SOILMATE

By

Emma Hoeger

Ysabella Lucero

Sigrior Vauhkonen

Final Report for ECE 445, Senior Design, Spring 2026

TA: Zhuchen Shao

06, May 2026

Project No. 32

Abstract

This report details the innerworkings of a newly designed plant notification system. It utilizes sensors to gather information about the plant's environment and uses an app connected through Bluetooth to send suggestions on how to improve the environment to the user. This report also goes in depth about the design and verification processes performed to gather accurate temperature, humidity, light, and soil moisture sensor readings to send reliable plant care notifications in combination with weather API (Application Programming Interface) data from the plant's origin. Finally, it concludes with the project's societal impacts, achievements, and future improvements.

Table of Contents

1. Introduction

1.1 Problem

Many houseplant owners struggle to take proper care of their plants. In fact, “60 percent of millennials said that what worries them most is ensuring plants get the proper amount of sunlight. Other anxieties include watering (56 percent of respondents) [and] keeping the plants alive (48 percent)” (Castillo). It can be difficult to keep track of when to water them and where to keep them, based on the species of plant and stage of life. Since all plants require water at different frequencies and amounts, it’s also easy to forget to water the plants on time and meet their different schedules. This causes many plants to die, and many owners become disheartened and avoid taking care of plants altogether.

1.2 Solution

Our solution is to create a notification system to inform houseplant owners of when they should water their different plants. It will also monitor and notify the owner of the conditions of the plant based on various sensors. This will be done by creating an app that the owner can download onto their phone, where they will be able to enter their type of plant. There have been many apps created to act as a reminder to water plants; however, many of them rely on a schedule rather than live data gathered from the plant. Also, those that do have live data from the plant do not track the weather. Our app will use the location of where a plant is originally from and use the weather patterns in that area to determine when it should be watered (i.e. when it’s raining). In addition, there will be a soil moisture sensor, humidity sensor, light sensor, and temperature sensor. The soil moisture sensor acts as a failsafe to prevent dangerously dry or damp conditions by alerting the owner to water the plant if the moisture is very low and prevent overwatering of the plant if the moisture is very high. The humidity sensor will alert the owner when the humidity is dangerously too high or too low for the plant, which is especially useful for tropical plants in a non-tropical environment (many houseplants are of a tropical background). The temperature sensor will alert the owner when the room temperature is not in the optimal range for the specific plant. The light sensor will ensure that the plant is in a place with the correct light exposure and will inform the owner when it is not.

With the integration of software and hardware subsystems, this effective plant notification system will make taking care of houseplants much more convenient for both beginners and experienced plant owners. Beginner plant owners will find it easier to learn about and keep track of the demands of their plants, preventing the most common mistakes that result in their death. Many experienced plant owners can have upwards of 20 plants, and this notification system would make it much simpler to keep track of when to water them all and meet their different needs.

1.3 High-Level Requirements

1) The sensors must be able to gather accurate and reliable data about the plant’s surrounding environment, including measuring the light, temperature, humidity, and soil moisture.
2) The microcontroller must be able to gather data from sensors via I2C (Inter-Integrated Circuit) communication and transmit it via Bluetooth to the app.

3) The mobile app must generate sensible suggestions for the user about the care of the plant, utilizing the sensor data and weather API data.

2 Design

2.1 Block Diagram

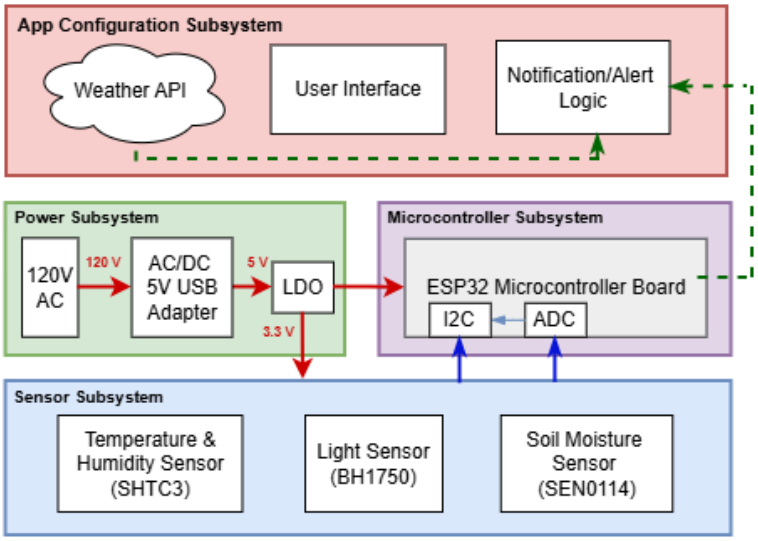

Figure 2: Block Diagram of Soilmate

The block diagram depicts our four subsystems- the sensor subsystem, microcontroller subsystem, power subsystem, and app configuration subsystem. The sensor subsystem consists of three sensors- one that measures both temperature and humidity, a soil moisture sensor, and a light sensor. The readings from these sensors are sent to the microcontroller subsystem via the I2C bus. The microcontroller packages the data received and sends it to the app configuration system via Bluetooth. The app configuration system is what the user interacts with- the sensor data is displayed, and it decides on care notifications to send to the user. It also communicates with a weather API, which is also used in the decision logic. Finally, the power subsystem provides power to the microcontroller and sensor subsystems with a wall adaptor.

2.2 Physical Design

Our physical design consists of a clear waterproof box where our PCB will be kept. This is to ensure it remains protected from potential water spills that an owner may cause when watering the plant. It must be placed near the plant because our soil moisture sensor will be connected with a cable on our PCB, and the sensor itself will be inserted into the plant. We will also have a power cable which is plugged into the wall and then plugged into our board. Since these cords must always be plugged into the PCB, we cut holes in our box so that they are able to go through. We cut a circular hole for the power cord on the side of the lid with a 10mm radius. We cut a rectangular hole for the soil moisture sensor connector at the top of the lid. We initially wanted the moisture sensor cutout to be on the side of the lid; however, this would pose a problem for the wires as they'd have to bend at a right angle through the hole, which could risk signal integrity. We also included vents on two sides of the box to

ensure that the ambient temperature and humidity of the surrounding environment could easily reach our PCB. The safety of this design will be discussed in Section 5 of this report. Below is an image of our enclosure with the PCB inside.

Figure 3: Top View of Enclosure

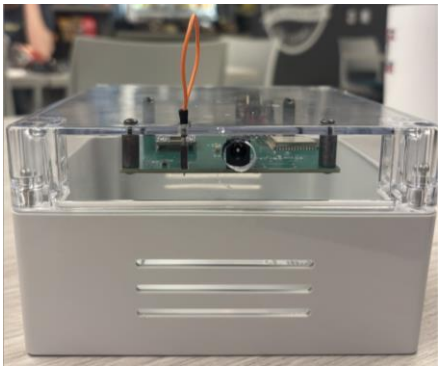

Figure 4: Side View of Enclosure

2.3 Subsystem and Requirements

2.3.1 Power Subsystem

The power subsystem will deliver power to the sensors and microcontroller systems. The microcontroller, temperature, humidity, soil moisture, and light sensors require 3.3V. The 3.3V will come from an external LDO, and we will use a 5V wall adaptor to convert the 120V AC from the wall to 5V. The LDO we will be using is the AP2112K-3.3TRG1 (SOT25), which has a fixed output of 3.3V. Two 1uF capacitors are used to stabilize the voltage signals coming into the LDO from the wall adapter and out of the LDO to the rest of the board. The pinout for the barrel jack connector, as shown below in Figure 5, shows the detection pin and ground pin connected since, after testing, the detection pin was acting as ground. This correction allowed the rest of the board to be grounded and receive the correct voltages.

