Plant Health Monitor

By

Tommy Bahary

Dishen Majithia

Esteban Roberts

Tbahary2; Dishenm2; Enr2

Final Report for ECE 445, Senior Design, Spring 2021

TA: Haoqing Zhu

5 May 2021

Project No. 51

Abstract

This report describes a plant health monitoring device that was developed and tested. This device utilized a temperature sensor, moisture sensor, pH sensor and light intensity sensor to record plant health metrics. These metrics were then sent to a web server where they were displayed to the end user. Additionally a pump was included to actively adjust the moisture level of the plant to reach a specific target. A battery system was also included to allow the design to be more versatile. The final result included a functioning web server which updated in real time with accurate sensor readings, with the exception of pH. The pump system was able to adjust the moisture level to the specified target, but the battery systems suffered reliability issues.

Contents

1. Introduction

1.1 Objective

The main issue with growing plants at home is monitoring their health. Plants regularly dry out or even die if not given the proper attention. Many people do not know the proper way to care for a plant and this project attempts to remedy this problem by giving the grower precise measurements about the plant's health and automating repetitive tasks.

We have designed a multi sensor device that can accurately measure various soil health indicators in real time, so the user can get live feedback and adjust the conditions accordingly. The device also allows watering of the plant automatically when needed. This provides the user with a way to conveniently check on and maintain their plants even when they are not around.

A similar product exists on the market but does not measure pH and only works with Bluetooth [1]. Our project differentiates itself by transmitting data remotely with WiFi so that the data is displayed on a web server. This data would provide the user with a way to conveniently check on their plants even when they are not around, as opposed to Bluetooth which only works when the device is nearby. Our automatic watering system also allows the user to leave the plants when they are traveling or away from home for an extended period of time.

1.2 Background

Growing plants at home is a hobby that many people enjoy but maintaining the health of the plants is not always easy. The main problem with plant health usually comes in the form of the soil. This includes bad pH levels and under/over watering the soil. Temperature and insufficient lighting can also be a factor. Our project would remedy this by monitoring each of these conditions. This way people do not have to do their own research into the more difficult aspect of plant growing, soil health. An additional concern for houseplants is travel. Being away from home for extended periods can leave plants dry and withered from the lack of watering and basic care. This project also attempts to remedy that by having an auto watering system and allowing the user to monitor the plant from anywhere.

1.3 High-Level Requirements

- Must obtain information from the various sensor modules and relay information to the user through WiFi and status lights at least every 5 minutes.
- Must be able to automatically adjust the moisture level of the soil to achieve a target level within ±10% in 1 hour.
- Must be able to freely switch the power source from an outlet to a battery that can individually provide at least 3-5 days of functionality.

2. Design

The project begins with the power supply. This takes in power from either the Lithium Polymer battery or the outlet. This outputs 3.3 V to the MCU and the battery voltage to the pump driver. From there, the driver turns the water pump on and off when given the signal from the MCU. The MCU is connected to the sensors and both receives and processes the information that the sensors output. After receiving the information from the sensors, the MCU sends the signal to the driver to water the plant if necessary, and then sends all the relevant information to the web server by sending an HTTP Request. These block interactions are shown below in Figure 1.

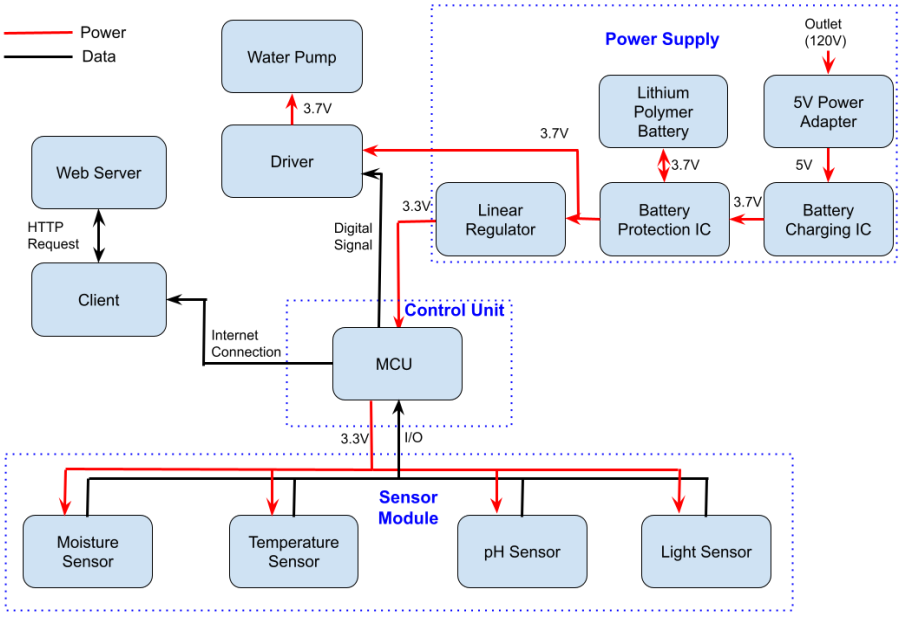

Figure 1, Block Diagram

2.1 Control Unit

The control unit primarily consists of the MCU and its supporting components. This block interacts with every other one in the system. It receives 3.3 V from the power system, takes data from each of the sensors, transmits this data to the web server, and uses the data to control the pump.

2.1.1 Microcontroller

The ESP32 S2 [2] is used in the control unit as the MCU. This has many advantages when compared to other WiFi capable MCUs. It is incredibly simple to implement. The WROOM module being used [3] contains memory, an oscillator and a WiFi antenna which greatly simplifies the design. It is also one of the cheapest WiFi capable devices, at $2. The ESP32 S2 module implementation can be seen below in Figure 2.

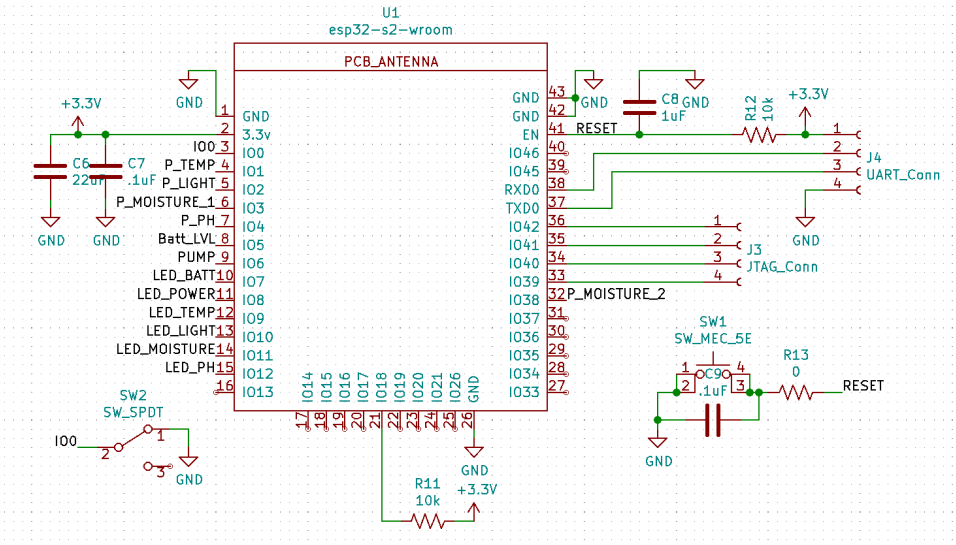

Figure 2, Control Unit Schematic

2.2 Power Subsystem

The power subsystem consists of several components which provide power to the system. It supplies 3.3 V for the microcontroller and sensors, and the battery voltage for the pump. This system is designed to power the entire project for up to 5 days using the battery. It can also be powered directly from USB. The transition between these states was designed to be seamless so that no interruption of power occurs. This is accomplished by using several components, whose interaction can be visualized in the 'Power Supply' section of Figure 1. An off the shelf 5 V USB adaptor is used to power the system. This power is supplied to the battery charging circuit which regulates it to the battery voltage. This is used to charge the battery and power the system. A battery protection circuit is placed in between the lithium polymer battery and the rest of the circuit, and is designed to disconnect the battery in conditions that would cause damage. Finally a linear regulator regulates the battery voltage to 3.3 V which is supplied to the MCU, while the battery voltage is supplied directly to the pump.

