned in order to build a successful schematic. In relation to this, during this project was the first time any of us had designed a printed circuit board. Learning KiCad and PCB design is a good skill we can add to our resumes for the future. We had all worked with Arduino programming, which is what we intended to use. However, we were forced to learn how to handle ports on a board which allowed us to become more versatile and adaptable in our programming. These hardware and software skills are ones that will strengthen us and engineers and employees.

5.4 Future Work

A more sustainable lifestyle is the future for most individuals. Our project was designed to fit this need at a more affordable rate than the average product on the market. Our design is made to hold approximately 18 herbs and provide the light and water they need to survive with a raw material cost of less than $100. As shown in Figure 12 below, our plants per cost ratio is much lower than the average on the market.

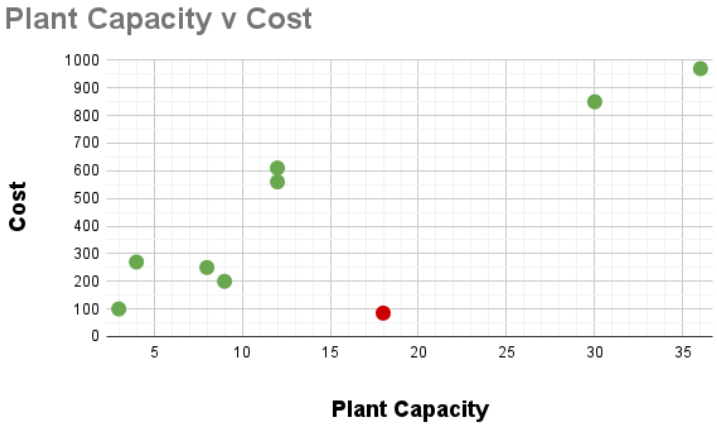

Figure 12: Market Comparison (Green) to Final Build (Red)

Our design does a good job of providing more for the plants than other ones that have watering indicator but does not actually water. While our design is good, there are many things we would improve in our design before bringing it to the market. One major change we can make is to utilize the real time clock feature on a microcontroller. This could allow for the user to set their own watering time schedule and to allow the back-end programming easier by using time instead of timers. In order to implement this, we would need to select a microcontroller that has a built in RTC and has an appropriate number of I/O pins, as our current one has too many unused pins.

One problem we had in the lab was moving the project around, it was a sturdy design, but it was heavy and the water spilled out easily if we tried to move it while it was only halfway full. We would want to build a structure that has a lid for the water reservoir with a lid to make sure that no water would get spilled during the process of filling the reservoir. We also would like to go with a clear front panel to allow the user to see when it needs to be filled. We can also look into materials that can decrease the weight of the design without making it too top heavy when filled, but a simpler solution would be to add small wheels so it can be moved more easily when needed. Another thing with watering that we noticed was that the water would mostly come out of the middle holes, so we would design a new pipe system that has holes closer to the bottom to increase the coverage of watering so all plants are watered evenly.

Our product is designed to give its user notifications about the watering cycle, but there is more plant information that can be useful for a more advance plant grower. If our target audience was the advance skills we could include a pH level sensor for the soil so the user can check that the herbs have healthy soil to promote growth. In addition, the user may like to have a soil and air temperature sensor to keep these within the tolerance of certain plants. These can be helpful improvements, however, temperature and pH levels are likely to remain fairly consistent and would drive up material and manufacturing costs more than anything.