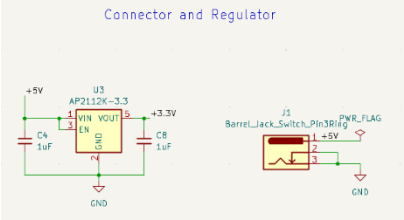

Figure 5: Power Subsystem Schematic

2.3.3 Microcontroller Subsystem

We must be able to blend our app configuration with our sensor subsystem to send an alert. We can do this by using the ESP32-S3-WROOM microcontroller. It will pull data from our sensors using I2C communication and will use BLE (Bluetooth Low Energy) to transfer the data to our app. It is cost-effective and has low power consumption, which will make it easy to integrate with our design. Furthermore, our group has experience with this specific microcontroller, so we are confident in its capabilities. Below is a schematic of our microcontroller.

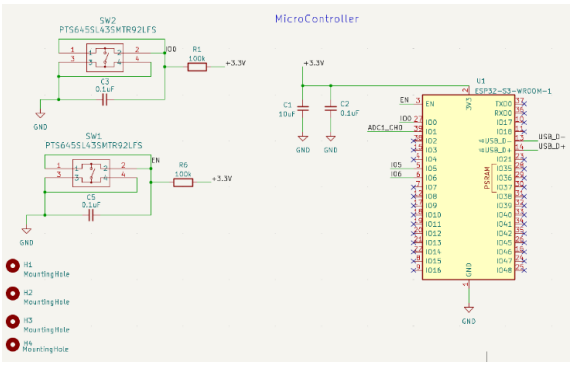

Figure 6: Microcontroller Subsystem Schematic

This microcontroller also has many GPIO pins that are available to be used, such as I2C, camera interface, ADC (Analog-to-Digital), and more. We will be taking advantage of the I2C and ADC GPIOS that are available on the chip. Our I2C consists of two pins, SDA (Serial Data) and SCL (Serial Clock). The intricacies of how I2C operates will be discussed in Section 2.3.4. Looking at Figure 6, we connected our SDA line to pin IO6 on the microcontroller and our SCL line to pin IO5. There are two ADC lines to choose from with channels ranging from 0-7. We chose ADC1_CH0 because it was easy to connect to via a jumper wire and it was not an important strapping pin, unlike some of the other channels. We also included decoupling capacitors on the power line of the microcontroller because we want to ensure that the input power will remain stable when the microcontroller is being programmed. The 10uF capacitor ensures this stability since it can provide energy when low and the 0.1uF in parallel ensures there is low noise under normal operating conditions. We will also include a USB-C connection to the microcontroller, connecting to USBD+ and USBD− pins as seen in Figure 7.

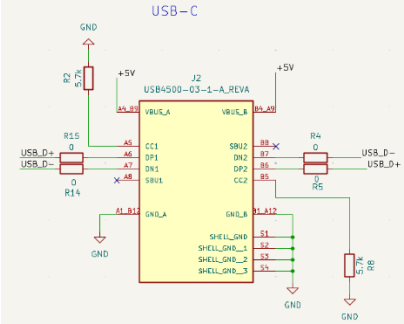

Figure 7: USB-C Schematic

It is especially convenient to use USB-C rather than USB-UART since the microcontroller already contains an integrated USB peripheral and bootloader, which makes programming easier. The CC1 and CC2 pins must be connected to ground through a 5.1kohm resistor as per USB-C standard. These pins are crucial as they detect cable orientation, identify which device is the host or client, negotiate power levels, and support high-speed data modes. The SBU1 and SBU2 pins are less important as they are used for auxiliary signals, which we will not need since we are only concerned with data transfer. We initially designed this USB-C component to be on the PCB, however due to using an incorrect footprint of the USB-C connecter, it was unable to fit on the board. So, we had to purchase USB-C breakout boards and directly jumper wire the USBD+ and USBD- pins to the microcontroller from the breakout board to program it. The VBUS pins on the USB-C will be connected through a jumper wire to our barrel jack that is holding 5V, as per USB-C standard.

To program, we must be able to connect GPIO0 and Enable pin to ground on the microcontroller. When pressing both buttons down, this puts the microcontroller into download mode which allows us to install our firmware. This is why our design includes two buttons, SW1 and SW2 in Figure 6. The SW1 is for our enable pin which allows us to easily reset the microcontroller when we want to erase programmed firmware. SW2 is for the GPIO0 pin which is only for putting microcontroller in download mode. We included a small delay with both the switches to ensure that the power is stable before being able to program the microcontroller. We used Equation 1 to determine the value of capacitors for our buttons. We wanted at least a 10ms delay so that it would have enough time to stabilize but not too much to the point that it is never registered by the microcontroller. We had 100kΩ resistors so all we had to do was find the capacitance.

$$C = \frac{\tau}{R} \qquad (1)$$

$$C = \frac{0.01}{100{,}000} = 0.1uF$$

Thus, our capacitance for our button circuit is 0.1uF. During normal operation, both the enable and GPIO0 pins are pulled high, allowing the device to boot from flash memory.

2.3.4 Sensor Subsystem

The sensor subsystem will use a resistive soil moisture sensor (SEN0114), a temperature and humidity sensor (SHTC3), and a light sensor (BH1750). All these sensors, except the soil moisture sensor which requires an ADC, will communicate with the microcontroller using I2C. The sensors will send their measurements to the microcontroller to be interpreted and relayed through the app. Our power subsystem will supply the correct voltages to the rated amounts of the sensors, 3.3V.

The I2C interface consists of an SDA and SCL. The SDA is our bidirectional data line that will transfer data from the sensors to the microcontroller. The SCL is the common clock reference that is determined via our microcontroller. We used only one I2C pin pair for our two sensors because we wanted to reduce multiple traces and we can connect all our sensors in parallel since it is a bus. We determined that this was possible since the two sensors have different addresses dedicated to them so one will not overwrite the other. The temp/humidity sensor will be at x70 and the light sensor will be at x23.

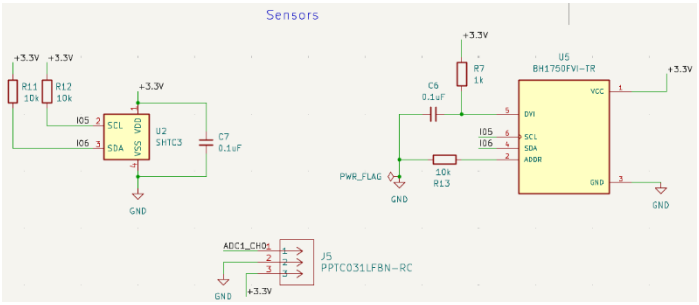

Figure 8: Sensor Subsystem Schematic

Also, we will have to include pull-up resistors on the SDA and SCL lines to improve speed and data quality as seen in Figure 8. This is because our sensors can only pull the line low and the high level is created by the pull up resistors. We determined the value of our pull-up resistors, 10kΩ, by referring to the breakout boards for our sensors as well as seeing what common practice for I2C buses is. In our schematic, we have also included a capacitor between the temp/humidity sensors power and ground to act as a decoupling capacitor. This will stabilize the supply voltage and filter out high frequency noise. Similarly, we have included a decoupling capacitor on the light sensor for the same reasons.

For the soil moisture sensor, we have a connector that will be attached to the cable of the sensor. The analog output will be directly wired to the ADC pin on the microcontroller. The soil moisture sensor outputs an analog signal that ranges from 0-900, which is a total of 10 bits. In our code when initializing the ADC pin, it is set to obtain 10 bits from the sensor at a rate of 1 sample/sec. This is a relatively slow sample rate, but this is because we only intend to send notifications once a day, so we do not need it to be fast.

