Smart Health System for Plants

ECE 445 Final Paper - Fall 2022

Team 13

Rohan Prasad [rprasad3]

Tilak Patel [tpatel80]

Yash Parikh [yparikh2]

Professor: Arne Fliflet

TA: Hojoon Ryu

Abstract

House plants serve a fundamental purpose in many people's lives. Whether it be for decor, ambiance, or oxygen, 66% of American households contain at least one houseplant [2]. A large percentage of these houseplants are often neglected for a variety of reasons. In these times, plants can wither, causing the owners to either purchase a new plant or throw out the old one completely. This is not only a problem of neglect, but also sustainability on a broader scope.

To combat this, we've created the Smart Health System for Plants. Through the use of various sensors, as well as light and water dispensing mechanisms, we created a system that ensures proper care of plants based on a proprietary algorithm developed. Additionally, we enable users to manually control water and light inputs in the case of additional needs. The implementation of this system will be discussed in the remainder of this report.

Contents

1 Introduction

1.1 Problem

There are many families in this world that - for a variety of reasons - are away from home and have plants sitting at home waiting for them to come back and provide water and sunlight. Further, many families love to have plants but don't have time to watch over them due to jobs and busy schedules. In these times, many plants can die out causing the owners to either purchase a new plant or throw out the old one completely. This is not only a problem of neglect, but also sustainability on a broader scope.

1.2 Solution

To solve this problem, we would create a smart health system for plants with a phone app. We would create a plant stand which is built in with our smart health system and provides connectivity to the owner through an app. In our system, we would use different sensors to measure values like humidity, soil moisture and sunlight provided to determine exactly how much water/sunlight the plant will need. We then pump in water from our water reservoir straight to the roots and provide light when needed. Further, through the app, the owner would also be able to provide manual water and artificial light when they want to and see critical values from the sensor module. Overall, this smart health system for plants will provide plants with the most ideal conditions they need to grow and survive and owners will never have to worry about dead plants due to their busy schedules and family vacations. Further, this system will consist of a sensor module, microcontroller, watering system, artificial sunlight system and a phone app.

1.3 Visual Aid

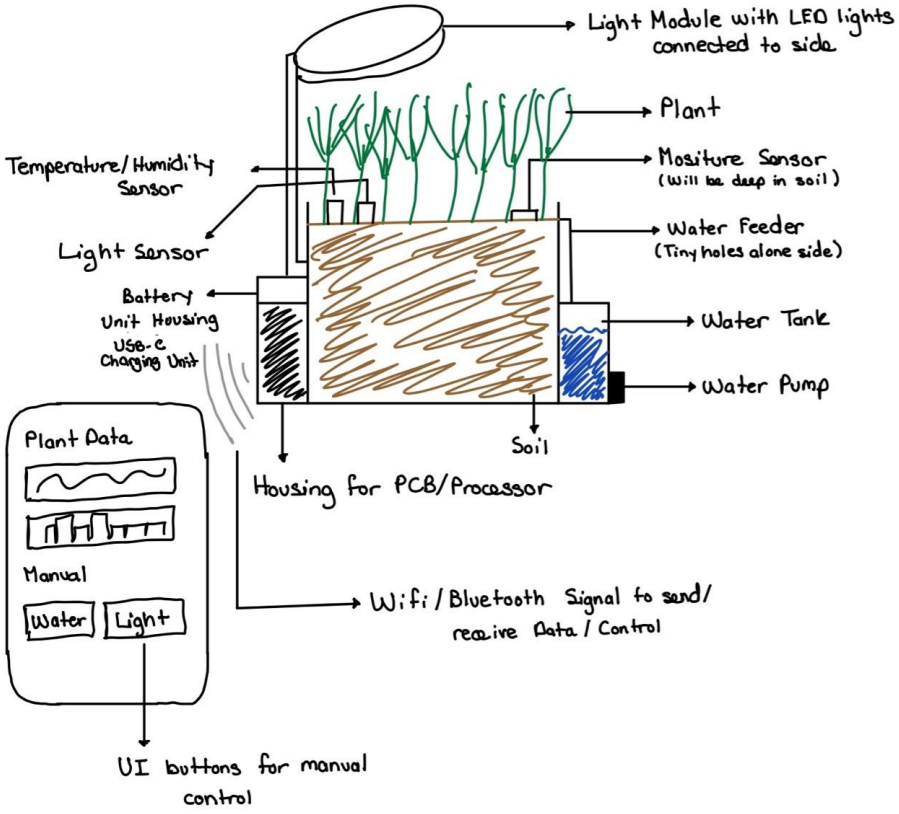

Figure 1: A rough visual aid for plant potter with our smart system

1.4 High Level Requirements

- The system should provide appropriate amount of water when required by the MCU/Microprocessor to do so using our created algorithm
- The system should provide appropriate amount of light when required by the MCU/Microprocessor to do so using our created algorithm
- MCU/Microprocessor is able to communicate with a central system that the phone app can poll from every few minutes and aggregate plant information and metrics to display to the user

1.5 Block Diagram

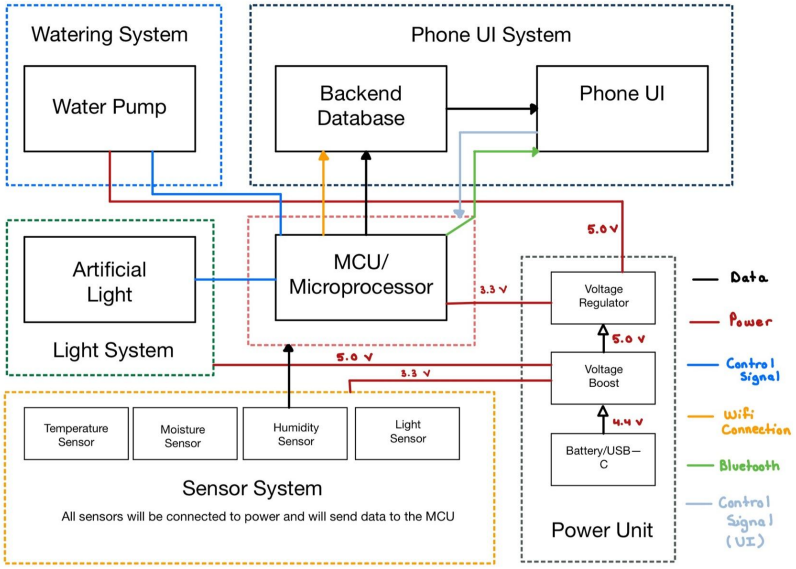

Figure 2: Block diagram of the layout of our system

In our design, we have a total of six subsystems. The subsystems include the power unit, sensor system, light system, MCU, water system and phone UI system. Additionally, the subsystems communicate in our design through a control signal from the MCU or a wifi signal. Figure 2 shows the flow and connections of our subsystems.

