

P.L.A.N.T.



EE 41440: Final Project Report



Silvio Azzam

Will Bernath

Robert Lukasiewicz

Tanner Morreale



Plant – Final Report

1 Introduction.....	2
2 Detailed System Requirements.....	4
3 Detailed project description.....	7
3.1 System theory of operation.....	7
3.2 System Block diagram.....	11
3.3 Plant Monitoring Subsystem.....	12
3.4 Plant Watering Subsystem.....	18
3.5 User Interface Subsystem.....	19
4 System Integration Testing.....	22
4.1 How the integrated set of subsystems was tested.....	23
4.2 How the testing demonstrates that the overall system meets the design requirements..	26
5 Users Manual/Installation manual.....	28
5.1 How to install your product.....	28
5.2 How to setup your product.....	29
5.3 How the user can tell if the product is working.....	30
5.4 How the user can troubleshoot the product.....	30
6 To-Market Design Changes.....	31
7 Conclusions.....	37
8 Appendices.....	38



1 Introduction

Household plants require a lot of care. In order to keep them thriving, one must make sure they are watered, get enough sunlight, live in the right temperature, and many more things. However, different plants generally require different care, and sometimes, these differences can determine whether or not the plant will survive. For instance, a cactus will thrive in a vastly different environment than something like a cattail. The cactus needs a dryer, warmer climate, while the cattail requires a marsh-like environment with lots of water.

Not only do plants require specific types of care depending on what type of plant they are, but caring for a plant is also an ongoing process. Typically, most plants need to be watered regularly. In some cases, if the plant isn't getting enough sunlight or if the room is too hot or cold, the plant needs to be moved to a more hospitable environment. All of this caretaking can become a lot of work for the plant owner. In some cases, the plant owner may not have the time to care for the plant, resulting in a withered or dead plant sitting in their house. Sometimes, people go on extended trips and, therefore, cannot physically care for their plants, again resulting in withered or dead plants merely taking up space in their living room unless they hire a caretaker. Some people are simply forgetful when it comes to taking care of their plants, once again ending with dead plants.

Our device provides a solution for all of these situations. P.L.A.N.T. is an all-in-one plant care system with notification and automatic watering capabilities. The user sets up their device by placing the container on the side of their pot and plugging it into a wall outlet. From there, they enter their email address for future notifications and specific plant type to set the parameters for a hospitable environment. The device will then sense the ambient light, surrounding temperature, and soil moisture level to compare against the plant specifications. If any of these sensed values are out of range for what is acceptable for the plant's survival, an email notification will be sent to the user's email address to warn them that they should make changes. Additionally, if the moisture is measured too low, the motorized pump will be turned on, and the plant will be watered for a predetermined amount of time. The device will take these measurements every thirty minutes to make sure the plant is in an environment where it can thrive. Additionally, the device will alert the user if the water tank needs to be refilled.



This device is intended to be a hands-off monitoring and caretaking device that, after setup, requires very few interactions from the user. Plant owners will no longer have to worry about forgetting to water their Ficus before leaving for work. The design of this device is one that can be used for all household plants, regardless of their needs.

Our design generally met our expectations as a team. In terms of the physical design, it connects to the side of a pot well, is mostly waterproof, separates the soil moisture sensor from the main board, holds an acceptable but not substantial amount of water, and is relatively pleasing to the eye. It does lack in its size, as we wished it to be smaller, its waterproofing capabilities as there are some, but minimal, weak spots, and its ability to blend in with the plant. In terms of functionality, it works the way we wanted it to. The temperature sensor takes readings from the surrounding environment and returns an accurate reading. The light sensor accurately reads how much light, in terms of lux, the plant is receiving. The soil moisture sensor has the ability to tell if the soil is too dry or too saturated but falls a little short in terms of accuracy. In terms of getting an exact reading on the moisture level, it is not able to recognize small changes in moisture. For the purpose of the device, though, it does function how it needs to in order to recognize a lack of water in the soil and tell the motor to pump water into the pot.

The user interface works well, allowing the user to easily enter their email address and the type of plant they will be caring for. It sends notification emails to warn the user when something falls out of range and also has the ability to alert the user if the water tank needs to be refilled, as well as a button to reset that alert. The user interface is straightforward and easy to use.

Overall, P.L.A.N.T. functions as we envisioned and serves the requirements outlined above. It is a great product for all levels of plant owners and offers great value for those looking to have vibrant, thriving plants in their house.



2 Detailed System Requirements

The idea for this project came to fruition with the shared experience of our parents (and ourselves) not being able to keep a plant alive. This led us to think of other people who possibly have this problem, such as casual or unexpected plant owners who don't have the time or knowledge to monitor their plants to the correct extent. Furthermore, many plant owners have the experience but are too busy to keep their plants alive. This led to the idea of creating a plant monitoring device that tracks multiple critical elements of a plant's health, alerting the user if they are incorrect and automatically watering the plant. This takes the hands-on aspect of plant care away and calms the worries and stress from what would be a usual very involved process of taking care of a plant.

When monitoring a plant, there are many parameters you must follow to ensure the absolute best health for your plant. While the list is long, some parameters are for the most advanced plant keepers and, in the end, do not play as critical of a role. This would include measurements like soil PH. We found the most critical to your plant is the sunlight it is receiving, the temperature it is in, and the moisture of its soil. Each of these plays a vital role in the health of a plant, and if they are not correct, they could lead to a very short tenure as a plant owner. In this section, we will explain the importance of each measurement to a plant, how we measured it, and the sets we took to ensure its solution.

To begin with, one of the most vital parts of a plant's life is its ability to enact photosynthesis. Photosynthesis is the process by which plants convert light energy into chemical energy. This energy is used to synthesize organic compounds, primarily glucose, from carbon dioxide and water. The plant then releases oxygen as a byproduct of this reaction. Without a plant receiving proper sunlight, this reaction cannot happen, and your plant will begin to suffer. Therefore, we identified monitoring the sunlight a plant receives as a critical element of our product.

To do this, we added a light sensor (TEPT4400) to monitor the plant, which sends an analog reading of the lux, the measurement for illuminance, indicating the amount of light per



unit area. Each plant has a range of lux the plant should receive in a day and is coupled with a plant database. When your plant is selected, the ranges are already programmed in and the device immediately begins to monitor. Instead of constantly monitoring the plant and sending a flag when something goes wrong, our device takes the average amount of sunlight it receives at the end of the day as its measurement. This eliminates the possibility of erroneous data like a shadow going over your plant for a small period of time as the device makes a reading. If the device finds that your plant did not receive the proper amount of sunlight in a day, the user will receive in an email telling them to reposition their plant.

Next, we have our temperature monitoring system with a temperature sensor (SHT40). The process for the temperature sensor almost directly mirrors the sunlight sensor, except it communicates with our board through I2C. With its already programmed temperature ranges for specific plants, when the average temperature the plant was in for a day is not sufficient, the user will receive an email to move their plant.

Another important element of photosynthesis is the amount of water your plant receives. While the sunlight your plant receives or the temperature of its environment can be changed with a simple relocation of the plant, watering is a much more involved aspect of keeping one's plant alive. Further, a user could consistently water the plant, but the plant could still perish if it is watered too much or too little. That is why, to fix this problem, we integrated a self-designed capacitive soil moisture sensor and pump to automatically water the plant.

The capacitive soil moisture sensor reads the dielectric constant, a measure of a material's ability to store electrical energy in an electric field of the soil across a capacitor. It sends an analog value of the moisture in the soil. This value can range from 0, dry, to 4095, completely saturated. Just like sunlight, each plant has a specific soil moisture in which it thrives, and if it does not get the correct amount, it can suffer. Our soil moisture sensor tracks the moisture in the soil every 30 minutes and, when the value is deemed too low for the threshold of a selected plant, it contacts our water pump and automatically waters the plant. We do this every 30 minutes to ensure the water has been given a chance to permeate through all the soil.



The pump is placed in a 3D-printed housing that contains the water reservoir. This reservoir is filled with water, and when it is low, the user will be contacted to refill the reservoir. This is accomplished by tracking the soil moisture and, if after 3 consecutive readings below the threshold, it is not increasing, shuts the pump off and contacts the user through email. This entire plant watering system allows the user to never have to deal with water until they are contacted to refill the reservoir.

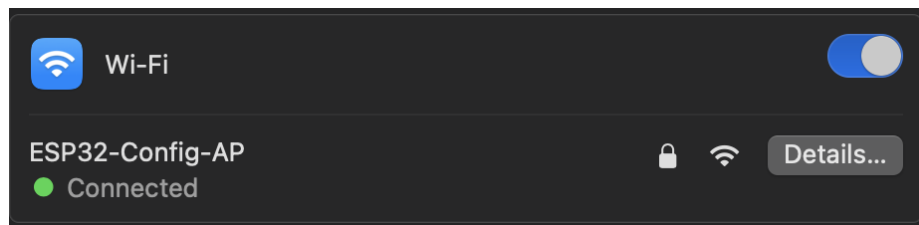
To continue fixing the problem of a plant owner who is inexperienced or too busy, we created a user interface that is easy to access and encompasses all that a plant user would need. Once your device is set up, you can access the user interface through a connection to the ESP32 access point and NFC tag. This will redirect you to a webpage where the user will only need to enter their WiFi network, email, and plant type. The user interface contains a plant database of all plants the user could keep and their specific ranges for sunlight, temperature, and soil moisture. After this is set up, the user will only need to go to their plant to refill the water reservoir or move the plant into a more optimal spot. This device takes the stress and time out of taking care of plants and allows the user the joy of keeping a plant without killing it!



3 Detailed project description

3.1 System theory of operation

- a. Upon power up, the device serves as an access point for the user to connect their mobile device. The user goes into their available wifi networks and selects the network “ESP32-Config-AP” and enters the password: “12345678”.



- b. Following this step, the user brings forth their mobile device to the NFC tag that is built into the reservoir. They will be shown a link to the wifi setup page on the local server. This page prompts the user to enter their home wifi SSID and Password information.



- c. The device then attempts to connect to the network, and the user is prompted to try again in case of an unsuccessful connection. While in the case of a successful connection, the user is redirected to the plant settings page.



Connecting to WiFi...

Connected to WiFi. [Proceed to settings.](#)

- d. On the plant settings page, the user selects the type of plant they want to monitor with the device and the email address where they would like to receive device updates.

A screenshot of a mobile application's 'Plant Settings' page. The title 'Plant Settings' is at the top. Below it is a dropdown menu labeled 'Select plant type:' with 'Lilly' selected. To the right of the dropdown is an email input field containing 'sazzam@nd.edu'. A dropdown menu is open below the email field, showing three options: 'Lilly' (checked), 'Cactus', and 'Ficus'.

