Greenie: The Smart Irrigator

UNIVERSITY OF CENTRAL FLORIDA

Department of Electrical Engineering and Computer Science

Dr. Samuel Richie
EEL4915: Senior Design II - Fall 2021

Senior Design 2 Documentation

Group 10
Elliott Gray - Computer Engineering
Patricia Mae Luzano - Computer Engineering
Kevin Rodriguez - Electrical Engineering
Angelica Vargas Martinez - Electrical Engineering

Table of Contents

1. Executive Summary

Today, water scarcity is becoming one of the biggest global issues. Just as water is a universal right and a necessity for humans and animals, it is also for plants. Agriculture is recognized for the great demand for water that it entails, where the waste of the limited resource is a significant problem, especially when using traditional mass irrigation techniques such as manually operated sprinklers and/or water channels. The unnecessary use of water is a notable problem in worldwide agricultural activity since many times the irrigation systems used by farmers supply inadequate amounts of water to crops. Therefore, the agricultural sector is one of the main stakeholders in developing methods that help with water conservation. This has motivated the development of automated irrigation systems that facilitate the adequate automatic supply of water and thus optimize the use of water in the food production process.

Our project, Greenie: The Smart Irrigator, hopes to assist the irrigation techniques implemented by worldwide agricultural activity and individuals and their communities. Greenie focuses on quality and efficiency to grow healthy and sustainable plants while using less water than traditional mass irrigation techniques. Although Greenie was designed with water conservation as its primary focus, our project was also directed towards people who enjoy gardening as a hobby. Since the start of the current pandemic, individuals have acquired new hobbies to combat the repetitiveness and lack of social interactions that result from global lockdown restrictions. The most common hobby adopted was gardening, with a focus on fruits, herbs, and vegetables. Without a large outdoor garden to alleviate the confinement, the average person resorted to planting wherever there was space for a planter, whether it was on terraces, inside the house, or in an apartment. Gardening has enjoyed an increase in popularity due to the quarantine, and even though lockdown restrictions are currently being lifted in many countries, individuals continue to practice growing their food.

Greenie was developed to grow and take care of different kinds of commonly used herbs in a smart manner. The herbs supported are basil, bay leaf, cilantro, chives, lemongrass, mint, oregano, rosemary, sage, and thyme. Our automatic irrigation system protects the user's plant by providing careful maintenance and data that helps the respective plant stay healthy. The product is controlled by a central microcontroller that acts as the "brain" of the system. Based on designated programming times that the user can assign through the use of our web application or by voice commands through Alexa, the herb will receive the right amount of water that it needs, resulting in water that is not needed to be saved. Watering requirements for the different herbs supported are saved in our database. Smart components such as soil moisture sensors, pH sensors, rain sensors, and humidity and temperature sensors are connected to our system. Through these, the user can access data regarding their herb at any time they would like. In addition to the advanced capabilities offered by our sensors, Greenie also has access to a weather station through which it shows the user the current weather forecast on our web application. If it is raining, the user can place the system outside, resulting in greater savings of water. In this document, we will be presenting our project

by discussing its administrative and technical content, including its associated specifics, constraints, standards, and design.

2. Project Narrative Description

In this section, we will be giving a narrative description of our project. The following subsections are going to serve as the transition statements between our declared issues and how we plan on helping to solve them. We are going to discuss the motivation behind designing and building a product that combines smarter irrigation systems with IoT, our project's goals and objectives, its function, and the requirement specifications. In terms of the function and requirements of our project, both hardware and software are going to be discussed. In addition, we are also going to discuss and present the House of Quality that we built for our product.

2.1 Motivation

Currently, about a third of the global population lacks access to potable water. As the global human population continues to increase, the availability of water could become an alarming situation. Another important fact to highlight is that an outstanding amount of water and land is consumed by agriculture. In the United States only, agriculture accounts for about 80 percent of the consumptive water use while in many Western States it accounts for over 90 percent[1]. This is where the greatest waste of this resource occurs, being that a large concentration of the water utilized for irrigation purposes is constantly lost due to events such as water overflow, evaporation, inefficiency resulting from the use of outdated tools and/or equipment, etc[2].

Irrigated agriculture has become one of the most important and lucrative sectors of U.S. agricultural production while still being the largest consumer of consumptive water. In 2012, about half of the total value of crops sold on 28 percent of U.S. harvested cropland was due to irrigated farms[1]. In the same year, approximately 7.6 percent of all U.S. cropland was irrigated, maintaining the decline in irrigated U.S. cropland since 2007. This decline has been linked to the scarcity of water as a supply, causing drought conditions across the farmable regions. Methods of using water smartly have been looked into because of this.

The future of worldwide irrigated agriculture depends, to a large extent, on the implementation of smart irrigation systems in both large scales, such as farms, and in smaller scales, such as home gardens. This allows the most efficient use of water, energy, and other resources in such a way as to increase the production levels using fewer, crucial resources. With intelligent irrigation, the profitability of fields is increased, the cost of human efforts is reduced, and the environmental impact of heavy-scale agriculture is minimized by reducing both the use of water and the contribution of polluting elements to the environment[2]. The implementation of smart irrigation systems is essential to ensure the sustainability of irrigated agriculture.

In hopes of aiding this situation, our team has decided to design a smart irrigation system using the Internet of Things technology, or IoT, by implementing a network of sensors and APIs to connect and exchange data over the internet. A smart irrigation system could be of great help in managing water utilization around the world and increasing productivity in fields. We have chosen to implement a more local approach with our project, giving individuals the chance to help their communities and the global population by growing herbs from the comfort of their homes.

2.2 Goals and Objectives

The goal of this project is to create an easy-to-use and portable smart irrigation system using IoT. Our product, Greenie, allows users to monitor and irrigate herbs remotely, making the gardening process hassle-free. Given that the system is compact, it can be placed anywhere inside a home or an outdoor garden. Once programmed, the system works independently to optimize irrigation with smart gardening tools, resulting in savings on water consumption compared to manual watering, as water is only used when needed and exactly the right amount is applied. Our system is time-saving, resource-saving, and reduces the workload of the user. Given that our system utilizes IoT, it must be connected to a Wi-Fi network to successfully perform its duties.

The strategy to accomplish this project is to acquire the necessary equipment to put together a smart irrigation system using IoT. Parts from different distributors were compared to obtain the best prices for each component needed. Once the parts were obtained, the necessary printed circuit board was constructed. A web application that tracks the plant's data, such as watering times, soil preference, pH preference, and the ideal humidity and temperature it requires from the surrounding air was also constructed. The application, which the user can access comfortably from anywhere, is available on any mobile device through the use of a link, making our device as easy to use for any type of user, whether they are beginners, intermediate, or advanced gardeners. This information will be invaluable to allow the user to make smart decisions on their plants.

2.3 Function of Project

Using our hardware, the user is able to acquire various information regarding the watering levels, soil dampness, and humidity through the use of our designed microcontroller that is displayed on the LCD. This information includes soil moisture, temperature and humidity, watering by rain, and pH levels. The sensors required for each were placed in the planter, which sends real-time data to the microcontroller. Depending on the herb, a soil moisture range is indicated and whenever the values are outside of the specified range, the microcontroller automatically turns on/off the water pump. Through the use of an integrated weather monitoring application, our system is able to inform the user if rainwater is able to irrigate the herb on any specific day of the week.

Using our software, the user can water their plant automatically through their mobile device or voice, and also present information regarding the plant. Through the use of our web application, the user is able to tap on their screen to either select automatic watering or water their plants manually. The user is also able to ask the system to do either option through the use of their voice thanks to Alexa integration. A Wi-Fi module and relay module are included for this purpose. Additional Alexa commands also work with our LCD screen, where the user can observe data such as soil moisture levels, whether the plant was watered by rain or not, temperature, humidity, and pH information.

2.4 Requirement Specifications

The following section will discuss the requirement specifications associated with our project's hardware and software. Under section **2.4.3 Specifications**, a list of attributes and their specified descriptions can be observed in **Table 1**. The requirement specifications will discuss the properties that we determined our project must meet when we submitted it for final approval. It also discusses how the system interacts with its environment, both hardware components and software programs, as well as with the user.

2.4.1 Project Hardware

Our design must utilize a Wi-Fi module, which allows users to develop and create IoT projects. The module is going to be used to develop a cloud interface for saving information online that the user can access wirelessly. The Wi-Fi module lets the hardware connect with the hardware wirelessly. The user is going to be able to send commands to the Wi-Fi module and transmit data to the cloud interface for analysis. The Wi-Fi module must be connected with an Atmega328P microcontroller that, once programmed, is going to take input from the soil moisture measurement sensor, rain sensor, temperature and humidity sensor, and pH sensor as well as control the other electrical components of the system that are going to be connected to it.

For our soil sensor, we plan on utilizing the FC-28, while for our rain sensor, we plan on utilizing the MH-RD. The rain sensor must detect if the plant has been watered by water droplets, and if it has not been, the soil sensor must communicate to the system, letting it know to deposit water to the plants. The rain sensor included in our system is also going to be combined with data supplied by a weather monitoring system. For our temperature and humidity sensor, we are going to utilize one that is already owned by one of our team members, the DHT-11. For our pH sensor, we plan on utilizing the PH0-14. The pH sensor is going to provide real-time data to the user regarding the current pH in the planter's soil. We are also utilizing an LCD screen already owned by one of our team members, the LCD1602. The LCD screen must be connected to the microcontroller unit and through it, the user is going to be able to cycle through to observe data on soil moisture, the required range of moisture for the specific plant, whether the plant was watered by rain or not that day, temperature and humidity

information within the air, and if the plant requires watering from the user or not given that it is going to constantly access data from a weather monitoring application.

2.4.2 Project Software

Our web application must contain the herb's daily statistics from the soil moisture sensor, rain sensor, humidity and temperature sensor, pH sensor, and an on/off button to control the system. The plant's data must be updated often, preferably once per hour, such that the user can observe the sensors' measurements on demand. Through the use of our database, the user is going to be able to select which plant they currently have planted and the system is going to decide how much water must be dispersed to each herb.

We planned for our system to integrate a weather application through which the rain sensor in the device would know in advance if watering is needed for the herb on any specific day. This weather monitoring system would acquire data from online weather stations to optimize irrigation needs. By performing data analysis, our system would have been able to automatically recalculate irrigation needs if, for example, rain is forecasted for the following day. After acquiring this information, the system would have informed the user that watering is not needed the following day due to rain through the use of printed messages on the LCD screen. We were able to integrate weather prediction into our system, but due to time constraints, we were not able to include all the features we had planned.

Alexa support must be integrated into the system so that users can water their plant, access its statistics, and turn on/off the system through voice commands. To accomplish this, we are going to use a relay module along with a Wi-Fi module and pair it with a small smart speaker, the Amazon Echo Dot 4th Generation. As it is commonly known, Alexa will be able to recognize voice input from the user, process the request, and then allow the system to perform actions as specified. The commands that it will use will be mostly short and simple ones.

2.4.3 Specifications

In **Table 1** (see below), the attributes required for our project can be observed as well as their numerical description. As seen in the table below, attributes such as the maximum dimension of our product, its maximum weight, the voltage range needed from the power supply to successfully power our microcontroller, the minimum quantity of pins we require from a microcontroller, the LCD dimensions we desire, what and how many sensors we plan on utilizing, how often we plan on updating sensor measurements, the response time of our relay module, the specific water source size and water pump requirements, our interface's response time, how many plants our database is going to support, the minimum quantity of Alexa commands our product is going to have, and the communication protocols we plan on utilizing for our system and sensors. The numerical description for each attribute can be observed in **Table 1** below.

Attribute	Description
Dimension	20” x 20” x 20” (should not exceed)
Weight	15 lbs. (should not exceed)
Power Supply	Range needed for microcontroller: 7 V - 12 V Duration: At least 12 hours
Microcontroller	I/O pins: 10 (min) Operating voltage: 5 V
LCD	16 Character x 2 Line
Sensors	1 soil moisture sensor 1 rain sensor 1 humidity and temperature sensor 1 pH sensor
Sensor Measurements	1 per hour or on demand by the user
Relay Module	Voice Response Time (Alexa) <= 5 seconds
Water Source	24 fl oz water container
Water Pump	Pump head: 2.5 M / 8.2 ft (min) Flow rate: 240L/H 63.4GPH (min)
Interface	< 5 seconds response time
Plants Supported	10 herbs total 1 supported at a time
Alexa Commands	At least 3 unique commands
Communication Protocol: System	Wi-Fi
Communication Protocol: Sensors	UART SPI I2C

Table 1: Project Specifications

2.5 House of Quality

The House of Quality is a type of visual diagram utilized to show the relationship between the needs of the customer with that of a project's engineering requirements. In some areas, the customer's needs will strongly overlap with requirements while in other areas they will not. What is important to recognize is that the desires of the customer are getting attention. By creating these types of diagrams, a team of engineers will be able to determine if their project requirements have a positive/strong positive or negative/strong negative alignment with the customer's needs. As seen in **Figure 1,** the customer's needs are represented on the left side while the engineering requirements are on top. The numbers next to the customer requirement represent the level of priority the requirement is to the customer and the numbers below the engineering requirement simply list what was mentioned in **Table 1**. Lastly, the middle section highlights the correlations between the customer requirements and engineering requirements, and the triangle at the top highlights the correlations of the engineering requirements amongst each other.

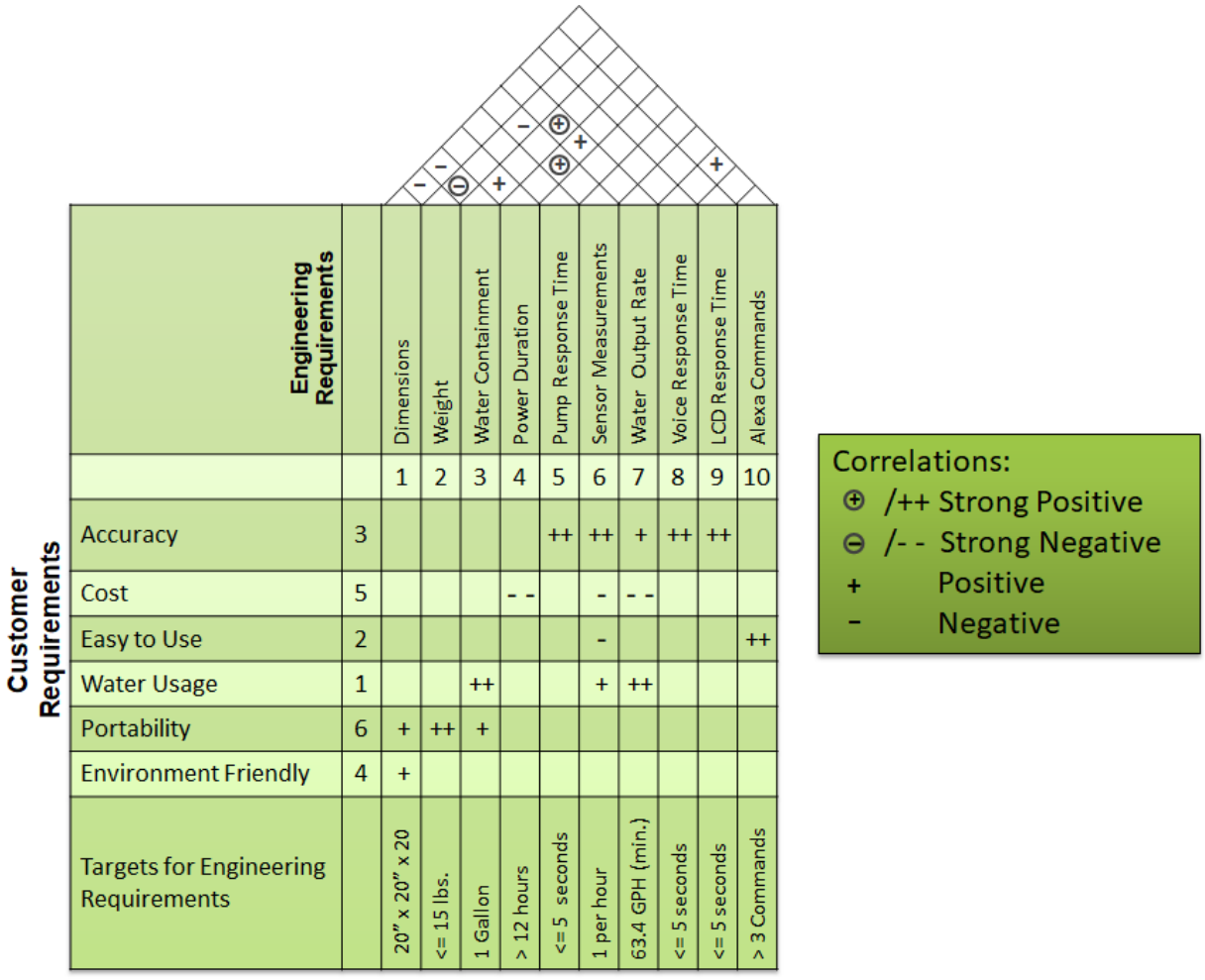


Figure 1: House of Quality

3. Project Constraints and Standards

3.1 Constraints

In this section, we will be discussing the constraints related to our project. Greenie, like every other engineering product, is subject to realistic design constraints, including economic, environmental, ethical, health and safety, manufacturability, political, social, sustainability, and reliability. Given the nature of our project and the current pandemic, constraints such as plant parameters, product availability, scalability, and time are also valid. All previously mentioned constraints will be reviewed in detail in this section.

3.1.1 Economic

The investment cost of systems dealing with irrigation varies highly given that operating and maintenance costs must be taken into account. Irrigation factors such as the water supply, the water pump head, the types of materials and technologies used, the market availability and condition of equipment and materials, as well as the distance between the irrigation sites and the irrigation equipment are all decisive aspects that must be taken into consideration whenever developing a new agriculture-based project. Even though our project is a small-scale, individual-based product, large-scale factors like the ones previously mentioned largely impacted our design for Greenie.

An economic constraint that always comes into play when getting any project realized is cost. Having unreasonably high expenses can often lead to projects not being completed at all. In the case of our project, due to it being self-funded, we looked into several different sources to find the most affordable materials possible. In doing so, we recognized that there were some components where the most affordable option was not always the best option. From source to source we noticed that while some products were more affordable than others, not all fell in line with what we outlined in our requirement specifications. So, to try and alleviate this issue, we decided to outline key features in our project where we could focus on quality and be more flexible in our budget. Therefore, we could have areas in our budget where some items may not be the most affordable, but the tradeoff is that all requirement specifications are met. In addition to finding the most affordable materials available, we also decided to utilize materials that we already owned. In doing this, we were able to limit this constraint by reducing the number of items that we would otherwise be paying out-of-pocket.

Another economic constraint that our project was under was its competitive price when compared to other similar products that already exist in the market. Because our project includes many components that have been donated by our team members to keep our costs low, we do not have an exact number of what the price for our product would be and how it could compete with other technologies that are low cost.

3.1.2 Environmental

The most common environmental constraints that products today are being restricted by are water, humidity, and temperature. Given that our project is intended to be a resource-saving product, water is going to be smartly utilized. Combined with our software, Greenie provides the user with water consumption reduction by performing irrigation at the optimal time for the plant. Even though we have this design in mind, if it is not implemented properly, Greenie can become as resource wasting as traditional mass irrigation techniques. If the included water pump is not installed properly or if it becomes damaged throughout the testing and prototyping stages, water could easily leak from the tubing. Issues could also arise with our database. If our database is not built properly and the watering requirements for each herb are not correctly saved, the water pump could disperse incorrect amounts of water. This scenario would result in the unfortunate misuse of water and the health of the plant being compromised.

Humidity and temperature, too, are environmental factors that do not necessarily affect our project unless our design incorrectly implements the use of our sensors. Greenie is going to include a humidity and temperature sensor, a rain sensor, a pH sensor, and three soil moisture sensors. These sensors are going to collect environmental data like soil humidity, air humidity and temperature, and pH levels to effectively plan irrigation cycles. If used incorrectly, the sensors could provide the system and the user with inaccurate data readings, which could result in the wrong use of resources and again, in the health of the plant being compromised.

An environmental constraint that our project is under is the limitation on crop productivity. Because we chose to go for a compact and low-cost smart irrigator option, we had to place some limits on our product. The main limitation is that Greenie will only provide support for ten herbs, these being basil, bay leaf, cilantro, chives, lemongrass, mint, oregano, rosemary, sage, and thyme. This means that any other plants not mentioned above will not be supported, which restricts what kind of crops the user can grow.

3.1.3 Ethical

There are some ethical considerations to take into account when discussing a smart irrigation system. We need to make sure that none of our components and the way that they are set up, can bring harm to any of the plants we are working with and bring harm to any animals or insects that could potentially interact with our system. To ensure that no harm comes to the environment, we made sure that the system we make follows the Toxic Substances Control Act of 1976. None of our materials have any toxic substances to them, which makes Greenie environmentally friendly and up to the standards set forth by the EPA. We also need to make sure that our product would not discriminate through time, meaning that we took care so that it can always be affordable to everyone. Finally, we were aware of how our product can make our target consumers feel, we also made sure that how it looks and how it is made will not offend any person and we shall make sure that our irrigation system is within the norms of society.