2 Design

2.1 MCU

The MCU we chose for this project is the ESP32 because it provides us with internal Wi-Fi and bluetooth capabilities which were used to connect to our phone app. The MCU also provides us with two cores which we needed to run two threads for reading sensor values and controlling the water pump and light module simultaneously. The MCU connects with the sensor system, light system, water system and phone app. Further, it is used to collect data from our sensors and run our created algorithm to actively control the light system and watering system. Furthermore, the MCU uses its built-in Wi-Fi to connect to our remote database to update plant related metrics every two minutes. From there, our phone app polls our remote database every minute which updates the metrics the user will see. Our MCU also uses its built- in bluetooth capabilities to connect to our phone app from which the user can manually provide their plant water and light. In summary, the MCU is used to control our watering system and light system to provide the plant water and light based on our sensor data and our algorithm. The MCU also plays a role in providing critical data to our phone app and polling manual user signals.

The MCU has many connections from our sensors, water pump and light module. Figure 3 shows how each of the modules will be connected to the microprocessor through the I/O ports.

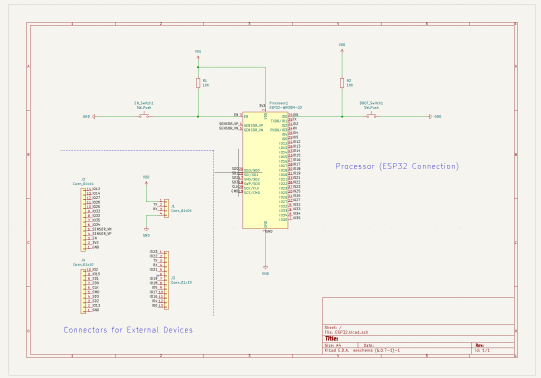

Figure 3: Circuit schematic of my PCB design

2.2 Power Unit

The power unit is critical in our design as it will provide each component their required voltage. To explain, our power system connects to our MCU, sensor system, watering system and light system which require either 5V or 3.3V to operate. Our power unit takes a lithium ion battery pack which provides 4.0V and uses a voltage regulator and voltage boost circuit to create our 3.3V and 5V operating voltages. Furthermore, we also use a USB charging circuit, so the user can plug our system into a wall outlet and charge the battery. Overall, the power unit uses a simple lithium ion battery pack and provides each component their respective operating voltage.

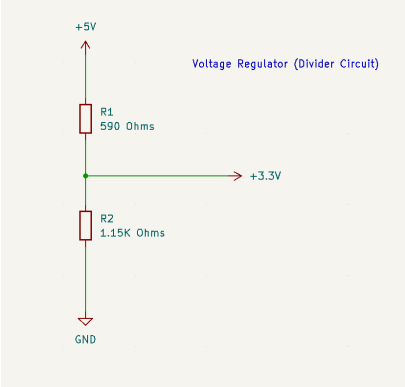

Figure 4: Voltage Regulator Circuit

Figure 4 shows the simple voltage regulator circuit that we implemented to create our 3.3V signal. This is a simple voltage divider circuit that we use to reduce the voltage with 2 resistors. To solve for the output voltage, we set a constant R1 value and then using the input voltage and R1 value, we solve for the variable R2 value.

$$Vout = \frac{R2}{R1 + R2} * Vin \quad (1)$$

Using Equation (1), we made Vin be 5V and R1 be a constant 590 Ohms. After solving Equation (1) with the constants declared, we are able to calculate the value for R2 which will make our output voltage exactly 3.3 V.

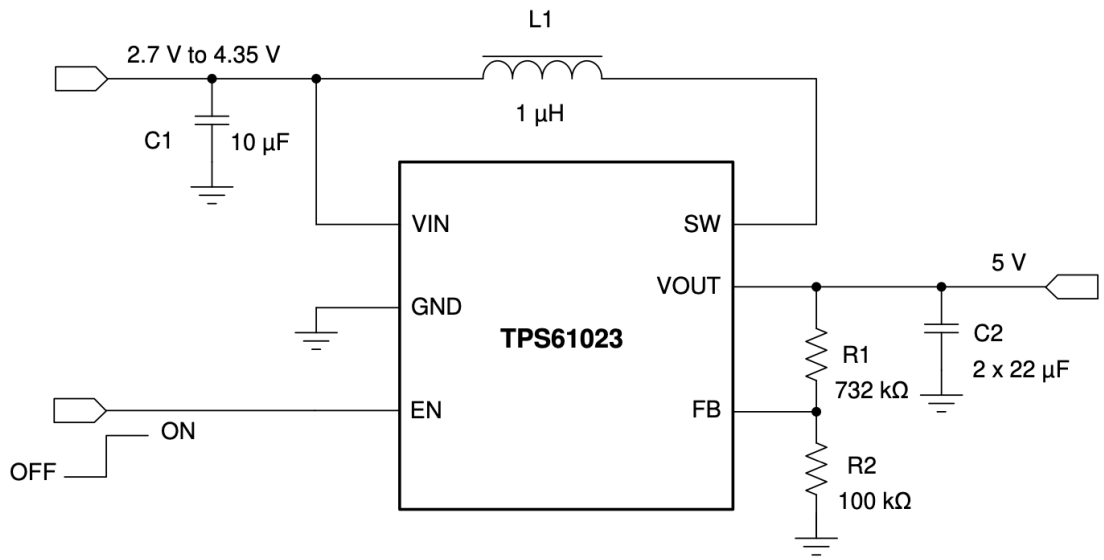

Figure 5: Typical Voltage Boost Circuit using TPS61023 IC to output 5V [1]

Figure 5 shows the voltage boost circuit that we implemented to create our 5V signal. Our voltage boost circuit takes our 4V input from the battery and boosts it up to 5V for our water pump and LED light ring. To create our voltage boost circuit, we use a voltage boost IC chip that connects to some resistors and capacitors to boost our output voltage up to 5V.

