

## **Auto Garden Bed - Group 9**

Team members:

Nicholas Chitty - Electrical Engineer

Brendan College - Computer Engineer

Scott Peirce - Optical Engineer

Justin Pham-Trinh - Electrical Engineer

# Contents

<b>1</b>	<b>Executive Summary</b>	<b>1</b>
<b>2</b>	<b>Project Description</b>	<b>2</b>
2.1	Project Background . . . . .	2
2.1.1	Motivation . . . . .	3
2.2	Goals and Objectives . . . . .	3
2.3	Specifications and Requirements . . . . .	3
2.3.1	Engineering Specifications . . . . .	5
2.3.2	Marketing Requirements . . . . .	7
<b>3</b>	<b>Research</b>	<b>10</b>
3.1	Previous and Related Works . . . . .	10
3.1.1	DIY near-IR spectrometer . . . . .	10
3.1.2	Chlorophyll Fluorescence Spectrometer . . . . .	12
3.1.3	Smart Garden Controller . . . . .	13
3.1.4	Automated Rotating Solar Plant Rack with Self-care Capabilities	14
3.1.5	Stem 'n' Leaf . . . . .	14
3.1.6	Green Steel Garden . . . . .	15
3.1.7	Summary . . . . .	16
3.2	Related Technologies . . . . .	17
3.2.1	Ocean insight: Ocean ST NIR Microspectrometer . . . . .	17
3.2.2	AgroCares Nutrient Soil Scanner . . . . .	18
3.2.3	Web Technologies . . . . .	20
3.2.4	Proportional-integral-derivative Control . . . . .	24
3.3	Part Selection . . . . .	24
3.3.1	Controller Subsystem . . . . .	24
3.3.2	Power Subsystem . . . . .	28
3.3.3	Sensing Subsystem . . . . .	43
<b>4</b>	<b>Design Constraints</b>	<b>47</b>
4.1	Related Standards . . . . .	48
4.1.1	JSON Web Token (RFC 7519) . . . . .	49
4.1.2	HTTP/1.1 (RFC 2616) . . . . .	50
4.1.3	WebSocket Protocol (RFC 6455) . . . . .	50
4.2	Constraints . . . . .	51
4.2.1	Economic . . . . .	51
4.2.2	Time . . . . .	51
4.2.3	Equipment . . . . .	51
4.2.4	Safety . . . . .	52
4.2.5	Environmental . . . . .	53
4.2.6	Manufacturability . . . . .	53
4.2.7	Ethical . . . . .	53

4.2.8	Sustainability . . . . .	53
<b>5</b>	<b>System Hardware and Software Design</b>	<b>54</b>
5.1	Controller Subsystem . . . . .	54
5.2	Power Subsystem . . . . .	69
5.2.1	Solar Panel Control . . . . .	70
5.3	Sensing Subsystem . . . . .	72
5.4	Web Subsystem . . . . .	76
5.4.1	Server Backend . . . . .	77
5.4.2	User Interface . . . . .	80
5.5	Subsystem Integration . . . . .	80
5.5.1	Sensing . . . . .	81
5.5.2	Power . . . . .	81
5.5.3	Web . . . . .	81
<b>6</b>	<b>Testing</b>	<b>82</b>
6.1	Controller Subsystem Testing . . . . .	82
6.1.1	Electrical Characteristics . . . . .	83
6.1.2	Hardware Behavior . . . . .	84
6.1.3	Software Behavior . . . . .	87
6.2	Power Subsystem Testing . . . . .	88
6.2.1	Voltage Regulator . . . . .	88
6.3	Sensing Subsystem Testing . . . . .	89
6.4	Web Testing . . . . .	89
6.4.1	API Testing . . . . .	89
6.4.2	User Interface Testing . . . . .	91
6.4.3	Socket Testing . . . . .	91
6.5	Integration Testing . . . . .	92
6.5.1	Web Integration . . . . .	92
6.5.2	Sensor Integration . . . . .	93
6.5.3	Power Integration . . . . .	93
6.5.4	Full Integration . . . . .	93
<b>7</b>	<b>Administrative Content</b>	<b>93</b>
7.1	Milestones . . . . .	93
7.1.1	Fall . . . . .	93
7.1.2	Spring . . . . .	94
7.2	Progress . . . . .	94
7.2.1	Senior Design I . . . . .	94
7.3	Budget . . . . .	95

## List of Figures

1	Overall block diagram . . . . .	4
2	House of Quality . . . . .	8
3	NIR Photodiode Spectral Sensitivity . . . . .	11
4	NIR Photodiode Spectral Sensitivity . . . . .	12
5	NIR Photodiode Spectral Sensitivity . . . . .	13
6	NIR Photodiode Spectral Sensitivity . . . . .	14
7	NIR Photodiode Spectral Sensitivity . . . . .	15
8	NIR Photodiode Spectral Sensitivity . . . . .	16
9	NIR Photodiode Spectral Sensitivity . . . . .	18
10	NIR Photodiode Spectral Sensitivity . . . . .	19
11	Docker architecture . . . . .	20
12	Difference between SQL and NoSQL . . . . .	23
13	General Linear Voltage Regulator Circuit Schematic . . . . .	35
14	LM7805 Pin Diagram . . . . .	36
15	Series Voltage Regulator . . . . .	37
16	Shunt Voltage Regulator . . . . .	38
17	General Switching Voltage Regulator . . . . .	39
18	LM2596 Pin Diagram . . . . .	40
19	Buck Converter (Step Down) Circuit Schematic . . . . .	41
20	Boost Converter (Step Up) Circuit Schematic . . . . .	41
21	Buck/Boost Converter Circuit Schematic . . . . .	42
22	802.11b/g/n channels . . . . .	48
23	TCP frame ( <a href="#">The Transmission Control Protocol</a> , Fig. 1) . . . . .	49
24	Example JWT from <a href="#">jwt.io</a> . . . . .	50
25	CC3220 networking subsystem ( <a href="#">CC3200 SimpleLink™ Wi-Fi® and Internet of Things Technical Reference Manual</a> ) . . . . .	55
26	TCP socket control flow ( <a href="#">CC3100/CC3200 SimpleLink Wi-Fi Internet-on-a-Chip User's Guide (Rev. C)</a> , Fig. 5-1) . . . . .	57
27	Bitwise representation of command . . . . .	60
28	UML diagram of the classes used . . . . .	61
29	Bitwise representation of data sent to AWS . . . . .	63
30	CC3200 ADC module architecture ( <a href="#">SimpleLink™ Wi-Fi® CC32xx ADC</a> , Fig. 1-2) . . . . .	64
31	UML diagram of the subsystem's development methodology . . . . .	66
32	CC3200 pinout diagram (red labels indicate pins will be connected to other components in a way that differs from the reference design) . . . . .	67
33	MCU subsystem schematic . . . . .	69
34	Power subsystem block diagram . . . . .	70
35	Stepper motor configuration . . . . .	71
36	Sensing subsystem block diagram . . . . .	72
37	Diffraction angle calculation . . . . .	74

38	Photodiode Signal Filter . . . . .	75
39	Photodiode Signal Filter . . . . .	75
40	Web component block diagram . . . . .	76
41	Entity relationship diagram . . . . .	78
42	Color palette . . . . .	80
43	Web-Plant Bed Socket Integration Sequence Diagram . . . . .	82
44	Swagger UI example . . . . .	90
45	GitHub workflow . . . . .	90
46	Cumulative Flow Diagram from Jira . . . . .	94

## List of Tables

1	Engineering Specifications . . . . .	6
2	MCU option breakdown . . . . .	26
3	Differences between CC3200 and CC3220 . . . . .	28
4	Controller subsystem bill of materials . . . . .	28
5	Battery Selection . . . . .	30
6	Solar panel types . . . . .	31
7	Solar panel part breakdown . . . . .	32
9	Charge Controller . . . . .	34
10	Voltage Regulators . . . . .	43
11	Silicon Photodiodes . . . . .	44
13	InGaAs Photodiodes . . . . .	45
15	Linear Stage Actuator . . . . .	46
17	Diffraction Gratings . . . . .	47
19	MCU power mode behavior . . . . .	60
20	MCU LED operation . . . . .	68
21	URI table . . . . .	79
22	Breakdown of budget by subsystem . . . . .	95

# 1 Executive Summary

New gardeners typically struggle getting their garden started due to a lack of tending to their plants. This project seeks to solve many of the problems that new gardeners have through sensing and control. The main issues with plant growth relate to soil composition, soil moisture, temperature, and sun light. This project seeks to use optics to measure the soil moisture and composition; then an MCU will capture this data and control solar shades to control sunlight and solenoids to control watering. A web component will be included to check the weather as well as notify the user of impending weather events that could affect their plants adversely (frost or heat wave). The entire system will be powered with solar panels that are capable of tracking the sun through the sky and can act as blinds over the plants.

This project all starts with scoping out the project. The team has immediately compiling a list of must-have requirements and some things the team would like to accomplish as “nice to have’s”. The team started this process by looking at all the similar projects that have already been done and looked at all the ways the team can expand on the work they have already accomplished. For example, the team liked the weather aspects of a project for getting rain information; a problem the team were thinking about was how to get the system in as much of a “set it and forget it” state as possible as it pertained to frost. The solution is to integrate with a weather service online and send notifications when there is a frost or freeze advisory.

After assembling the list of requirements, the team set out to create a high-level functional block diagram for each of the subsystems. This helps the team see where the different systems integrate for the future as well as breaking out all the different components that may need to be purchased.

The ultimate novelty in this project is all of the sensing that will be done through spectroscopy. The team has found a plethora of research on the topic and has started familiarizing themselves with the limitations and capabilities of the available technology. Ideally, the team would like to find a scalable solution to the sensing in which the optical sensing could be attached to a drone or satellite to survey fields for farming.

## 2 Project Description

At the core, this project is a microcontroller project that anybody with a slight understanding of digital communications could accomplish. Our team in the background of our project will cover the novelties of this project that separates us from any “Gardduino” project that can be found on numerous blogs. In our Goals and Objectives, we will cover how we will accomplish the novelties as well as the specifics as to why our project is better and more marketable.

### 2.1 Project Background

Home gardening is a valuable hobby that helps people get outdoors, create something beautiful, and earn tangible rewards. Unfortunately, plants are sensitive organisms that require consistent attention. Plant life also involves complex relationships between the organisms and their environment. These two problems can discourage potential gardeners. This team proposes to build a system that addresses these issues through mechanical automation and user notification.

In 2022 there is great potential for realizing these advantages. Remote sensing, wireless communication, API integration, and closed-system power and water control are both available and economical. Many “DIY smart-gardens” and “Gardduino” projects have been published to the internet. In the agriculture industry, there are commercially available technologies with high performance systems for water distribution, network communication, and remote sensing. This project is intended to advance the field by producing a system that can maintain a suitable environment for plant growth by autonomously sensing and modifying the conditions of the garden bed. In addition, it will feature a notification system that allows the plants to “speak” by prompting the user to make accommodations when bad weather is projected, such as a cold snap.

Especially worth noting is that this project will feature on-the-rise technology in the form of a Near Infrared Spectrophotometer. Smart agricultural systems need to determine several variables, including moisture level, nutrient content, and acidity. There is a wide variety of techniques for sensing soil moisture, but by far the cheapest and most used is electrical conductivity. This involves pressing electrical nodes into the ground and up against the wet soil matrix, which inevitably leads to corrosion. DIY and commercial systems require other discrete sensors and even chemical analysis to characterize the state of the soil. Near Infrared Spectroscopy is an alternative method of soil sensing that offers many advantages over these traditional technologies. It works by stimulating a response from weak molecular bonds in the soil, isolating the frequencies of that response, and comparing the signal strengths to that of a known sample. In addition to being a noninvasive, corrosion-immune source of information about soil moisture, the NIR Spectrometer gathers data about the chemical contents of the soil. This means that the same scan will detect the presence and quantity of soil nutrients and water acidity as well. A high speed, high precision spectrometer

costs between \$5,000 - \$10,000, but this application requires only rudimentary sensing capabilities. It may even be possible to acquire the parts for less than \$500, and if this is the case, then it would be a significant step towards increasing the adoption of state-of-the-art sensing technologies.

The proposed system integrates sensing with a power control scheme. The garden bed will sit under a gantry, which will serve as the foundation for a solar panel array. If deemed efficient, the solar panel will have a swivel mount to maximize energy captured. This power will be stored in a battery and used to control a solenoid that will release water into the garden bed when prompted by the sensing system. It will also power the microcontroller, NIR Spectrometer, Solar panel swivel mount, and wireless communication.

### **2.1.1 Motivation**

The idea came from seeing the “Garduino” style projects all over hobbyist forums and websites but the idea really took hold in that each member of the team saw an opportunity to explore a new facet of engineering they held an interest in. This project provided the team an opportunity to apply our knowledge on power systems and delivery, controls, digital signal processing, and optical sensing. These are all areas that the team wanted to demonstrate a high level of understanding in and grow at the synthesis level.

## **2.2 Goals and Objectives**

The primary goal of this project is to have a plant bed. This plant bed should be able to actively measure soil moisture content, soil OH group content, and soil acidity. Based on these measurements the plant bed will communicate with a web interface to serve this data to users as well as allow users to control the watering of their plant bed remotely. The plant bed will be powered via solar panels to charge batteries in order to provide a truly “set it and forget it” gardening experience.

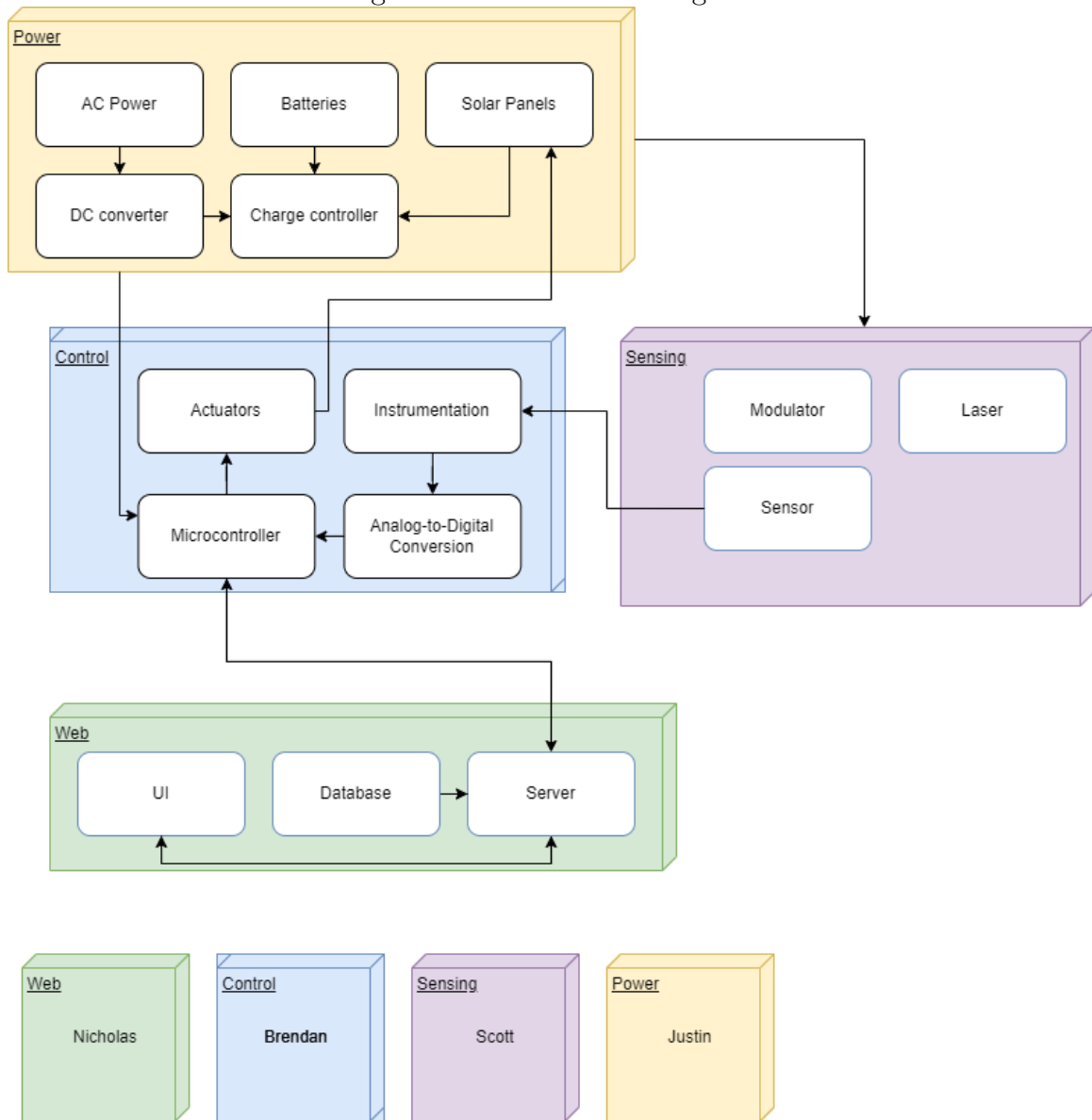
**Function of the Project** Per the motivation of this project, the plant bed should serve to ease new gardeners into the hobby by removing the monotony of watering and allowing users to focus on researching the proper parameters for their plants.

## **2.3 Specifications and Requirements**

This project idea requires the integration of power, systems, computer, optical, and web engineering to achieve a final product of a self-sustaining plant bed. The design in Figure 1 shows four different subsystems the team has designated as power, control, sensing, and web.



Figure 1: Overall block diagram



**Control** The control subsystem is the brains of the entire operation. This subsystem has to accomplish four distinct tasks:

1. Actuate mechanical components (linear rail, solenoid valve)
2. Convert analog sensor data to digital data
3. Send plant bed telemetry to web subsystem
4. Receive commands from web subsystem and modify system accordingly

In order to accomplish the first task, the chosen microcontroller (MCU) must be capable of driving the currents for these components and also support pulse-width modulation (PWM) for interacting with the motor controller. The second task requires that the MCU have an analog-to-digital converter. The third and fourth tasks necessitate WiFi connectivity as well as firmware support for either HTTP requests or WebSockets. These requirements for the control subsystem weigh heavily in the discussion of which MCU to choose found in Section 3.3.1.

**Power** The power system’s function is self-explanatory, supply power to the entire system. This will be accomplished in two ways. First, using a DC barrel-plug to the wall, the system could be powered this way. The second way, is via the solar panels, batteries, and charge controller. Both manners of supplying power to the system must be regulated. Discussed in Section 3.3.2 are the different ways that the charge controller can efficiently switch between battery power and the panels to increase battery health and charge level.

**Sensing** The sensing subsystem is where the majority of the novelty of this project lies. A lot of problems with electronic sensing is decay in the wet environs of plant beds. To improve upon this problem and hopefully offer a more sustainable and expandable approach, the team is going to try to accomplish sensing optically via infrared spectroscopy. Through the use of spectroscopy the team will be able to collect a wider range of data than just soil moisture content which includes soil OH group content, temperature, and acidity.

**Web** The web component of this project accomplishes three things; data analysis, reporting, and user specified control of components. The data analysis is occurring on the web because of the greater availability of compute power and greater ease of programmability. For reporting, the web will use databases to store data ad infinitum and serve this data in the form of graphs or “live” metrics. The web component also will be able to take a user’s input to send commands back to the control system to actuate different parts such as the solenoids or to ask for a more recent data reading.

### 2.3.1 Engineering Specifications

The team compiled a set of specifications for each of the subsystems that would need to be implemented to accomplish the requirements from above. Table 1 shows breaks down these specifications by the corresponding subsystem.

Table 1: Engineering Specifications

Subsystem	Metric	Specification
<b>Control</b>	Input voltage	2.8-5.5 V
	Power consumption	$\leq 1.50\text{W}$ nominal power draw
	Shutdown power draw	$\leq 100\text{ uW}$
	Processor speed	$\geq 20\text{ MHz}$
	ADC resolution	$\geq 8\text{-bit}$
	ADC sampling rate	$\geq 1\text{ ksps}$
	Transmit power	$\geq 10\text{ dBm}$
	Receive sensitivity	$\geq -50\text{ dBm}$
<b>Power</b>	Power	20-30W
	Battery Capacity	8Ah/95Wh
	Battery Nominal Voltage	9.6-12.8V
	Charging Time	<4 hours
	Regulator Efficiency	>65%
<b>Sensing</b>	Spectra	400nm - 1700nm
	Resolution	<10nm
	Output voltage range	0-1.8V
<b>Web</b>	Up-time	95 $\pm$ .1%
	Storage Capacity	>16 GB
<b>Miscellaneous</b>	Dimensions	1 m <sup>3</sup>
	Weight <sup>1</sup>	<50lb

**Control** The specifications in the control portion of the table are focused on future integration. This is best exemplified by the need for a large ADC (above 8-bits) as well as the wireless capabilities. The power and voltage specifications are for convenience when integrating with the power subsystem.

**Power** The power specifications were made throughout the design process. For example, the power output was an unknown until all the subsystems had a firm idea of their power requirements. The maximum instantaneous power draw of the entire system is in the 20-30W range, hence where that specification is drawn from. The battery capacity and charging time are derived from the duration the team decided the system should last on battery power and the input power from the solar panels.

**Sensing** The spectra was chosen

**Web** The web engineering specifications were hard to put into words. In general, the team wanted a reliable service quantified by the up-time of the web service. The

<sup>1</sup>The weight of the system includes a full soil load

service also needs to be able to store data in perpetuity for at least a single plantbed. Due to the efficiency of data storage and compression as well as the frequency of scans, the team felt that 16GB was effective enough to showcase the full system for now.

**Miscellaneous** The current miscellaneous specifications regard the physical qualities of the system. The dimensions of a meter cubed are necessary for the depth needed for various plants as well as solar panels. The weight is heavily correlated to weight of batteries, control and sensing components, as well as the soil and water hosing system.

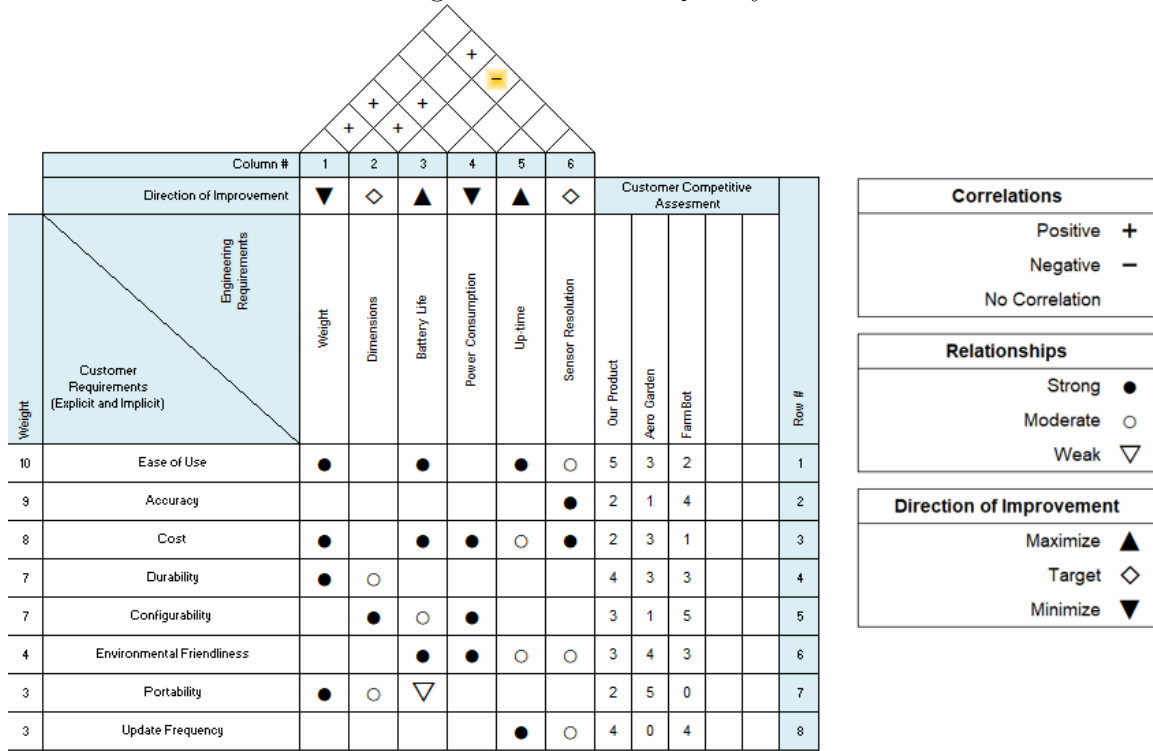
### 2.3.2 Marketing Requirements

When thinking about the target market for a project like this, the team put together a list of requirements important to the consumers. The most important feature for the consumer would be how easy the product is to use and navigate. This feature ranks the highest and would be the main determining factor on whether a consumer purchases this product over a competitor. Two other highly important features to a consumer would be the accuracy of the product and the cost. The accuracy is important because it directly correlates to the success of the customer's garden. If the product is not accurate in watering the plants, the customer will be unsatisfied and the plants will not survive. Cost because every consumer has to consider their budget - how much they are willing to spend for the features they deem most important. Other customer requirements the team brought up include the durability, configurability, environmental friendliness, portability and update frequency; listed from more important to less for consumer preferences. Each of these requirements have a direct correlation to at least one of the engineering requirements set for the product.

The engineering requirements set by the team are: weight of the product, dimensions of the product, battery life, how much power it consumes, up-time of the website, and quality of sensor resolution.

Many of the engineering requirements are directly correlated with each other. As weight improves, or lessens, so will product dimensions and vice versa. As dimensions decrease for the product, the amount of power needed decreases which means power consumption lessens which is also a positive correlation for the requirements. The only negative correlation in the engineering requirements the team found was that as sensor resolution increases, the life of the battery decreases as the lighting for the sensor must remain on longer.

Figure 2: House of Quality



When considering the customer requirement of ease of use, the team considered the impact of all the engineering requirements. Weight, battery life, and up-time all have a positive impact on customer experience. The lighter the product is, the easier it is for the customer to use because they can move it and adjust things within the product easier. The longer the battery life, the less time the consumer needs to worry when there are days with lower sunlight because the batteries can withstand a longer duration without being recharged. When the product has more up-time, the less time the consumer has to spend waiting on product updates. One of the engineering requirements that has a moderate impact on the customer experience is the sensor resolution. Although it can make a large impact on the overall product performance, its direct correlation to ease of use is not as strong as other requirements. The second most important marketing requirement listed by the team was accuracy. The only engineering requirement that had any correlation, had a strong correlation which is sensor resolution. This requirement is important to this marketing requirement because it could be a tipping point on whether a consumer purchases this product versus another. If the sensor resolution is not high enough to detect watering needs for the plant, the product will not be successful. This is why it is such an important requirement to focus on for both marketing and development of the product.