2.2.1 5V USB Adaptor

A 5 V USB adaptor in conjunction with a micro USB connector is used in this design for simplicity. These components are standardized and commonplace. This provides convenience for the user while supplying the ideal voltage for lithium polymer battery charging. It also makes this one of the safest ways to power electronics from a wall plug.

2.2.2 Battery Charger

The primary component of the battery charging circuit is the TI BQ24050 battery charging IC [4]. This IC has several useful features which make it ideal for this use case. The first of which is the USB adaptor detection. USB ports from computers could be connected to this charging circuit, which sources up to 1 A from USB. These computer USB ports can only supply 100 mA in some cases, so detection of these ports is important to prevent overloading them. The circuit is also designed to be able to supply power to the system while the battery is charging. This is important for allowing a seamless transition from

battery charging to discharging, without interrupting the device operation. The completed circuit is shown below in Figure 3.

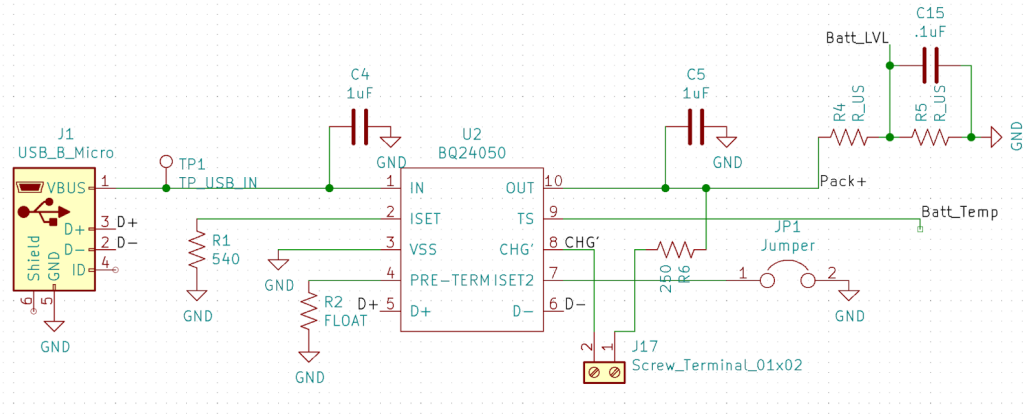

Figure 3, Battery Charger Circuit Schematic

Several design decisions have been made to ensure that the circuit performs optimally for this project's use case. The first of which is the ISET resistor R1, which is sized to 540 Ω to set the charge current to 1 A. This is the maximum possible for this IC, and is well below the battery's limit. In this case, faster charging times maximize convenience for the user at no additional cost. The ISET2 pin is used to determine the charge current once the USB source is determined. Shorting the pin to ground defaults to the ISET current which is the standard use case. However a jumper was added for debugging purposes. Similarly the PRE-TERM resistor determines the current in precharge and termination modes. This does not need to be changed from the default and can be left floating, but a space for R2 was added for debugging purposes.

2.2.3 Battery Protection

The battery charging circuit has features built in to prevent overcharging of the battery and overcurrent while charging. No such protections exist for discharging the battery. Over discharging a lithium battery can destroy the battery and potentially cause fires. This could potentially occur if the circuit continues to draw power from the depleted battery. The battery protection circuit prevents this overdischarge case and adds a second layer of protection while charging. In all, the battery protection circuit prevents overcharging, overdischarge, and overcurrent in both directions. The circuit is shown below in Figure 4.

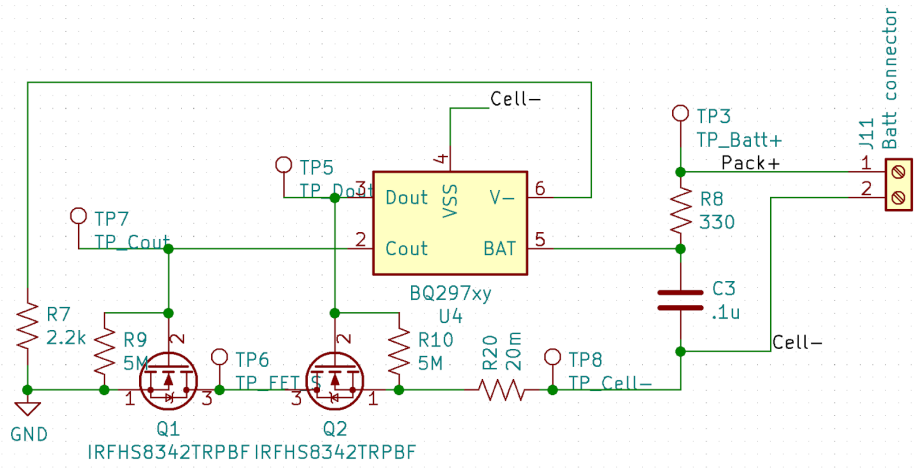

Figure 4, Battery Protection Circuit Schematic

In this circuit the TI BQ29704 battery protection IC [5] controls two MOSFETS. The MOSFET connected to the Cout pin is turned off to prevent charging, and the Dout MOSFET can be turned off to disable discharging. The Cout MOSFET is turned off when battery voltage dips below 2.5 V or charge current exceeds the thresholds. This prevents further charging. The Dout MOSFET is turned off when battery voltage exceeds 4.425 V or charge discharge current exceeds 1.5 A. This prevents further discharging. Voltage thresholds are based on the part number of the device. Current is measured by using the voltage between GND and Cell-. When this voltage exceeds 1 V during charging or goes below -1.25 V while discharging, the overcurrent protection is tripped. Current cannot be allowed to exceed 2 A in either direction due to the specifications of the battery. Charge current is expected to reach 1 A during standard operation, while discharge current must be able to reach 1.5 A. The IRFHS8342TRPBF MOSFETS [6] and R20 have been sized so that this overcurrent voltage reaches 1 V with a discharge current of 1.5 A. This means that overcurrent protection during discharge will trip at 1.5 A. Overcurrent protection during charge has a wider range for thresholds due to the lower max charge current of 1 A, and will therefore trip at 1.2 A.

2.2.4 Lithium Polymer Battery

A 2000 mAh lithium polymer battery is used in this design [7]. The sizing of this battery is primarily constrained by the max current that it could output. The estimated battery capacity requirement for 5 days of standard operation is about 950 mAh. This uses the assumption that the 3 W 200 L/h pump will supply 1 liter per day. It is also assumed that for every 5 minute interval, the MCU transmits over WiFi for 10 seconds, operates normally for 10 seconds, and is in light sleep for the rest of the interval. These numbers are used in conjunction with datasheet current specifications [2] to estimate the required capacity below in Equation (1), where $Capacity_{pump}$ is the estimated capacity required by the pump, $Capacity_{MCU}$ is the estimated capacity required by the MCU, and $Capacity_{total}$ is the total capacity estimate.

$$6\ Wh/200\ L\ = .03\ Wh/day$$
$$.03\ Wh\ /\ 3.3V\ = 9.09\ mAh/L\ =\ 9.09\ mAh/day$$
$$Capacity_{pump}\ =\ 9.09\ mAh/day\ *\ 5\ days = 45.45\ mAh$$

$$I_{avg} = 200\frac{10}{300} + 12\frac{10}{300} + .45\frac{280}{300} = 7.486\ mA \quad (1)$$
$$Capacity_{MCU}\ =\ 7.486mA\ *\ 5\ days\ *\ 24\ hours = 898.4\ mAh$$

$$Capacity_{total}\ = 898.4 + 45.45 = 943.85\ mAh$$

An estimate for max battery discharge current is calculated below in Equation (2) using similar power use information as Equation 1 with I_{pump}, I_{MCU} and $I_{total_{max}}$ being the maximum current use by the pump, MCU and system, respectively. This max current is the limiting constraint on the battery. Batteries smaller than 2000 mAh typically cannot source the 1500 mAh needed. The extra battery capacity is also useful to ensure that the high level requirements are met.

$$I_{pump} = 3W/2.8V\ =\ 1071\ mA$$
$$I_{total_{max}} = I_{pump} + I_{MCU} = 1071\ mA\ +\ 200\ mA = 1271\ mA \quad (2)$$

2.2.5 Linear Regulator

The LP38692MPX-3.3 linear regulator [8] is used in this design. The battery voltage can reach up to 4.2 V in standard operation. The linear regulator will dissipate the power and output 3.3 V for the MCU. The regulator is rated for 1 A and outputs at a fixed voltage of 3.3 V. Therefore the circuit simply consists of two filter capacitors and the regulator.