2.3 Sensor System

The sensor system consists of a temperature sensor, humidity sensor, moisture sensor and light sensor. The sensors provide data to the processor at an interval of 2 minutes which is processed by our algorithm to trigger either the watering system or light system or both. The sensor system is connected to the power unit which provides all the sensors their 3.3 volt operating voltage.

Temperature & Humidity sensors:

The temperature and humidity sensor (DHT22 Sensor) is used to collect data about the environmental temperature around the plant. The sensor sends the raw data to the processor through an I/O port which converts the data and uses it in our algorithm. We decided to use the DHT22

Sensor because it was the most accurate and provided us with both a temperature and humidity reading.

Light sensor:

The light sensor (TEMT6000 Ambient Light Sensor) gathers data on how much light is provided to the plant. The light sensor provides us with three different light sensing modes including infrared mode, full-spectrum mode and human visible mode. We use the full-spectrum mode which will return a value in the units of lux to our processor. We convert the unit from lux to something meaningful in our algorithm. Lastly, the sensor connects to our processor using an I/O port which collects the data. We decided to use the TEMT6000 Ambient Light Sensor because it was able to provide us with the most accurate reading and was small enough to fit well in our system.

Moisture sensor:

The moisture sensor (Capacitive Soil Sensor) is placed about two inches into the soil and gathers data on moisture levels of the soil. This sensor is critical in our design because we use it to mainly control the watering system. The sensor will be connected to our processor through an I/O port which will collect the data and use it in our algorithm. We decided to use the Capacitive Soil Sensor since it was able to provide us with the most accurate moisture reading since it was placed directly in the soil, and small enough to fit inside our small plant.

In summary, our sensor system will be used by the processor to collect raw data and convert it to be used in our algorithm. The sensor data is critical because it will determine when our processor will turn off and on our watering system and artificial light system.

2.4 Light System

The artificial light system consists of a LED ring light which is attached above the plant. The purpose of this system is to provide light to the plant just like the sun would when there is not enough

sunlight reaching the plant. The LED light will be connected to the power unit which will use our voltage boost circuit to provide the LED light the ideal 5 volt operating voltage. Furthermore, the LED light will be connected to the processor which will signal the light to turn on and off after our algorithm processes the data collected from the sensors. We decided to use a LED light ring as it provides us the ability to code it to any color and brightness. This is needed for our project because different plants will need different light intensity and color. In summary, the LED ring light is a critical component in our design as it is needed for one of our high level requirements and will help provide artificial sunlight to the plant when instructed to do so.

2.5 Water System

The watering system consists of two main components which include the water reservoir and the water pump. The water reservoir is used in our system to store water which feeds the plant around 12 times before requiring a refill. The water pump is used to take the water from our reservoir and provide it to the plant when needed. In our design, we used a 5 volt submersible pump which has one pipe connected. The pipe is connected to the output of the pump which leads water into the plant Furthermore, our water pump is connected and controlled by the processor which signals the pump to turn off and on after our algorithm processes the data collected from the sensors. The water pump is also connected to our power unit which uses a voltage boost circuit to provide our water pump enough voltage needed to operate. In summary, our 5 volt submersible water pump and our water reservoir work together to provide the plant water when instructed to do so by our processor.

2.6 Phone UI

The phone app is used to provide the user a graphic interface to see critical sensor information and trigger the watering system and lighting system manually. The app was created using Flutter and Firebase. Additionally, the app connects to a database and backend over HTTP requests in order to update tables regarding plant information and relay signals that need to occur in the plant's environment. The app polls data from the databases every 2 minutes, and converts the continuous

information stored in the database into meaningful graphical format which users can interpret. The backend is also connected to the ESP32 chip and microprocessor via HTTP, in which requests about plant conditions are sent and stored in the database. In summary, the phone app that's been created to serve this project communicates with the cloud as well as the CPU over HTTP, and displays users features to change manual controls, as well as monitor environment conditions with a strong user experience.

3 Design Verification

3.1 MCU

The MCU was the most important part of our design. The MCU had many key responsibilities including being able to read sensor values, control the water and light systems, provide data to our database for our phone app and be able to read manual signals from users through the phone app.

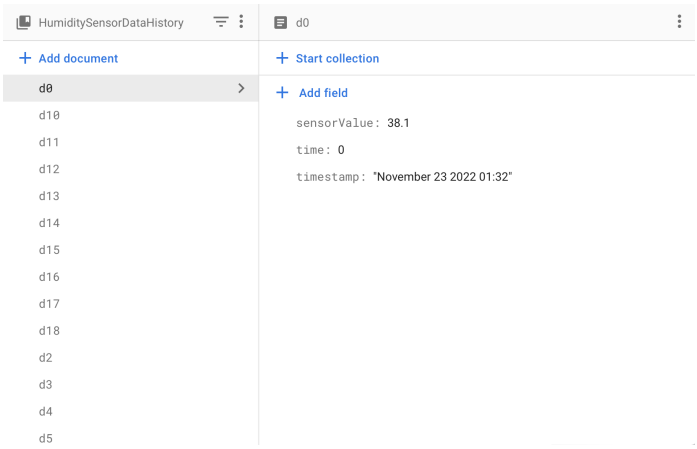

Figure 6: Firebase Database with data sent from MCU

Figure 6 shows a few of the key features our MCU achieved. The MCU is able to send the data it reads from the sensors and send it to our remote database. From this database, the phone app will be able to poll the data and display it to the user. Lastly, using our algorithm, we were able to control our water and light system through signals from the MCU.