Plant Settings

Select plant type: Lilly

Email: sazzam@nd.edu

- ✓ Lilly
- Cactus
- Ficus



Summary

Connected SSID: SDNet
Plant Type: Lily
Email: sazzam@nd.edu

Current Temperature:
19.4 °C

Sunlight Status:
In Range

Moisture Status:
Below Range

Refresh this page for the most up to date plant conditions!


[Edit Settings](#)

[Check WiFi Connection](#)

[Restart Setup](#)

- e. Following this selection, the user receives an email confirming the type of plant they selected, as well as information regarding the wifi network they are connected to and they are redirected to the summary page.

Plant Monitor Confirmation Email



Plant Monitor
to me ▾

Email Confirmation!

Mail Generated from ESP32

Connected WiFi SSID: SDNet

Selected Plant Type: Lilly



Following the initial setup, anytime the user taps the NFC tag, they will be served the summary page, and can decide to make changes to any of the parameters that were set: WiFi network, email for notifications and plant type.

The device now begins its monitoring process. It completes the reading of the temperature using the SHT40 sensor, followed by the moisture reading using our designed sensor, and activates the water-pump in case the moisture level is below the desired range. Lastly, it measures the sunlight level using the TEPT4400 diode. This reading process occurs every 30 minutes and the temperature and sunlight values are stored in a 48 place array.

Following the reading, storage, and watering, if needed, the average value of both arrays are calculated and compared to the pre-calibrated bounds for the monitored plant type. In case of conditions that vary from the bounds, an email is sent to the user describing the issue at hand. This reading cycle continues until the user unplugs or makes any adjustments to the device's parameters.

Plant watering is halted in the case that the plant moisture is below the required range, and the moisture readings have decreased over the past 3 readings. The device deems that the water reservoir is empty, and promptly, an email notifying the user of the issue is sent. The user is served with the option to refill the tank on the summary page, and once confirmed and refilled, the device continues its watering operations as normal, and the button disappears.

Tank Empty

The water tank is empty. An alert email has been sent.

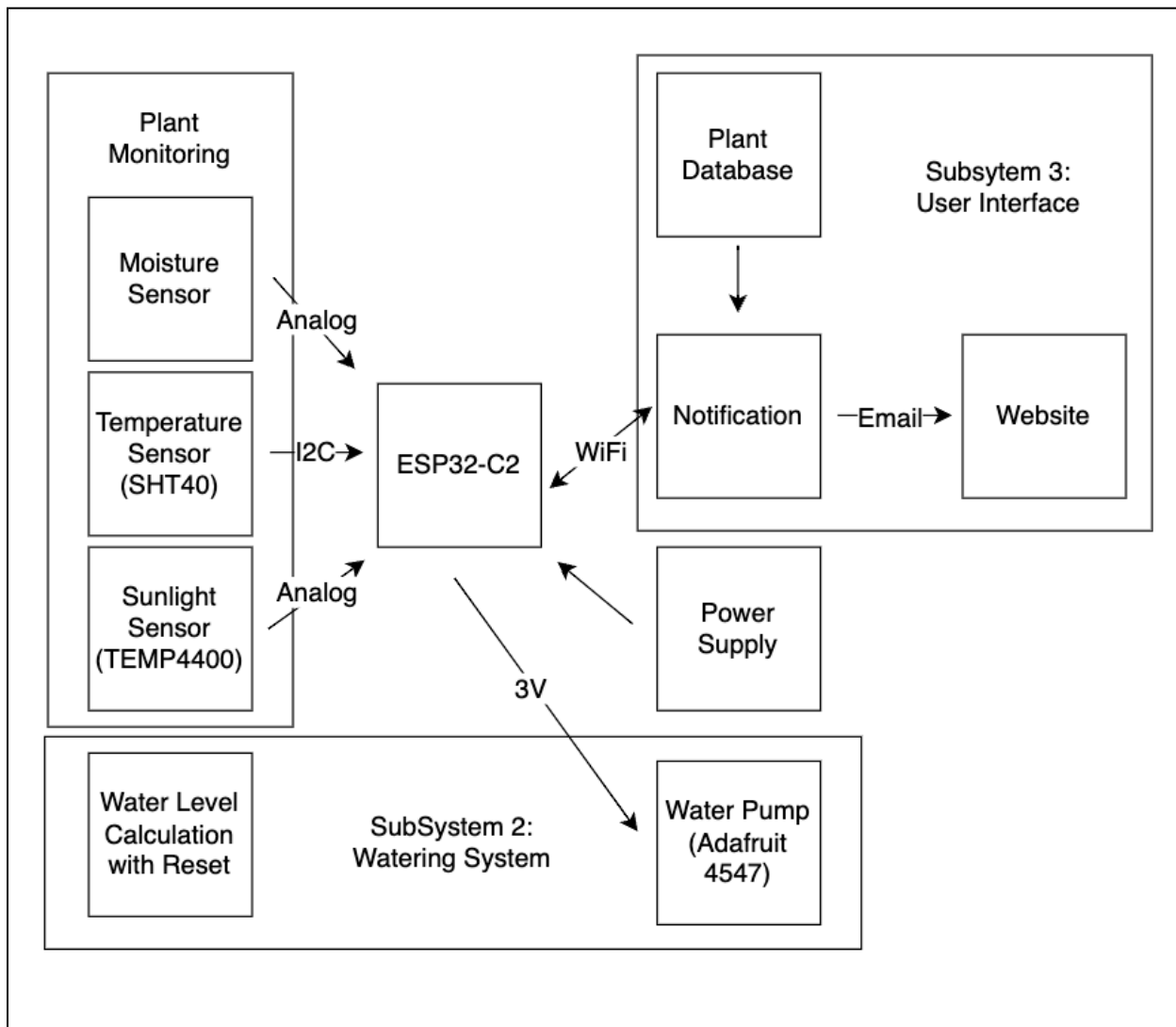
Refill and Reset [Back to Demo](#)



Plant – Final Report

3.2 *System Block diagram*

Overall system block diagram showing how it is divided into subsystems, and the interfaces between the subsystems





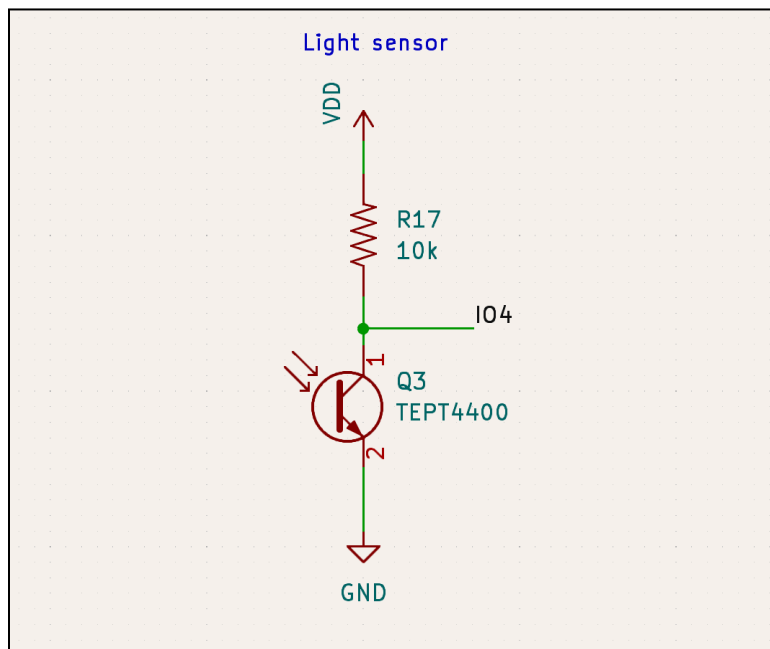
Plant – Final Report

3.3 Plant Monitoring Subsystem

The plant monitoring subsystem has the following requirements:

- Sense the ambient light, temperature, and moisture level of the soil.
- Interface with the ESP32 and send data every 30 minutes when testing is done.
- Return values that can be interpreted and used for email notifications and automatic watering flags.

The light sensor chosen to best test the ambient light in the plant's environment was the TEPT4400 ambient light sensor. This sensor was ideal because of its high photosensitivity and its peak sensitivity at 570 nm. High photosensitivity is necessary as it helps P.L.A.N.T. get the most accurate readings of the ambient light. Photosynthesis for most plants is best when the plant is exposed to light of wavelengths 425 to 450 nm and 600 to 700 nm. The best, available product with high photosensitivity and reliability at the time of design was the TEPT4400. The schematic of this sensor is shown in the picture below.



This sensor is pulled up to IO4 and a 10k ohm resistor to VDD and pulled down to ground. Pin IO4 is used to read the analog data from the sensor. The 10k ohm resistor is used to

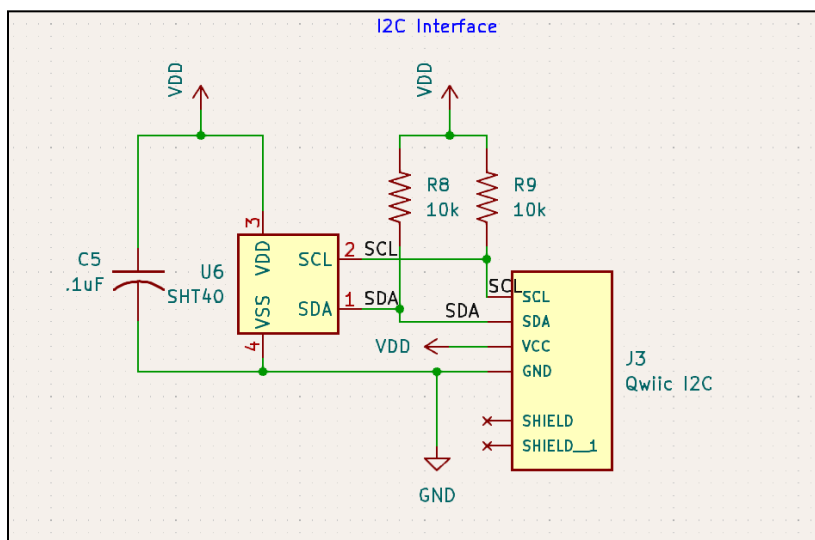


Plant – Final Report

control the current running through the sensor to make sure it is not being over driven with current. This sensor was tested first using complete darkness and a flashlight directly on the sensor with a breakout board to test the actual top and bottom bounds of its range for light. From there, smaller changes in light were made to see what the sensitivity was. Research was conducted to find lux ranges for the light best for photosynthesis and those values were used to test against the TEPT4400. The code associated with the reading is as follows.

```
void MeasureSunlight() {
    int sensorValue = analogRead(SUNLIGHT_ANALOG_PIN);
    sunlightReadings[readingIndex] = sensorValue;
    readingIndex++;
    delay(3000);
}
```

The temperature sensor used is the SHT4x 16-bit Relative Humidity and Temperature Sensor (SHT40). This temperature sensor was chosen because of its temperature accuracy of up to ± 0.1 °C, its I2C communication protocol, and its large operating range of -40 to 125 °C. Accuracy on this level makes it effective when there are small changes in the temperature to get a reliable reading. Also, while no house on earth is going to be at either end of the temperature range, it is good that the sensor can withstand drastic changes in temperature. The schematic of the sensor is shown below.