2.3 Sensors

The sensor subsystem consists of the 4 main sensor modules that each measure a different value, moisture, light, temperature, and pH. These four sensors work with the MCU to receive instructions and send data to the web server and the pump module. Out of these 4 sensors, the pH and moisture sensors are in the form of probes that need to be pressed into the soil to get measurements. The light and temperature sensors are over the board and need to be mounted onto the PCB.

2.3.1 Moisture Sensor

The Moisture sensor was made from scratch using fondue forks attached to terminal blocks as probes. We tested with 2 different types of moisture sensors, Capacitive and Resistive. The advantage of a Capacitive moisture sensor is that it only has current through the probes when measurements are being taken. This reduces the effects of electrolysis on the probes which increases their lifetime as they corrode less. On the other hand, Resistive Moisture sensors are basically a voltage divider circuit with a fixed resistor and the resistance between the two probes with the soil acting as the medium, which means they have a current passing through them at all times. This makes them more prone to corrosion. For this reason, we were hoping to use the capacitive moisture sensor as our main module. The circuit for this is shown in Figure 5, and is just a RC circuit that makes the probes mimic a capacitor's plates with the soil posing as the dielectric [9]. However, we found that the effect of soil packing and type was very

high in the capacitive moisture sensors. For the same sample of soil with the same moisture level, we got different results between loosely packed and tightly packed soil. This made this sensor extremely hard to calibrate, since the variations in moisture level for different conditions were high and gave very inconsistent readings.

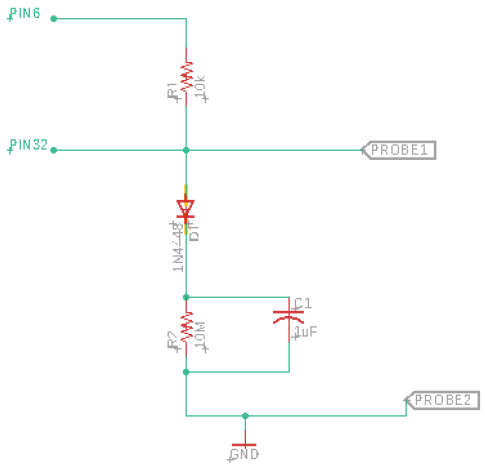

Figure 5 , Capacitive Moisture Sensor Circuit Schematic

Because of these inconsistencies, we also assembled the resistive moisture sensor and ran tests on it. The circuit for this is shown in Figure 6. The results for this were better than the capacitive sensor because we were able to get more uniform readings across the same soil sample without large variations based on the packing. After these tests, we decided to use the resistive moisture sensor as our main module.

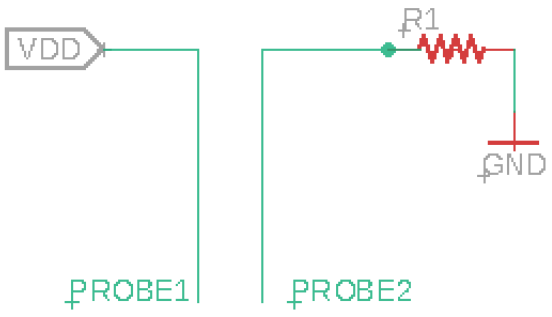

Figure 6, Resistive Moisture Sensor Circuit Schematic

2.3.1 Temperature Sensor

The temperature sensor we used is the DS18B20 OneWire sensor [10]. This package has a data pin that gives the necessary information that can be accessed using the DallasTemperature [11] and OneWire [12] libraries in the Arduino IDE. This proved to be a very useful tool for the process since it simplified

the circuit greatly. We are also using the through hole package instead of the waterproof temperature probe to reduce costs. We found that the ambient temperature in most indoor rooms accurately reflects the temperature of the soil since there are usually no extreme changes in temperature during short time intervals.

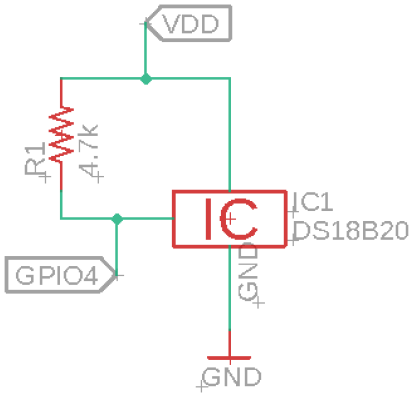

Figure 7, Temperature Sensor Circuit Schematic [13]

2.3.1 Light Sensor

The light sensor is a photoresistor in a voltage divider configuration with a 1.5k resistor as shown in Figure 8. The resistor is a constant value while the photoresistor is a variable resistance that changes with the amount of incident light. We use this to get values of the brightness levels around the plant, and calibrate it so that the maximum brightness would be a 100 and the minimum would be 0. These upper and lower bound values are not exact, since it's very hard to create a perfectly dark and fully bright environment around the sensor, but they are good approximations and give a relatively accurate depiction of the brightness levels.

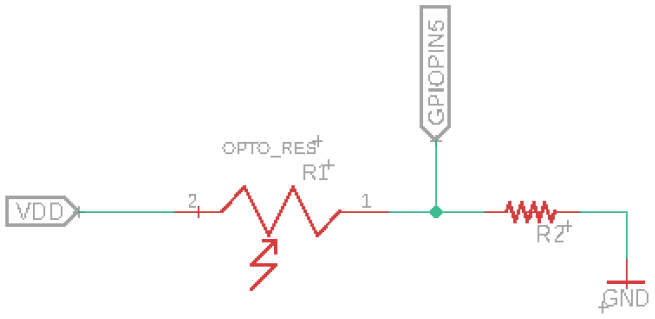

Figure 8, Light Sensor Circuit Schematic

2.3.1 pH Sensor

For the pH sensor, most of the options available to us were very expensive. Since we did not have the time, resources, or the expertise required to make our own, we decided to use a cheaper sensor[14] that we would reverse engineer to work with our circuits. The sensor that we got, however, ended up breaking after just a few uses. We also did a decent amount of testing on this sensor to try and identify what the signals from its output wires looked like and whether they were discernible for samples with known pH levels, and found that that was not the case. Because of these issues, we decided to scrap the pH sensor and focus on the other parts of the project.

2.4 Pump

For the pump circuit, we are using an underwater submersible pump with a 3.5-9 V input voltage range [15]. This pump is connected to a MOSFET based switching circuit shown in Figure 9 with a PWM signal generated by the MCU as the gate signal. The built-in PWM generator was not working well on the ESP, however, so we manually generated the PWM signal using digitalWrite and delay functions. This gate signal turns on the pump for 5 seconds every time a measurement is taken by the moisture sensor that falls below the threshold set by the user. The 5 second time period was chosen so that the pump does not push the moisture level too high above the threshold, since it would wait till the next reading is taken before getting the on/off signal again. This way, the circuit would increase the moisture level incrementally until it reaches the threshold, and would also give the water enough time to seep across the entire soil sample before taking another moisture reading.