Something every team and every consumer is going to consider when designing or purchasing a product is the cost. The team rated this marketing requirement as

an 8/10 for this product. It was not the highest priority because for a product like this, consumers will pay more to have a more accurate and easy-to-use product. The team also considered that customers using this product will do more research about the technical features because they are most likely growing fruits or vegetables that they will consume. Unfortunately, most things that increase the quality and appeal of the product, have a negative relationship that is strong in correlation. When the team looks at lighter materials, those materials increase the price. When adding a longer battery life and lowering power consumption, price will increase proportionally to how much those requirements change. To increase the resolution of the sensor, the team and consumers will have to pay more for the increase in quality and performance. The only requirement that correlates to cost that was not strong, but still has an impact is the up-time. The more up-time of the product services, the more cost that is involved, but it is a smaller margin compared to the other requirements that directly impact the cost of the product.

Durability is most affected by the engineering requirements of weight and dimension. If the product materials are different, the durability will change and if the dimensions change shape or length, the construction will change affecting product strength. Weight will have a stronger impact on durability because it is something consumers will consider more heavily and the materials will impact more. This is why weight has a strong correlation and dimensions are moderately correlated. Another marketing requirement that is similarly affected by weight and dimension is the portability of the product. The battery life also affects the portability. As battery life increases, so does the size and configuration of the battery which affects how easy it is to move. Weight has the largest affect on portability, followed by dimensions and then battery life. The configurability of the project is strongly affected by both dimension and power consumption. Battery life has a smaller affect on this, but still changes the set-up of the product.

Although environmental friendliness rates lower on the consideration for a consumer, it is affected by more engineering requirements. Battery life has a small impact on how environmentally friendly this product is because of the waste that will exist once the battery is retired. The factor considered most impactful is the power consumption. The greater the power consumption, the less friendly the product will be to the environment. If the team is able to focus on lowering the power consumption, marketing will be easiest for environmental friendliness. Up-time and sensor resolution have a moderate affect on this requirement and should have some consideration in design of the product, but will be considered with more correlation to other requirements than environmental impact.

The last requirement comparison done by the team is update frequency. This marketing requirement is strongly correlated to up-time and moderately correlated to sensor resolution. Both of these engineering requirements affect the time spent on

updating the software for the product.

**Competition** This product already has a competitive edge in the current market because it is unique. It will be the first all-inclusive garden bed that can be used outside. When looking at products that offer something similar, the team found two main options that a consumer may consider next to this product. The first competitor being considered is the Aero Garden. This product is a small indoor herb garden that provides UV light to help the plants thrive. A consumer wanting to start only an herb garden indoors may consider this product because it is more affordable, but would choose our product for its ease-of-use. The full garden bed design is more durable and waters the plants for the consumer. This is a big factor for a consumer to consider because many gardeners over or under water their plants. So although the Aero Garden may be more competitive indoors, the target market for our product will be won over with the requirements our team has focused on. The second competitor we considered is the FarmBot. This product is an automatic watering system that can be installed on an outside garden bed. When consumers glance at both of the products, they will automatically lean towards ours because of the value and price point. The FarmBot may have more accuracy, but not significant enough to justify the price difference. Additionally, this product is harder to assemble and larger which won't accomodate the average home-gardener. FarmBot is for a consumer that has more time, space, and money, but not for the majority of consumers.

## 3 Research

Our research covers three core components: previous works, related technology, and part selection.

### 3.1 Previous and Related Works

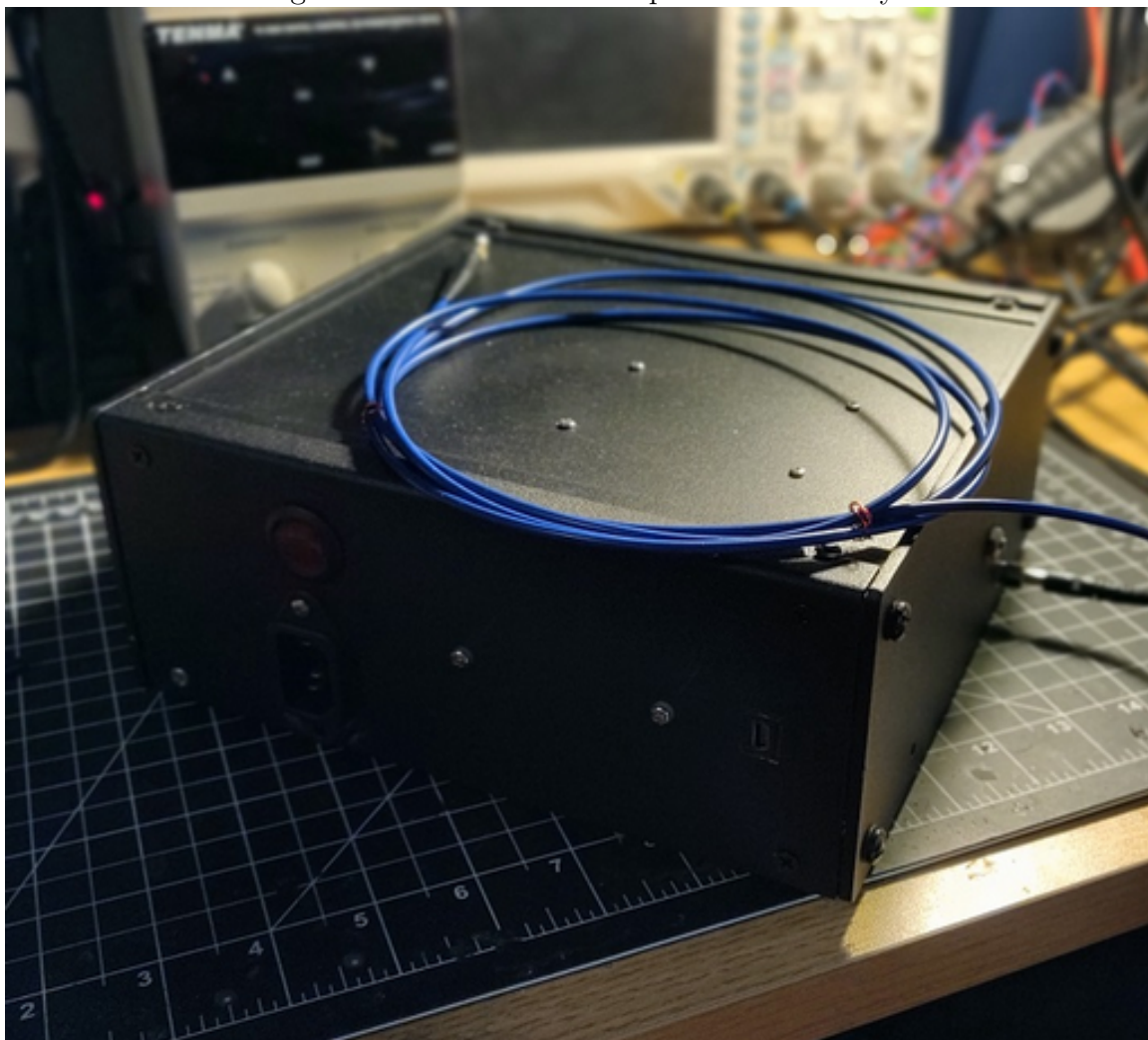
These are projects that accomplished similar objectives to what we hope to accomplish or highlight a technology we plan to work with.

#### 3.1.1 DIY near-IR spectrometer

Yuan Cao is a Ph.D. student at MIT with a blog where he posts his unofficial projects. He published a project titled "A \$500 DIY near-IR spectrometer that would sell for \$10,000." In it, he describes his idea, designs and results for a low cost infrared spectrometer. His design features surprisingly high performance for its price. The system uses an InGaAs photodiode, a reflective diffraction grating, a fiber collimator, some cheap optics, and a microcontroller to create a spectrograph of any light source. It boasts very high signal to noise ratios, 5-6nm spectral resolution, and a price point under \$500. There are definitely some design features worth imitating, specifically the

diffraction grating “scanner” that is rotated to pass wavelengths across the surface of the photodiode, as well as the filter circuitry that cleans up the electrical signal.

Figure 3: NIR Photodiode Spectral Sensitivity



That said, there are some reasons why this project differs in application from the Auto Garden Bed Near Infrared Spectrometer. First, the detector has a spectral range from 800 – 1600nm. This is the Near Infrared Regime, but it is not very far into the NIR, which in some contexts refers to wavelengths as far out as 2500nm. If Soil Spectroscopy requires this depth, or frequencies in the visible spectrum, the design will have to be modified. Second, this system was built for lab use, specifically to characterize light sources. The reported data was very promising, but in every case the spectrometer targeted an object that was emitting strong optical power in every direction. Soil does not fluoresce, so the Auto Garden Bed will require additional components to probe the soil with an electromagnetic wave. In the end, what’s most important is that this project demonstrates that low cost spectroscopy can be



achieved.

### 3.1.2 Chlorophyll Fluorescence Spectrometer

In 2021 UCF's ECE Senior Design Group 1 designed a Chlorophyll Fluorescence Spectrometer. The purpose of this project was to design a system that would detect chlorophyll in plants through stimulation by UV rays. It featured a diffraction grating and a monochromatic CMOS sensor. It is interesting to note that unlike the previously explored project, spatial separation of light frequencies is achieved with a monochromatic camera. This means that each intensity can be mapped to the position on the CMOS sensor, and the system can "stare" rather than "scan." Eliminating moving parts is a major benefit in projects requiring sensitive optical alignment, making this a feature worth seriously considering.

Figure 4: NIR Photodiode Spectral Sensitivity



This project produces a design with similar goals to the Auto Garden Bed, because it features a light source, a target object, focusing optics, wavelength selection, and generating and interpreting a spectrograph. However, the optical regime of UV rays may be limited to fluorescence spectroscopy and unfit for proximity soil sensing. The research will have to determine what sensors are viable and whether this imposes further constraints on the system.

### 3.1.3 Smart Garden Controller

The Smart Garden Controller was a UCF senior design project with very similar motivations to the Auto Garden Bed. The goal was to create a system that reduced user labor by automating the irrigation schedule of garden areas and reducing waste by sensing moisture levels. The system was also designed to anticipate the informational needs of the gardener and come prepared with a set of popular vegetables and plants, corresponding to variables the team had already investigated to maximize flourishing. It used sensors to detect the health of the plant environment, a microcontroller to facilitate irrigation, and a web API to interface to the user for control of the system. The group also set goals to offer ease of setup, competitive price, and Smart Device integration by integrating with a device like Amazon’s “Alexa.” The last similar design features are wall plug power consumption, lithium ion battery backup power, and http secure information protocol over wifi.

Figure 5: NIR Photodiode Spectral Sensitivity

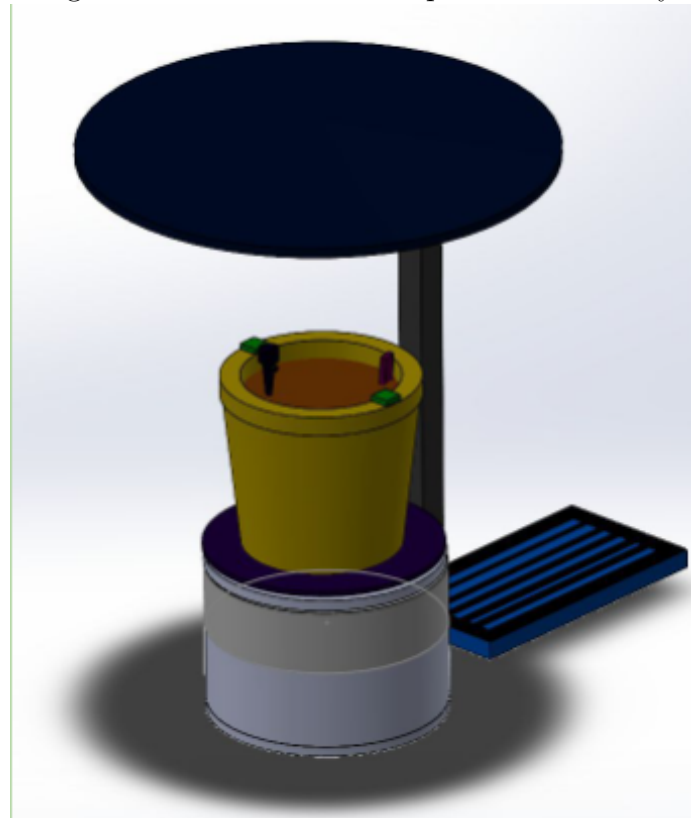


This project features much of the same core functionality as the Auto Garden bed, however, closer review of the scope and feature set reveal that the designs are incompatible. The Smart Garden Controller was designed to facilitate more precise control of an entire garden, allowing the user to isolate different areas for growing different plants. The area of interest was 100 squared feet, and the system was intended to relieve the gardener of the complication of watering each section for different amounts of time. This meant a wide reaching irrigation network as well as a network of sensors throughout the yard. The scope of the Auto Garden Bed allows for single source irrigation and sensing, which allows for greater flexibility of mechanical design, including power generation and environmental controls.

### 3.1.4 Automated Rotating Solar Plant Rack with Self-care Capabilities

This is another home garden system with a different target solution. The goal is to facilitate the growth of indoor plants and ensure maximum plant health by rotating the plant base so that the forces driving it to grow sideways cancel out in every direction. Like previous projects, the system is integrated with wireless communication and a schedule selector for different plant types. Unlike previous projects, one of the means of achieving plant health is by protecting sensitive plants from excessive exposure to sunlight, which is a feature that the Auto Garden Bed seeks to implement.

Figure 6: NIR Photodiode Spectral Sensitivity



Another point of interest in the Automated Rotating Solar Plant Rack is its rotating foundation. Maximizing sun exposure is a goal of this project, and although the application is energy collection, there may be parallel design opportunities with the Solar Plant Rack.

### 3.1.5 Stem 'n' Leaf

Stem 'n' Leaf<sup>2</sup> is a UCF Senior Design project. The focus of this project was to be a “modular hydroponics system” wherein each plant unit can be fed my a singular

<sup>2</sup><https://www.ece.ucf.edu/seniordesign/sp2021su2021/g07/>

control unit. Thus, the “stem” is the control unit and the “leaves” are the plant units. Their design was focused on modularity and scalability. Hence their design is stackable and tileable design of the bed itself. The key features of this plant bed is that it self-regulates pH and integrates with a mobile application.

Figure 7: NIR Photodiode Spectral Sensitivity



The project uses a liquid pH-sensor to obtain the relevant data. Our team wants a similar result but will be trying to achieve this via optical means. Their team mentioned the durability of the sensor which is our team’s chief concern seeing as acidic and basic solutions tend to be corrosive and promote oxidation of metals thus the optical approach should improve longevity of the project.

### 3.1.6 Green Steel Garden

Green Steel Garden<sup>3</sup> is another UCF Senior Project that our team is pulling some inspiration from. Another hydroponics system that measures soil nutrients and pH for controlling the parameters of the garden bed. The key feature of this project is the nutrients system and the pH sensor.

---

<sup>3</sup><https://www.ece.ucf.edu/seniordesign/su2021fa2021/g11>

Figure 8: NIR Photodiode Spectral Sensitivity



This project utilizes a water reservoir and reservoirs of chemicals to balance the pH of the water entering the garden bed. They use peristaltic pumps to accomplish the mixture and a combination of pH and electrical conductivity sensors to get data from the water. The current design consensus for our team is that we will be hooked up to hose which is limited by a solenoid valve, but being able to “treat” the water that is entering the system may be a design decision that has to be considered.

### 3.1.7 Summary

The previous works discussed here highlight the key components of our garden bed: sensing in conjunction with optics. Something of note in these deliberations is the use of hydroponics in all of these projects; something that is rendered unnecessary and seemingly inefficient in an outdoor garden bed.

The previous works related to optics focus on wavelengths outside of our team’s consideration (Infrared), instead focusing on UV and visible light through near-Infrared. Their research highlighted potential flaws in some assumptions we had made. Firstly, that it is possible to reduce the moving components of our optical sensing, obtaining a wider field. Secondly, that using IR spectroscopy is practical for getting data from soil as it is used in commercial projects today and shows tremendous value in commercial sectors.

Most of the other garden bed projects focus on hydroponics but something of interesting note is the rotating solar rack project. The control scheme regarding the degrees of rotational freedom and the means of achieving success will be paramount

in implementing our own moving solar array. Of interesting note is that all the other garden bed projects focused primarily on hydroponics as it gives greater measures of control to the soil nutrients which had not been a consideration of our team before looking into the previous works. Instituting a chemical pump into our design may be necessary to achieve our goal of being “set it and forget it” while still providing updates for when to refill the chemical tanks of integration with the water.

All of the garden projects tried to integrate some means of a mobile app or web user interface. They all mentioned that they needed to have started this component earlier. Each of the mobile application projects were not able to achieve this by their final meeting which is of special concern. Also of note is their choice of stack; many of the projects chose a NoSQL stack which seems counterintuitive given the highly relational nature of the data that is being collected. Our team can learn from this by starting this integration as early as possible and choosing to implement the smallest possible feature set that accomplishes the goals.

## **3.2 Related Technologies**

TBD, DESCRIPTION NEEDED HERE.

### **3.2.1 Ocean insight: Ocean ST NIR Microspectrometer**

Ocean Insight is a local manufacturer of high-end, low size, weight, and power spectrometers. The ST NIR Microspectrometer is about 40 cubic centimeters in volume with a scan speed of 10ms, a signal to noise ratio of 190:1, and a spectral resolution of 2.2nm. Its spectral range is from 645nm to 1085nm, and it was specifically designed to be integrated into larger systems for customers who were interested in a flexible, low-cost design. Added to that, the system is rugged, and offers a variable slit input size, increasing its flexibility even further. While the designs are proprietary, this system serves as a benchmark for what can be achieved by the industry, and no doubt there are major design changes that can be made to achieve a similar result for the application intended for the Auto Garden Bed. That being said, the selling price for one is \$1,750.

Figure 9: NIR Photodiode Spectral Sensitivity



### 3.2.2 AgroCares Nutrient Soil Scanner

AgroCares offers a Near Infrared Spectrometer specifically designed for Proximity Soil Sensing. Its spectral range is from 1300 to 2500nm and it uses Micro Electrical Mechanical Systems or MEMS to capture EM Waves reflecting off the soil. The real value of the product is in its wireless communications system. The device uses Bluetooth 4.0 to send data to a cloud data center. There, spectrographs of large data sets of soil with known nutrient contents are compared with the reading, cutting out the need for on-sight calibration. The system is handheld and uses eight tungsten halogen bulbs to blast the soil with energy. This light is collected in an extremely small area, sampling 65 squared millimeters. It would be worth researching to see if the Tungsten bulbs were linked to the 1300 to 2500nm spectral range or if another probe and sensor would suffice.



Figure 10: NIR Photodiode Spectral Sensitivity



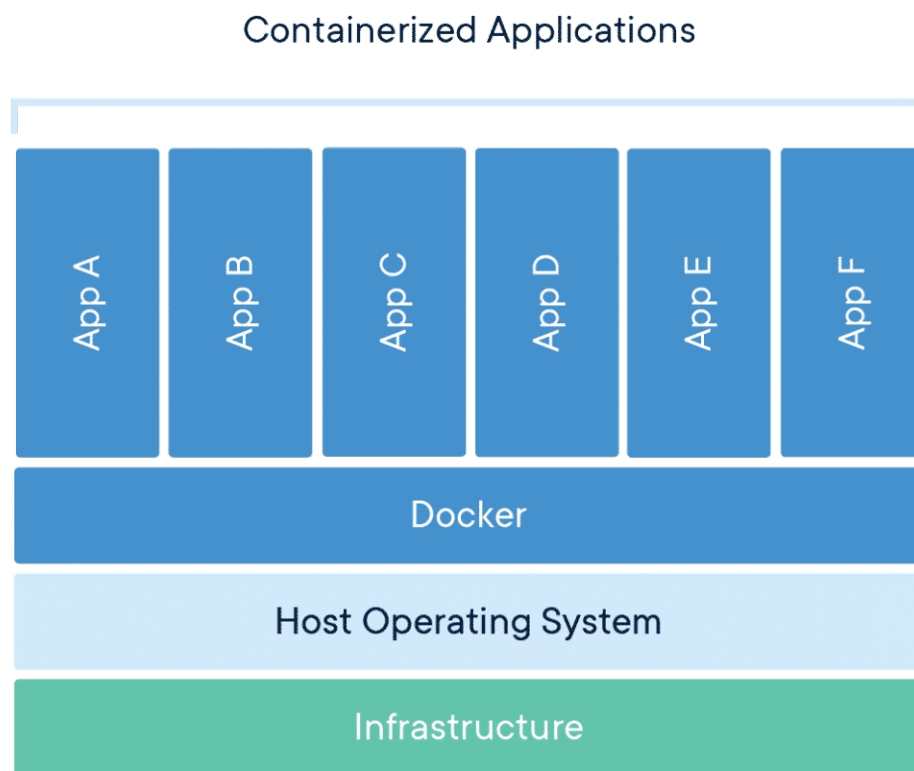
AgroCares offers a Near Infrared Spectrometer specifically designed for Proximity Soil Sensing. Its spectral range is from 1300 to 2500nm and it uses Micro Electrical Mechanical Systems or MEMS to capture EM Waves reflecting off the soil. The real value of the product is in its wireless communications system. The device uses Bluetooth 4.0 to send data to a cloud data center. There, spectrographs of large data sets of soil with known nutrient contents are compared with the reading, cutting out the need for on-sight calibration. The system is handheld and uses eight tungsten halogen bulbs to blast the soil with energy. This light is collected in an extremely small area, sampling 65 squared millimeters. It would be worth researching to see if the Tungsten bulbs were linked to the 1300 to 2500nm spectral range or if another probe and sensor would suffice.



### 3.2.3 Web Technologies

**Docker** “Docker is a platform designed to help developers build, share, and run modern applications. We handle the tedious setup, so you can focus on the code.” This quote comes directly from Docker’s website on why developers should switch to Docker. This technology makes items more portable by making the executable platform agnostic through the use of a docker kernel and containers. The figure below should provide some clarity:

Figure 11: Docker architecture



A developer can program applications such as those demonstrated in figure 11, package them into images and run them in containers. Docker works with the OS kernel to provide the same environment to the application each and everytime.

One of the largest advantages to using and implementing Docker that the team sees is that one team member can program and package the image and it should

“just run” on any other team member’s machine. This fact will prove very useful in integration testing especially for the socket server detailed in Section 5.4.

**User Interface** Because these are web technologies we will be using Javascript frontend frameworks/libraries for creating the frontend. Based upon cursory research, the most feature-rich and most widely used are React and VueJS.

**React** React is a component-based library for building user interfaces. Many libraries have extended the functionality by adding components to React for use by other developers. A React component is a stateful element in the Document Object Model (DOM). Essentially, based on the state information, a component is rendered in raw HTML to the browser. One such example of a component library is MaterialUI (MUI). MUI is a popular library for adding components like hamburger menus, tables, graphs, etc. Choosing React decreases the time spent developing due to the ease of tossing boilerplate components at the problem. The biggest issue is that all rendering is done in the browser which means that loading an uncached page may take a long time. Frameworks built ontop of React such as NextJS help speed up this process by moving some of the processing to the server.

**VueJS** VueJS is a framework for building user interfaces. The difference between a library and a framework is that a framework provides a control flow while libraries are just used. VueJS is not too unlike raw HTML and JavaScript wherein `<script>` tags are used to create dynamic pages in response to user actions. The key difference between Vue and the above is that Vue provides the access to component-based programming. Similar to React, Vue works on the basic of components but it uses the default HTML DOM and adds functionality to pre-existing components. This comes with the limitation that components are not nearly as interleaved with data from a web server. Vue is performant and designed for single-page applications, which may not server this project well in the long-run.

**Web and Socket Server** This section will host information related to technologies for building out a web and socket server for connecting the user interface to the backend. .NET, Java and JavaScript all have reasonable solutions to these but based on the JavaScript user interface, this discussion will be kept to Java and JavaScript solutions and technologies.

**Java Spring Boot** Java Spring Boot is a full featured library that has different webservers embedded such as Apache Tomcat. Web servers are the means for which an application can be accessed from the outside world. Java Spring Boot makes this process incredibly easy. Part of this project will be communicating via TCP packets as well as through HTTP requests. Java at first glance seems like the easier way

of accomplishing both tasks through the `java.net` libraries and through the use of Spring Boot `@Controller`.

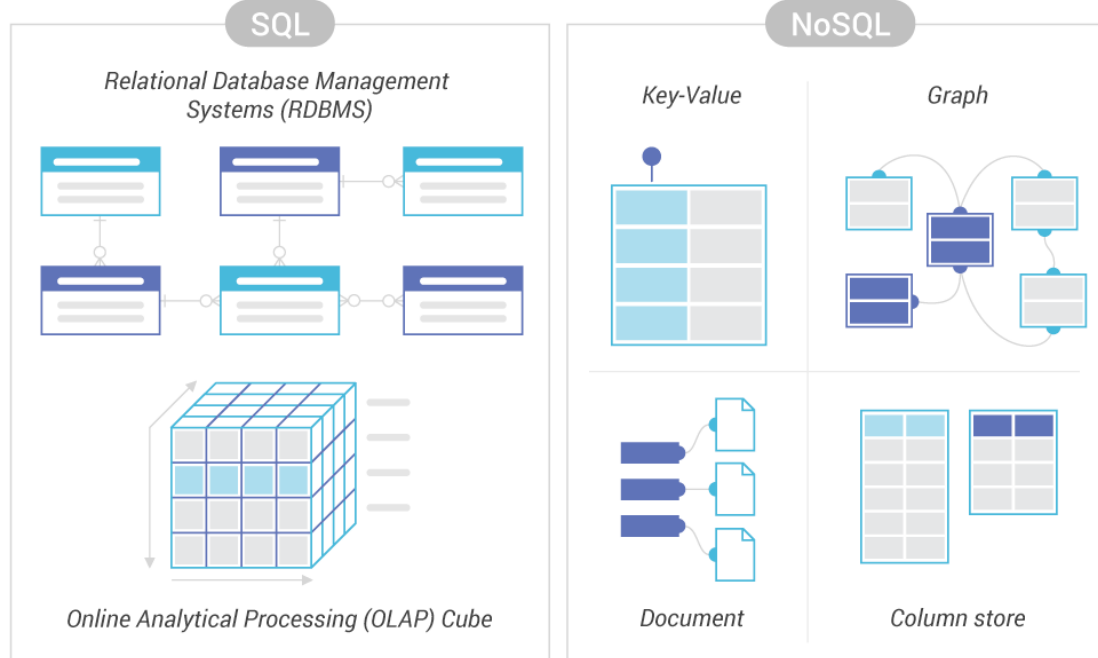
The `java.net` libraries are essentially a port of the network protocols that were made in the C language for creating communications via packets. By deploying using Java Spring Boot, it is possible to create two services packaged into one, the socket server and the web server. Java's memory management and VM environment leave a lot to be desired. The current state of the language and its artifact leave a lot to be desired as well. Because of the nature of running as a server and potentially servicing multiple plant beds, Java's lack of callbacks and lacking multi-threading support means it is downgraded given its overhead.

