Auto Garden Bed - Group 9

Team members:

Nicholas Chitty
Brendan College
Scott Peirce
Justin Pham-Trinh

UCF Senior Design Fall 2022 - Spring 2023

Contents

List of Figures

List of Tables

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, temperatue, 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 adversively (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 haves”. 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

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 "Garduino" 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

The system is divided into four subsystems that reflect each of the team member’s specialties. The Goals Table lists the system goals in three categories: core goals, stretch goals, and advance goals. Subsystems are grouped, except when they are split up by goal category. In addition, each goal corresponds to a tagline, for reference in the Objectives Table below.

Tagline	Core goals	Area of Focus
Size	The entire system will be the size of a small garden	General
Power supply	The power subsystem will collect its own power	Power
Power supply	The power subsystem will store power	Power
Power supply	The power subsystem will be able to connect to an external power supply as a backup	Power
Motor controls	The microcontroller will regulate the power generation system	MCU, Power
Motor controls	The microcontroller will control the water regulation system	MCU, Power
Motor controls	The microcontroller will control the sensing subsystem	MCU, Sensing
Signal Analysis	The microcontroller will read an electrical signal from the sensing subsystem	MCU, Sensing
Signal Analysis	The microcontroller will calculate variables based on the signals from the sensor.	MCU, Sensing
Cost	The sensing subsystem will be cheap to produce	Sensing
Wireless Comm/ Networking	The microcontroller will connect to the internet	MCU, Web
Wireless Comm/ Networking	The microcontroller will prompt messages to the user through the web interface	MCU, Web
Wireless Comm/ Networking	The microcontroller will use internet weather projections to determine rainfall and temperature	MCU, Web

Table 1: Core Goals

Stretch goals		
Motor controls	The microcontroller will control solar panel position and balance system power with plant need for sunlight, and surrounding temperature	MCU, Power
Motor controls	The sensing subsystem will prompt dispensing of fertilizer Sensing,	Power
Motor controls	The sensing subsystem will automatically level the soil and position the sensor for data collection	Sensing, Power
Motor controls	The sensing subsystem will collect readings from several positions in the garden bed	Sensing, Power
Wireless Comm/ Networking	The web interface will use a secure communications protocol	Web
Wireless Comm/ Networking	The web interface will allow the user to select a plant and identify the best system parameters for growth	Web

Table 2: Stretch Goals

Advance Goals		
Wireless Comm/ Networking	The web interface will allow the user to control the system parameters based on the selected plant	Web, MCU, Power

Table 3: Advance Goals

2.3 Objectives

Refer Objectives back to goals using their tagline.

Tagline / Goal	Objective Area of Focus	
Size	The system will be small, about a cubic meter	General
Power supply	The power subsystem will use solar panels to collect power	Power
Power supply	The power subsystem will use a battery to store enough power for at least 24 hours of operation	Power
Power supply	The power subsystem will be AC wall plug compatible	Power
Power supply	The power subsystem will have current leakage protection	Power
Motor Controls	The MCU will control the position of the solar panels	Power, MCU
Motor controls	The MCU will control water dispensing to the garden bed using a solenoid	Power, MCU
Signal Analysis	The sensing subsystem will be a spectrometer that uses diffraction to separate frequencies spatially	Sensing
Signal Analysis	The sensing subsystem will use photodiodes to sensitive the spectral regime corresponding to soil moisture, acidity, and nutrient content	Sensing
Signal Analysis	The sensing subsystem will use a stepper motor or servo to scan the beam across the sensors	MCU, Sensing
Signal Analysis	The sensing subsystem will increase the SNR using LEDs to illuminate the soil	Sensing
Signal Analysis	The sensing subsystem will have a fiber collimator to contain the beam	Sensing
Cost	The cost of the sensing subsystem will be less than $1000	Sensing
Wireless Comm/Networking	The MCU will make HTTP Requests to connect to the internet	MCU, Web
Wireless Comm/Networking	The MCU will communicate through a secure protocol such as TLSv1.2 or later	Web

Table 4: Objectives

2.3.1 Requirements Specifications

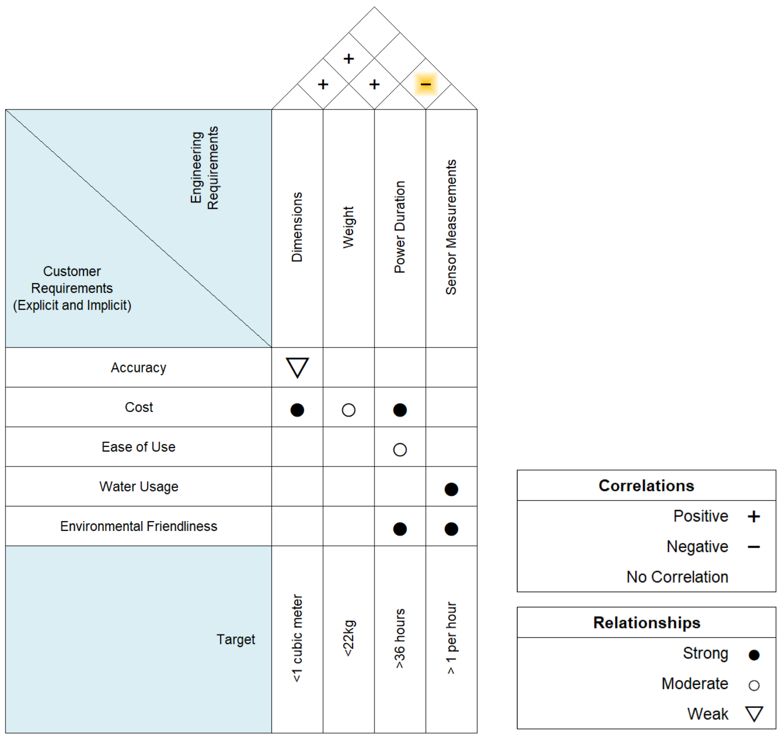
* "The system" refers to the plant bed and its subsystems (control, sensing, power and web)

- The system shall take up no more than a meter cubed of volume
- The system shall use solar energy and battery power with an AC source backup
- The power system shall have an overcharge protection
- The power system shall have current leakage protection
- The system shall be able to control the water supply
- The microcontroller shall be capable of internet communication via HTTP requests
- The microcontroller shall be capable of digital signal processing
- The sensing subsystem shall have a spectral resolution of less than 10nm
- The sensing subsystem shall have a range from 400 to 1700nm
- The system shall be able to convert analog signals to digital signals for processing
- The system shall be weatherproof
- The system shall weigh no more than 40 pounds empty of soil
- The system shall be of physical construction pursuant to any governing construction standards

2.4 Marketing Requirements and Engineering Requirements

Find our House of Quality figure below:

Figure 1: House of Quality

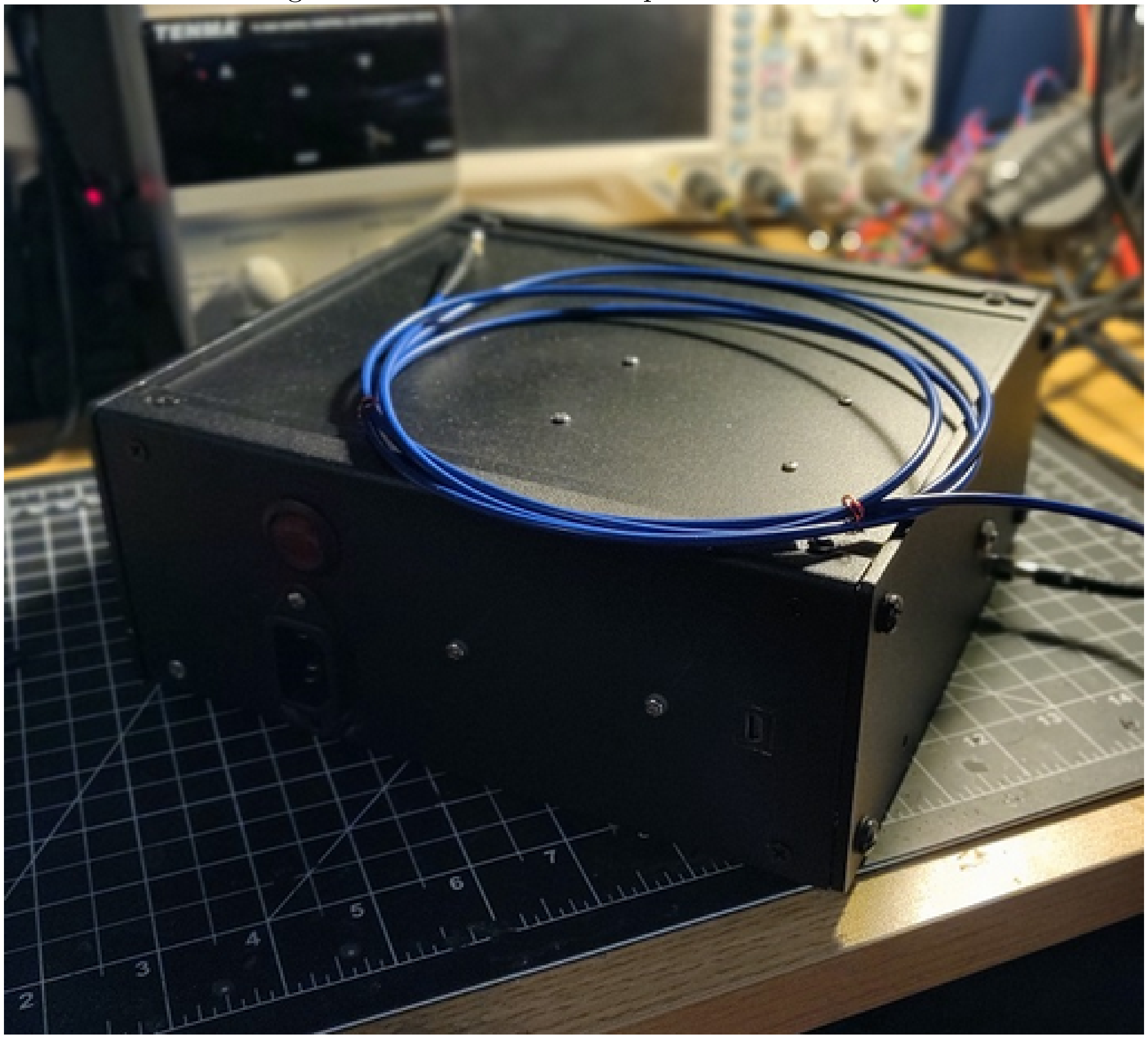


3 Research

3.1 Previous and Related Works

3.1.1 DIY near-IR spectrometer

Figure 2: NIR Photodiode Spectral Sensitivity

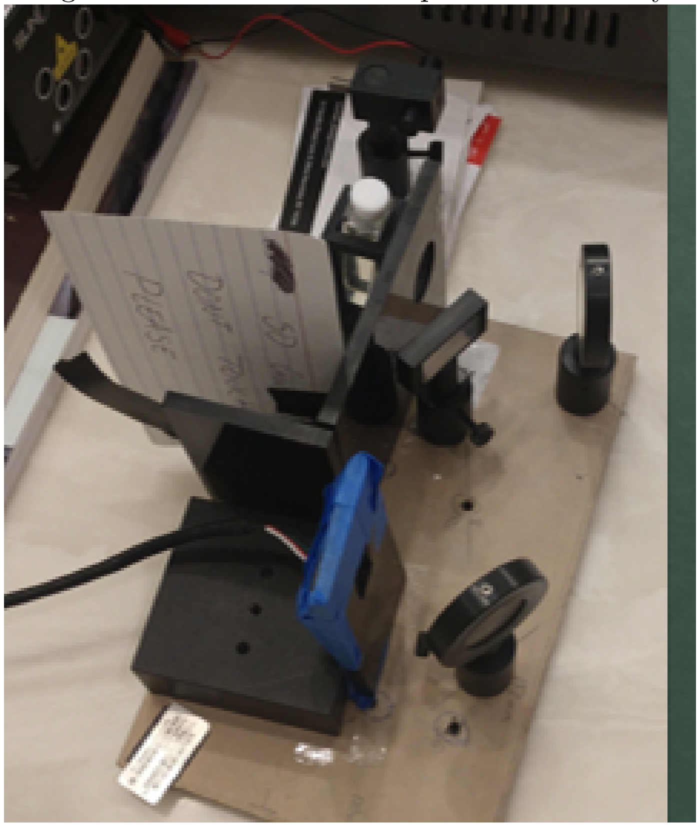


Yuan Cao is a Ph.D. student at MIT with a blog where he posts his unofficial projects. He published a project titled “A $500 DIY near-IR spectrometer that would sell for $10,000.” In it, he describes his idea, designs and results for a low cost infrared spectrometer. His design features surprisingly 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.