Plant – Final Report

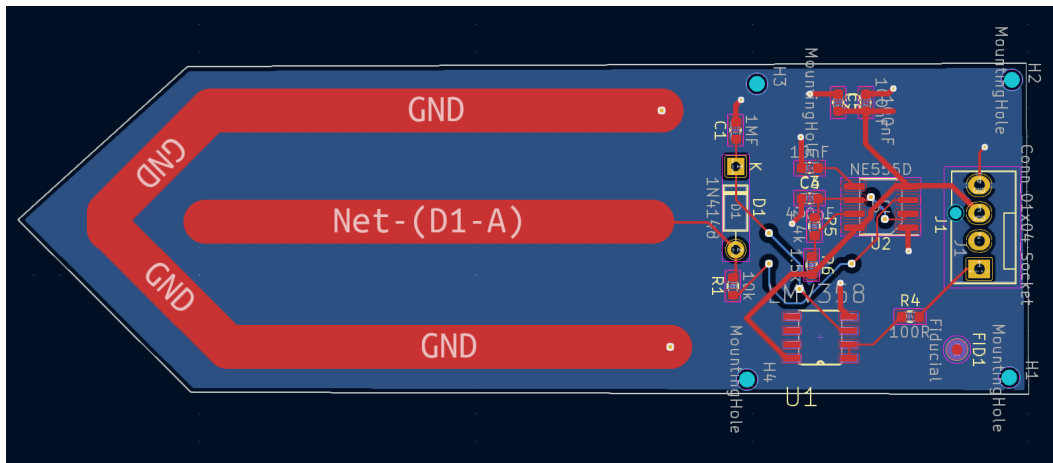
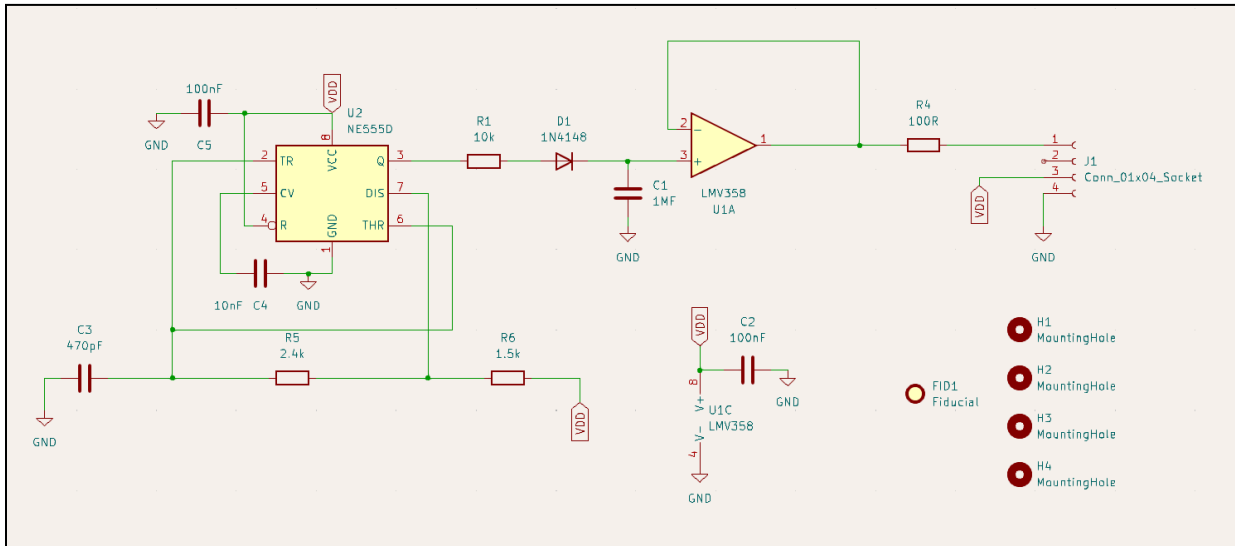
This temperature sensor has connections to SCL and SDA for I2C protocols in order to establish communication between the board and the sensor. Testing for this sensor was more difficult as there was a slight delay in our ability to heat up or cool down the sensor because much of the space was roughly at the same temperature. However, to test drastic changes, we would heat it up by breathing on it, rubbing it with our fingers, or, at times, bringing it outside and inside if there was a big difference in temperature. The code associated with that reading is as follows.

```
void MeasureTemp(){
  sensors_event_t humidity, temp;
  uint32_t timestamp = millis();
  if (! sht4.begin()) {
    Serial.println("Couldn't find SHT4x");
  }
  else{
    sht4.getEvent(&humidity, &temp);

    int temperature_reading = temp.temperature;

    tempReadings[readingIndex]=temperature_reading;
    Serial.println("Temperature value was stored");
    delay(3000);
  }
}
```

The team created the soil moisture sensor, inspired by other capacitive soil moisture sensors currently available for sale. The schematic and PCB for the sensor are included below.



This sensor is designed to use the soil as a capacitor element and the soil charge storing capacity to calibrate the water content. The difference in the dielectric is measured between the traces for ground and Net - (D1 - A). The software for this subsystem returns a value between 0 and 4095, which are associated with different levels of moisture. Below is the software snippet associated with reading the value.

```
int sensorValue = analogRead(MOISTURE_ANALOG_PIN);
```




Plant – Final Report

The moisture sensor design includes the traces extending downward as a way to keep the other components, like resistors and capacitors, from getting wet. Also, this is the only part that needs to be placed in the soil to measure the moisture, so we wanted to avoid having any unnecessary board components on that part.

Testing for this sensor was a multistep process. First, we purchased and tested a capacitive soil moisture sensor already on the market as a way to start getting the software assembled and working in the right direction while designing and building our own. To test for this, we connected it to our main board and had two cups of soil, one wet and one dry, to see what values it returned at each. From there, we were able to calibrate it better by using a soil moisture meter, shown below, connected to an analog scale. When the soil was wet, we took the readings from the moisture meter and capacitive moisture sensor and compared them against each other as a standard. Once we finished our capacitive soil moisture sensor's design, fabrication, and assembly, we followed this idea and tested it against the soil moisture meter to get our bounds.





The calibration process for the temperature and sunlight sensors involved simulating ideal environments for each plant and using the readings from the sensors to set the boundary conditions. The final calibrated figures can be found below.

```
if (selectedPlantType == "Lilly"){
    MOISTURE_UPPER = 1500;
    MOISTURE_LOWER = 2500;
    TEMPERATURE_UPPER = 30;
    TEMPERATURE_LOWER = 20;
    SUNLIGHT_UPPER = 2200;
    SUNLIGHT_LOWER = 0;
}
else if(selectedPlantType == "Ficus"){
    MOISTURE_UPPER = 4096;
    MOISTURE_LOWER = 2200;
    TEMPERATURE_UPPER = 30;
    TEMPERATURE_LOWER = 20;
    SUNLIGHT_UPPER = 2200;
    SUNLIGHT_LOWER = 4000;
}
else if (selectedPlantType == "Cactus"){
    MOISTURE_UPPER = 4095;
    MOISTURE_LOWER = 2900;
    TEMPERATURE_UPPER = 30;
    TEMPERATURE_LOWER = 20;
    SUNLIGHT_UPPER = 4095;
    SUNLIGHT_LOWER = 4000;
}
```

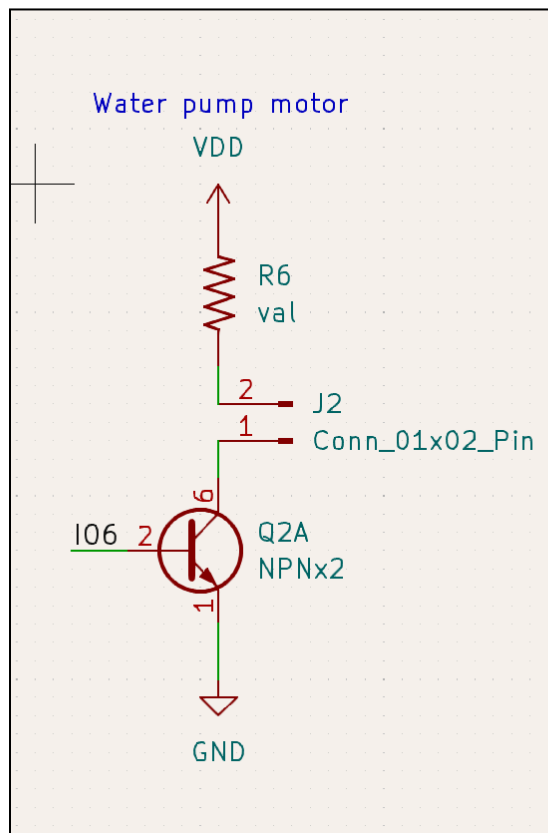
The plant monitoring subsystem consists of three separate sensors communicating with the ESP32-C3. The software accompanying these sensors sets bounds for the plants based on user input and will alert the user via email notification if any of these characteristics fall outside of the desired range for the specified plant.



Plant – Final Report

3.4 *Plant Watering Subsystem*

The three main components of the watering subsystem are the Adafruit 4547 water pump, water level calculation system, and a 3D-printed water tank from which the pump will draw water. This system may be simple, but it is one of the most important parts of our design. If this is not constructed correctly, we could over- or under-water the plants to the point that they do not survive. The water pump will be controlled by a motor that is connected to the board through a 2-pin connector. The circuitry is shown below.



When the soil moisture level reading is too low, the motor will be prompted to run, which will water the plant. The pump will only run for 5 seconds as it was deemed a sufficient amount of water to bring the soil moisture above threshold standards. The NPN transistor in our design is connected to pin IO6.

The water level calculation system involves no sensors but instead tracks the number of times the moisture sensor finds the soil to be below the threshold. Since the pump is



Plant – Final Report

programmed to water the plant each time it is below the threshold, if it goes three consecutive measurement cycles without increasing soil moisture, the user will get an alert that their tank is empty and the pump will turn off. Additionally, when your water tank is refilled, there is a button on the user interface that allows you to notify the device your tank has been refilled. This system allows for no extra sensors to be a part of the design and gives a safety backbone for both your plant and the water pump.