Appendix A: Schematic Diagrams

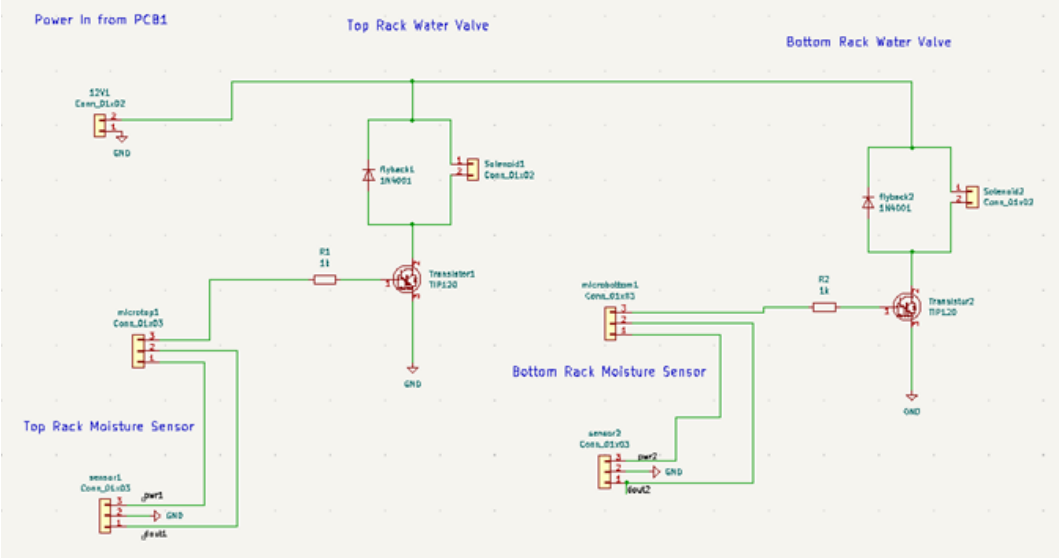

Figure 13: Schematic of the Watering Subsystem

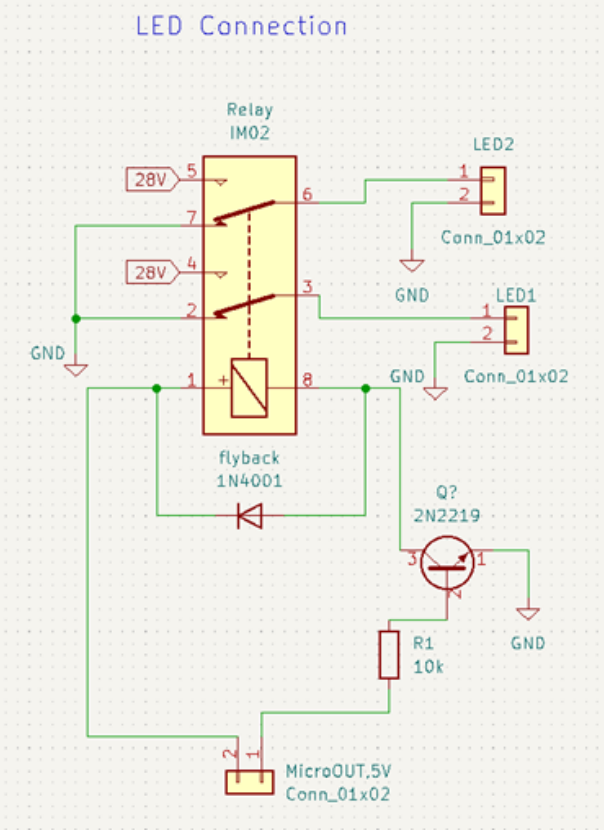

Figure 14: Schematic of the Light Subsystem

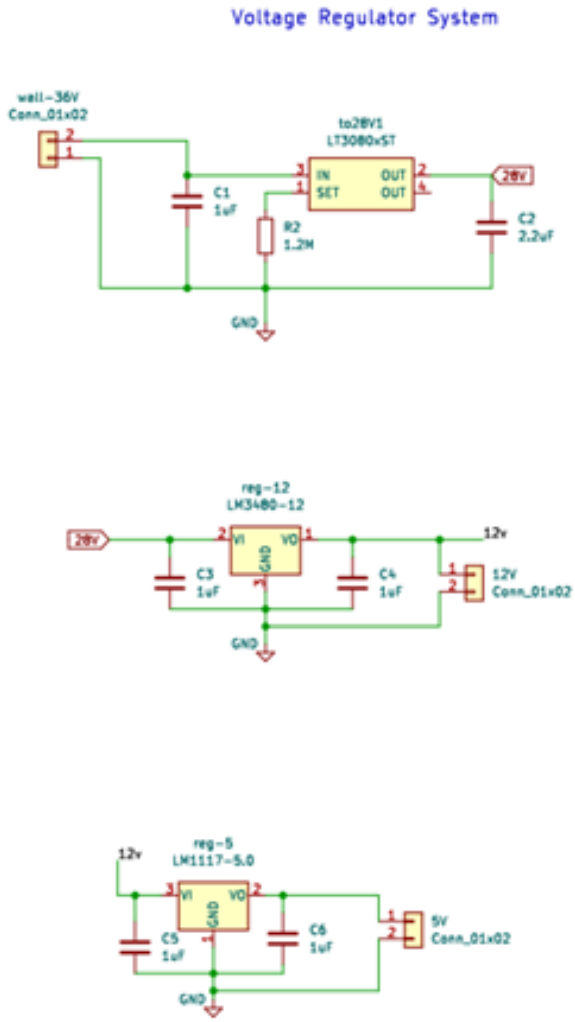

Figure 15: Schematic of the Voltage Regulators

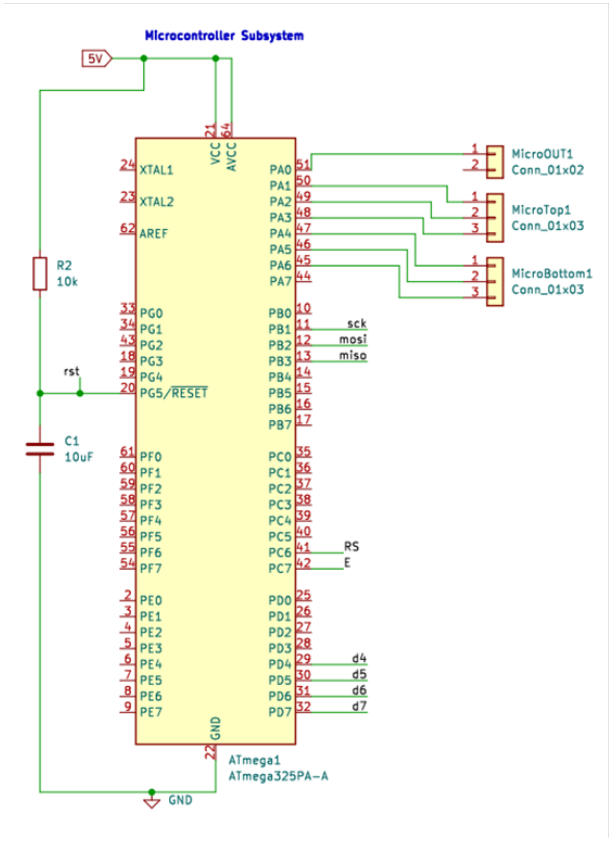

Figure 16: ATmega325PA Schematic and Pin Connections to Other Subsystems

Appendix B: PCB Design

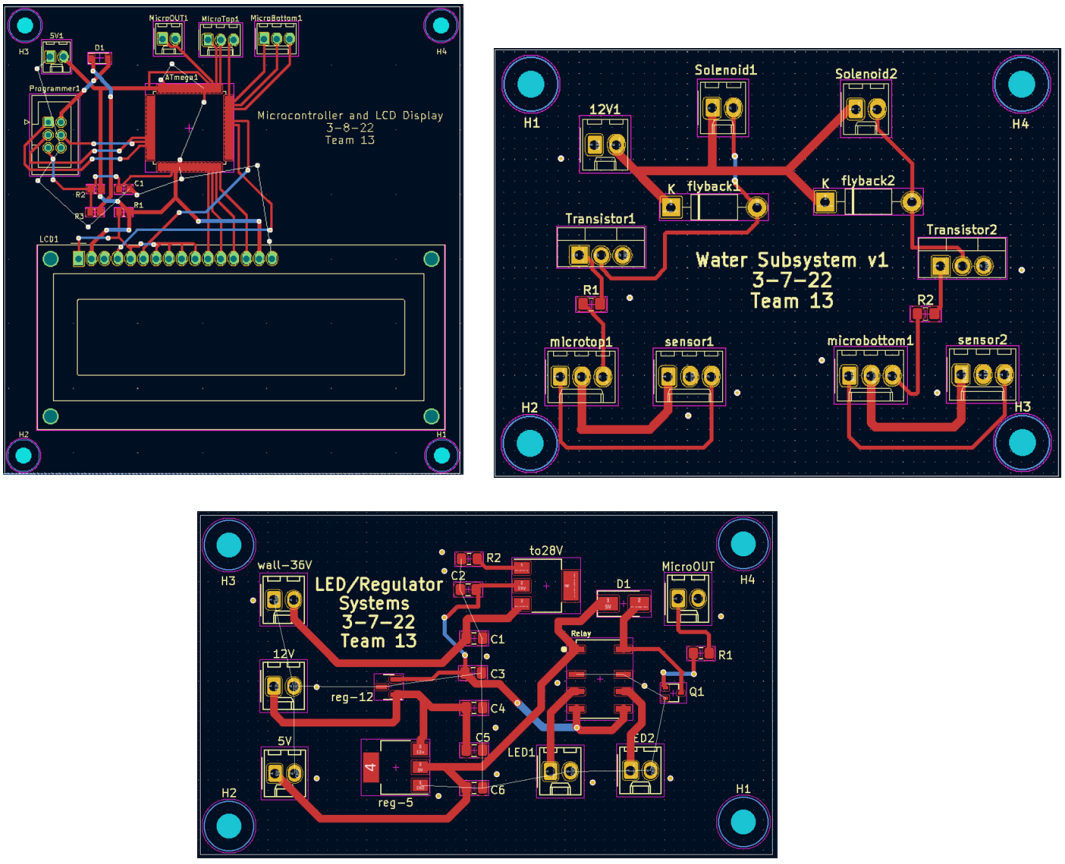

Figure 17: PCB Layouts

We designed three different printed circuit boards to separate voltages. The microcontroller and the LCD are our more sensitive components so we wanted only 5 V to be on the board to avoid accidentally frying any components. We split up the other two boards because we were not sure of where they would be placed since they connected to parts that are spread across our design. This allowed us to worry less about long wires connecting everything into one board.

Appendix C: Requirements and Verification Tables

Table 4: Water Subsystem RV Table

Requirements	**Verification**
1. Water Valve opens when 12 V is applied	1A. Apply 12V directly for 10 minutes and check that valve opens fully and stays open upon application of voltage 1B. Apply 1V of power and check that the valve stays closed