3.1.4 Health and Safety

These constraints are intended for use by all those responsible for preventing risks that technologies could create for health, safety, and security. To build a product that is safe to use for users, we must take into account the health and safety constraints that it entails. As a team, we must understand the possible health hazards that our product could cause to make sure that it does not compromise the health of the future user. Because our product combines electrical components and water, there is potential for fatal accidents to occur. Given this possibility, care must be taken to minimize the likelihood of this occurring. As a precaution, the water pump in our system will be placed as far away as possible from the electric parts. To add more protection for users, we attempted to shield the electrical components from moisture as well. This could be accomplished by adding a conformal coating. This conformal coating would cover the PCB of our project, ensuring that our components are safe not only from humidity but also from dust, chemicals, and extreme temperature changes. To keep our team from harm, the members assigned with this task will make sure to apply the coating in a well-ventilated area while avoiding as much external contamination as possible. The team member will also be wearing the necessary protective gear. Due to time constraints, our wish to include the conformal coating on our components was not completed.

3.1.5 Manufacturability

This constraint is based on having the right components for the project. Important components are needed for this to work right. We need to make sure that the soil, rain, pH, and humidity and temperature sensors work well and correctly to make sure that the project works properly. The biggest thing that we need to make sure works is the PCB. The PCB is one of the most important parts so it must not have defects that will cause trouble. We must have this made a reasonable amount of time in advance to ensure that it works as intended and so that it does not produce more difficulties in later parts of the design. Another thing that we need to make sure works well is the Wi-Fi capability of the parts going in the system.

3.1.6 Plant Parameters

When designing the ideal software for the project, we originally intended on having input from the user to identify the plant being watered. Through further research, we acknowledged that allowing user input could lead to more problems than benefits. In the case where the user inputs the wrong information on the plant, the outcome could lead to the entire irrigation system being flawed. Additionally, we noticed that allowing user input adds complexity to the user experience where it is not needed. It was decided that it would be better to add complexity to functions like Alexa integration to enhance the user experience. This is why we decided to have preset herbs built into the design so users can select the appropriate plant for watering.

When it comes to the pre-set design, we noticed that there is still a small possibility for error in the cases of selecting the wrong herb or picking one randomly. However, in these cases, we acknowledge that while having minor user errors may not be preferable, it is still better than the alternative where a system failure is possible.

3.1.7 Political

There are no restrictions when installing an irrigation system for personal use at home. However, there are policies concerning irrigation systems on large areas such as farms. Irrigation is a major component of sustainable agricultural development. It accounts for about 80 percent of water usage in the United States which led to concerns by the government regarding irrigation efficiencies and water conservation. This is why irrigation and water policies are implemented for more efficient water management. A permit from the government is also required when planning to use an irrigation system in large, public areas. Since our project focuses on personal use, there is no need to acquire permission from the government. Our group aims to execute a more efficient and more effective irrigation system to decrease unnecessary water consumption.

3.1.8 Product Availability

Due to the current pandemic, finding the proper materials is becoming more and more of a constraint. Manufacturers are experiencing supply shortages and long delivery times have become the new normal. As a result, we have to ensure our project is as thoroughly planned as possible before placing orders. Should an error occur, it is very likely that replacement materials may not be available and/or will not arrive in time for the expected project demonstration. Furthermore, not thoroughly planning before placing orders can lead to a final product with several missing features.

With this type of constraint, we understand that while it is something we have no control over, it is still something that still requires initiative on our part. So, when deciding on parts to select, we will also be considering the ones with the earliest arrival dates.

3.1.9 Social

Social constraints are social behaviors that influence the effectiveness of a project within a community. These constraints are mostly directed to the users that are making use of our project. One of the goals of our project is to provide users a portable, smart gardening system that is user-friendly. Some of the social constraints that we are considering during the development of Greenie include language, accessibility, and ease of use.

Language is an important part of culture. It is a way to communicate thoughts and ideas to one another. To communicate the goals of our project to the users, the language that we used on the web app and the CSS system is understandable by a lot of users

around the world. We decided to use Northern American English to convey information and data to the users. This language is used throughout the web app, the system, and the user's manual. The words are simple and easy to understand so that the users will not struggle in installing and controlling the system. As we progress, other languages like Spanish and Chinese will be considered to accommodate a wider range of users.

Since our project is accessible through a web application, an internet connection is required to access the data of the plants and to control the system without having to interact with the controller physically. Without an internet connection, the user won't be able to turn on or off the system manually by pressing the button on the controller. Additionally, the users will not have access to the daily statistics of the plants and will not be able to control the system remotely.

Greenie caters to all kinds of users from plant enthusiasts to hobbyists to individuals who just want to grow herbs at home. This means that we need to make sure that our project is functional while maintaining its simplicity. It is important that we consider the level of usability of our project so that more users will be able to utilize and benefit from using the system. To maintain its simplicity, we decided to minimize the buttons both on the plant enclosure and on the web app. The initial set-up of the system and the web app is easy to follow and easy to navigate. As mentioned above, the words that are used on the entire system are simple and understandable. By keeping all these factors in mind, we are able to provide a simple to use product at the end.

3.1.10 Sustainability and Reliability

Sustainability is about how long the product is going to last. The irrigation system should be made so that the farmers do not need to replace parts constantly. The idea is that they would need to spend less money on parts and maintenance or at least make it so that the farmers can do it themselves. By doing this, we can make it so that the irrigation system is more self-sufficient and more appealing to use. By making a long-lasting product, we can ensure that our irrigation system would be seen as trustworthy to anyone that would want to use it for their lands.

The project must also have a certain level of reliability. It needs to work as intended and do what it is supposed to do, which is to irrigate plants with little to no user input. To accomplish this, we have to look at every part while comparing and contrasting with similar products to see which one performs the best, in terms of their strengths and weaknesses.

3.1.11 Scale and Time

One of the constraints of the project is its scale. The project is being made with the idea that it can irrigate large amounts of land for farmers. In practice, however, we cannot match the scale of this because of factors like cost and the amount of time it would take to test a large area. Instead, we constrained ourselves to a smaller plot with the idea that we could implement a larger area if needed. Thus, if we focus on making a smaller

plot, making sure that it works as intended, and creating the project around that idea, we will know that it can be expanded upon to accommodate farmers with different plot sizes.

Time is a big factor in how a project can progress. There are only about five months worth of time where we can physically work on the project, including the time it takes to order parts from different areas of the world. Since we do not have a lot of time to work on this, we need to be careful about how we spend our time on the project. Many features can be included in the design of a smart irrigation system, but we need to make the project fairly quickly while also being mindful that many of the group members are taking other classes. For this reason, some of the features will have to be left out or scoped down to accommodate how much time each person can put into the project. This can be mitigated as much as possible by trying to create a flexible deadline for the project and allowing room for error, in case the unexpected happens and we cannot reach certain milestones. Doing this will allow us to make the most out of the time that we have available. Finally, we need to be sure that the project works well before the due date. We need to make sure that testing is done and completed well in advance of the due date to confirm that our project performs as intended.

3.2 Standards

In this section, we will be discussing the standards related to our project. Greenie, like every other engineering product, must meet realistic design standards. Our product must meet various standards such as wireless communication, communication protocol, and programming standards. All previously mentioned standards will be reviewed in detail in this section.

3.2.1 Wireless Communication

The microcontroller sends the data collected from the sensors to the cloud server via wireless communication. The communication between the system and the user is wireless. The type of wireless communication that we utilized is Wi-Fi. IEEE 802.11, which is more commonly known as Wi-Fi, is the set of standards established by the Institute of Electrical and Electronics Engineers (IEEE) that specifies communication for wireless local area networks. To ensure that our subsystems are able to communicate with each other accordingly so that data can be passed from the device to more suitable avenues, the following wireless standards were considered throughout the development of our project.

802.11b
802.11b is the extension of the original standard 802.11, therefore it uses the same frequency band as the original which is 2.4 GHz. It supports a maximum rate of 11 Mbps and has a range of up to 150 feet. This standard may be cheaper and the most popular in the consumer market, however, it also has the slowest maximum rate out of all the Wi-Fi standards. Devices using 802.11b may encounter interference from other devices operating in 2.4 GHz such as microwave ovens, cordless phones, and other home

appliances.

802.11a
This standard operates in the 5 GHz band and has a maximum rate of 54 Mbps. Since it operates in a high-frequency band, it is less prone to signal interference from other devices. Having a higher frequency also shortens the range of 802.11a networks. In addition, the signals have more difficulty in penetrating walls. This type of standard was commonly used in business applications.

802.11g
The 802.11g standard is a combination of the two previous standards as it operates in 2.4 GHz while having a maximum rate of 54 Mbps. This standard is backward compatible with 802.11b which means 802.11b devices can connect to 802.11g access points but at a slower data rate that matches 802.11b. Like 802.11b, devices using 802.11g may also experience interference from other devices that operate in 2.4 GHz so other options should be explored.

802.11n
The 802.11n standard supports MIMO (Multiple Input Multiple Output) and operates on both 2.4 GHz and 5 GHz. The maximum data rate that could be reached is up to 600 Mbps. Compared to the previous standards that were described, 802.11n has a better range due to the increased signal intensity. This standard is also backward compatible with 802.11a/b/g.

802.11ac
The 802.11ac is the most used Wi-Fi standard today. Most home routers are compliant with this standard. It operates in the 5 GHz band and could support up to eight antennas compared to 802.11n which can only support 4. 802.11ac is backward compatible with 802.11a/b/g/n. Since it works in the 5GHz, some vendors include 2.4 GHz technologies to support devices operating in 2.4 GHz.

802.11ax
This standard will probably replace the 802.11ac standard in the future. 802.11ax operates on both the 2.4 GHz and 5 GHz bands and can have a maximum rate of up to 9.6 Gbps. Another benefit of this standard is the increased power efficiency. This standard is designed to improve wireless internet in congested areas such as stadiums, shopping malls, and other public spaces.

3.2.2 Communication Protocol

As outlined in our requirement specifications, we will have three different communication protocols to consider when deciding on what to implement in our project. Each one has different standards to consider and each comes with its own set of advantages and disadvantages. We will pick the one that best suits our needs. The protocols are as follows:

UART
The first is the Universal Asynchronous Receiver/Transmitter (UART), which is a hardware component for asynchronous serial communication. It is full-duplex communication wherein data can be sent and received simultaneously. In addition, UART requires only two wires to transmit data. Since it is asynchronous, therefore the data rate between the devices should match. The maximum data rate of UART is about 5 Mbps.

SPI
The second is the Serial Peripheral Interface (SPI). It is a synchronous serial communication that is considered as a full-duplex communication based on master-slave architecture. Since it is synchronous, it uses separate lines to communicate between the master and the peripherals. These data lines are used for a clock signal, MOSI (Master Out, Slave In), MISO (Master In, System Out), and a signal that enables the slave. SPI has a faster data transmission rate compared to UART and I2C due to parallel transmission.

I2C
Lastly, we have the Inter-Integrated Circuit (I2C), which combines some of the features of SPI and UART. Like SPI, I2C is synchronous and supports master-slave architecture. It can either have a single master and multiple slaves, or multiple masters and a single/multiple slaves. Similar to UART, I2C only uses two bidirectional lines for serial clock and serial data. There is only one line for sending and receiving data, therefore it is a half-duplex communication. I2C utilizes the ACK/NACK functionality for better error handling.

3.2.3 Programming Standards

To create the web application and communicate with the hardware, several programming languages are utilized, and with each language comes a new set of standards to follow. Some standards are applicable to all the languages and some are language-specific. C programming is used for communicating with the hardware and a combination of JavaScript, HTML, and CSS is used to create the web application. Their breakdown is as follows:

General
- All comments are descriptive and easy for others to interpret
- Every important function has a comment
- A line does not exceed 100 characters
- Relevant and descriptive names will be used for variables, functions, etc.
- Proper spacing and indentation (i.e. no excessive gaps)

C-Specific Standards
- Variable names are camel case (ex. camelCase)
- Braces will start underneath created functions

- Block comments will only be used when needed
- Spaces will be used for conditional statements and loops
- Default returns will be 0

JavaScript-Specific Standards
- Variable and method names are camel cased (ex. camelCase)
- Class names are capitalized camel cased (ex. CapitalizedCamelCase)
- Braces will start underneath created functions
- Block comments will only be used when needed
- Spaces will be used for conditional statements and loops

HTML-Specific Standards
- All element names will be lowercase
- All elements will be closed where applicable
- Attribute values will be in quotes
- There will be separate CSS files for styling
- CSS-Specific Standards
    - One property declaration per line
    - All names will be lowercase/camel case

3.2.4 Software Testing Standards

Different standards exist for software when people are testing. These are sets of rules that are implemented as guidelines and instructions to help improve the quality needs of the software and should be adhered to. These standards were developed by many organizations around the world including the International Organization for Standardization (ISO) and the Institute of Electrical and Electronics Engineers (IEEE). This section discusses some of the standards that we will adhere to when testing the software of our project.

ISO/IEC 9126
This ISO standard addresses the quality of a software application based on the quality mode, external metrics, internal metrics, and quality standard in use metrics. This standard ensures that the software is functional, reliable, usable, efficient, maintainable, and portable.

ISO/IEC 9241-11
This standard covers the interaction between humans and computers. The usability of the software is considered based on user performance and satisfaction. It is used to verify how well the requirements are fulfilled based on users' experience. Three components describe this standard. System effectiveness refers to the ability of the users to complete a specific task. System efficiency describes the resources that the user utilized to complete a task. The last component is system satisfaction which refers to the feedback of the users.

ISO/IEC/IEEE 29119-4

This particular standard talks about software test methods or techniques that companies can use during the test design and implementation process. It is intended for software testers, developers, and everyone involved in the implementation of software testing. This standard is divided into three categories. The first category is the specification-based test design techniques which are based on the functional specification of a product. Some tests under this category include syntax testing, scenario testing, and random testing. The second category is called structure-based test design techniques which are based on the internal structure of the system. Some tests under this category include branch testing, branch condition testing, and data flow testing. The last category is the experience-based techniques which depend on the experience of real users.

IEEE 1008-1987
This standard defines possible approaches for unit testing and the expectations when performing unit testing. It specifies the criterion for identifying the completeness of the software unit testing. Activities and tasks that need to be executed while doing the unit testing are also described under this standard.

4. Research

This section will discuss the research that we have conducted so far regarding the different technologies that could be utilized in our project. While designing any product, a technology investigation must be made to create a successful initial prototype. This part of our document also aids us as a team to make sure that the requirements that we previously specified are met by the best technology that we can afford.

4.1 Existing Projects and Products

Several smart irrigation systems already exist in the market. This section will talk about existing products that have similar functions as our project. This section will also include previous Senior Design projects from UCF. These existing devices vary from smart sprinkler controllers to hydroponics. By comparing these devices, our group will be able to gain additional knowledge from the different technologies and methods that were used on these products and projects.

4.1.1 EasyHerb

EasyHerb is an automated plant growing system that was developed by a group of Electrical and Computer Engineering students from UCF in 2020. It is a hydroponic system that focuses on growing and maintaining herbs indoors. The system monitors the daily needs of herbs such as water, light, and soil nutrients. The data is collected from various sensors, including soil moisture sensors, water level sensors, pH sensors, temperature sensors, and light sensors. The data can be accessed through the LCD screen attached to the device and through a web app. Users will receive a notification

whenever the system is running out of water and soil nutrients. The whole system is contained in one device and is powered by a wall outlet.

When comparing our project to EasyHerb, we found several similarities. In terms of achieving our objectives, we are both utilizing a variety of sensors to obtain information on the quality of products. The metrics that we share in particular are soil moisture, temperature, humidity, and pH. We both also care about the experience the user has with our products. So, when it comes to the ability to allow the user to water plants remotely, both of our projects share this feature. To accomplish this, we both plan to create an interactive software application that is easy to use for the user.

When contrasting our project to EasyHerb, we also found several differences. Starting with portability. For Greenie, one of the key points we are emphasizing is our ability to use outdoor and indoor use. With EasyHerb, the main focus in their motivation is that they wanted to create a hydroponic system that works in settings like a kitchen or living room. We recognize this a great distinction because we acknowledge that the weather can be a great element when saving water in plant growth. This is why we decided to incorporate parts like our rain sensor. Another difference that Greenie has over EasyHerb is the materials being utilized. For our project, we are looking to include materials like newer model voltage regulators to improve the overall efficiency and a smart speaker with Alexa integration. By including these newer parts and additional features into our smart irrigation system, we believe that the user experience will ultimately be enhanced. For more information on similarities and differences, refer to their project located on the UCF EECS Senior Design website.

4.1.2 Smart Garden Controller

Smart Garden Controller is a UCF Senior Design project from 2018 that focuses on automated home gardening systems intended for the outdoors. The system uses a soil moisture sensor, temperature sensor, wind speed sensor, light sensor, humidity sensor, and an atmospheric pressure sensor to help determine when to water the plants and how much water to release. Irrigation will be reduced or skipped once the moisture level of the soil reaches a certain threshold. Users will be able to control the system's schedule and settings through a web application and the buttons on the device. The system is portable and is powered by rechargeable batteries.

Comparing our project to the Smart Garden Controller, we found more similarities. In terms of important factors or metrics, we both care about the following: soil moisture, temperature, humidity, and the weather. Some of which are also shared by EasyHerb. Additionally, we both care about having a project that is convenient through automation. So, when it comes to saving water, we both believe that it would be best for the plants and the user if there is an option available where an automatic watering schedule can be set in place.

Contrasting our project to the Smart Garden Controller, we go back to the portability of our project. For the Smart Garden Controller, they have emphasized having their project

outside. We plan to have Greenie capable of operating both inside and outside, either on a farm or in a garden. Another difference that Greenie has is regarding the technology used. For our project, we are looking to create a web application that utilizes HTML, CSS, JavaScript, and a web service model. More of which can be seen in the later sections. With the Smart Garden Controller, they decide to use a Tomcat server to deploy their code to. Something that utilizes a very different structure. This along with several other technology differences can be found on the UCF EECS Senior Design website.

4.1.3 Orbit B-Hyve 57950

The Orbit B-Hyve 57950 is one of the least expensive smart sprinkler controllers in the market. It is available in both six and twelve zones. The system is housed in a weatherproof enclosure and can be installed indoors or outdoors. The system can be programmed and controlled through a Wi-Fi connection with the mobile app or the web app. It can also be configured manually via the controller's LCD screen and buttons. In addition, Orbit B-Hyve is compatible with Amazon Alexa. Users can use voice commands to water all zones or a specific zone, turn the timer on and off, activate and cancel the rain delay, or ask Alexa about the past and upcoming watering schedule.

The Orbit B-Hyve utilizes technologies such as Smart Watering and WeatherSense. Smart Watering allows users to set a watering schedule based on the plant's properties. This feature uses information about the plant, soil, water and sun necessities, and sprinkler type to create an efficient watering schedule. On the other hand, WeatherSense gathers data from local weather to modify the schedule based on the past and expected rainfall. The system can also be programmed to initiate rain delays manually or based on the collected weather data.

When comparing the Orbit B-Hyve 57950 to Greenie, they have several similar aspects. Both products are smart irrigators that offer very similar benefits to their users. The two provide both indoor and outdoor utility, internet connection, weather integration, Alexa compatibility, the ability to utilize a web application to fully automate plant watering, and to set designated times for when the user would like to water. Also, both products are capable of providing care to plants by their site conditions, such as soil type, making sure that the right amount of water is delivered to plants.

Although both products have very similar characteristics, they also differ in several areas. The Orbit B-Hyve 57950 can provide watering information on a plant from their position on the ground and whether they are under the sun or the shade. This product comes in a lockable, weather-resistant cabinet that must be wall-mounted. These are areas where the Orbit B-Hyve 57950 and Greenie differ. As opposed to the competitor product, Greenie utilizes a soil sensor, a rain sensor, a pH sensor, and a humidity and temperature sensor to read and store data regarding the herb currently planted. Our product is going to be portable, lightweight, and will not require wall mounting from the user.

4.1.4 Rachio 3 Smart Sprinkler Controller

Rachio 3, which is available for eight or sixteen zones, is an easy-to-install smart sprinkler controller that connects to the Wi-Fi so users can control the system anywhere using the mobile app or the web app. The round button attached to the controller is used to manually start and stop watering specific zones. This system considers the vegetation and soil type, sun exposure, and slope of the area to determine the best possible watering schedule for the garden. Flex Daily and Flex Monthly are two scheduling options recommended on the app. Users can either choose from these schedules or they can create a manual schedule. Regardless of which type of schedule is chosen, users can enable the Weather Intelligence setting that skips or delays watering when soil is too wet, it is too windy, or when a freeze alert is detected.

The Rachio 3 controller offers advanced features including dual-band Wi-Fi, Premium Weather Intelligence Plus, and Wireless Flow Meter. Dual-band Wi-Fi allows the controller to connect to 2.4 GHz and 5 GHz wireless networks. Rachio's Premium Weather Intelligence Plus provides the most accurate watering schedule based on weather data collected from more than 300,000 weather stations around the area to deliver hyperlocal forecasts. Rachio Wireless Flow Meter is a separate device that helps detect water leaks and monitors water flow rates. It alerts the user by sending notifications when leaks and blockages are detected, and the leaking zone automatically shuts off. Rachio 3 controller also works with Amazon Alexa and Google Assistant voice commands and IFTTT Applets.