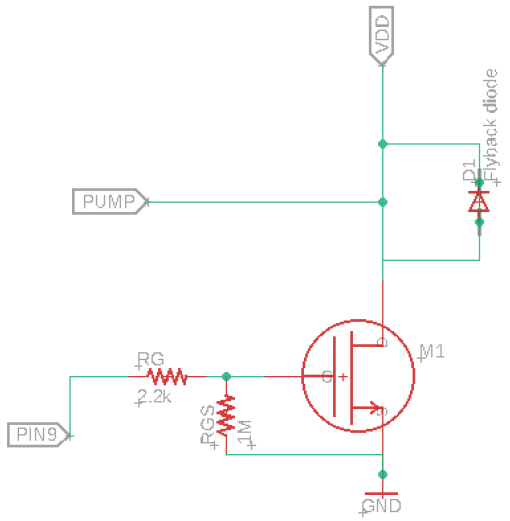

Figure 9, Pump Control Circuit Schematic

2.5 Web Server

The web interface was originally a simple web server created by the microcontroller that displayed the sensor values and updated with the manual refreshing of the page. We decided to make the web server asynchronous instead, so that the user could read sensor changes in real time [16]. To make the change from regular web server to asynchronous, a github library called ESPAsyncWebServer was needed [17].

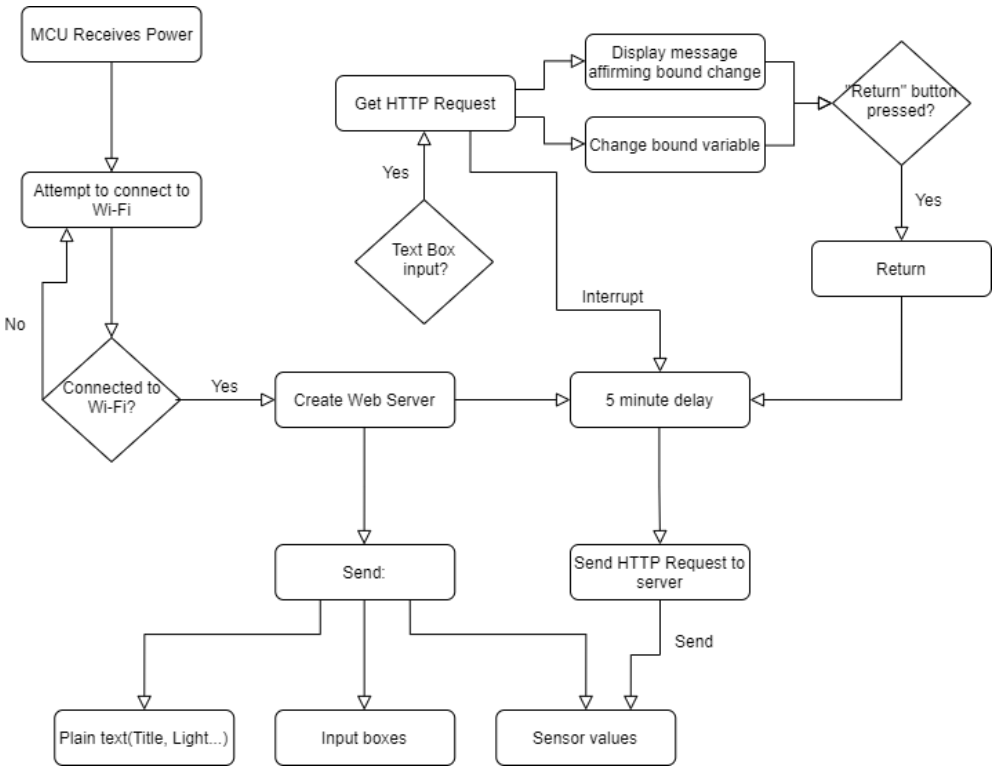

Figure 10, Web Server Flowchart

The ESP32-S2 was chosen for its WiFi capabilities. It can also create its own web server through WiFi. As shown in Figure 10, once the microcontroller connects to WiFi and creates a web server, it must send the server data to display. This includes the plain text(Title, sensor names, images), the input boxes, and the first sensor readings. Then, after a 5 minute delay all of these readings are updated with the use of HTTP requests. HTTP requests were used for the sensor values, input boxes, and error messages. The error messages are shown when any sensor value goes outside of the given bounds.

Outside of the main loop, there is an interrupt for the input boxes. If a value is typed into any of the input boxes, the microcontroller first gets an HTTP request and interrupts the main loop. The web server will display a message that states what the new bound for the given sensor is and it changes the variable that holds the value. If a return button is pressed, then the server returns to the main loop. The choice of having the message appear and having an interrupt was a design one. It is not necessary to do but it gives the user instant feedback on what the new value of the bound is and it also lets the user know if there was an error updating the bound.

2.6 PCB

The PCB incorporates all portions of the project onto a 5x10 cm area. This can be connected to the external pH and moisture sensors. There is also a connector for the battery and the micro USB charging cable. The board is divided into three sections, as shown in Figure 11. Several design decisions were taken to make the PCB configuration more practical. First, components were condensed in order to make the PCB as small as possible, which would make it more practical to use with smaller plants. Unfortunately this made working on the PCB more difficult and complicated debugging. Another design choice was to include a ground plane on the backside of the PCB. This helps simplify the routing of traces on the PCB which reduces complexity and noise. The large ground plane is also used by the MCU and battery charger for thermal dissipation. This is very important for the battery charger because in some cases it needs to dissipate power in excess of 1 W. Additionally the MCU module was placed with the antenna area outside of the PCB, as recommended by the ESP32 S2 hardware design guidelines [18]. This helps prevent interference between the copper ground plane and the WiFi signals being transmitted by the antenna area.

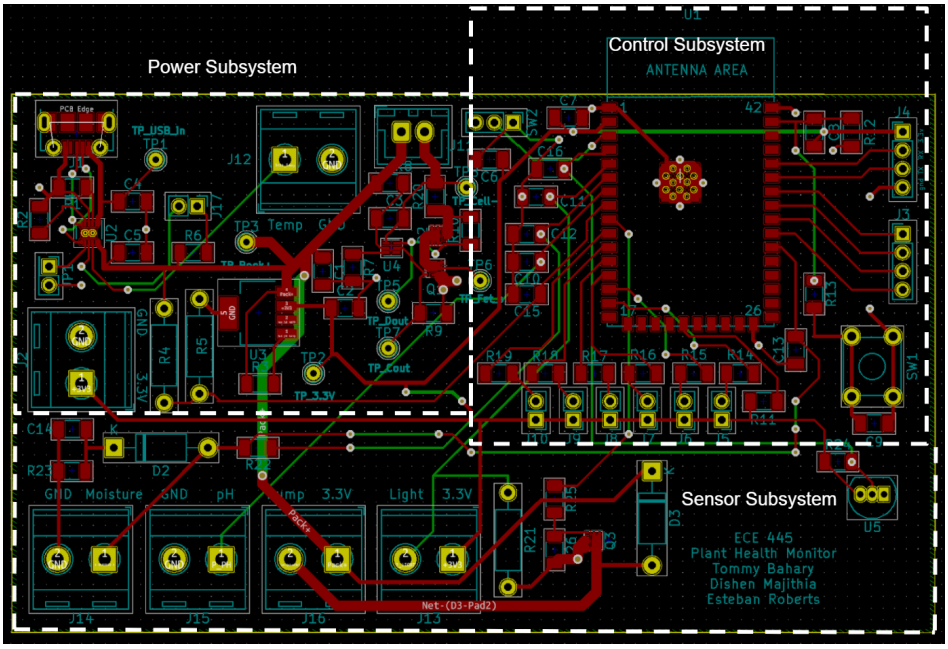

Figure 11, PCB Design [19]

Only the verification and testing of the power system was completed on the PCB. All other portions of the PCB were assembled, but were ultimately verified on a breadboard. There are no issues with these portions of the PCB and the circuits are the same as those on the breadboard, but there simply was not enough time to implement and test these on the PCB.