3.2 Power Unit

The power unit had two key responsibilities which included providing 3.3V and 5V to the components in the rest of the system. To explain, some of the components in our design required 5V to operate and other components required 3.3V to operate. In order to verify we can get 3.3V and 5V

from the voltage boost and voltage regulator circuits, we connected our battery to our voltage regulator and voltage boost circuits and then used a voltmeter to see the exact output voltage from each circuit.

Figure 7: Voltage from Battery

Figure 7 shows the output voltage from the battery. This is our expected voltage which we will pass into our voltage boost and voltage regulator circuits to create our 3.3V and 5V signals.

Figure 8: Voltage Regulator Output Voltage

Figure 7 shows the output voltage from the voltage regulator circuit we designed. We can see that the output is 3.3V which means that we can provide this to any component that requires 3.3V in our design to operate.

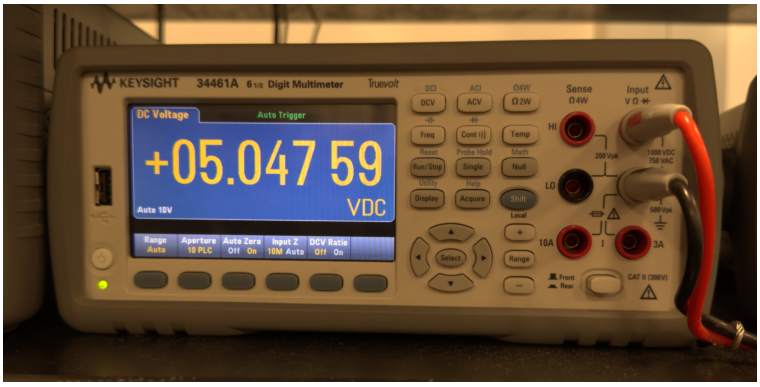

Figure 9: Voltage Boost Output Voltage

Figure 8 shows the output voltage from the voltage boost circuit we designed. We can see that the output is 5V which means that we can provide this to any component that requires 5V in our design to operate.

3.3 Sensor System

The DHT22 sensor reads both temperature and humidity, and provides values of 73.3°F and 34.6% respectively. The Capacitive Soil Moisture Sensor provided a value of 99.5% moisture, but this was because we had just filled the plant with water. Finally, the TEMT6000 Light Sensor provides a reading of 4567 Lux for the environment. Provided below is a table comparing our sensor values to the actual values given at that time to demonstrate the sensors' tolerance:

Table 1: Sensor Readings

Sensor Name	Sensor Reading	Actual Reading	Tolerance (% Error)
DHT22- Temperature	73.3°	73.7°	0.54%
DHT22- Humidity	34.6%	34.9%	0.86%
TEMT6000 Light Sensor	4567 Lux	4602 Lux	0.76%
Capacitive Soil Moisture	99.5%	99.3%	0.20%

As Table 1 shows, all of our sensors were within 1% of the actual readings, which is significantly better than the 5% tolerance we specified in our R&V tables. We were able to ensure that our system has the highest possible accuracy by ensuring that our sensors would be able to feed in the most accurate readings to our algorithm. Accurate sensor readings also helped us make sure that we maintained our plant in the most optimal conditions possible, with research showing that the plant should be kept in an environment of 65°F-80°F as well as anywhere between 30%-40% humidity and relatively moist soil conditions.

3.4 Light System

The Light System was able to provide light to plants, both using our 'Smart Control' feature through the app, and manually using a button on our app. We were also able to control the specific intensity and color of the light, and for our specific plant, we used the RGB values of 253, 243, 198 respectively to replicate the color of 'natural light'.

3.5 Water System

The Water System was able to provide water to plants, also both using our 'Smart Control' feature through the app, and manually using a button on our app. We were able to control how much water went into the plant by limiting how long our water pump ran for, and this way, we ensured that our plant wasn't being either overwatered or underwatered. For our specific plant, we set the pump to run for 0.5 seconds and for the pump to run once every 4 days since our plant was relatively low maintenance.

3.6 Phone UI

The Phone UI was able to provide a succinct user interface and experience for users of our product to interact with the ecosystem as well as view plant metrics as fit. We were able to control how

much water was delivered to the plant by manually setting flags within the app to trigger the water pump. Additionally, we're able to toggle the light in the environment via a slider that exists within the Phone UI. Another feature present in our app is the ability to see plant environment conditions as a chart over the prior 20 minutes. This is done by making HTTP requests to our database to determine the last 20 minutes of data, and compiling this as continuous data to display in the phone UI.

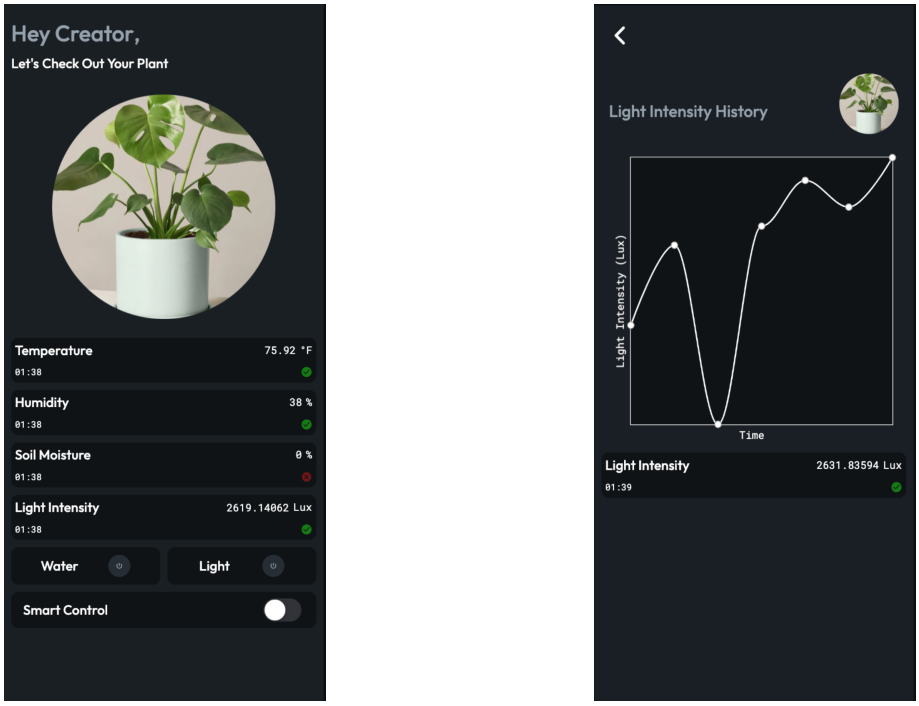

Figure 10: Phone UI Screenshots

As seen in the above images, we successfully accomplished these features in a robust UI and provide users with a functional experience.