When comparing the Rachio 3 Smart Sprinkler Controller to Greenie, they have several aspects that are quite similar. Both products are smart irrigators that offer alike benefits to their users. The two provide the use of a web application to read and store plant data as well as either automating or setting timers for appropriate watering. Both products can inform the user of possible watering done by rain through the use of weather integration applications. Similar to greenie, the Rachio 3 Smart Sprinkler Controller is compatible with Alexa, allowing the user to utilize voice commands to communicate with the product.

Even though both products are pretty alike in many ways, they also differ in several areas. The Rachio 3 Smart Sprinkler Controller can create personalized watering times given the specific needs of the user's plants and/or lawn, based on plant type, soil type, and sun exposure. Greenie can perform personalized watering times, but they have to be previously created and set by the user as our product does not include the technology to create these schedules on its own. Similar to the Orbit B-Hyve 57950, the Rachio 3 Smart Sprinkler Controller comes in a lockable cabinet that must be wall-mounted, as opposed to our product which does not.

4.2 Project Hardware

This section will discuss the research that we have conducted so far regarding the different hardware technologies that could be utilized in our project. We will compare the

different hardware components that we have investigated by describing them and differentiating them specifications-wise. The following items will be discussed: LCD screen, microcontroller, power supply, sensors, relay module, and water pump.

4.2.1 LCD

In our project, the user needs to be able to visualize and/or monitor parameters, and an alphanumeric LCD is the most practical solution. This is due to their low consumption, different sizes available, and how they work with alphanumeric characters. This allows us to present to the user various information about the herb, so an LCD allows us to present this in the best way.

When investigating which screen would work best for our project, we came across a few different options. After careful consideration of price and what products were available, the top three Arduino-compatible screens we considered were the LCD-013-420, the LCD1602, and the 0.96" OLED Module by DIYmall. For a more detailed comparison between the three previously mentioned screens, refer to **Table 2**. As seen in the table above, both LCD screens are almost identical. These LCDs only differ in their cost and text dimensions. The LCD-013-420 can display four lines with up to twenty characters each while the LCD1602 can display two lines with up to sixteen characters each. The DIYmall 0.96” OLED Module is the most different, as it is a 128 x 64 OLED display.

Attribute	**LCD-013-420**	**LCD1602**	**0.96" OLED Module**
Communication	I2C	I2C	I2C
Text Dimensions	4 x 20	2 x 16	128 x 64 OLED
Text Color	White	White	White
Address	0x27	0x27	0x3C
Cost	$6.99	$10.99 for two (2)	$9.99

Table 2: Different LCD Screens Specification Comparison

The LCD screens in **Table 2** were the top contenders we considered when selecting an LCD for our project. While researching, we also came across the HiLetgo 20x4 LCD Display and the Coolwell Technology 2inch LCD. These screens, although promising, were not selected as we found ones that fit our project better. Below, we are going to briefly describe the two scrapped LCD screens.

HiLetgo 20x4 LCD Display
This LCD has four lines with twenty characters that can be controlled by jumpers or the program. It looks very similar to the 1602 version since it also uses the same amount of

pin definitions and also has the yellow backlight. This module uses two IO ports instead of the 1602s seven I/O ports and also has a potentiometer that can be adjusted to control the contrast of the screen. The LCD needs a voltage of 5 V to operate. The size of this is 4.8 x 3 x 0.5 inches, making it a bit on the larger side because of the four lines that are used.

Coolwell Technology 2inch LCD
This is an IPS screen. The main draw of this module is the ability to reproduce color. The operating voltage of this LCD is 3.3 V and it has a 2-inch screen. This LCD is on the expensive side, mostly because of the ability to reproduce color. However, there is only one color that could be displayed, with various online reviews stating they are easy to set up and just as many saying it is difficult to do. This LCD also has the most IO ports at eight, which makes it less desirable to use.

4.2.2 Microcontroller

A microcontroller is a device that is required for our project and many like it as we need to utilize a device that can be programmed to carry out the specific actions and/or instructions that we want our system to do. The design of our system is heavily dependent on this control unit, given that our project requires the use of several hardware components, such as an LCD screen, water pump, a relay module, and several different sensors. A microcontroller unit, or MCU, is crucial to the seamless integration of these parts into one system because of their need to interact with one another and transfer information while performing different actions because of this information.

As observed in **Figure 2**, the MCU will serve as the “brain” of our hardware setup, given that not only are all three soil moisture sensors, the rain sensor, the pH sensor, and the humidity and temperature sensor connected to it, but also the relay module, the LCD screen, the water pump, and the Wi-Fi module with our software, are going to be connected to it as well. The Wi-Fi module is going to ensure that our microcontroller is constantly connected to the internet and can access our database. By having a microcontroller with an internet connection, we will be able to remotely control the parts needed for our various features. This is crucial since the irrigation process has to work as intended.

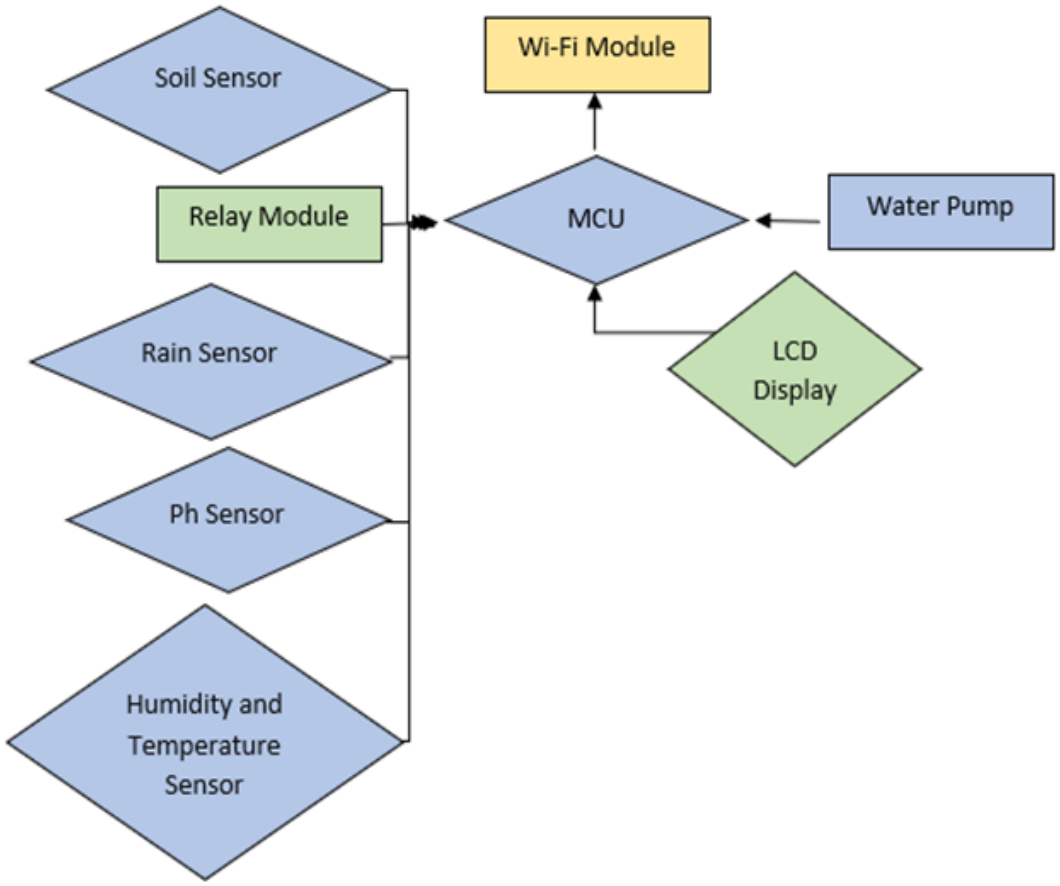


Figure 2: MCU Hardware Connections

When investigating which microcontroller would work best for our project, we came across several different options. The top three microcontrollers we considered were the Atmel ATmega328P, the Texas Instruments ARM Cortex-M3, and the Texas Instruments MSP430 LaunchPad. For a more detailed comparison between the three previously mentioned microcontrollers, refer to **Table 3**.

Atmel ATmega328P

The Atmel ATmega328P is Atmel's high-performance, low-power, and advanced architecture microcontroller from the megaAVR family optimized for C/C++ compilers. It is most commonly included in the Arduino Uno board. The Arduino Uno board is based on an open hardware design, so users can simply remove the ATmega328P from the board and apply it to other electronics. The Microsoft Visual Studio integrated development IDE allows, through an installation package, the code for Arduino to be edited. The Atmel ATmega328P is compatible with three different communication protocols, which are the Universal Asynchronous Receiver/Transmitter (UART), the Serial Peripheral Interface (SPI), and the Inter-Integrated Circuit (I2C).

Texas Instruments ARM Cortex-M3
The Texas Instruments ARM Cortex-M3 is a product of the ARM Cortex-M processor family developed by Texas Instruments. It is based on low-cost, energy-saving ARM processor designs. The ARM Cortex-M3 is capable of performing general data processing, including hardware division instructions. It also includes memory access instructions that support 8-bit, 16-bit, 32-bit, and 64-bit data and instructions for transferring various 32-bit data. The Texas Instruments ARM Cortex-M3 is compatible with three different communication protocols, which are the Serial Communications Interface (SCI), the Serial Peripheral Interface (SPI), and the Inter-Integrated Circuit (I2C).

Texas Instruments MSP430 LaunchPad
The Texas Instruments MSP430 LaunchPad is Texas Instruments' beginner-friendly, ultra-low power kit that allows you to program and debug the Texas MSP430 series of microcontrollers through the USB interface. It is known for its very low electrical consumption and for being ideal for battery-powered devices. It uses the same programming interface and the same programming language as Arduino, the same basic examples and various Arduino libraries are present. The MSP430 gives the user the possibility of expanding peripherals through additional boards, known as BoosterPacks. This microcontroller was the first one considered as our team has the most experience utilizing this board. We programmed it using C language on the Code Composer Studio integrated development IDE in our embedded Systems course. The Texas Instruments MSP430 LaunchPad is compatible with three different communication protocols, which are the Universal Asynchronous Receiver/Transmitter (UART), the Serial Peripheral Interface (SPI), and the Inter-Integrated Circuit (I2C).

Attribute	**Atmel ATmega328P**	**Texas Instruments ARM Cortex-M3**	**Texas Instruments MSP430 LaunchPad**
Communication Protocol	UART SPI I2C	SCI SPI I2C	UART SPI I2C
Memory	32 KB	256 KB	128 KB
Maximum I/O Pins	23	52	83
Maximum Operating Frequency	20 MHz	50 MHz	16 MHz
Operating Voltage	5 V	2.5 V	3.3 V

Table 3: Different Microcontrollers Specification Comparison

4.2.3 Power Supply

The power supply allows the whole system to run. We will be using wall power to run the irrigation system during the day. To properly utilize the wall outlet as a power source, we must use a voltage converter since the United States provides alternating current (AC) as its power source and we are going to utilize direct current (DC). More research on these topics was performed and recorded in section **4.2.7 Voltage Regulator**. We require a 12 V voltage regulator so that we can provide power to the water pump and both 5 V and 3 V voltage regulators so that we can power all of the other parts of the system. The flow of power in the power supply is shown in **Figure 3**. At night, the Solar Power Bank will be used to power the irrigation system. This was decided to cut costs in the long term.

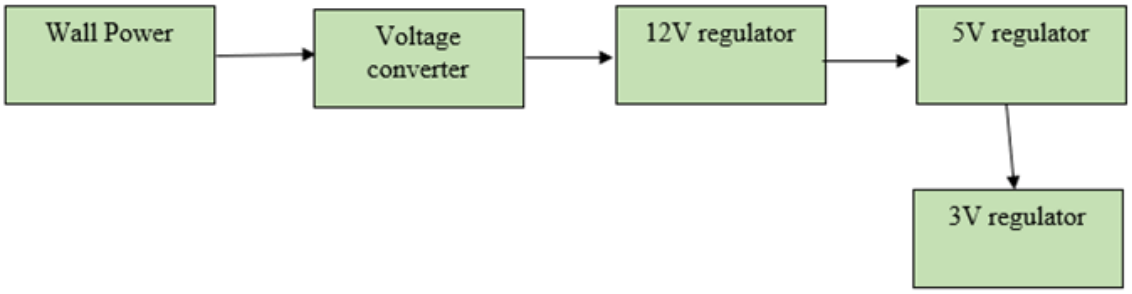


Figure 3: Power Supply Diagram

Part	**Hours Working**	**Rating (mA)**	**Total mAh**
LCD1602 Module	12	1.1	13.2
Microcontroller	12	200	2,400
Soil Sensor	12	5	60
Rain Sensor	12	15	180
H&T Sensor	12	2.5	30
Relay Module	12	5	60
pH Sensor	12	10	120
Speakers	12	2	24
Water Pump	12	375	4500
Wi-Fi Module	12	80	960
Total			8,647.2

Table 4: Calculation of mAh

The information in **Table 4** was used to determine how many mAH will be used during the night. The time period of a twelve-hour worst-case scenario was used to prepare for a situation where the components need to be constantly running. It was found that during the night, a total of 8515.2mAh is used, which is under the 10,000mAh of the battery pack. This means that the Solar Power Bank can keep the irrigation system powered.

When investigating which power supply would work best for our project, we came across a few different options. The top three power banks we considered were the 10,000mAh Solar Power Bank and Solar Panel Charger, the 20,000mAh Solar Power Bank and Solar Panel Charger, and the Solar Power Bank 26800mAh by Riapow. We also researched the 12V 15Ah Mighty Max Battery. Below, the three power supplies and the battery are going to be discussed.

10,000mAh Solar Power Bank and Solar Panel Charger
The 10,000mAh Solar Power Bank and Solar Panel Charger allows for the usage of 10,000 mAh by the irrigation system. There is also a solar panel that allows for the use of charging and gaining power back during the day. However, many of the reviews on the product note that the solar charging that occurs is something that should be used in an emergency. The voltage range of the power bank is around 9 to 12 V.

20,000mAh Solar Power Bank and Solar Panel Charger
The 20,000 mAh version of the Solar Power Bank comes with more solar panels to help with the charging aspect. Where the 10,000mAh version has one panel, this power bank comes with 6 panels that can be laid out. This device provides a voltage of 9 to 12 V. This device is much heavier and larger than its lower 10,000 mAh version and is more expensive.

Riapow Solar Power Bank 26,800mAh
The Riapow Solar Power Bank supplies about 26,800 mAh and delivers up to 8 days of power on a single charge. Like the two solar banks mentioned above, this one also includes a small solar panel that allows the battery to charge itself during the day. It has a voltage rating of 5 V.

12V 15Ah Mighty Max Battery
The 12V 15Ah Mighty Max Battery has a nominal voltage of 12 V. One important thing to notice is that the battery has reduced capacity starting at around 32 °F. Capacity also reduces after a set amount of time has passed, at about three months there is 90% capacity of the regular size. This battery is rechargeable and is spillproof with a high discharge rate. This battery is also resistant to shocks and vibrations allowing it to be used in more areas.

Although we wanted to implement a second source of power in our system, due to time constraints, we unfortunately had to cut this feature. We decided to just go with AC/DC and DC/DC power for our system.

4.2.4 Sensors

Sensors play one of the most important roles in our project. Through the use of sensors, the smart irrigation system is going to be able to receive data regarding the herb that is currently planted, allowing the user to quickly observe crucial information about the soil's state and future irrigation needs while also allowing the system to make decisions based on what information is acquired. We have decided to utilize three soil moisture sensors, a rain sensor, and a humidity and temperature sensor for our project.

To measure the moisture in the planter's soil, our system requires a soil moisture sensor. The data acquired from this sensor is crucial given that this sensor is going to inform the user if the soil has received enough water on any specific day. When investigating which soil moisture sensor would work best for our project, we came across a few different options. The top three soil moisture sensors we considered were the SparkFun Soil Moisture Sensor, the Adafruit STEMMA Soil Sensor, and the KeeYees LM393. In **Table 5**, the differences between these models can be observed. All three are similar, being different in their output type and the methods utilized to take measurements.

Attribute	**SparkFun Soil Moisture Sensor**	**Adafruit STEMMA Soil Sensor**	**KeeYees LM393**
Arduino Compatibility	Yes	Yes	Yes
Operating Voltage	3.3 - 5 V	3 - 5 V	3.3 - 5 V
Output Type	Analog	Analog	Analog and Digital
Measurement Method	Resistivity Measurement	Capacitive Measurement	Resistivity Measurement
Number of Prongs	2	1	2
Cost	$5.95 for one (1)	$7.50 for one (1)	$7.99 for five (5)

Table 5: Different Soil Sensors Specification Comparison

Our system requires a rain sensor that is going to allow the system to detect and respond to drops of moisture by either switching the water pump on or off, depending on if the plant has been watered or not by rain. Two models are widely used in irrigation systems today, the FC-37 and the MH-RD. The FC-37 is most commonly found. Similar to the soil moisture sensor, these two models measure values from 0 for a fully soaked plate, to 1023 for a fully dry plate. In **Table 6**, the differences between these models can be observed. They are very similar, their dimensions and weight slightly vary, but they work quite similarly.

Attribute	**FC-37**	**FC-37**	**MH-RD**
Manufacturer	ACROBOTIC	HiLetgo	Teyleten Robot
Operating Voltage	3 - 5 V	3 - 5 V	3.3 - 5 V
Output Current	15 mA	15 mA	15 mA
Type of Output	Analog and Digital	Analog and Digital	Analog and Digital
Cost	$8.99 for one (1)	$5.99 for three (3)	$5.88 for three (3)

Table 6: Rain Sensor FC-37 versus MH-RD

To measure the humidity and temperature in the air, our system requires a humidity and temperature sensor. Through this sensor, the user is going to receive data regarding the ideal air humidity and temperature needed from the room to optimize their plant's needs. When investigating which humidity and temperature sensor would work best for our project, we came across a few different options. The top three humidity and temperature sensors we considered were the RHT03 (also known as DHT22), the DHT11, and the AM2302. In **Table 7**, the differences between these models can be observed. As seen in the table, all three models are relatively similar but different in terms of their temperature range and cost.

Attribute	**RHT03**	**DHT11**	**AM2302**
Operating Voltage	3.3 - 6 V	3 - 5 V	3 - 5 V
Maximum Current	1 - 1.5 mA	2.5 mA	2.5 mA
Humidity Range	0 - 100% RH	20 - 80% RH	0 - 100% RH
Temperature Range	40 - 80 °C	0 - 50 °C	-40 - 80 °C
Measurement Accuracy	± 2% RH Accuracy ± 0.5 °C Accuracy	± 5% RH Accuracy ± 2 °C Accuracy	± 2% RH Accuracy ± 0.5 °C Accuracy
Cost	$12.95	$8.88 for five (5)	$18.49 for four (4)

Table 7: Different Humidity and Temperature Sensors Specification Comparison

To maintain the overall health of the user's herb, our system requires a pH sensor. The pH is a measure of acidity or alkalinity of a solution and there exists something that is called a pH scale which varies from 0 to 14, with 0 being the most acidic and 14 being the least acidic. A value of 7 is neutral. The correct soil pH is essential to ensure optimal plant growth and crop yield, as it allows nutrients to be freely available for plants to absorb. Each plant will have different pH needs, so this information will be needed to be

taken into account. An inappropriate pH range directly affects the capacity of the root system, because if the pH values are extreme, it can lead to precipitation of certain nutrients, making them no longer available. Our pH sensor is going to read the pH level of the soil to maintain the health of the user's plant.

When investigating which pH sensor would work best for our project, we came across a few different options. The top three pH sensors we considered were the DONGKER pH Sensor Module, the GAOHOU PH0-14 Sensor Module, and the BOOTOP PH0-14 Sensor Module. In **Table 8**, the differences between these models can be observed. The three pH sensors are extremely similar, being different mostly on their measuring temperature and their zero point, which covers their accuracy.

Attribute	**DONGKER pH Sensor Module**	**GAOHOU PH0-14 Sensor Module**	**BOOTOP PH0-14 Sensor Module**
Operating Voltage	5 V	5 V	5 V
Working Current	5 - 10 mA	5 - 10 mA	5 - 10 mA
Measuring Range	0 - 14 pH	0 - 14 pH	0 - 14 pH
Measuring Temperature	0 - 60 ℃	0 - 80 ℃	0 - 80 ℃
Zero Point	7 ± 0.01 pH	7 ± 0.25 pH	7 ± 0.25 pH
Cost	$36.99	$35.59	$35.19

Table 8: Different pH Sensors Specification Comparison

Another sensor that is helpful is a piezoelectric vibration sensor. This sensor is used as a flow sensor that would detect the flow of water. The main purpose of this sensor is to let the user know if there is a leak in the system, allowing the user to fix the leak and not waste water. When water is flowing through the tube, it would send a signal to let the user know that water is in fact flowing through that part of the tube. If it was not, it would indicate that there would be a leak in the system. This is done so that the user stops sending water through the system if none of the vibrations are detected. When investigating which piezoelectric vibration sensor would work best for our project, we came across a few different options. The top three piezoelectric vibration sensors we considered were the HiLetgo Analog Ceramic Vib Sensor, the HiLetgo Vibration Sensor Module, and the HiLetgo 801S Vibration Sensor Module. In **Table 9**, the differences between these models can be observed. The three piezoelectric vibration sensors are quite different, being only similar in terms of operating temperature and their cost, as all three are below $10. Although we wanted to implement the piezoelectric vibration sensor, due to time constraints, we unfortunately had to cut this feature.