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 Flourescence Spectrometer

Figure 3: NIR Photodiode Spectral Sensitivity



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.

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

Figure 4: NIR Photodiode Spectral Sensitivity

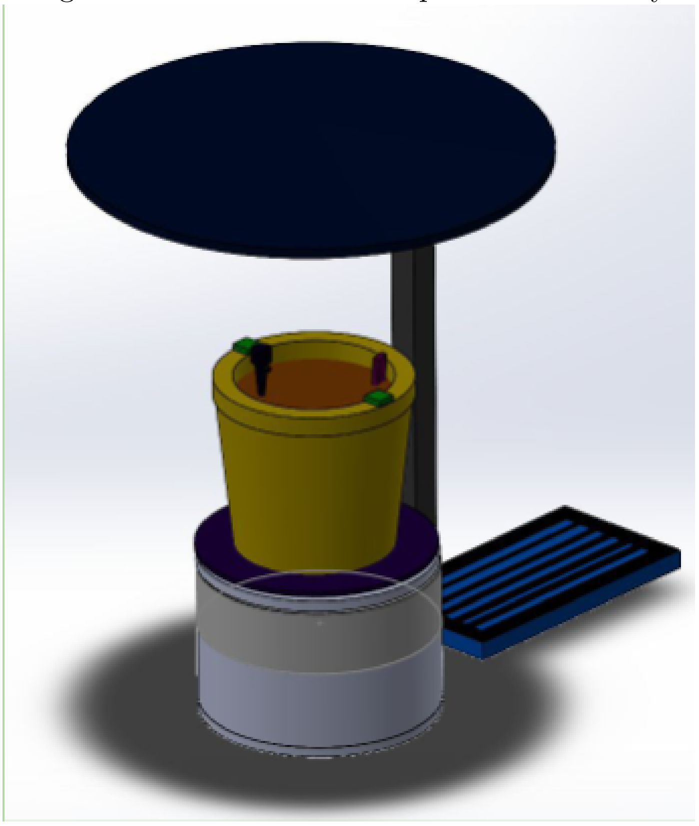
The Smart Garden Controller was a UCF senior design project with very similar motivations to the Auto Garden Bed. The goal was to create a system that reduced user labor by automating the irrigation schedule of garden areas and reducing waste by sensing moisture levels. The system was also designed to 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.

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

Figure 5: NIR Photodiode Spectral Sensitivity

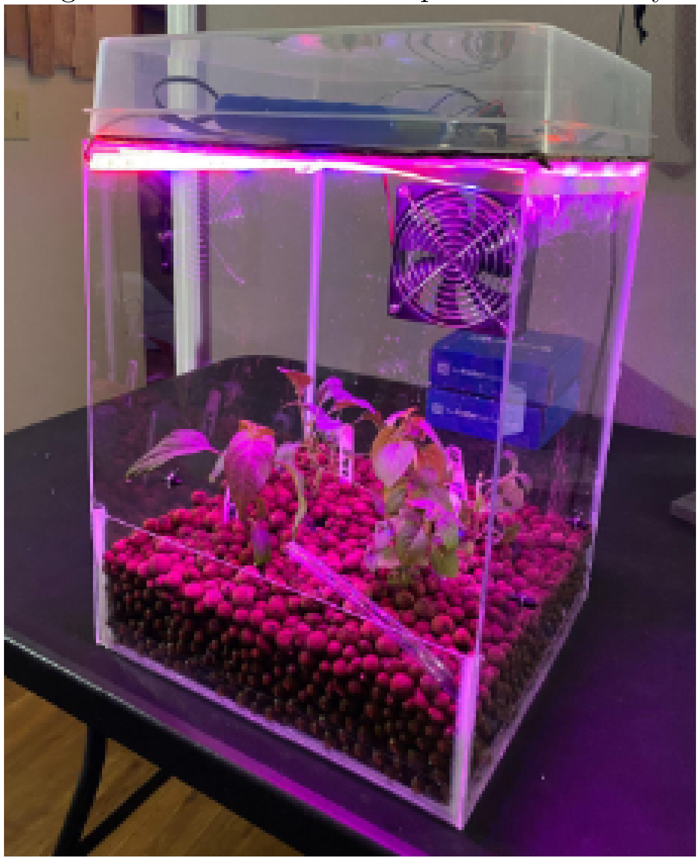


This is another home garden system with a different target solution. The goal is to facilitate the growth of indoor plants and ensure maximum plant health by rotating the plant base so that the forces driving it to grow sideways cancel out in every direction. Like previous projects, the system is integrated 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.

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

Figure 6: NIR Photodiode Spectral Sensitivity



Stem 'n' Leaf[1] 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

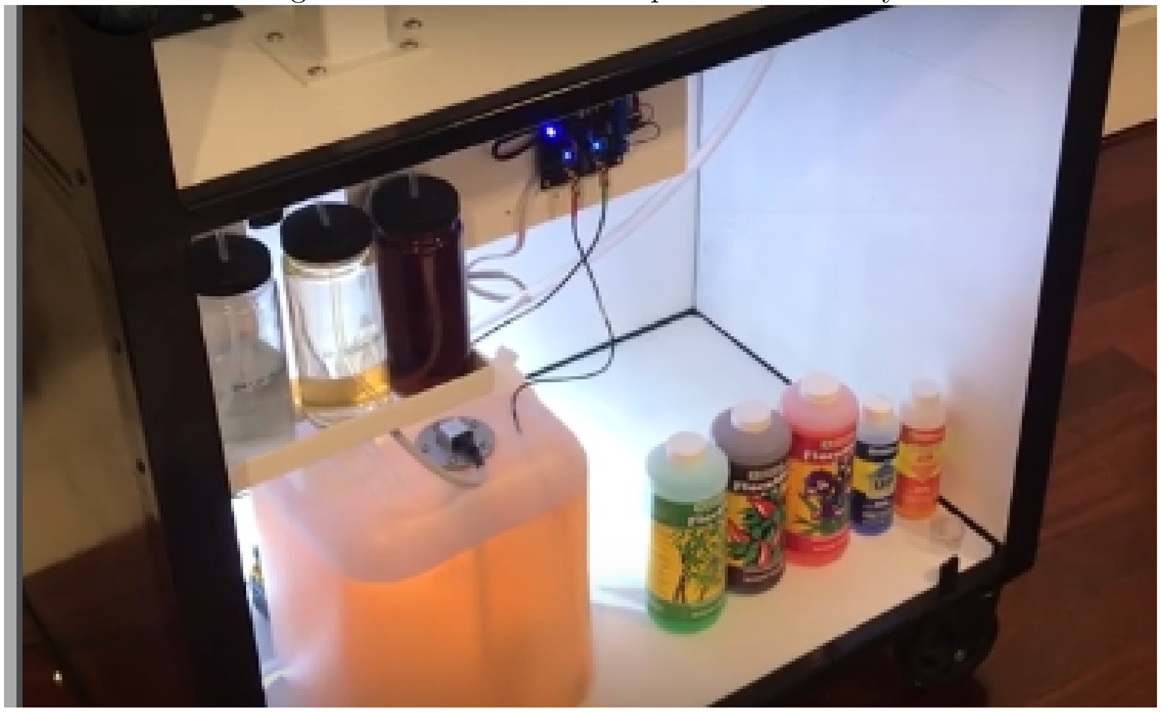
[1]https://www.ece.ucf.edu/seniordesign/sp2021su2021/g07/

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.

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

Figure 7: NIR Photodiode Spectral Sensitivity



Green Steel Garden[2] 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.

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

[2]https://www.ece.ucf.edu/seniordesign/su2021fa2021/g11

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

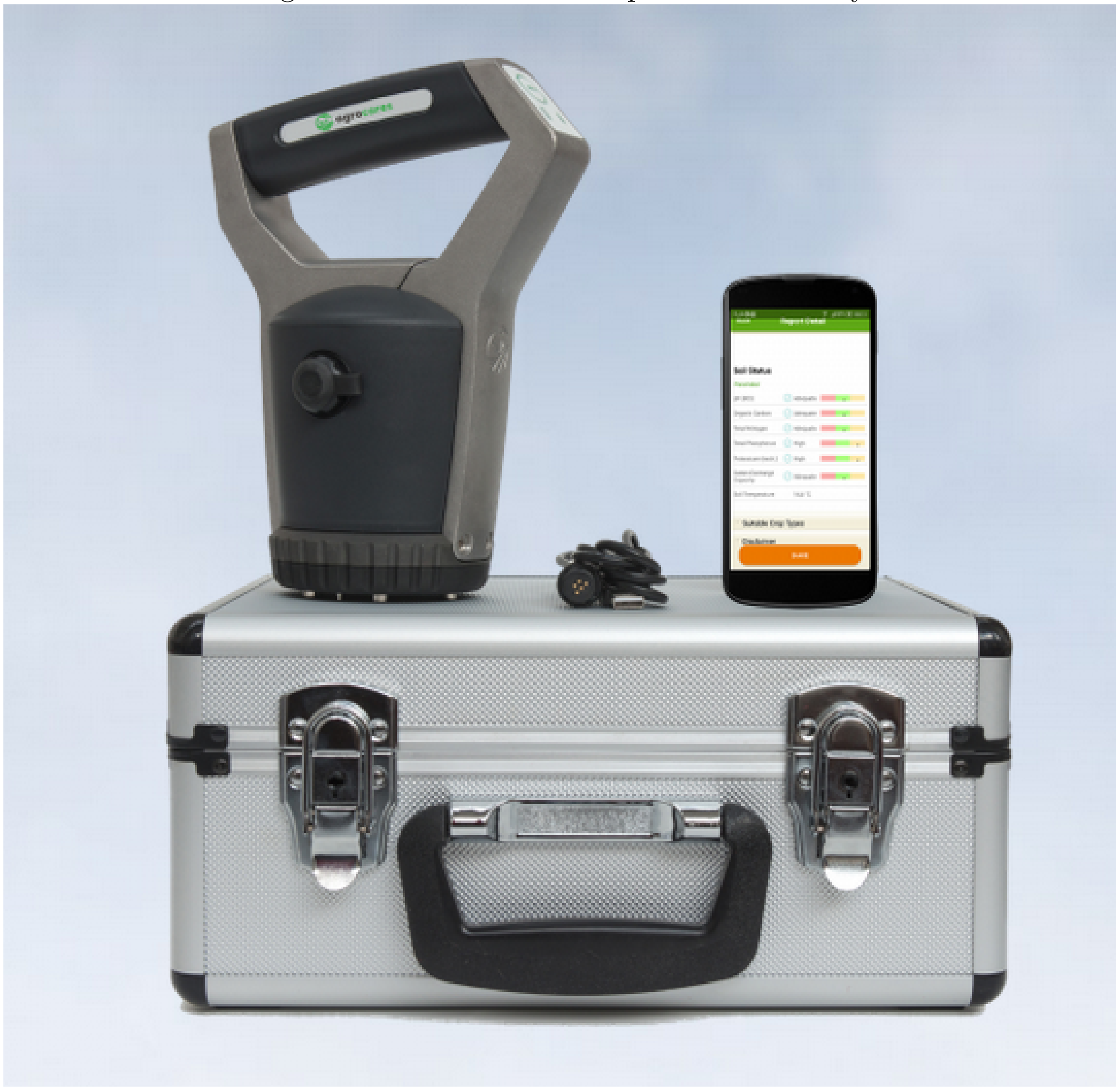
3.2.1 Ocean insight: Ocean ST NIR Microspectrometer

Figure 8: NIR Photodiode Spectral Sensitivity

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.