2.3.5 App Subsystem

This subsystem consists of the user interface and the decision engine for our system. It will collect the base environmental measurements of the plant: soil moisture, temperature, humidity, and light, as well as weather conditions from the plant's place of origin. With this data, plant care decisions will be made using pre-existing thresholds for the species of plant.

The app will be implemented using Flutter and Android Studio. There are a few different sets of data to consider. The first is the species lookup table- the app will contain a plant database of predefined data, which contains optimal soil moisture (very dry to very moist), an optimal temperature range (in Celsius), an ideal humidity range (in %Relative Humidity), an ideal light range (dark to bright light), the origin region of a number of plant species, and the correct soil depth to measure moisture. This data will be obtained from existing data from reputable sources. The user will be able to view this table in one of the app pages and select which species their houseplant is. If the species the user is looking for is not in the table, the user is able to add a row of data and input custom values. With the lookup table, the app will use the target ranges for soil moisture, temperature, humidity, light, and

origin location for the plant species selected to compare the sensor data to. The app will also inform the user of how deep to insert the moisture sensor. We also have the sensor data, which is the data received from the sensor device via BLE. This will contain the current soil moisture, temperature, humidity, and light exposure of the plant. This amount of data is well within the bandwidth of BLE- each sensor data value will be around 2 bytes of data, so 8 bytes total. Finally, there is the weather data for each plant's origin location. Once per day, the app will call a weather API (Free Weather API) using the stored origin location of a plant to obtain the precipitation probability.

The data contained and collected will be used to make decisions to notify the user. To send a watering notification to the user, the current weather data and soil moisture sensor data will be consulted. There are two cases: the first is if the precipitation probability of the plant's origin region is above a threshold of 50%, AND the soil moisture reading is below a threshold high moisture value, and the second is if the soil moisture reading is below a threshold low moisture value. Essentially, the plant watering notification will be sent if the chance of rain at the origin is over 50%, making sure the soil isn't oversaturated, and if the soil moisture is dangerously low. A notification will also be sent if the soil moisture reading is above the threshold high value, informing the user that the soil is oversaturated, and that it might help to improve drainage. A notification to lower temperature will be sent if the temperature sensor reads a value above the recommended maximum temperature, and a notification to increase temperature will be sent if the temperature sensor reads a value below the recommended minimum temperature. A notification to increase ventilation will be sent if the humidity sensor reads a value above the recommended maximum humidity, and a notification to mist the plant or use a humidifier will be sent if the humidity sensor reads a value below the recommended minimum humidity. A notification to move the plant to a brighter window (perhaps a South-facing window) will be sent if the light sensor reads a value below the recommended light exposure, and a notification to move the plant to a dimmer location will be sent if the light sensor reads a value above the recommended light exposure.

The app will be made up of a home screen and three other pages. The home screen displays the current sensor data values, which are updated every 3 seconds, as well as the current chance of rain at the plant's origin. It also has a button that can be used to connect with the sensor device via Bluetooth, as well as three other buttons leading to the other three pages. It displays the logo of our project and includes a text bubble welcoming the user and informing them to insert the soil moisture sensor 2 inches deep in the soil. Then, there is a page that contains the plant species lookup table, where the user selects the species of their plant and can see the ideal ranges of the environmental data. There is also a page that contains outstanding care notifications that the user has not yet acknowledged. If the conditions causing the notification are fixed, then it automatically goes away, and the user is also able to click it to remove it from the screen. Finally, there is a page that stores the sensor data history- it stores a row of data each minute for an hour, before it starts replacing the older values. This data history table helped us with the verification of our sensors. Below is the final app design.

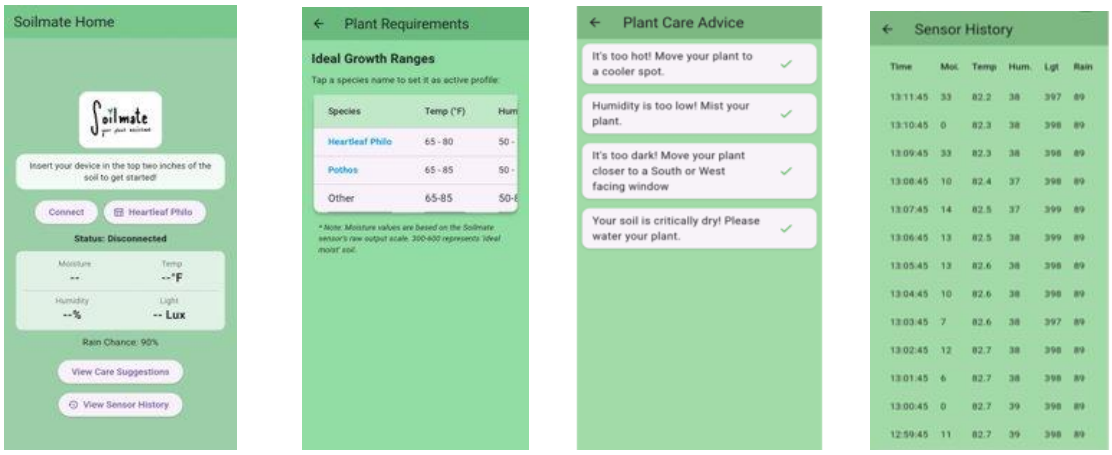

Figure 9: App Design

3. Design Verification

3.1 Power Verification

The power subsystem is composed of two components, the wall adapter and the LDO voltage regulator. To verify that the adaptor was outputting the 5V it is rated for, as well as being within the range of 2.5 to 5.5 volts, which is the minimum and maximum supply voltage of the LDO, it was probed with a digital multimeter. It was connected to the barrel jack connector, and the connector's voltage and ground pin were probed and measured to be around 5.23V. Similarly, the LDO's output voltage and ground pins were probed and measured to be around 3.297V. These verifications satisfied our requirements as they provided the correct amount of voltage to the sensor and microcontroller subsystems.

3.2 Microcontroller Verification

The purpose of the microcontroller is to essentially provide communication between the subsystems. It must pull data from the sensors using I2C and ADC and then it must send that data to the app using BLE. So, we have two requirements, which can be seen in Appendix A, where the microcontroller must be able to send data to the app and it must do this continuously for at least 100 times without browning out. We ran longevity tests to test the stability of the sensors, which will be discussed more in Section 3.3, but essentially, we had our product plugged in for at least 5 hours and we were pulling data from the sensors to analyze their behavior. The product was connected to the app on a laptop via BLE where it was displaying the real live data values. We pulled data from the sensors once a minute, so we pulled data a total of 300 times. This surpasses our initial requirement of 100 times and during this time, the microcontroller did not shut down or brown out and continued to behave normally. The values were expected and there were no extraneous outliers thus verifying the microcontroller.

3.3 Sensors Verification

The sensors had more requirements since we had to ensure that all three sensors were operating as expected. These requirements and validation can also be seen in Appendix A. We had tests that were focusing on the sensors being able to respond to changes in the environment as well as longevity testing where we ran different configurations for at least an hour each. To begin, we wanted to verify the soil moisture sensor by releasing drops of water in dry soil to see how the soil moisture reads. Our graph is shown below in Figure 10.