Attribute	HiLetgo Analog Ceramic Vib Sensor	HiLetgo Vibration Sensor Module	HiLetgo 801S Vibration Sensor Module
Operating Voltage	3.3 - 5 V	5 V	3 - 5 V
Working Current	< 1 mA	~15 mA	5 - 10 mA
Interface Type	Analog Signal output	Digital Switching Output	Analog output signal
Operating Temperature	-10 - 70 ℃	0 - 80 ℃	0 - 70 ℃
Cost	$8.49	$5.99	$7.75

Table 9: Different Piezoelectric Vibration Sensors Specification Comparison

4.2.5 Relay Module

Our system automatically turns on and off the system and pumps whenever they are not being utilized. To fully implement the turn on and turn off functions of our project we need to utilize a relay module (refer to **Figure 9**). A relay module is required for this implementation since it must be utilized whenever an application needs to switch from high to low or vice versa within the same circuit. The relay is going to be used as a mechanical switch, which will be electrically operated, that can be turned on or off, allowing current to pass or not.

When investigating which relay module would work best for our project, we came across a few different options. The top three relay modules we considered were the HiLetgo 5V, the WINGONEER KY-019 5V, and the KeeYees 5V Relay Module. For a more detailed comparison between the two previously mentioned relay modules, refer to **Table 10**. As seen in the table, the three relay modules are only different in their cost.

Attribute	HiLetgo 5V	WINGONEER KY-019 5V	KeeYees 5V Relay Module
Operating Voltage	5 V	5 V	5 V
Maximum AC	AC 250 V / 10 A	AC 250 V / 10 A	AC 250 V / 10 A
Maximum DC	DC 30 V / 10 A	DC 30 V / 10 A	DC 30 V / 10 A
Cost	$5.98 for two (2)	$8.49 for five (5)	$9.99 for five (5)

Table 10: Different Relay Modules Specification Comparison

4.2.6 Water Pump

A distinctive feature of our project is that it requires a water pump to irrigate the plant. The water pump is essential for efficient water use and distribution. It is going to move the water between the container and the planter, becoming especially necessary since the resource needs to be transported from a container located below, which it could not otherwise reach due to the lack of pressure. For our project, we require our water pump to be submergible and to include both inlet and outlet pumps. Many brands on the market sell this type of product. In **Table 11**, the differences between three different water pump models that we are considering can be observed. Although their dimensions, current, and rated power vary slightly, these three pumps are quite similar.

Attribute	**LEDGLE**	**Mavel Star**	**MOUNTAIN_ARK**
Rated Voltage	DC 12 V	DC 12 V	DC 12 V
Rated Power	3.6 W	4.8 W	4.5 W
Lift	3 M / 9.8 ft	3 M / 9.8 ft	3 M / 9.8 ft
Flow Rate	240L/H 63.4GPH	240L/H 63.4GPH	240L/H 63.4GPH
Current	300 mA	350 mA	400 mA
Cost	$8.99	$12.99	$10.99

Table 11: Different Water Pump Specifications Comparison

4.2.7 Nutrient Pump

Similar to the water pump, a nutrient pump is also essential for the proper growth and maintenance of the herbs supported by the system. Whether ornamental or horticultural, all plants need to obtain minerals and other elements to grow properly. If they lack nutrition, the health of the plant is at risk. Therefore, it is essential to guarantee an adequate supply of nutrients for the herbs.

To accomplish this task, the system contains a nutrient pump, specifically the Gikfun 12 V DC Dosing Pump. This nutrient pump boasts a “Snap-in” type design through which it is easy to remove the pump head, making it very convenient for pump tube replacement and cleaning. The Dosing Pump has a voltage of 12 V DC, a current of 80 mA, and a flow rate of 0 to 100 milliliters per minute (ml/min). It also has a small pump head size with a diameter of 31.7 mm, making it ideal for precisely pumping small amounts of nutrient solution.

4.2.8 Voltage Regulator

The voltage regulator is an important part of the project. This component will allow the system to use the correct voltages for all of the different pieces of technology that exist in our product. Voltage regulators also protect from voltage spikes. It is imperative to protect electric equipment from voltage spikes as they can be dangerous. These can occur, for example, because lightning strikes nearby. No electrical network is exempt from these spikes that can reach thousands of volts. However, most of a regulator's power goes into trying to stabilize voltage fluctuations, not stopping voltage spikes.

To properly utilize a voltage regulator, research on converting alternating current to direct current along with direct current to direct current power conversion was performed so that we can give an accurate description of how this will affect the various components on our project. We also investigated older and newer technologies in the voltage regulator field as these have evolved over the recent years. These topics are going to be discussed in detail in the sections below.

4.2.8.1 AC/DC Power Conversion

If we are going to make use of the power from transmission lines, we need to make sure that the alternating current power from the outlet can be used as direct current power for the devices that we need like the pump and MCU, thus AC/DC power conversion is going to be required to power our product. AC to DC power converters, or adaptors, are used to operate small items that work with direct current. These adapters convert the high-voltage AC we draw from the wall outlet to low-voltage DC for the products we are powering, such as phones, laptops, and small appliances. AC power switches its direction back and forth at a rapid velocity, flowing in both directions along a power line. This bi-directional flow could damage DC equipment, which can only handle power flowing in one direction, hence the need for an AC/DC adapter.

The standard in the United States is 120 V and 60 Hz AC electricity[4], therefore, we must use a voltage converter since we are going to be utilizing DC electricity for our product. To properly power any device, the rated output voltage of the adaptor should match the rated input voltage of the electric components it is going to power. Using a power adapter rated for a higher amperage than the electronic equipment it is going to power is safe, as the equipment will only draw the necessary current. We require a 9 to 12 V regulator to provide power to the system and its parts, so we must select a power adapter that matches the voltage rating or is higher. Failure to do so will result in the adaptor not being able to properly power our system.

For an AC/DC converter, three main parts need to be taken into account. When the input comes in, we need a transformer, a rectifier, and then an output filter to get the desired DC output. The transformer usually comes in a large size, it would also usually produce a lot of heat, but our device does not require a large amount of power so it will not pose an issue.

When investigating which AC/DC power adapter would work best for our project, we came across a few different options. The top three adapters we considered were the Corporate Computer Power Supply Adapter, the SmoTecQ Store Power Supply Adapter, and the TMEZON Power Adapter Supply. In **Table 12**, the differences between these models can be observed.

As seen in the table below, all three models are similar in terms of their connector dimensions, as all three are 5.5 mm x 2.1 mm wide, and in their polarity, given that the center or tip is positive and the sleeve is negative. The Corporate Computer Power Supply Adapter varies the most, as its input voltage, output voltage, and current rating are smaller than its two other competitors. In terms of price however, it is the cheapest one between the three options we considered. The SmoTecQ Store Power Supply Adapter and the TMEZON Power Adapter Supply are almost identical, being only different in terms of their price and the amount of pieces included.

Attribute	**Corporate Computer Power Supply Adapter**	**SmoTecQ Store Power Supply Adapter**	**TMEZON Power Adapter Supply**
Input Voltage	110 V AC	240 V	240 V
Output Voltage	9 V DC	12 V DC	12 V DC
Current Rating	1 A	2 A	2 A
Polarity	Positive center/tip, negative sleeve	Positive center/tip, negative sleeve	Positive center/tip, negative sleeve
Connector Dimensions	5.5 mm x 2.1 mm	5.5 mm x 2.1 mm	5.5 mm x 2.1 mm
Cost	$6.99 for one (1)	$11.99 for two (2)	$7.99 for one (1)

Table 12: Different AC/DC Power Adapter Specifications Comparison

4.2.8.2 DC/DC Power Conversion

DC/DC power conversion is necessary for our project. DC to DC converters are circuits capable of transforming voltage levels into others, temporarily storing energy, and discharging it in such a way that the final voltage levels are those that are desired. This transformation is done through the use of inductors, transformers, and capacitors. Given that the AC/DC power adapter we are going to utilize is going to output a voltage of 12 V DC, this value must be converted to different voltages, as different electronic components in our project require different voltage values. The voltage values required for the other parts are around 3 V and 5 V.

For this purpose, we are going to be utilizing voltage regulators, circuits that perform the voltage-switching operation. There are many types of voltage regulators depending on the need. The most common are the buck converter, the boost converter, the buck-boost converter, and the SEPIC converter. In the buck converter, the output voltage is less than that of the input due to it stepping down the voltage. These converters are most commonly used in self-regulating power supplies as well as in communication systems. As opposed to the buck converter, in the boost converter, the output voltage is higher than that of the input due to it stepping up the voltage. These converters are most commonly used in regulated power supplies and portable device applications. The buck-boost converter can step either up or down a voltage, resulting in a voltage that is either lower, equal, or higher than the input voltage. This converter could be used to supply a 12 V output from a 12 V battery whose voltage can range from 10 V to 14.7 V[5]. Like the buck-boost converter, the SEPIC converter can step a voltage up or down, producing the same result. The SEPIC converter differentiates itself from the buck-boost converter by producing a non-inverted output, meaning that the output has the same voltage polarity as the input.

Given that we must step down our 12 V power adapter to both 3 V and 5 V for our electronic components, we are going to utilize a buck converter. A buck-boost converter could also be utilized but its boost ability would not be utilized as it is not needed.

4.2.8.3 Older Regulator Technology

Voltage regulators have been around for years given that they have been utilized throughout history to ensure that electric devices receive the correct voltage supply. As we were investigating which voltage regulator to utilize in our project, we came across different types of technologies. The voltage regulators that stood out to us the most were the L7805 and the LM317T.

The “78” series of voltage regulators indicates that it is a positive regulator, as opposed to the “79” series, for example, that regulates negatively. The XX after “78” tells what voltage it will regulate. These voltage regulators are most commonly utilized in projects with Arduino programming due to their practicality and simplicity. The L78XX series is used to guarantee a constant voltage source, which reduces the possibility of damaging circuits due to fluctuations in voltage levels. This series of voltage regulators has three pins. The first pin is used for the input voltage, the second pin is used as the ground, and the third pin is used for the output voltage. The L7805 is a voltage regulator that can control a positive voltage of 5 V and 1 A current. This regulator is capable of producing an output voltage of 5 V to 18 V from an input voltage of 35 V and an output voltage of 20 V to 24 V from an input voltage of 40 V. Its output current and power dissipation are internally limited and it boasts an operating junction temperature range of -55 °C to 150 °C.

The LM317T voltage regulator is one of the most used regulators for being variable. This regulator has different configurations or circuits according to the application. Its variability is controlled by different arrangements or resistors. Similar to the L7805, the

LM317T is a positive-voltage regulator that has three pins. The first pin is used for the input voltage, the second pin is used as the ground, and the third pin is used for the output voltage. These pins connect to the various configuration circuits. The LM317T is a voltage regulator that has an output voltage range adjustable from 1.25 V to 37 V with an input voltage range of 3 V to 40 V. It boasts an output current greater than 1.5 and an operating virtual junction temperature of 150 °C. The LM317T also includes current limiting, thermal overload protection, and safe operating area protection[6]. The LM317T circuit can be supplied with an input voltage that is at least 3 V different from the desired output voltage. That is, if we want to regulate from 3 V to 5 V, a minimum 8 V input source is required. In our case, our input source is going to be 12 V.

When investigating which L7805 and LM317T packages to utilize, we came across the Bojack Voltage Regulators, the Valefod Voltage Regulator, and the Yiwanson Voltage Regulator.

Bojack Voltage Regulators
An important part of the power supply is the voltage regulators. Since we will be drawing upon the local power to use Greenie, we must have voltage regulators that can make it so that the correct voltage can be transferred into the device. Our device uses components that require 12 V, 5 V, and 3 V, so we will need the necessary regulators. We have the BOJACK series which is a trusted voltage regulator that can be used.

Valefod Voltage Regulator
There is also the Valefod voltage regulator. This voltage regulator can be changed from 3 V to 40 V by using a screwdriver to adjust the value of the built-in potentiometer. The maximum current is 3 A and is a small size at 45 mm * 23 mm * 14 mm. The converter has high Q inductance and has solid-state capacitors that are meant to filter out high-frequency noise. These pieces are priced very well too.

Yiwanson Voltage Regulator
The Yiwanson Voltage Regulator promises short circuit protection and thermal shutdown protection so that the equipment does not get damaged from being used. These voltage regulators go from 5 V to 37 V depending on the device used, coming in at a very small size as well. The current of these voltage regulators is at about 1 A to 1.2 A also depending on which device is being used.

4.2.8.4 Newer Regulator Technology

Although the L7805 and LM317T are widely popular voltage regulators and are largely utilized in many projects like ours, they are still a bit outdated. Given their age, newer projects are discouraged from utilizing them. In hopes of providing Greenie users a more advanced experience, we also decided to research newer technologies in the voltage regulator area. The voltage regulators that stood out to us the most were the LM2576 and the LM2596.

The LM2576xx is a series of popular step-down (buck converter) switching voltage regulators. These regulators are available in the market with fixed output voltages of 3.3 V, 5 V, 12 V, 15 V, and an adjustable output version[8]. This series is simple to use, includes fault protection, and is also commonly utilized today to replace popular three-pin linear voltage regulators such as the L7805. The LM2576 regulator has five pins. The first pin is used for the input voltage, the second pin is used for the output, the third pin is used as the ground, the fourth pin is used as the feedback, and the fifth pin is used as the on/off. The maximum supply voltage for this regulator is 45 V and it has a specified 3 A output current. Its power dissipation is internally limited and its maximum junction temperature is 150 °C.

Similar to the LM2576, the LM2596 is a step-down (buck converter) switching voltage regulator. This regulator also has a five-pin design, with its pins being used for the same purposes as the LM2576. Today, this regulator is used over designs with three pins given its high efficiency. It is capable of driving a 3 A load with excellent line and load regulation[9]. This regulator comes in an adjustable output voltage range of 1.23 V to 37 V. The LM2596 has a maximum supply voltage of 45 V and its power dissipation is internally limited. This regulator boasts an operating junction temperature range of -40 °C to 125 °C.

When investigating which LM2576 and LM2596 packages to utilize, we came across the D-FLIFE 5 V 16-Channel LM2576 Power Relay and the Zixtec LM2596 DC-DC Buck Converter Step Down Module Power Supply. Both are discussed briefly below.

D-FLIFE 5 V 16-Channel LM2576 Power Relay

The D-FLIFE LM2576 Power Relay boasts 16-channels of 5 V active low where each one requires a driver current of 15 to 20 mA. It is equipped with a high-current relay of AC 250 V / 10 A and DC 30 V / 12 A. To work properly, the control signal must be within the voltage range of 0 to 2 V for the trigger signal and the high voltage must not exceed 5 V. The D-FLIFE LM2576 module is a 16-channel relay interface board that is supported and can be controlled by multiple microcontrollers such as Arduino, 8051, AVR, PIC, DSP, ARM, ARM, MSP433, and TTL logic. Given that this package can control several different types of equipment with large current values, it is commonly used in projects regarding microcontroller control, Programmable Logic Controller (PLC) control, and smart home control. The module also includes LED lights to indicate the work status. The D-FLIFE 5 V 16-Channel LM2576 Power Relay comes at $15.69 on Amazon for 1 piece.

Zixtec LM2596 DC-DC Buck Converter Step Down Module Power Supply

The Zixtec LM2596 Module boasts an input voltage range of 3.2 V to 35 V DC. For all LM2596 packages, it is imperative that the input voltage is higher than the output voltage by at least 1.5 V to prevent failures such as the system not being boosted. The output voltage ranges from 1.25 V to 30 V DC. The voltage is adjustable as it has a clockwise boost and a counterclockwise buck. The Zixtec LM2596 Module has high efficiency and a maximum output current of 3 A although it is recommended to use under 2 A of current. All solid capacitors in the module use SANYO technology and the

circuit boards are on the thicker side, at 36u. It also includes High-Q inductors with a LED indicator light for when high power output occurs. The module also comes with built-in functions such as over-temperature protection function, current limit function, and output short-circuit protection function. The Zixtec LM2596 DC-DC Buck Converter Step Down Module Power Supply comes at $13.89 on Amazon for 10 pieces.

We also researched individual LM2576 and LM2596 from different manufacturers, as we may not need packages for our project. The most popular pieces were from Mouser Electronics, one of the most popular online suppliers of electronics components. Both pieces are briefly described below.

LM2576-12WU
The LM2576-12WU is described as a step-down (buck) switching voltage SMPS regulator with an output current of 3 A. The product is manufactured by Microchip and it has a mounting style of SMD/SMT. It has support for one output at a time, an input voltage range of 4 V to 40 V, an output voltage range of 1.23 V to 37 V, a switching frequency of 52 kHz, and an operating temperature ranging from -40 °C to 125 °C. The LM2576-12WU comes at $2.39 on Mouser Electronics for 1 piece, $50.00 for 25 pieces, and $182.00 for 100 pieces.

LM2596S-5.0/NOPB
The LM2596S-5.0/NOPB is described as a step-down (buck) switching voltage regulator with an output current of 3 A. The product is manufactured by Texas Instruments and it has a mounting style of SMD/SMT. It has support for one output at a time, an input voltage range of 4.5 V to 40 V, an output voltage of 5 V, a switching frequency of 150 kHz, and an operating temperature ranging from -40 °C to 125 °C. The LM2596S-5.0/NOPB comes at $7.15 on Mouser Electronics for 1 piece, $64.60 for 10 pieces, and $154.00 for 25 pieces.

4.2.9 Smart Speaker

Smart speakers that listen to and obey a user's orders through the use of voice commands are increasingly common in our homes. They have become personal assistants and will do everything the user asks, from playing music or popular radio stations, giving weather predictions, turning lights on and/or off, and even creating and/or managing calendar events. Most smart speakers today are compact, intelligent, relatively affordable, and boast decent sound quality.

The inclusion of a smart speaker is an important part of our project. The user must be able to utilize voice commands to activate or deactivate certain actions regarding the product and how it interacts with the user's plant. Through the use of our software, the user can ask the smart speaker to water their plant automatically and also present/read out loud information regarding the plant. The user will also be able to ask the system to either select automatic watering or water their plants manually through the use of their voice. Additional voice commands will also work with our LCD screen, where the user

can observe data such as soil moisture levels, whether the plant was watered by rain or not, temperature, humidity, and pH information.

When investigating which smart speaker would work best for our project, we came across a few different options. The top three smart speakers currently in the market were considered. These are the Apple HomePod Mini, the Amazon Echo Dot, and the Google Nest Mini. For a more detailed comparison between the three previously mentioned smart speakers, refer to **Table 13**.

Apple HomePod Mini
The Apple HomePod Mini is Apple's smart speaker. This speaker is considered to be the best smart speaker for many users even though it is higher in price than its competitors. The HomePod Mini is a smart speaker exclusively made for Apple users, meaning that it can only be used with other technology within Apple's ecosystem, such as the iPhone, the iPad, or Mac. It works with the Siri voice assistant, which although is not as sophisticated as Alexa or Google Assistant, is capable of doing a good job for basic commands. Although Apple's smart speaker lacks flexibility, it possesses great security and privacy, an aspect that is important for many users today. Nothing said to the speaker will be shared without the user's consent and none of the commands are linked to the user's Apple ID.

Amazon Echo Dot
The Amazon Echo is Amazon's smart speaker. It is part of the wide family of speakers made by Amazon whose voice assistance is Alexa. Thanks to years of updates, Alexa has become one of the most intelligent smart speakers, adding more abilities and ways to interact with the user. Amazon's smart speaker is compatible with the largest number of devices, making it one of the best options for controlling different types of technology gadgets. The Echo Dot can be integrated with Zigbee, a wireless technology based on the IEEE 802.15.4 personal area network standard used for creating personal area networks. Zigbee deals with the unique requirements of low-power, low-cost wireless IoT networks.

Google Nest Mini
The Google Nest Mini is Google's smart speaker. It is part of the growing family of speakers made by Google whose voice assistant is Google Assistant. Google Assistant is one of the most natural voice assistants today, matching Amazon's Alexa in terms of intelligence. It has advantages that focus on integration with Google applications, which are commonly used today, such as Google Calendar, Google Docs, Gmail, etc. The Nest Mini boasts a machine learning chip that recognizes and learns the user's most used commands, being able to respond quickly to these. This technology also allows the Nest Mini to respond to the user's most used commands without needing to access the cloud.

Attributes	Apple HomePod Mini	Amazon Echo Dot	Google Nest Mini
Released In	2020	2020	2019
Voice Assistant	Siri	Alexa	Google Assistant
Microphones	Four-microphone design	Four-microphone design	Three-microphone design
Dimensions (length x width x height)	3.9” x 3.3” (97.9 x 84.3 mm)	3.9” x 3.9” x 3.5” (100 x 100 x 89 mm)	3.85” x 1.65” (98 x 42 mm)
Weight	0.76 lbs (345 g)	0.75 lbs (341.3 g)	0.39 - 0.4 lbs (177 - 183 g)
Cost	$99.00	$49.99	$49.00

Table 13: Different Smart Speakers Specifications Comparison

As seen in **Table 13** above, all three smart speakers include up-to-date technology, as the Google Nest Mini was released in 2019 and both the Apple HomePod Mini and the Amazon Echo Dot were released in 2020. In terms of design, the Homepod Mini and the Echo Dot are the most similar, having practically the same number of microphones, dimensions, and weight. The Nest Mini differs the most, having a three-microphone design, smaller height, and relatively less weight. As mentioned before, Apple’s smart speaker is higher in price than its competitors, being sold at a price point of $99.