4 Costs and Schedule

4.1 Parts

Parts Table in Appendix A

4.2 Labor

The average salary for Computer Engineering graduates from UIUC is around $105,000 [3]. Using this information, our hourly rate comes out to about $54.68. In out project, we can expect a salary of $54.68/hour * 2.5 hours a day * 60 days = $8202 per group member. In total, salaries for members will be about $24606 in labor cost.

4.3 Total Cost

The table (1) in Appendix A illustrates that the total price before shipping and taxes is $107.73. Assuming shipping and taxes adds 10% to this cost, this adds another $10.77. With a salary of $24,606 and parts cost of $118.5, the total cost comes out to around **$24,724.5**.

4.4 Schedule

See Schedule in Appendix B

5 Conclusion

5.1 Accomplishments

With the completion of this project, many successes and accomplishments were seen. The primary success of our project is the integration across the entire stack. From software to hardware, our system is succinct and functional. Starting in the cloud, we're able to communicate from microprocessor to phone UI over HTTP, and subsequently affect the plants ecosystem with water and sunlight as needed. An additional success in our project is the ability to make plant decisions based on metrics identified by the various sensors located around our system. Depending on if any of our metrics were suboptimal for plant growth, our system is able to water the plant or provide light. We plan on extending this feature in the future by allowing users to input parameters such as plant type, watering needs, and light needs, and incorporate those into our algorithm. Finally, another major achievement in the development of our project is the visualization shown in our phone UI. With the addition of charts to provide users meaningful information on the health and environment of their plant, we're able to boost the user experience of our product to something that would actually go to market. This end to end implementation of our project was a huge success, yet still paves the way for future extensions of the project.

5.2 Uncertainties

With the implementation of any project, along come some uncertainties. Throughout planning and design, our group aimed to make as many components modular so we could avoid catastrophic failures down the road. This allowed us to avoid any uncertainties that came along with project design initially. However, there were some uncertainties that arose while developing and debugging our project. First, there was some uncertainty around the placement of the light sensor. Would it capture the light being dispensed to the plant properly? Would any external factors affect the readings accounted for in the algorithm? These were questions that our team asked and addressed when determining the placement of the sensor. The results of these discussions and future scope of the

position are discussed further below in the future work portion of this report. Additionally, we had some uncertainty in the amount of light being dispensed by the system. Would the light from an LED (multiple LEDs) be enough to provide for a plant? Was the light emitted by the LED optimal for plant growth? Again, we addressed these questions via research on optimal growing conditions for plants, and how different shades of light would affect plant growth. Beyond environmental concerns, our team had a few concerns surrounding the PCB design and components, but these were discussed and squared away in the design process of iteration, and the correct decisions were made at that time.

5.3 Ethical Considerations

When considering ethics and safety of the product, we must consider potential pitfalls and safety concerns in the products being placed in an individual's living space. We must consider the components that exist in the system - primarily the microcontroller, sensors, and watering/sunlight mechanisms. Although there isn't a huge safety concern at first glance, potentially malfunctioning or overheating of the microcontroller or sunlight systems may impose a risk of fire depending on where the device is located. In order to "avoid harm", we plan to ensure that elements are properly insulated, and don't impose a risk of mal-action in the case of misuse or malfunction.

On the note of ethics, our system simply attempts to aid families and plant owners maintain plants in a more convenient manner - this doesn't pose an ethical risk in regards to a broad objective. In accordance with the IEEE Code of Ethics Section 7.8.I.1, we will strive to implement an ethical design and follow all of the sustainable practices possible, while not endangering the public or the environment [4]. Additionally as stated in section 7.8.I.2, we plan to improve society's understanding and educate them of our project's capabilities, all while doing so in a safe and ethical manner [4]. As per section 7.8.I.5 , we plan to accept constructive criticism regarding our work, and will correct any and all errors regarding our project and its overall design and implementation [4]. This feedback and criticism will be provided to us by our professors, TAs, and even amongst ourselves. Furthermore, we plan to only perform tasks for which we are qualified for after the adequate technical training, as stated

in section 7.8.I.6 [4]. To fulfill this, we are required to perform activities such as the 'Lab Safety Training', 'CAD Assignment', and 'Soldering Assignment' so that we are fully aware and prepared to use tools such as CAD to assist with building PCBs, and implement proper soldering practices to ensure no one is harmed. Finally, according to section 7.8.III.10, we will make sure that as a group, all three of us will check in on each other to make sure that we uphold the IEEE Code of Ethics at all times and encourage ethical and safe behavior and practices at all times [4].

5.4 Future Scope

Although we've made considerable progress on the project, there's still room for improvement in future work. In the design of our project, we only included one light sensor. This sensor was positioned at the far corner of our plant stand (reasonably far from the light source). Because of this placement, our light source wasn't accurately picked up at times. In order to combat this, in the future we aim to incorporate multiple light sensors surrounding the light source, and triangulate sensor readings to come up with an accurate measurement. Another area of improvement that we aim to address is to provide a more robust and comprehensive user experience for users of our phone UI. This entails adding more granular manual features, as well as displaying more granular data in charts for users to analyze. As the needs of plants become more complex, we also aim to enable manual inputs for metrics such as plant type and watering needs to support a wider range of plants in the system. Finally, another area of future work we aim to accomplish is to scale the plant ecosystem and sensors to support multiple plants in a cohesive manner. This involves adding one of each sensor per plant, or triangulating readings from having multiple sensors per plant. Additionally, we'd add multiple reservoirs and light sources to encompass any additional plants that we're looking to add to the system. Overally, these are three areas of future work that we're looking to accomplish after the culmination of this part of the project, and would pave the way for developing an industry-ready solution to our identified problem.