3.2.2 AgroCares Nutrient Soil Scanner

Figure 9: NIR Photodiode Spectral Sensitivity

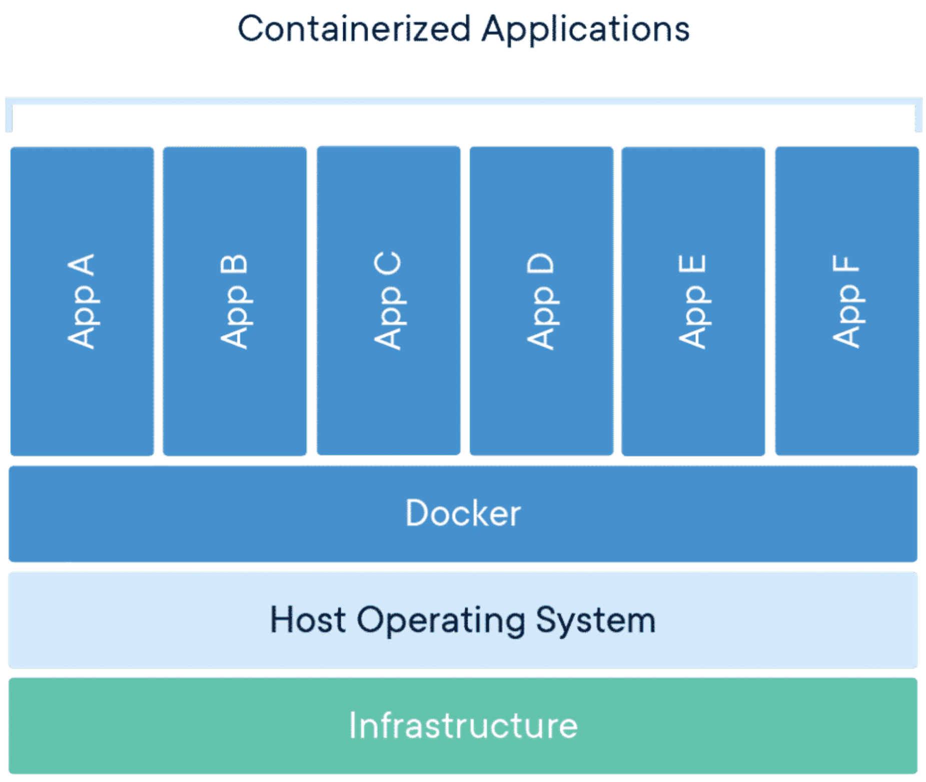


AgroCares offers a Near Infrared Spectrometer specifically designed for Proximity Soil Sensing. Its spectral range is from 1300 to 2500nm and it uses Micro Electrical Mechanical Systems or MEMS to capture EM Waves reflecting off the soil. The real value of the product is in its wireless communications 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 10: Docker architecture



A developer can program applications such as those demonstrated in figure 10, 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`.

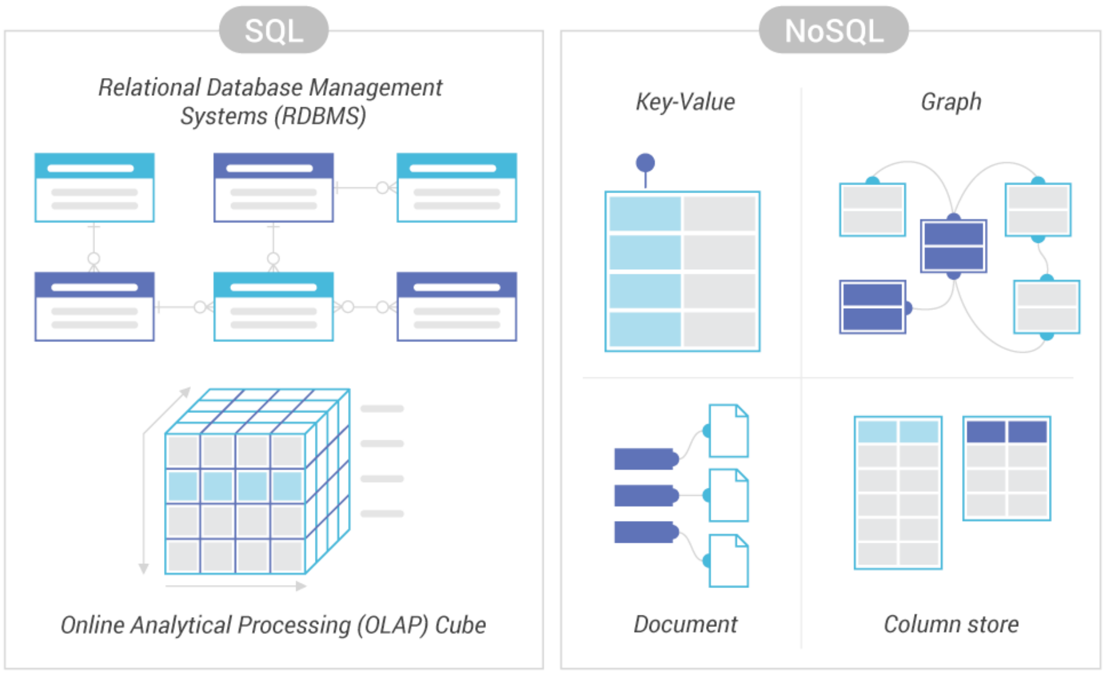
The `java.net` libraries are essentially a port of the network protocols that were made in the C language for creating communications via packets. By deploying using Java Spring Boot, it is possible to create two services packaged into one, the socket server and the web server. Java's memory management 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 11: Difference between SQL and NoSQL


SQL
Relational Database Management Systems (RDBMS)
Online Analytical Processing (OLAP) Cube
NoSQL
Key-Value
Graph
Document
Column store

SQL vs NoSQL In Figure 11 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.

3.3 Part Selection

3.3.1 Controller Subsystem

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

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

These features would be ”nice to have” on any MCU selected, but are not required:

- Digital-to-analog converter (DAC)
- 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)

The selections, listed in Table 5 and not in any particular order, match the above criteria and are being considered for selection.

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

Table 5: MCU option breakdown

Model	**LAUNCH-XL-CC26X2-R1**	**LAUNCH-CC3220-MODASF**	**Pico W**	**Nano 33 BLE**	**B-L4S5I-IOT01A**
Manu-facturer	Texas Instruments	Texas Instruments	Raspberry Pi	Arduino	STMicro-electronics
Micro-controller	CC2652R	CC3220-MODASF	RP2040	nRF52840	STM32-L4S5VIT6
Processor	1x ARM Cortex-M4F	1x ARM Cortex-M4	2x ARM Cortex-M0+	1x ARM Cortex-M4	1x ARM Cortex-M4
Maximum Speed (MHz)	48	80	133	64	120
Memory (KB)	256 ROM, 352 flash, 100 SRAM	1024 flash, 256 RAM	16 ROM, 264 SRAM	1024 flash, 256 SRAM	2048 flash, 640 RAM
Wireless capability	BLE5.2, Zigbee, Thread	802.11b/g/n	802.11n	BLE5.3, Zigbee, Thread, Matter	BT4.1, 802.11b/g/n, NFC
Serial capability	UART, I2C, I2S, SPI	UART, I2C, SPI	UART, I2C, SPI, USB1.1	UART, I2C, I2S, SPI, USB2.0	UART, I2C, SPI, USB2.0
Price ($)	40, maybe free	60, maybe free	6	28	53
ADC	8-channel, 12-bit	4-channel, 12-bit	4-channel, 12-bit	8-channel, 12-bit	16-channel, 12-bit
Clock capability	Timer, RTC	Timer, RTC, WDT	Timer, RTC, WDT	Timer, RTC, WDT	Timer, RTC, WDT
GPIO (pins)	31	29	30	13	16
PWM (channels)	Supported	Supported	16	4	6
Required voltage (V)	1.8 – 3.8	2.3 – 3.6	1.8 – 3.3	4.5 – 21	4.75 – 5.25

Use of an external Wifi module is discouraged due to the following:

- 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.

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.

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 what kind of battery and how many is needed, how long we want the system to run, optional secondary power source, and optional battery bank must be put into consideration as well. As follows, we begin to research solar panels from type, efficiency, power rating, etc. Then, a solar charge controller must be selected, a device that sits 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 6: Battery Selection

Manufacturer	**Ampere Time**	**Eco Worthy**	**Expert-Power**	**Eco Worthy**
Voltage	12	12	12	12
mAh	6000	10000	5000	8000
Watt per hour	76.8	120	64	96
Cost	\$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 7: Solar panel types

Solar Panel Type	**Monocrystalline**	**Polycrystalline**	**Thin - Film**
Efficiency	>20%	15 - 17%	6 - 15%
Power Rating	$\leq$300W	240 - 300W	Indefinite
Performance	Most efficient	Efficient	Least efficient
Temperature	High Tolerance	Low Tolerance	High Tolerance
Cost per Watt	\$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 8: Solar panel part breakdown

Sku	P108	L02M10-1	NPA10S-12H	SLP010-12U
Manu-facturer	**Voltaic Systems**	**Eco Worthy**	**Newpowa**	**SolarLand**
Solar Panel Type	Monocrystalline	Monocrystalline	Monocrystalline	Polycrystalline
Dimensions	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
Peak Current	570mA	580mA	630mA	580mA
Open Circuit Voltage	20.45V	20.6V	19.83V	21.6V
Peak Voltage	17.34V	17.3V	16.77V	17V
Wattage	9 watt	10 watt	10 watt	10 watt
Power Tolerance	±10%	±3%	±3%	±5%
Cost	$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 10: Charge Controller

Manufacturer	**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 Consumption	≤10mA	≤10mA	<10mA	≤13mA
Price ($)	23.99	69.99	34.99	69.99

While the model is charging during the day or as a back up, this will be plugged into a wall outlet for power. This means we have to be able to convert the AC voltage coming from the wall is converted to DC voltage for the model to use.