4.2.10 Solenoid Valve

Solenoid Valves are needed for our project in order to control how much and when water can flow into the different herbs. A solenoid valve is a combination of two things, first is a valve that is electrically controlled to either stay closed or to stay open depending on whether it is a normally closed valve or normally open valve. The other part is the top of the device which is a solenoid that is meant when current passes through the various loops causing the valve to open or close. We want to focus on getting a solenoid valve that is normally closed. This is because the solenoid gets hot and requires power whenever it is used, since in a base state we normally do not want water to pass through the system getting a valve that is normally closed will allow us to save power on the long run and extend the usage of our solenoid valve since it will only require power once we need the valve to be open and allow water to pass through the tube into the plant. Solenoid valves allow many different kinds of fluids that can pass through the valve, generally most kinds of fluid like gasoline, oil, or water are used and gases like air are used with the solenoid valves. For our project, we will be passing through mostly water which means we need to stay conscious of what materials the

valves are made of so that when water is passing through it, it will not affect the plant in a strong way.

When looking through what solenoid valves were going to be used we came across three main options. Three different companies: Digiten, Kako, and SNS all make solenoid valves of different sizes and from different materials. When investigating which solenoid valve would be the best to use, we need to make sure of the things that we have mentioned before, making sure that the valve is normally closed and that the material of the valve will not affect the water passing through that much. Depending on this, we will determine which of these three options would be the best to use for our project.

Digiten DC Inlet Feed Water Solenoid Valve ¼"
This is a 12V DC Quick connect valve that is normally closed. The pressure that this valve works best at is in the 0.02-0.08 Mpa range and the working temperature is 0-70℃. Although the temperature should not exceed 60℃ for long periods of time because it causes the solenoid part to overheat, so care should be taken that this does not happen. The valve part is made of plastic and can withstand temperatures of about 150℃. The rated power of this device is 4.8W.

Brass Electric Solenoid Valve from Kako 1"
This is also a normally closed valve that works in 110V AC. The size of the tube is one inch and it has a working temperature of 23-176°F. The valve part is made of brass and is resistant to corrosion. The valve is made with a VITON seal that is used mostly when dealing with oil and gasoline. The pressure range is 0-145 Psi. Many reviews say that this valve should not be used when working with water projects because it is made in brass.

SNS NPT Brass ELectric Solenoid Valve ¼"
This valve is a DC12V operated valve that is normally closed. The working temperature range of this valve is -5 ℃~85 ℃. The valve part is made of copper. This solenoid valve works well with different materials like air, water, and oil. The port diameter is 2.5mm and the working pressure when using it with water as the passing fluid is about 15~70 psi.

4.3 Project Software

This section will discuss the research that we have conducted so far regarding the different software technologies that could be utilized in our project. We will compare the different software components that we have investigated by describing them and differentiating them specifications-wise. The following items will be discussed: communication, database, interface, and Wi-Fi module.

4.3.1 Communication

Different types of wireless communication that can be used to provide interaction between multiple devices exist. The devices within the system require different forms of communication to perform their function. Our project utilizes a mix of wireless and wired communication. Possible wired communications that are supported by our microcontroller are described (on **3.2.2 Communication Protocol**) under the Standards section of this document (**3.2 Standards**). The following wireless technologies were inspected to determine which wireless communication is the best fit for our project.

Wi-Fi
Wi-Fi is a wireless networking technology that allows devices to communicate via the Internet without direct cable connections. It is the most commonly used means of wireless communication. Protocols for wireless local area networks (LAN) are specified by a set of standards known as IEEE 802.11. Typically, Wi-Fi uses radio waves to transmit data at 2.4 GHz and 5 GHz frequencies. Lower frequencies like 2.4 GHz provide stronger signals which allow more devices to connect. While higher frequencies like 5 GHz are faster than 2.4 GHz, it also shortens the signal range and is more susceptible to interference.

Bluetooth
Bluetooth uses ultra-high frequency radio waves to directly transmit information between devices within a short distance. Bluetooth devices operate on a frequency band between 2.4 GHz and 2.48 GHz and have a maximum range of approximately 30 feet. Compared to Wi-Fi, Bluetooth is more safe and secure against hacking. This is due to devices hopping between various frequencies every second. Standards for Bluetooth technologies were originally defined by IEEE 802.15.1 but are now defined by the Bluetooth Special Interest Group (SIG).

LPWAN
Low-Power Wide-Area Network (LPWAN) allows devices to send small bits of data over great distances on a single battery that lasts for many years. This technology offers low power, low cost, and wide-area coverage that is perfect for wireless sensor networks. This is commonly used in Internet of Things (IoT) devices that do not require high bandwidth. LPWAN allows IoT devices to operate on small, inexpensive batteries for 10 to 15 years. It can support small packet sizes from 10 to 1000 bytes and its long coverage range can vary from 2 km to over 100 km. LPWAN requires less infrastructure and hardware which results in lower costs.

Zigbee
Zigbee is an alternative to Wi-Fi and Bluetooth which transmits data over long distances through a mesh network. This wireless technology is used when creating personal area networks with low power like home automation. Zigbee devices communicate with each other based on the IEEE 802.15.4 personal area network standard. Security is guaranteed with this technology because of its 128-bit symmetric encryption. Zigbee

operates on a 2.4 GHz frequency and has a data rate of up to 250 Kbps. The range of Zigbee can reach up to 100 meters indoors.

4.3.2 Database

A database is commonly known as a useful source where a large collection of data is efficiently stored and retrieved. In developing Greenie, we decided to utilize a database as a way to better integrate with our full-stack web application. As seen in **Table 14**, one table in our database was created to store 10 specific herbs as well as valuable information relating to each one. The 10 herbs that our product is going to support are basil, bay leaf, cilantro, chives, lemongrass, mint, oregano, rosemary, sage, and thyme. Our database is going to store data on watering requirements, soil preference, ideal humidity, ideal temperature, and ideal pH for all 10 herbs supported.

By interacting with the user interface, we will be able to extract the key components needed from each herb and use that to decide how the irrigation system will operate. For example, looking at basil, we see that it has a water requirement of lightly every day. Taking note of that, we will use that information, tie it into the typical amount of fluid ounces dispersed for basil, and then have it set so the process is automatic. As for specific fluid ounces for each herb, we will itemize them in the future.

With the information shown in **Table 14,** our database is going to better maintain the health of the user's plant. Through the use of the database and the interface, the user is going to be able to access and view this data on our web application. The web application is going to be accessed anytime and through any device as it is going to be opened through the use of a link. For generic data collection and displaying statistics, we plan to utilize a separate table for storing information. Some examples of generic data include current soil moisture, current water level, number of times watered, and the total amount of water dispersed.

For deciding on the best database to use, we looked into several options and decided to weigh the pros and cons of using each one. During our research, we looked to see what each database could do and what features it could provide in helping construct our project. Properties like complexity and restrictiveness were all taken into consideration when coming to a final decision. We began our research by first breaking down the difference between SQL databases and NoSQL databases. Our findings can be seen in **Figure 4.**

Herbs Supported	Watering Required	Soil Preference	Ideal Humidity	Ideal Temperature	Ideal pH
Basil	Every day	Well drained	40 - 60%	70 - 90 °F	5.5 - 6.5
Thyme	Every 2 - 3 days	Moist / Well drained	40%	60 - 70 °F	5.5 - 7.0
Bay Leaf	Every 2 - 3 days	Well drained	40%	> 20 °F	6.0 - 7.0
Lemongrass	Every 2 days	Well drained	40%	65 - 85 °F	6.5 - 7.0
Oregano	Every 2 - 3 days	Moist / Well drained	40%	50 - 70 °F	6.5 - 7.0
Mint	Every 2 days	Moist / Well drained	70%	60 - 70 °F	7.0 - 8.0
Sage	Once a week	Well drained	40%	60 - 70 °F	5.5 - 6.5
Rosemary	Every 2 days	Well drained	45 - 55%	55 - 80 °F	5.0 - 6.0
Chives	Every 2 days	Well drained	40%	40 - 85 °F	6.0 - 7.0
Cilantro	Every 2 days	Moist / Well drained	75%	40 - 75 °F	6.5 - 7.0

Table 14: Herbs Included in Database

When it comes to using the weather component in our web application application, we are looking to include a separate table for that as well. In other words, we are looking to take the information we get from our API and send that over to its own designated section. One reason we think it is beneficial to set up the weather component in this manner is because of organization. In regard to the table holding our general information, while we think it is useful to have a section of our database that specifically caters to that, we also think it is useful to not have a section that becomes filled with too many random pieces. We recognize that at a certain point having all of our information stored into one or two tables is not fully taking advantage of what a database is offering us.

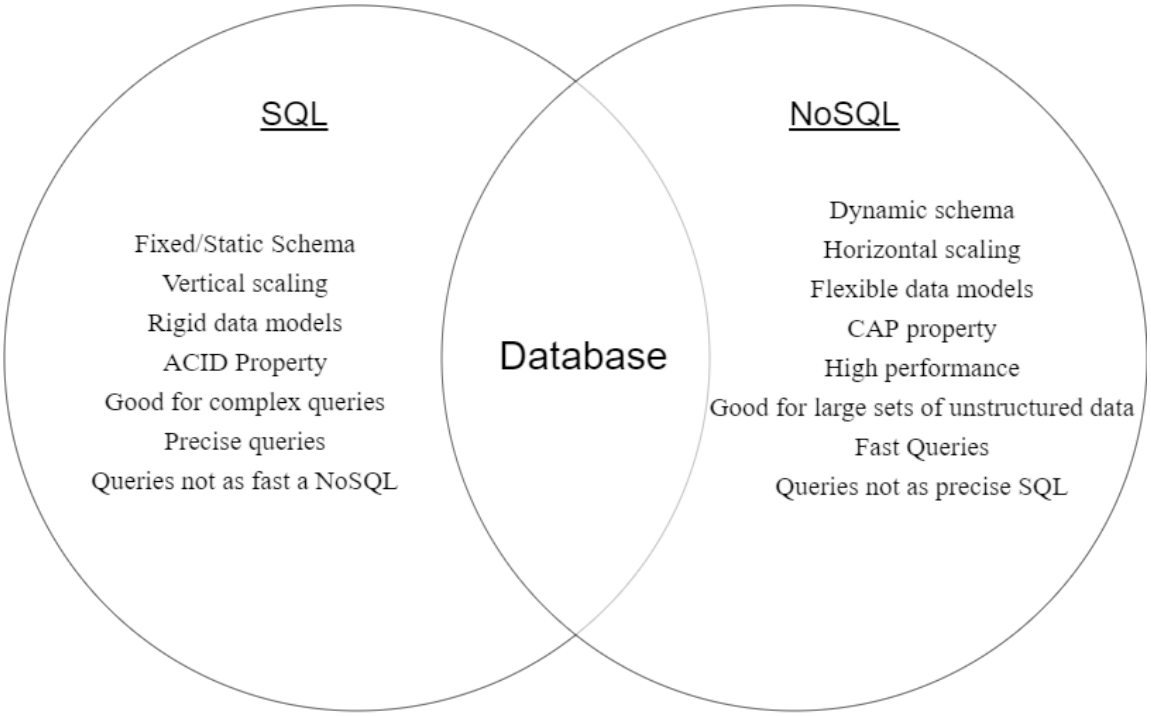


Figure 4: SQL vs No SQL

After analyzing the similarities and differences between SQL and NoSQL we then went over specific databases to see what each of them could offer. To break down each database we decided to outline each one's pros and cons. We recognize that because each database has a large number of features to offer that we can not go in-depth on everything, but we have decided to highlight the values we found to be most valuable. Also, in constructing our list, we decided to highlight the level of familiarity our group has with each database.

MySQL (SQL Database)

- Familiarity = Medium
- Pros
    - Free and open source
    - Supports additional languages (Tcl, Perl, etc.)
- Cons
    - Need 3rd party for LINQ queries
    - Need to back up data by extracting SQL statements
    - Cannot cancel query while running without ending the entire process

Microsoft SQL Server (SQL Database)

- Familiarity = Medium
- Pros
    - .NET framework. Can do LINQ queries
    - Can backup large amounts of data

    - Can truncate database query without ending the entire process
- Cons
    - More expensive. Need to pay for a license to run a server
    - Not as many languages

MongoDB (NoSQL Database)

- Familiarity = High
- Pros
    - Simple setup with MongoDB Atlas
    - Advanced manipulation features
- Cons
    - Joining documents is difficult
    - Slower without proper indexing

Redis (NoSQL Database)

- Familiarity = None
- Pros
    - Simple setup and is easy to use
    - One of the fastest for caching
- Cons
    - Difficult for large cloud deployment
    - Limited database size

CouchDB (NoSQL Database)

- Familiarity = None
- Pros
    - Fast indexing and retrieval of data
    - Can be used to store any type of data
- Cons
    - A large amount of space is used for overhead
    - Huge datasets can result in it being very slow

Firebase (NoSQL Database)

- Familiarity = None
- Pros
    - Custom Libraries that work with devices
    - Realtime database so immediate value readings
- Cons
    - Less support for IOS devices
    - Realtime database not great for complex queries

4.3.3 Interface

The user interface allows users to interact with the system. One of its main goals is to provide users with effective and efficient control of the system. A few user interface designs were considered for our project. This includes a web application, a mobile

application, and an onboard control using buttons. Due to time and budget constraints, our group decided to utilize a web user interface or web application as our project's user interface design. One of the advantages of using a web application is accessibility. It provides users access on any device through a web browser. It also requires less time to build compared to a mobile application. The following lists some tools that can be used when designing and developing the front-end of the web application.

Bootstrap
Bootstrap provides a huge collection of reusable codes written in HTML, CSS, and JavaScript that enables web developers to build responsive web pages quickly. Design templates such as grid systems, tables, and buttons are predefined therefore there is no need to code for your design. Users can fully customize their web page. It also adjusts the images automatically based on the current size of the screen. It is compatible with Google Chrome, Firefox, Edge, Safari, and Opera.

React
React is a free and open-source JavaScript library that is used to create interactive user interfaces for both web and mobile applications. It allows for the easy creation of dynamic applications due to less coding. React contains reusable components with unique logic and controls that can be reused throughout the application which makes it time-efficient. One of its main features is having a virtual DOM (Document Object Model) which allows complex applications to have a better and more effective performance. This web development tool is being used by huge companies including Netflix, DropBox, and PayPal.

TypeScript
TypeScript is a superset of JavaScript that is designed for large applications. It uses existing JavaScript code and supports JavaScript libraries. One of its features is the optional static typing. This will alert the web developers of any possible errors in the code and will result in increased productivity and less error during the execution. It provides flexibility since users can change the level of type strictness. Another great feature of TypeScript is IntelliSense which gives users some hints while coding. This front-end web development tool can operate on any device, browser, and operating system.

4.3.4 Wi-Fi Module

One key piece to ensuring that the irrigation works as intended is an established internet connection. By having an internet connection, we will be able to remotely control the parts needed for our various features. To accomplish this, however, some type of Wi-Fi module antenna will be needed. Through our research, several modules stuck out. The ESP8266 made by Hiletgo stuck out as the most affordable option, the ESP32 by Hiletgo stuck out as a strong option in performance, and the Bolt IoT stuck out for its additional features. The modules cost $7, $11, and $150 respectively.

Comparing the ESP8266 by Hiletgo to the Bolt IoT Module, we discovered that both devices ended up being pretty similar. Both utilize the standard 802.11 wireless connection, the same 3.3 operating voltage, similar dimensions, etc. Looking even closer at the specifications for the Bolt IoT module, we noticed that it also utilized the ESP8266 framework itself. What mainly proved to be the difference between the two modules were the extra services that Bolt provided to its users. Such as their cloud platform, LEDs, and their unique design.

Comparing the ESP32 module to the ESP8266 and the Bolt IoT module, we found more differences in technical specifications. In the ESP32, some of the differences include more GPIOs, faster clock speed, and Bluetooth support. Performance-wise, we continued to find slight improvement after slight improvement to the ESP8266 going down the list. Logically this made sense because the ESP32 is the successor to the ESP8266. Looking at **Table 15**, we can see the detailed specs that highlight these specific instances. What we did notice, however, is that once again the ESP32 does not have the Bolt IoT extra services.

Attribute	**ESP8266 NodeMCU**	**ESP32**	**Bolt IOT Module**
Wi-Fi	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Operating Voltage	3.3 V	3.3 V	3.3 V
Clock Speed	80 MHz	160 MHz	80 MHz
GPIOs	17 Pins	34 Pins	5 Digital Pins
Bluetooth	X	Bluetooth 4.2 and BLE	X

Table 15: Wi-Fi Module Specification Comparison

4.3.5 Web Service

To obtain access to our web application from a URL, we will need some type of web server or service to give our users commercial access. When it comes to web services, there are many options that developers can choose to go with. Originally, we looked into utilizing the Platform as a Service (PaaS) model. The Platform as a Service is a type of service that allows cloud computing platform users to fully develop and deploy applications. From small projects to startups to medium-sized businesses, the Platform as a Service model is becoming more and more of a viable choice in modern web app development. However, for our project we decided to also add in new alternatives with cloud service capabilities. For Greenie, we have recognized the convenience of usings cloud models and have decided to utilize it moving forward. In our research, we found

some of the popular services that use these models and decided to analyze them. The following are our findings:

Microsoft Azure
Microsoft Azure is one of three popular cloud services that is known for offering several types of sub-services to its users. Its main competitors are Amazon Web Services (AWS) and Google Cloud Platform (GLP). The three sub-services that are offered by Azure are the Platform as a Service model, the Infrastructure as a Service (IaaS) model, and the Software as a Service (SaaS) model. Regarding the Platform as a Service model, it has its own environment for getting applications developed and deployed and is incredibly versatile. Some features that we decided to highlight are as follows:

- Familiarity = None
- Works well with their services (Ex. Microsoft SQL Server)
- Takes some time to adjust to
- Only free for a certain period

Heroku
Heroku is a platform that utilizes Amazon Web Services as the framework for getting web applications developed and deployed. It can work in conjunction with repository platforms like Github and Gitlab and it is known for making the development process incredibly smooth. Some highlights that we have outlined in using Heroku are as follows:

- Familiarity = High
- Allows us to spend more time on coding as opposed to repeated server management
- Easy to use/More user-friendly
- Has an option that is free to use

Google App Engine
Google App Engine is a type of Platform as a Service that utilizes a sandbox-type environment for code to be housed and run. Similar to other services it is compatible with a wide variety of programming languages and handles the development and deployment process well. One of the things the service is known for is showcasing its compatibility with Google's programming language Go. Some of the features we outlined for it are as follows:

- Familiarity = None
- Offers automatic scaling to assist in management
- Has an option that is free to use
- Works best for Google applications

Drive to Web
Drive to Web is an online cloud service that allows users to make their code accessible on the internet. It is similar to the Platform as a Service model, but operates in a different fashion. Unlike the providers mentioned above that utilize the Platform as a Service model, Drive to Web allows developers to house their code on Google Drive and make it available in the same way as if it were housed on Github or Gitlab. The

convenience that comes with using Google Drive over processes like the Gitlab continuous integration/continuous deployment can in some cases be seen as a benefit.

- Familiarity = None
- Automatically updates code when added
- Free to use
- Gives developers more time to work on coding

4.3.6 Weather Integration

For our project, another feature we would like to incorporate is a form of weather integration. Given that our project is a smart irrigation system, we figured that our product would not be that smart if we are using water from the water pump while it is raining outside. So to account for this, we have decided to incorporate the weather as a useful tool throughout the user experience. A couple of uses can be seen through our web application. First, on the main page, we will have the weather posted for the user so they can decide whether or not to leave Greenie outside. Second, in the app, we can have a popup for the user so they get further clarification on if they are okay with their decision to utilize the pump system. By doing this, we believe the user will become more informed on their habits when watering the herbs. To accomplish these tasks though, we will have to retrieve weather data relative to the user's location and send that information to our database. The method we plan to use to retrieve the data is through a weather API provider. In our research, we found several providers that would work well for us. However, for selection purposes, we decided to list out some of our options and see the features that they offered. That way we could choose based on what seemed most intriguing to us. Our findings are as follows:

OpenWeatherMap

- Has a free option
- Works with JSON
- Provides minute by minute forecasts and historical data
- A large number of weather stations around the globe

Tomorrow.Io

- Has a free option
- Works with JSON
- Offers an all-in-one endpoint with multiple data fields (real-time weather, pollen, air quality, etc.)
- Can receive notifications

WeatherBit

- Has a free option
- Works with JSON
- Uses 5 different APIs
- Limited to 500 API calls a day

4.3.7 Repository Management

Another key piece to ensuring that our web application is set up properly is managing a software repository. With a software repository, we will be able to store all of our code and deploy it to a web service such as the various models described in section 4.3.5. Additionally, we will have the ability to write code collaboratively given that we can share access to projects within the platform. For most modern coding projects today the repository platforms that are often used are Github, Gitlab, and Bitbucket. In other cases, platforms like Google Drive can also be utilized to house code.Each platform has its advantages and its disadvantages, but they are all capable of getting the job done for our project. So, to assist us in coming to a selection we decided to weigh the pros and cons of our options and figure out which platform would be of the greatest use to us.