References

[1] “TPS61023 3.7-a boost converter with 0.5-V Ultra-low input voltage ...” [Online]. Available: https://www.ti.com/lit/ds/symlink/tps61023.pdf?ts=1596144931979&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS61023. [Accessed: 28-Oct-2022].

[2] “Houseplant statistics in 2022 (incl.. Covid & Millennials),” *Garden Pals*, 09-May-2022. [Online]. Available: https://gardenpals.com/houseplant-statistics/. [Accessed: 05-Dec-2022].

[3] Grainger Engineering Office of Marketing and Communications, “Computer Engineering,” *The Grainger College of Engineering | UIUC*. [Online]. Available: https://grainger.illinois.edu/academics/undergraduate/majors-and-minors/computer-engineering. [Accessed: 28-Sep-2022].

[4] IEEE Board of Directors. “IEEE Code of Ethics.” IEEE Code of Policies, Section 7 - Professional Activities (Part A - IEEE Policies). [Online]. June 2020. https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 15-Sep-2022].

Appendix A Parts List

Table 2: Parts Cost

Description	Manufacturer	Quantity	Total Price	Link
DH22 Temperature and Humidity Sensor	Adafruit	1	$4.50	Link
Stemma Soil Sensor	Adafruit	1	$7.50	Link
Grove - Digital Light Sensor	Grove	1	$6.60	Link
DC Water Pump	Gikfun	1 (3 Pack)	$13.48	Link
Ring Lamp Light 24 Bits	Adafruit	1	$9.59	Link
ESP32 Processor	Espressif Systems	1	$3.69	Link
USB to Uart	Amazon	1	$8.29	Link
USB Charging IC	Adafruit	1	$6.95	Link
3.7 V Lithium Ion Battery	Adafruit	1	$29.95	Link
0.1uF Capacitors	Adafruit	1 (10 Pack)	$1.95	Link
Plastic Water Container	Pioneer Plastics	1	$8.82	Link
10K Resistor	Adafruit	1 (25 pack)	$0.75	Link
3.3 V Regulator	SparkFun	1	$2.10	Link
5 V Boost	Adafruit	1	$3.56	Link

Appendix B Schedule

Table 3: Schedule

Dates	Tilak	Rohan	Yash
Week of 9/26	• Finish Design Document & start PCB Design	• Finish Design Document & start PCB Design	• Finish Design Document & start PCB Design
Week of 10/03	• Finish PCB design and seek instructor approval	• Consult with TA and start ordering initial parts	• Finish PCB design and seek instructor approval
Week of 10/10	• Start working on code	• Order any remaining parts • Start working on code	• Order any remaining parts
Week of 10/17	• Continue working on code • If PCB and parts come in, start assembling parts and soldering PCB	• Continue working on code	• Continue working on code • If PCB and parts come in, start assembling parts and soldering PCB
Week of 10/24	• Assemble any remaining parts	• Start initial code testing	• Start initial code testing
Week of 10/31	• Start basic testing of other subsystems • Order PCB if necessary	• Continue refining code	• Continue refining code and start basic testing of other subsystems • Order PCB if necessary
Week of 11/07	• Continue to test and refine all subsystems and ensure they	• Continue to test and refine all subsystems and ensure they	• Continue to test and refine all subsystems and ensure they

	work together • Make any small changes necessary	work together • Make any small changes necessary	work together • Make any small changes necessary
Week of 11/14	• Prepare for and give Mock Demo • Make minor adjustments wherever possible	• Prepare for and give Mock Demo • Make minor adjustments wherever possible	• Prepare for and give Mock Demo • Make minor adjustments wherever possible
Week of 11/21	• Fall Break	• Fall Break	• Fall Break
Week of 11/28	• Prepare for and give Final Demo	• Prepare for and give Final Demo	• Prepare for and give Final Demo
Week of 12/05	• Prepare for and give Final Presentations • Work on and submit Final Paper	• Prepare for and give Final Presentations • Work on and submit Final Paper	• Prepare for and give Final Presentations • Work on and submit Final Paper

Appendix C PCB

This appendix shows our PCB design and renderings of our PCB. It also shows any issues we ran into for our PCB