2. Moisture sensor gathers data of the moisture level of the soil when powered on	2A. Supply the sensor with 5V and check the output signal using the serial monitor on arduino IDE 2B. Over saturate soil with water and check that moisture sensor is measuring greater than 500(60% saturated) moisture level
3. Based upon the moisture sensors readings, if the soil is over saturated (500 or more) the watering cycle should skip, if it is under 500, the watering should continue as scheduled.	3A. At a scheduled watering time, create two separate scenarios wherein the moisture detected from the moisture sensor is above the maximum saturation level (500, 60% saturated) and one where it is below the minimum saturation level 3B. In the case where it is saturated greater than 60% (500+/- 5), the microcontroller should not supply voltage to open the valve. 3C. In the case where saturation is less than 60%(500+/- 5), the microcontroller should allow the valve to open
4. The solenoid water valve is able to provide at least 118.3mL (.5 cup) of water in less than 30 minutes	4A. Provide a closed structure that can collect water below the watering tubes 4B. Provide 12V to the solenoid valve and allow the water to drip for 15 min 4C. After 15 min, remove voltage and pour collected water into measuring glass to determine collected water and flow rate
5. When directed by the control system, the valve should stay open for 15 min before being told to close by the control system	5A. In the case where the moisture level is below the saturated amount (500, 60%), using the same code but with modified time variables (on for 60 seconds, off for 2 min until moisture sensor reads again) to demonstrate the ability of the code to open and close the valve based upon time inputs and the moisture sensor valve. Monitor for 6 min or 2 full cycles

Table 5: Lighting Subsystem RV Table

Requirements	**Verification**
1. The LED grow lights turn on when supplied with 36 V	1A.Supply 36 V with the power supply to the light system and check to see that the row of lights is fully illuminated 1B. Lights do not turn on when 25V is applied with the power supply
2.The light system adheres to the designated cycle (12 hours on, 12 hours off in 24 span)	2A. 2A. Using the same code but with modified time variables (on for 1 minute off for 1 minute) to demonstrate the ability of the code to turn on and off the lights based upon time inputs. Monitor for ten minutes or five full cycles.