The `@Controller` is such a great feature in Spring Boot. Java is strictly typed, leading to a lot really helpful features in executing backend requests. Also, because of the overhead, Java ORMs tend to be more fully featured and have support for a variety of other tools that increase velocity when programming. One such tool is Liquibase. Liquibase is a database changelog tool for creating and implementing database migrations based on a code.

**ExpressJS** Express is a lightweight backend framework for implementing routes and middleware in JavaScript. The biggest disadvantage of using Express is the lack of low-level capabilities. Express was designed to be multiple layers of abstraction away from the kernel which may prove to be tumultuous when trying to create a socket server that communicates with the MCU.

**Databases** A database is essential for tracking data and persisting it for use later. There are two main types of database, SQL and NoSQL. With these two types of database comes a variety of implementations and on top of that a variety of tools to work with them. Let's compare SQL vs NoSQL first.

Figure 12: Difference between SQL and NoSQL



**SQL vs NoSQL** In Figure 12 you can see the abstracted way to think about these two different systems. SQL records are exactly that, a record at a point in time. If one entity encapsulates another then another table holds that information. In NoSQL, the information is abstracted to a document with key-value pairs to get specific data. These documents can make references to other documents but for time-ordering this is highly ineffective. SQL records are highly effective for logs and indexing a large amount of data based on a higher level entity such as user that has many garden beds; garden beds that have a lot of data, etc.

**MySQL vs PostgreSQL** The two SQL databases widely used in industry are MySQL and PostgreSQL. MySQL is touted as “a simple relational database [... that is] very efficient and user-friendly” while PostgreSQL is widely used in data analytic and scientific applications because of the extensibility, scalability and object models. Immediately this seems like the better option but PostgreSQL may be harder to stand up immediately. Special consideration should be given to AWS solutions as well.

**MongoDB vs DynamoDB** MongoDB is “a general-purpose, document-based” database. DynamoDB is AWS’ proprietary solution to NoSQL databases. DynamoDB is more difficult to work with as it is the newer service, it could also potentially be more expensive over time, however, DynamoDB has improvements over MongoDB for indexing and building out reference documents.

### 3.2.4 Proportional-integral-derivative Control

Proportional-integral-derivative control (PID control) is a common control algorithm (especially in industrial control systems) to allow a control loop to have reliable performance in a variety of conditions. Simply put, this algorithm allows a controller to receive an input and calculate a proportional output, accounting for error and rapid changes in the process. This algorithm may be useful to our application because it will allow the product to better control its facilities without user input.

The control function is defined in Equation 1, where  $u(t)$  is the control variable (e.g. the garden bed's water control solenoid),  $K_p$ ,  $K_i$ , and  $K_d$  are gain factors, and  $e(t)$  (the error) is the different between a desired setpoint and measured process variable (e.g. the difference between the desired and current ounces of water dispensed).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (1)$$

**Proportional Term** The proportional term of Equation 1 is  $K_p e(t)$ , hence referred to as the P-term. The P-term is proportional to the current error, and the gain  $K_p$  determines the magnitude of the P-term. If the gain is too large, then the process variable will oscillate.

**Integral Term** The integral term of Equation 1 is  $K_i \int_0^t e(\tau) d\tau$ , hence referred to as the I-term. This term accounts for previous values of  $e(t)$  by taking the integral of the error, and the gain  $K_i$  determines the magnitude of the I-term. The I-term aims to account for residual error in the control loop.

**Derivative Term** The integral term of Equation 1 is  $K_d \frac{de(t)}{dt}$ , hence referred to as the D-term. The D-term aims to control future values of  $e(t)$  by taking the derivative of the error, and the gain  $K_d$  determines the magnitude of the D-term. This term acts to damp rapid changes in the control loop. Higher values of gain may make the control loop more sensitive to noise and lead to instability.

## 3.3 Part Selection

In our research for part selection we wanted to create a generalized comparison of the options we have available. We formatted this information in the tables below.

### 3.3.1 Controller Subsystem

**Minimum Requirements** At minimum, any chosen microcontrollers (MCUs) shall support natively, or by addition of a module, these features and traits:

- Ability to connect to external antenna
- Analog-to-digital converter (ADC)

- IEEE 802.11
- In stock and available to order
- JTAG module or equivalent
- Module communication bus (UART, I2C, SPI)
- Network stack
- Onboard CPU sufficient for our purposes
- Onboard memory sufficient for our purposes
- Onboard nonvolatile memory
- Sockets support
- Pins dedicated to analog input
- Pins dedicated to digital I/O

**Nice To Have** These features would be "nice to have" on any MCU selected, but are not required:

- Digital-to-analog converter (DAC)
- Integrated antenna
- microSD card slot
- Onboard battery
- Pins dedicated to pulse-width modulation (PWM)
- Timer(s) and an RTC
- USB compatibility
- Additional wireless communication protocols (e.g. BT or BLE, Zigbee)

**Selections** The selections, listed in Table 2 and not in any particular order, match the above criteria and are being considered for selection.

**Single-board Computers** Use of single-board computers (SBCs) was considered, but will not need to be used; sockets will be used on an MCU in conjunction with [Amazon EC2](#) services will allow us to offload computing to a cloud solution.

Table 2: MCU option breakdown

Model	LAUNCH-XL-CC26X2-R1	LAUNCH-CC3220-MODASF	Pico W	Nano 33 BLE	B-L4S5I-IOT01A
<b>Manufacturer</b>	Texas Instruments	Texas Instruments	Raspberry Pi	Arduino	STMicroelectronics
<b>Micro-controller</b>	CC2652R	CC3220-MODASF	RP2040	nRF52840	STM32-L4S5VIT6
<b>Processor</b>	1x ARM Cortex-M4F	1x ARM Cortex-M4	2x ARM Cortex-M0+	1x ARM Cortex-M4	1x ARM Cortex-M4
<b>Maximum Speed (MHz)</b>	48	80	133	64	120
<b>Memory (KB)</b>	256 ROM, 352 flash, 100 SRAM	1024 flash, 256 RAM	16 ROM, 264 SRAM	1024 flash, 256 SRAM	2048 flash, 640 RAM
<b>Wireless capability</b>	BLE5.2, Zigbee, Thread	802.11b/g/n	802.11n	BLE5.3, Zigbee, Thread, Matter	BT4.1, 802.11b/g/n, NFC
<b>Serial capability</b>	UART, I2C, I2S, SPI	UART, I2C, SPI	UART, I2C, SPI, USB1.1	UART, I2C, I2S, SPI, USB2.0	UART, I2C, SPI, USB2.0
<b>Price (\$)</b>	40, maybe free	60, maybe free	6	28	53
<b>ADC</b>	8-channel, 12-bit	4-channel, 12-bit	4-channel, 12-bit	8-channel, 12-bit	16-channel, 12-bit
<b>Watchdog timer?</b>	No	Yes	Yes	Yes	Yes
<b>GPIO (pins)</b>	31	29	30	13	16
<b>PWM (channels)</b>	Supported	Supported	16	4	6
<b>Required voltage (V)</b>	1.8 – 3.8	2.3 – 3.6	1.8 – 3.3	4.5 – 21	4.75 – 5.25

**External WiFi Module** Use of an external WiFi module is discouraged due to the following reasons:

- Added cost
- Added complexity
- Modules in common use by hobbyists often have poor or no proper documentation, to the extent of:
  - Quick start guide
  - User’s guide
  - Datasheets
  - Theory of operation
  - Application uses
  - Troubleshooting guide
  - Schematics and mechanicals
  - Quality and reliability
  - Errata

Therefore, all of the MCUs listed above support either the 802.11 or Bluetooth standards.

**External ADC Module** Use of an external ADC was considered, but ultimately decided against. An external ADC module would only add further cost and complexity to a project where the on-chip ADC is sufficiently effective for the project’s requirements. The sample rate and resolution of an on-chip ADC is more than enough for our needs, and the low-cost goal of the project directly conflicts with the idea of purchasing an external ADC module.

**Selection** Ultimately, the [LAUNCHCC3220MODASF](#) was chosen as the microcontroller development board for this project. In the event that the aforementioned LaunchPad is not able to be obtained, the [CC3220SF-LAUNCHXL](#) has equivalent capability for the project’s needs.

These boards, henceforth referred to as the CC3220, are able to be requested from our university at no upfront cost to our team. This was the driving factor behind choosing the CC3220 over other microcontroller development boards. It was also determined that the microcontroller *must* be able to interface via the 802.11 (WiFi) standard, for reasons that are detailed in subsection 5.1—therefore, the LAUNCHXL-CC26X2R1 and Nano 33 BLE were disqualified from selection. The Pico W was considered due to its low cost, and the B-L4S5I-IOT01A considered because of its abundant peripherals, but both ultimately lost out to the Texas Instruments products.



Table 3: Differences between CC3200 and CC3220

Model	CC3200	CC3220
Microcontroller	CC3200	CC3220SF
Secure Flash (MB)	n/a	1
Simultaneous TCP/UDP Sockets	8	16
WiFi Receive Sensitivity (dBm, 1 DSSS)	-95.7	-96
WiFi Receive Sensitivity (dBm, 54 OFDM)	-74.0	-74.5
Hibernate Current Draw ( $\mu$ A)	4	4.5

Table 4: Controller subsystem bill of materials

Qty	Cost (\$/ea)	Manufacturer	Item	Model Number
1	0.00	Texas Instruments	SimpleLink Wi-Fi CC3200 LaunchPad	CC3220- LAUNCHXL
2	10.88	Texas Instruments	CC3200	CC3200R1M2RGCR
2	3.80	Taiyo Yuden	Surface Mount 2.4 GHz Antenna	AH316M245001-T
2	6.95	Adafruit	Plastic Water Solenoid Valve	997

**CC3200** Due to difficulties in acquiring the CC3220 (second-generation SimpleLink device), our team made the decision to use the CC3200 (first-generation SimpleLink). These parts are mostly identical, with the differences detailed in Table 3. Development will take place on the [CC3200-LAUNCHXL](#), with the PCB MCU part to be [CC3200](#) (if needed).

**Bill of Materials** The bill of materials for the controller subsystem can be found in Table 4.

### 3.3.2 Power Subsystem

**Power Supply** The power system is a key factor to this model and trying to make this an independent system. This is key because this power system needs to be able to power all of the many different components, while also charging itself when not operating.

As part of the goal to make this an independent system, solar energy plays a great role in this and making this system run. Before anything as well, this part of the system must operate first before it can power other components. For this system to run, the key parts include: solar panels that convert light energy into electrical energy; solar charge controller to regulate output voltage from the solar panel into the battery; the battery to directly power the other components in the model.

In this model there are many different sensors, electrical, and mechanical components all of which require power. This requires a design of how the power system will flow and operate. It first begins with determining the total power needed for the whole system to run. This can be determined by finding the individual power ratings of each component and calculating them all together. After that is found, we can then choose what type of battery and the quantity needed for the model. When choosing

between the solar panel and the battery to regulate how much power is going into the battery. Once that is done, a voltage regulator must be determined, to regulate voltage from the battery to the smaller components that need to be powered. After all that is done, then we can find other components that will make the power system more reliable and efficient in any way.

**Power Requirement** Sensors, electrical, and mechanical components all require power but all consume different amounts of power. As mentioned before, analyzing the total power requirement is important and crucial because this will help us determine the right parts that will be best for this system and for the components.

With the total power determined, the Watts per hour needed for all of the components to run must also be found. The watts per hour is important too because this helps figure out how long each component will operate for. After that is found, we used the altE calculator to help pick out what kind of battery can be used, then solar panel and solar charge controller, respectively. The altE calculator is a great resource because with the proper measurements, it can help us choose what kind of battery we can use, by determining the capacity needed in watt-hours or amp-hours. This calculator can also help us choose how big of a solar panel we need and how big of a solar charge controller we need as well.

**Rechargeable Battery Selection** Solely relying on solar energy isn't always ideal. This is because the weather may not always guarantee sunlight, this will hinder its power retention. For this reason, solar panels are paired with a battery so that the power can be stored and then used at a later time. As part of the goal to have this run as an independent system, an additional battery may be used so that one battery can power the system while the other one can charge.

There are many different types of batteries including Nickel-Cadmium, Nickel-Metal Hydride, Lithium ion, etc. Of the three batteries, they will be compared to see which will best fit our model and which will fulfill the requirements on the basis of power output, efficiency, etc.

Nickel Cadmium (NiCd) batteries in today's time are used for RC vehicles, power tools, photography equipment, and more. They would also be considered as old technology. Even though they are old, they still have their advantages such as being less expensive, they are super powerful and charge fast, they require little maintenance, and more. These batteries though also has its disadvantages, one being they suffer from "memory" problems and as a result of that, it may reduce that capacity of charges and future battery life. These batteries are also environmentally concerning because cadmium is toxic.

Nickel-Metal Hydride (NiMH) is similar to nickel cadmium, the only difference is that hydrogen is used instead of cadmium as the active element. Hybrid cars, toothbrushes, and phones are just a few of many products that use nickel-metal hydride batteries and have been used in these appliances because of the trouble free service they grant. It is also because even being partially discharged, it can be

charged as many times and will always be at full capacity. Even with advantages like that nickel-metal hydride batteries produce a lot of heat when in use, have a high self-discharge rate, and have memory issues as well, just not as bad as NiCd.

Lithium ion batteries, one of the most popular types of rechargeable batteries for portable products. It is considered the best because lithium ion batteries have high open circuit voltage, low self-discharge rates, and little to no memory effect. On top of that they are growing within the military, electric vehicle companies, and aerospace industry with little to no maintenance. Even though they have a lot of advantages, some of the disadvantages include sensitivity towards high temperatures, it cannot be fully discharged, and the cost.

Through much consideration and investigation, the four batteries listed in the following table were picked. All of which are 12V lithium iron phosphate (LiFePO4) batteries. Then the selected battery that we decided we were going to use for this model is the Eco Worthy 12V 8Ah LiFePO4 battery. We decided to go with this battery because although it is a little expensive, the Watts per hour and the Ah rating that this battery provides, was great for what we plan on having.

Table 5: Battery Selection

<b>Manu- facturer</b>	<b>Ampere Time</b>	<b>Eco Worthy</b>	<b>Expert- Power</b>	<b>Eco Worthy</b>
<b>Voltage</b>	12	12	12	12
<b>mAh</b>	6000	10000	5000	8000
<b>Watt per hour</b>	76.8	120	64	96
<b>Cost</b>	\$29.99	\$59.99	\$35.99	\$43.99

**Solar Panel Selection** Solar has been a growing source of energy in the past years, with new developments and breakthroughs with solar cell technology. As the whole purpose of solar energy is to collect sunlight and convert it into electrical energy, that is the minimum for this model to run as an independent system. Then as a stretch goal, we would apply the concept of solar tracking panels to create blinds with the solar panels so it can open and close according to the position of the sun.

On a basic level, solar panels are made of solar cells and these cells do the collecting and converting. These solar cells are made from crystalline silicon that is melted down into ingots and then cut into sheets. In the solar industry there are 3 main types of solar panels. These types of panels are monocrystalline, polycrystalline, and thin-film panels all of which have different compositions. Each type has different efficiencies, generating different amounts of power, etc.

Monocrystalline solar panels are solar panels that are made with monocrystalline solar cells. These solar cells are composed of a single silicon crystal which provides electrons more space to move because of the electricity flow that is generated. This

makes them more efficient, yet at the same time more costly. Polycrystalline solar panels are solar panels that are made with polycrystalline solar cells. Similar to monocrystalline solar cells, they are made of a silicon crystal, the only difference is that instead of a single crystal, they use several fragments of silicon to form an ingot and that is cut into sheets. As a result of melting several fragments into one it creates a mosaic look as well as giving it a blue hue, whereas the monocrystalline solar panel will have a uniform color and look. Aesthetically, they might look nicer but they are less efficient than monocrystalline solar panels. This also means that they aren't going to be as pricey compared to the monocrystalline panels because of the efficiency and manufacturing process. Thin-film solar panels differ from crystalline solar panels greatly, all because they are made with different materials. The three main types of thin-film solar panels are amorphous silicon (a-Si), cadmium telluride (CdTe), and copper indium gallium selenide (CIGS). Currently, thin-film solar panels are the least efficient, costly, and have the shortest lifespan. It is predicted that they will have a major growth in the solar industry because although they are the least efficient, they have a higher theoretical efficiency than both monocrystalline and polycrystalline.

The choice of solar panels will be based on multiple aspects of each type of solar panel. As we know, the efficiency rating and cost from most to least will go from monocrystalline, polycrystalline, and thin-film, respectively, but we must also look at its temperature coefficient, power rating, and more to be able to determine which solar panel we exactly need. Temperature coefficient for solar panels is the power lost as the temperature rises. This plays a very important role because we live in Florida, temperatures can get hot. So, when it comes to which solar panel will still be more efficient in higher temperatures, monocrystalline solar panels are still the best. As follows, it then goes from polycrystalline and then thin-film. This doesn't mean we shouldn't count them out though, because depending on where you live the temperatures may not be high so it won't impact it as much. It could also be more cost efficient as well because the other two types of solar panels are less expensive. Another factor to keep in mind is the power capacity of each solar panel type. For monocrystalline solar panels, because of their single crystal structure it allows for a higher power output, compared to polycrystalline, its power output capacity isn't as high. For thin-film panels, because they don't have uniform sizes, they won't have a standard for power capacity.

Table 6: Solar panel types

Solar Panel Type	Monocrystalline	Polycrystalline	Thin - Film
<b>Efficiency</b>	>20%	15 - 17%	6 - 15%
<b>Power Rating</b>	$\leq 300\text{W}$	240 - 300W	Indefinite
<b>Performance</b>	Most efficient	Efficient	Least efficient
<b>Temperature</b>	High Tolerance	Low Tolerance	High Tolerance
<b>Cost per Watt</b>	\$1 - \$1.50	\$.70 - \$1	\$.43 - \$.70

Through research and investigation, different solar panels were compared to see which would be best fit for our model. With the different types of solar panels, we didn't specifically choose what kind of solar panel type we wanted to go with because they all were great options and benefited in various ways. Of the four choices listed in the following table though, the monocrystalline solar panel was a popular type. These solar panel choices that were picked, all have a power rating of 10 Watts and a voltage rating of 12V, except for the Voltaic solar panel which was 18V.

The selected solar panel that we chose was the Eco Worthy 10W 12V monocrystalline. There were a lot of factors that went into this choice, one of them was because of the cost of the solar panel, it was a great price for what we were getting. It was also because of the size and weight, it makes it very easy to move around. Also, it had many great reviews and is available on different websites to order.

Table 7: Solar panel part breakdown

<b>Sku</b>	P108	L02M10-1	NPA10S-12H	SLP010-12U
<b>Manu-facturer</b>	<b>Voltaic Systems</b>	<b>Eco Worthy</b>	<b>Newpowa</b>	<b>SolarLand</b>
<b>Solar Panel Type</b>	Monocrystalline	Monocrystalline	Monocrystalline	Polycrystalline
<b>Dimensions</b>	10.9 x 8.8 x .16	13.3 x 8.1 x .7	14.37 x 7.68 x .91	14.06 x 11.89 x 1.18
<b>Peak Current</b>	570mA	580mA	630mA	580mA
<b>Open Circuit Voltage</b>	20.45V	20.6V	19.83V	21.6V
<b>Peak Voltage</b>	17.34V	17.3V	16.77V	17V
<b>Wattage</b>	9 watt	10 watt	10 watt	10 watt
<b>Power Tolerance</b>	$\pm 10\%$	$\pm 3\%$	$\pm 3\%$	$\pm 5\%$
<b>Cost</b>	\$49	\$25.99	\$25.99	\$35.53

**Solar Charge Controller Selection** Solar charge controllers play an important role in this system because this system is running on solar energy. A solar charge controller is a regulator that goes in between the solar panel and the battery, regulating the total output power coming out of the solar panel. This is important because this will prevent the battery from overcharging and possibly reducing its effectiveness. If the wrong charge controller was picked, it could result in a loss of power that is generated, and can harm any device. As stated in the power requirement section,

finding the total power needed for the whole system is important because we can use that to help determine the proper charge controller to get.

There are two main types of solar charge controllers: Maximum Power Point Tracking (MPPT) and Pulse Width Modulated (PWM). Choosing the right charge controller is based on current and voltage characteristics. This is because they regulate input voltage coming in from the solar panel and output voltage of what is coming out to the battery.

Maximum power point tracking (MPPT) is a technique that observes and regulates energy coming from the solar panel and into the battery. What makes this special is that it can match the solar panel voltage to the battery voltage, which allows it to maximize the charge efficiency. They operate as a DC to DC converter, taking in high DC input from the solar panel, changing it to high AC voltage, then back down to DC voltage.

Pulse width modulated (PWM) solar charge controllers are considered the original charge controller compared to the MPPT charge controller. They are less expensive and the technique behind this controller is also simpler. On a basic level, the PWM controller acts as an on-off regulator. When the battery voltage reaches a certain level, the PWM controller will slowly reduce the charging current, up until the battery reaches the maximum amount of energy. This makes it great for smaller installations because the solar panel and the controller can match better.

While looking at different solar charge controllers, we came across 4 different controllers, all of which are PWM charge controllers. We did not specifically choose the PWM controller but through our search for what charge controller we wanted to use all of the choices we found were PWM. This would be great for our model regardless because our system will not be large and it will be less expensive compared to MPPT charge controllers.

The solar charge controller we chose for our model is the Eco Worthy 10A PWM solar charge controller. The reason we chose this controller is because it comes as a kit along with the solar panel. As a result of it coming as a kit with the solar panel they will already be very compatible together and the price for both of them together is not expensive at all.

Table 9: Charge Controller

Manu- facturer	Eco Worthy	Renogy	Expert- Power	Expert- Power
Charge- Controller Type	PWM	PWM	PWM	PWM
Output Voltage	12/24V	12/24V	12/24V	12/24V
Rated Charge Current	10A	10A	10A	20A
Max PV Voltage	30V	50V	55V	55V
Self Con- sumption	$\leq 10\text{mA}$	$\leq 10\text{mA}$	$< 10\text{mA}$	$\leq 13\text{mA}$
Price (\$)	23.99	69.99	34.99	69.99

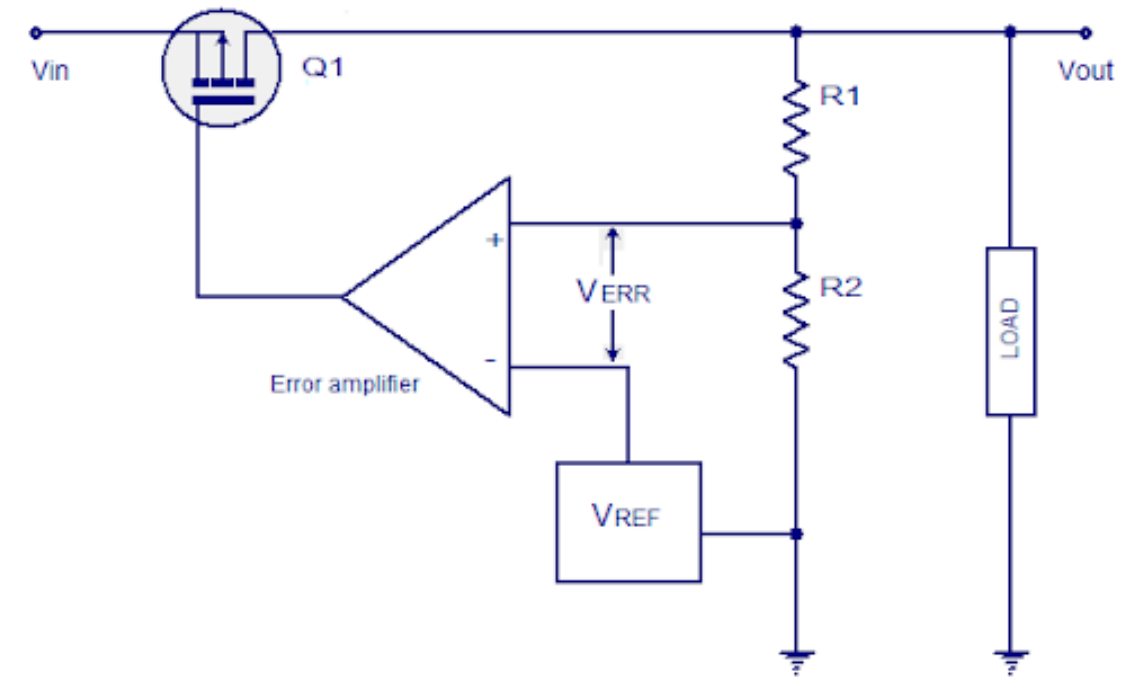
**Voltage Regulator** Voltage regulators play an important role in almost every electronic device. All electronic devices operate at different voltage ranges, some requiring a constant voltage. Common operating voltages are 3V, 5V, and 12V. To provide this voltage range or constant voltage, a voltage regulator is added into the circuit design. Voltage regulators help regulate voltages during power fluctuations and different variations in loads, preventing damage to any component. They can also regulate DC and AC voltages. For this power system, we will be focusing on DC voltage. This is because solar panels produce DC voltage.

As said before, voltage regulators play an important role in almost every electronic device. They are used for smaller devices to power components such as sensors, op-amps, and other modules, but can also be used in bigger applications such as TV's, automotive vehicles, industrial applications, and many more. There are two main voltage regulators: linear and switching. Both of these types have the same goal, regulating a system's voltage but operate differently depending on the application that it is used for.

**Linear Voltage Regulators** Linear voltage regulators, just as the name suggests, is a type of regulator where the linear and electrical components are placed in series with the input and output. The base of the linear voltage regulator is the use of an active pass device (such as a BJT or a MOSFET) which is controlled by a high gain amplifier. To maintain a constant output voltage, the regulator uses a closed feedback loop to bias the active pass device. Linear voltage regulators are also known as step-down converters because the output voltage is always less than the input voltage. As the power is consumed and dissipated in the transistor and

then is converted into heat while generating a constant output voltage. The figures below you will see the general schematic of the linear voltage regulator but also the pin-out diagram for a LM 7805 voltage regulator. As you can see, this type of voltage regulator has three pins for input, output and ground.