3. Design Verification

3.1 Control Unit

For the control unit, all the requirements were tested while verifying other components and subsystems. For instance, the functionality was tested for each sensor with the MCU and the web server, and since those requirements were verified, the control unit was also verified along with it. Similarly, the control unit WiFi functionality was tested while connecting and testing the web server. The only part of the control unit that we couldn't verify was the LED portion. These LEDs were meant to flash red if the sensor level they corresponded to was out of bounds. We were unable to test this because we ran out of time, and had to ultimately remove this part of the project.

3.2 Power System

3.2.1 Battery Protection Circuit

The battery protection circuit was the first system to be tested. This component needed to be verified first independently of the battery to ensure safety while performing charging and discharging tests. For overcharge protection a power supply was connected to the battery load end to simulate the battery charge voltage, with an artificial load connected to the battery terminals. The simulated charge voltage was raised starting from 3.5 V until Cout turned off and charge current reached 0 A. In testing, this occurred at 4.5 V. Overcurrent during charging was tested with a similar setup. The current setting on the artificial load was incremented until the Cout MOSFET turned off. This occurred at 1.5 A.

The discharge states were tested in a similar manner, but with the load and the source switched. A power supply was connected to the battery terminals to simulate the battery voltage, with an artificial load connected to the battery load. The simulated battery voltage was lowered starting from 3.5 V until the Dout MOSFET turned off. In testing this occurred at 2.7 V. A similar test was performed for the overcurrent case, where the current was adjusted at the artificial load until Dout turned off. This also occurred at 1.5 A.

These voltages and currents vary slightly from those specified in the R&V tables. This inconsistency is due to stock issues requiring a change in BQ297XX part number which changed the built-in threshold voltages. Despite this, the parameters are still within the safe range for the battery and will not disrupt typical operation of the circuit. Therefore these tests can be considered passing and successfully verified.

3.2.2 Battery Charger

The battery charger was first tested by powering the circuit through USB and connecting an artificial load set to a constant voltage of 3.7 V. The thermistor was also connected to best simulate a real battery. This gave very erratic results. At times 500 mA was sustained at 3.7 V. Other times no current flowed at all with a periodic square wave like waveform varying between 4.2 V and 1.3 V. Other occasions involved wildly varying voltages and currents going into the load. At any given time, each of these three scenarios were equally likely to occur, seemingly without reason. This was caused by the control scheme of the charging IC expecting the output voltage to be fixed by a battery. This was not the case because the load did not set the battery voltage unless current was applied, which caused erratic behavior of the charging

circuit. This was fixed by repeating the same tests with the thermistor disconnected. The IC was no longer expecting a battery to be connected and therefore acted accordingly. For this case the battery charger supplied 750 mA at 3.7 V for 10 minutes without fluctuations.

Next the lithium polymer battery with a voltage of 3.7 V was connected to the powered circuit through the battery protection IC. The thermistor was connected and attached to the battery. Current flowing into the battery was measured at 900 mA, but dipped to 700 mA at times. The voltage of the battery was slowly increasing as current went into the battery. This indicated that the battery was successfully charging. The battery remained near room temperature during charging, indicating that it was charging safely.

Unfortunately the battery charge tests were unable to be replicated successfully. Upon repeating these tests, no current flowed into the battery. Repeating the battery protection circuit verification revealed that the MOSFETS were not sufficiently turned on. Both gate voltages were measured at around 2.45 V for a battery voltage of 3.8 V, which kept the MOSFETS turned off or in the linear region which dissipated most of the power. According to the battery protection IC datasheet [5], this gate voltage will be above 3.4 V or below .4 V when the battery is at 3.8 V. Violation of this specification clearly indicates a potential malfunction of one or more components in the design. Replacing the MOSFETS and protection IC still failed verification, but yielded different results. After all of these parts were replaced, the gate voltage of Cout measured 3.5 V and Dout measured .4 V. This result indicates that there is a procedural failure occurring. Most likely Is that the components cannot be soldered properly without some damage occurring. These parts are difficult to solder because of their size, so attempts at soldering can expose them to high heat for long periods of time and cause damage. The next step in the debugging process would be to redesign the PCB and replace the small MOSFETS with much larger ones. Unfortunately there was not time to do this, so the final design needed to circumvent the battery, as described in section 3.2.3. As a result, battery discharge tests were unable to be performed safely. These portions of the R&V table therefore remain incomplete, despite the anticipated success of these verifications.

3.2.3 Power System Tests

The battery cannot be connected to the final version of this design, because circumventing the malfunctioning battery protection circuit would be unsafe. Therefore, all of the battery components were bypassed using a wire and the battery was removed from the system. The 5 V connection from the USB was directly connected to the battery output beyond the protection circuit. This treated the USB voltage as if it were output from the battery. The system functions as planned when connected to the micro USB cable, but will not operate without it. Varying loads were sourced from this system both at 3.3 V and 5 V in order to ensure that it could supply the amount of power needed.

Table 1, Power System Loads and Voltages

Load (mA)	Pump Voltage (V)	MCU voltage (V)
100	5.09	3.308
250	5.06	3.291

500	5.01	3.256
750	4.97	-
1000	4.91	-
1250	4.86	-
1500	4.80	-

The results of these tests are shown above in Table 1, and show that the power system can supply the required power without an impactful drop in voltage.

3.3 Sensors

3.3.1 Moisture Sensor

The first test for the moisture sensors was to check whether they show uniform readings within 10% of each other for the same soil samples. This test was passed by the resistive moisture sensor, but the capacitive one had larger variations. These variations only increased when the solid packing level was modified between readings, and when the readings were interspersed with readings of different samples in between. We also saw that we were getting the same reading that we got for the 'dry' measurement of the probes in air (4914) for a couple of the soil samples based on their packing and how the probes were placed. This led us to believe that the capacitive moisture sensor would be too unreliable for us to use in this project, and so we decided to use the resistive sensor.

Table 2, Resistive vs Capacitive Moisture sensors for different packing level(Not Calibrated)

Packing level	Resistive Moisture Sensor	Capacitive Moisture Sensor
1	1136	3276
2	1136	4914
3	1154	4914
4	1157	5014
5	1176	6240

The second test for the moisture sensor was to be able to send the values recorded by the sensors to the web server within 10 seconds of measurement. This requirement was tested along with the web server by making extreme changes to the probe conditions and seeing the delay. This was done by switching the probes from completely wet conditions in water to the completely dry conditions in air. After multiple attempts of doing this with the resistive moisture sensor, we were able to verify that the changes do indeed show up on the web server in the required delay time. This time can also be modified

in the code very easily by using the delay() function. We also made sure that the sensor measures data only once every 5 minutes, which was also easily verified using the web server and the delay function on the Arduino IDE.

3.3.2 Temperature Sensor

For the temperature sensor, the first test was whether it gives accurate temperature results. The only way for us to verify this was to have it run in a room with a controlled temperature using a thermostat. We found that the temperature reflected by the sensor matched the thermostat temperature fairly accurately. After this, we needed to check if changes in the temperature are reflected on the web server 10 seconds after the measurement. We tested this by taking the sensor and moving to the outdoors and back indoors and checking the readings. This was done on a fairly cold day where the temperature was under 60F and the indoor temperature was set to 74F. This change in temperature was reflected on the web server accurately.