Table 6: User Interface Subsystem RV Table

Requirements	**Verification**
1.The control system is able to alert the user when the water cycle is skipped or when it will water in 10 minutes	1A. Using the same code with modified time variables 30s alert before watering to demonstrate the ability of the code to notify the user before watering and when a cycle is skipped due to over saturation of soil

Table 7: Power Subsystem RV Table

Requirements	**Verification**
1. Voltage steps down to 12 V +/- 5%	1A. Apply 36V with the power supply to the input of the regulator and probe the output with a multimeter to check that it stepped down the voltage
2. Voltage steps down from 12V to 5V +/- 5%	2A.Apply 12V with the power supply to the input of the regulator and probe the output with a multimeter to check that it stepped down the voltage

Table 8: Microcontroller Subsystem RV Table

Requirements	**Verification**
1. Microcontroller is able to be programmed by the computer	1A. Microchip Studio verifies that the code has been uploaded to the microcontroller

Appendix D: Water Test

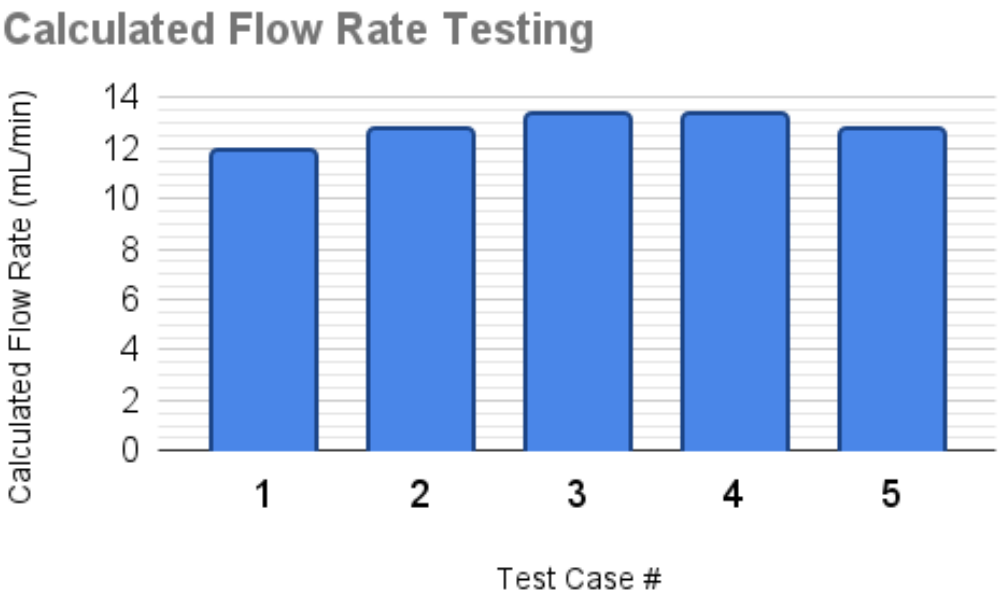

Figure 18: Water Flow Rate Testing

References

[1] "AcuRite Blog - Soil Moisture Guide for Flowers, Plants and Vegetables," www.acurite.com. https://www.acurite.com/blog/soil-moisture-guide-for-plants-and-vegetables.html

[2] "Grow Light Spectrum Explained: Ideal LED Spectrum for Plants," BIOS Lighting, Apr. 29, 2020. https://bioslighting.com/grow-light-spectrum-led-plants/grow-lighting/

[3] Grainger Engineering office of Marketing and Communication. Salary Averages. [Online] Available: https://ece.illinois.edu/admissions/why-ece/salary-averages

[4] "IEEE code of ethics," IEEE, Jun-2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 22-Mar-2022]

[5]"Recommended Practice for the Protection of Electric Equipment in Nuclear Power Generating Stations from Water Hazards," in IEEE Std 833-2005 (Revision of IEEE Std 833-1988) , vol., no., pp.1-20, 19 May 2006, doi: 10.1109/IEEESTD.2006.216286.