Figure 13: General Linear Voltage Regulator Circuit Schematic



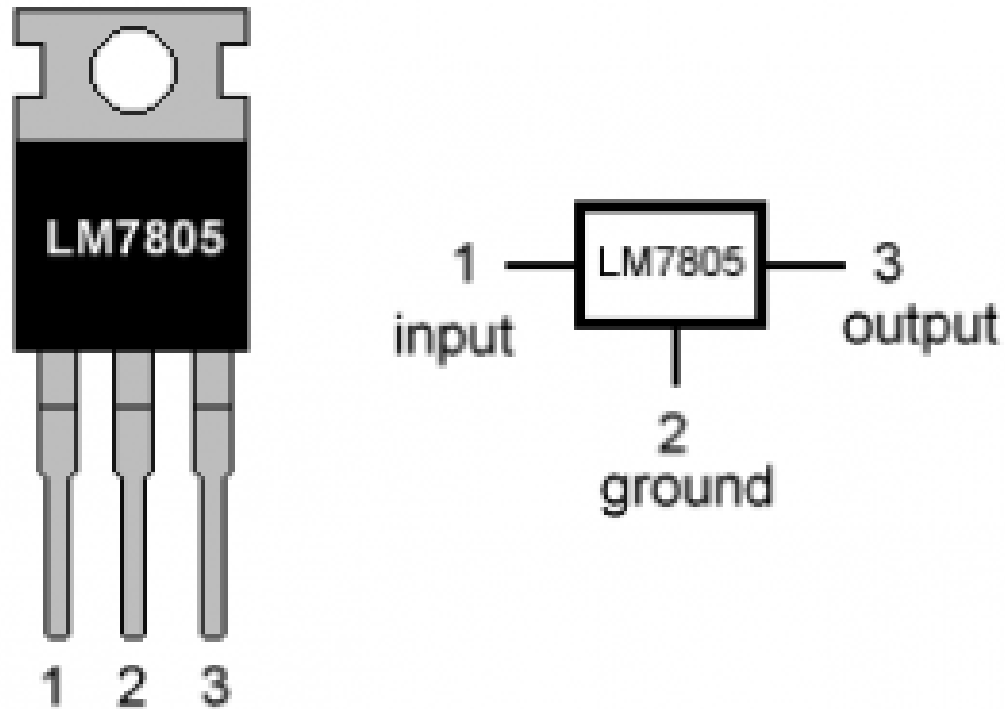
LDO voltage regulator schematic

[www.circuitstoday.com](http://www.circuitstoday.com)



Figure 14: LM7805 Pin Diagram

## LM7805 PINOUT DIAGRAM



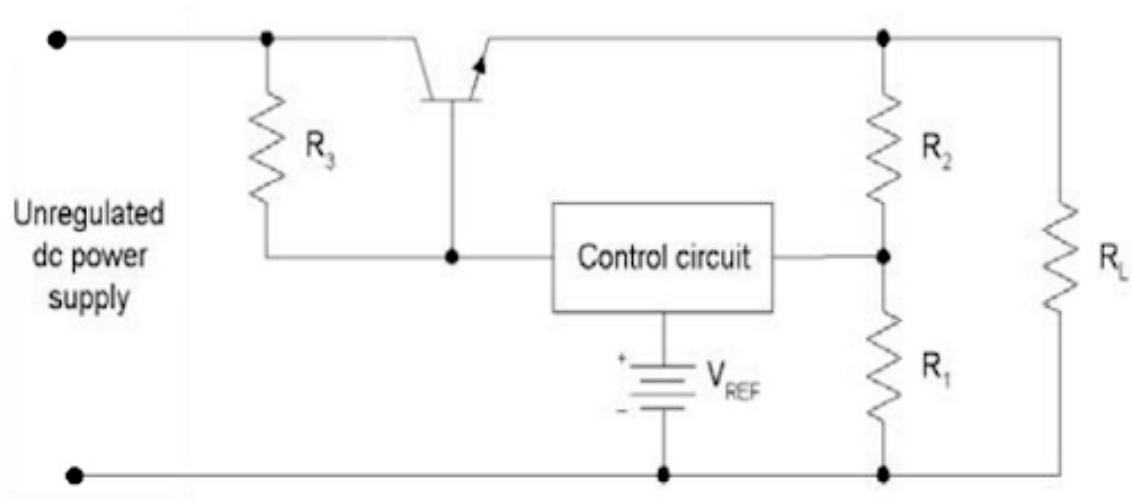
Just as any electrical device, linear voltage regulators will have their advantages and disadvantages. Linear voltage regulators are highly integrated devices. They are simple, cheap, responsive to changes in input voltage, load voltage, and there is no switching noise. Unlike linear voltage regulators, other voltage conversion circuits will have high-frequency switching noise, which may cause problems in the power system. Linear voltage regulators don't have to worry about that issue. The biggest disadvantage is that they are inefficient. This inefficiency is due to the voltage drop across the active pass device and causes the regulator to produce a lot of heat.

Just as voltage regulators can be broken up into linear and switching voltage regulators, linear voltage regulators can be broken down into different types as well. There are two main types of linear voltage regulators: series voltage regulator and shunt voltage regulator. The main difference between the two is that with a series regulator, the active pass device is connected in series where in a shunt regulator it is connected in parallel.

As the name suggests for a series voltage regulator, the pass element in the circuit will be connected in series with the load, as shown in the figure below. Looking at the

circuit, the output voltage is sensed through the voltage divider network in  $R_1$  and  $R_2$ , and is compared to the  $V_{REF}$ . As a result, the voltage drop across the transistor can be varied to make sure that the voltage across the load is constant.

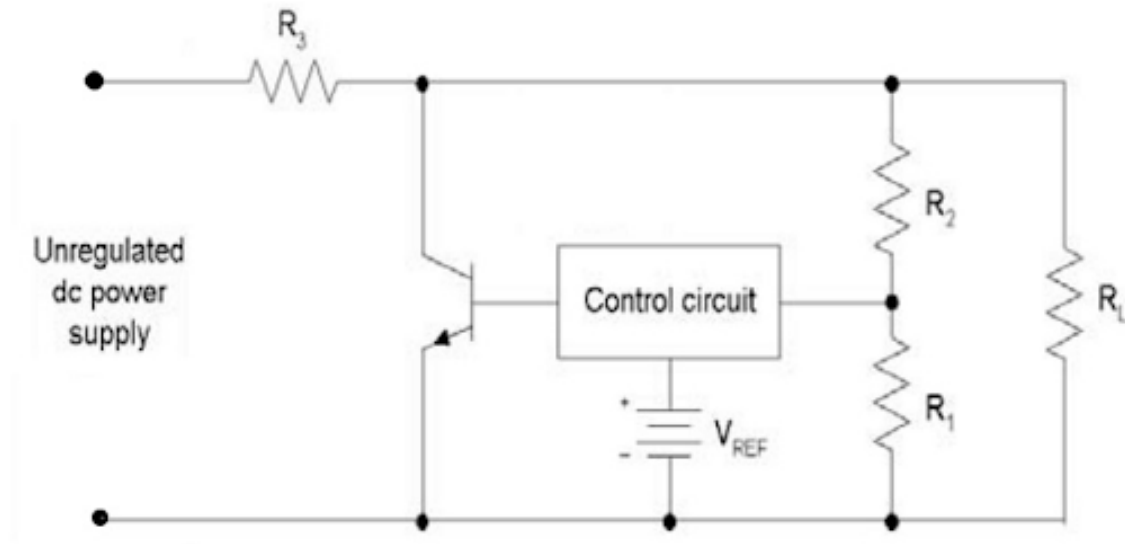
Figure 15: Series Voltage Regulator



The main advantages to this type of linear voltage regulator is that the current is effectively used by the load. This makes it more efficient than shunt voltage regulators. Regardless, the efficiency is still low compared to a switching voltage regulator but it is simple and the output will not have any switching spikes.

The figure below shows the circuit schematic of a shunt voltage regulator. The pass element in the circuit is connected in parallel while the resistors are connected the same as in the series voltage regulator. For this voltage regulator, voltage is maintained through the current drawn through the resistors and as the current is varied, the output voltage across the load remains constant.

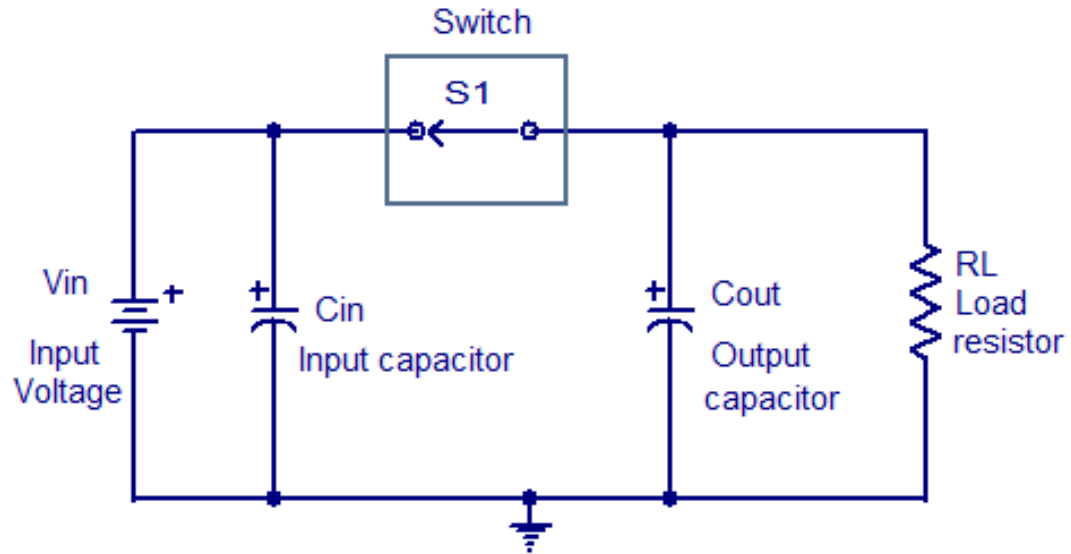
Figure 16: Shunt Voltage Regulator



When compared to the series voltage regulator, it is just slightly less efficient, but is simpler to implement into the circuit. This type of linear voltage regulator is less common and is used commonly in low-powered circuits and in voltage reference circuits.

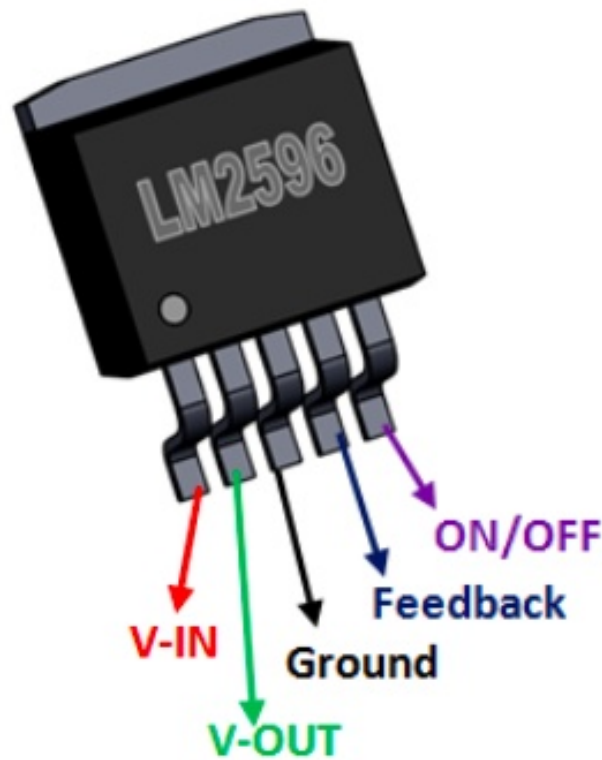
**Switching Voltage Regulator** Switching voltage regulators is a type of regulator that acts as a switch, where input power is turned on until the desired voltage is reached. After the desired voltage is reached, the switch element is turned off and stops any input power from coming in. With this type of voltage regulator, switching noise occurs because of the high-frequency from the reference voltage and the amplifier. As a result of the noise, capacitors, inductors, and other electrical components are used to smoothen out and reduce the noise. Regardless of the noise that occurs with switching voltage regulators, this type still remains very efficient because of the process of switching on and off at such high speeds. As said before, this type of regulator repeats its operation at high speeds, which allows it to supply voltage more efficiently and reduce heat generation. The figures below will show what the general schematic of a switching voltage regulator looks like but also the pin-out diagram for a lm2596 voltage regulator. You can see the difference in how many pins the lm2596 voltage regulator and the lm7805 voltage regulator. That is because switching voltage regulators require two more pins for the switch element as well as the feedback.

Figure 17: General Switching Voltage Regulator



Schematic of a simplified switching regulator

Figure 18: LM2596 Pin Diagram



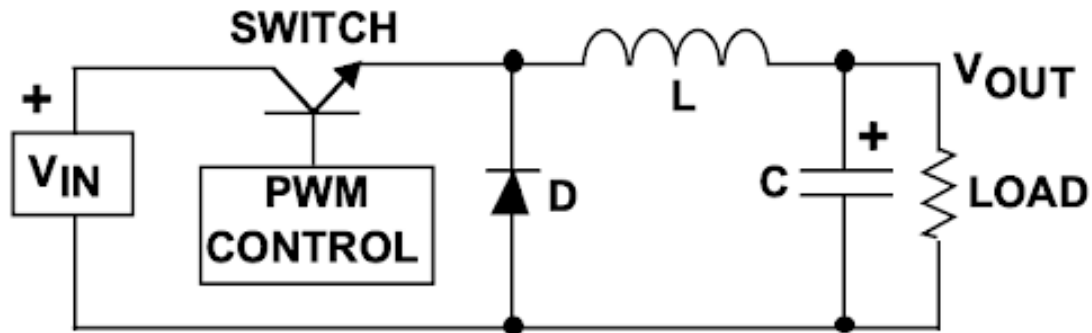
Some advantages to this type of voltage regulator is that it has a much higher efficiency, it does not produce a lot of heat, and it is capable of higher power efficiencies. The downside to this is that it has a more complex design, produces more noise which will require it to have more external components added to the circuit design. The choice between linear and switching voltage regulators always depends on what it is being used for because each design requires different things but switching voltage regulators are also a popular choice because of how efficient they are with power input and its general efficiency.

There are 3 main types of switching voltage regulators: buck converter, boost converter, and buck/boost converter. These three types of voltage regulators perform similar actions but all produce different output voltages.

Buck converters, also known as step-down converters, will lower the output voltage than the input voltage. This is similar to linear voltage regulators because linear voltage regulators work to make sure the output voltage is always lower than the input. The difference is that there will be less waste. The figure below shows the circuit schematic of a buck converter. Normally, a transistor is used as the switching

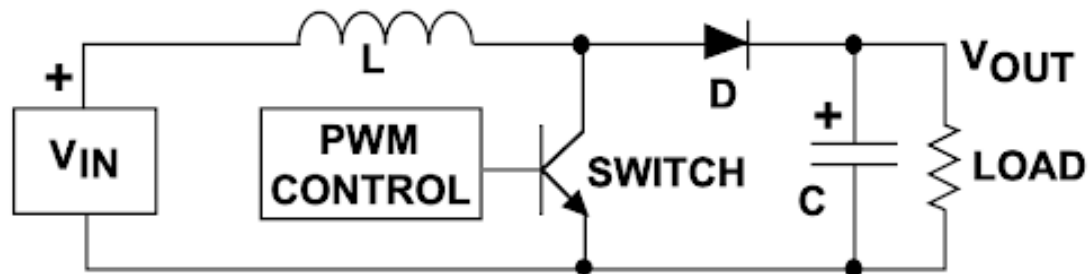
element, connecting and disconnecting the input voltage to the inductor.

Figure 19: Buck Converter (Step Down) Circuit Schematic



Boost converters, also known as step-up converters, will provide a higher output voltage than the input voltage. Even though the output voltage can be higher than the input voltage, the power being provided still has to be regulated within the output power specification of the circuit. Figure 8 shows the circuit schematic of the boost converter. As shown, the components are switched around compared to the buck converter. This allows the output voltage to increase, because when the switch is on, the voltage has to go across the inductor, increasing the current. When the switch is off, the diode is forward biased and charges the load higher than the input.

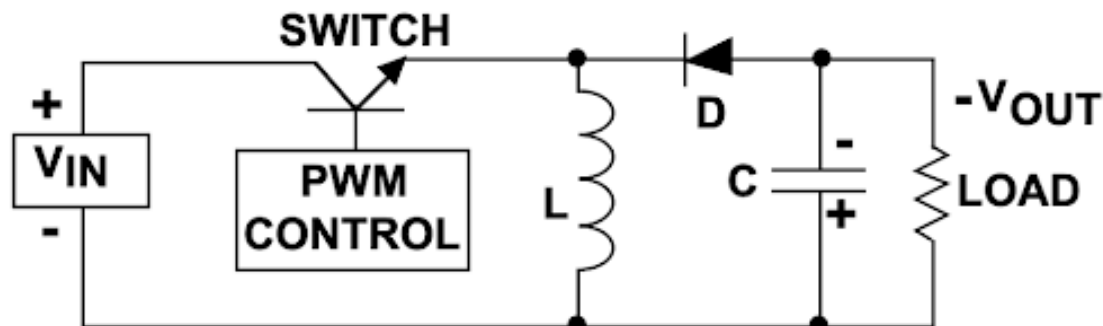
Figure 20: Boost Converter (Step Up) Circuit Schematic



The last type is the buck/boost converter and as the name suggests, this type of converter is able to supply either a larger or smaller output voltage, but can also invert the polarity. This type of voltage regulator is common with battery operated products because at first use the battery will supply full power but throughout time the battery as well as input power will depreciate. Figure 9 shows the circuit schematic of the buck/boost converter. This converter operates similarly to both the buck converter

and the boost converter but differs because it is capable of inverting the polarity. This is done by forward-biasing the reverse-biased diode when the switch is off.

Figure 21: Buck/Boost Converter Circuit Schematic



This type of converter offers many benefits. One of the reasons that it offers so much is because it brings both the buck converter and the boost converter together. It also offers lower operation cycling and is more efficient between the input and output voltages. A few downsides to this is that there is no isolation between the input and output and the output is always inverted, which results in complex sensing and feedback in the circuit.

**Voltage Regulator Selection** The main function of a voltage regulator is to regulate output voltage from the power source, which is the 12V battery powered by solar panels, to the microcontroller and other components, ensuring that each part receives the correct amount of voltage. The microcontroller we will be using, the Texas Instruments CC3200, operates at a 3.3V input voltage and as a result will require a voltage regulator that can lower the output voltage.

As mentioned before, not all components are going to have the same operating voltages. With the various components and voltage requirements, multiple types of voltage regulators will be tested to see which one will be the most efficient and best fit for our model.

For our system, we will be looking at the LD1117 series (Linear), LM317 (Linear), LM2576 (Switching), and LM2596 (Switching). Of the various types of regulators, these were picked because the required input voltage for the CC3200 is 3.3V. These different voltage regulators will be compared based on efficiency, output voltage, cost, and any other feature that it may offer.

From the datasheets of each voltage regulator we were able to make the comparisons in the table below.

Table 10: Voltage Regulators

Feature	LD1117-series	LM317	LM2576	LM2596
Voltage-Regulator Type	Linear	Linear	Switching	Switching
Operating Voltage	15V	3V - 40V	3V - 40V	4.5V - 40V
Output Voltage	3.3V	1.25V - 37V	3.3V, 5V, 12V, 15V	3.3V, 5V, 12V
Output Option	Fixed	Adjustable	Adjustable	Adjustable
Operating Temp	0°C - 125°C	0°C - 125°C	-40°C - 125°C	-40°C - 125°C
Efficiency	$\leq 62\%$	Varies	75%	73%
Frequency	N/A	N/A	53kHz	150kHz
Cost (\$)	\$0.90	\$0.99	\$3.69	\$6.73

Through comparison, it was agreed upon to select the LM317 (linear) and the LM2576 (switching) because of what was given to us by the datasheets. These were also selected so that we can observe and compare them even further to see which one will perform the best and be a good fit for our model.

### 3.3.3 Sensing Subsystem

The sensing subsystem will consist of a Visible / Near Infrared Spectrometer.

The spectrometer will consist of several optical, electrical and mechanical components:

- Silicon Photodiode
- Near Infrared Spectrum InGaAs Photodiode
- Linear Stage Actuator Carriage Rail
- Reflective Diffraction Grating
- Focusing Optic
- Fiber Patch Cable with attached Collimator
- LED Diode Array



**Silicon Photodiode** The silicon photodiode will be one of the two detectors in the system, responsible for covering the visible spectrum of the soil emissions. Silicon photodiodes are cheap and overdeveloped for this application, much of the marketing emphasizes high speed current rise times and low dark current. Our spectrometer will not require either. Instead, it will be useful to have a detector with large surface area. This will reduce the precision required of other optical elements in the system, while retaining the flexibility to reduce the active area of the detector with a small aperture if needed. As always, low cost will be favored.

Table 11: Silicon Photodiodes

Manu- facturer	Model	Range (nm)	Area (mm <sup>2</sup> )	Dark Current (nA)	Cost (\$)
Newark	BPX 61	400-1100	7.02	2	13.06
Digi-key	ODD-1B	400-1100	1	0.2	15.60
Digikey Opto Diode Corp	ODD-5W	300-1100	5	1	14.50
Edmund Optics	PIN-3CD	350-1100	3.2	0.15	34.00
Thorlabs	FD11A	320-1100	1.21	0.002	15.69
Thorlabs	FDS100	350-1100	13	1	16.08

The BPX 61 by Newark has a substantially better size to cost ratio than other relevant devices on the market, however it also has a higher ambient current than all others. One option is to select it and then compensate by investing in more LEDs to illuminate the target area and boost the signal up over the higher dark current, or by investing in more circuitry to clean up and boost the signal. Another option is to forgo the larger surface area and choose the sensor with the least ambient noise, Thorlabs' FD11A. There don't appear to be any standout choices aside from those two. All the other devices make compromises between their specs, while these two have standout specs, but require compromises in design. The BPX 61 offers the greatest flexibility, so it's our choice for the visible-spectrum photodiode.

**Near Infrared Spectrum InGaAs Photodiode** The NIR InGaAs photodiode is responsible for detecting the majority of the spectral area of interest in this soil spectrometer. Once again, the relevant specs are active area, cost, and dark current. Most NIR photodetectors are designed for highspeed free space optical communication, but this application very forgiving in terms of sensor speeds.

Table 13: InGaAs Photodiodes

Manufacturer	Model	Range (nm)	Area (mm <sup>2</sup> )	Dark Current (nA)	Cost (\$)
Digikey Excelitas	C30617BH	800-1700	0.1	1	43.58
Digikey Advanced Photonix	0800-3111-011	800-1700	1.36	0.2	50.21
Thorlabs	FGA015	800-1700	0.018	0.5	63.00
Thorlabs	FGA01	800-1700	0.01	0.05	67.55
Edmund Optics	N/A	800-1700	0.07	0.03	88.00
Edmund Optics	N/A	800-1700	0.12	0.05	88.00
Edmund Optics	N/A	800-1700	0.3	0.3	94.00
Edmund Optics	N/A	800-1700	0.4	0.4	94.00

The Advanced Photonix 0800-3111-011, distributed by Digikey has the highest active area, the second lowest cost, and an acceptable midrange dark current. This makes it the obvious choice for the system's NIR detector.

**Linear Stage Actuator Carriage Rail** There are several methods of detecting broad spectral regimes. Scanning is much more viable than staring in this case, since staring detection requires an array of detectors. This system is intended to keep costs low for the target user, and detectors are a disproportionately expensive part. Instead, this system will have a pair of detectors mounted on a carriage which will pass through the spatially separated beams of light. There are several important criteria for this part. It must be able to take small, precise, increment steps in order for the detectors to capture the soil emission with high resolution. It must have sufficient range of motion to cover the whole spectral range, from 400nm to 1700nm. And it must meet the size, weight, and power constraints imposed by the rest of the system.

Table 15: Linear Stage Actuator

Manufacturer	Rattmmotor	TOAUTO	TOAUTO	Zeberoxyz
Model	T0601-50	T0601-100	T0601-101	SFU1605
Stroke Length (mm)	50	100	50	200
Step Angle (°)	1.8	1.8	1.8	1.8
Step Size (mm)	.005	.005	.005	0.025
Voltage (V)	24	24	24	N/A
Current (A)	0.8	0.6	0.6	1.6
Cost (\$)	39.00	67.00	89.00	83.89

The ratio between the stroke length and step size needs to be well above 1,500 in order to meet the criterion for spectral resolution. Luckily, the product with the least range is well over that limit, at 10,000:1. That allows cost to become the driving factor. The T0601-50, distributed by Rattmmotor, is just over half the price of the next cheapest option. This is the group's choice of scanner

**Reflective Diffraction Grating** Diffraction is the operating principle behind the spectrometer. The soil emissions will be composed of different wavelengths representing the chemical structure of the soil, and this will be increased under illumination and then separated by diffraction. A diffraction grating separates incident electromagnetic waves of different frequencies by confining them into a tightly bounded region on reflection or transmission. This confinement causes the waves to disperse, like a prism would, with waves of higher frequencies experiencing greater dispersion. Transmission gratings will not be very effective for weak signal diffraction, as in this application, because most of the optical power is concentrated straight through and not separated at all, with only weaker orders fully separating. Reflective diffraction gratings will provide the best efficiency for soil spectroscopy. In order to prevent the beam from diffracting so far that it reflects straight back into itself, where it cannot be collected, it will be necessary to acquire a diffraction grating with grooves which are blazed at an angle, so that the surface normal of the grating has a greater angle than 90 degrees. The blaze angle corresponding to the 1000 nm diffraction regime will suffice. Next, it will be necessary to calculate the angular spread of diffraction from 400nm to 1700nm off a diffraction grating from several incident angles. The first order diffraction must not overlap with the incident beam if it is to be measured. Increased size is also desirable, in order to maximize the number of wave-groove interactions and boost diffraction efficiency.

Table 17: Diffraction Gratings

Manu- facturer	Model	Density lines/mm	Dimensions (mm)	Blaze Angle (nm)	Cost (\$)
Thorlabs	GR13-0610	600	12.7x12.7x6	1000	76.58
Thorlabs	GR25-1210	1200	25x25x6	1000	125.77
Edmund optics	Stock 43-745	600	12.7x12.7x6	1000	80.00
Edmund optics	Stock 43-753	1200	12.7x12.7x6	1000	80.00
MKS Newport	33025FL01- 520R	600	12.5x12.5x6	1000	155.00
MKS Newport	33025FL01- 530R	1200	12.5x12.5x7	1000	155.00
ScienceTech	631-0024	1200	50x50x9.5mm	1000	500.00

The spectral regime is too wide to be able to reliably use a 600 nm grating without losing information in the infrared. Unfortunately, efficiency is not a variable that can be compared from product to product, since it varies with wavelength and per polarization. This means the only remaining comparison is the ratio from cost to active area. ScienceTech can be ruled out immediately. Its cost to size ratio is not far from the other 1200 nm gratings on this list, but it would be difficult to justify doubling the project budget in order to achieve a marginal increase in grating power efficiency. This spec can also be compensated by cranking up the optical power probing the soil, presumably for less than the \$45 difference between the Edmund Optics and Thorlabs models.