The final component of the watering subsystem is the 3D-printed board housing and water reservoir. This component includes an indented rectangular platform that extends from the body of the housing, allowing the plastic board casing to fit seamlessly. On the bottom of this platform is an extended piece that serves as a “clip” for the entirety of the housing to fit stably on the side of a pot. The water reservoir and its removable lid extend down in front of this platform and clip. In the corner of the lid, there is a hole for the tube delivery system to extend out of in order to transfer water from the reservoir to the plant in the pot. The reservoir is in a half-cylindrical shape with the curved portion facing outward in order to fit on the side of a pot without extending too far off of its sides. The interior of the reservoir was coated with Flex Seal in order to waterproof the housing and ensure that there were no leaks.





3.5 User Interface Subsystem

The user interface has the following requirements:

- SMTP Protocol and a dedicated Outlook email address
- Wifi Capabilities
- The ability to serve as an Access point
- The ability to operate using mDNS

The subsystem must be able to serve as an access point in order to remotely connect it to a wifi network without a physical interface connection. The device must be able to use the mail client and its dedicated email address to send an email notification to the user using SMTP with data that the user has input on the web server. It is also required that the device communicates any changes made to the settings to the device's operations, including a change to the recipient email address, home wifi network, and plant type. The device must also be able to display information regarding the plant's environmental conditions upon refresh on the summary page.

SMTP is used to send emails from a client to a server or between servers. It is responsible for initiating and terminating the email transmission process. Our device operates over port 587 since it is encrypted by Transport Layer Security. SMTP operates using a series of commands and responses. The process generally involves these steps:

Connection Establishment: The email client establishes a TCP connection to the SMTP server.

Mail Transfer Phase:

- MAIL FROM: Specifies the sender's email address.
- RCPT TO: Specifies the recipient's email address.
- DATA: Begins the message data section, which ends with a single period (.) on a line by itself.

Email Sending: The server processes the message, determines the recipient's server using DNS, and forwards the message.

Connection Termination: The connection is closed using the QUIT command.



Overview of mDNS

Purpose: mDNS provides a way for devices to discover each other and communicate on a local network without manual configuration or the need for a DNS server.

Port: In our case, it uses port 80.

Scope: The protocol is limited to local network segments and cannot route across different subnets or wider internet networks.

Query: When a device needs to resolve a hostname to an IP address, it sends a multicast query to all other devices on the local network segment. This query asks if any device knows the corresponding IP address for the hostname.

Response: Devices on the network listen for these queries and respond when they recognize the hostname for which they are responsible. The responding device sends a message back to the querying device with the necessary IP address information.

Cache: mDNS-capable devices often cache responses to reduce network traffic and improve resolution times for future queries.

The user interface subsystem was rigorously tested to ensure a clean and sleek design was provided to the user. It was also fine-tuned via testing between the groups on the easiest way to operate where different options and methods were presented, and we finally settled on the current setup.



4 System Integration Testing

The System Integration Testing phase was essential for detecting and resolving any inconsistencies, bugs, or performance issues that came to light when our individual subsystems were linked together and relied on values from other subsystems rather than hypothetical placement values. Our testing involved the rigorous assessment of the integrated subsystems regarding their functionality together, interface testing and relationships, error handling, performance testing, and accuracy measurement. These tests represented real-world scenarios of our product, P.L.A.N.T., in use and assured that our system behaved as expected under a variety of conditions. Through this testing, we demonstrated that the overall integrated subsystems met the design requirements outlined in the project overview and served as a checkpoint to identify and address any discrepancies from the expected actions of the system.



4.1 *How the integrated set of subsystems was tested.*

The fully integrated system operates as follows:

- I. User sets up the device
 - A. Connect to wifi network “ESP32-Config-AP” with password “12345678”
 - B. Tap NFC tag to enter wifi SSID and password
 - C. The device connects to the network and the user is redirected to settings
 - D. Type of plant selected
 - E. The email address entered and confirmation message sent
 1. Redirected to summary page
 - F. NFC tag redirects the user to a summary page of their plant
- II. Plant monitoring values measured every 30 minutes
 - A. Soil moisture sensor
 - B. Temperature sensor
 - C. Sunlight sensor
- III. Values compared with hardcoded boundary values determined in individual subsystem testing
 - A. Within boundary conditions
 1. Returns in range on summary page
 2. Water pump not activated
 3. Email not sent
 - B. Not within boundary conditions
 1. Returns out of range on summary page
 2. Water pump activated
 - a) User notified if the pump is deemed empty
 3. Email sent



After connecting all of the subsystems together, we tested the integrated set of subsystems in the following order:

- I. Device Setup and Connectivity:
 - A. Manually test setting up the device by connecting to the "ESP32-Config-AP" WiFi network with the provided password.
 - B. Manually test by tapping the NFC tag to enter the WiFi SSID and password.
 - C. Ensure the device connects to the correct network and redirects the user to settings upon successful connection.
- II. NFC Tag Functionality:
 - A. Manually test the NFC tag redirection to the summary page of the plant.
 - B. Ensure the redirection works correctly and displays the relevant information about the plant's status.
- III. Plant Monitoring:
 - A. Manually test selecting the type of plant and entering the email address.
 - B. Simulate various scenarios to test if the device accurately measures soil moisture, temperature, and sunlight at regular intervals (every 30 minutes).
 - C. Ensure the values measured by the sensors are within expected ranges by comparing them with the hardcoded boundary values.
 - D. Test the system's response when the values are within or outside the boundary conditions.
- IV. Integration Testing:
 - A. Test the integration between different subsystems to ensure seamless functionality.
 - B. Verify that the system accurately redirects users to the appropriate pages based on their actions.
 - C. Test the summary page to ensure it displays accurate and updated information about the plant's status.



- V. Edge Cases and Error Handling:
 - A. Test edge cases such as network disconnection, sensor malfunction, or incorrect user input.
 - B. Verify that the system handles errors gracefully and provides appropriate error messages or prompts for corrective actions.
- VI. Manual Testing:
 - A. Manually skewed the values for each of the sensors so that the system would respond how it would in commercial use
 - 1. Different soil moistures (dry, wet, moist)
 - 2. Different temperatures (cold, room temperature, and hot)
 - 3. Different amounts of sunlight (dark, room light, intense)



4.2 *How the testing demonstrates that the overall system meets the design requirements*

- I. Device Setup and Connectivity Testing:
 - A. By manually testing the device setup and connectivity, including connecting to the WiFi network and tapping the NFC tag, we ensured that users could easily set up the device.
 - B. Verifying correct network connection and redirection upon successful connection confirms that the device behaves as expected during setup.
- II. NFC Tag Functionality Testing:
 - A. Manually testing NFC tag redirection ensures that users can access relevant plant information conveniently.
 - B. Confirming that the redirection works correctly and displays accurate plant status information validates this feature's functionality.
- III. Plant Monitoring Testing:
 - A. Simulating various scenarios to measure soil moisture, temperature, and sunlight ensures the accuracy and reliability of sensor data.
 - B. Comparing measured values with hardcoded boundary values validates that the system accurately interprets sensor data and responds accordingly.
 - C. Testing the system's response to different scenarios (within and outside boundary conditions) verifies its ability to control the water pump and send email notifications appropriately.
- IV. Integration Testing:
 - A. Testing integration between subsystems ensures seamless functionality and proper navigation throughout the system.
 - B. Verifying accurate display of plant status information on the summary page demonstrates successful integration and real-time data updating.
- V. Edge Cases and Error Handling Testing:
 - A. Testing edge cases such as network disconnection or sensor malfunction ensures that the system handles unexpected scenarios gracefully.
 - B. Validating error handling mechanisms confirms that users receive appropriate prompts for corrective actions.



VI. Manual Testing:

- A. Manually skewing sensor values replicates real-world conditions, allowing you to verify that the system responds appropriately to different environmental factors.
- B. Testing with various soil moisture levels, temperatures, and sunlight conditions ensures the system's robustness across different scenarios.

By thoroughly testing each aspect of the system, we demonstrated that it meets its design requirements by:

- Providing reliable plant monitoring and control functionality.
- Ensuring accurate sensor data interpretation and boundary condition handling.
- Offering seamless integration between subsystems and intuitive user experience.
- Handling edge cases and errors gracefully, enhancing system reliability.
- Validating system performance across a range of environmental conditions.



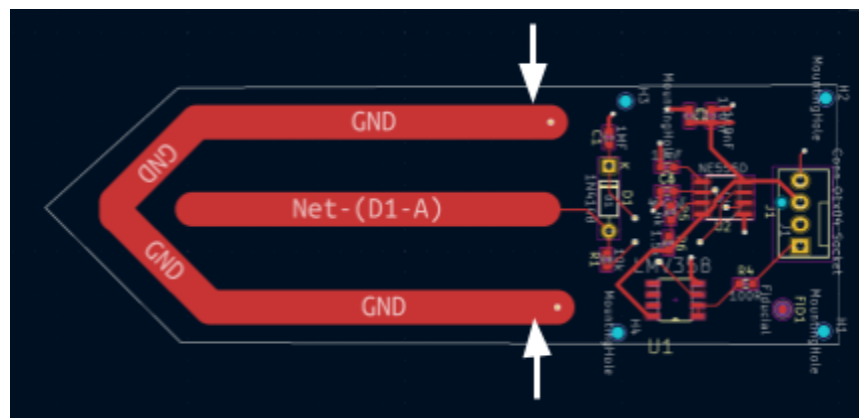
5 Users Manual/Installation manual

5.1 *How to install your product*

When your product arrives, you first need to position the housing on a pot of sufficient size. The most important step in positioning is placing the housing between your plant and the main light source it will receive. This is because the light sensor on the device has been oriented to face directly at the sun in this position. If not positioned correctly, you will not get the optimal performance out of your light sensor.

After this step, the user will need to place the moisture sensor and water spout in the soil. These two should be positioned on opposite sides of the plant. There are two reasons for this. The first is that we do not want the moisture sensors' electronics to get mass quantities of water. Like any electronic device, continued excessive exposure to water will corrode the device. This step ensures longevity in your product. The second reason is that we want our sensor to get a reading of a place where water has been able to permeate through the plant and not just directly next to the pump. The reason we measure this sensor every 30 minutes is to ensure the water has been given a chance to distribute around the plant.

With the moisture sensor, you also want to make sure it is not inserted any deeper than the metal tracks shown below.