3.3.2 Light Sensor

The main requirement for the light sensor was to accurately measure the light readings around the plant and send them to the web server within 10 seconds. This was tested in a very similar way to the moisture sensor where we made extreme changes to the light and monitored the web server for changes. This was done by placing our finger on the photoresistor top to make a dark environment and then flashing a torch on it to create a bright one. These changes in readings passed our test and were also how we calibrated the sensor readings. The second requirement was to have uniform results for the same conditions. This was tested by leaving the sensor in the same position and taking readings interspersed with covering the top of the sensor. We saw very uniform results for this test as well, as the sensor would return back to its original ambient value after the finger was moved.

3.3.3 pH Sensor

When we took the sensor apart, the signals from the output wires in the circuitry did not make sense and were indiscernible across samples of varying pH levels. They were also not signals that could simply be read by the ESP32's GPIO pins. Because of this, we were unable to get a pH sensor module working.

3.4 Pump

The requirement for this pump was to be able to turn on for a small interval based on a signal from a GPIO pin on the MCU. The interval was first set to be 10 seconds, but after testing the amount of water pumped out in that time, we found that it should be a little lesser, and changed it to 5 seconds. This was first tested by generating a PWM signal with a 10 second period and a 50% duty cycle from the MCU and sending it to the gate of the switching circuit MOSFET. This did indeed turn the pump on for the desired amount of time. The next test was to be able to send this signal to the pump based on the input from the moisture sensor and the threshold. This was verified by making a block of code that compares the moisture sensor value to the threshold and setting a flag for the PWM signal to be generated or not. This was also verified as the pump would wait until a reading from the sensor was found to be 'dry' before turning on for the 5 seconds, and then waiting for the next reading to be taken before turning on again.

3.5 Web Server

The web server worked perfectly given the requirements put forth as shown in Figure 12. Adding water to the plant, blocking the light sensor, and touching the temperature sensor all changed the values in under 10 seconds. The web server was also updated after a 5 second delay during the demonstration instead of the required 5 minutes so as to not waste time unnecessarily. If a sensor value is out of bounds, an error message appears. Figure 13 shows the error message for low moisture. The input boxes were not originally part of our project. They were later added as a stretch goal to allow the user the freedom to input the necessary bounds for their specific plant [20].

There were some issues with the microcontroller and the web server. The main issue was that the Arduino IDE does not support the ESP32-S2, only its older counterpart the ESP32. To fix this, we needed to download an unofficial library from github to allow the code to work on the S2 [21]. This gave rise to some other issues. A big problem was that for a couple weeks the microcontroller would not connect to WiFi. This was because the ESP32-S2 is sometimes unreliable in its connectivity. We discovered that after around 5 seconds, the microcontroller stopped trying to connect to WiFi. To fix this, we added a simple while loop which reinitialized the ESP's network settings if the microcontroller did not connect to WiFi after 8 seconds.

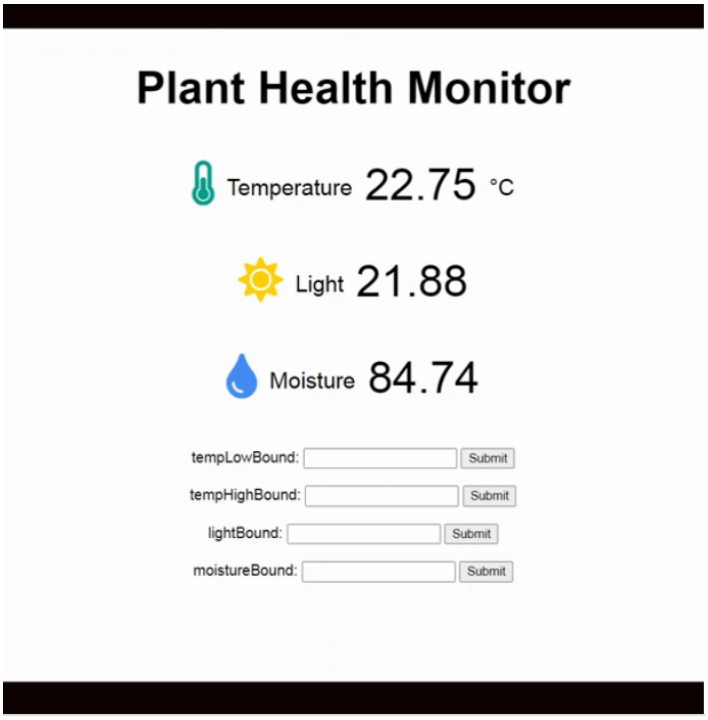

Figure 12, Web Server Design

Figure 13, Error Message

4. Costs

4.1 Parts

Table 3, Parts Costs

Part	Manufacturer	Retail Cost ($)	Total Retail Cost ($)	Bulk Purchase Cost ($)
MCU	Espressif systems	1.99	5.97	1.99
Battery Charger IC	Texas Instruments	1.36	3.97	0.561
Battery Protection IC	Texas Instruments	0.60	1.68	0.255
Linear Regulator	Texas Instruments	1.63	3.22	0.699
Lithium Polymer Battery	MikroE	12.5	12.5	12.5
Thermistor for battery	Semitec	2.25	2.25	0.335
Submersible water pump	Driew	19.49	19.49	19.49
Tube for pump	Uxcell	6.99	6.99	6.99
Photoresistor	Adafruit	0.95	2.85	0.95
Temp sensor	Maxim Integrated	5.27	5.27	5.27
pH sensor	Vivosun	10.9	10.9	10.9
Micro USB Type AB Connector	Molex	0.81	1.62	0.375
Battery connector	JST	0.15	0.45	0.05237
Terminal block x2	CUI Devices	0.259	3.735	0.155
SPDT switch	Nidec Copal	1.03	1.03	0.519
Reset Button	C&K	0.41	0.41	0.202
Green LED	Vishay	0.38	1.52	0.086
Red LED	Vishay	0.46	3.68	0.09
MOSFET	Infineon	0.51	3.06	0.158
Diode	ON Semiconductor	0.1	0.30	0.01
549Ω Resistor 1206	Bourns	0.1	0.30	0.007
250Ω Resistor 1206	ARCOL / Ohmite	0.18	2.7	0.069
5MΩ Resistor 1206	Susumu	0.57	2.28	0.153
330Ω Resistor 1206	Vishay / Dale	0.19	0.57	0.006
2.2kΩ Resistor 1206	Yageo	0.23	0.69	0.006
10kΩ Resistor	Vishay / Dale	0.33	1.65	0.039

1206				
1uF Capacitor 10V 1206	Vishay / Vitramon	0.19	1.9	0.11
.1uF Capacitor 10V 1206	Vishay / Vitramon	0.22	2.2	0.064
10uF Capacitor 10V 1206	Samsung Electro-Mechanics	0.3	0.90	0.046
22uF Capacitor 10V 1206	Samsung Electro-Mechanics	0.49	1.47	0.106

4.2 Labor

1. $47/hr x 2.5 x 120hrs = $14100 (Esteban - Computer Engineer)
2. $39/hr x 2.5 x 120hrs = $11700 (Tommy - Electrical Engineer)
3. $39/hr x 2.5 x 120hrs = $11700 (Dishen - Electrical Engineer)

Total cost of labor = $37,500

5. Conclusion

5.1 Accomplishments

We were able to accomplish most of the goals of this project, and some of the stretch goals as well. We were able to get the MCU and the sensors working very well with the web server. It displays accurate information for each sensor except for pH. We were also able to get the pump module working in conjunction with the moisture sensor readings. The web server was able to update information automatically. It even had the ability to accept user inputs as thresholds for each sensor. The power supply portion was also tested and verified, even though we had issues with reliability involving some of the components.

5.2 Ethical considerations

There are some ethical considerations from the IEEE Code of Ethics that we kept in mind while working on this project. The main one is #1: “to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment;” [22]. We know that there are certain harmful metals like cadmium and arsenic that can enter the soil or deposit itself on the plant through various sources like cigarette smoke and water. This is a big issue for home plants since a majority of these tend to be herbs and small vegetables and fruits that people use in their food. Exposure to such harmful materials can cause health complications to the grower in the long run [23]. These harmful substances are more prone to enter the soil when the pH is off balance. Our product aims to safeguard their health and safety by using the pH sensing technology to ensure that the soil pH is maintained at a safe 6.5-7 range.