**Focusing Optic (placeholder)** Fiber Patch Cable with attached Collimator (placeholder)

**LED Diode Array (placeholder)**

## 4 Design Constraints

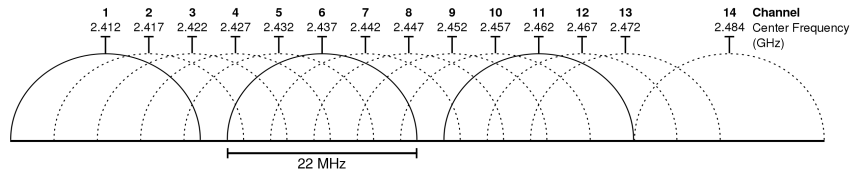
In this section we will cover the standards we will be adhering to throughout the development of our project as well as some of the realistic constraints that are imposed on us.

## 4.1 Related Standards

**C++20** C++ will be programmed according to the C++20 standard, formally known as [ISO/IEC 14882:2020](#). C++ is a superset of C, and builds upon it by introducing object-oriented programming concepts while maintaining the functional language aspect of C.

**802.11** The microcontroller (MCU) supports transmission through the Institute of Electrical and Electronics Engineers (IEEE) 802.11b/g/n standard of wireless communication. This standard uses the S band of radio frequencies and operates at 2.4 GHz. There are 14 accessible channels, each spanning a band width of 22 MHz (pictured inFigure 22).

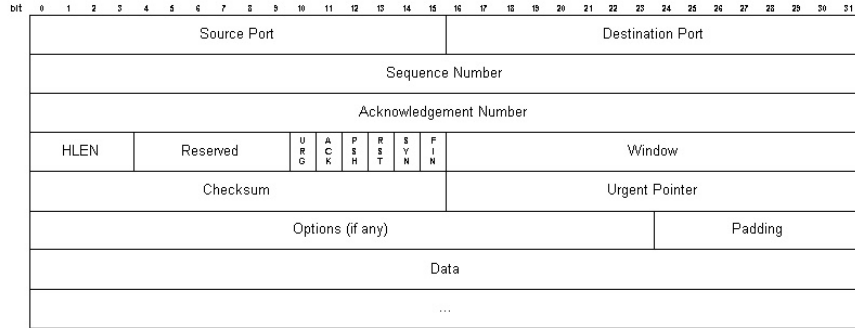
Figure 22: 802.11b/g/n channels



These channels specifically reside in an industrial, scientific and medical (ISM) band. This standard also provides datagram frames for the transport layer.

**TCP** Transmission Control Protocol (TCP) will be used to satisfy transport layer requirements of the product, and will be used to transmit symbols (i.e. from any commands, data, settings, telemetry, etc.) between Amazon Web Services (AWS) and the microcontroller (MCU). TCP was chosen over other protocols, such as User Datagram Protocol (UDP), mainly due to its reliability. The extent of TCP's reliability includes features such as checksums, duplicate data detection, retrying of transmissions, sequencing, and timers. Such reliability is favored over higher bandwidth or lower latency, as neither of the latter are required for the kilobytes of information being relayed between AWS and the MCU. A standard TCP frame is shown in Figure 23.

Figure 23: TCP frame ([The Transmission Control Protocol](#), Fig. 1)



**IPv4** TBD.

#### 4.1.1 JSON Web Token (RFC 7519)

This standard specifies a “compact, URL-safe means of representing claims to be transferred between two parties.” The means is via the JSON Web Token (JWT). JWTs are split into 3 parts, the header which specifies the algorithm the key(s) used to encrypt the message and the type of token, JSON Web Encryption (JWE) or JSON Web Signature (JWS). The second part is the payload. The payload is a JSON formatted object which carries claims. And the third part is the verification signature which is the Base64 encoded header plus a ‘.’ plus the Base64 encoded payload, another ‘.’ and finally this is all encrypted by the key. When a JWT is received, the header and payload can be read by just Base64 decoding these portions of the token. The JWT is verified by decrypting the verification signature, if the signature cannot be decrypted with the key then the token is invalid. Figure 24 gives an example of a JWT that is signed with a secret key.

Figure 24: Example JWT from jwt.io

**Encoded**
PASTE A TOKEN HERE

**Decoded**
EDIT THE PAYLOAD AND SECRET

```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0X3VzZXIiLCJyb2xlcYI6WyJST0xFX1VTRVUiXSwiZXhwIjoyNTE2MjM5MDIyfQ.srGzwwPg_MrNI6blBebwrmxvwPVtKm7lyGdNKEYsntE

```

**HEADER: ALGORITHM & TOKEN TYPE**

```

{
  "alg": "HS256",
  "typ": "JWT"
}

```

**PAYLOAD: DATA**

```

{
  "sub": "test_user",
  "roles": ["ROLE_USER"],
  "exp": 2516239022
}

```

**VERIFY SIGNATURE**

```

HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☒ secret base64 encoded

```

**Claims** Claims are in the payload of a JWT. The basic claims specified by this standard are **iss** “issuer”, **sub** “subject”, **aud** “audience”, **exp** “expiration”, **nbfi** “not before”, **iat** “issued at”, and **jti** “JSON Token ID”. For our purposes, all of the JWTs will be self-signed so the issuer, JSON Token ID, audience, and issuer claims can all be disregarded as they are meant for cross-service authentication. However, following the standard, the “subject” claim will be the user principal, and the expiration and issued at claims will be used to invalidate a token cookie after a certain period of time. The standard allows for public and private claims outside of these as well. Public claims are registered with the IANA while private claims are collision prone, neither of these are of true concern to us.

#### 4.1.2 HTTP/1.1 (RFC 2616)

This standard defines an application-level protocol communicating across the internet. RFC 2616 is an update to previous protocols that increase the capabilities of the original in the form of persistent connections and the ability to send files in the form of MIME types. MIME types hold metadata and the binary data of a file. This standard also serves to define how dates/times are handled on the web as well.

#### 4.1.3 WebSocket Protocol (RFC 6455)

The WebSocket Protocol standard is important to frame the way that the web server and microcontroller communicate. It specifies how the connection is made, how the data is formatted, and some security considerations. Most of these items are

abstracted away from users in the forms of libraries however the security concerns are likely the most important part of this standard in terms of this project. The reason being that we are receiving location and IP data from the packets in order to build out commands like where to point the solar panels. Exposing PII such as location to the world wide web is a major concern that we would like to avoid by following this standard.

## **4.2 Constraints**

The constraints discussed in the following subsections refer to limitations on the team's ability to design the system. Each section will discuss possible and realized problems with regard to the specific category of constraint. Following the discussion of the problem the team will discuss some possible or implemented solutions.

### **4.2.1 Economic**

Although different sponsorship options were explored as a source of funding for our product, ultimately our team has not received any sponsorship for this product. This process will be entirely self-funded and evenly divided between the four team members. As a result of this, component selection heavily considers unit cost as a factor. It follows that the parts selected for this endeavor will meet the minimum requirements, but may not have some of the "nice-to-have" features one may want from a specific component.

### **4.2.2 Time**

When it comes to time constraints, this project is on a deadline. Most projects are on a deadline in order to ensure the customer can make full use of the project as ordered. In this case, the project is part of a larger organization that runs through business cycles every year. The deadline in place is imposed by the restructuring of the university as a workforce.

An additional time constraint is university leave. As a workforce, UCF mandates that work be suspended for certain occasions, including hurricanes, as we were recently reminded.

Our design process has revolved around flexibility, and as a result we have not been forced to wait in order to acquire the products we need to design our project. However, as the spring approaches and designs become finalized, we will lose this flexibility.

### **4.2.3 Equipment**

There are three main equipment constraints on our ability to produce our project. One is limited access to software, another is heavily shared tools, and the third is having facilities to build.



This project arises out of a tradition of problem solving that is used in educational facilities and industries all over the country. A chief part of the College of Engineering and Computer Science program is to teach students to use the tools industry uses to solve the problems industry faces. So far, our team has used multiple integrated development environments, version control platforms, word processors and file viewers, text, video and audio communication services, and software for designing optical beam paths, CAD models of physical structures, and PCB simulations in order to express and determine design features for our project. Each of us has had tools that we would like to have used, but could not, either because of licensing, inexperience, or the need to work cooperatively with the larger team.

Another equipment constraint has been in the form of shared workspaces. In order to leverage these tools, we need relatively quiet space to sit where we can communicate with each other and run power hungry devices while connected to the internet and organizational networks. Our University has provided us with design labs, but their tools are managed by students, and inventory is often untraceable and disorganized.

The third equipment constraint is unique to the nature of this project, but it comes from needing to build outdoors. Building a garden bed requires land, if only a little, and although parts can be fabricated in clean spaces the system is designed to contain wet, heterogeneous dirt. Space outdoors is necessary to build and that means either permission from the University or leveraging team member access to land.

Each of these equipment constraints has a different effect on the project, some of them are easy to get around, by finding open source alternatives to licensed software, by cooperating with other students to get more out of leftover optical components, and by using student networks to determine where and when our group will have space to meet. Some are more difficult, causing us to trade time and money that we would rather not have had to give. In the end, equipment constraints will not likely impact the project deadline or affect the features as laid out in the document.

#### **4.2.4 Safety**

This project features several threats to human safety that must be addressed. The power subsystem is rated with sufficiently high voltage and current to cause cardiac arrest. The user must be protected from direct exposure to electrical conductors within the system, especially as this is an outdoor system with water management mechanisms which may add to the risk. The optical subsystem involves the use of infrared probes and focusing lenses. If handled carelessly, these could potentially create an eye hazard during testing or product use that the victim would not be able to detect. Steps must be taken to indicate the nature of the threat where it exists. Other risks involved in the project include the chance of a minor cut on a sharp surface becoming infected from exposure to the soil. The structure and components of the project must be sufficiently dull to ensure against the possibility of this.

#### **4.2.5 Environmental**

This project features a garden bed, which means that it will be directly engaging with the environment. Potential risks to the environment include project placement disturbing existing ecosystems. This can occur through fertilizer runoff, sound pollution, light pollution, introducing invasive species, poisoning local wildlife, entrapping or injuring animals, or destruction of a natural resource part of the ecosystem relies on. Another environmental consideration is the effect this project will have on the environment through engaging human systems such as drawing on the city power and water supply. Our team hopes to use subsystems to mitigate and even eliminate these potential consequences.

#### **4.2.6 Manufacturability**

Manufacturability is a set of important criteria for early in the design process to avoid making costly mistakes. There are two reasons a design, once complete, might not be manufacturable. One, the design cannot feasibly be produced. Two, the cost of the design is prohibitive. Our team has protected against cost prohibitive manufacturability issues by ensuring that our component selection is traceable and that each component is in production and within our price range at cost, not due to clearance discounts. Another step our team is taking against future constraints is selecting the bulk of our components and design features before taking steps toward production.

#### **4.2.7 Ethical**

Thankfully, our project makes use of institutions and practices that are well within the bounds of ethical behavior. Our garden bed will encourage age old activities that do not threaten the well being of the user or the surrounding community. Our project is designed to accommodate and minimize its environmental impact. Our use of wireless communication is extremely limited in its capability to cause harm to the user or to create a security risk for them or their neighbors. Users who engage with the product will not be able to use it to harm others, knowingly or unknowingly. The product does not in its nature risk damaging any cultural heritage or societies. Our team commits to submit to University policy as regards the goals and activities of this project.

#### **4.2.8 Sustainability**

Sustainability means creating a system that will produce as much of the resources as it uses so that the system does not collapse. This project does exactly that. The power generation subsystem is intended to ensure that our system does not need to draw on the power grid in order to function correctly. The water detection system is intended to ensure that water is not wasted reirrigating plants that have already been

fed by the rain. In a broader sense, our project will increase the wellbeing of people's lives, producing new plants and possibly food. One obstacle to sustainability is power storage. Our system uses a solar panel to collect energy, but that energy needs to be stored in order to distribute it in the right amounts and at the right times.

## 5 System Hardware and Software Design

### 5.1 Controller Subsystem

In order for our system to be as self and power efficient as possible from an end-user perspective, it was determined that our product would require an internet connection to offload remote command-and-control to an Amazon Web Services EC2 instance (hence referred to as "AWS" and detailed in subsection 5.4). To make the process of operating our product as hands-off as possible to end-users, the microcontroller will connect to the user's home WiFi network for access to AWS. Bluetooth, Zigbee, Thread, and other short-range 2.4 GHz communication protocols were disfavored over WiFi, as we predict most users will not have a device dedicated to connecting our product via such protocols. Long range (LoRa) protocols were deemed unnecessary, as the intended placement of our product is outside, near or next to the user's home. We do not expect our product to produce or receive large amounts of data, so the decreased bandwidth of a WiFi-enabled product being beyond the outdoor walls of a building is not a significant drawback to our application. A wired connection (802.3/Ethernet) was deemed too invasive to the end-user. It is expected that most, if not all, end-users have a wireless access point and internet access. Thus, connection via the 802.11/WiFi standard was a natural choice for our use case.

**CC3200 Overview** The Texas Instruments CC3200-series (hence referred to as the "CC3200", the "MCU", or the "microcontroller") of microcontrollers are WiFi-enabled chips with an ARM Cortex-M4 central processor and a WiFi network processor, along with many useful peripherals and power management modules. This series of processors is delivered alongside a software development kit (SDK) provided by Texas Instruments to ease the development of Internet-of-Things (IoT) applications.

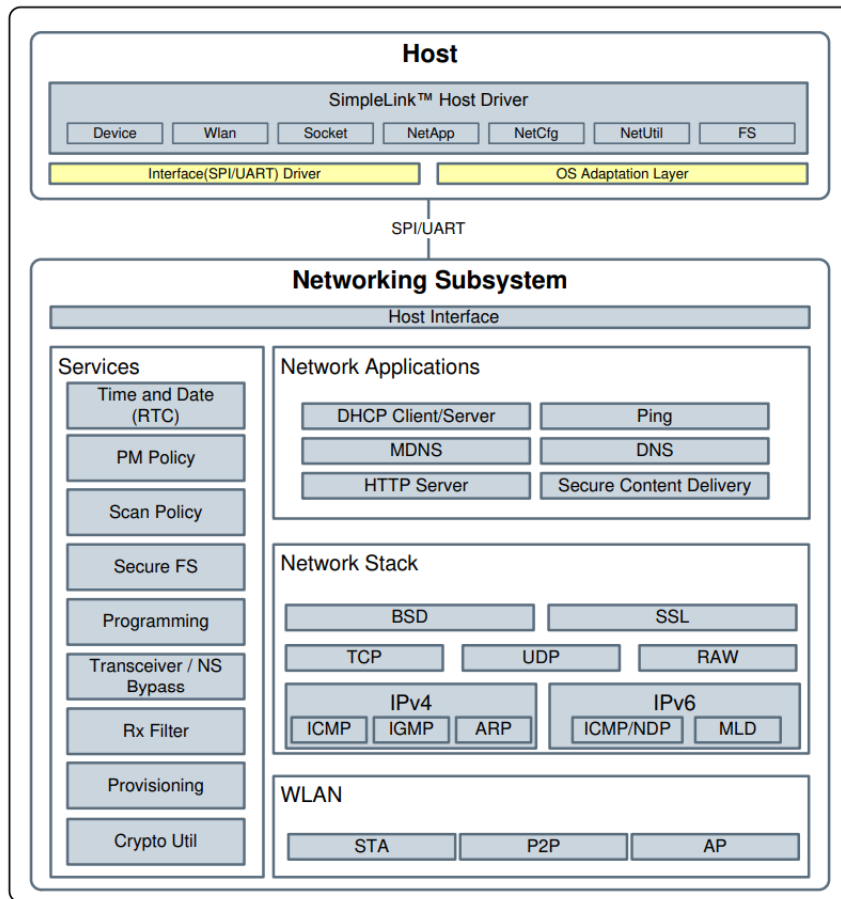
The CC3200's WiFi network processor supports the following standards/features useful to our development (see Figure 25):

- WiFi standards: 802.11b/g/n
- WiFi security: WEP, WPA/WPA2 PSK, WPA2 enterprise, WPA3 personal, WPA3 enterprise
- WiFi provisioning: SmartConfig, WPS2
- IP protocols: IPv4, IPv6

- IP addressing: static IP, DHCPv4, DHCPv6
- Transport: UDP, TCP, RAW
- Host interface: UART, SPI

The CC3200's networking subsystem is identical to the CC3220's, and is further detailed in Figure 25.

Figure 25: CC3220 networking subsystem (CC3200 SimpleLink™ Wi-Fi® and Internet of Things Technical Reference Manual)



**Connection** During development of our product, our microcontroller shall connect to a WLAN via a hardcoded password (and username, if needed), or via SmartConfig. The system shall receive an IPv4 address via DHCP, or it shall be able to be statically assigned.

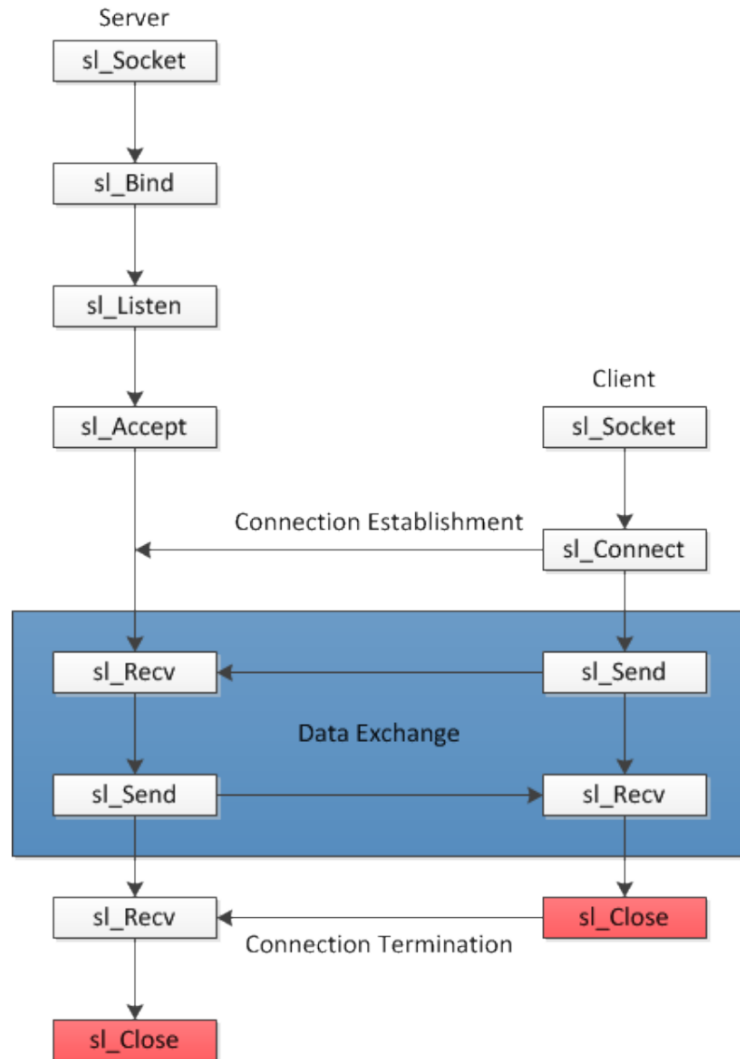
**Web Interface Option** The first option would allow unanimous adaptation of our product for home users. This option consists of performing first-time setup through

a WiFi Direct connection to the MCU and allowing the user to login through a web portal to finish setup of their device. The portal would allow a user to connect to their WLAN of choice, and would also allow the user to login to the MCU locally to view telemetry and settings. The user would also be able to configure the MCU for static IP addressing or DHCP IP addressing.

**WPS Option** The second option would be to implement a momentary switch which is programmed to connect to the MCU's WiFi Protected Setup (WPS) feature. This would still allow the user to connect to their network of choice—however, the user's wireless access point (WAP) would need to support WPS. Furthermore, DHCP IP addressing would be forced, and there would be no web portal for the user to access for viewing telemetry and settings. This option would greatly ease development, though, and would let the MCU and web teams focus on developing and refining their respective subsystems.

**Communication Method** The MCU will communicate with AWS through TCP sockets. After connection to the user's home network, the MCU will check if there is an internet connection. Once an internet connection has been established, the MCU will open a socket and connect to the AWS instance via its URL (using the default DNS nameserver provided to network clients). The software control flow of TCP sockets as our product will implement them is detailed in Figure 26.

Figure 26: TCP socket control flow (CC3100/CC3200 SimpleLink Wi-Fi Internet-on-a-Chip User's Guide (Rev. C), Fig. 5-1)



**Connection Parameters** Once the MCU has established a connection to the internet and the AWS instance, it attempts to send current system settings and telemetry, as well as sensor readings to AWS. The microcontroller does this at least once every 15 minutes. Sending and receiving may occur more often if commanded to by AWS. Because TCP is being used to connect AWS and the microcontroller, any manual retries on a failed send or receive most likely will be futile. Therefore, any sort of link error handling will be performed by the link and not the microcontroller program.

**Over-the-Air Updates** At this time, our team does not intend to provide a method for over-the-air updates (OTA), however, this is a provision that may be developed in the future.

**Startup** Upon startup, the system shall initialize these peripherals in the following order within 30 seconds of startup: watchdog timer, GPIO, ADC, network stack/WiFi module. The system shall then check for interrupts, and execute ISRs if necessary. The system shall then verify its connection to the internet, if set, by attempting to ping 8.8.8.8 and 8.8.4.4 16 at most times over the period of 8 seconds. If 4/16 responses have been received at any point, the system shall attempt to verify network DNS status by pinging `google.com` 16 at most times over the period of 8 seconds. If 4/16 responses have been received, the system shall continue with startup and clear the no connection flag (if set). If no connection has been established (i.e. the system fails to ping the aforementioned addresses at either step), the system shall set a flag to indicate no connection has been received, and reset. If no connection has been established, and the flag is set, the system shall go into critical power mode and flash the yellow LED for 1 second, 1 time every 2 seconds. The user shall be able to correct the issue via user interface (e.g. SmartConfig or other developed interface). The system shall measure the battery voltage, and set itself to the appropriate state described earlier.

**Shutdown** The system shall generate an interrupt to shutdown the system. The shutdown ISR shall run and perform the following actions. The system shall finish any RF transmissions within 5 seconds. The system shall terminate all remaining RF transmissions after the 5 second window has elapsed. The system shall save any network parameters to nonvolatile memory to be used upon startup within 1 second. The system shall safely shutdown any initialized peripherals within 5 seconds. The system shall safely shutdown any sensing components within 5 seconds. The system shall "hand-off" any control it had over the power subsystem to the battery's charge controller within 5 seconds. The system shall go into the shutdown power mode, stopping the processor within 1 second.

**Reset** In the event of a reset state, the system shall perform the shutdown ISR, with each section of the ISR performing its necessary tasks within the allotted time mentioned above. The system shall perform its startup ISR, with each section of the ISR performing its necessary tasks within the allotted time mentioned above. There shall be a time period of no longer than 5 seconds between the end of the shutdown ISR and the beginning of the startup ISR.

**Networking** The system shall be able to connect to the user's A 2.4 GHz-based wireless network. It may either have no security, or be secured with WPA2 (Personal or Enterprise). The system shall be able to connect if the signal strength of the network is nominal (around -70 dBm). The network shall adhere to the standards of 802.11b, g, or n. The system shall received an IPv4 address assigned by network DHCP, or the user shall be able to statically assign a local IPv4 address. The network shall use NAT, and not expose the product to the WAN. The system shall use DNS provided by the WLAN gateway. The system shall be able to connect to the internet

within the timeframe described by the startup sequence. The system shall be ready to establish a connection to AWS within 5 seconds of verifying connection to the internet.

**AWS** The system shall first establish a connection to AWS by sending a burst of 5 data packets of the size and containing the variables described in Figure 29 over a period of 25 seconds. If no connection is established over a period of 15 minutes, the system shall send the same burst of data packets and repeat every 15 minutes until a connection is established.

**Receive (Sockets)** The system shall listen for commands from AWS via TCP sockets. If no traffic is received from AWS within a period of 40 minutes, the system shall terminate and attempt to reopen the socket. The system must be able to open and configure a socket within 30 seconds.

**Send (Sockets)** The system shall send data to AWS via TCP sockets. If no traffic is successfully sent to AWS within a period of 40 minutes, the system shall terminate and attempt to reopen the socket. The system must be able to open and configure a socket within 30 seconds.

**Receive (Command)** While a connection is established with AWS, the system shall continually listen for commands from AWS. The system shall be able to fully receive and decode commands from AWS within 10 seconds of the start of the reception. The system shall be allotted 1 second per command to schedule the received commands.

**Send (Data)** While a connection is currently established with AWS, the system shall send a burst of 2 data packets over a period of 10 seconds every 15 minutes. The data packets shall be of the size and contain the variables described in Figure 29.

**Solar and Battery** The system shall monitor and update the solar array current and voltage output values at least every 10 seconds. The system shall run in active mode between sunrise and sunset, while the battery has 40-100% calculated charge remaining. The system shall run in low power mode between sunrise and sunset while the battery has 10-40% calculated charge remaining, and between sunset and sunrise while the battery has 10-100% calculated charge remaining. The system shall run in critical power mode while the battery has 5-10% calculated charge remaining. The system shall be in shutdown mode while the battery has 0-5% calculated charge remaining. The system shall be able to switch power modes within 5 seconds of the interrupt being triggered.



Table 19: MCU power mode behavior

Time of Day	Calculated Battery Charge (%)	Power Mode
Day time	100-40	Active
Night time	100-40	Low
Any time	40-10	Low
Any time	10-5	Critical
Any time	5-0	Shutdown

**Sensing** The system shall monitor and perform analog-to-digital conversion on the spectroscopic sensor values at least every 10 seconds while not in shutdown or critical power mode.

**Commands from AWS** The MCU will receive commands and data from AWS in the following form:

$\mathbf{t\ x\ c\ y}$

where  $\mathbf{t}$  (character) marks the beginning of the command string,  $\mathbf{x}$  (unsigned integer) indicates how many seconds to wait before executing command  $\mathbf{c}$  (character) with parameter  $\mathbf{y}$  (ambiguous). The following characters occupy the spot of  $\mathbf{c}$ , and translate to the following commands:

- $\mathbf{w\ y}$ : water,  $\mathbf{y}$  is boolean
- $\mathbf{a\ y}$ : solar  $\theta$ ,  $\mathbf{y}$  is double
- $\mathbf{b\ y}$ : solar  $\phi$ ,  $\mathbf{y}$  is double
- $\mathbf{l\ y}$ : low power,  $\mathbf{y}$  is unsigned int for flags
- $\mathbf{s\ y}$ : send data,  $\mathbf{y}$  is unsigned int for flags

For example, if AWS instructs the MCU to turn on the water source in 3 minutes, it would send the command  $\mathbf{t\ 180\ w\ 1}$ . If AWS would like the MCU to change the  $\theta$  of the solar panels to  $45^\circ$  immediately, it would send  $\mathbf{t\ 0\ a\ 45.0}$ . It is important to note that the values of  $\mathbf{x}$  and  $\mathbf{y}$  are not strings (e.g.  $45^\circ$  will not be sent as " $45.0^\circ$ "), but the actual encoding of  $45^\circ$  as a double-precision float. This is to maintain a consistent command string size amongst all transmissions.