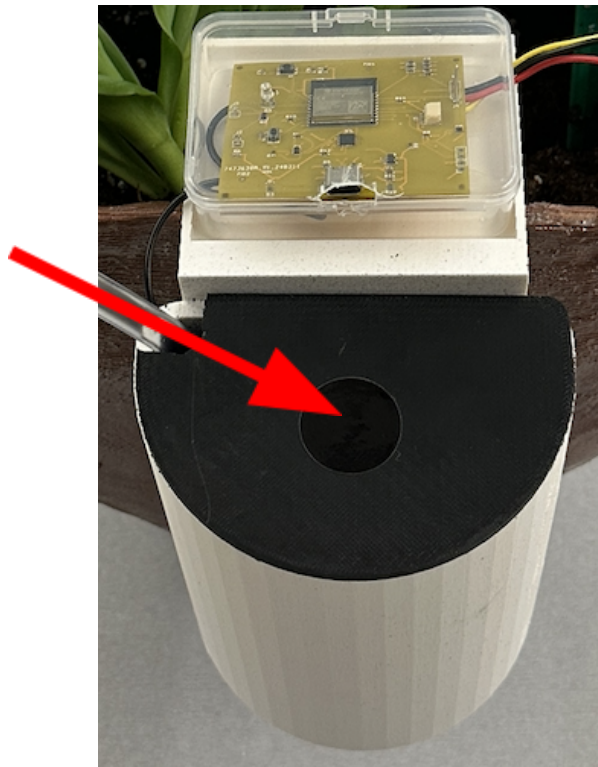
This again ensures the longevity of your device by not getting soil onto your electronics and also optional reading with just the tracks in the soil, the part that will be reading the moisture in the soil. Once all this has been accomplished, the user may not plug the device in and fill the reservoir with water.



Plant – Final Report

5.2 *How to setup your product*

After your device has been fully set up, it is time to enter your settings. In order to do so, the user must go to the wifi settings on their phone and connect to the ESP32 Access Point. Once that connection is established, the user will tap their phone on the NFC tag attached to the lid of the water reservoir.



In the case that the device you are using doesn't have NFC capabilities, the user can access the setup page by entering "<http://esp32-webserver.local>" on the device (Note that to access the setup page, you must be connected to the Access Point, regardless of NFC capabilities).

This will direct you to a webpage that will request that you enter your SSID (WiFi network name) and the password for this network. Once this is entered, you will enter your email address and the type of plant you want monitored. Once this is completed, you should receive an email notifying you that the setup was successful!



5.3 *How the user can tell if the product is working*

The user can tell if the product is working in a variety of ways. First, if the green power light is on the main board. Next, if a confirmation email from the device is received stating the WiFi network was successfully set up. If the user wants to make sure their sensors are working, there is a tab on the user interface to see live sensor readings. Finally, if your water reservoir has any water remaining, you will see the plant being watered or the soil being moist. If any of these ways to tell your device is working cannot be met or you have concerns, go directly to the troubleshooting section below.

5.4 *How the user can troubleshoot the product*

The first troubleshooting step you can take is to press the reset button on your device and set the system back up. If this does not work, continue reading for specific troubleshooting problems. If the green light on your mainboard is not on, make sure your device is connected to a working outlet. You must also check no wires to the pump, moisture sensor, or power outlet are in contact with shipping. Next, if you do not receive your confirmation email setup, the user should restart the setup process. Make sure you are connected to the ESP32 access point and then make sure your SSID, password, and email address are all entered correctly. Finally, if this continues not to work, make sure your WiFi router is also on and working correctly. Finally, if you are not getting proper readings on your sensors, you need to check two possibilities. The first is that your housing is set in the proper position between the plant and the main light source. Next, you must check that the moisture sensor has been properly inserted into the soil. If either of these is not working, it could take a period of time for your device to adjust to its new environment. If none of the above steps have worked to fix your board, you may need a replacement device.



6 To-Market Design Changes

Having developed a functional prototype within the constraints of budget and time limitations, the process has provided invaluable insights into the strengths and weaknesses of the initial design. As is often the case, the experience gained from building the prototype has illuminated areas where improvements can be made to better address the problem the project aims to solve. With the luxury of hindsight and a deeper understanding of the problem space, the focus now shifts towards refining the design to align more closely with market needs and user expectations. This section outlines the key changes and enhancements that would be implemented before bringing the product to market, leveraging the lessons learned during the prototyping phase to create a more robust and competitive solution. These design modifications include both the physical look, feel, and functionality of the device, as well as various improvements regarding each individual subsystem.

Physical Modifications

The first set of design modifications that we would introduce would revolve around the fit of our P.L.A.N.T. monitoring device housing prototype to different plant pot sizes and shapes. This is different from our current “one-size-fits-all” prototype that is made from rigid 3D printed material. These improvements include the following, along with their justification:

- I. Adjustable pot clip: allows the clip situated under the board casing to be adjustable to ensure a snug, secure fit with the edge of the pot rather than the rigid clip we had.





-
- II. Rubber reservoir back: allows the back of the reservoir, which will be the part of the reservoir making contact with the pot, to fit to the shape of the pot while retaining waterproof characteristics. This will ensure that the reservoir and device has a sleek fit to the plant pot without taking up too much space. Additionally, with the reduced space it takes up, the reservoir could be placed facing inside or outside of the pot without taking too much room from the plant inside the pot.

 - III. Adjustable water reservoir size: an alternative to the rubber reservoir back, the water reservoir could also be made to be modular so as to allow consumers to customize the height that the reservoir is. This allows consumers to use the device in smaller pots or to accommodate for a more water-demanding plant.

The next set of improvements we would make to increase the marketability of our system would include those related to enhancing the board casing and water reservoir of the system housing. These actions serve to increase the robustness of our system housing and prevent any malfunctions such as water leaks into the plant, onto the ground, or water reaching the board. These solutions and explanations are listed below:

- I. Embedded board: modify the housing so that the board is embedded within the housing, leaving nothing exposed to potential corrosion or water damage. This would include a modified sunlight and temperature system interface that exposed the sensors to light via a small hole in the top of the casing that is protected by a fitted piece of plastic. The wires that connect to the pump and moisture sensor will be fed through small, sealed, waterproof holes from inside the casing.

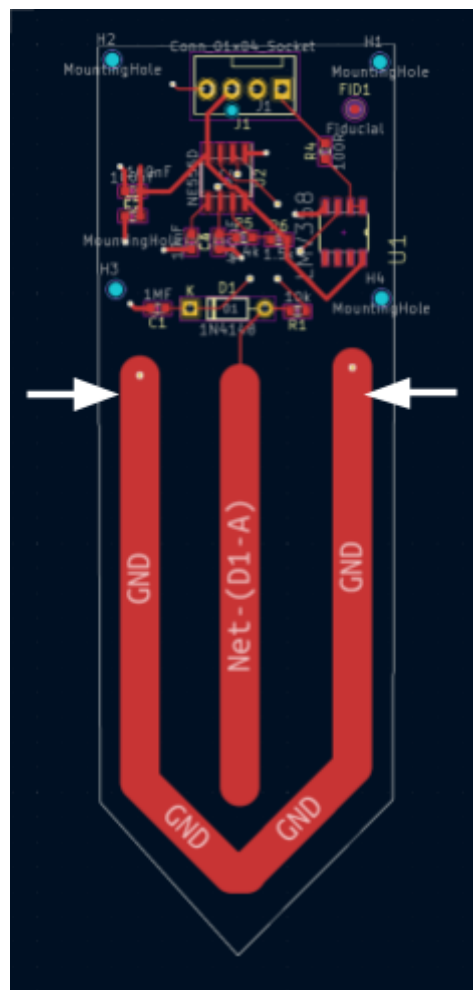
- II. Wires routed through casing: allows the wires to be protected from corrosion, tampering, or water damage that could result in faulty connections. The wires would run through the reservoir interior to the pump and to the moisture sensor.

- III. Utilize waterproof material: our housing design was made via 3D printing, resulting in the water reservoir and housing system overall not being completely



waterproof. We would use a waterproof material or lining in the water reservoir to ensure there are no leaks or water losses and that the board is sufficiently protected from water damage.

- IV. Moisture sensor protection: the moisture sensor that we designed is susceptible to deterioration if its components are placed below the soil line for an extended period of time. To combat this, we would add a protective cap on to the sensor for the user to grab to place it in the soil and serve as a point to indicate to the user that it should not go any further into the soil. Additionally, this would protect the sensor's components, located near the top, and prolong its lifetime in terms of corrosion resistance and value accuracy. The arrows on the image below indicate where the cap would extend to.





There are also improvements that can be made to the tubing that feeds the water from the water reservoir to the soil in the pot.

- I. Tube attachment: This attachment would function like the shower setting on a garden hose and consist of a small screen cap with holes in it fitted over the tube to disperse the water coming out. This would enable the plant to be watered more evenly.
- II. Tube clamps: These would be placed along the pot rim or inside the soil to hold the tube in place and allow the user to set up the tube as they place it. They would come as separate attachments and not be pre-attached to the tube.

Finally, there are cosmetic improvements that we can undertake to improve the overall look of the system. These improvements would make the device more attractive to consumers and make them more willing to have our P.L.A.N.T. device in their homes.

- I. Improved color scheme: The white body and black lid design for the system housing are not ideal for display within a consumer's home. We would ideally offer different colors for the consumer to choose from, but we would offer a standard color scheme of a green body and a green lid.
- II. Reduced board size: We would also like to reduce the size of the board so that the extending lip on the side of the device can be reduced in size. This would allow the housing system to be more compact and sleeker, increasing its marketability and reducing its material cost.
- III. Moisture sensor attachment: we would provide attachments so that the user can put the moisture sensor on the reservoir or clip side, depending on how they orient the system (water reservoir inside or outside of the plant). We would also cover the wire that connects the moisture sensor to the board with protective tubing to reduce potential damage to it and faults, as well as make it look sleeker than a loose wire.



Subsystem Modifications

We would also tweak the individual subsystems if we were to take our product to market. These tweaks are far less involved and fewer in number than the physical improvements, yet they are fixes that would improve our product. We decided that our fully integrated system operates as we would like it to for commercial purposes, but we would change a few things regarding the individual subsystems.

Plant Monitoring Subsystem: many of the fixes that we would like to make have been covered in the physical modifications section. These include the placement of the moisture sensor and sunlight sensor, as well as the improvements regarding the water reservoir system. An improvement in this subsystem that we would like to make before bringing this device to market is the following.

- I. Increased plant library: currently, the plant monitoring subsystem is only programmed to monitor and check the relevant plant conditions of three different types of plants: a Lilly, a Cactus, and a Ficus. For an introduction to the market, we would like to have many more types of plants included in this database, as well as their relevant values. Ideally, we would include all plants commonly sold at flower shops for in-home planting, wild plants, and food-producing plants.