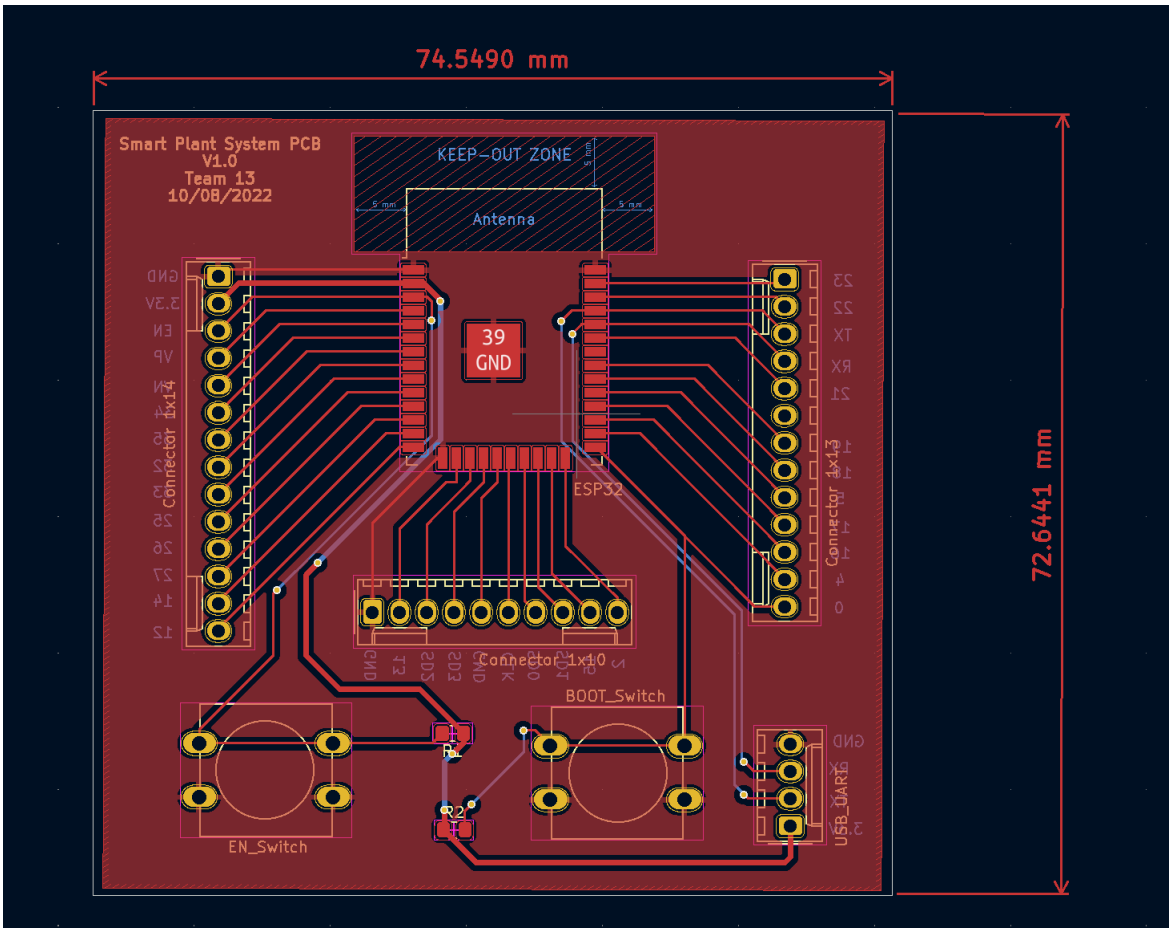

Figure 11: PCB layout front

Figure 11 shows our design for our PCB. We created a simple breakout board for our ESP32 chip and connected wires to each connector spot.

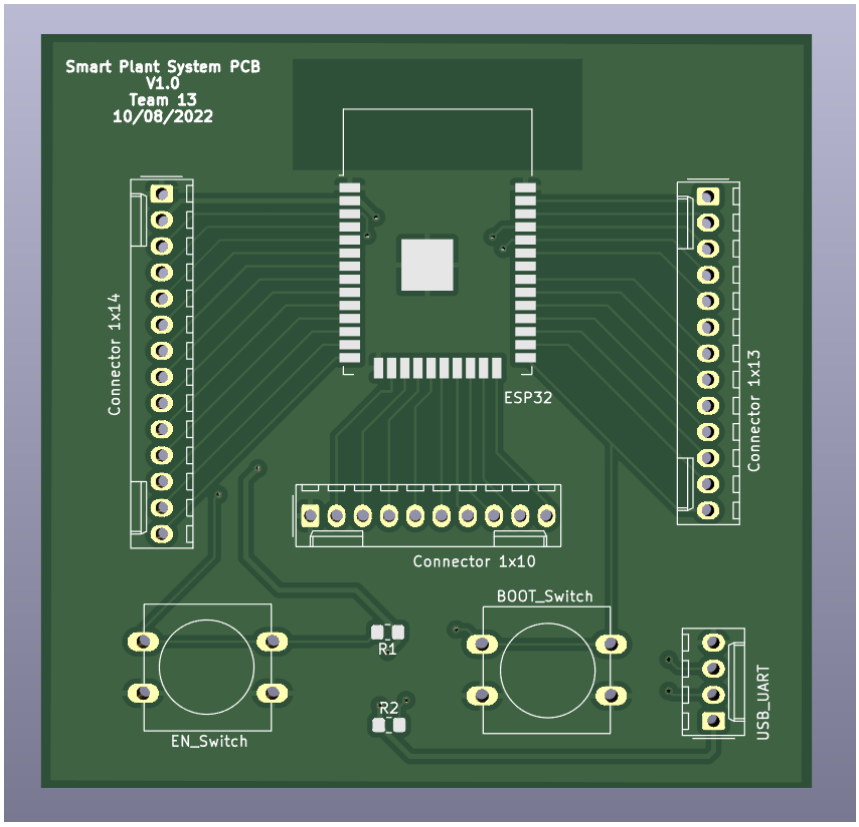

Figure 12: 3D PCB Render

Figure 12 shows the 3D render for our PCB design. After soldering our PCB, we ran into two main issues. The first issue was difficulty soldering the ESP32 chip on with the soldering irons in the lab. It was difficult to solder the chip without the connections touching. Further, we connected the button incorrectly. The connections for the buttons should be diagonal and not across from each other. If we had more time, we would have made these corrections and created a fully functional PCB.

Appendix D R&V Tables

Table 4: R&V Table for MCU

Requirement	*Verification*
The processor must be able to receive manual signals from the phone app and override the algorithm to run the manual signal	To test that the processor can receive the manual signal from the phone app: 1. Power on the processor and start running the coded algorithm 2. Connect the phone app to the processor via bluetooth 3. Press the manual watering or lighting button in the app and verify we can see water pump start or LED light ring turn on
The processor must be able to transfer processed sensor data into our remote backend service via wifi connectivity every 2 minutes	To test that the processor can transfer the data into the backend service 1. Power on the processor and verify it is connected to the wifi 2. Upload the algorithm and start running the program 3. Verify in backend service that POST calls are occurring every 2 minutes
The processor must be able to collect data from the sensors every minute and process it into the algorithm to trigger the watering system or artificial lighting system	To test that the processor is collecting data every minute from the sensors 1. Power on the processor and upload the algorithm 2. Start running the program 3. Manually simulate sensor values using a power supply or by using the sensor directly 4. Verify if the sensor values are in the coded range for the watering system then the system is turned on and off after proper calculated run time 5. Repeat same procedure for artificial lighting system

Table 5: R&V Table for Sensor System

Requirement	*Verification*
The sensors should be able to provide new data to the processor every 2 minutes	To test that the sensors are providing new data every 2 minutes 1. Power on the processor and verify all sensors are connected 2. Manually change sensor values every 2 minutes 3. Verify the processor is able to receive the new sensor values using print statements
The sensors are able to provide accurate data with a 5% tolerance	To test that the sensors are providing accurate data 1. Power on the processor and verify all sensors are connected 2. Create a controlled testing environment for each sensor (i.e controlled temperature/humidity room for temperature/humidity sensor, controlled soil with equal amounts of water for moisture sensor, different lighting environments for light sensor) 3. Verify sensor is able to provide accurate values for similar control environments with a ±5% tolerance