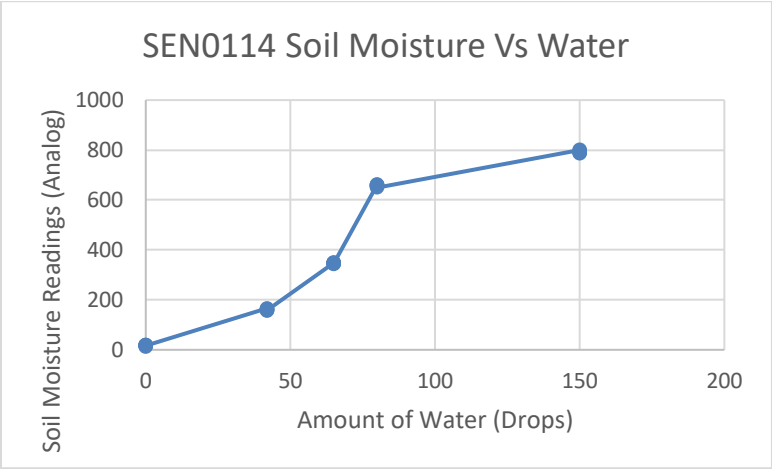

Figure 10: Graph of SEN0114 Soil Moisture vs Water

We started with a near zero reading due to the plant being initially dry and it confirmed our requirement of dry soil measuring between 0-300 analog. We then added 40 drops of water to get a reading just below the threshold for moist soil, afterwards we had 20 drop increments to show the readings increase. We had one measurement that was within our moist data range, 373 analog at 64 drops of water, which satisfied that requirement. The last data point is fully waterlogged soil that had a sensor reading of around 800 analog, which matches our last requirement. Figure 10 proves that our sensor is operating as expected as the soil moisture is increasing with increasing amounts of water.

We ran a test on our light sensor to confirm that it will correctly respond to light changes in the environment. We tested this by placing a ruler above the sensor and then shined a flashlight an inch from the sensor and then increased the distance in increments of one inch. Our graph of this experiment can be seen in Figure 11. This demonstrates the sensors' ability to respond to the variability of light. It starts as a steep line when the flashlight is close to the sensor but then becomes much shallower when the flashlight is farther away. This could be due to the light dispersing over a greater area so it is not as concentrated and therefore does not have as large of a difference between data points.

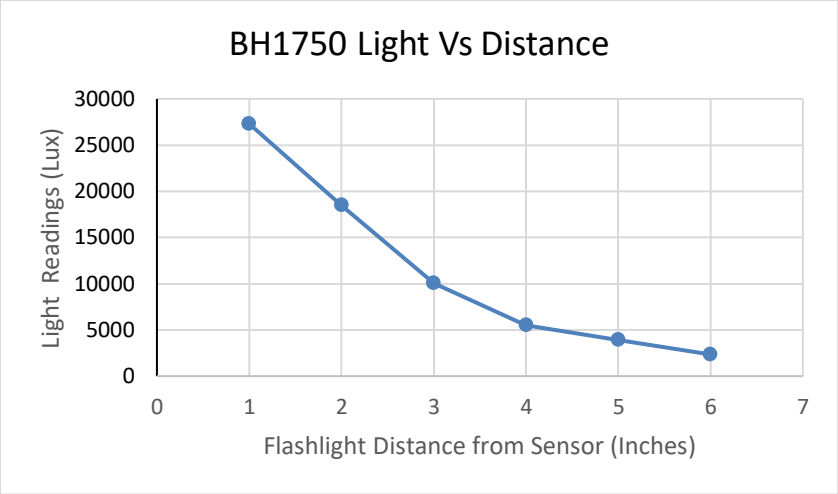

Figure 11: Graph of BH1750 Light vs Distance

We also ran three different longevity tests with each test running for at least an hour to validate the sensors abilities to measure ambient environments. These tests lasted for a total of five hours; this included the time to test and to set it up as well. Our different test configurations consisted of:

1) Run product in enclosure near window with soil moisture sensor inserted in plant pot
2) Run product in enclosure near window with soil moisture inserted in small cup of dirt
3) Run product in enclosure in dark areas with soil moisture inserted in small cup of dirt

We ran these tests in order of how they're laid out above. We began testing at 11:30AM and ended at 4:00PM which is important to note for the tests taken by the window. We set a digital hygrometer/thermometer device next to our product to be able to intermittently check the temp/humidity readings and compare it with the data that we were gathering in the app. We also had the Lux Light Meter Pro app open which we would compare our lux values with.

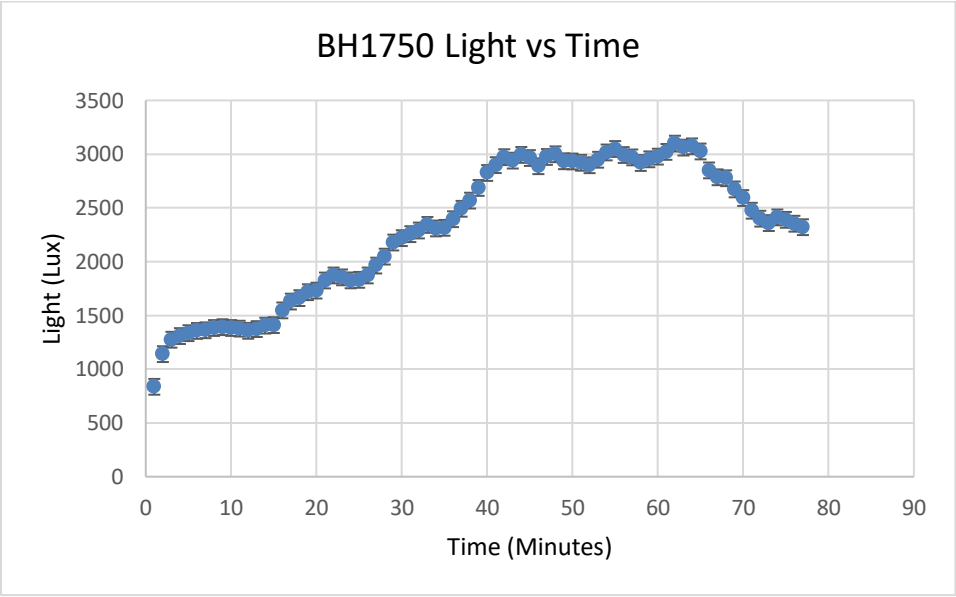

Figure 12: Graph of Test 1 with BH1750 Light vs Time

We can begin by seeing how the light sensor varied due to it being by the window. During the nearly 80 minutes we tested this for, the sun was becoming higher in the sky and brighter as we reached peak sunlight hours. This is evident in Figure 12 as the light sensor corresponds to these changes and continued to increase lux values as we reached 12-12:30PM with a peak of 3048 lux. After the sun was at its highest point in the sky, we see our lux values begin to drop off. The temperature sensor correctly corresponds to the light readings as it was capturing higher temperatures due to the sun shining on the sensor and effectively heating it up. Figure 13 is the temp vs time graph during this same period. Figure 13 follows the same trend as Figure 12 which demonstrates how our sensors line up with each other and further cements the fact that the sun through the window was the reason for these fluctuations in our data. It is also important to note that the base temperature of the apartment, which is what it was set to on the air conditioning, was 74°F. We see in Figure 13 that the sensor initially started at an approximate temperature of 75°F before the sun began to make it hotter. This is how we want the sensors to react to the sun, as this is exactly what temperature the plant would be feeling. In proves that the sensor is necessary for an owner to be able to check intermittently because it is possible that the plant could be reaching much hotter temperatures than the owner thinks they are. This could affect the plant in harmful ways by causing wilting or scorching the leaves. This is why it is important to keep track of this data and necessary for us to provide suggestions for helpful plant care.