Table 11: AC/DC Converter breakdown

Manufacturer	**SmoTecQ**	**ANLINK**	**TMEZON**
Input Voltage	240V	100 - 240V	100 - 240V
Output Voltage	12V	12V	12V
Current Rating	2A	2A	2A
Connector	5.5 mm x 2.1 mm	5.5 mm x 2.1 mm	5.5 mm x 2.1 mm
Price ($)	12.99 for 2	11.59	8.99

3.3.3 Sensing Subsystem

The required sensing subsystem capabilities include:

- Spectral frequency separation
- Spectral range from 400 to 1700nm
- High Signal to Noise Ratio

Silicon photodiodes have a spectral range across the visible spectrum, usually spanning from 350 to 1000nm. In order to generate spectrographs with the relevant molecular fingerprints of soil substances, we're going to need an infrared detector. InGaAs stands out as the ideal material for this range, with most sensors ranging from 800 to 1700nm. There are several manufacturers of silicon photodiodes, and most of them offer a small selection of InGaAs photodiodes at low prices as well. Below are the four of each.

Table 12: VIS Sensors

Model	BPX 61	ODD-5W	FDS100	PIN-3CD
Manufacturer	Newark	Digikey Opto Diode Corp	Thorlabs	Edmund Optics
Spectral Range (nm)	400-1100	300-1100	350-1100	350-1100
Active Area (mm sqrd)	7.02	5	13	3.2
Cost ($)	13.06	14.50	16.08	34.00

Table 13: NIR Sensors

Model	C30617BH	0800-3111-011	FGA015	N/A
Manufacturer	Digikey Excelitas	Digikey Advanced Photonix	Thorlabs	Edmund Optics
Spectral Range (nm)	800-1700	800-1700	800-1700	800-1700
Active Area (mm sqrd)	0.1	1.36	0.018	0.12
Cost ($)	43.58	50.21	63.00	88.00

From the above, we selected the Newark BPX 61 for our Silicon photodiode and the Excelitas C30617BH for our InGaAs photodiode.

Here are the spectral sensitivity curves for each:

Figure 12: VIS Photodiode Spectral Sensitivity

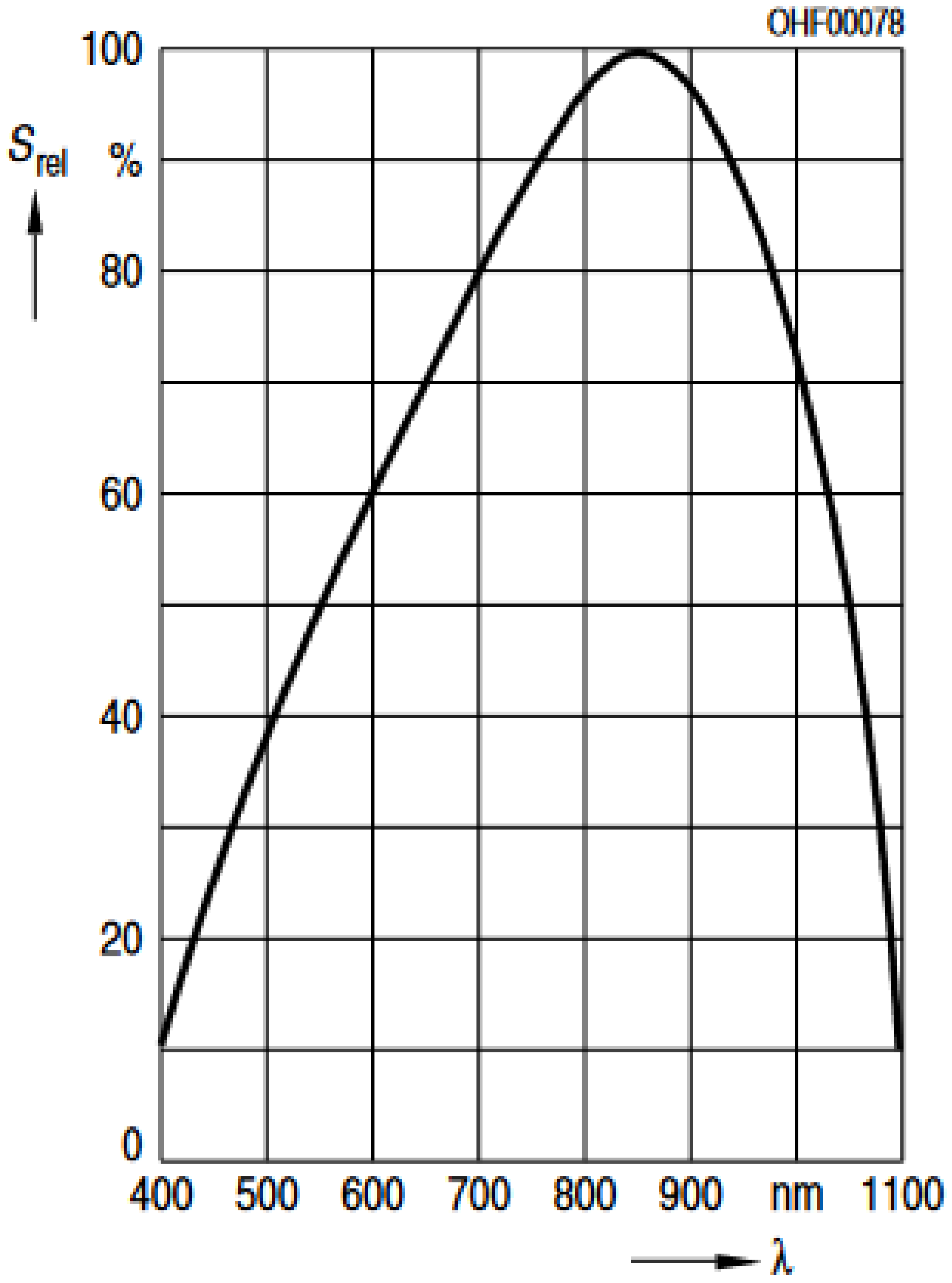
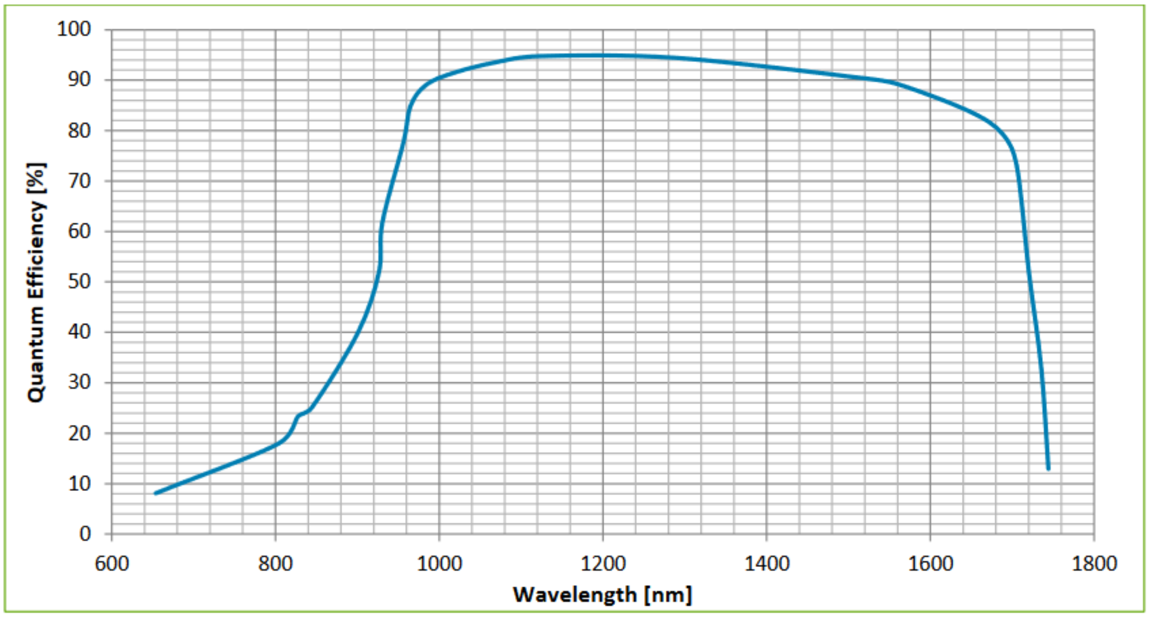


Figure 13: NIR Photodiode Spectral Sensitivity



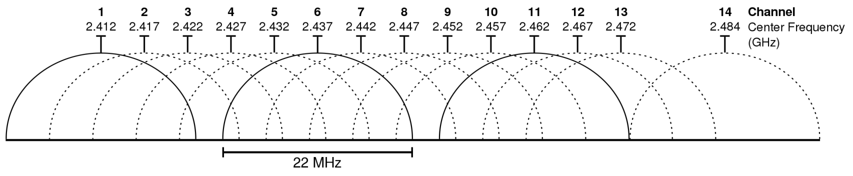
4 Design Constraints

4.1 Related Standards

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.

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 frequences and operates at 2.4 GHz. There are 14 accessible channels, each spanning a band width of 22 MHz (pictured inFigure 14).

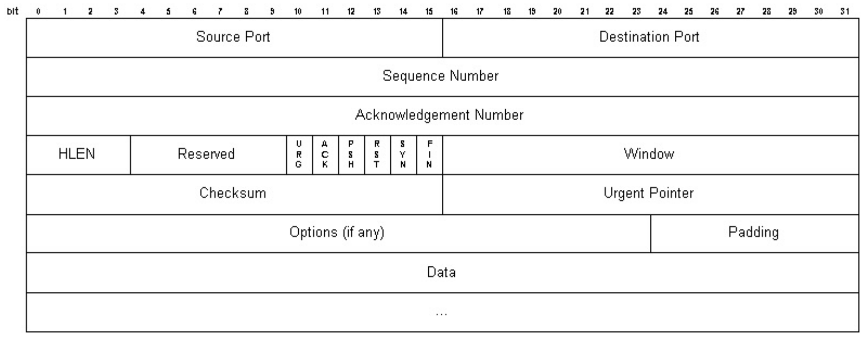
Figure 14: 802.11b/g/n channels


1 2 3 4 5 6 7 8 9 10 11 12 13 14 Channel
2.412 2.417 2.422 2.427 2.432 2.437 2.442 2.447 2.452 2.457 2.462 2.467 2.472 2.484 Center Frequency (GHz)
22 MHz

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

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 15.