The handling of lithium polymer batteries was another factor for the safety considerations of this project. Lithium based batteries may react violently if exposed to improper electrical, thermal or physical conditions. For this reason the course battery safety guidelines were followed at all times [24]. Additionally, the battery was never operated without the verified protection circuit. This led to the exclusion of the lithium polymer battery in the final design in order to avoid operating the battery without the protection circuit.

5.3 Future work

Our main goal for improving on this would be to have our entire project working on a PCB. Issues with integration and time constraints because of delayed parts and testing caused us to have to implement it on a breadboard. However, we did not find any issues with our PCB design, so our first goal would be to test the existing PCB using the same configurations as the breadboard. We also would like to have the pH sensor working either by purchasing a more reliable version, or by investing in and learning how to make our own. This would give the user some very important information about the soil and the plant’s health. We would also like to incorporate the pump module into the web server, since the delay on the HTTP requests was interfering with the delay on the PWM signal because it was manually generated. The fix for this is simple, as we just need to get a more reliable MCU with a PWM generator so we do not need to add manual delays that interfere with the web server.

References

[1] “Northfifteen Connected Home Plant Monitor.” [Online]. Available: https://www.amazon.com/Northfifteen-Connected-Home-Plant-Monitor/dp/B07L491F8K. [Accessed: 17-Feb-2021].

[2] Espressif Systems, “ESP32-S2 Family,” [PDF]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s2_datasheet_en.pdf. [Accessed 27-Feb-2021].

[3] Espressif Systems, “ESP32-S2-WROOM,” [PDF]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s2-wroom_esp32-s2-wroom-i_datasheet_en.pdf. [Accessed 27-Feb-2021].

[4] “BQ24050DSQR Datasheet,” Mouser Electronics. [Online]. Available: https://www.mouser.com/ProductDetail/Texas-Instruments/BQ24050DSQR/?qs=c7TMWs5treSEPMnWshozzw%3D%3D. [Accessed: 03-Apr-2021].

[5] Texas Instruments, “BQ297xx Cost-Effective Voltage and Current Protection Integrated Circuit for Single-Cell Li-Ion and Li-Polymer Batteries,” [PDF]. Available: https://www.ti.com/lit/ds/symlink/bq2973.pdf?ts=1614561305006&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ2973. [Accessed 28-Feb-2021].

[6] “IRFHS8342TRPBF Datasheet,” Mouser Electronics. [Online]. Available: https://www.mouser.com/ProductDetail/Infineon-Technologies/IRFHS8342TRPBF?qs=Z8%252BeY1k3TILJK9l3AOTdfA%3D%3D. [Accessed: 03-Apr-2021].

[7] MikroE, “Li-Polymer Battery 3.7V 2000mAh,” MikroElektronika. [Online]. Available: https://www.mikroe.com/li-polymer-battery-37v-2000mah. [Accessed: 03-Mar-2021].

[8] Texas Instruments , “LP38690, LP38692 1-A Low Dropout CMOS Linear Regulators,” 05-Jan-2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/lp38692.pdf?ts=1617580645405.

[9] Pedro52, “Arduino Capacitive Soil Moisture Sensor (DIY) with ESP32,” *Hackster.io*, 16-Nov-2020. [Online]. Available: https://www.hackster.io/Pedro52/arduino-capacitive-soil-moisture-sensor-diy-with-esp32-d7ad72. [Accessed: 20-Mar-2021].

[10] Mouser, “Programmable Resolution 1-Wire Digital Thermometer,” [PDF]. Available: https://www.mouser.com/datasheet/2/256/DS18B20-370043.pdf. [Accessed 3-Mar-2021].

[11] M. Burton, “Arduino-Temperature-Control-Library,” *GitHub*. [Online]. Available: https://github.com/milesburton/Arduino-Temperature-Control-Library. [Accessed: 18-April-2021].

[12] “OneWire,” *OneWire - Arduino Reference*. [Online]. Available: https://www.arduino.cc/reference/en/libraries/onewire/. [Accessed: 18-April-2021].

[13] R. T. (E. Engineer), “ESP32 DS18B20 Tutorial: DS18B20 Temperature Sensor with ESP32,” *Electronics Hub*, 27-Mar-2021. [Online]. Available: https://www.electronicshub.org/esp32-ds18b20-tutorial/. [Accessed: 05-May-2021].

[14] “GP,” *Amazon*, 2011. [Online]. Available: https://www.amazon.com/gp/product/B01DNUIGUY/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1. [Accessed: 05-Mar-2021].

[15] “Driew USB Water Pump, Water Pump, for Fountain Water Fountain Pump Submersible Water Pump 3W DC 3.5-9v (200L/H),” Amazon, [Online]. Available: https://www.amazon.com/Driew-3-5-9V-Submersible-Brushless-Waterproof/dp/B01CG2YE6K/ref=pd_ybh_a_8?_encoding=UTF8&psc=1&refRID=3G4V92NCQF15AVKP7S9B. [Accessed 27-Feb-2021].

[16] R. Santos, “ESP32 DHT11/DHT22 Web Server using Arduino IDE,” *Random Nerd Tutorials*, 07-Aug-2019. [Online]. Available: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/. [Accessed: 05-May-2021].

[17] Me-No-Dev, “me-no-dev/ESPAsyncWebServer,” *GitHub*. [Online]. Available: https://github.com/me-no-dev/ESPAsyncWebServer. [Accessed: 05-May-2021].

[18] Espressif Systems, “ESP32-S2 Technical Reference Manual,” [PDF]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s2_technical_reference_manual_en.pdf. [Accessed: 27-Feb-2021]

[19] Pratikmokashi, “pratikmokashi/ESP32-S2,” GitHub. [Online]. Available: https://github.com/pratikmokashi/ESP32-S2. [Accessed: 27-Feb-2021].

[20] R. Santos, “Input Data on HTML Form ESP32/ESP8266 Web Server Arduino IDE,” Random Nerd Tutorials, 30-Jul-2020. [Online]. Available: https://randomnerdtutorials.com/esp32-esp8266-input-data-html-form/. [Accessed: 05-May-2021].

[21] Espressif, “espressif/arduino-esp32,” GitHub. [Online]. Available: https://github.com/espressif/arduino-esp32. [Accessed: 05-May-2021].

[22] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 17-Feb-2021].

[23] E. W. Support, “Should I Worry about Heavy Metals in My Garden Soil?,” *OSU Extension Service*, 25-Feb-2021. [Online]. Available: https://extension.oregonstate.edu/gardening/techniques/should-i-worry-about-heavy-metals-my-garden-soil. [Accessed: 16-Feb-2021].

[24] University of Illinois, “Safe Practices for Lead Acid and Lithium Batteries,” [PDF]. Available: https://courses.engr.illinois.edu/ece445/documents/GeneralBatterySafety.pdf. [Accessed 28-Feb-2021].

Appendix A Requirement and Verification Table

* Further battery operation is unsafe due to malfunctioning battery protection circuit. Because of this portions of the circuit remain unverified. Specifically battery charger requirement 2 and battery requirement 3. See section 3.2 for details.

† Designates tests which were verified successfully at one point, but were not functioning in the final version of the project.