Watering Subsystem: much like the Plant Monitoring Subsystem, we addressed many of the fixes we would like to make before taking the product to market in the physical modifications section. Regarding the software aspects of this subsystem, we would like to make the following improvements:

- I. Water dispensing time: we would like to modify the amount of water (based on the time the pump runs) to better suit the needs of each individual plant. As it is currently a set amount of time each occasion that the moisture sensor indicates that the soil needs to be watered, we want to adjust this so that the pump can output more or less water depending on the individual needs of the plant.



-
- II. Empty reservoir indicator: We would also like to include an additional sensor that indicates when the reservoir's water level is depleted and needs to be refilled. Our current method is a logic-based approach based on the soil moisture level not improving after consecutive watering. Still, a sensor in the reservoir would provide a definite need for refilling the water.

User Interface Subsystem: the improvements that we would like to make regarding this subsystem have not yet been discussed. Before taking this product to market, we would like to further simplify the process of plant monitoring using our P.L.A.N.T. device. These improvements include the following:

- I. Monitoring texts: We would like to send a text to the customer when their plant needs attention rather than sending an email. This would be implemented in order to avoid long email sending times, emails being directed to spam folders, or users who do not regularly check their emails.
- II. User app: we would also like the complete system to be accessible through an app rather than a website. This app would include the setup of the system and every functionality that involves the user, including the plant monitoring values and detected problems.

We believe that, after implementing these physical and subsystem modifications, the P.L.A.N.T. monitoring system would be ready for introduction to the market as a consumer product.



7 Conclusions

P.L.A.N.T. accurately and consistently provides the average plant owner with important updates regarding their beloved plant's health. By monitoring the conditions most vital to a plant's survival and flourishing—the soil moisture level, the amount of sunlight it receives, and the temperature of its environment—the product ensures that your plant remains happy and healthy. Furthermore, our device ensures that the plant is never thirsty by watering it when its soil becomes too dry.

Caring for plants can be difficult. For beginners, there are many nuanced tips and tricks that are not readily apparent for the continued success of their household plants. The learning curve for advanced, professional plant care is steep and often requires a lot of time, effort, and, unfortunately, failure. Recognizing these challenges, our team developed P.L.A.N.T. to ensure that these problems were adequately addressed among the novice plant-parent community.

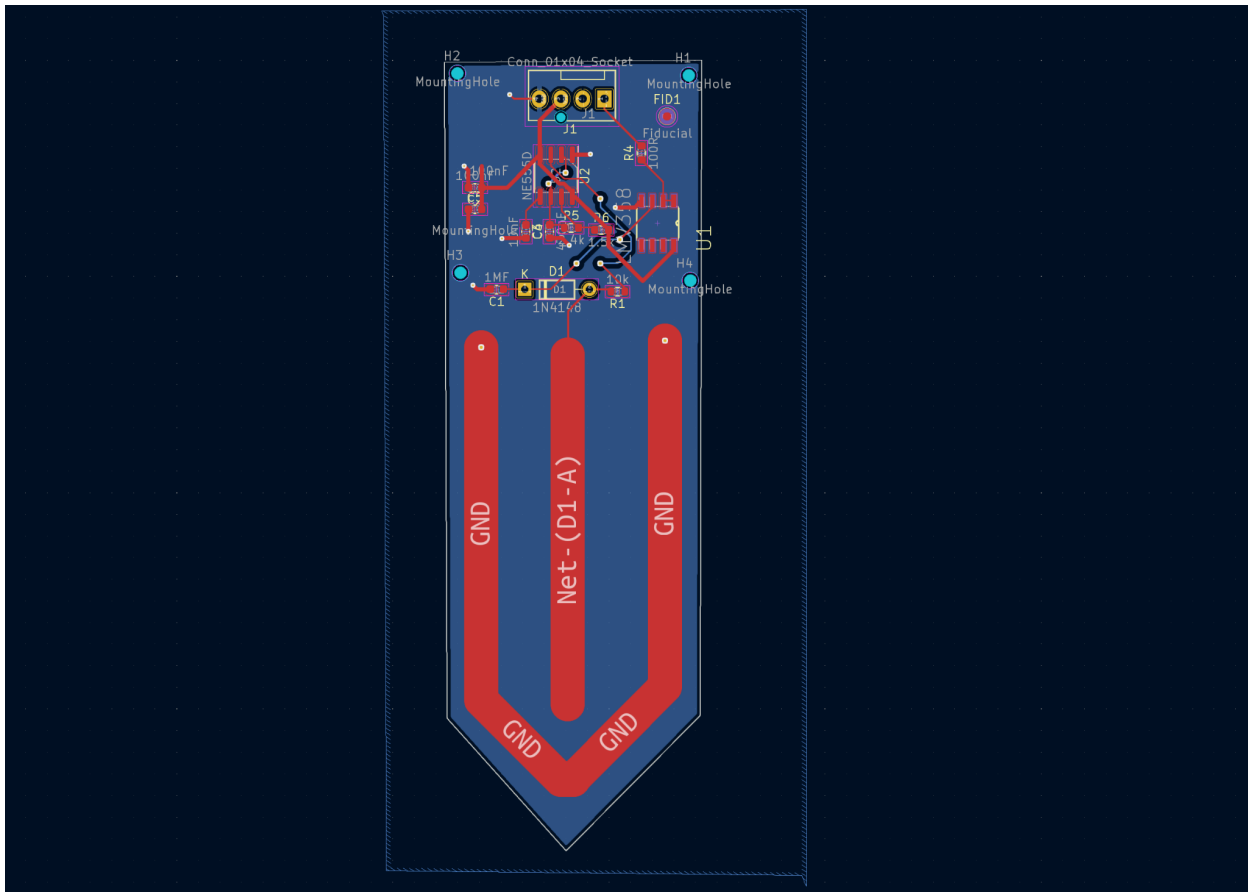
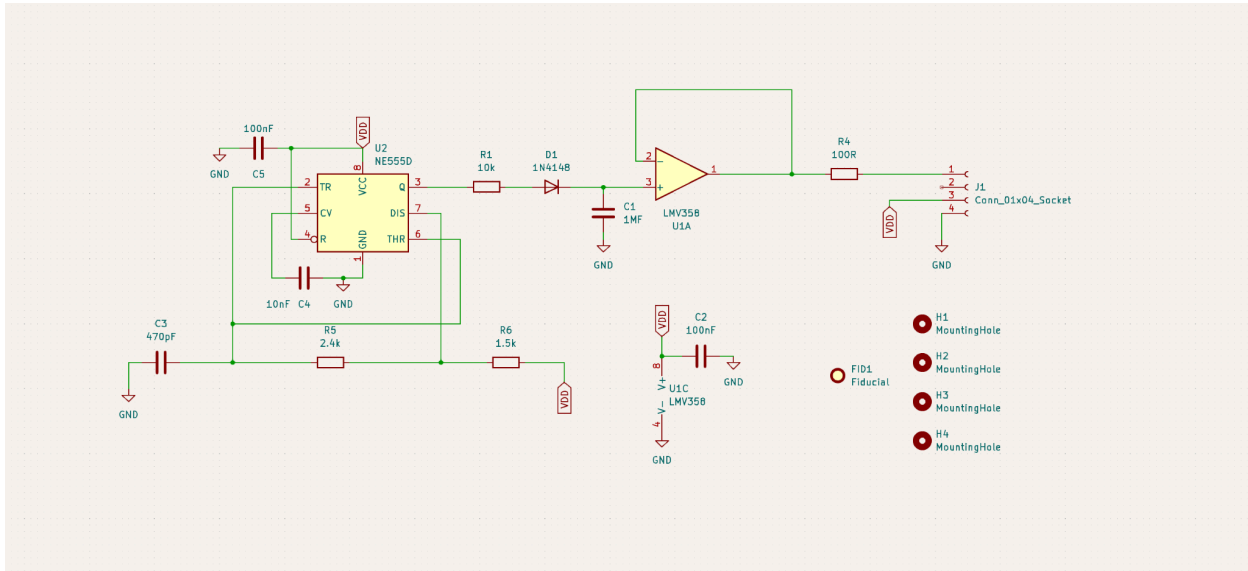
These issues are not only felt amongst new plant enthusiasts, however. There are plenty of plant lovers who are consistently traveling, stuck in the office for hours at a time, or simply have other stuff going on in their lives. P.L.A.N.T. offers an all-in-one plant care solution for this demographic. Our device remotely takes care of the plant while the user is out doing whatever it is that they do, alerting them when their plant needs their attention. Their plant will be well taken care of even when they are across the country.

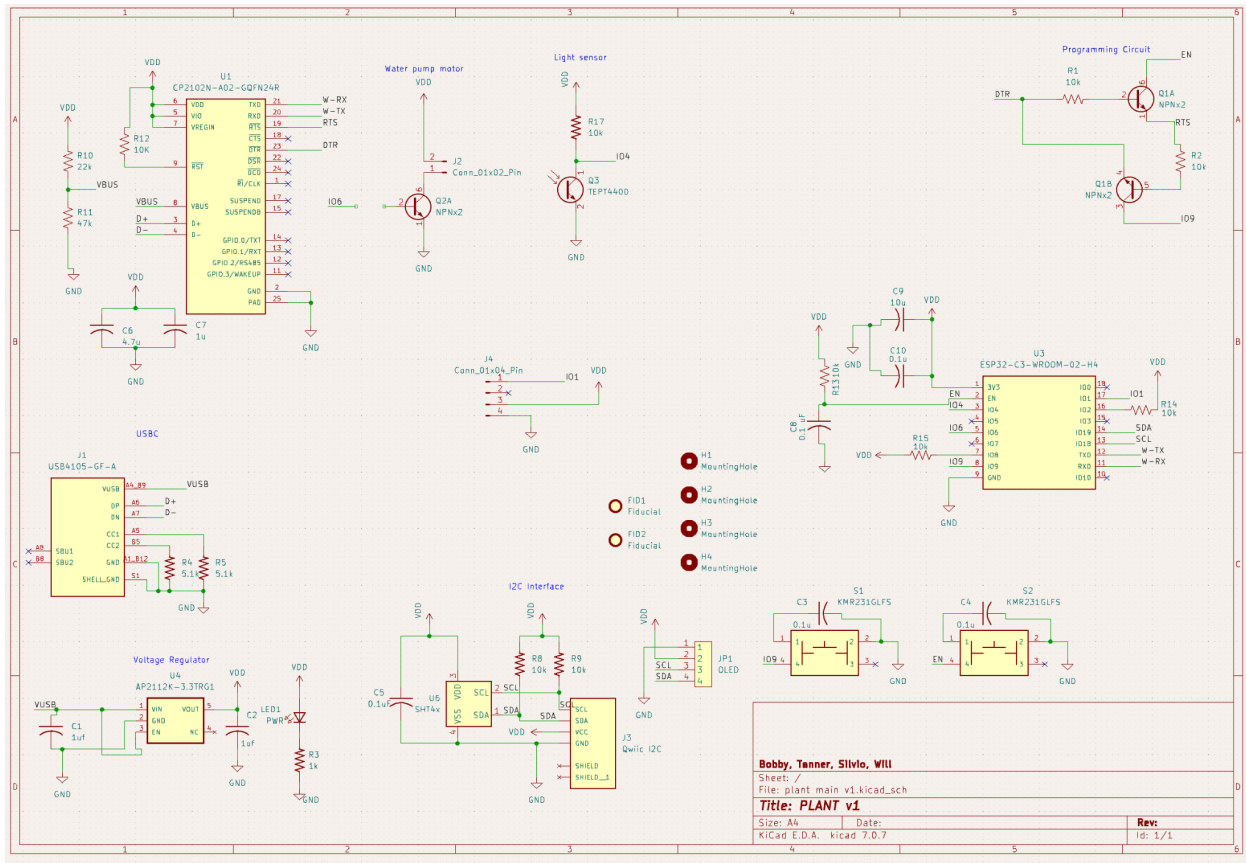
P.L.A.N.T. is an instrumental device in ensuring that new and experienced plant parents alike are able to allow their beloved flowers, succulents, fruits, or any other type of household plant to grow and prosper. Whether a user does not know how to properly care for a plant, does not have the time to do so, or even does not want to, P.L.A.N.T. will ensure its needs are met. This fully integrated solution holds much promise in promoting healthy plant care and a lot fewer plants ending up in the garbage.