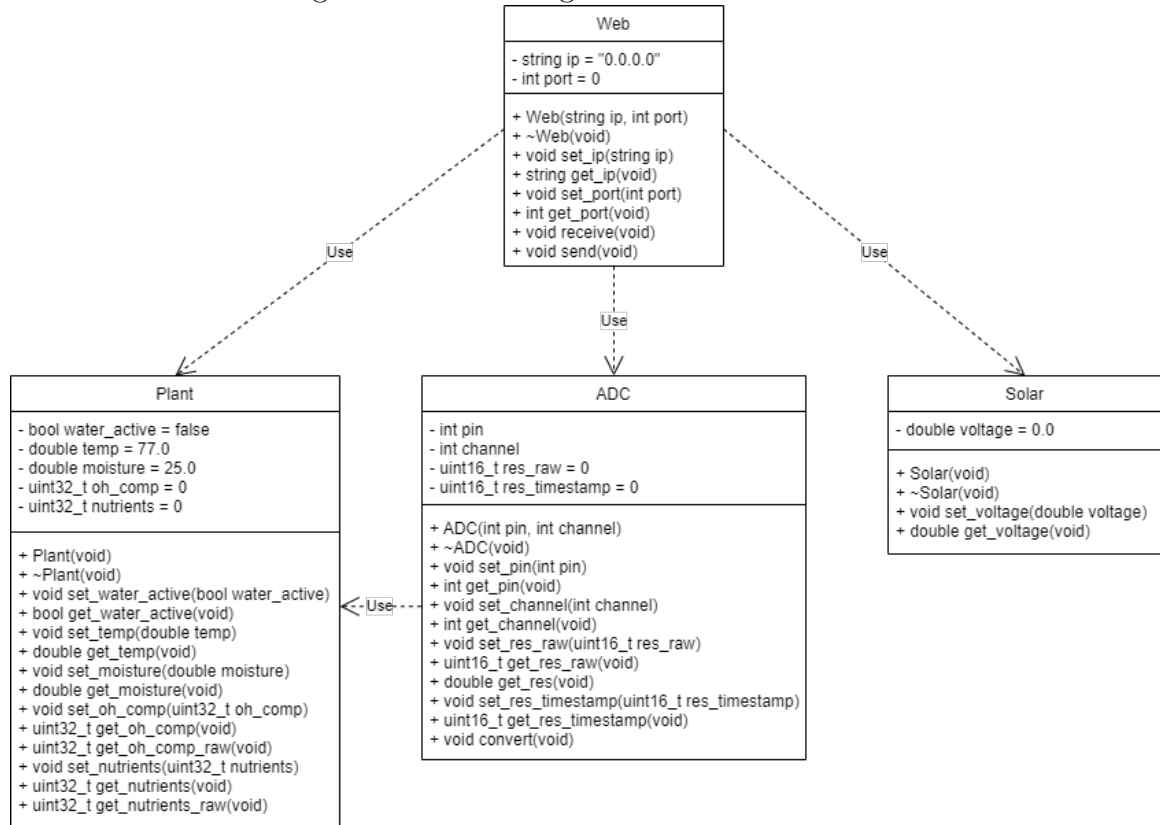
Figure 27: Bitwise representation of command

$\mathbf{t}$ char [111:104]	$\mathbf{x}$ unsigned int [103:72]	$\mathbf{c}$ char [71:64]	$\mathbf{y}$ ambiguous [63:0]
-----------------------------------	--	---------------------------------	-------------------------------------

**C++ Structure** It should be expected that all headers from the C++ standard library (as defined in C++20) will be used, along with the following nonstandard libraries: Texas Instruments SimpleLink™ CC32xx SDK. Additionally, it is expected that the MCU program will schedule tasks using a Singleton Pattern thread to ensure thread safety when accessing variables.

**Classes** There will be a few classes defined in the MCU's programming, detailed in Figure 28. Classes will be laid out in the header file according to the definitions given in the UML diagram.

Figure 28: UML diagram of the classes used



**Web Class** The Web class will have 2 private variables: `ip` (string) and `port` (int). `ip` signifies the IPv4 address of the AWS instance to connect to, and `port` is the port to access AWS through. The default state of this class is `ip = "0.0.0.0"` and `port = 0`. It will have a public destructor and a constructor, requiring the variables string `ip` and int `port` to be passed to the constructor. The class will have 4 public setters and getters: `void set_ip(string ip)`, `string get_ip(void)`, `void set_port(int port)`, `int get_port(void)`. The class will have 2 public functions: `void receive(void)` and `void send(void)`. `receive()` will be used to receive the next command from AWS, and will then decode the command and use functions from

other classes to execute the command received from AWS. `send()` will use functions from other classes to combine and send the current sensor readings and settings to AWS. Only 1 object of class `Web` will be instantiated.

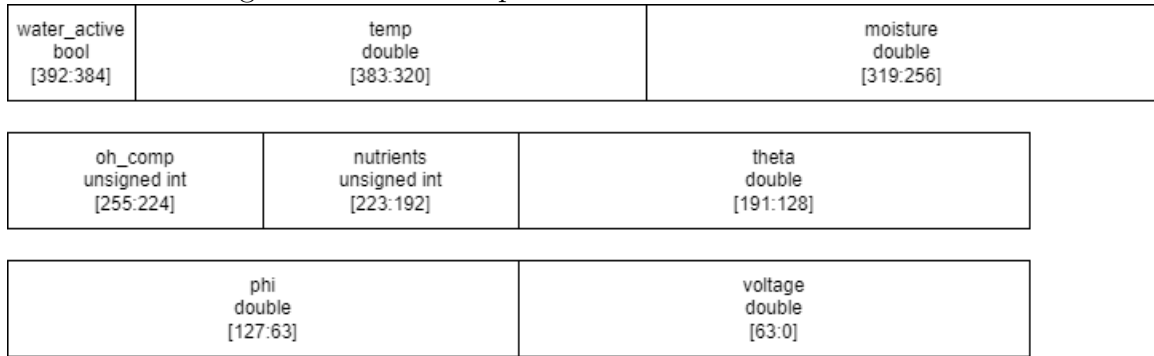
**Plant Class** The `Plant` class will have 5 private variables: `water_active` (bool), `temp` (double), `moisture` (double), `oh_comp` (unsigned 32-bit int), and `nutrients` (unsigned 32-bit int). `water_active` is the state of the water source feeding the plant, `temp` is the current air temperature near/around the plant, `moisture` is the moisture percentage of the soil near/around the plant, `oh_comp` is an encoded word of parameters relating to the spectroscopic analysis of the soil's OH composition, and `nutrients` is an encoded word of parameters relating to the spectroscopic analysis of the soil's nutritional composition. The default state of this class is `water_active = false`, `temp = 77.0` (room temperature in °F), `moisture = 25.0` (approximate average soil moisture), `oh_comp = 0`, and `nutrients = 0`. It will have a public deconstructor and a constructor, requiring no variables to be passed to the constructor. The class will have 12 public setters and getters: `void set_water_active(bool water_active)`, `bool get_water_active(void)`, `void set_temp(double temp)`, `double get_temp(void)`, `void set_moisture(double moisture)`, `double get_moisture(void)`, `void set_oh_comp(uint32_t oh_comp)`, `uint32_t get_oh_comp(void)`, `uint32_t get_oh_comp_raw(void)`, `void set_nutrients(uint32_t nutrients)`, `uint32_t get_nutrients(void)`, and `uint32_t get_nutrients_raw(void)`. The purpose of the normal getters for `oh_comp` and `nutrients` are to get human-readable formats for the OH composition of, and nutrients in the soil. The raw getters are to get the raw values of `oh_comp` and `nutrients`. The setters accept the raw values of `oh_comp` and `nutrients`. Only 1 object of class `Plant` will be instantiated.

**ADC Class** The `ADC` class will have 4 private variables: `pin` (int), `channel` (int), `res_raw` (unsigned 16-bit int), and `res_timestamp` (unsigned 16-bit int). `pin` is the physical pin of the voltage input to the ADC on the CC3200, `channel` is the ADC channel corresponding to the selected pin. `res_raw` is the raw result of the ADC for the corresponding channel. `res_timestamp` is the timestamp of the ADC result for the corresponding channel. The default state of this class is `res_raw = 0` and `res_timestamp = 0`. It will have a public deconstructor and a constructor, requiring the variables int `pin` and int `channel` to be passed to the constructor. The class will have 9 public setters and getters: `void set_pin(int pin)`, `int get_pin(void)`, `void set_channel(int channel)`, `int get_channel(void)`, `void set_res_raw(uint16_t res_raw)`, `uint16_t get_res_raw(void)`, `double get_res(void)`, `void set_res_timestamp(uint16_t res_timestamp)`, `uint16_t get_res_timestamp(void)`. The `get_res()` getter converts the raw result of the ADC into a voltage. The class will have 1 public function: `void convert(void)`. This function manually triggers conversion in the ADC module of the CC3200. Multiple objects of class `ADC` will be instantiated.

**Solar Class** The Solar class will have 1 private variable: `voltage` (double). `voltage` is the voltage of the battery updated at regular intervals. The default state of this class is `voltage = 0.0`. It will have a public deconstructor and a constructor, requiring no variables to be passed to the constructor. The class will have 2 public setters and getters: `void set_voltage(double voltage)`, and `double get_voltage(void)`. Only 1 object of class Solar will be instantiated.

**Data to AWS** Unlike commands received from AWS, the MCU will simply send data directly from its classes to AWS. This is done to minimize the overhead of sending extraneous symbols and preserve power stored in the battery, as receiving uses far less power than transmitting in the MCU subsystem. Further optimization can be performed in the future to further reduce overhead (e.g. reducing `water_active` to occupy 1 bit and using the rest of the symbol to encode other data).

Figure 29: Bitwise representation of data sent to AWS



**Global Variables** No global variables plan to be implemented at this time. Macros may be defined for configuration of the ADC and any other dependent modules.

**Interrupts and ISRs** If the charge controller indicates that the battery has fallen below a certain voltage (named "low voltage"), the MCU will raise an interrupt and execute `isr_low_power()`. This ISR performs housekeeping before putting the MCU into a low power state, limiting use of its functions and consuming less current.

If the charge controller indicates that the battery has fallen below a certain voltage (named "critical voltage"), the MCU will raise an interrupt and execute `isr_critical_power()`. This ISR performs further housekeeping before putting the MCU into an extreme low power state, limiting all but the features necessary to maintain a connection with AWS and consuming a minimal amount of current.

If the MCU, AWS, or any other devices transmit indication of a dangerous state or if the MCU, AWS, or any other devices transmit indication of a shut down, the MCU will raise an interrupt and execute `isr_shut_down()`. This ISR immediately shuts down all able subsystems, and puts all other subsystems in a fail safe state. This ISR fails safe the entire system.

If the MCU receives indication of a start up (via a momentary switch), the MCU will raise an interrupt and execute `isr_start_up()`. This ISR starts up all relevant subsystems and begins the MCU's programming. All classes will be initialized to default values. This ISR starts up the entire system.

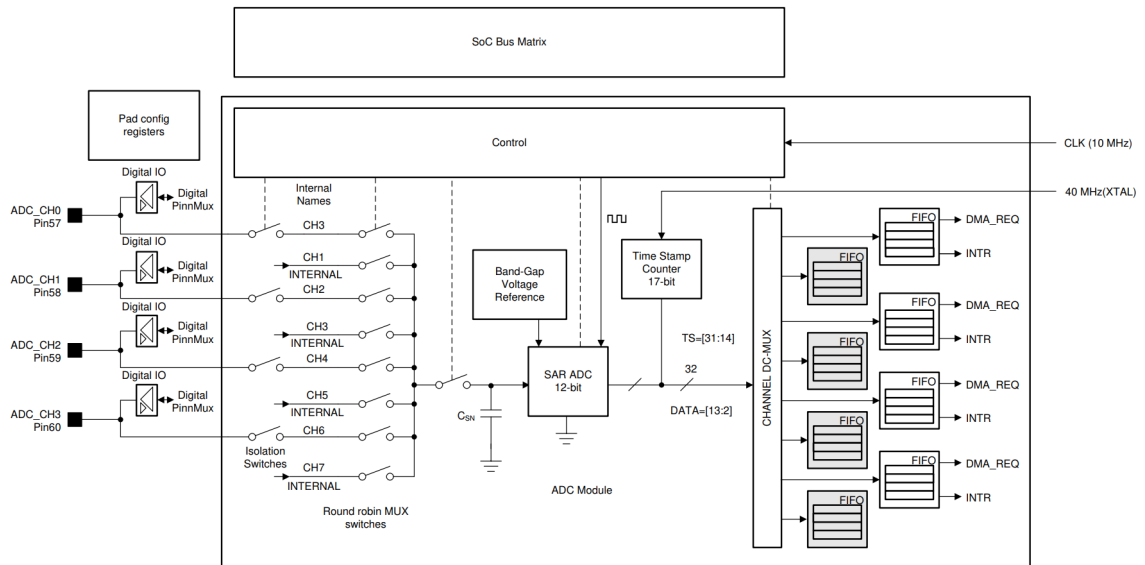
If the MCU determines it must reset the system to a default state, the MCU will raise an interrupt and execute `isr_default_state()`. This ISR returns all relevant subsystems to their default state, as if the MCU had just executed `isr_start_up()`. All classes will be initialized to default values. This ISR resets the entire system.

If the watchdog timer needs to be reset to 0 in the course of normal operation, the triggered interrupt shall execute `isr_wdt()` to reset the WDT.

**File Organization** Classes, macros, functions, and variables will be defined/prototyped to the extent required in a header file to be named `garden.h`. These classes, functions, and variables will be further defined in a separate source file named `garden.cpp` as needed. The `main()` function and the remaining classes, macros, functions, and variables, required will be located in `main.cpp`.

**Analog-to-digital Conversion** The CC3200 contains a 12-bit general purpose analog-to-digital converter (ADC) with 4 externally-accessible channels and a sampling periodicity of 16  $\mu$ s per channel (62.5 ksp/s per channel). The architecture of the CC3200's ADC is shown in Figure 30.

Figure 30: CC3200 ADC module architecture (SimpleLink™ Wi-Fi® CC32xx ADC, Fig. 1-2)



It is expected that the spectrometer circuit will provide a current between 20  $\mu$ V and 80 mV. The 12-bit ADC provides for an input range of between 0 and 1.8 V.

Therefore, the resolution of the ADC is calculated below:

$$\frac{(1.8 - 0) \text{ V}}{2^{12} \text{ steps}} = 0.4395 \text{ mV/step} \quad (2)$$

Thus, given our ADC resolution and the expected range of our spectrometer, the number of discrete steps of range is given below:

$$\frac{(80 - 0.02) \text{ mV}}{0.4395 \text{ mV/step}} = \lfloor 181.98 \rfloor = 181 \quad (3)$$

These steps will be used to measure OH composition and nutrients in the soil, and functions from the class `ADC` will be used to perform analog-to-digital conversion to update values in the class `Plant`.

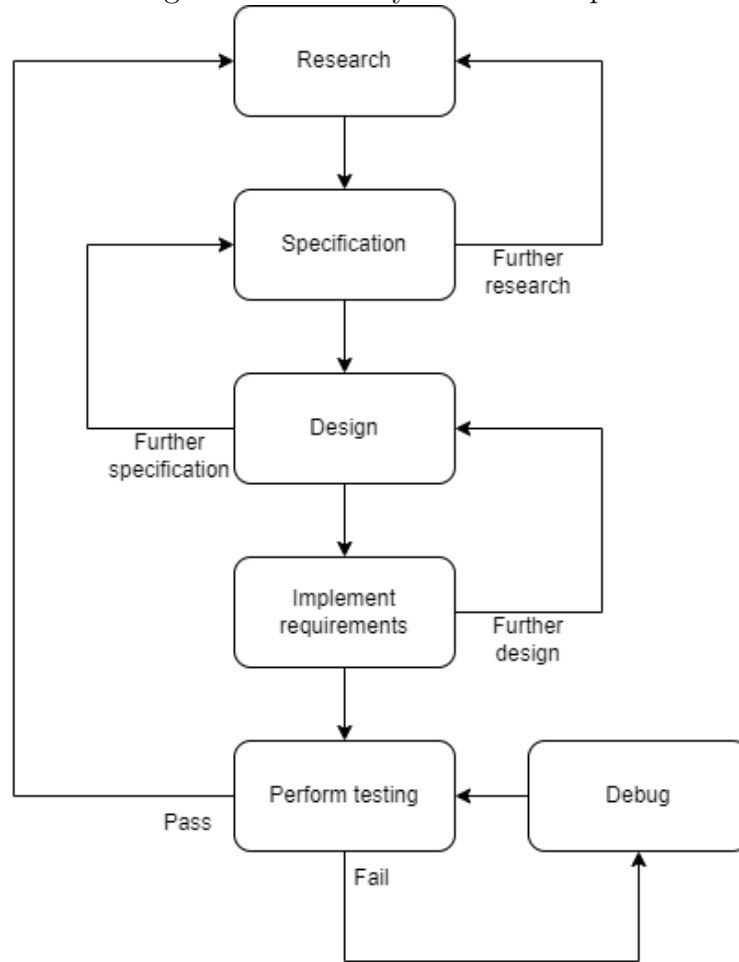
In the future, a more precise ADC module could be explored to give the CC3200 more resolution when performing spectroscopy.

**Communication with Servos** The MCU will directly control GPIO and bit-bang values to the servo motors controlling the orientation of the solar panels when using functions from the class `Solar`. This method of control may be refined further in the future.

**Telemetry** No data that would be exclusively considered telemetry will be transmitted between AWS and the MCU (e.g. processor temperature). Instead, all "telemetry" values will be handled with interrupts and ISRs/functions on-chip, while current settings and sensor readings will be transmitted back to AWS.

**Development Model** An Agile development model will be used. Code reviews will be performed on an as-needed basis by a convening of members of the MCU subsystem and the web subsystem teams.

Figure 31: UML diagram of the subsystem’s development methodology

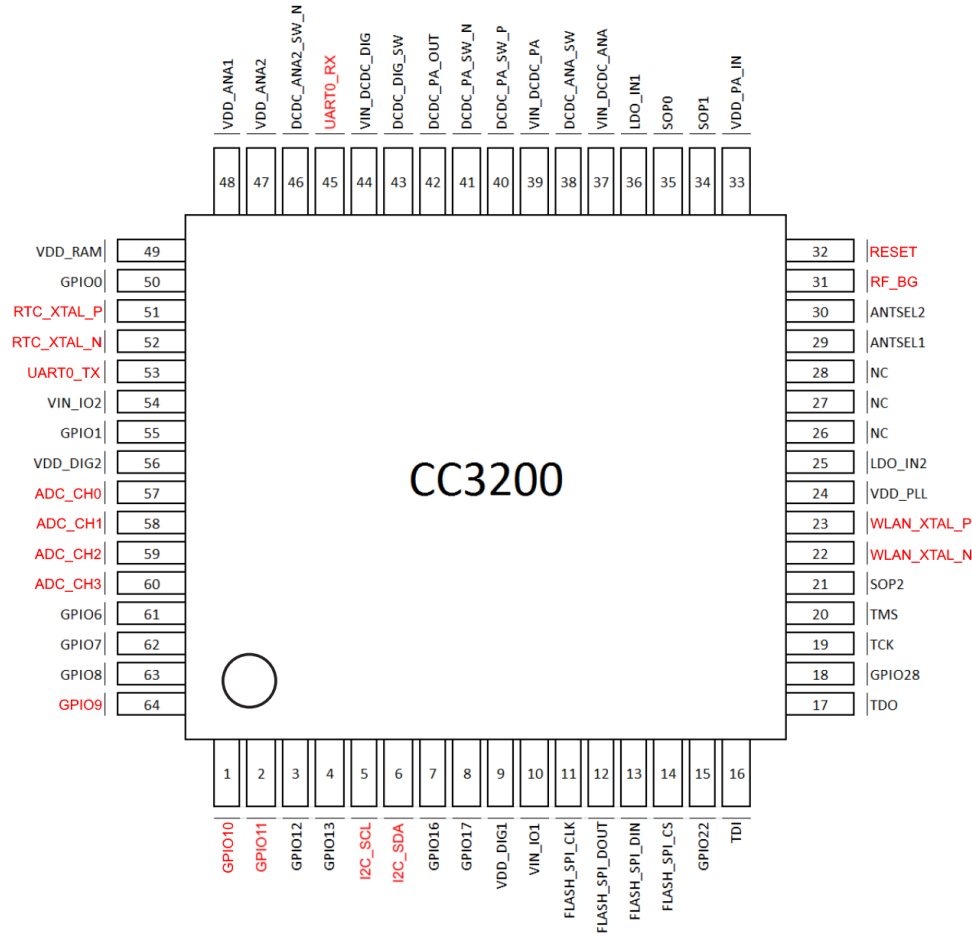


**IDE and Git** Texas Instruments Code Composer Studio v12 will be used to program, compile (via TI ARM compiler v20), and debug the C++-based project. GitHub will be used as a repository for the project, using Git for version control.

**Power** The MCU will be connected to power from the power subsystem detailed in subsection 5.2. It is expected to receive 3.3 V, and this will be fed to the integrated DC/DC converter on the CC3200. The converter is able to convert voltages from 2.3 to 3.6 V.

**Pinout Diagram** The CC3200 will be connected to the following components as described by the pinout diagram shown in Figure 32.

Figure 32: CC3200 pinout diagram (red labels indicate pins will be connected to other components in a way that differs from the reference design)



**LEDs** The system shall be able to make use of its onboard LEDs for notifying the user or developers of system states. When the system is in active mode, the green LED shall be solidly illuminated. When the system is in low power mode, the green LED shall flash 1 time for a period of 0.5 seconds, every 2 seconds. When the system is in critical power mode, the red LED shall flash 2 times for a period of 0.5 seconds per flash, separated by 1 second between each flash, every 30 seconds. When the system is in shutdown mode, no LEDs shall be active. When the system is starting up, the green and red LED shall be solidly illuminated until the startup sequence is completed and the system transitions into a different power mode. The yellow LED shall solidly illuminate while receiving data through the RF antenna for the duration of the reception. The yellow LED shall blink 1 time every 0.2 seconds for a period of 0.1 second while transmitting data through the RF antenna for the duration of the transmission.



Table 20: MCU LED operation

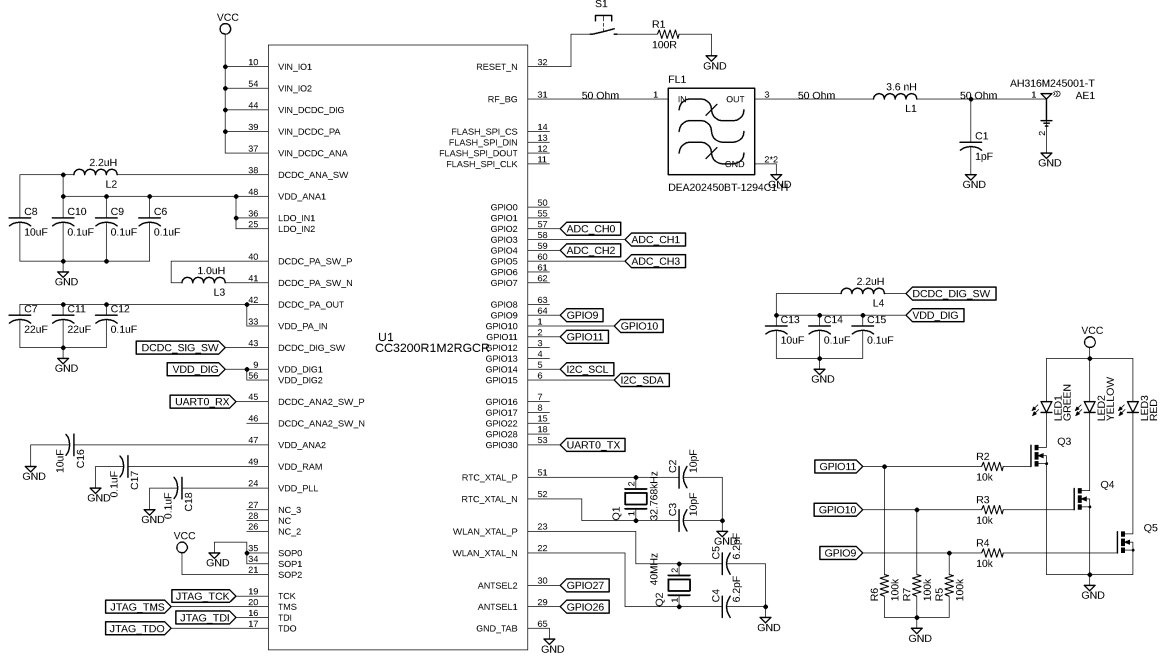
Power Mode	Red LED	Yellow LED	Green LED
<b>Active</b>	Not active	RX: Solidly illuminated, TX: 1 flash of 0.1 s, every 0.2 s	Solidly illuminated
<b>Low</b>	Not active	RX: Solidly illuminated, TX: 1 flash of 0.1 s, every 0.2 s	1 flash of 0.5 s, every 2 s
<b>Critical</b>	2 flash of 0.5 s, 1 s separation, every 30 s	No connection: 1 flash of 1 s, every 2 s	Not active
<b>Shutdown</b>	Not active	Not active	Not active
<b>Startup</b>	Solidly illuminated	Not active	Solidly illuminated

**Watchdog Timer** The CC3200 has an onboard watchdog timer used to detect and recover from freezes, crashes, or lockups. When the watchdog timer reaches zero, the watchdog module shall trigger an interrupt. The ISR triggered by the interrupt shall reset the watchdog timer. In the case the ISR is not able to be triggered by the interrupt (e.g. system freeze), the watchdog module shall reset the system. After the system is reset by the watchdog module, the system shall start up normally.

**Printed Circuit Board** Our controller subsystem will be developed and implemented on a Texas Instruments [CC3200SF-LAUNCHXL](#) LaunchPad development board with a plug-in daughterboard hat containing sensing and other components. In an ideal scenario, our team would like to develop a custom PCB for our controller subsystem—however, decreased supply and increased lead times, as well as the extra time needed to develop, test, and implement a PCB forced our hand to use the development board in the final product.

If Texas Instruments’ global integrated circuit supply increases and MCUs become more available, and our team is on-time with project goals and objectives, we would like to explore design of a PCB for the controller subsystem. Because the product is using the reference design currently, our team does not yet have a custom schematic for the controller subsystem. Thus, our product’s schematic may be found in Figure 33.