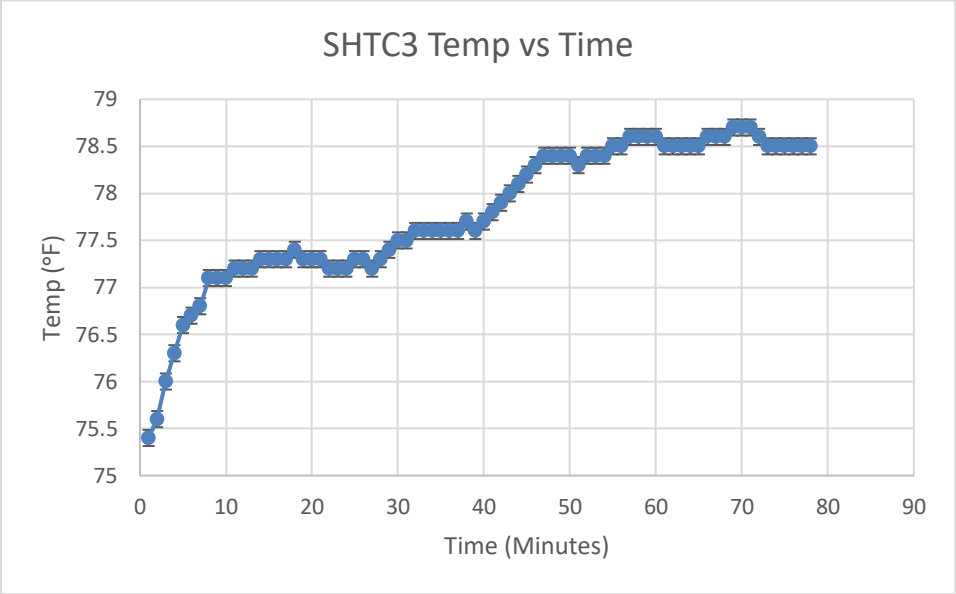

Figure 13: Graph of Test 1 with SHTC3 Temp vs Time

Our soil moisture sensor was inserted in a plant that we had just watered before beginning the test, which is why our initial soil moisture of 700 analog is above the rated soil moisture for the plant, which can be seen in Appendix B. We wanted to observe the soil moisture decreasing as time passed to confirm our beliefs that the soil moisture will decrease as the water is dispersing throughout the soil and being pulled into the plant's roots. Looking at Figure 14, we can see that our assumptions were correct as the soil moisture decreased from 700 analog to 570 analog through the hour we were measuring for.

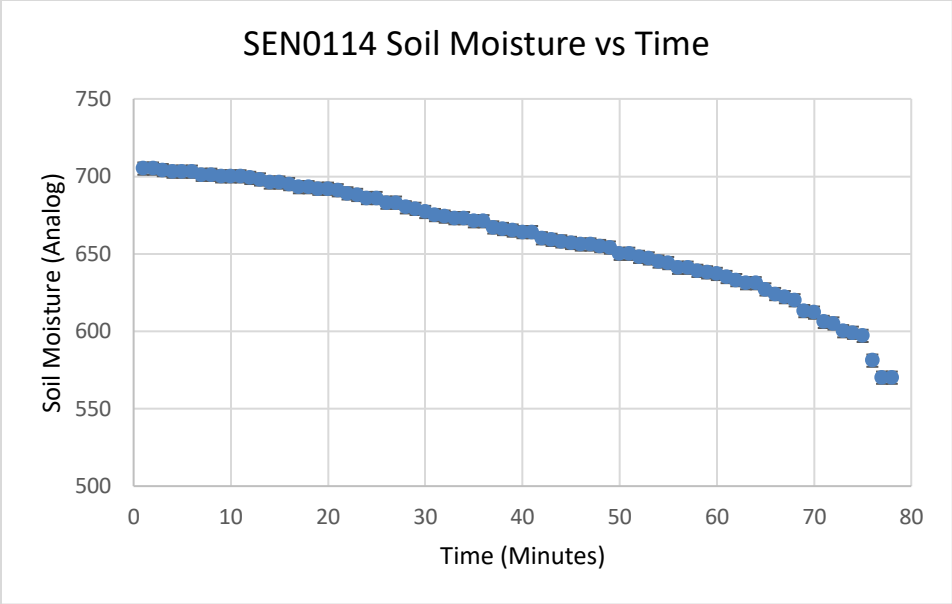

Figure 14: Graph of Test 1 with SEN0114 Soil Moisture vs Time

We also monitored the humidity of the apartment using our product. The humidity stayed constant during the five hours of testing. The hygrometer read 35% throughout the hours of testing. This is too low humidity for the plant we are demoing on, as it needs at least 50% as seen in Appendix B. Our sensor reading aligned with our hygrometer value as we read a consistent 36% during the hours of testing which completes our requirement for the humidity sensor. The humidity readings did not fluctuate as we did not have a humidifier running and we never misted the plant, even though the app was suggesting we do, so there is only one humidity vs time graph which comes from Test 1, as it becomes repetitive after that. We can determine the accuracy of our humidity sensor using the percent error formula shown in Equation 2 since these readings were more consistent than our light, temperature, and soil moisture readings.

$$Humidity\ \%error = \frac{|humidity_{actual} - humidity_{measure}|}{humidity_{acutal}} \quad (2)$$

$$Humidity\ \%error = \frac{|35 - 36|}{36} = 2.78\%$$

This verifies that our sensor has an accuracy of at least 5% since we measured an accuracy of 2.78%.

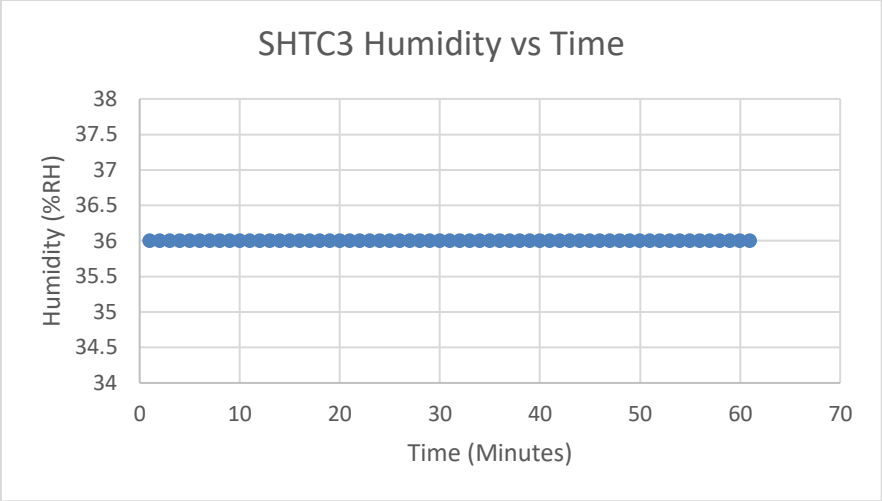

Figure 15: Graph of Test 1 with SHTC3 Humidity vs Time

For Test 2, we switched from having the soil moisture sensor in a plant pot to a cup that was filled with 2 inches of soil. We made this change because we hypothesized that it would dry quicker than the plant pot so we would be able to see a larger range of soil moisture readings. However, upon testing, we found that it actually had a smaller range of readings. Figure 16 is the graph of the soil moisture vs time with the sensor being inserted in a slightly watered cup of soil. We expected that since there was so little dirt, it would dry quickly although that was not the case. We think that it had a smaller range of soil moisture values because there was less soil for the water to disperse to, there were no roots sucking up excess moisture, and there were no drainage holes. All these factors could have been the reason for the soil moisture only decreasing by 15 analog while the plant decreased by 135 analog. This is why it is important for the owner to have proper drainage in their plant so that the excess soil does not continue to saturate the plant if they accidentally overwater it which could risk rotting the roots.