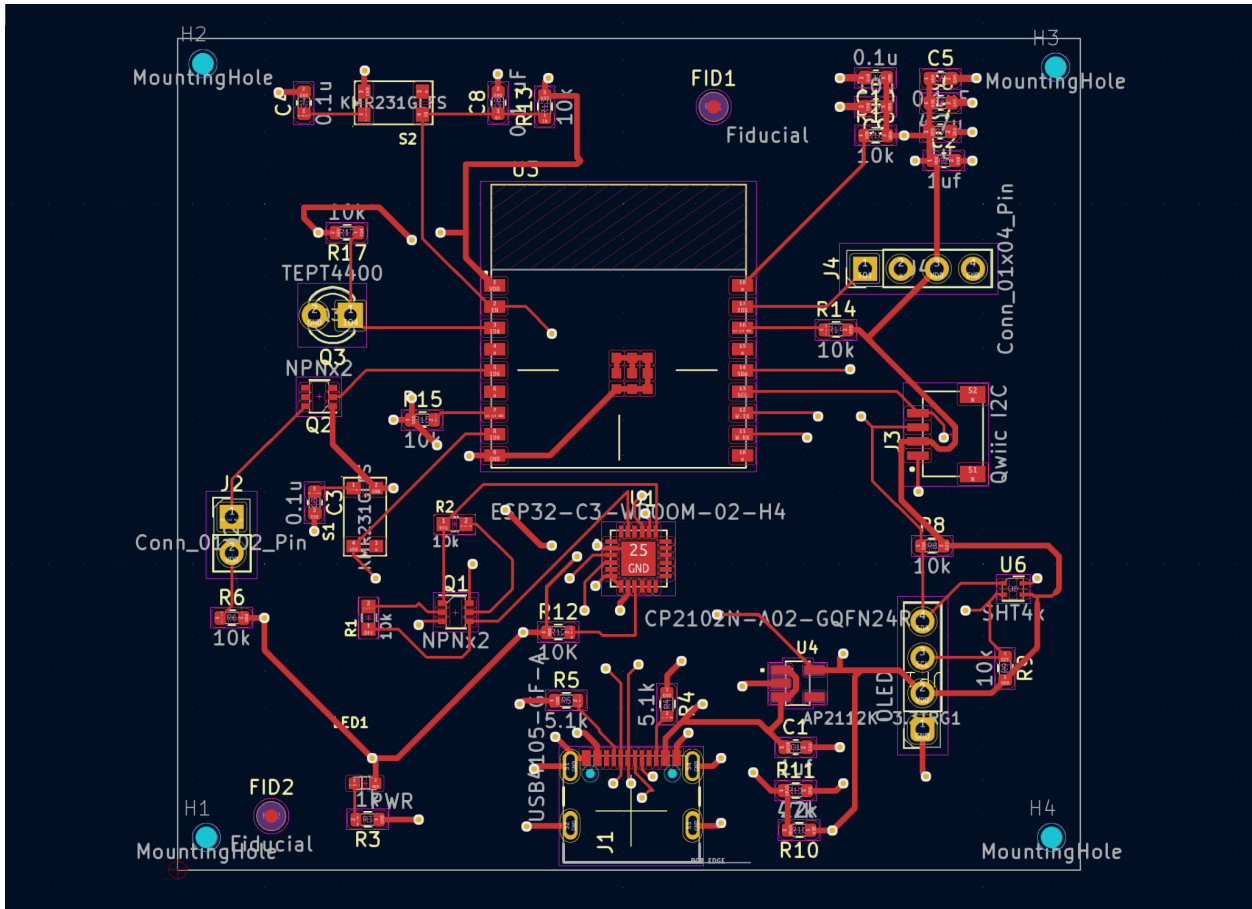


8 Appendices

Complete hardware schematics









Overall Device Design





Complete Software listing

```
[env:esp32-c3-devkitc-02]
platform = espressif32
board = esp32-c3-devkitc-02
framework = arduino
monitor_speed = 115200
lib_deps =
  mobizt/ESP Mail Client@^3.4.15
  adafruit/Adafruit SHT4x Library@^1.0.4

#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <ESP_Mail_Client.h>
#include "Adafruit_SHT4x.h"

#define SMTP_server "smtp.office365.com"
#define SMTP_Port 587
#define sender_email "plant.ee.2024@outlook.com"
#define sender_password "PlantEENotreDame2024"

#define SUNLIGHT_ANALOG_PIN 4
#define MOISTURE_ANALOG_PIN 1
#define PUMP_ACTIVATION_PIN 6

int MOISTURE_UPPER = 0;
int MOISTURE_LOWER = 0;
int SUNLIGHT_UPPER = 0;
int SUNLIGHT_LOWER = 0;
int TEMPERATURE_UPPER = 0;
int TEMPERATURE_LOWER = 0;

const int ARRAY_SIZE = 48;
const unsigned long MEASUREMENT_INTERVAL = 1800000;

unsigned long lastMeasurementTime = 0;
int tempReadings[ARRAY_SIZE] = {0};
int moistureReadings[ARRAY_SIZE] = {0};
```



```
int sunlightReadings[ARRAY_SIZE] = {0};
int readingIndex = 0;

const int i2cSDA = 19;
const int i2cSCL = 18;

const char* AP_SSID = "ESP32-Config-AP";
const char* AP_PASS = "12345678";
const char* MDNS_NAME = "esp32-webserver";
WebServer server(80);

bool setupComplete = false;
bool sendEmail = false;
bool moistureFlag = false;
bool temperatureFlag = false;
bool sunlightFlag = false;
bool reservoirFlag = false;

String selectedPlantType = "";
String enteredEmail = "";
String connectedSSID = "";

SMTPSession smtp;
ESP_Mail_Session session;
Adafruit_SHT4x sht4 = Adafruit_SHT4x();

void SendEmail();
void MeasureTemp();
void MeasureSunlight();
void MeasureMoisture();
void WaterPlant();
void SendStatusEmail();
float ProcessReadings(int readings[]);

void setup() {
  delay(4000);
  Serial.begin(115200);
  delay(1000);
  Wire.begin(i2cSDA, i2cSCL);
  pinMode(PUMP_ACTIVATION_PIN, OUTPUT);
  digitalWrite(PUMP_ACTIVATION_PIN, LOW);
```



```
WiFi.softAP(AP_SSID, AP_PASS);

Serial.println("Access Point started");
Serial.print("AP IP Address: ");
Serial.println(WiFi.softAPIP());
smtp.debug(1);
session.server.host_name = SMTP_server;
session.server.port = SMTP_Port;
session.login.email = sender_email;
session.login.password = sender_password;
session.login.user_domain = "";

if (!MDNS.begin(MDNS_NAME)) {
Serial.println("Error starting mDNS responder!");
} else {
Serial.println("mDNS responder started");
Serial.print("You can now access the ESP32 WebServer at http://");
Serial.print(MDNS_NAME);
Serial.println(".local/");
}

server.on("/", []() {
if (setupComplete) {
server.sendHeader("Location", "/summary", true);
server.send(303);
} else {
String html = "<html><head>"
"<style>"
"body {"
" display: flex;"
" justify-content: center;"
" align-items: center;"
" min-height: 100vh;"
" margin: 0;"
"}"
"form {"
" text-align: center;"
"}"
"</style>"
"</head><body>"
"<form method='POST' action='/connect'>"
"<h1>Setup WiFi</h1>"
```



```

"SSID: <input type='text' name='ssid'><br>"
>Password: <input type='password' name='password'><br>"
"<input type='submit' value='Connect'>"
"</form>"
"</body></html>";
server.send(200, "text/html", html);
}
});

server.on("/connect", HTTP_POST, []() {
String ssid = server.arg("ssid");
connectedSSID = server.arg("ssid");
String password = server.arg("password");
WiFi.begin(ssid.c_str(), password.c_str());
server.send(200, "text/html", "<html><head><meta http-equiv='refresh'
content='5;url=/check-connection'></head><body>Connecting to
WiFi...</body></html>");
});

server.on("/check-connection", HTTP_GET, []() {
if (WiFi.status() == WL_CONNECTED) {
Serial.println("Connected to WiFi successfully.");
server.send(200, "text/html", "<html><body>Connected to WiFi. <a
href='/settings'>Proceed to settings</a>.</body></html>");
} else {
Serial.println("Connecting to WiFi...");
server.send(200, "text/html", "<html><head><meta http-equiv='refresh'
content='5;url=/check-connection'></head><body>Checking WiFi connection
status...</body></html>");
}
});

server.on("/settings", HTTP_GET, []() {
String html = "<html><head>"
"<style>"
"body {"
" display: flex;"
" justify-content: center;"
" align-items: center;"
" min-height: 100vh;"
" margin: 0;"
" text-align: center;"