Figure 15: TCP frame (The Transmission Control Protocol, Fig. 1)



4.2 Economic Constraints

4.3 Time Constraints

4.4 Equipment Constraints

4.5 Safety Constraints

4.6 Environmental Constraints

4.7 Manufacturability Constraints

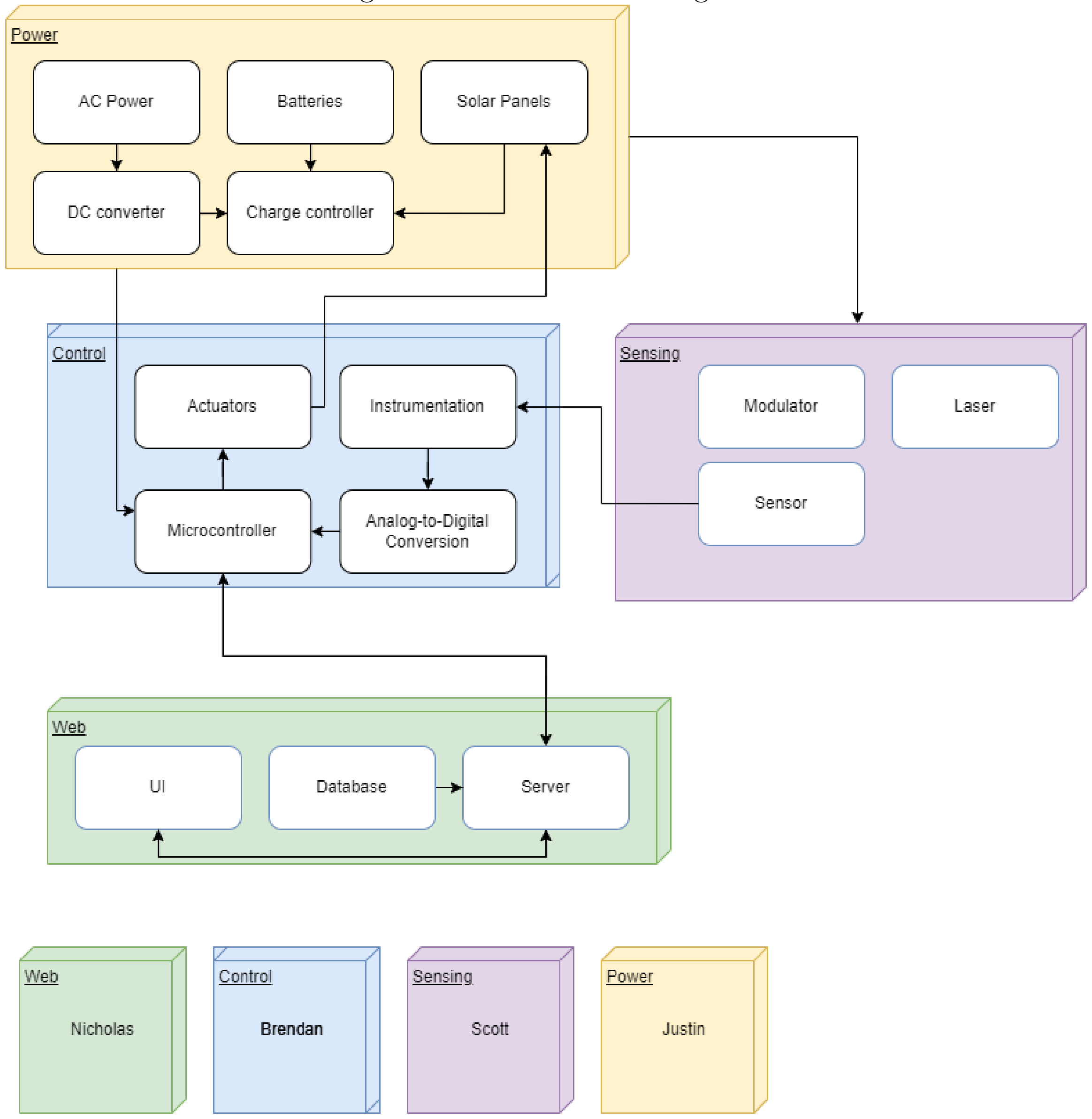
4.8 Ethical Constraints

4.9 Sustainability Constraints

5 System Hardware and Software Design

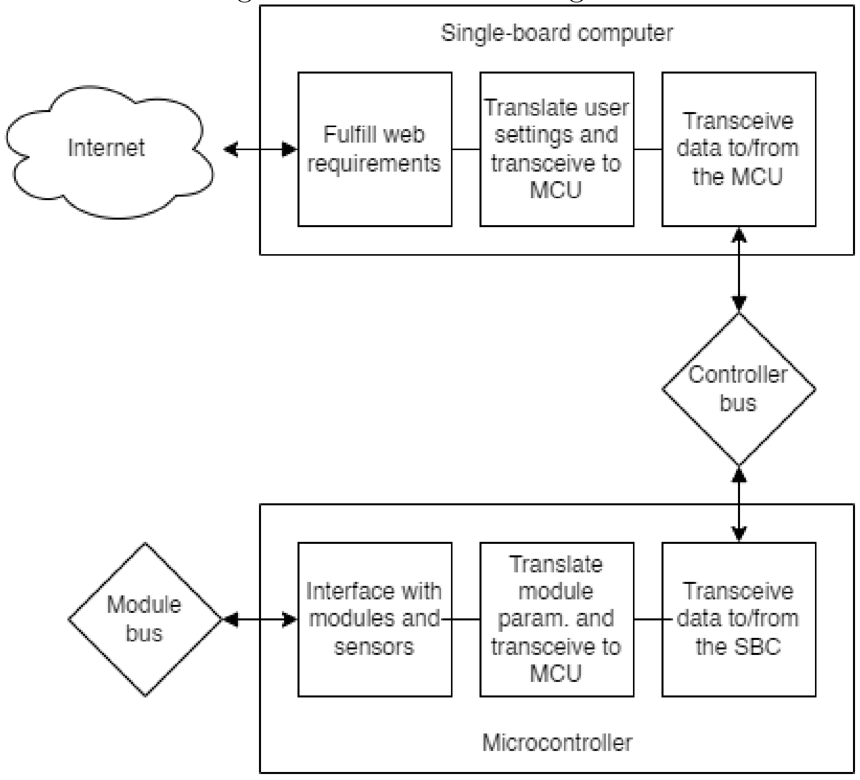
High level overview of design:

Figure 16: Overall block diagram



5.1 Controller Subsystem

Figure 17: MCU block diagram


Single-board computer
Internet
Fulfill web requirements
Translate user settings and transceive to MCU
Transceive data to/from the MCU
Controller bus
Module bus
Interface with modules and sensors
Translate module param. and transceive to MCU
Transceive data to/from the SBC
Microcontroller

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 unncessary, 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.

The Texas Instruments CC3220-series (hence referred to as the "CC3220", 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.

Figure 18: CC3220 system-on-chip block diagram (CC3220 SimpleLink™ Wi-Fi® and Internet of Things Technical Reference Manual, Fig. 1-1)

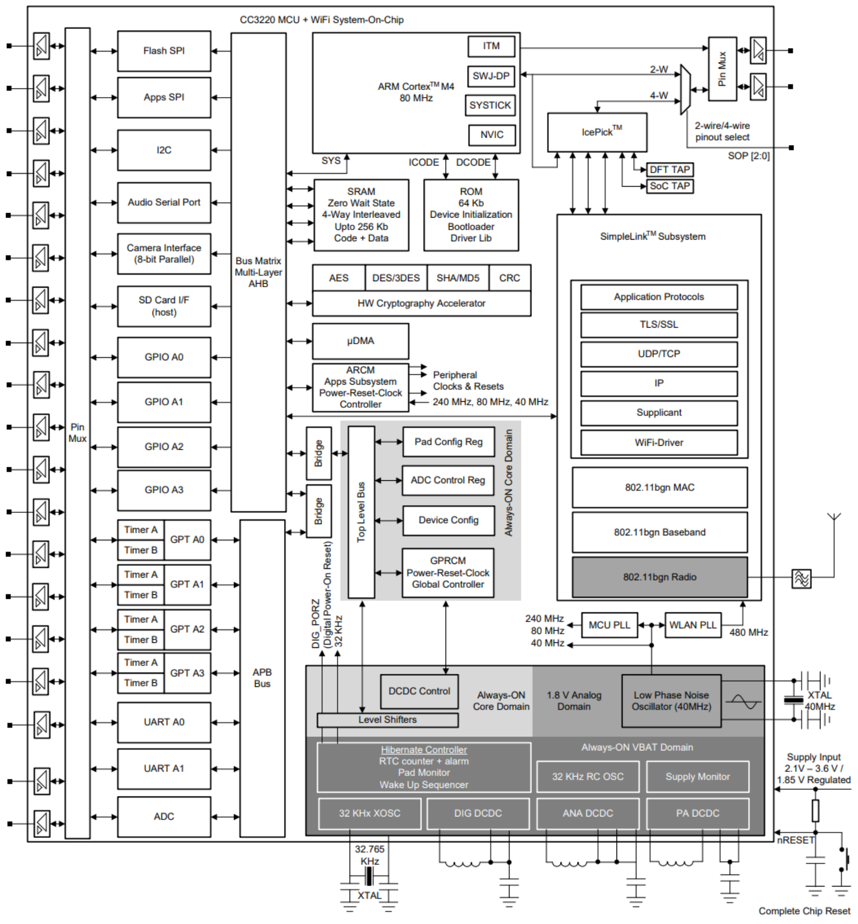
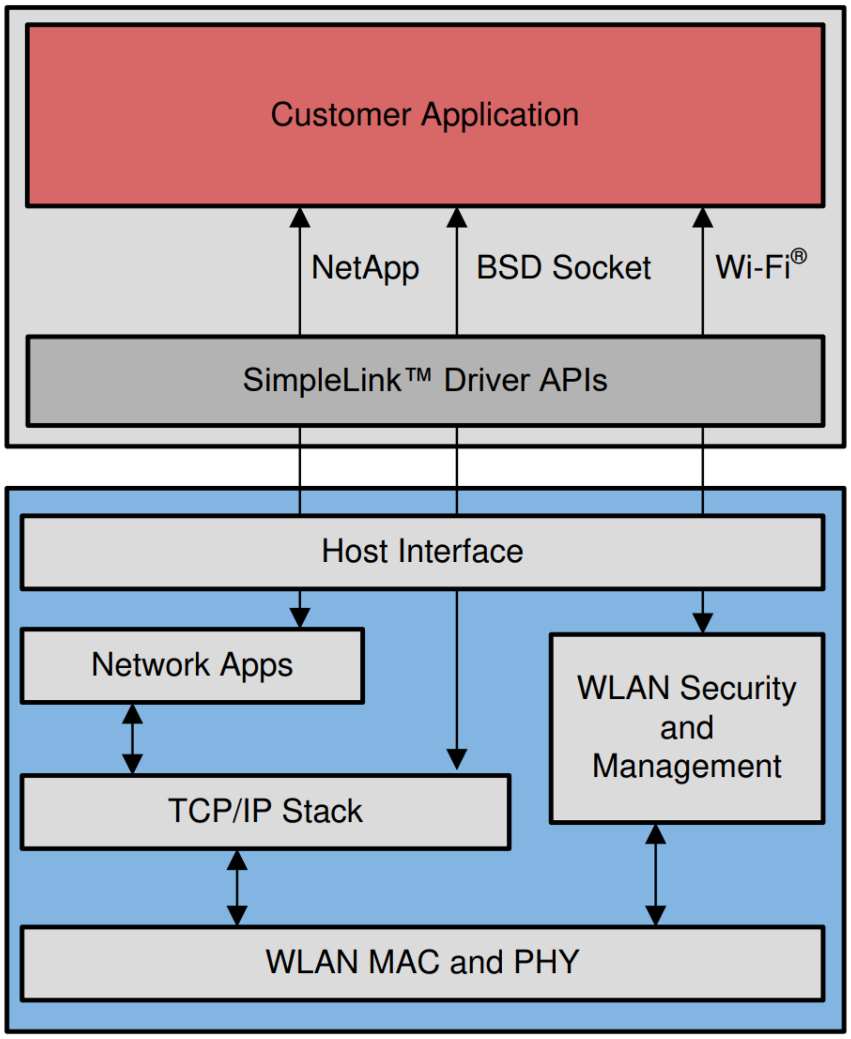


Figure 19: CC3220 embedded software overview (SimpleLink™ Wi-Fi® CC3x20, CC3x3x Network Processor (Rev. M), Fig. 1-2)