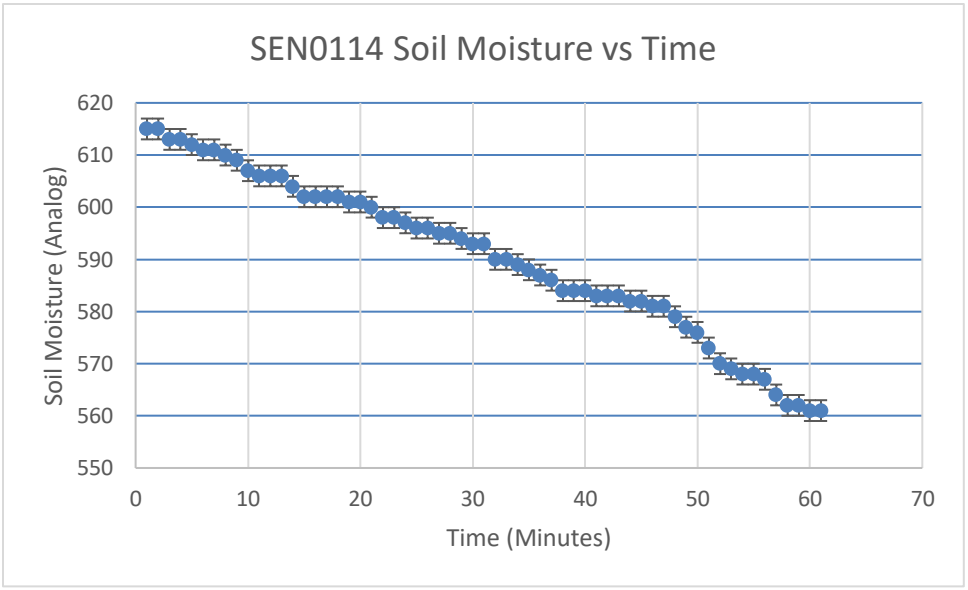

Figure 16: Graph of Test 2 with SEN0114 Soil Moisture vs Time

The other sensors in this setup were hardly affected from Test 1. The humidity vs time graph was identical to Figure 15 and the temperature/light vs time graphs remained aligned with each other but decreased values due to the sun beginning to set. The light sensor began at 2500 lux and decreased to 1268 lux by the end of the test and the temperature started at 78°F and ended at 76.8°F.

We started Test 3 moving our product to the floor, still powered on, and placed a piece of paper over the lid. This caused our board to be completely covered. Again, the humidity showed the same trend as Figure 15 and the soil moisture decreased by another 50 analog since the sensor was still inserted in the cup of soil. The light sensor decreased instantly and fluctuated between 2 and 3 lux during the hour of testing. This verifies our sensor behavior since it accurately changes with light, and we can see it staying consistently dark when we placed it in a controlled area. The lux reading from our iPhone app measured 2 lux during the timing of testing, which was the average reading of our sensor reading, further verifying our sensor subsystem.

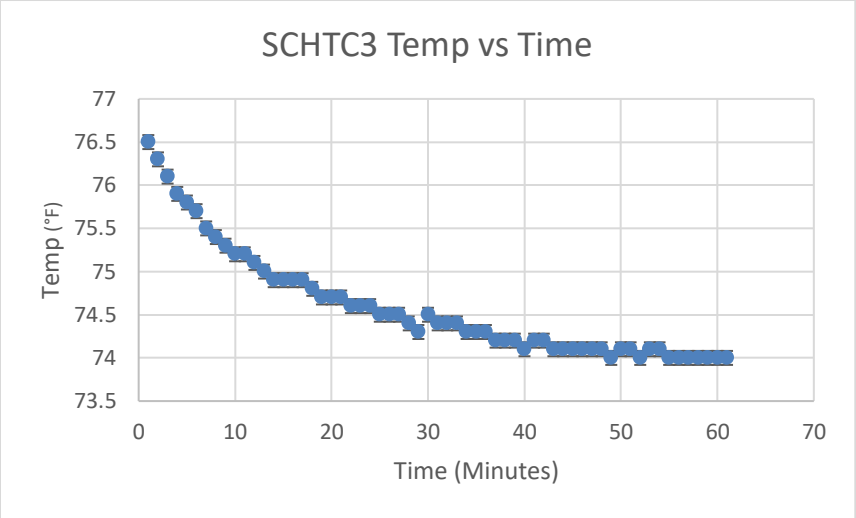

Figure 17: Graph of Test 3 with SHTC3 Temp vs Time

Above in Figure 17 is the graph of our temperature sensor during the third test. As stated before, the sensor read 76.8°F at the end of Test 2 and we kept the product powered on while we transferred it to the controlled environment. This is why the temperature initially reads a high value of 76.5°F. We can see the temperature decreasing as time increases due to it now cooling down, since it is no longer in the sun, and correctly measuring the rooms' ambient temperature. As previously stated, the air conditioning was set to 74°F which matches perfectly with the temperature that the sensor stabilized at. This third longevity test verified the temperature sensor and the light sensor as being accurate to at least 5%, since it was the exact readings of the devices we used to verify the readings with.

3.4 App Verification

A major requirement of the app is that it is able to display the correct current sensor values from the plant for temperature, humidity, light, and soil moisture. Essentially, we had to ensure that the data was being packaged correctly by the microcontroller before being sent via Bluetooth to the app, and that the app was able to decipher the data correctly and separate the package into the different sensor readings to display. We verified this by comparing sensor data values in the Arduino IDE serial monitor to the values that displayed in the app, and making sure that they were the same. We also sent test values that we knew were without doubt accurate, such as the soil moisture sensor value being 0 when it was not

near any moisture and ensured that the app displayed these values. We also needed to test the accuracy of the notifications sent. For moisture related notifications, we made sure that notifications only send in three cases: when the soil is bone dry, when the soil is oversaturated, and given that the moisture is below the threshold high, when the chance of rain at the origin is above 50%. We replicated all these conditions, and we verified that a notification showed up in the user's phone, and that it went away when the issue causing the notification was resolved, or if the user acknowledged it in the app. We also guaranteed that notifications are sent to the user for unsuitable temperature, humidity, and light conditions.

4. Cost and Schedule

4.1 Cost Analysis

4.1.1 Labor

The average starting salary from the ECE department at the University of Illinois at Urbana-Champaign is $90,115 for electrical engineers and $103,222 for computer engineers. If we average these, we get $96,668.5 overall. If we assume a 40 hour work week for 52 weeks a year, we get an hourly rate of $46.48/hour. Since this is a four-credit-hour class, the average time spent outside of class should be about eight hours a week for an approximate 14 weeks. As a result, it should take 112 hours per person to complete the course and 336 hours as a team. Using the formula, ($/hour) x 2.5 x hours to complete, we get a total of $13,014.40 per person and $39,043.2 for the team. We also estimate 3 hours of labor from the machine shop for our physical design. If we estimate an hourly wage of $40/hr for the machine shop, we get a total of $120.