```



```

    }"
    "form {"
    "  display: inline-block;"
    }"
  "</style>"
  "</head><body>"
  "<form action='/submit' method='post'>"
  "<h1>Plant Settings</h1>"
  "Select plant type: "
  "<select name='plantType'>"
  "<option value='Lilly'>Lilly</option>"
  "<option value='Cactus'>Cactus</option>"
  "<option value='Ficus'>Ficus</option>"
  "</select><br>"
  "Email: <input type='email' name='email' required><br><br>"
  "<input type='submit' value='Submit'>"
  "</form>"
  "</body></html>";
  server.send(200, "text/html", html);
});

server.on("/submit", HTTP_POST, []() {
  selectedPlantType = server.arg("plantType");
  enteredEmail = server.arg("email");
  if (selectedPlantType == "Lilly"){
    MOISTURE_UPPER = 1500;
    MOISTURE_LOWER = 2500;
    TEMPERATURE_UPPER = 30;
    TEMPERATURE_LOWER = 20;
    SUNLIGHT_UPPER = 2200;
    SUNLIGHT_LOWER = 0;
  }
  else if(selectedPlantType == "Ficus"){
    MOISTURE_UPPER = 4096;
    MOISTURE_LOWER = 2200;
    TEMPERATURE_UPPER = 30;
    TEMPERATURE_LOWER = 20;
    SUNLIGHT_UPPER = 2200;
    SUNLIGHT_LOWER = 4000;
  }
  else if (selectedPlantType == "Cactus"){
    MOISTURE_UPPER = 4095;

```



```

MOISTURE_LOWER = 2900;
TEMPERATURE_UPPER = 30;
TEMPERATURE_LOWER = 20;
SUNLIGHT_UPPER = 4095;
SUNLIGHT_LOWER = 4000;
}
setupComplete = true;
lastMeasurementTime = 0;
sendEmail = true;

String html = "<html><head><meta http-equiv='refresh'
content='5;url=/summary'></head><body>Setup complete. Redirecting to summary
page...</body></html>";
server.send(200, "text/html", html);
});

server.on("/summary", HTTP_GET, []() {
String temperatureHtml = "Not available";
String sunlightHtml = "Not available";
String moistureHtml = "Not available";

if (readingIndex > 0) {
sunlightHtml = (sunlightReadings[readingIndex - 1] >= SUNLIGHT_LOWER &&
sunlightReadings[readingIndex - 1] <= SUNLIGHT_UPPER)
? "<span style='color:green;'>In Range</span>"
: "<span style='color:red;'>" + String(sunlightReadings[readingIndex - 1] <
SUNLIGHT_LOWER ? "Below Range" : "Above Range") + "</span>";

moistureHtml = (moistureReadings[readingIndex - 1] >= MOISTURE_LOWER &&
moistureReadings[readingIndex - 1] <= MOISTURE_UPPER)
? "<span style='color:green;'>In Range</span>"
: "<span style='color:red;'>" + String(moistureReadings[readingIndex - 1] <
MOISTURE_LOWER ? "Below Range" : "Above Range") + "</span>";
}

String html = "<html><head>"
"<style>"
"body {"
" display: flex;"
" flex-direction: column;"

```




```

" justify-content: center;"
" align-items: center;"
" min-height: 100vh;"
" margin: 0;"
" text-align: center;"
}"
"button {"
" margin: 10px;"
}"
"</style>"
"</head><body>"
"<h1>Summary</h1>"
"Connected SSID: " + WiFi.SSID() + "<br>"
"Plant Type: " + selectedPlantType + "<br>"
"Email: " + enteredEmail + "<br><br>"
"Current Temperature: " + temperatureHtml + "<br>"
"Sunlight Status: " + sunlightHtml + "<br>"
"Moisture Status: " + moistureHtml + "<br>"
"<p>Refresh this page for the most up to date plant conditions!</p>"
"<button onclick=\"location.href='/settings'\">Edit Settings</button><br>"
"<button onclick=\"location.href='/check-connection'\">Check WiFi
Connection</button><br>"
"<button onclick=\"location.href='/restart-setup'\">Restart Setup</button>";

if (reservoirFlag) {
html += "<br><button onclick=\"location.href='/refill-reservoir'\">Reservoir Was
Refilled</button>";
}

html += "</body></html>";
server.send(200, "text/html", html);
});

server.on("/restart-setup", HTTP_GET, []() {
setupComplete = false;
connectedSSID = "";
enteredEmail = "";
selectedPlantType = "";

server.setHeader("Location", "/", true);
server.send(303);
});

```



```
server.on("/refill-reservoir", HTTP_GET, []() {
  reservoirFlag = false;
  server.setHeader("Location", "/summary", true);
  server.send(303);
});

server.begin();

}

void loop() {
  server.handleClient();
  if (sendEmail == true) {
    SendEmail();
  }
  if ((setupComplete == true) && (millis() - lastMeasurementTime >=
  MEASUREMENT_INTERVAL)) {
    lastMeasurementTime = millis();
    if (readingIndex < ARRAY_SIZE) {
      MeasureTemp();
      MeasureMoisture();
      MeasureSunlight();
      readingIndex++;
    }
    if (readingIndex == ARRAY_SIZE) {
      float averageTemp = ProcessReadings(tempReadings);
      float averageMoisture = ProcessReadings(moistureReadings);
      float averageSunlight = ProcessReadings(sunlightReadings);
      readingIndex = 0;
      if ((averageTemp > TEMPERATURE_UPPER) || (averageTemp < TEMPERATURE_LOWER)){
        temperatureFlag = true;
      }
      if ((averageSunlight > SUNLIGHT_UPPER) || (averageSunlight < SUNLIGHT_LOWER)){
        sunlightFlag = true;
      }
      if ((averageMoisture > MOISTURE_UPPER) || (averageMoisture < MOISTURE_LOWER)){
        moistureFlag = true;
      }
    }
    if (temperatureFlag || sunlightFlag || moistureFlag) {
```

**Plant – Final Report**

```
SendStatusEmail();
}

Serial.print("Average Temperature: "); Serial.println(averageTemp);
Serial.print("Average Moisture: "); Serial.println(averageMoisture);
Serial.print("Average Sunlight: "); Serial.println(averageSunlight);
}
}
}

float ProcessReadings(int readings[]) {
int maxIndex = 0, minIndex = 0;
for (int i = 1; i < ARRAY_SIZE; i++) {
if (readings[i] > readings[maxIndex]) maxIndex = i;
if (readings[i] < readings[minIndex]) minIndex = i;
}

int sum = 0;
for (int i = 0; i < ARRAY_SIZE; i++) {
if (i != maxIndex && i != minIndex) {
sum += readings[i];
}
}
return (float)sum / (ARRAY_SIZE - 2);
}

void MeasureTemp(){
sensors_event_t humidity, temp;
uint32_t timestamp = millis();
if (! sht4.begin()) {
Serial.println("Couldn't find SHT4x");
}
else{
sht4.getEvent(&humidity, &temp);

int temperature_reading = temp.temperature;
int humidity_reading = humidity.relative_humidity;

tempReadings[readingIndex]=temperature_reading;
Serial.println("Temperature value was stored");
```



```
delay(3000);
}
}

void MeasureSunlight() {
int sensorValue = analogRead(SUNLIGHT_ANALOG_PIN);
sunlightReadings[readingIndex] = sensorValue;
readingIndex++;
delay(3000);
}

void MeasureMoisture() {
int sensorValue = analogRead(MOISTURE_ANALOG_PIN);
if ((sensorValue < MOISTURE_LOWER) && (!reservoirFlag)){
WaterPlant();
Serial.println("Moisture was too low, watered the plant");
}

if (readingIndex >= 2) {
if (moistureReadings[readingIndex - 1] <= moistureReadings[readingIndex - 2] &&
sensorValue <= moistureReadings[readingIndex - 1]) {
reservoirFlag = true;
SendStatusEmail();
Serial.println("Water reservoir is empty, email was sent");
return;
}
}

moistureReadings[readingIndex] = sensorValue;
Serial.println("Moisture Value was stored");
delay(3000);
}

void WaterPlant(){
delay(5000);
digitalWrite(PUMP_ACTIVATION_PIN, HIGH);
delay(1000);
Serial.println("Pump on");
delay(3000);
digitalWrite(PUMP_ACTIVATION_PIN, LOW);
Serial.println("Pump off");
}
```



```
}

void SendEmail() {
  if (!smtp.connected()) {
    if (!smtp.connect(&session)) {
      Serial.println("Failed to connect to SMTP server.");
      return;
    }
  }

  SMTP_Message message;
  message.sender.name = "Plant Monitor";
  message.sender.email = sender_email;
  message.subject = "Plant Monitor Confirmation Email";
  message.addRecipient("User", enteredEmail);

  String htmlMsg = "<div style=\"color:#000000;\"><h1>Email Confirmation!</h1><p>Mail
Generated from ESP32</p><p>Connected WiFi SSID: " + connectedSSID + "</p><p>Selected
Plant Type: " + selectedPlantType + "</p></div>";
  message.html.content = htmlMsg.c_str();
  message.text.charset = "us-ascii";
  message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

  if (!MailClient.sendMail(&smtp, &message)) {
    Serial.println("Error sending Email, " + smtp.errorReason());
  }

  sendEmail = false;
}

void SendStatusEmail() {
  if (!smtp.connected()) {
    if (!smtp.connect(&session)) {
      Serial.println("Failed to connect to SMTP server.");
      return;
    }
  }

  SMTP_Message message;
  message.sender.name = "Plant Monitor";
```



```
message.sender.email = sender_email;
message.subject = "Plant Monitoring Alert";
message.addRecipient("User", enteredEmail);

String htmlMsg = "<div style=\"color:#000000;\"><h1>Plant Monitoring
Alert</h1><p>Issues detected with your plant settings:</p><ul>";

if (temperatureFlag) {
htmlMsg += "<li>Temperature out of desired range.</li>";
}
if (sunlightFlag) {
htmlMsg += "<li>Sunlight levels are not optimal.</li>";
}
if (moistureFlag) {
htmlMsg += "<li>Moisture levels need attention.</li>";
}
if (reservoirFlag) {
htmlMsg += "<li>Water reservoir needs to be refilled.</li>";
}

htmlMsg += "</ul><p>Please check your plant's environment to ensure it is
optimal.</p></div>";

message.html.content = htmlMsg.c_str();
message.text.charset = "us-ascii";
message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

if (!MailClient.sendMail(&smtp, &message)) {
Serial.println("Error sending Email, " + smtp.errorReason());
}

temperatureFlag = false;
sunlightFlag = false;
moistureFlag = false;
}
```



TEPT4400

Adafruit 4547 water pump

SHT4x

ESP32-C3-WROOM-02

CP2102

AP2112 voltage regulator

NE555P Precision Timer

LMV358ID