Figure 33: MCU subsystem schematic

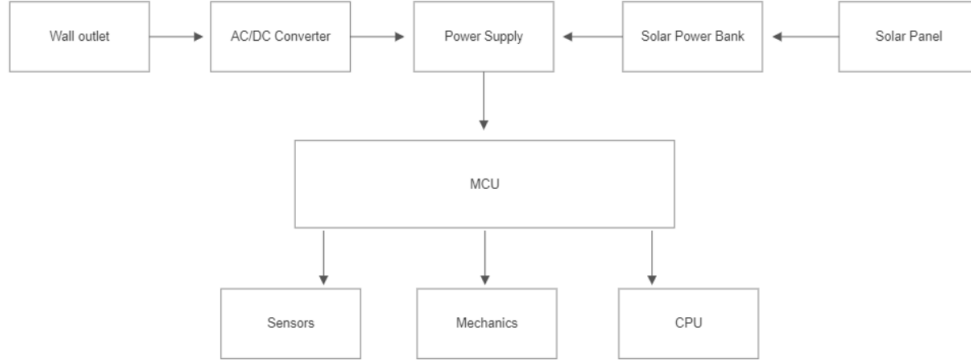


## 5.2 Power Subsystem

The power system is an important part to any electrical device or component that requires any amount of power. It simply starts from a power source such as a battery or wall outlet then is converted into energy to operate what is being used. At the minimum this model is designed to be an independent system, having the capability to operate on its own. The way that can be achieved is through solar power.

Solar power has been such a strong growing source of energy and will play a vital role in our system. Power is collected through the solar panels and then regulated through the charge controller. The power is regulated through the charge controller so that the battery is not being over charge, which could potentially damage it. Of course, power collected from the solar panels needs to be stored in a battery, which will also be used to power our system. From there, voltage needs to be regulated once more for the microcontroller and other components are powered properly. The block diagram in Figure 34 shows the flow of power through the system.

Figure 34: Power subsystem block diagram



The block labeled “Mechanics” in Figure 34 refers to the solenoid valve for controlling water flow as well as the control scheme for actuating the solar panels to maximize power efficiency.

### 5.2.1 Solar Panel Control

To get a greater degree of accuracy the use of stepper motors are used. First, based on the choice of solar panels we get the mass and dimensions of the solar panels, these are .76kg and .336m x .2m respectively. This is important for calculating the torque. We only need to provide two degrees of freedom, one about the short axis (the horizontal axis that bisects the .2m side) and the vertical axis (the axis at the intersection that bisects the two sides of the panel). From the length and mass we can calculate the force per unit length:

$$\frac{.76\text{kg}}{.336\text{m}} = 22.62 \frac{\text{N}}{\text{m}} \quad (4)$$

Now, knowing that torque is  $F \times d$  we can formulate the torque for any given angle about this axis in the following:

$$\int_0^{.168} 22.62 \times x \times \sin \theta \, dx \quad (5)$$

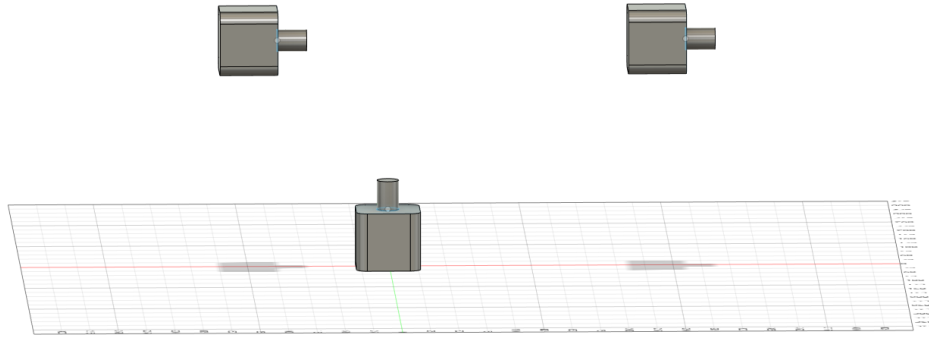
Equation 5 gives the torque for any given angle. Something you might have noticed is that this is for only one side of the panel. The full torque about the central axis is as follows:

$$\int_0^{.168} 22.62 \times x \times \sin \theta \, dx - \int_0^{.168} 22.62 \times x \times \sin 180 - \theta \, dx \quad (6)$$

From Equation 6 we can find the maximum and minimum values of the torque about the axis by finding the crossings of the first derivative with respect to  $\theta$  and finding the concavity in the second derivative with respect to  $\theta$ . Doing so proves the intuition that there is no torque around either axis, this equation evaluates to 0 for all  $\theta$ .

To save on power Figure 35 shows that two motors will work to rotate the panels about the short axis while the center axis is actuated by a single motor and belt system.

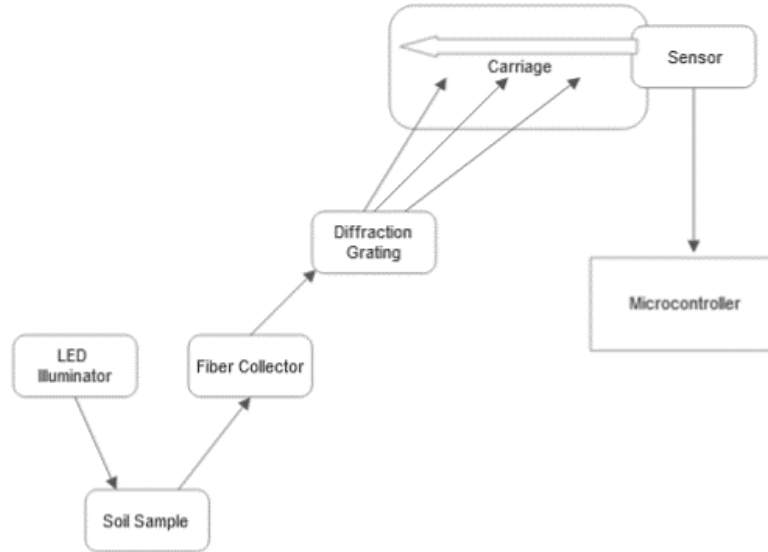
Figure 35: Stepper motor configuration



The reason we have to use two stepper motors for the short axis rotation is that because as the panels rotate about their central axis any rod or pulley or tensioner system would come out of alignment. Another advantage of this due to the lack of torque in the system is that using a motor controller the two motors can be driven with a single H-bridge with a minimum penalty to power, this will ensure that both panels are always pointed at the same angle in the desired direction.

### 5.3 Sensing Subsystem

Figure 36: Sensing subsystem block diagram



The sensing subsystem is an infrared spectrometer that uses two photodiodes as detectors, a diffraction grating as a spectral separator, an array of LEDs to illuminate the target area, and optics to collect, collimate, and focus the beam. In order to determine the component positions and interaction, each component will have to be addressed.

- Ground
- LEDs
- Fiber in
- Fiber out collimated
- Reflective Diffraction Grating
- Focusing Lens
- Carriage driven Photodiodes
- Filter Circuitry

**Dirt** That's right, the first part of the optical subsystem is the soil. Structures in the soil emit trace electromagnetic signals that correspond to the soil matrix, soil temperature, soil moisture, and chemical content of the soil. (Link papers) In order to ensure sufficient signal to noise ratio, the dirt is going to have to be probed by a bright, broadband source covering the visible and Near Infrared regime. Halogen lamps will do (link papers) but it may be possible to achieve the same effect with LEDs near the surface. An additional concern is that irregularities in the soil topography may reflect back light that would otherwise indicate organic components or water. In order to counter this, it is going to be necessary to flatten the dirt with a stamp or roller.

**LEDs** Placeholder until followup with ocean insight.

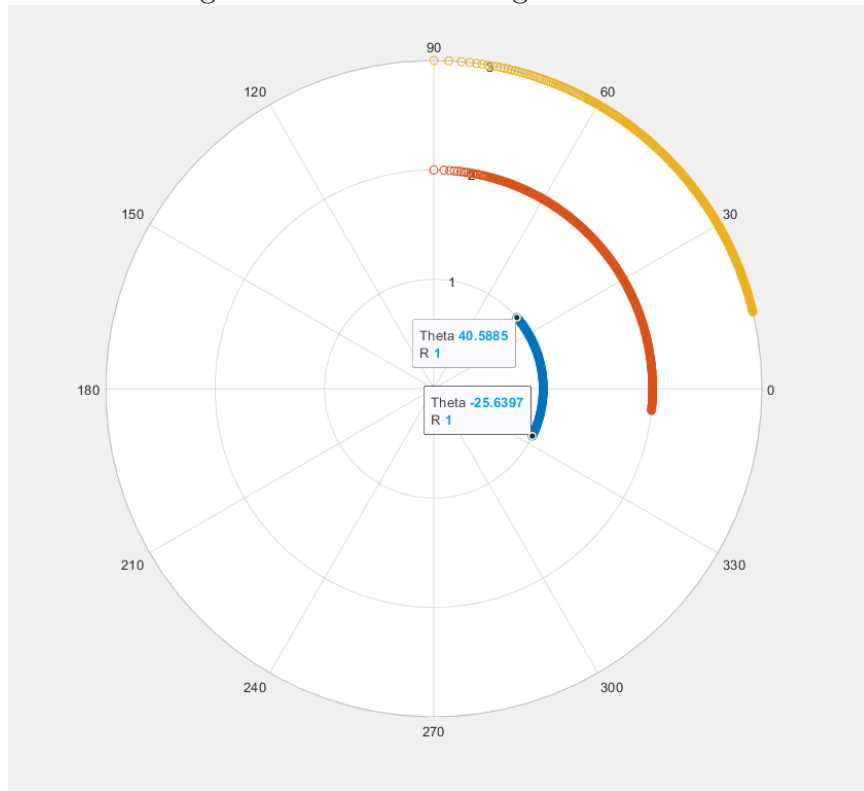
**Fiber in** placeholder

**Fiber out collimated** The fiber collimator selection hasn't been finalized yet, but for now I'm assuming a beam of about 2mm diameter will come out of the fiber head collimated toward the grating.

**Grating** The grating will be a 1200 groove per mm aluminum coating 1000nm blazed reflection grating.

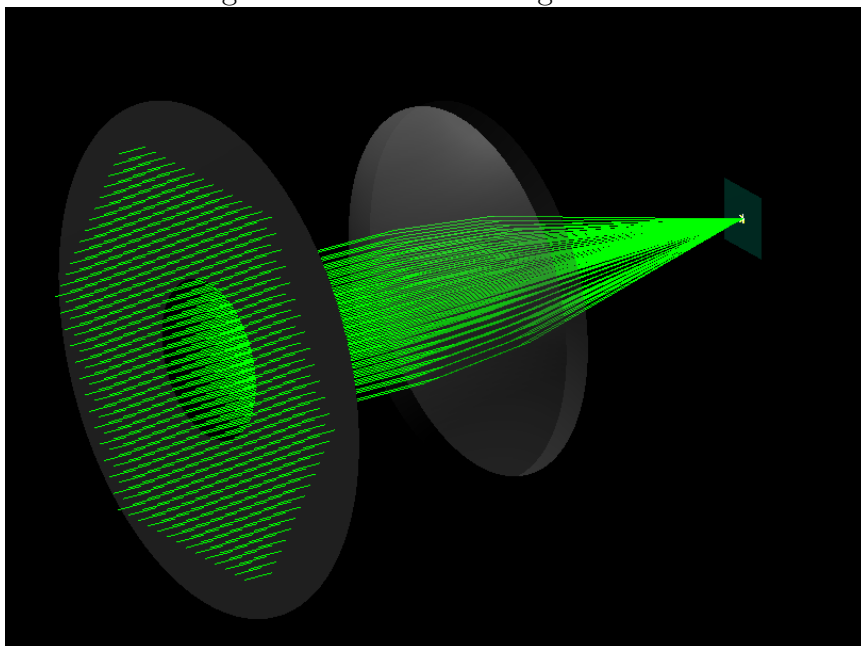
The grating equation governs the angle at which light of different frequencies will reflect. The first three orders of diffraction are plotted below for a 1200 groove per mm grating. The polar plot is degrees, ranging from 400nm to 1700nm.  $\rho$  represents the order of diffraction. An angle of incidence of 45 degrees will produce backreflection toward the fiber collimator, however an angle of incidence at 50 degrees will not.

Figure 37: Diffraction angle calculation



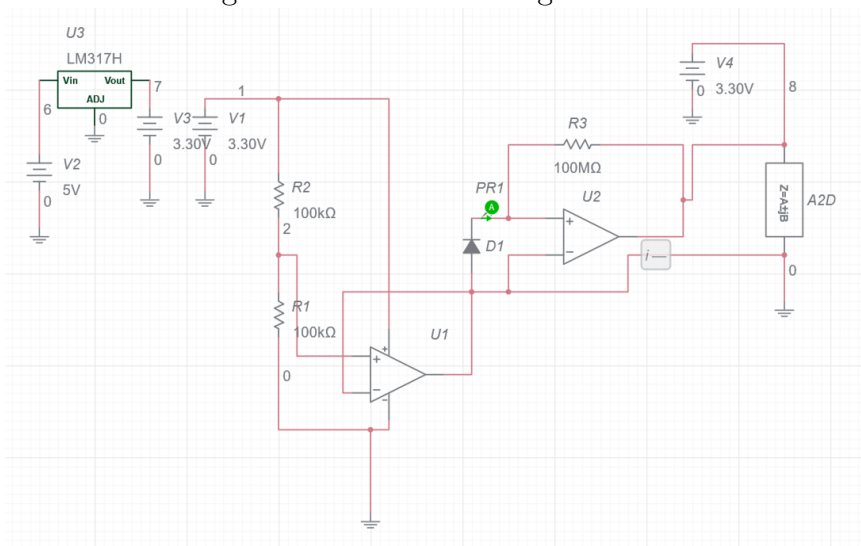
Focusing Lens

Figure 38: Photodiode Signal Filter



## Carriage driven Photodiodes

Figure 39: Photodiode Signal Filter



**Signal Filtering** The Photodiode works by converting a small portion of the incident light into electrical current across the face of the semiconductor. The diode will have a small current running through the circuit hooked up to an amplifier for boosting the signal to a detectable level. Then another op amp will cut out the electrical



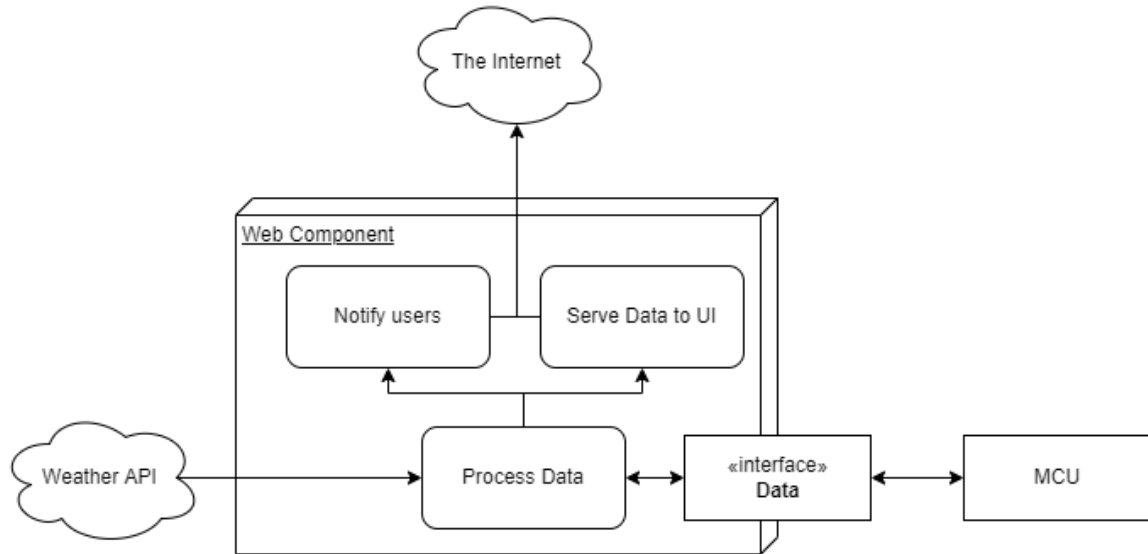
noise created by unwanted frequencies generated by the diode and electromagnetic interference.

## 5.4 Web Subsystem

The web subsystem will be split into several distinct parts to achieve several goals. First and foremost, the web will communicate over TCP with the MCU to get data, process the data and send commands back to the MCU. Secondly, there will need to be a database in order to log data and be able to support multiple of these garden beds in a scaled solution. Third, there needs to be a user interface that allows the user to set settings and read the data about their garden bed(s). Lastly, the web component of this project will communicate using HTTP requests with a weather service to get upcoming weather such as rain, sunlight, and freeze warnings in order to help the user with maintaining their plant bed.

The block diagram below shows a high level overview of flow of data between different sources. The microcontroller and web component share a two way interface to transmit controller data to the web and commands back to the MCU. Refer to the MCU subsection for more details on the commands. Part of this component will be a way to process the data received from the weather service and the MCU to serve it to the UI and thus the user.

Figure 40: Web component block diagram



Section 3.2.3 has information on the technologies in the stack while the following sections will go into more detail about how these choices will be used in order to achieve the goals outlined above. These choices are a React frontend as it is a technology that the team is familiar with and has good support and allows for an event-driven UI as would be useful to serve the polling data collection of the backend.

Express will be used as a middleware and routing layer for the user interface to the server backend. This choice was made again to keep the language in JavaScript and the vastness of support on the platform. Express also supports Socket.js which will be necessary for communicating over TCP with the microcontroller. MySQL will be used as a relational database as the data is relational. SQL databases also typically serve data-driven applications better over service-driven.

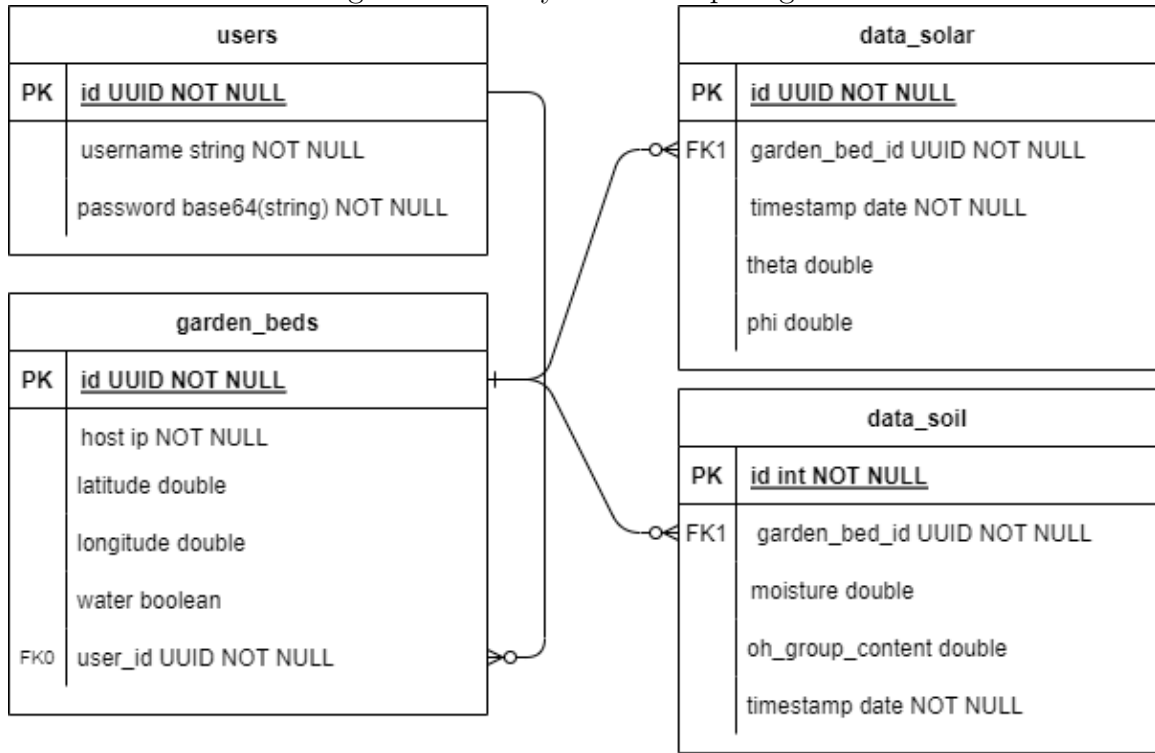
#### 5.4.1 Server Backend

As mentioned above, the backend will consist of two parts, a MySQL database as this is a free option and serves the purposes well especially for logging data, as well as an Express backend which will serve all the API endpoints for the user interface as well as a server to create TCP connections with the microcontroller.

**Communication** This section will cover how we plan on implementing communications with the microcontroller. As covered in 5.1 these two systems will communicate over TCP. The means of implementing this will be through sockets. This backend will serve as a server to make connections to. Through the use of the Socket.io library, it will be nearly superficial to create this connection and read raw data and process it. Once data is received will be stored in the database for computation.

**Database** The database will be relational. Relating a plant bed and its current state to the logs of data stored within the database as well. The way this will be implemented is there will be a table, `garden_bed` which will store a garden bed and the client socket information. From this point, the ID of the `garden_bed` will be used in a lookup in other tables such as `data_solar` or `data_soil` to be able to present this data to the user in the UI upon request. These data tables will have a timestamp of their creation to be able to lookup the most recent data and retrieve those items and also to create a log and graphs if we so chose. Figure 41 has some details on the type of data and where it can be found within the database given the design above.

Figure 41: Entity relationship diagram



Another option for the organization of the database is to create a table for each plant bed, which for this project would only be one, that holds all the data and logs regarding to that plant bed. This solution is slightly harder to implement. It is harder to implement because ORMs (Object-Relational Mapping) do not like the dynamic nature of tables. ORMs know the schema of a database table per the name, and when the database becomes highly dynamic in this architecture, the ORM typically requires more work to setup. For more detail on ORMs refer to the technology section of this document.

**Data Processing** All data will be processed on the EC2 instance within the server. Using some heuristic methods and research into peak conditions for different plant types, the server will be able to determine different actions to be made on the plant bed in order to promote growth.

**Accounts and Plant Bed Linkages** Each user account will be linked with potentially multiple plant beds through the database designed above. The backend will validate user logins to endpoints using JWTs in cookies in order to validate each user that logs in.

**API Endpoint Design** Representational State Transfer (REST) is a paradigm that indicates a certain API is platform agnostic thus allowing for any number of clients to call the API so long as they follow the standards that are implemented. REST is typically implemented through HTTP requests, this is the case for our project as well. HTTP comes fully packaged with various methods of calling different URIs (unique resource identifier). The best way to represent this is that a top-level domain refers to the server while appendages to the domain denote the resource and the verb. For example, `http://autogardenbed.com/garden`, is the location of the garden resource. From there we can use HTTP methods to instruct the server on what action to take. The most common operations are POST, GET, PUT, DELETE which corresponds to create, read, update, and delete. Our API will be no different.

The URIs will follow the convention of `.../[noun]/[verb]/{identifier}`. The “noun” refers to the type of object that the request will interact with. In our case this should be something like “garden” or “data”. Verbs will be used rarely but a situation that might arise is where the user wants to send a command to their garden bed. This will be accomplished through the verb portion of the URI. Table 21 gives a high level overview of the URIs, the associated methods and what they will accomplish.

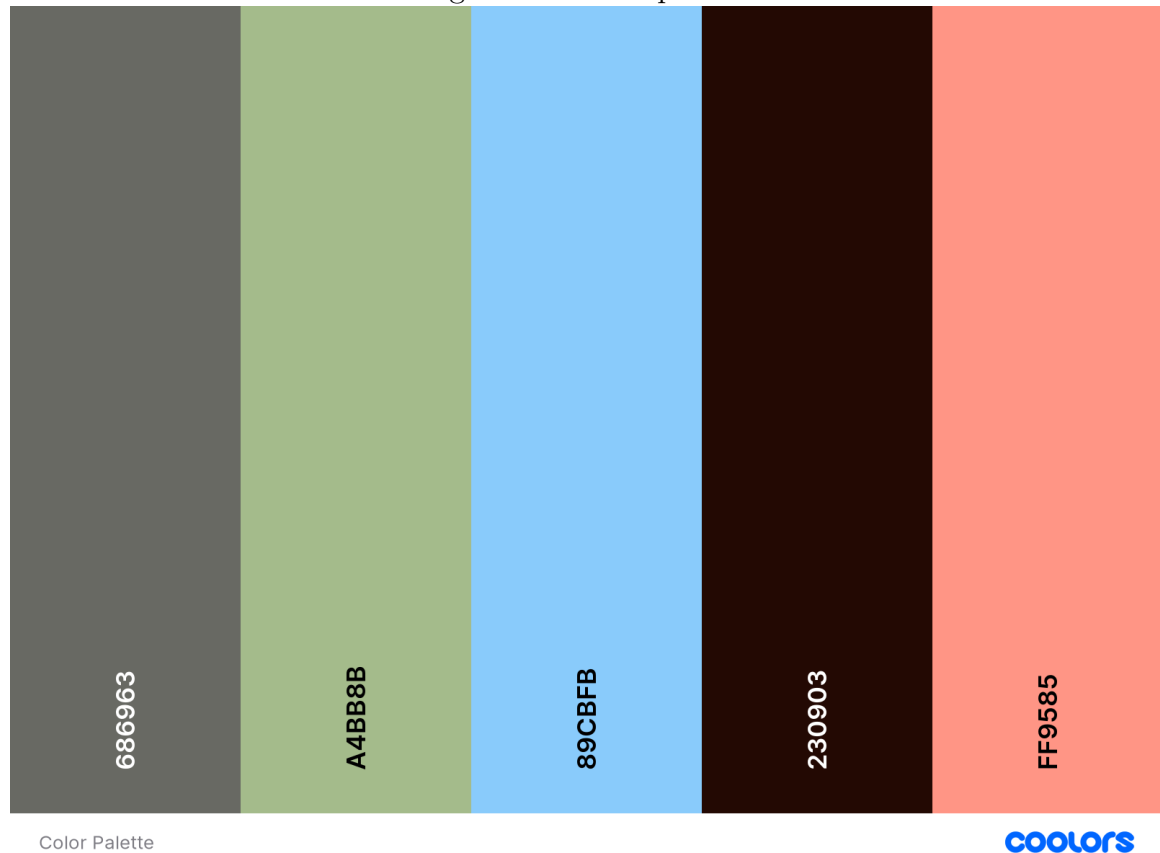
URI	HTTP Method	Function
.../user	POST	Create a new user
	GET	Get a user’s information such as a list of associated garden beds
	PUT	Update user information such as a password
	DELETE	Delete user account
.../user/auth	POST	Takes user credentials and validates against database; returns a JWT for validation in other endpoints
.../garden	POST	Creates a new garden bed
	GET	Get data on a garden bed, join garden bed info with data
	PUT	Update garden bed information, such as IP, location, etc.
	DELETE	Delete the garden bed from the database and ensure no records are orphaned
.../garden/{garden_id}/command	POST	Sends the command specified by the request object to the garden bed

Table 21: URI table

### 5.4.2 User Interface

The below hex values were selected using a color picking service that guarantees a palette that is useable for users who might be colorblind.