Control Unit

Requirement	**Verification**	**Verification status (Y or N)**
1. Must be able to transmit collected data over WiFi every 5 minutes in under 10 seconds.	1. a. Print to the console when data transmission over WiFi begins, confirm change to website occurs within 10 seconds. b. Make any significant change to a sensor condition, change will be visible on the website in less than 5 minutes and 10 seconds.	Y
2. Upon sensor reading, must be able to decide when values are not optimal and let the user know with the LEDs.	2. a. Connect a sensor ADC pin to a power supply, and sweep between 0-3.3V to simulate the entire range of sensor inputs. b. Using the console, take note of each data value upon sensor reading. c. When data is not within the programmed range for that sensor, the corresponding light will turn on. d. Repeat for each sensor.	N
3. Must be able to take readings from each of the sensors at least every 5 minutes.	3. a. Print sensor data to console to make sure that the values are changing over time. b. Take time between readings to prove that the readings are output every 300 seconds +/- 10.	Y

Battery Charger

Requirement	Verification	Verification status (Y or N)
1. Must be able to output at least a combined 750 mA for battery charging and circuit operation.	1. a. Connect circuit to battery and measure current to battery with a series ammeter. b. Connect a parallel load to the battery. Use an ammeter to measure current to this parallel load and esure these currents sum to be greater than 750mA. c. Battery must be sufficiently charged while these tests are performed, to ensure that the battery charger is outputting current for fast charging.	Y†
2. Battery must be able to source deficit current when load exceeds 750 mA up to a max of 1.5A, without voltage dipping below 3.3V	2. a. Connect a varying load to the output of the battery circuit while charging until an ammeter shows the load to be drawing 1.5A. b. Measure the voltage across the load to ensure that it is above 3.3V	*

Battery Protection

Requirement	Verification	Verification status (Y or N)
1. The battery must be disconnected and charging terminated when charge voltage exceeds 4.3V.	1. a. Use a power supply to slowly raise the voltage across the BAT pin starting from 3.5V. b. When voltage exceeds 4.3V, MOSFET with gate connected to COUT will turn off. This will be tested by using a series ammeter to confirm that the no current flows through the MOSFET past this point. c. The ammeter will also be used to confirm that no current flows through the battery past this point.	Y†
2. The battery must be	2.	Y†

disconnected and discharging terminated when charge voltage dips below 2.8V.	a. Use a power supply to slowly lower the voltage across the BAT pin starting from 4V while grounding the V- pin. b. When the voltage dips below 2.8V the MOSFET with gate connected to DOUT will turn off. This will be tested by using a series ammeter to confirm no current flows through the MOSFET past this point. c. The ammeter will also be used to confirm that no current flows through the battery past this point.	
3. The battery must be disconnected and charging terminated when charge current exceeds 2A.	3. a. Use a power supply to apply 3.7V across the input terminals. b. Apply a load where the battery would be in order to simulate battery charging, while using an ammeter to measure current. c. Increase the load until the current surpasses 2A, at which point the CHG MOSFET will be turned off. This will be tested by using a series ammeter to confirm no current flows through the MOSFET past this point. d. The ammeter will also be used to confirm that no current flows through the battery past this point.	Y†
4. The battery must be disconnected and discharging terminated when discharge current exceeds 2A.	4. a. Use a power supply to apply 3.7V	Y†

	across the terminals where the battery would be connected. b. Apply a load where the protection circuit would connect to the rest of the system in order to simulate battery discharging, while using an ammeter to measure current. c. Increase the load until the current surpasses 2A, at which point the DSG MOSFET will be turned off. This will be tested by using a series ammeter to confirm no current flows through the MOSFET past this point. d. The ammeter will also be used to confirm that no current flows through the battery past this point.	

Battery

Requirement	**Verification**	**Verification status (Y or N)**
1. Battery can charge with a current of at least 100 mA.	1. a. Provide charging circuit with power, and ensure that the battery has been plugged in long enough to initiate fast charging. b. Use a series ammeter to measure the current into the battery.	Y†
2. Battery can discharge with a current of at least 1500 mA.	2. a. Leave the charging circuit unpowered so that all output current comes from the battery. b. Use a series ammeter to adjust the load until a current of 1500 mA is reached.	Y
3. Battery can power the entire system for at least 3 days on a full charge.	3. a. Plug the entire system into USB adaptor until charging light turns off, indicating a full charge. b. Unplug system and leave operational for 3 days. c. Ensure the microcontroller is still powered by using a voltmeter to	*

	measure voltage across 3.3V pin.	

Linear Regulator

Requirement	Verification	Verification status (Y or N)
1. Linear regulator can convert an input voltage ranging from 3.3-4.2V to an output voltage of 3.3V +/- .1V at output currents up to 500mA.	1. a. Connect input of regulator circuit to a power supply. Vary supply voltage between 3.3 and 4.2V. b. Connect load to output of regulator and adjust until series ammeter current is 500mA. c. Measure the load voltage at this point to ensure it meets requirement 1.	Y

Moisture Sensor

Requirement	Verification	Verification status (Y or N)
1. Measurements of similarly prepared samples must be within 10% of each other.	1. a. Place the sensor probes in water and then just in the air to get the 'wet' and 'dry' measurements. We will now get values in between these two numbers when we place it in soil. b. Test the sensor measurements using multiple soil samples, each made with specific amounts of water, and ensuring we get consistent results. All readings should be within 10% even if the probe is reinserted.	Y
2. Must be able to send values to the website within 10 seconds after the data is measured, and sample the soil once every 5 minutes.	2. a. Connect the sensor to the ESP32 with our circuit and write code to pull data from the sensor. Start with basic measurements and then test qualitatively with different soil samples. b. Test the time delay by checking for the time difference between changing	Y

	values that we know. Ex - Switch the probe from soil pot to a jar of water and see how long it takes for the value to reflect this change.	

Temperature Sensor

Requirement	Verification	Verification status (Y or N)
1. Must be able to get accurate temperature values near the plant with 5%.	1. a. The tests for this sensor will be done using a room with a controlled temperature and changes in the temperature will be checked for consistency. This will be done by using a separate temperature gauge	Y
2. Must be able to send values to the website within 30 seconds after the change, and check the values every 5 minutes.	2. a. Verify by running an experiment where we move the probe into a freezer and check the delay in measurement. The device is rated for -55° Celsius to 120° Celsius.[9]	Y

pH Sensor

Requirement	Verification	Verification status (Y or N)
1. Must be able to measure pH within 0.5 of the correct value.	1. a. We will compare the readings taken by the pH sensor to readings taken by a universal indicator on samples of varying pH. This will give us an idea of the accuracy of the probe.	N

Light Sensor - 3 points

Requirement	Verification	Verification status (Y or N)
1. Measure the ambient light around the photoresistor and send data within 10 seconds of	1. a. Run an experiment where we incrementally vary the amount of	Y

the change.	ambient light around the photodiode and check that the multimeter reading during those variances is affected by the correct increments. b. Check that the data shown on the website reflects a change consistent with the change in the multimeter reading and that it is updated within the required time frame.	
2. Must have consistent readings within 10% of each other when exposed to the same light environments for 10 minute intervals.	2. a. We will be leaving the photoresistor under a constant lamp source in an otherwise dark room and record readings for 10 minutes. Then we will plot all the readings and ensure they are within the desired range.	Y

Pump

Requirement	**Verification**	**Verification status (Y or N)**
1. Respond to on/off signals from the Microcontroller and be able to turn on for 10 second intervals.	1. a. We will set up a block of code to simply send a signal to the MOSFET for 10 seconds to turn on the pump, and check whether the pumping capacity outputs the correct amount of water. b. We will also do the same test with a block of code that sends this signal based on another GPIO pin input on the MCU, to check if it works with the logic of the entire system.	Y

Website

Requirement	**Verification**	**Verification status (Y or N)**
1. Properly display the correct values on the web page every 5 minutes.	1. a. Using the microcontroller, print out to console the values that it reads from the sensors and compare to those displayed on the website.	Y

	b. Take the time between each value change to make sure that it is done in 300 seconds +/- 10.	
2. The local host must successfully send an HTTP request to the web server when attempting to send the sensor values.	2. a. On the console, check if the HTTP response status code is one that indicates that the web server has successfully responded to the HTTP request[11].	Y