Table 6: R&V Table for Water System

Requirement	*Verification*
The water reservoir must be able to hold enough water to feed the plant three-four times before needing a refill	To test that the water reservoir can hold enough water to feed the plant three-four time: 1. Power on the processor and set the signal to the water pump to high 2. Repeat this three-four times and verify we do not run out of water before the third or fourth time
The water pump must be able to turn on and off	To test that the water pump turns on and off for

Requirement	*Verification*
when instructed by the processor for the calculated amount of time from our algorithm	a specific amount of time: 1. Power on the processor then upload and run the algorithm 2. Code the algorithm to turn the water pump on for a specific amount of time 3. Manual get data on the amount of time the water pump turned on using a stopwatch 4. Verify the manually collect time and the coded time are within ±1%

Table 7: R&V Table for Light System

Requirement	*Verification*
The LED light ring must be able to turn on and off when instructed by the processor for the calculated amount of time from our algorithm	To test that the LED ring light turns on and off for a specific amount of time: 1. Power on the processor then upload and run the algorithm 2. Code the algorithm to turn the LED ring light on for a specific amount of time 3. Manually get data on the amount of time the LED ring light turned on using a stopwatch 4. Verify the manually collected time and the coded time are within ±1%

Table 8: R&V Table for Power System

Requirement	*Verification*
The power unit should be able to provide 5 volts to components that require 5 volts operating voltage	To test that the power unit is providing 5 volts to the component that require 5 volts accurately: 1. Power on the Power Unit and ensure that the 3.7 volt lithium ion batteries are functioning 2. Verify that the voltage coming out of the

Requirement	*Verification*
	voltage boost circuit is 5 volts with a ±0.1 volt tolerance using a voltmeter
The power unit should be able to provide 3.3 volts to components that require 3.3 volts operating voltage	To test that the power unit is providing 3.3 volts to the component that require 3.3 volts accurately: 1. Power on the Power Unit and ensure that the 3.7 volt lithium ion batteries are functioning 2. Verify that the voltage coming out of the voltage boost circuit is 3.3 volts with a ±0.1 volt tolerance using a voltmeter

Table 9: R&V Table for Phone UI

Requirement	*Verification*
The phone app should be able to show users sensor information and provide them with critical and insightful data about the health of their plants.	To test that the phone app is conveying accurate and up-to-date information to the user: 1. Launch the phone app and connect to the device using Bluetooth. 2. Attempt to trigger the watering and lighting systems through the app. 3. Using Wifi, we should be able to see a graphical representation regarding plant and soil health. 4. Ensure that the physical sensor values are corresponding to the values being displayed on the app, with an accurate timestamp of when the data was collected.

Appendix E Algorithm Flow Chart

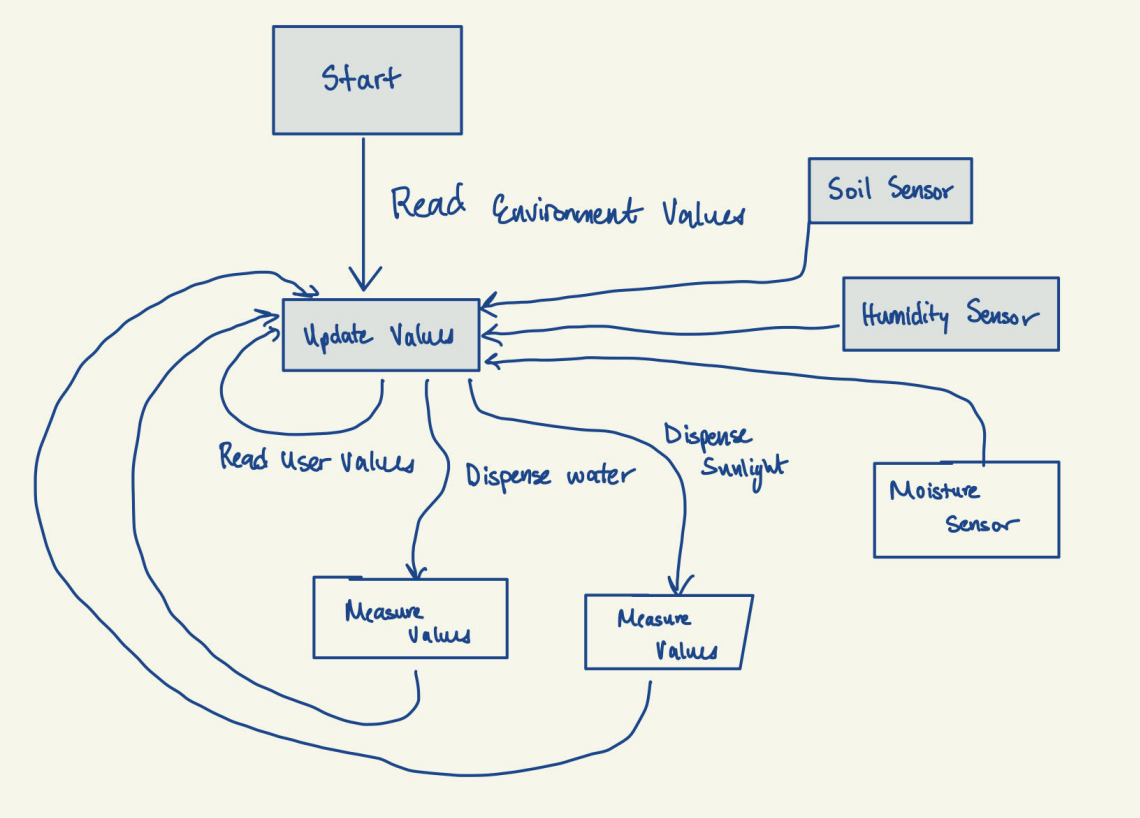

Figure 13: Algorithm Flow Chart used to make decisions