4.1.2 Parts

Description & Part Number	Manufacturer	Vendor	Quantity	Unit Price	Total Price	Datasheet
Microcontroller ESP32-S3-WROOM-1	Espressif Systems	Eshop	5	N/A	$0	Datasheet
Moisture Sensor SEN0114	DFRobot	DigiKey	3	$2.70	$8.10	Datasheet
Temperature & Humidity Sensor SHTC3	Sensirion AG	DigiKey	3	$2.06	$6.18	Datasheet
Light Sensor BH1750FVI	Rohm Semiconductor	DigiKey	3	$3.29	$9.87	Datasheet
ADC Module ADS1015	Adafruit Industries LLC	DigiKey	3	$9.95	$29.85	Datasheet
6V 1A Power Supply Adapter	Security-01	Amazon	1	$8.99	$8.99	Datasheet
Voltage Regulator LDO-AP2112K-3.3TRG1	Diodes Incorporated	Eshop	8	N/A	$0	Datasheet

PCB Mounting Female DC Power Barrel Jack	California JOS	Amazon	1	$6.99	$6.99	Datasheet
Capacitor - 0.1µF	YAGEO Group	Eshop	15	N/A	$0	Datasheet
Capacitor - 1µF	YAGEO Group	Eshop	8	N/A	$0	Datasheet
Capacitor - 22µF	YAGEO Group	Eshop	8	N/A	$0	Datasheet
3 Position Header Connector PPTC031LFBN-RC	Sullins Connector Solutions	DigiKey	3	$0.30	$0.90	Datasheet
Resistor - 100k Ω	N/A	Eshop	6	N/A	$0	N/A
Resistor - 10kΩ	N/A	Eshop	24	N/A	$0	N/A
Resistor - 5.1kΩ	N/A	Eshop	6	N/A	$0	N/A
Resistor - 4.7kΩ	N/A	Eshop	14	N/A	$0	N/A
Resistor - 1kΩ	N/A	Eshop	4	N/A	$0	N/A
Resistor - 0Ω	N/A	Eshop	16	N/A	$0	N/A
Tactile Switch PTS645	Littlefuse	Eshop	8	N/A	$0	Datasheet
USB-C Port USB4500	GCT	DigiKey	2	$0.67	$1.34	Datasheet
Light Sensor-Dev board BH1750	Adafruit Industries LLC	DigiKey	2	$4.50	$9.00	Datasheet
Temperature & Humidity Sensor-Dev board SHT31-D	Adafruit Industries LLC	DigiKey	1	$13.95	$13.95	Datasheet
USB-C Port USB4500-03-1-A	GCT	DigiKey	2	$0.72	$1.44	Datasheet
Temperature & Humidity SensorSHTC3	Sensirion AG	DigiKey	2	$2.15	$4.30	Datasheet
Light Sensor BH1750FVI	Rohm Semiconductor	DigiKey	2	$3.42	$6.84	Datasheet
USB-C Breakout Board 1528-2873-ND	Adafruit Industries LLC	DigiKey	2	$2.95	$5.90	Datasheet
USB-C Port 4827-217B-CA05TR-ND	ATTEND Technology	DigiKey	2	$1.98	$3.96	Datasheet

5V 1A Power Supply Adapter	MTDZKJG	Amazon	1	$9.41	N/A	Purchased on own
4pcs USB Type-C Breakout Board	Treedix	Amazon	1	$7.99	N/A	Purchased on own
Shipping			3	$6.99	$20.97	
Tax-DigiKey					$0.24	
Tax-DigiKey					$0.81	
Tax-DigiKey					$2.30	
Tax-Amazon					$1.48	
Total Cost					**$143.41**	
Remaining					**$6.59**	

4.1.3 Total

Team Labor Hours	336
Machine Shop Labor Hours	3
Total Labor Hours	**339**

Team Labor Hour Costs	$39,043.2
Machine Shop Labor Hour Costs	$120
Cost of Parts	$143.41
Total Cost	**$39,306.61**

4.2 Schedule

Week	**Task**	**Person**
February 23	Order all parts	Emma
	Design Document	All
March 2	Test & Verify Wall Adaptor	All

	Design Review	All
	Schematic Design	Emma
March 9	Bread Board Assembly and Testing	Emma & Ysabella
	PCB Design and Ordered	Ysabella
	Begin App Development	Sigrior
Spring Break		
March 23	Contact Machine Shop for Physical Design	Emma & Ysabella
	Debug App Development	Sigrior
March 30	Individual Progress Reports	All
	Test App Development	Sigrior
April 6	Progress Demo	All
	Cont. Testing and Revising	All
April 13	Begin Soldering PCB	Emma & Ysabella
April 20	Continue Soldering	Emma & Ysabella
	Test PCB and sensors	All
April 27	Finish Testing and Debugging	All
	Final Demo	All

	Final Presentation	All
May 4	Final Report	All

5. Conclusion

5.1 Accomplishments

Throughout the semester we made several accomplishments and achievements. Firstly, we successfully completed our power subsystem requirements and were able to get the correct voltages for our sensors and microcontroller. Secondly, we were able to program our microcontroller using the USB-C breakout board and Arduino IDE. Thirdly, we were able to gather accurate data from our sensors on our PCB using our microcontroller I2C communication. Lastly, we were able to combine this with our app to display the sensor data for the user and integrate the weather API to determine what notifications should be sent to the user's phone. Overall, we completed and verified all of our high-level requirements and subsystem requirements.

5.2 Uncertainties

Despite meeting our main requirements and major goals for our project, we do have some areas that could use improvement. First, our device is powered by a plug-in wall adaptor, so the device, and therefore the plant, must be placed in close proximity to a wall outlet. This is rather inconvenient, as there could be no outlets near where the user wants to place the plant. Additionally, our device is only able to connect to the app within the room. This isn't very convenient for the user- the point of our project is that the user does not have to take on any of the mental load of plant care, and having to reconnect to the device every time you leave the room and come back is definitely frustrating. It would be better if the user was able to receive sensor information and care notifications no matter where they were, even outside of their home. Another factor of our project that we are not satisfied with is the physical design. It is very large, nearly triple the size of our PCB. Our thought behind this was that we wanted the PCB to be well ventilated, but the disadvantage the size causes outweighs this benefit. The temperature and humidity sensor is not quite placed near enough to the plant, resulting in readings that are not as accurate as it could be, it takes up a lot of space the user may not have next to their plant, and it is not very visually appealing. It would be far more convenient if the device was small enough to be discreet. We will discuss the possibility of improving these issues in the next section.

5.3 Future work

Although we have completed our high-level requirements, there are some future improvements that we would have liked to make. For example, a battery powered subsystem would be more convenient for the user, giving them more flexibility on where they want to place their plant. Also, the enclosure for our PCB is quite large and inconvenient, so reducing its size would be another improvement that would benefit the user. We would like to design an enclosure for the PCB that would keep the side ventilation with a much smaller size, and could discreetly be attached to the side of a plant pot instead of being in plain sight next to the plant. This way, the temperature and humidity sensor would also be closer to the plant, resulting in a more accurate reading. In addition, sealing the holes in the lid for the soil moisture sensor and barrel jack connecter with epoxy would help ensure our product is waterproof. We also would have liked to collect information about the plant's pH levels by adding another sensor. Another improvement we would like to make is having and connecting multiple PCB

sensor devices to the app since many plant owners have more than one plant. It would also be a lot more convenient for the user to be able to receive care notifications and see plant sensor data from wherever they were, not just within the same room. We could do this by instead sending the data from the sensor device to a cloud platform using WiFi via the user's home router, which the app can access from wherever the user is, if they are connected to the internet. Lastly, we would have liked to send encouraging confirmation notifications when the plant's environment is optimal to motivate the user. In conclusion, we completed what was proposed and if time was not a limiting factor some of these improvements could have been implemented, but overall the project was successful.

5.4 Ethical Considerations

As we built our project, we strove to follow the IEEE and ACM codes of ethics while working as a team. We prioritized honesty and transparency within our group, making sure to maintain communication throughout the project, respect each member's contributions, and professionally resolve conflicts that came up. We ensured that we have been honest in the reporting of our project, with not falsifying results, and accurately portraying the capabilities and limitations of our project.

Regarding the app, we are using weather forecasting data that is publicly available through a public API. We are not tracking or collecting the location of the user and using the weather forecast information of their location. Instead of this, the app stores only plant-specific environmental information. As we are not gathering any personal information of the user, there is little risk relating to the security of the user's personal information. Despite the system not handling sensitive information, we will still make sure to responsibly handle data. Any stored plant information will remain local to the app unless explicitly required. For the WiFi connectivity that will be used in API calls, we will make sure to follow the communication protocols defined by the standard libraries that are publicly available. The BLE communication in the system occurs only between the sensor device and the app, so there is little security risk there as no sensitive information will be sent. Also, we believe that the UI for the app is relatively user-friendly. Notifications are clear and not excessively sent, the text is in a readable font, and the app is easy to navigate.