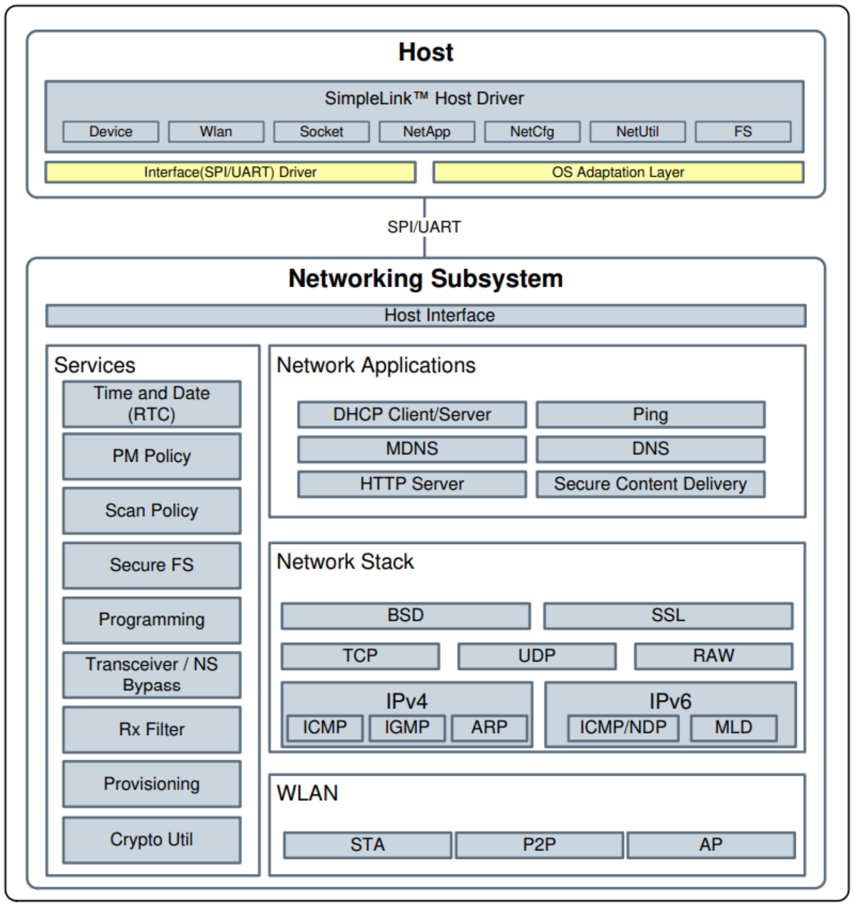
The CC3220's WiFi network processor supports the following standards/features useful to our development (see Figure 20):

- 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
- Built-in transceiver and 2.4 GHz antenna

The CC3220's networking subsysem is further detailed in Figure 20.

Figure 20: CC3220 networking subsystem (CC3220 SimpleLink™ Wi-Fi® and Internet of Things Technical Reference Manual)



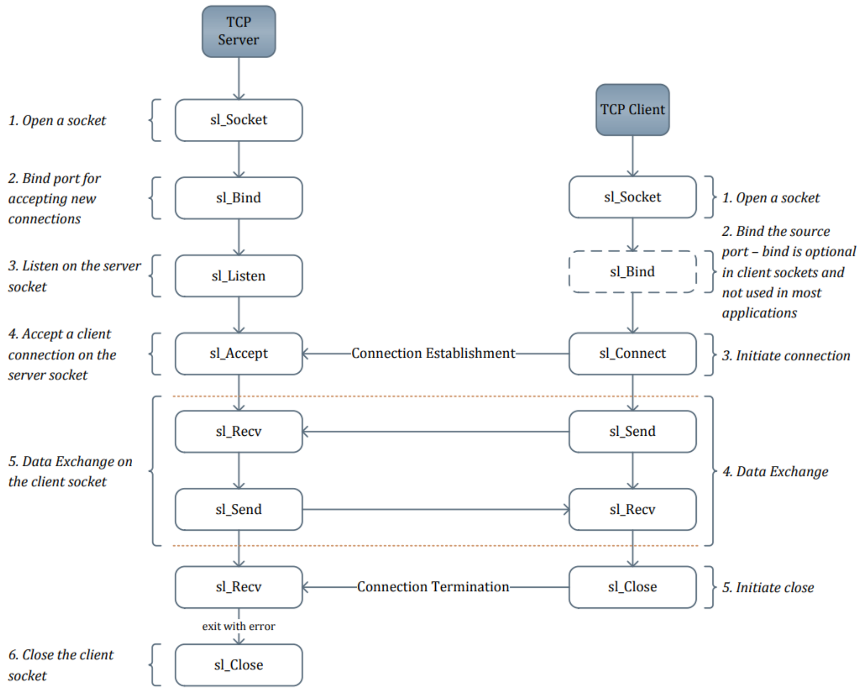
During development of our product, our microcontroller will be hardcoded for the selected WLAN it will access. In the future, there are multiple options our team will be able to explore to allow our product to connect to a wide variety of networks.

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.

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.

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 21.

Figure 21: TCP socket control flow (SimpleLink™ Wi-Fi® CC3x20, CC3x3x Network Processor (Rev. M), Fig. 6-1)



TCP Server
TCP Client
1. Open a socket
sl_Socket
2. Bind port for accepting new connections
sl_Bind
3. Listen on the server socket
sl_Listen
4. Accept a client connection on the server socket
sl_Accept
Connection Establishment
sl_Connect
5. Data Exchange on the client socket
sl_Recv
sl_Send
sl_Recv
Connection Termination
sl_Close
exit with error
6. Close the client socket
sl_Close
sl_Socket
1. Open a socket
sl_Bind
2. Bind the source port – bind is optional in client sockets and not used in most applications
3. Initiate connection
4. Data Exchange
5. Initiate close

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.

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.

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

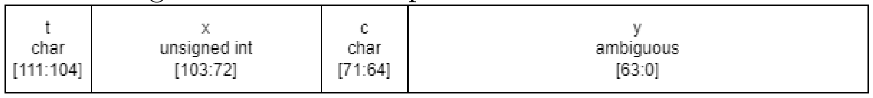
`t x c y`

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

`w y`: water, `y` is boolean
`a y`: solar θ, `y` is double
`b y`: solar ϕ, `y` is double
`l y`: low power, `y` is unsigned int for flags
`s y`: send data, `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 `t 180 w 1`. If AWS would like the MCU to change the θ of the solar panels to 45° immediately, it would send `t 0 a 45.0`. It is important to note that the values of `x` and `y` are not strings (e.g. 45° will not be sent as `"45.0"`), but the actual encoding of 45° as a double-precision float. This is to maintain a consistent command string size amongst all transmissions.

Figure 22: Bitwise representation of command

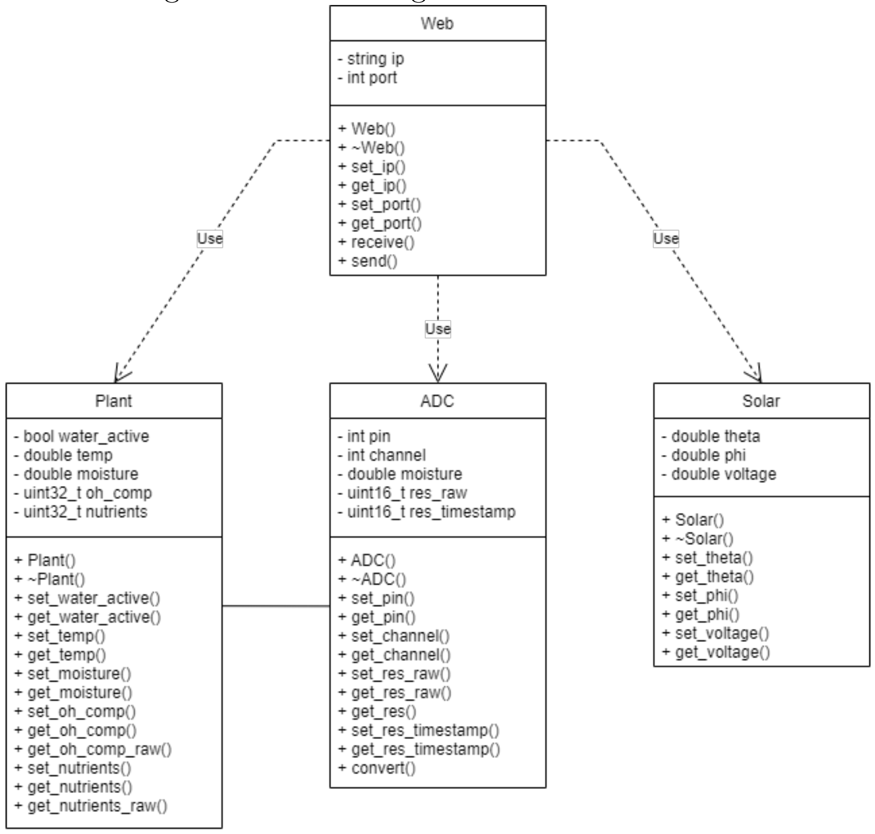


t char [111:104]	x unsigned int [103:72]	c char [71:64]	y ambiguous [63:0]

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.

There will be a few classes defined in the MCU's programming, detailed in Figure 23.

Figure 23: UML diagram of the classes used

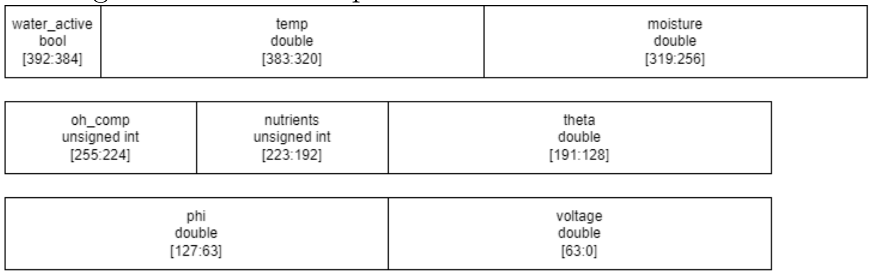


Classes will be laid out in a header file using the following defintions:

```
class ADC {
  private:
    int pin;
    int channel;
    uint16_t res_raw;
    uint16_t res_timestamp;
  public:
    ADC();
    ~ADC();
    set_pin();
    get_pin();
    set_channel();
    get_channel();
    set_res_raw;();
    get_res_raw();
    get_res();
    set_res_timestamp();
    get_res_timestamp();
    convert();
};

class Plant {
  private:
    bool water_active;
    double temp;
    double moisture;
    uint32_t oh_comp;
    uint32_t nutrients;
  public:
    Plant();
    ~Plant();
    set_water_active();
    get_water_active();
    set_temp();
    get_temp();
    set_moisture();
    get_moisture();
    set_oh_comp();
    get_oh_comp();
    get_oh_comp_raw();
    set_nutrients();
    get_nutrients();
    get_nutrients_raw();
};

class Solar {
  private:
    double theta;
    double phi;
    double voltage;
  public:
    Solar();
    ~Solar();
    set_theta();
    get_theta();
    set_phi();
    get_phi();
    set_voltage();
    get_voltage();
};

class Web {
  private:
    string ip;
    int port;
  public:
    Web();
    ~Web();
    set_ip();
    get_ip();
    set_port();
    get_port();
    receive();
    send();
};
```

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 24: Bitwise representation of data sent to AWS


water_active bool [392:384]
temp double [383:320]
moisture double [319:256]
oh_comp unsigned int [255:224]
nutrients unsigned int [223:192]
theta double [191:128]
phi double [127:63]
voltage double [63:0]

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

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.

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.

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.