Github
Github is the repository platform that is often viewed as the most popular and is known for having many developers on it. The philosophy of Github differs from something like that of Gitlab because it focuses on ensuring that the overall performance of the platform is good rather than having more things to offer. It also started in 2008 which is a few years earlier than platforms like Gitlab, so some may argue that the platform is more stable. Meanwhile, others may say that regardless of the year it started, platforms like Gitlab have caught up to it.

- Familiarity = High
- Pros
    - Large Community Support
    - Good for infrastructure support
    - Easy to learn for new users
- Cons
    - Continuous Integration/Continuous Deployment (CI/CD) with external tools
    - Maximum 3 contributors on the free plan (private projects)
    - Security is not the greatest

Gitlab
Gitlab is a repository platform that is often known for having a lot of features. When it comes to their competitors, Gitlab prides itself in having more things to offer than everyone else. It started in 2011 and is the youngest out of three, but it still has a strong following. Also, unlike the other two Gitlab is known for using what are called merge requests. With merge requests, the developers can combine all the code they worked on separately into one branch. Github and Bitbucket have a similar process, but they refer to it as pull requests.

- Familiarity = High
- Pros
    - Good for features-based support (Usability Testing, DevOps Reports, etc.)
    - Continuous Integration/Continuous Deployment (CI/CD) built into the platform
    - Allows for Subgroups
- Cons

    - Slightly higher learning curve for newer users
    - Pipeline fails could cause long delays
    - Platform can be slightly slower

Bitbucket
Bitbucket is a repository platform that is often not talked about as much as Github or Gitlab but it is still capable of getting the job done. With Bitbucket, the developer has access to functionality similar to that of Github. It also started in 2008, and some may say that it is just as stable as Github. However, when it comes to features it is also known for working well with several external integration tools like Jira.

- Familiarity = Low
- Pros
    - Has API that allows you to build your own integrations
    - Has a useful Jira integration tool for tracking
    - Is flexible for code deployment
- Cons
    - Not open sourced
    - Slow for heavy loads
    - Maximum 5 Contributors

Google Drive
While it may be not as commonly known as the platforms mentioned above, Google Drive is also capable of acting as a repository for code storage. In many cases where programming projects can be complex, Google Drive is probably not utilized as the sole platform, but it can act as a great backup for sharing code easily and quickly.

- Familiarity = Medium
- Pros
    - Platform is easy to use
    - Is flexible for code deployment
- Cons
    - Limited community support
    - Limited support for different programming languages

5. Design

5.1 Overview

In the previous section (**4. Research**), we discussed the research behind each part needed for our project. We compared and contrasted different pieces of technology to better understand which components would work better for us. In this section, we will discuss which components we chose to incorporate into our product and why.

5.2 Technology

This section discusses the decisions made in software technology. Here, we will discuss the conclusions that we arrived at while researching which technologies to utilize. We will be discussing our chosen communication method, database platform, interface application, Wi-Fi module component, web service, weather API provider, and repository platform.

5.2.1 Communication

The communication between the microcontroller and the cloud server is wireless. As seen in **Figure 5**, the microcontroller sends data from the sensors to the server through wireless communication. This allows users to get information about their plants. Since the statistics of the herbs will be uploaded to a cloud and can be accessed through a web app, we decided to use Wi-Fi for wireless communication. With a Wi-Fi module, the transmission of data from the microcontroller to the cloud is direct and the system can be installed anywhere close to a Wi-Fi access point. Wi-Fi is the preferred form of wireless communication because of its increased efficiency since the transmission of data between devices is usually faster when using Wi-Fi. Because of the flexibility and the availability of Wi-Fi technologies, the users of Greenie will be able to access the web app anywhere using any device as long as they are connected to the Internet. In contrast, the sensors will send data to the microcontroller through wired communication.

SPI, I2C, and UART are the possible communication protocols that will be utilized for the transmission of data between the sensors and the microcontroller. Since the microcontroller and all the sensors are compatible with I2C, we decided to use I2C to integrate all the sensors into the microcontroller. The following reasons were also considered when choosing the appropriate communication protocol for our system. I2C can handle multiple components that work simultaneously without jeopardizing the communication path. Our project consists of multiple sensors and with I2C, the communication between the sensors and the microcontroller would hopefully be smooth. Another advantage of using the I2C protocol is that only two wires are required to transmit data between the devices, which means we can lessen the wires in our project and avoid complicated wiring.

Figure 6 demonstrates how users will interact with the system and how the system will respond to users. The user will be the plant owner while the system will be a microcontroller and the sensors. The user will interact with the system through a server. The server will receive all the requests from the user and send it to the system. The server will then gather the response from the system and then send it to the user.

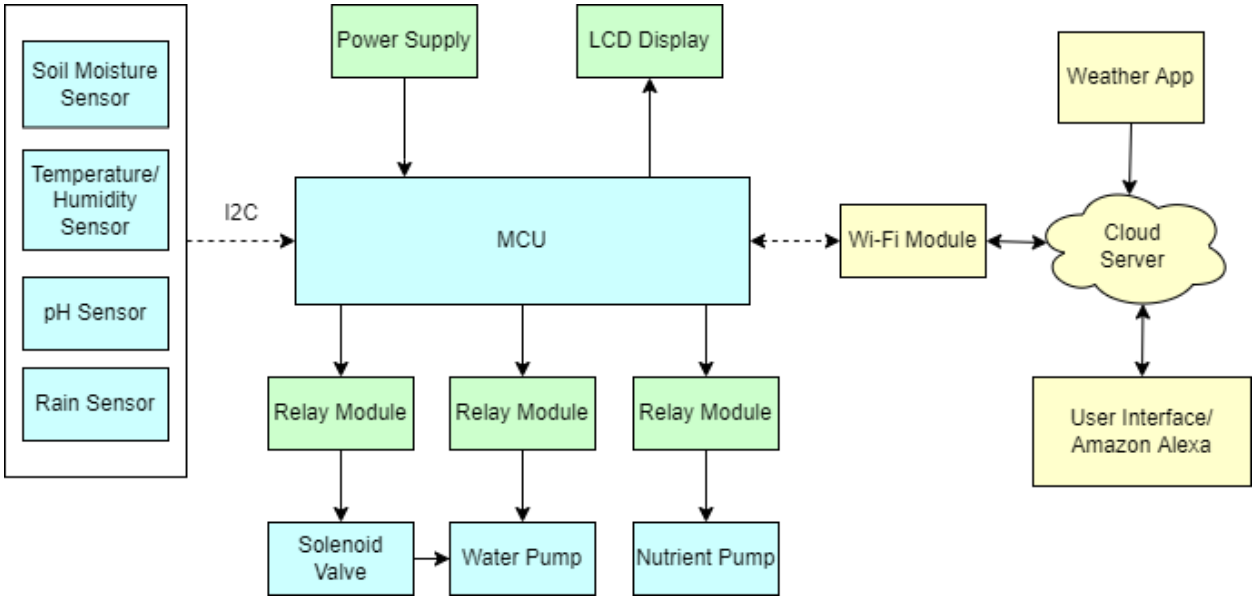


Figure 5: Hardware and Software Communication

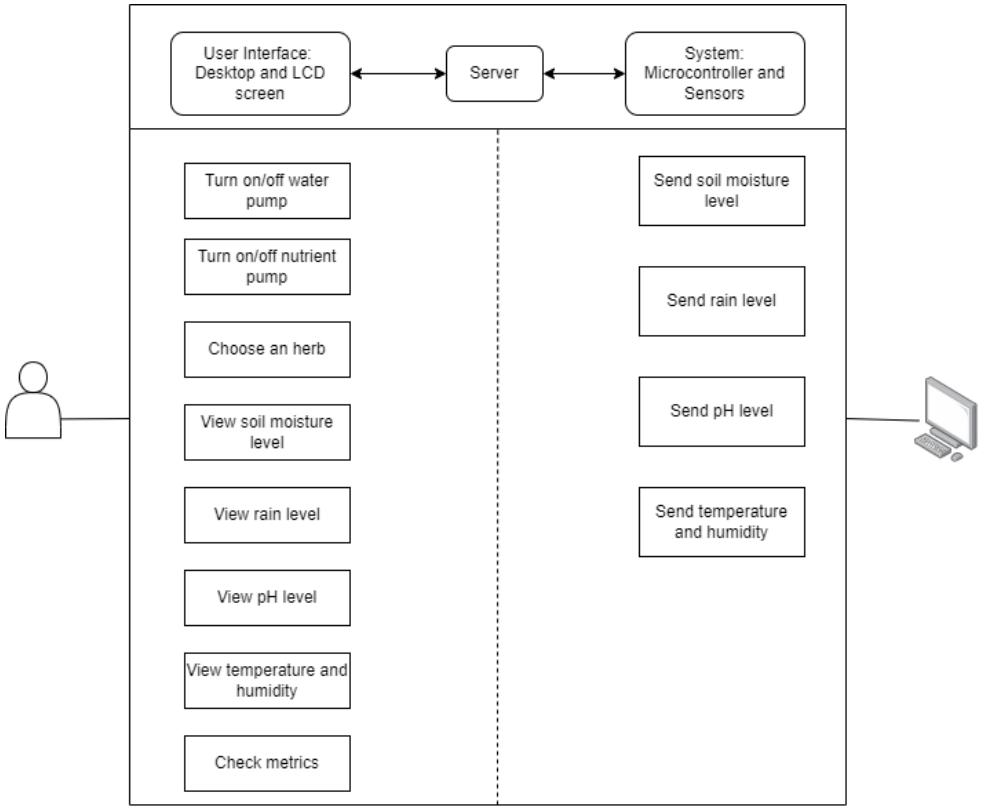


Figure 6: Communication between the user and the system

5.2.2 Database

For our database, we originally decided to use MongoDB, but we ended up switching to Firebase. Initially our reasons for selecting MongoDB were because of two areas: familiarity and utility. When it came to familiarity, out of all of the choices we went through, we recognized that the MongoDB database platform was the one most members of our team were comfortable with. MongoDB offers its Atlas service to quickly construct our tables and get our connection synched up with our server and it also provides us with a way of easily giving shared access amongst our team. However, given the developments that occurred in constructing the project, we realized that switching to Firebase would be more appropriate. For starters, we recognized that in working with the hardware, getting a reliable connection to send values to the database ended up taking more time than anticipated. So, when we noticed that Firebase had custom libraries that specifically worked with our Wifi module we recognized that switching to it would be a huge time saver. Additionally, we noticed that after switching away from the MERN stack and utilizing pure HTML, CSS, and Javascript, Firebase also made it simple for us to sync stored values with objects like buttons and lines of text. As for utility, with the type of application we planned to create, we recognized that having a NoSQL (Not only SQL) database was still ideal. To be specific, we still intended on creating a web application that had a stack like framework where JavaScript would be throughout the entire web application. For us to complete such a task, we acknowledged even after switching to Firebase, we would still get the advantages of a NoSQL database as well as a good amount of community support. We can say however, that a small downside that came from the switch was not implementing tables in their original concept. So things like separate tables weren't created for the weather portion of the databases, but rather status sections and various subsections we utilized to carry out our plans.

5.2.3 Interface

The main interaction between the users and the system is through a web application. The user will be able to view the herb's daily statistics through the web app. They can access the web app remotely on any device that has a web browser and an internet connection. The herb's statistics include the current soil moisture, the rain level, the pH level, and the current temperature and humidity. The web app has a list of herbs that the user can choose from, which contains information about the herb to determine how the watering system will operate. All information is going to be stored in the cloud database. The web app also provides an option for the user to manually turn on or off the water pump and the nutrient pump. **Figure 7** (see below) details the overview of the web application in an easy-to-follow diagram.

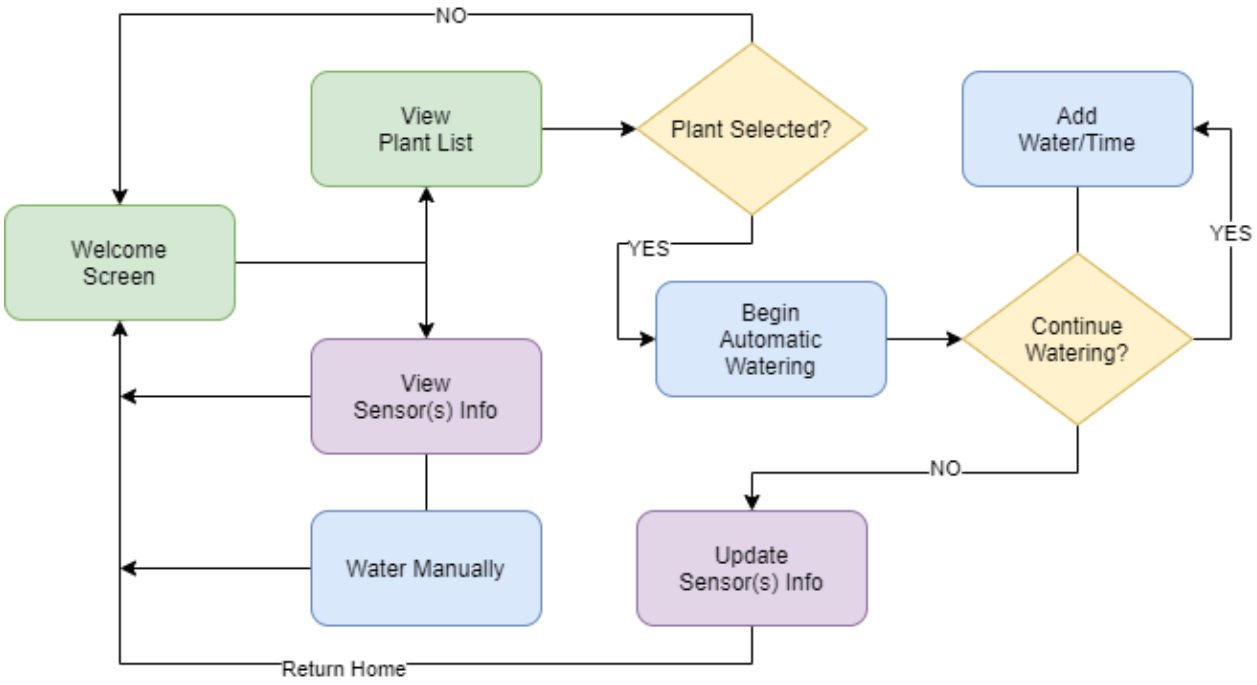


Figure 7: Web Application High-Level Overview

For the front end of our web application, we originally decided to utilize React. There are several reasons why we chose this specific web development platform to build our system's user interface. The first reason is the members' familiarity with this platform. Some of our members have experienced developing a web app through React. This saves us a lot of time since we do not have to learn a new tool and software. We just have to apply what we know so that we can create the web app easily and smoothly. Because of its very simple library, React is easy to learn, and having JavaScript skills will make it even easier to adapt to this environment. Other reasons are the advanced features that React has which are discussed below.

One of its best features is the Virtual DOM (Document Object Model) where a virtual representation of the actual DOM is kept in the memory. With this feature, any modifications that are made will be performed initially on the virtual DOM and not on the actual DOM. It will then compare the differences between the actual and the virtual DOM and will update the actual DOM based on the changes made instead of updating all the components on the user interface. This selective modification guarantees an increased performance of our web application and a better experience for the users.

Traditionally, reusing a code when building a web app is not ideal since making changes on one component will affect all the same components. This is where React gains an advantage over other web development platforms because it provides reusable components. Each component has its logic and controls which are easy to manipulate. React allows web developers to reuse these components in other parts of the code that have the same functionality. This increases our productivity while developing the web app and help us maintain the structure of our code.

Most JavaScript frameworks follow a multi-directional data flow wherein changes in the child can affect its parent as well. React, on the other hand, follows a downward data flow which means the parent is not affected by the changes made to the child. This helps us to have a more stable code. This feature also allows any member to track all the changes made to a specific part of the code.

For the web application designed for our project, we decided to switch to pure HTML, CSS, and JavaScript. Bootstrap was also used as an add-on rather than a framework by adding one extra line of code to the HTML file. Although Bootstrap provides predesigned grid systems, tables, and buttons, additional styling and font links were added to provide users with the best user experience possible.

5.2.4 Wi-Fi Module

For our Wi-Fi module, we decided to use the ESP8266 NodeMCU. Our reason for choosing this module came down to the cost. We noticed during our research that we did not need materials that go beyond our required specifications. All we needed was a part that could perform in the way we outlined. As seen in **Table 16**, our module does utilize Wi-Fi as well as the UART, SPI, and I2C protocols mentioned in our requirement specifications. During our research, we found that this module not only works well with our microcontroller of choice but that it also makes features like Alexa integration even easier.

Attribute	**Description**
Operating Voltage	3.3 V
Clock Speed	80 MHz
Flash Memory	4 MB / 64 KB
UART / SPI / I2C	1 / 1 / 1
Wi-Fi Built-In	802.11 b/g/n

Table 16: ESP8266 NodeMCU Technical Specifications

5.2.5 Web Service

For our web service, we originally decided to go with Heroku, but we ended up switching to Drive to Web. We initially intended on using Heroku as our platform of choice once again for two reasons, cost, and familiarity. However, given the developments that occurred while working on the project, we realized that we need to adjust to a platform that would demonstrate our web application working in a timely manner. This can be further expanded upon by looking into the switch from the MERN stack to regular HTML, CSS, and JavaScript. Given the stack format and given that

Heroku is known to be useful for it's compatibility with NodeJS, we realized that after moving away from NodeJS that Heroku was no longer the top contender. With Drive to Web we noticed that when working with pure HTML, CSS, and Javascript, that putting on Google Drive made the process even easier than using the Gitlab continuous integration/continuous deployment process. The Drive to Web Providers would simply update their servers so that whenever new code was uploaded. The process is much simpler and there is no need to continually manage pipelines to see if jobs fail or pass. Lastly, a reason we decided to switch to Drive to Web is because we recognized that similar to Heroku that using this platform was also free. All we needed was a Google account that gave the Drive to Web providers the ability to make our code public on the internet.

5.2.6 Weather Integration

For weather integration, we have decided to use OpenWeatherMap as our API provider. From a technical standpoint, it provides all the information we need and goes beyond that. For Greenie to be successful, we will need minute-by-minute forecasts. This is important because we do not wish to delay the water process for a long time when it is not necessary. In the scenario where the forecast is set to rain all day and there has not been an update in hours, then that could lead to the event where the herbs never get the water they need. Another reason why we decided to go with OpenWeatherMap is because of the need for information based on where you are located. With OpenWeatherMap, we saw that it has thousands of locations spread across the globe and that one should be able to access their weather information wherever they go. We found this as an important point to us because one of the key elements we are looking to include when designing Greenie is that it is portable. When it comes to the other options we researched, we recognized that both Tomorrow.io and WeatherBit also had great features to offer. So much so that all 3 options appeared pretty even to us. So, when it came down to selecting out of the three, we ultimately ended up choosing the one that we found to be the best representative of Greenie's mission.

As for the design plan we decided to implement, the outline can be seen in **Figure 8**. In this design, we can see that based on the information that is gathered from OpenWeatherMap, we check to see if it is raining first. If there's no rain then we can proceed with the process outlined in the interface section after a button is pressed. Otherwise, if the user is looking to manually or automatically water the herbs remotely, then they will be notified that it is raining and that it would be better to save water by keeping the product outside.