We recognize that inaccurate sensor readings can result in incorrect recommendations that harm the user's plant. We did our best to make our system as reliable as possible by thoroughly testing all our sensors and understanding the values that we obtain and what conditions they correspond to, rather than relying on the metrics given by the datasheets. We ensured that we obtain reputable and accurate data for plant health to compare to the sensor data, and we clearly stated the limitations of our device so that the user understands the extent of the device's capabilities.

References

[1] ACM, “ACM Code of Ethics and Professional Conduct,” Association for Computing Machinery, 2018. Accessed: Feb. 24, 2026. [Online]. Available: https://www.acm.org/code-of-ethics

[2] A. Chaturvedi, “How to NOT become a plant murder 101,” Medium. Accessed: Feb. 04, 2026. [Online]. Available: https://avantikachat.medium.com/how-to-not-become-a-plant-murder-101-8d561ba9c3bb

[3] Adafruit, “Adafruit 4-Channel ADC Breakouts,” ADS1015, Liz Clark. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-4-channel-adc-breakouts.pdf

[4] Adafruit, “Adafruit VEML7700 Lux Sensor - I2C Light Sensor,” 4162, Adafruit. [Online]. Available: https://www.mouser.com/datasheet/2/737/Adafruit_4162_Web-3358314.pdf?srsltid=AfmBOopDM3WwLZ02u-W_l13Wc8AjIB-ZqZPf7_tH2-xlQQmw5mamytgO

[5] B. Ray, “Examining 5 IEEE Protocols - Zigbee, WIFI, Bluetooth, Ble, and WiMax.” *IoT For All*, Dec. 02, 2024. Accessed: Feb. 13, 2026. [Online]. Available: www.iotforall.com/ieee-protocols-zigbee-wifi-bluetooth-ble-wimax.

[6] DFRobot, “Moisture Sensor (SKU:SEN0114),” SEN0114, DFRobot. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2280/SEN0114_Web.pdf

[7] “ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U,” ESP32-S3-WROOM-1, Espressif. [Online]. Available: https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[8] Filo, “Mobile device data statistics phone,” iStock. Accessed: Feb. 04, 2026. [Online]. Available: https://www.istockphoto.com/illustrations/mobile-phone-market

[9] Flaticon. Accessed: Feb. 04, 2026. [Online]. Available: https://www.flaticon.com/free-icon/microcontroller_2752878

[10] G. Pauljoe, “Setup Arduino IDE to flash a project to ESP32,” Medium. Accessed: Feb. 27, 2026. [Online]. Available: https://medium.com/@pauljoegeorge/setup-arduino-ide-to-flash-a-project-to-esp32-34db014a7e65

[11] IEEE, “IEEE Code of Ethics,” IEEE.org, 2020. Accessed: Feb. 24, 2026. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[12] I. D. Castillo, “This Is How Many Houseplants the Average Plant Parent Has Killed,” *Apartment Therapy*, May 11, 2022. Accessed: Feb. 04, 2026. [Online]. Available: https://www.apartmenttherapy.com/dead-houseplants-average-37076429

[13] J. Hylén, N. Zambetti, and K. Söderby, “Inter-Integrated Circuit (I2C) Protocol,” ArduinoDocs. Accessed: Feb. 27, 2026. [Online]. Available: https://docs.arduino.cc/learn/communication/wire/

[14] K. Lang, “The Benefits of Having Houseplants.” *SDSU Extension*, 29 May 2025. Accessed: Feb. 27, 2026. [Online]. Available: extension.sdstate.edu/benefits-having-houseplants.

[15] “NASA Plant Research Offers a Breath of Fresh Air.” *NASA*, NASA, 2019. Accessed: Feb. 27, 2026. [Online]. Available: spinoff.nasa.gov/Spinoff2019/cg_7.html.

[16] ROHM, “Digital 16bit Serial Output Type Ambient Light Sensor IC,” BH1750, ROHM. [Online] Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6165/bh1750fvi-e.pdf

[17] S. Santos, “ESP32 Bluetooth Classic with Arduino IDE - Getting Started,” Random Nerd Tutorials. Accessed: Feb. 27, 2026. [Online]. Available: https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/

[18] S. Santos, “ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals,” Random Nerd Tutorials. Accessed: Feb. 27, 2026. [Online]. Available: https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/

[19] Sensirion, “Datasheet SHTC3 Humidity and Temperature Sensor IC,” SHTC3, Sensirion, July 2018. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/906/HT_DS_SHTC3_D1.pdf

[20] Siebenthalers, “Indoor Houseplant Watering Guide: Siebenthaler’s.”, Accessed 27 Apr. 2026. www.siebenthaler.com/indoor-houseplant-watering-guide/.

[21] Soltech, “Heartleaf Philodendron.”, Accessed 27 Apr. 2026. soltech.com/products/heartleaf-philodendron-care?srsltid=AfmBOooGRmFDfulngQuYi8EO69Xnps21zZLmrlL7C-7yDmU3oNmCQHor.

Appendix A Requirement and Verification Table

Table 1 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Power Subsystem Requirements a. The LDO must be able to output a voltage of 3.0V to 3.6V to power the microcontroller and all the sensors b. The wall adaptor must be able to output a voltage of at least 2.5V and no more than 5.5V, with an accuracy of at least ±5%	1. Verification a. Using an oscilloscope or DMM we can probe the output pin and measure the output voltage b. Using an oscilloscope or DMM we can probe the output pin of the barrel jack connector and measure the output voltage	Y
2. Microcontroller Subsystem Requirements a. The microcontroller must successfully transmit sensor data to app via BLE b. The microcontroller must remain operational during transmission and gather sensor data without resetting or shutting down	2. Verification a. Establish a two-way serial Bluetooth communication and request data from I2C bus and send it to the app b. With the previous requirements in place, create a loop to continuously pull sensor data at least 100 times and ensure data is expected	Y
3. Sensor Subsystem Requirements a. The SEN0114 must output analog signal between 0-900 to accurately represent the dampness of the soil b. The SHTC3 must accurately determine the temperature and humidity of the room with an accuracy of at least ±5% c. The BH1750 must accurately determine the brightness of the room in lux with an accuracy of at least ±5%	3. Verification a. Ensure that bone dry heartleaf philodendron plant measures 0-300 b. Ensure that moist soil measures 300-600 c. Ensure that oversaturated soil measures 600-900 d. Compare temperature readings to the temperature of the room by using a thermostat e. Compare humidity readings to the humidity of the room by using a digital hygrometer f. Compare brightness readings to the brightness of the room using an iPhone app called *Lux Light Meter*	Y

	Pro	
4. App Configuration Subsystem Requirements a. The app must display accurate BLE data from the microcontroller b. The app must only send a watering suggestion when the correct conditions are met, and not otherwise c. The app must generate environmental alerts after receiving sensor data that indicates plant conditions are not met	4. Validation a. Send known test values from microcontroller and confirm that values match in the app UI b. Check that the following conditions generate an alert • Bone-dry soil • Oversaturated soil • Soil is not oversaturated and chance of rain is above 50% c. Check that alerts are generated for unsuitable temperature, humidity, and light conditions	Y

Appendix B Requirements for Heartleaf Philodendron

Refer to Appendix A for citations 20 and 21 detailing the plant's recommended environment ranges.

Table 2 Recommended Ranges for Heartleaf Philodendron	
Conditions	**Recommended Range**
Soil Moisture	300 - 600 (25% - 50%)
Temperature	65°F - 80°F (18°C - 27°C)
Humidity	50% - 80% RH
Lighting	1000 lux - 3000 lux (indirect)