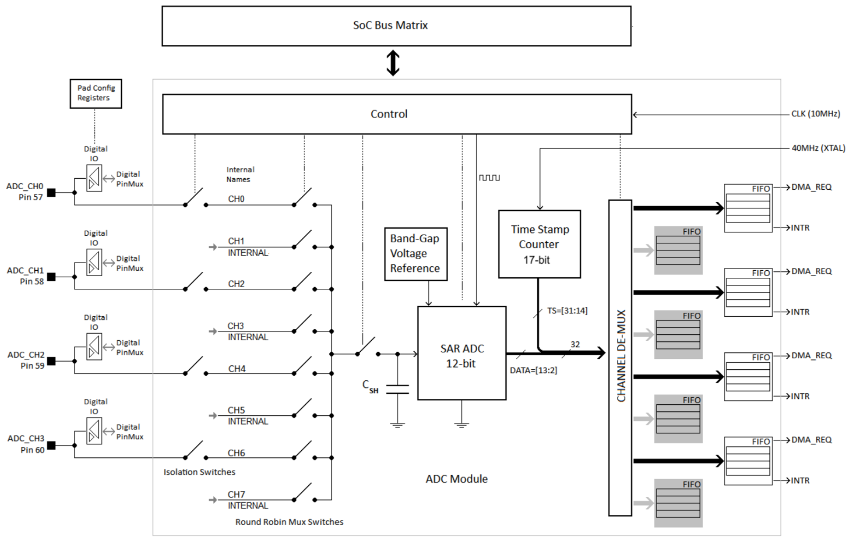
If the MCU receives indication of a start up (via a momentary switch), the MCU will raise an interrupt and execute `isr_start_up()`. This ISR starts up all relevant subsystems and begins the MCU's programming. All classes will be initialized to default values. This ISR starts up the entire system.

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.

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 seperate 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`.

The CC3220 contains a 12-bit general purpose analog-to-digital converter (ADC) with 4 externally-accessible channels and a sampling periodicity of 16 µs per channel (62.5 ksps per channel). The architecture of the CC3220's ADC is shown in Figure 25.

Figure 25: CC3220 ADC module architecture (CC3220 SimpleLink™ Wi-Fi® and Internet of Things Technical Reference Manual, Fig. 13-1)



It is expected that the spectrometer circuit will provide a current between 20 μV 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:

$$\frac{(1.8 - 0)\,\mathrm{V}}{2^{12}\,\mathrm{steps}} = 0.4395\,\mathrm{mV/step} \tag{1}$$

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

$$\frac{(80 - 0.02)\,\mathrm{mV}}{0.4395\,\mathrm{mV/step}} = \lfloor 181.98 \rfloor = 181 \tag{2}$$

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`.

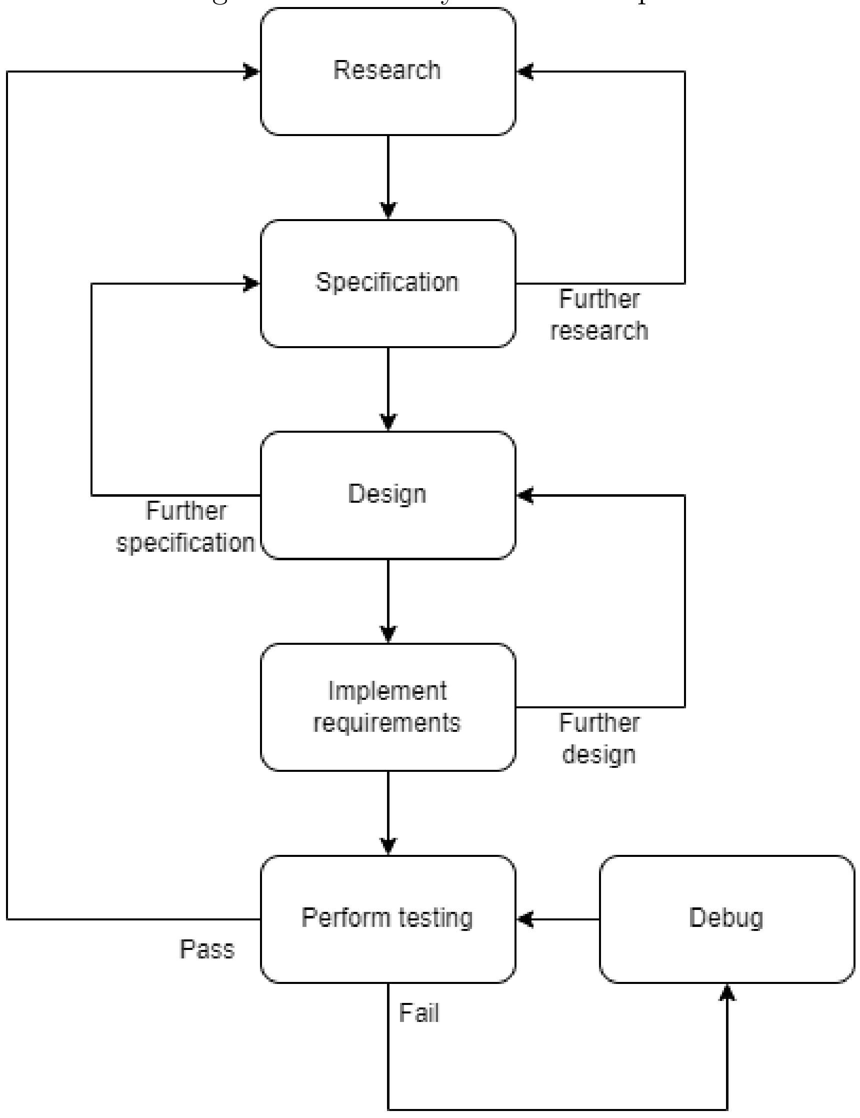
In the future, a more precise ADC module could be explored to give the CC3220 more resolution when performing spectroscopy.

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.

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.

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.

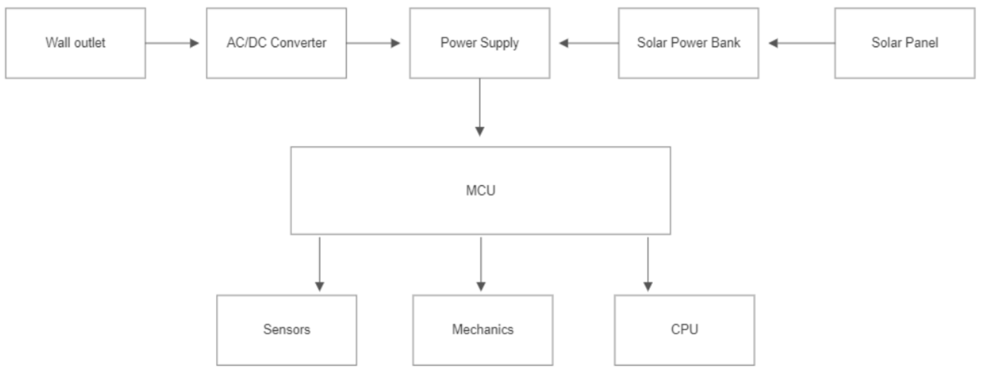
Figure 26: UML diagram of the subsystem's development methodology



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.

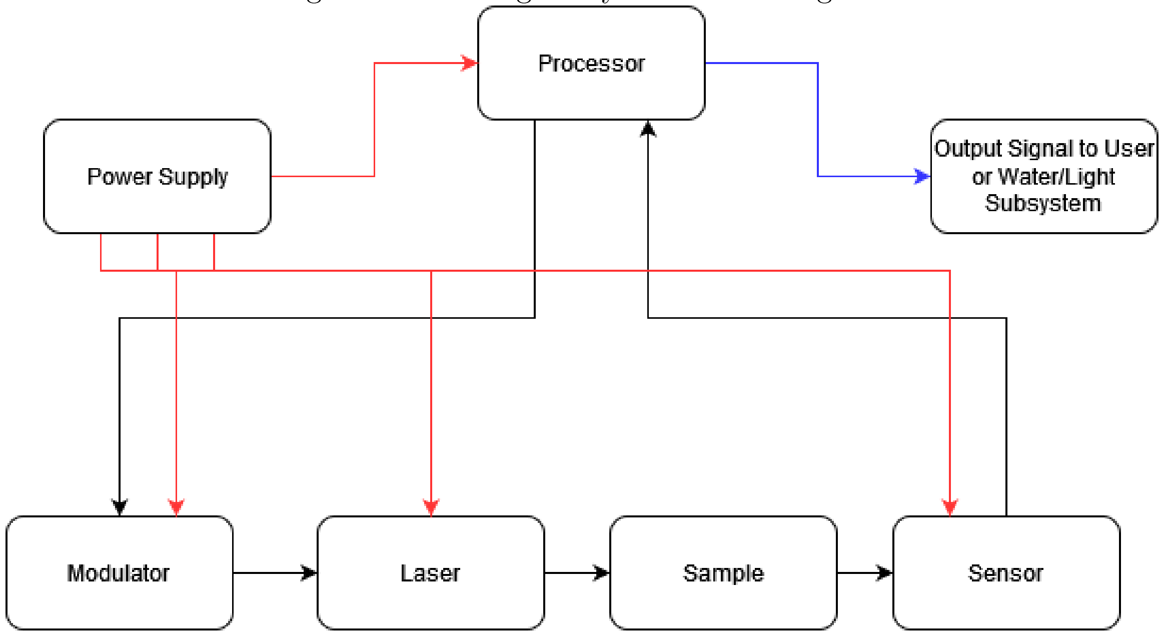
5.2 Power Subsystem

Figure 27: Power subsystem block diagram



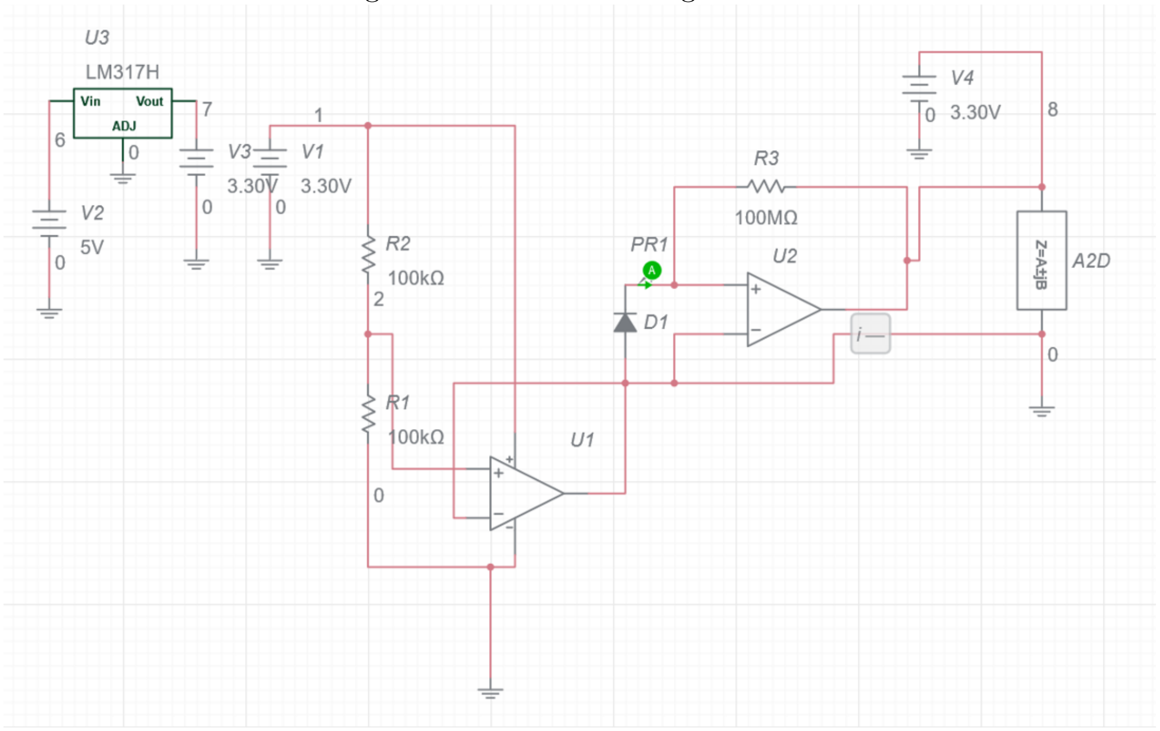
5.3 Sensing Subsystem

Figure 28: Sensing subsystem block diagram



The purpose of the Sensing subsystem is to generate a spectrograph of the soil through data delivered to the MCU. Then the MCU will interpret the wavelengths by measuring the amplitude of the signal at several key frequencies. This will determine whether the MCU will proceed to engage the Power and Web subsystems to modify the environment of the plant.