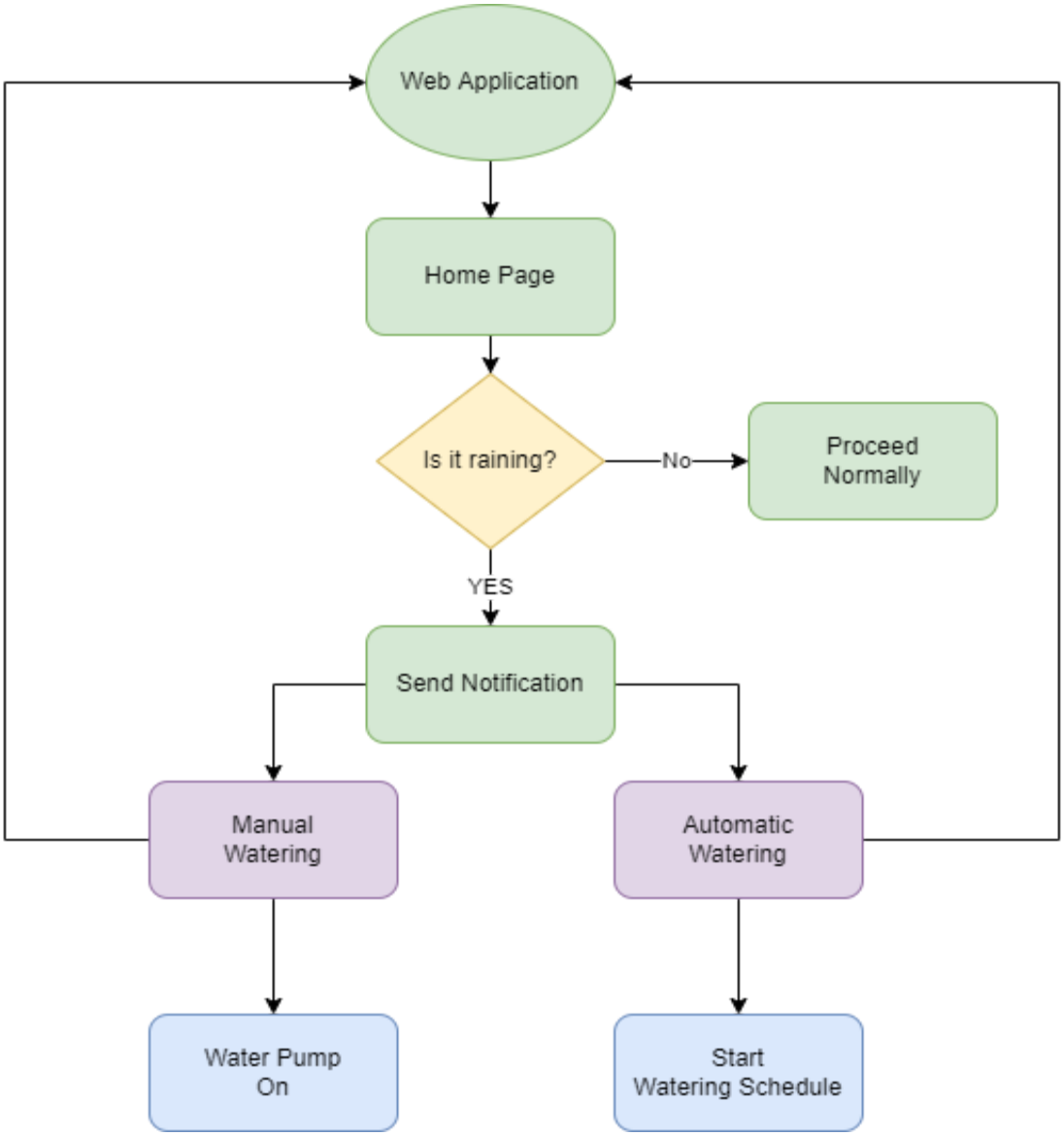


Figure 8: Weather Check Process

5.2.7 Repository Management

For our repository platform, we originally decided to go with just Gitlab, but we decided to switch to a combination of Gitlab and Google Drive. We originally wanted to utilize just Gitlab because it had one of the greatest features to assist in constructing our web application, the built-in continuous integration/continuous deployment (CI/CD) tool. With this tool, we were going to place all of our code in a single project and not have to manually send over updates to our web service. To be more specific, with every commit (i.e. change) we pushed into our Gitlab project, Gitlab was going to automatically construct a pipeline and send our code over to our web application within a matter of minutes. By having that type of feature built into the platform, we would be able to save

a tremendous amount of time in the development process. However, given that during the development we decided to move away from the MERN stack model, we realized that using the Gitlab CI/CD process became less realistic. So, what we decided to do as a team was still utilize Gitlab for code storage and Google Drive for the code deployment. Additionally, given that Drive to Web as a web service was something that was still new to the team, we didn't want to rely solely on Google Drive as the platform for handling the code sharing and storage. Which ultimately goes back to our constraint with time. We recognized that because we have such a short amount of time to complete our project, we could not afford to experiment on tools that did not need experimenting on.

5.3 Architecture

This section discusses the decisions made in hardware architecture. Here, we will discuss the conclusions that we arrived at while researching which components to utilize.

5.3.1 LCD

For our project, we have decided to utilize the LCD1602. The main factor behind us choosing this LCD was that it is our most affordable option since one of our team members already owns one, helping us maintain our desired budget for the project. The device is lightweight, small, and inexpensive to use. This LCD is also versatile, being able to be used for several different purposes, such as Arduino projects, IoT projects, home animation, and smart building. This LCD is made of two lines of sixteen characters to make different kinds of numbers, letters, and symbols. This LCD allows the use of about 160 characters that are produced to be used. The LCD1602 we acquired has a 6-pin interface (standard), an operating voltage of +5 V or +3.3 V (typical), and supports the I2C communication protocol and development board.

Design-wise for our LCD, we were planning to allow the user to see a large variety of data. We would have started with a standard welcome screen that tells the user that they are using Greenie. We would have then gone on from that screen and have scrolling text that continually gives updates on the status of herb being watered. The two most prominent being the data retrieved from the sensors (temperature, humidity, soil moisture, pH) and the weather. However, this does take into consideration that a plant was selected and the automatic watering process has been set up. We are also looking to implement elements such as the number of times watered in day, the amount of water dispersed in a day, or the water schedule in use. By including elements such as these into the LCD we think that we are not only giving the user more of a reason to use the LCD, but we are also making the data analysis process more convenient for the user. Our rationale for including these elements goes back to the user experience.

Due to time constraints, our wish to include the desired specifications mentioned above was not completed. We were still able to allow the user to observe data from the sensors on our LCD screen as well as use voice commands to observe this data. It was

important to us to show basic data as, looking at the case where the user just wants to see the status of their herbs and they do not have the time or desire to load their electronic device, we think the user should have the ability to do so. Given the world we live in today is fast paced and there are times users need quick status updates, we think the LCD will come in handy. Additionally, this setup works well for us as developers because it allows our LCD to standout as a feature and not just an add-on component.

5.3.2 Microcontroller

For our microcontroller, we have chosen to utilize the Atmel ATmega328P. Our decision was based on three factors: cost, familiarity, and versatility. In terms of cost, the ATmega328P is our most affordable option for an MCU since one of our team members already owns one. This helps us maintain our total budget for the project as low as possible while still accounting for all of the other components we want to integrate into our system. In terms of familiarity, the ATmega328P is our best option given that it is compatible with C/C++ and its libraries. C/C++ is a universal coding language that all members of this group are comfortable with, making it more feasible to work with the Atmel MCU.

The ATmega328P is one of the most popular MCU's that are currently being used in the market for IoT projects, given that it is a versatile, high-performance, low-power microcontroller that is Arduino-based and optimized for C/C++ compilers. It is most commonly utilized in machines to receive, analyze, and output data. The Atmel MCU allows hardware and software components within a system to connect with the use of Wi-Fi modules, which is why we plan on utilizing it. Another benefit of choosing this microcontroller as the "brain" for this system is the fact that it can be programmed using the popular and easily available Arduino Software IDE. The technical specifications of the ATmega328P can be observed in **Table 17** below. As seen in **Table 17**, this MCU is equipped with fourteen pins, which meets our requirement specification of at least ten. These pins are going to be utilized for the sensors and other hardware components (refer to **Figure 2**).

For this project, two ATmega328P chips were utilized to maximize performance and guarantee the best experience possible for the user.

Attribute	Description
Microcontroller	ATmega328P
Operating Voltage	5 V
Input Voltage	7 - 12 V (recommended) / 6 - 20 V (limit)
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	Current per I/O Pin 40 mA DC Current for 3.3 V Pin 50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM/EEPROM	2 KB / 1 KB (ATmega328P)
Clock Speed	16 MHz
Length/Width/Weight	68.6 mm / 53.4 mm / 25 g

Table 17: Atmel ATmega328P Technical Specifications

5.3.3 Power Supply

The part that would have been chosen as a second source of power for the system was the 12V 15Ah Mighty Max Battery while also using the 10Ah power bank as a backup power source. These two parts were chosen over the others because of the power needed to make it work at night (refer to **Figure 3** and **Table 4**). Taking into consideration that in twelve hours there is only about 8000 mAh that is being used in a scenario where all of the parts in the project are being used without anything being shut off or taking a break, it can be safe to assume that during the day, the solar bank can be charged with the solar panels and can be used during a single charge. At night, which we assume to be about a 12 hour period, since the 20,000 mAh bank is about 2.5 times more than what is used, we can simply use the 10,000 mAh version of the power bank for our design because the 15Ah battery can help with any issue that is left. Not only is the 10,000 mAh version lighter, smaller, and more cost-effective, but it also uses less voltage. The Mighty Max battery is needed mainly because of the pump; the pump is a 12V device, so we need a sufficiently strong battery to give power to the pump and any other component that is being used in the project.

As previously mentioned, although we wanted to implement a second source of power in our system, due to time constraints, we unfortunately had to cut the 10Ah power bank as a backup power source. We decided to just go with AC/DC and DC/DC power for our system.

5.3.4 Sensors

The soil moisture sensor we are going to utilize is the KeeYees LM393. Soil moisture sensors with the LM393 chip are widely used in automatic irrigation systems to detect when it is necessary to activate the water pumping system. It is a simple sensor that measures soil moisture by varying its conductivity. The KeeYees LM393 is supplied with a standard measurement board that allows the measurement to be obtained as an analog value or as a digital output, activated when the humidity exceeds a certain threshold. The values obtained from the sensor range from 0 submerged in water, to 1023 in the air. The KeeYees LM393 sensor we acquired has the following technical specifications:

- Chip: LM393 (stable operation)
- Operating Voltage: 3.3 - 5 V
- Power Indicator (RED)
- DO LED (GREEN)
- Mode: Dual output mode (accurate output)

The rain sensor we are going to utilize is the Teyleten Robot LM393. Similar to our soil moisture sensor, this rain sensor contains an LM393 chip, making it ideal for irrigation projects like ours. The Teyleten Robot LM393 surface is coated with a nickel plating treatment, giving the rain sensor a higher conductivity and a longer service life. The sensor can be used to monitor different kinds of weather conditions and translate the results into an output signal and analog output. We chose this rain sensor over the other options we found since it was the most cost-effective, given that it was $5.88 for three sensors, as opposed to the ACROBOTIC sensor, which was $8.99 for one sensor. The Teyleten Robot LM393 sensor we acquired has the following technical specifications:

- Chip: LM393 (stable operation)
- Operating Voltage: 3.3 - 5 V
- Output Current: 15 mA
- Item Weight: 9g / 0.32oz

The humidity and temperature sensor we are going to utilize is the DHT-11. This sensor is a simple, low-cost sensor that uses a digital pin to send information. Because it is digital, the sensor is more protected against noise. Our system requires a humidity and temperature sensor that is going to allow the system to read data from the environment to maintain the plant's health. The main factor behind us choosing this sensor was that it is our most affordable option since one of our team members already owns one, helping us maintain our desired budget for the project. This sensor has the following technical specifications:

- Power: 3 - 5 V
- Maximum Current: 2.5 mA
- Humidity Readings: 20 - 80% RH
- Temperature Readings: 0 - 50° C
- Accuracy: ±5% RH, ± 2° C

The pH sensor we are going to utilize is the GAOHOU PH0-14 Sensor Module. The pH electrode probe included in this sensor is accurate and reliable, supplying the user with almost instantaneous readings (less than one minute). This sensor is also versatile, being able to be used for several different purposes, such as aquariums, hydroponics, laboratories, etc. We selected the GAOHOU PH0-14 pH sensor because as we were researching pH sensors, we came across a couple of irrigation projects that utilized the same one and achieved good results. Compared to the DONGKER pH Sensor Module, the GAOHOU PH0-14 has a better measuring temperature range and zero point, also contributing to us choosing the latter sensor. The GAOHOU PH0-14 Sensor Module we acquired has the following technical specifications:

- Heating voltage: 5 ± 0.2V (AC DC)
- Working current: 5 - 10 mA
- Working temperature: -10 - 50 ℃ (nominal temperature 20 ℃)
- Humidity: 95% RH (nominal humidity 65% RH)
- Module Size: 42 mm × 32 mm × 20 mm
- Output: analog voltage signal output

The sensor that was chosen as the vibration sensor as the piezoelectric sensor was the HiLetgo 801S vibration sensor module. This sensor module has a LM393 chip and would provide data towards the MCU to let the user know that water is flowing into the water. This sensor was chosen because of its cheap price, and because we need one that is sensitive enough to detect water flowing through the tube. The small size also allows it to be as unobtrusive as possible. The sensor will be placed at the furthest end of the tube and the soil meets, and if it is sending a signal this would mean that the water is in fact going into the tube. The HiLetgo 801S vibration sensor module has the following technical specifications:

- Operation Voltage: 3 - 5 V
- Working Current: 5 - 10mA
- Operating Temperature: 0 - 70 ℃
- Dimensions: 45mm x 10mm x 15mm
- Interface Type: Analog output signal

As previously mentioned, this sensor was removed from the final design as we were limited on the features to include in our project due to the time constraints we experienced.

5.3.5 Relay Module

For our relay module, we have chosen to utilize the HiLetgo 5V Relay Module. This is a relay module that boasts a stable performance and a strong driving capability. The HiLetgo 5V relay also contains a “fault-tolerant” design[4] where the relay will not operate if the control line is broken. This makes this relay module ideal for switching power loads. The relay module’s contacts are designed to switch between loads with maximum AC and DC levels of 10 A and AC 250 V and DC 30 V respectively. We plan to leave a margin below the previously mentioned values when we utilize this relay module to not exceed its limitations. The relay module also includes two LED lights. The

green LED is the power indicator and the red LED functions as the relay status indicator. The HiLetgo 5V Relay Module we plan on utilizing has the following technical specifications:

- Current Rating: 10 A
- Contact Type: Normally Open, Normally Closed
- Trigger Select: High or low level
- Module Size: 50 mm * 26 mm * 18.5 mm (L * W * H)

The design planned for the relay module can be seen in **Figure 9** below**.** As seen in the figure, through the use of the HiLetgo 5V Relay Module, our system is going to automatically turn on and off the system and water pump whenever they are not being utilized. Our microcontroller, the Atmel ATmega328P, is going to direct the action.

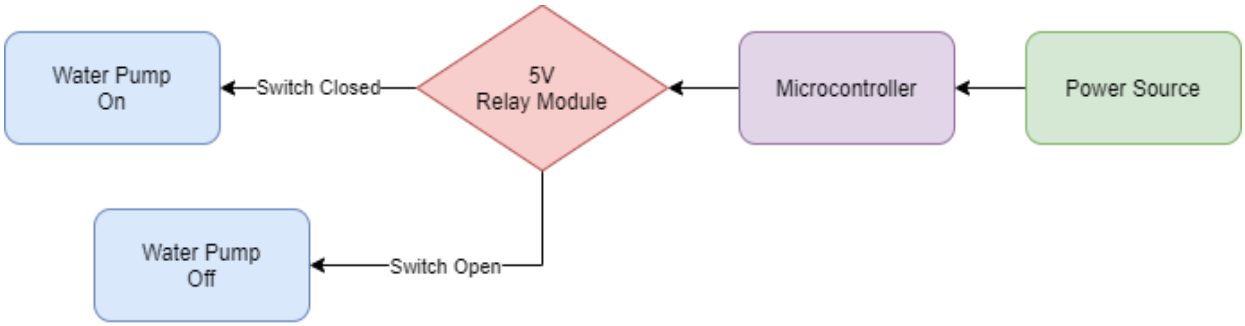


Figure 9: Relay Module Connection

5.3.6 Water Pump

The water pump we decided to utilize is the LEDGLE Mini Submersible Water Pump. This is an easy-to-install water pump that has both inlet and outlet pumps, which is something our team prioritized. Having a water pump with both inlet and outlet pumps gives the user more flexibility, allowing the user to choose if they would like to install the water pump outside or inside the water container. The LEDGLE water pump is low-noise, features anti-explosion and no-spark properties, and is IP68 waterproof. This water pump is also versatile, being able to be used for several different purposes, such as fish tanks, aquariums, fountains, etc. We chose this water pump because it was the most cost-effective option, being only $8.99, as opposed to $10.99 and $12.99 from the other models we viewed. Another factor for us choosing the LEDGLE is that it claims to have a long lifespan and higher efficiency due to its lower energy consumption. The LEDGLE Mini Submersible Water Pump we acquired has the following technical specifications:

- Power Source: Corded Electric
- Inlet/outlet diameter: 8 mm
- Volume: ≦ 40 dB
- Max water temperature: 60 °C

5.3.7 Nutrient Pump

The nutrient pump we decided to utilize is the Gikfun 12 V DC Dosing Pump. As previously mentioned, this nutrient pump boasts a “Snap-in” type design through which it is easy to remove the pump head, making it very convenient for pump tube replacement and cleaning. The pump has the following specifications:

- Motor RPM: 5000 RPM
- Operating voltage: 12 V DC
- Flow rate: 0 - 100 ml/min
- Current: 80 mA
- Diameter: 31.7 mm

5.3.8 Voltage Regulator

Taking into consideration that we are trying to use more modern components with our project, we are going to go with the LM2576. These types of voltage regulators are better than the older versions because these are step-down switching regulators that can accurately drive 3 A with great line and load regulation. Many different voltages can be used, the main ones that our system will be using are the 5 V and the 12 V. This value is also adjustable from a range of 1.23 V to 37 V. There is a high-efficiency procedure in these regulators that allows for a large reduction in the amount of heat that is created. Built into these regulators is a 4% tolerance on the output of the regulator that also comes with a 10% tolerance on the oscillator frequency. With its high resistance to temperature and taking into account the tight tolerance that the new voltage regulators provide compared to the older LM317T, the LM2576 is a much better voltage regulator. The regulators have some important technical specifications that are important to keep track of when looking into how to use these:

- 52 kHz frequency oscillator
- Dimensions of 10.16 mm x 8.51 mm
- 3 A output current
- Maximum operating temperature of 150 °C

5.3.9 Smart Speaker

For our smart speaker, we have chosen to utilize the Amazon Echo Dot. Our decision was based on three factors: cost, familiarity, and versatility. In terms of cost, the Echo Dot is our most affordable option for a smart speaker since one of our team members already owns one. This helps us maintain our total budget for the project as low as possible while still accounting for all of the other components we want to integrate into our system. The Echo Dot is the smart speaker that is most familiar to us since we have all interacted with one. Given its popularity and wide-range accessibility, it has become a product that is commonly found in homes today. In terms of versatility, the Echo Dot is our best option given that it is compatible with the largest number of technology devices. Given that the Echo Dot can also be integrated with Zigbee, it is the best choice for our project. This will allow us to seamlessly integrate our wireless IoT networks into Greenie.

As seen in **Figure 10**, the Echo Dot is a crucial component of our project. Through the use of the speaker's smart assistant, Alexa, the user is going to be able to command different actions to be performed and listen to their results. The user is also going to be able to listen to stored data within our cloud. Once the user calls Alexa, they will be able to activate the sensors, the water pump, and the weather prediction application. Through the sensors, the user can ask to listen to the current soil moisture, pH, humidity, and temperature levels. With the combined use of the sensors and the weather prediction application, the user is going to be able to ask and listen to whether the plant has been watered by rain or not and if it will be in the predictable future. Lastly, through the activation of the water pump, the user is going to be able to ask the system to either water the plant immediately or to set up a scheduled time to do so. The user can also ask to listen to whether the plant has been watered that day or not. In **Figure 10** below, the process explained in this paragraph can be observed in a flowchart manner, making it easier for the reader and/or user to follow.

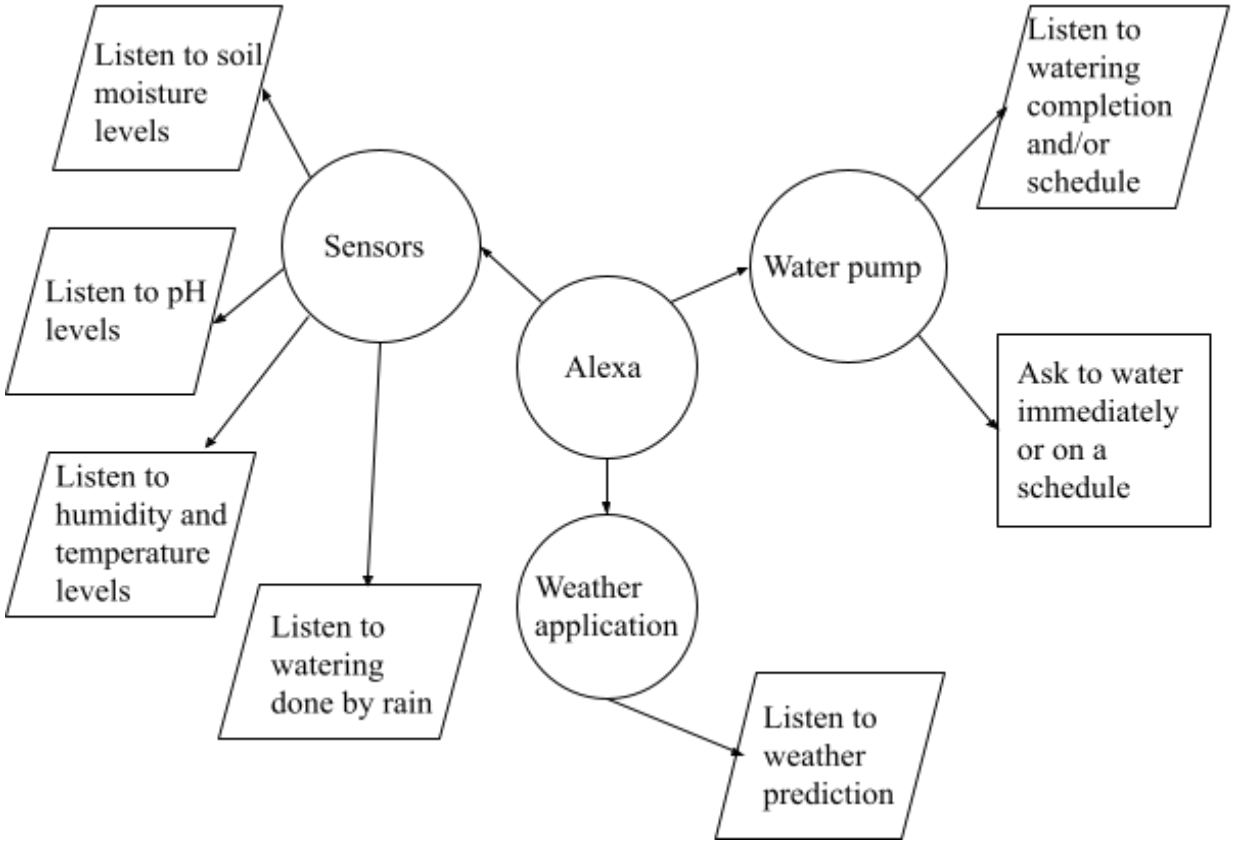


Figure 10: Alexa's Capabilities within Greenie

Some of the previously mentioned features regarding Alexa integration were removed from the final design as we were limited on the features to include in our project due to the time constraints we experienced. For the final design, we were able to accomplish the following. Once the user calls Alexa, they can activate the sensors and the pumps. Through the sensors, the user can command Alexa to show the current soil moisture,

pH, rain condition, humidity, and temperature levels on the LCD screen. The user can also command Alexa to either turn on or off the water pump. The commands previously mentioned belong to the system, being three unique commands to Greenie. These commands can be voiced by the user at any point as long as both the system and the smart speaker are connected to the same Wi-Fi network. The three unique commands included are, explicitly, “Alexa, turn on the water pump,” “Alexa, turn off the water pump,” and “Alexa, turn on X display.” The “X” in “X sensor” represents any of the four different types of sensors contained in the system.