Figure 42: Color palette



For user experience, catering to colorblindness is important. From left to right the purpose of each color is: component color, emphasis, highlight, background color, error.

To design the user interface we will be using Figma. Figma offers some prototyping and features in order to quickly build a user interface. In conjunctions with React as a framework and CSS, putting everything together should be relatively trivial.

## 5.5 Subsystem Integration

The integration of all the subsystems is centered around the control subsystem. In the following sections, the team will discuss how these integrations will occur as the design process was centered around creating individual black boxes for each system to work against.

### 5.5.1 Sensing

The sensing subsystem has three key components for the control subsystem to interact with: the linear rail and carriage the spectrometer is mounted to, the laser generating photons for the spectrometer to receive, and the spectrometer itself.

**Linear Rail and Carriage** The spectrometer will be mounted to a carriage that may slide back and forth on the linear rail. The position of the carriage on the linear rail determines the wavelength of light sampled, and therefore the wavelength can be interpolated by the position of the stepper motor controlling the spectrometer carriage. It is a natural conclusion that the microcontroller will be able to direct the spectrometer to measure a specific wavelength by moving the stepper motor a certain number of predetermined steps. The microcontroller will sample the entire spectral range and compile this data to prepare to send to the web subsystem for processing.

(7)

**Laser** The laser mounted to the Carriage will be commanded to power on by the microcontroller whenever the microcontroller wishes to obtain a sample from the spectrometer. The laser itself draws a minimal amount of current and is compatible with the voltages regulated and output by the microcontroller, and will simply be connected to one of the microcontroller's GPIO pins.

**Spectrometer** The spectrometer itself will be stationary relative to the spectrometer's carriage. The spectrometer is a passive instrument that generates current depending on the flux density on the sensor. Due to the position of the spectrometer, this flux density is based on the strength of the specific wavelength emission. This current is connected to the sensing subsystem circuitry described in 5.3. This circuitry will proportionally convert the current to a voltage, and the output of the circuit will be connected to an analog-configured pin of the microcontroller, which will be taken as an input to the microcontroller's analog-to-digital converter. The ADC will convert the voltage to a value able to be measured in programming. The current (and by extension, voltage) level measured will be paired with the wavelength and transmitted by the MCU to the Amazon EC2 instance.

### 5.5.2 Power

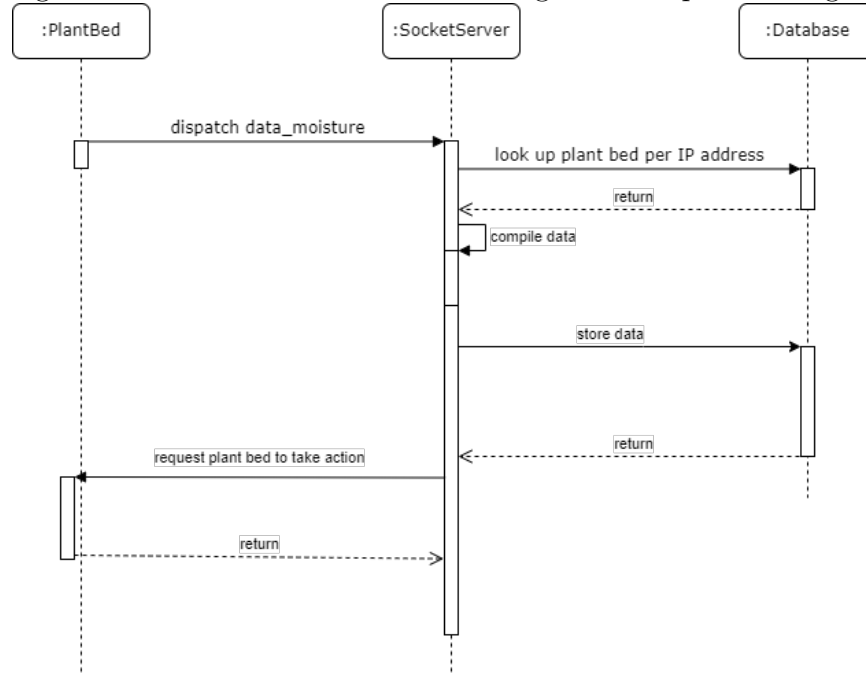
The charge controller will deliver voltage of the battery to the microcontroller via

### 5.5.3 Web

The web and control subsystems interact via two way WebSocket protocol (4.1.3), which will utilize TCP (4.1) as the transport layer protocol. Section 5.1 talks about the format of the raw texts in the packets. The web subsystem will be deployed to a

fixed domain so the memory of the control system can use a fixed host for the Socket server. The socket server will always start by processing the address information of the incoming socket, and grabbing previous data from the database accordingly. The socket server will process the data, storing it in the database for reporting on later. If necessary, the socket server will send a request to the plant bed to do some function. This is laid out in Figure 43.

Figure 43: Web-Plant Bed Socket Integration Sequence Diagram



## 6 Testing

Incremental unit testing will be used during the development process to ensure each subsystem fulfills design requirements as described in ???. Controller subsystem testing will be performed in subsection 6.1, power subsystem testing will be performed in subsection 6.2, sensing subsystem testing will be performed in subsection 6.3, and web subsystem testing will be performed in subsection 6.4. After subsystems have fulfilled unit testing requirements, integration testing will be performed with the criteria defined in subsection 6.5.

### 6.1 Controller Subsystem Testing

The MCU subsystem will undergo a littany of testing. The electrical characteristics, as well as hardware and software behavior will be tested to ensure a working product. Any use of "system" or "the system" in this section refers exclusively to, and the entirety of the MCU subsystem.

### 6.1.1 Electrical Characteristics

The MCU subsystem shall adhere to the following electrical characteristics related to power draw and voltage input in the following states. Unless noted, all voltages will be in direct current, and all conditions will be tested at room temperature (77 °F).

**General Requirements** The system shall be able to accept a voltage of -0.5 to 3.8 V into  $V_{BAT}$ , -0.5 to 4.3 V into any digital IO-configured pin, -0.5 to 2.1 V into any RF-configured pin, and -0.5 to 2.1 V into any analog-configured pin without damage. The system shall be able to accept a current of 0 to 620 mA into any pin without damage. The system shall be able to regulate voltage going into  $V_{BAT}$ , and provide  $3.3 \pm 0.3$  V on its 3V3 rail to any components in the sensing subsystem that may require it.

**Results** TBD.

**Operating Temperature** The system shall be able to operate nominally within the temperature range of 30 to 140 °F. Crystal oscillators shall fall within the thermal drift ratings provided in the MCU technical reference material.

**Results** TBD.

**Shutdown Power Draw** The system shall not draw more than 10  $\mu$ A in shut down mode. The system shall not consume more than 100  $\mu$ W in shut down mode.

**Results** TBD.

**Critical Power Draw** The system shall not draw more than 1 mA in critical power mode. The system shall not consume more than 5 mW in critical power mode.

**Results** TBD.

**Low Power Draw** The system shall not draw more than 100 mA while receiving or idling in low power mode, and shall not draw more than 250 mA while transmitting in low power mode. The system shall not consume more than 500 mW while receiving or idling in low power mode, and shall not consume more than 1.25 W while transmitting in low power mode. The system shall be able to sufficiently power any components in the sensing subsystem in low power mode.

**Results** TBD.



**Active Power Draw** The system shall not draw more than 150 mA while receiving or idling in active mode, and shall not draw more than 300 mA while transmitting in active mode. The system shall not consume more than 750 mW while receiving or idling in active mode, and shall not consume more than 1.50 W while transmitting in active mode. The system shall be able to sufficiently power any components in the sensing subsystem in active mode.

**Results** TBD.

**Startup Power Draw** The system shall not draw more than 750 mA within the first 20 seconds of powering on or starting up the system. The system shall not consume more than 4.00 W within the first 20 seconds of powering on or starting up the system.

**Results** TBD.

**Solar Panels and Battery System** The system shall be able to operate nominally solely powered off of the solar and battery subsystem detailed in subsection 5.2. The system shall not require any external power to operate in any modes (e.g. startup, active, etc.).

**Results** TBD.

#### **6.1.2 Hardware Behavior**

The MCU subsystem shall pass the following tests related to use of the MCU's on-board hardware, peripherals, and supported operations. The programming developed and test the below criteria will be intended to only perform testing on the below modules, and may or may not be present in the product software build.

**Upload Programming** The developers shall be able to successfully upload programming (software) that meets the requirements detailed in subsection 5.1. The developers shall be able to successfully upload programming that allows and/or performs unit testing (as detailed in this section, subsection 6.1). Programming uploaded to the MCU shall execute either upon powering the system, upon command, or when executing an ISR to start up the system.

**Results** TBD.

**I2C** The developers shall be able to initialize I2C. I2C shall function nominally if used for communication with any other modules in the overall garden bed system.

**Results** TBD.

**Onboard LEDs** The system shall be able to make use of its onboard LEDs for notifying the user or developers of system states according to the specification in subsection 5.1.

**Results** TBD.

**Watchdog Timer** The developers shall be able to initialize the watchdog timer. The watchdog timer shall behave according to the specification in subsection 5.1.

**Results** TBD.

**WiFi Module** The WiFi module shall conform to the 802.11b/g/n standards. The developers shall be able to configure the WiFi module to support 802.11b, 802.11g, or 802.11n in WiFi station mode. The developers shall be able to configure the WiFi module to support 802.11b, or 802.11g in WiFi Direct mode. The WiFi module shall allow the system to connect to either an open WLAN, or a WLAN secured by WPA2 (Personal or Enterprise). While connected, the WiFi module shall be capable of sending and receiving data at speeds of at least 10 kbps at least 80% of the time.

**Results** TBD.

**GPIO** The developers shall be able to configure the GPIO pins to the configurations available as described by technical reference documentation. The developers shall be able to set the state of the desired GPIO pins to the configurations available as described by technical reference documentation.

**Results** The developers are able to configure GPIO pins to the configurations available as described by technical reference documentation via the Texas Instruments [SYSCONFIG](#) system configuration tool.

**Interrupts/ISRs** The developers shall be able to configure the interrupts available on the MCU as described by technical reference documentation to trigger an ISR of their choice. ISRs shall run within 1 second of triggering the interrupt linked to the specific ISR. An empty ISR shall return the MCU to its previous state before the interrupt (i.e. clear any flags and return any registers to their previous state).

**Results** TBD.

**ADC** The developers shall be able to configure the ADC on the MCU to adhere to the parameters described in the technical reference documentation. The ADC shall be able to measure signals with a voltage of between 0 and 1.8 V. The ADC shall be able to accurately measure a voltage level down to a resolution of  $\leq 1$  mV. The ADC shall be able to measure at a sampling rate of  $\geq 50$  kilosamples per second per channel. The ADC shall support conversion on up to 4 channels. The ADC shall support the ability to timestamp samples with the clock mentioned by the technical reference documentation.

**Results** TBD.

**Reset Button** The reset button shall be able to "power cycle" the system. The reset button shall not erase any programming uploaded to the MCU. The reset button shall not keep the system from starting for more than 1 second after the button has been released.

**Results** The reset button "power cycles" the system, does not erase any programming uploaded to the MCU, and does not keep the system from starting for more than 1 second after the button has been released.

**Startup** The system shall not take longer than 10 seconds to begin its programming upon sufficient power delivery to the MCU as described by the by the technical reference documentation. The system shall not perform the programmed startup sequence unless interrupted to do so, or if given sufficient power as described by the by the technical reference documentation after being shut down.

**Results** The system does not take longer than 10 seconds to begin its programming upon sufficient power delivery to the MCU. The system does not perform the programmed startup sequence unless the parameters described above are met.

**Processor** The central processor of the MCU shall maintain at least a 20 MHz clock speed while in active mode.

**Results** The central processor of the MCU is able to maintain an 80 MHz clock speed while in active mode.

**Memory** The programming uploaded to the MCU shall not take up more than 64 KB of space in memory.

**Results** TBD.

### 6.1.3 Software Behavior

The MCU subsystem shall pass the following tests related to use of the MCU to support application of the product. The programming developed and test the below criteria is intended to be the product software build.

**Startup** The microcontroller shall behave according to the specifications on startup in subsection 5.1 during its startup sequence.

**Results** TBD.

**Shutdown** The microcontroller shall behave according to the specifications on shutdown in subsection 5.1 during its shutdown sequence.

**Results** TBD.

**Reset** The microcontroller shall behave according to the specifications on reset in subsection 5.1 during its reset sequence.

**Results** TBD.

**Networking** The system shall be able to connect to the user's WLAN if it meets the following standards:

- A 2.4 GHz-based wireless network
- Contains no security or is secured with WPA2 (Personal or Enterprise)
- $\geq -70$  dBm signal strength
- Adheres to the standards of 802.11b, g, or n
- DHCP or statically-assigned IPv4 addressing
- Uses NAT and does not expose system to WAN
- Provides DNS to LAN devices
- Able to resolve and successfully connect to 8.8.8.8 and google.com within 5 seconds

**Results** TBD.

**AWS** The microcontroller shall behave according to the specifications on connecting to and communicating with AWS in subsection 5.1.

**Results** TBD.

**Sockets** The microcontroller shall behave according to the specifications on sockets in subsection 5.1.

**Results** TBD.

**Solar and Battery** The system shall monitor and update the solar array current and voltage output values at least every 10 seconds. The system shall behave according to the specifications on power modes and battery percentages in subsection 5.1.

**Results** TBD.

**Sensing** The system shall monitor and perform analog-to-digital conversion on the spectroscopic sensor values at least every 10 seconds while not in shutdown or critical power mode.

**Results** TBD.

## 6.2 Power Subsystem Testing

The Power subsystem is an important part in making sure that the whole system is fully operational. With various components such as the microcontroller, sensors, mechanical components and more, we need to make sure that each component is operating properly. This is a confirmation that the whole system is operating but also a test to see which parts work best with each other and what works best under given situations. One thing that will be tested is the voltage regulators to see which of the selected voltage regulators is most efficient and regulate power the best see fit.

### 6.2.1 Voltage Regulator

Providing power is a very important part to operating a system, but providing the right amount of power is key to make sure all the components are operating properly. As said in section 3 in the Power Subsystem, the LM317 (linear) and the LM2576 (switching) were selected to be tested and compared to see which would be the most efficient in regulating voltage. In this test, a simple circuit will be built with a battery, capacitors and the voltage regulator. The output voltage will be measured and displayed on an oscilloscope to verify which of the voltage regulators were able to regulate voltage more efficiently and successfully.

## 6.3 Sensing Subsystem Testing

There are three components in the Sensing subsystem that will require testing on arrival.

- The Diffraction Grating
- The Optical Sensors
- The Linear Actuator

The diffraction grating will be tested for efficiency and for line spacing. This will also determine if the blaze angle is correctly labeled.

The photodiodes will be connected to circuitry and placed under beams to validate their responsivity curves and dark current.

The linear actuator will be tested to ensure it responds properly to the specified voltage and current inputs and to ensure that it runs at the proper speed. Then the actuator will be tested to determine if the incremental steps are of the proper length.

## 6.4 Web Testing

There are three components to test in the web system. The first is the API endpoints and backend functionality. The second is the integration of the user interface and the API. And lastly, the functionality of the socket server must be tested as well.

### 6.4.1 API Testing

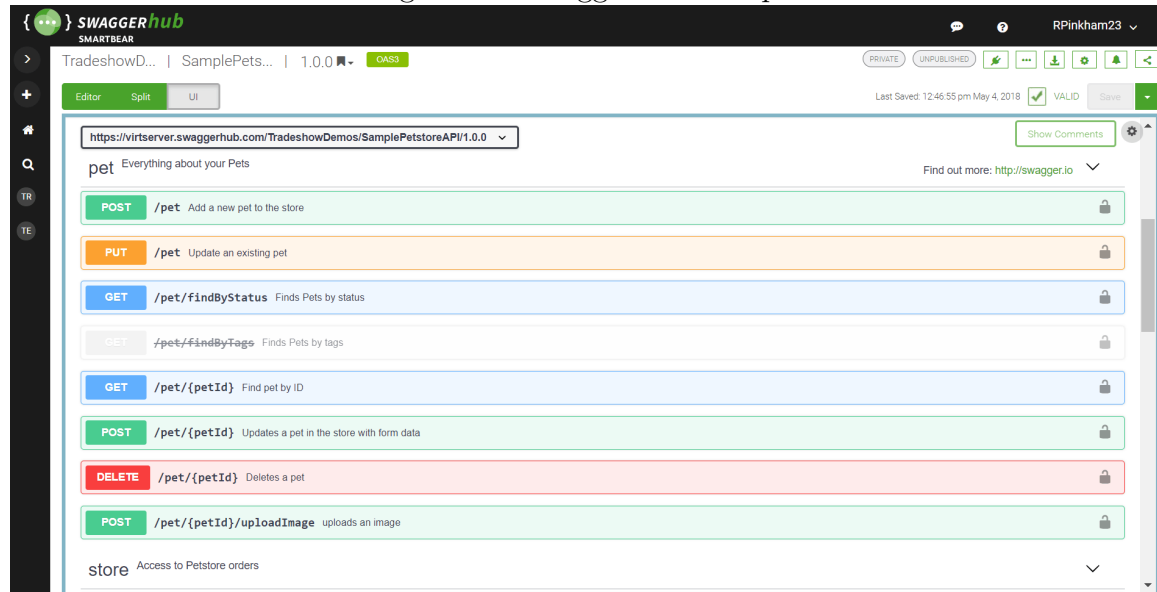
There will be two key ways that the API testing will be conducted: unit tests and integration tests. In a web context these are slightly different than in a strict engineering context. Generally speaking, a unit test of a an API or backend service checks that business logic is working as expected. For example, in our project there will be some filter to grab the correct information from the database based on the plant bed. In a unit test, we might mock the database with dummy data and ensure that this is occurring properly.

On the other hand, the integration test of the API will be done in two ways which will also serve to self-document the API. Using a mock client, we can program requests for certain conditions such as certain failure or success. In this instance, the request is actually routed through all the middleware, connects to the database and performs the function. We can validate the result by looking at the database and “asserting” all these values are correct.

The self-documenting test will be done using Swagger and OpenAPI 3.0. Swagger is a tool for building UI elements that can build and run requests on the web server

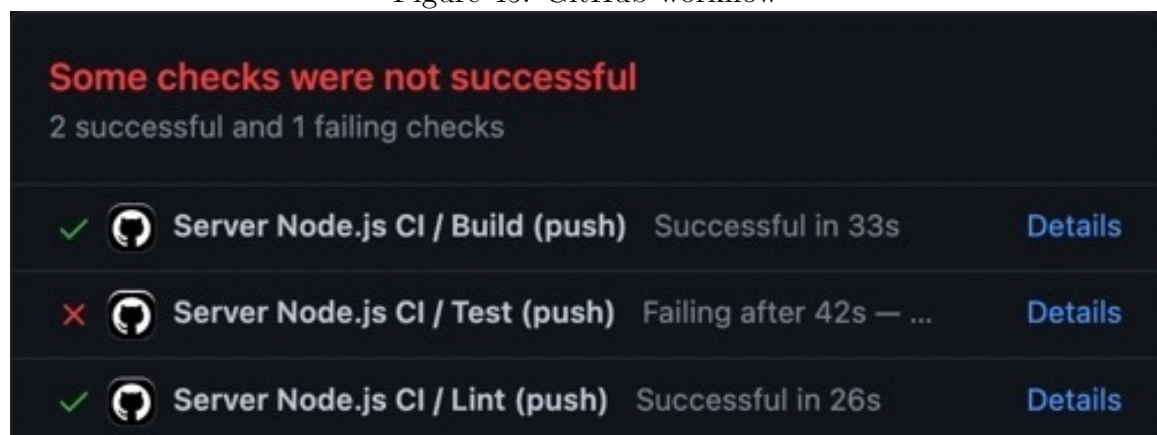
and the response elements can be validated against data types. See Figure 44 for a visual representation as to what this might look like.

Figure 44: Swagger UI example



**Test Plan** The unit tests and integration tests will be ran on each push of code changes to GitHub and through the use of GitHub actions, passing tests can be validated and added to the branch protection rules. Branch protection rules prevent broken code from becoming a part of the code base. These rules are integral to our test plan. See Figure 45 to see how passing/failing tests would appear.

Figure 45: GitHub workflow



The key to integrating this quality of life feature is designing good tests. The API in general has four functions: create, read, update, delete. The unit tests will

be configured to ensure error handling and data composition are done properly. For example, when making a request about a particular plant bed, if the ID does not exist, we want the error to be verbose but the request to fail. Similarly, we want to ensure that the logic for building data tables from the database occurs properly. The integration tests will have one well-formed request and one malformed request in order to test the error handling and full integration of correct data.

### 6.4.2 User Interface Testing

There are two key ways to be able to test the user interface: using prototypes and integrating with the backend. Using React the expectation is that the user interface will be reactive. Thus, we use prototypes which are components built on hardcoded data that reacts to user input, thus we can test the viability of components. For example, we want to test that a graph component when hovered shows the data point in a floating box. We would build the UI with static data and ensure this result in the UI. Secondly, we need to test that the integration with the backend is working properly. This occurs by running an “end-to-end” test. Essentially, this is a final test. The plan is to give the testable product to consumers and let them have their way with the UI to discover bugs.

React comes packaged with a testing library that allows for quick unit testing of components in a similar way to the unit tests mentioned in the API section (6.4.1). Implementing this will automate the unit tests as opposed to manually building the entire program and checking manually each component. We will be using [this documentation](#) to build out the automatic test suite.

### 6.4.3 Socket Testing

Unit testing sockets will be done in a similar way to unit testing the API endpoints however the unit test will create a mock socket client and the two features will be tested separately: sending and receiving.

**Sending Packets to Client** To test that the client is receiving the packets properly in isolation, the mock client will open a connection to the server and the unit test will call the send packet function. The client will then expect certain values in a certain character set and this will all be validated by the assertion.

**Receiving Packets from Client** The mock client will send a packet to the socket server with static data that is formatted exactly as it would come from the MCU. From there, the unit test will assert that the received packet has all the required metadata, uses the correct character set and is formatted properly.



## 6.5 Integration Testing

The microcontroller is the key linkage between all the other subsystem's. Integration testing between these subsystems will be done in isolation before assembling the entire project. The integrations will be done and tested in the following order before a final full system test is completed.

### 6.5.1 Web Integration

Due to the use of Docker, the web integration test can be done at all points in development. There are two ways this can be done, downloading and running the image, or building the image from the source code. The processes for each are as follows:

#### Docker Image

1. Web developer uses docker to build the image
2. Microcontroller developer downloads the built image from docker hub
3. Use Docker Desktop to run image
4. Test

#### Build Docker Image from Source

1. Web developer pushes changes to Github
2. Microcontroller developer pulls changes from Github
3. Developer builds image from new source code
4. Run the image and test

**Testing Areas** There are two key areas that need to be test in this integration. First, that the microcontroller and web are communicating appropriately following the standards and design criteria. Second, that the microcontroller is able to read and translate commands sent by the server and responds accordingly.

Through the use of UART on the microcontroller and verbose logging on the server application we can validate that the packets that are exchanged between the two systems are received and translated properly. Part of this test will be forcing packets to be sent, so the microcontroller will have commands in UART in order to force send packets to the web server so these can be verified. Similarly, the web component will need a way to force send packets to the microcontroller. These packets can be verified by looking at the metadata associated, the charset, length, and payload.

Once the packets have been verified we need to test that the microcontroller is responding to the packets it is being sent. This can be done in two ways. First, using UART we can print a response to the packet to show that the logic is working. The second part of this testing is ensuring digital logic on the pins to the control mechanisms are also working appropriately.

### **6.5.2 Sensor Integration**

The sensor integration testing will be relatively simple. Through the use of UART, we will be able to print the raw sensor data on a schedule using a timer.

### **6.5.3 Power Integration**

Through the use of multimeters we will be able to test the power into the MCU and be able to do a full integration test.

### **6.5.4 Full Integration**

The full integration test will bring together all parts of the system and test them in conjunction in order to see if all the systems are working.

## **7 Administrative Content**

### **7.1 Milestones**

From a project management perspective, the team will be utilizing Agile to get quick turn arounds on small deliverables. Choosing to use agile, the team will also be using Jira to track progress and add reports throughout the process. A high-level of the goals by semester can be found below.

#### **7.1.1 Fall**

- Select components for each subsystem
  - Document selection reasoning
  - Order to ensure on-time delivery
- Model physical bed
- Build physical bed
- Understand how subsystems will integrate:
  - Communication protocols (REST, I2C, SPI, DSP, etc)
  - Power requirements

- UI for Web subsystem

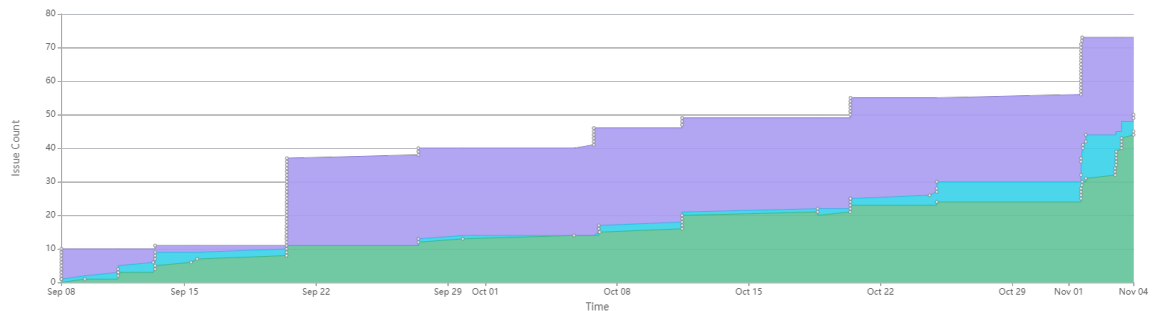
### 7.1.2 Spring

- Test subsystems in isolation
- Start integrating subsystems
- Control scheme for moving solar panels with sun and to provide shade
- Web API complete
- MCU coding complete
- Stretch goals

## 7.2 Progress

### 7.2.1 Senior Design I

Figure 46: Cumulative Flow Diagram from Jira



In Figure 46: purple designates tasks that are marked unfinished in the the backlog and current sprint, blue represents in progress tasks, and green represents finished tasks.

Throughout Senior Design I we have been gathering research and have started laying out the design of our garden bed and have completed the majority of our part selection. The Figure in 46 may be a little misleading at this point because we have not refined our backlog to fully encapsulate meaningful tasks instead breaking it down into larger subsystem requirement-esque tasks.

## 7.3 Budget

Table 22: Breakdown of budget by subsystem

Subsystem	Estimated Cost	Comment
MCU	\$60	The MCU, wiring harness
Power	\$200	Solar panels, batteries, control system
Sensing	\$200	Components for sensing, optical sensors
Web	\$30	Web service pricing
Non-Subsystem	\$100	The plant bed, soil, water, fittings, etc
Total	\$590	