Figure 29: Photodiode Signal Filter



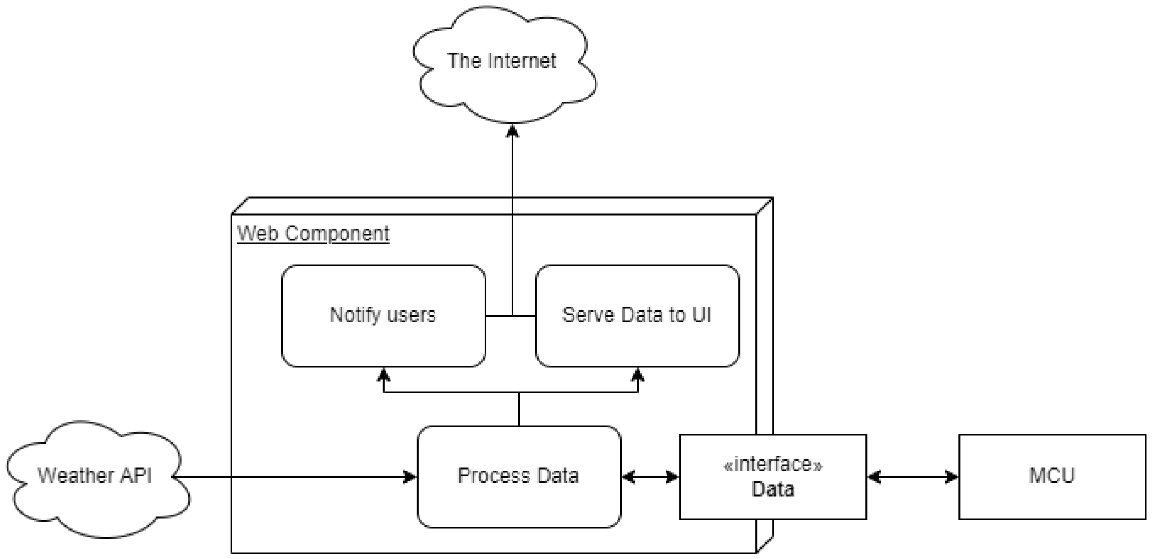
The Photodiode works by converting a small portion of the incident light into electrical current across the face of the semiconductor. The diode will have a small current running through the circuit hooked up to an amplifier for boosting the signal to a detectable level. Then another op amp will cut 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 the 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 30: Web component block diagram



Section **??** 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 suppport 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.

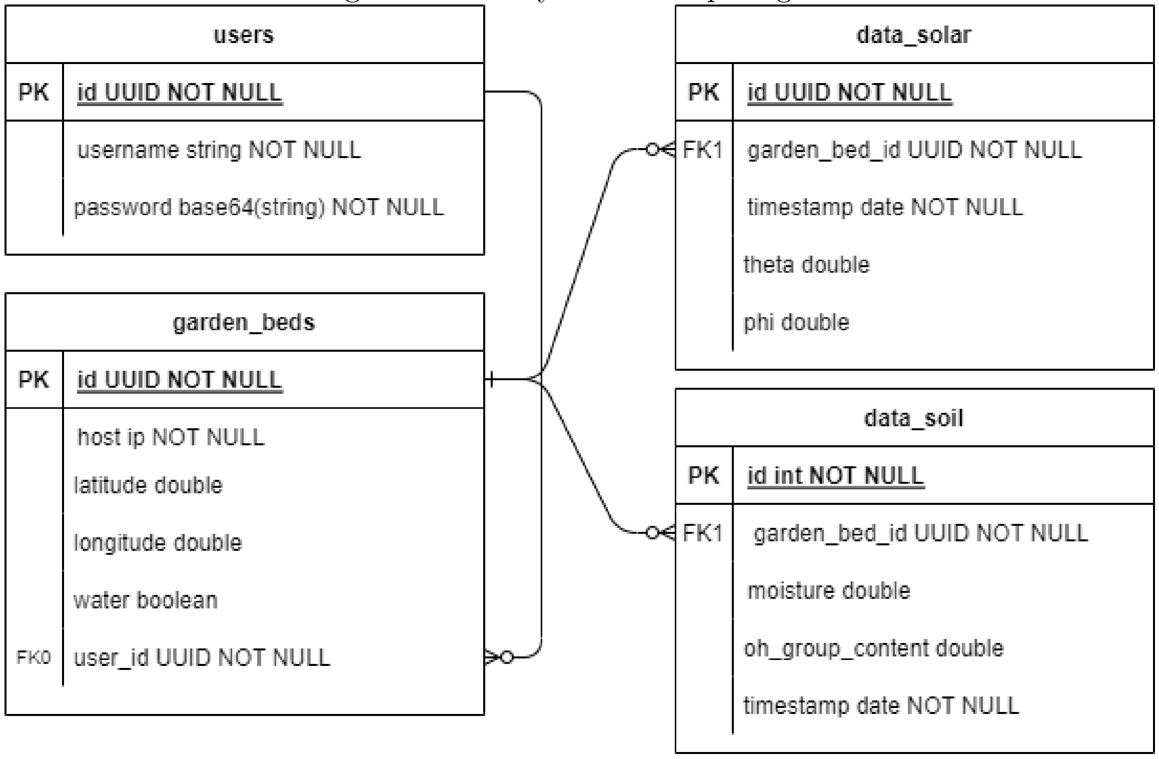
5.4.1 Server Backend

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 serve 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 31 has some details on the type of data and where it can be found within the database given the design above.

Figure 31: 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.

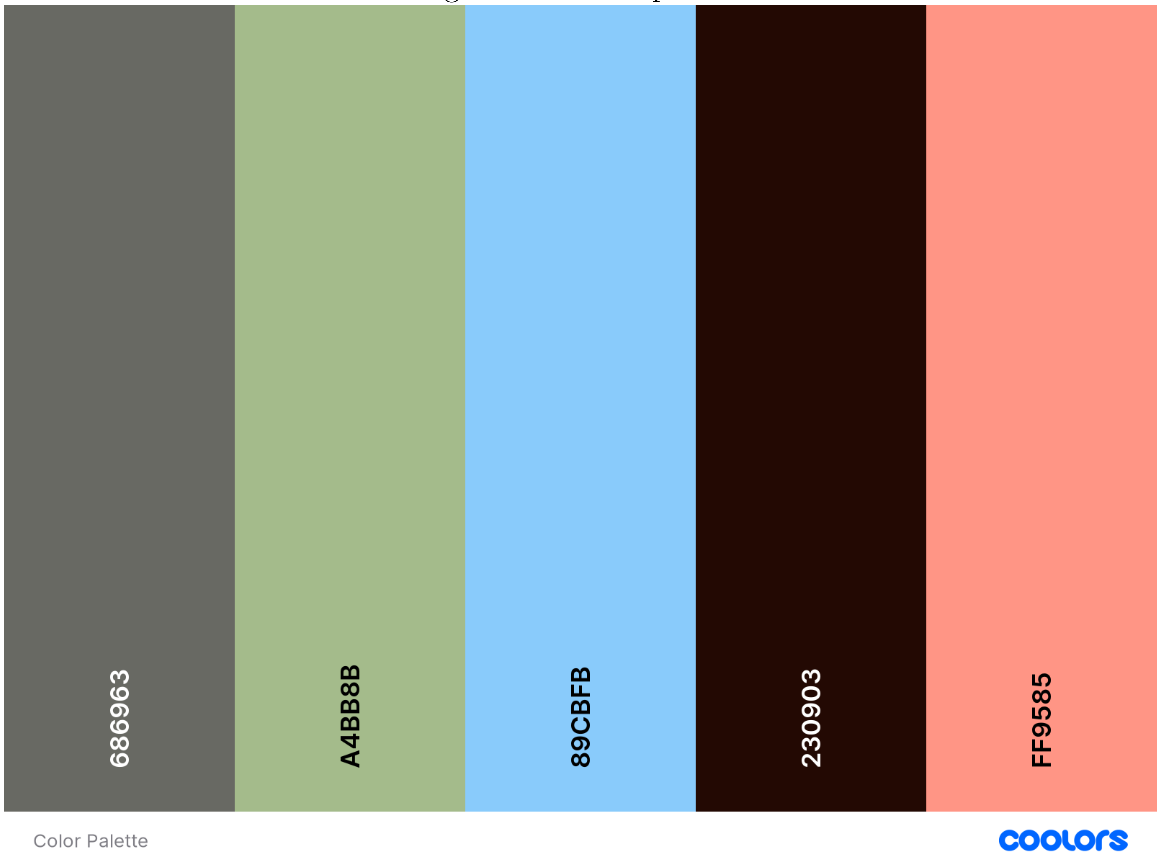
Data Processing All data will be processed on the EC2 instance within the server. Using some heuristic methods and research into peak conditions for different plant types, the server will be able to determine different actions to be made on the plant bed in order to promote growth.

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.

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 32: 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.

6 Testing

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

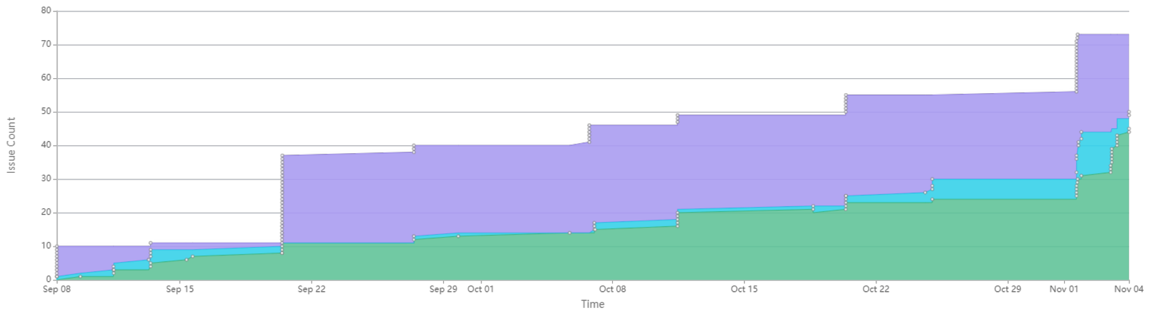
7.1.2 Spring

- 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 33: Cumulative Flow Diagram from Jira



In Figure 33: 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 33 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 14: Breakdown of budget by subsystem

Subsystem	Estimated Cost	Comment
MCU	$60	The MCU, wiring harness
Power	$200	Solar panels, batteries, control system
Sensing	$100	Components for sensing, optical sensors
Web	$30	Web service pricing
Non-Subsystem	$100	The plant bed, soil, water, fittings, etc
Total	$490	