5.3.10 Solenoid Valve

The Solenoid valve that we will be using for the project is the Digiten DC Water Solenoid Valve. This Solenoid Valve was chosen mostly because it is made from plastic, which would ensure that the pH of the water will not change as drastically if something like brass was used instead. It seems as though the minerals have a tendency to mix in with the water after having used it for a long period of time, so to ensure that this does not happen, plastic would be a better material to use as the valve part of the solenoid valve. To use plastic, finding one that would have a high tolerance to temperature is needed since having the plastic deform would cause problems to the flow of the water coming through the tube. The valve part has a high tolerance for temperature making it an ideal choice for use in high temperature environments, it will ensure that this solenoid valve stays the way it is.

It was also desirable to find one that would fit the tubing we acquired, so the ¼” hole is a perfect size to use as a solenoid valve. This will allow the water pressure to stay more consistent, since using a 1” hole will make the stream be much wider and will not allow enough pressure to flow through the tube. The next reason was because out of the different solenoid valves, this one was rather inexpensive at about $7.50 per valve which is great since there will be multiple herbs being used in the project. The plan is to have ten different herbs in Greenie, so having the tight tubing makes it so that water can flow well throughout the system without having much trouble to make this happen. The low cost that is associated with these valves allows one to test if the part will be useful or not. This valve also operates in DC, which is in line with the rest of the components in the projects.

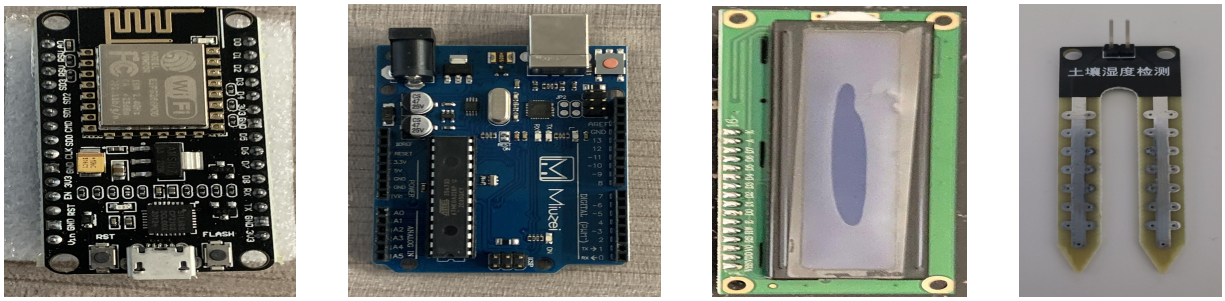
When we were preparing to build our system, the solenoid valve we desired was out of stock. We replaced the Digiten DC Water Solenoid Valve with the HFS 12 V DC Electric Solenoid Valve. It is a normally closed valve that is composed of an electromagnetic coil, which is energized to generate a magnetic field that controls the input and output ports. The solenoid valve has a flow aperture of 2.5 mm and a power of 8 W.

5.4 Parts Acquired

Below, the parts that we have selected for our prototype and that we have received so far can be observed. Images of all the components have been combined in a collage manner titled **Figure 11:** Parts Purchased. Given the current pandemic, meeting in person has become a bit difficult for our team members so we decided to take individual

pictures of our components instead of an image with all of them together. The components were divided among us group members, making each of us responsible for specific parts, mostly for testing the preliminary conditions of our components. In **Table 18**, a number label for each component has been created along with the name of the part in order to be more easily read.

1 2 3 4



5 6 7 8

9 10 11 12

13 14 15

Figure 11: Parts Purchased

Number Label	Part Name
1	HiLetgo ESP8266 Development Board
2	Miuzei Board with Atmel ATmega328P
3	LCD1602 Module (with Pin header)
4	KeeYees LM393 Soil Moisture Sensor
5	Teyleten Robot LM393 Rain Sensor
6	DHT-11 Temperature and Humidity Sensor
7	Solar Power Bank 10,000mAh
8	Bojack Voltage Regulator Assortment Kit
9	Quickun Pure Silicone Tubing
10	LEDGLE Mini Submersible Water Pump
11	PH0-14 Sensor Module + PH Electrode Probe
12	2 5V One Channel Relay Modules
13	Gikfun 3m Silicone Tubing 2mm
14	Gikfun 12V DC Dosing Pump
15	HFS 12V DC Solenoid Valve

Table 18: Number Label and Part Name of Components Acquired

5.5 Potential Product Design

Throughout the process of researching our project, several potential designs were considered for Greenie. The design of our product is important given that we must understand the architecture of our product to better meet the needs of users. The prospective design of Greenie takes into consideration aesthetics as well as small-garden efficiency. In **Figure 12**, the potential front view of our finalized product can be observed. Greenie will consist of three different sections enclosed in one rectangular container. These three different sections are the “Herb and Soil” encasement, located at the top of the container, the “Electronics,” located on the left side, and the “Water” enclosure, located on the rightmost side of the container.

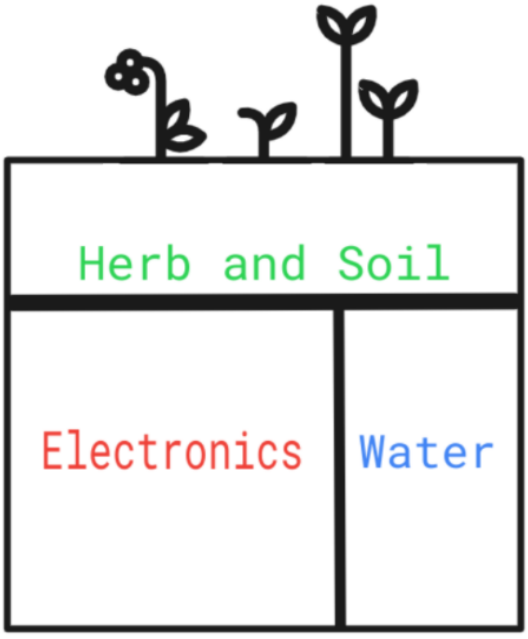


Figure 12: Potential Front View of Product

As we built our system and integrated more features into our design, such as a nutrient pump, our product's design had to change. As of building, Greenie consists of four different sections enclosed in two rectangular containers. These four different sections are the "Herb and Soil" and "Water" encasements, located on the top container, and the "Electronics" and "Nutrients" encasements, located on the bottom container. In **Figure 13**, the final front view design for Greenie can be observed.

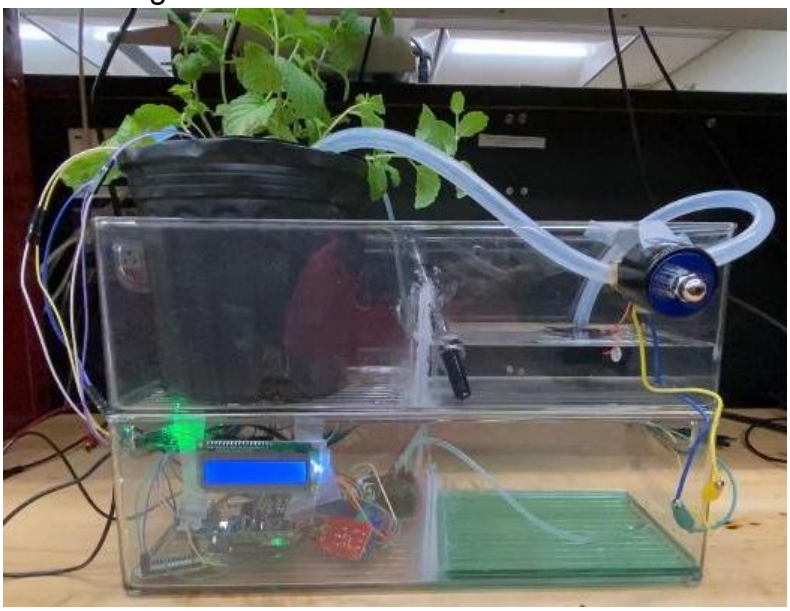


Figure 13: Final Front View of Product

5.5.1 Herb and Soil Compartment

As seen in **Figure 12**, the “Herb and Soil” encasement is going to house the user’s selected herb out of the 10 supported by our software. The user is going to be able to fill the case with their selected soil type, allowing the user to personalize. All of our sensors, as well as the outlet tubing of our water pump, are going to be located in this section.

Our sensors consist of three soil moisture sensors, one rain sensor, one pH sensor, and a humidity and temperature sensor. In **Figure 14**, the potential top view of our planter can be observed. The three soil moisture sensors are going to be inserted into the soil and placed as near the plant as possible, to ensure that the soil around it is fitting for its growth and health. The pH sensor is going to be inserted into the soil as well and located close to the plant and the water tubing to make sure that the pH levels from the soil and water are ideal. The rain sensor is also going to be placed near the plant as our system must inform the user if their herb was properly watered by the rain if placed outside. Lastly, our humidity and temperature sensor is going to be placed in the enclosure as well. Given that this sensor measures the humidity and temperature of the room, it can be placed at a distance from the herb. Space for the outlet water tubing is also taken into consideration. The water tubing is placed on the rightmost side of the “Herb and Soil” compartment since this placement ensures that the tubing can reach from the “Water” compartment located below.

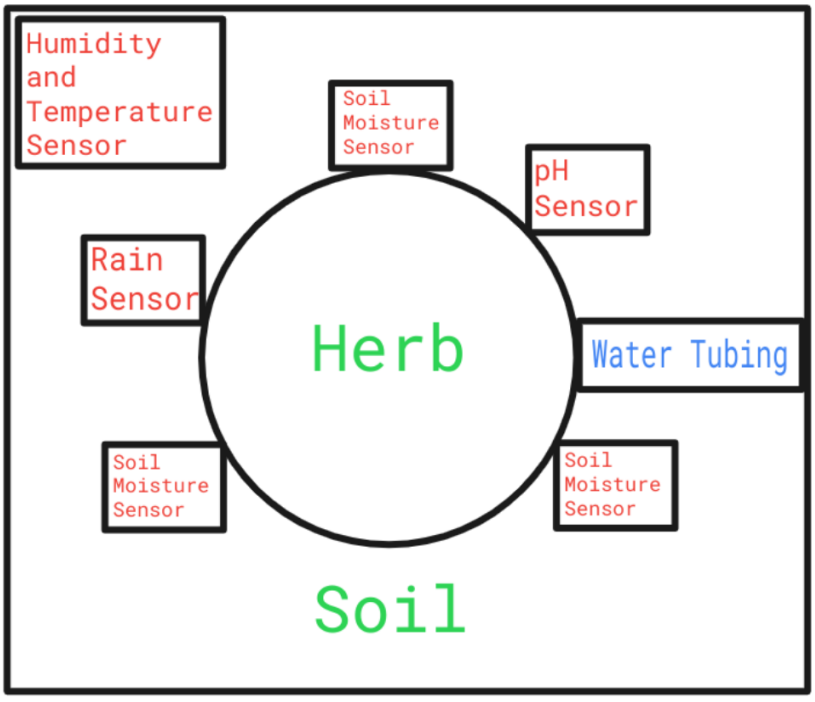


Figure 14: Potential Top View of Planter

Due to changes made in the design, two of the soil moisture sensors had to be cut. This resulted in our system only having one soil moisture sensor instead of three, as shown in **Figure 14**. This change was made since more space was needed on our PCB to fit the pins necessary for new additions, such as our nutrient pump and the relay modules required for it to function with our voice commands. The pH sensor was also moved from this section to the “Water” compartment, to ensure that the pH of the water supplied to the herb was appropriate.

5.5.2 Electronics Compartment

In the “Electronics” section, the parts and components required for our project to function are going to be placed. In this compartment, we are going to be placing our PCB, our microcontroller, our Wi-Fi module, our relay modules, our voltage regulators, and any other common electronic parts needed, such as resistors, capacitors, inductors, etc. Our LCD screen, external power supply, and smart speaker are going to be located outside, but close to, our enclosure so that the user can easily access these parts when needed.

For our sensors and power adapter, we are going to be drilling holes into the compartment to make sure that the cables required for these components fit properly and can reach their specified areas. Our sensors must be connected using long enough wires that will allow the sensors to be planted in the “Herb and Soil” section above. The same approach must be applied to our power adaptor as it also needs to reach the wall outlet specified by the user. Special care must also be taken for our LCD screen, as it must be placed outside for the user to be able to read the data being shown on it while not letting it get wet. We made accommodations for it once we started constructing this design.

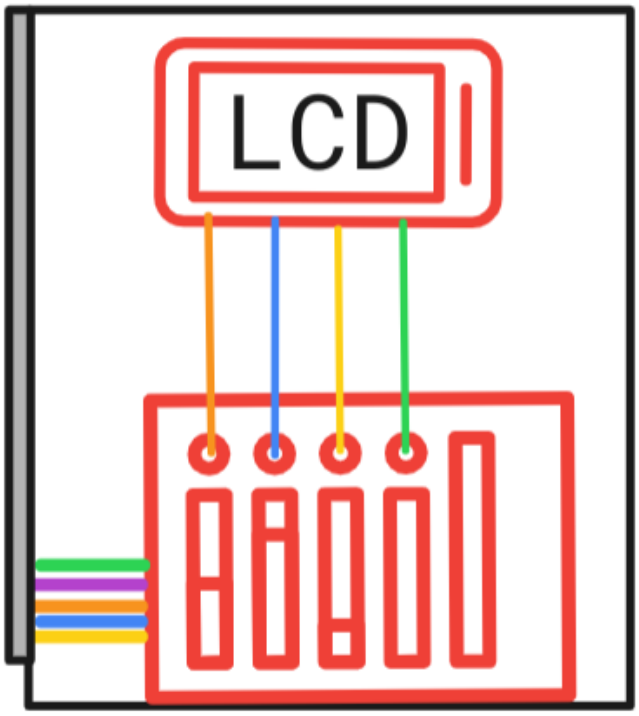


Figure 15: Inside View of “Electronics” Compartment

In **Figure 15** above, the inside view of the “Electronics” compartment can be observed. As previously mentioned, all of our electronics except for our smart speaker and LCD screen are going to be placed inside this enclosure. The LCD will be connected to our microcontroller as seen in **Figure 15**. It will be mounted onto the front of the compartment by drilling holes into the enclosure. The wires necessary for connection are going to be passed through the holes. To ensure that the LCD screen stays in place, we are going to glue it onto the plastic enclosure. The cables needed for our sensors are going to be placed inside a hollow PVC pipe, represented by a gray rectangle in our Inside View Figure. This is going to be done for neatness and aesthetic purposes, as it would not be ideal for the wiring to be convoluted.

Given the changes made in our design as we built it, extra features such as drilling holes for wires to go through enclosures were not necessary. Instead, our enclosure included handles that were also used as access points, making it easier for the components on the bottom to reach the compartments at the top.

5.5.3 Water Compartment

In the “Water” encasement, the water required to irrigate the herb is going to be contained. In this compartment, we are going to place our inlet/outlet water pump. To make sure that the herb is properly watered, we are going to be drilling a hole the size of our water tubing on the upper, rightmost side of the container. The hole will be drilled in the upper, rightmost part of the planter to allow our water tubing to reach the herb

located above and allow as much water as possible to be contained. This is a concern as it would not be ideal for the user to constantly concern themselves with refilling their water source. To allow access to the electric components and water pump located in the “Electronics” and “Water” compartments respectively, we plan on incorporating access doors on the backside of this encasement.

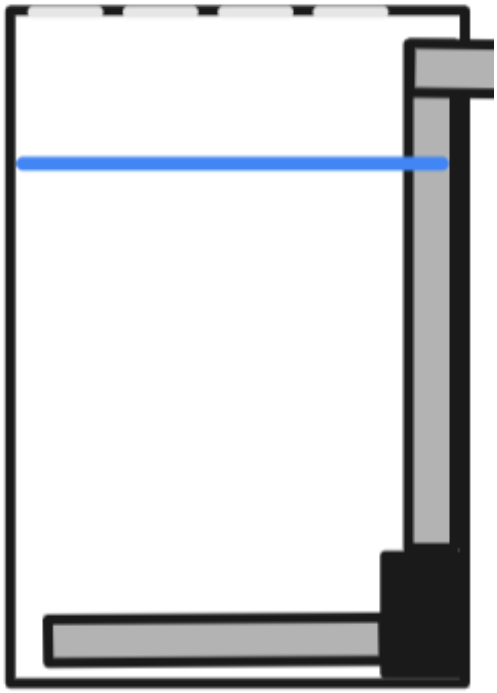


Figure 16: Inside View of “Water” Compartment

In **Figure 16** above, the inside view of the “Water” compartment can be observed. As previously mentioned, the drainage holes are going to be drilled in the plastic dividing the “Herb and Soil” and the “Water” compartments. The water pump along with its inlet and outlet water tubings can be seen in **Figure 16** as well, with the water pump being the black box and the tubes being the grey rectangles. Looking at the upper right part of the container, it can be observed that the outlet water tube is going to leave the compartment through a dilled area to reach the herb supposed to be watered above. To ensure that the tube stays in place, small holes are going to be drilled along the tube’s path and zip ties are going to be utilized to hook the tubing onto the product.

Given the changes made in our design as we built it, drilling was not necessary, as the “Water” compartment was moved to the top container, making it easier for the components to reach the herb.The solenoid valve included in our system is not submergible so it was placed outside of the compartment but still connected to the water pump tubing.

5.5.3.1 Water Drainage

Watering is a crucial step in growing and maintaining a plant. Therefore, being knowledgeable in watering and water drainage is important for anyone that wishes to create a garden, no matter the size or environment it is in. When watering a plant, occasionally the water can be puddled in the upper part of the soil, something that usually happens since the substrate cannot evacuate the excess water. Whenever this occurs, the roots suffocate in the puddled water, resulting in the roots becoming rotted. This is a grave condition that can easily kill plants. To prevent this from happening, it is important to facilitate water drainage.

One of the biggest problems that can arise when planting is a lack of holes at the bottom of the planter. Holes allow water to easily drain, allowing air to reach the roots. The most important thing is that the holes in the base of the container are kept free of obstruction, as this would result in water not being able to properly flow outward. To prevent rotting from occurring to the user's plant, we are going to be drilling small holes on the bottom of our "Herb and Soil" encasement. These tiny holes are going to be located on the rightmost part of the enclosure, to ensure that the water flows from the soil to the water encasement and not the electronics. Given that the drained water will flow back into the "Water" enclosure, this valuable resource will be reused.

Due to time constraints, we were not able to implement this idea. This was one of our priorities if we had more time to implement more functions.

5.5.3.2 Water Detection

Water detection is a crucial factor that must be addressed for our product. If proper water detection is not set up, the user is going to constantly have to open the "Water" compartment through the access door being placed in the back of the enclosure. This action can become tedious which is something we want to avoid, given that we are designing a smart irrigation system. Poor water detection can not only become a monotonous activity for the user but it can also be dangerous to the health of the plants. If the user cannot observe the water levels, they might forget to refill, leading to the health of the plants being compromised due to lack of watering.

To avoid these issues, we are planning on designing the "Water" compartment with the use of clear hard plastic. This is going to allow the user to visualize how much water is currently stored in the enclosure. The use of clear plastic is also going to allow the user to not concern themselves with refilling the container until it is needed. Given that separate pieces of plastic are going to be glued together to create this compartment, the use of waterproof glue and clear silicone is crucial since we do not want water leaks.

5.5.3.3 Water pH Maintenance

Our pH sensor is going to determine what pH the soil of the plants are. We will also provide the users with a way to change the pH if there is an issue with the level of the pH that the soil is at. Many different materials exist to raise or lower the pH of the soil,

by mixing these with the water we can allow the user to change the pH should the user have need of it.

There will be a separation in the existing water container that contains water mixed with aluminum sulfate. Aluminum sulfate allows us to lower the pH of the soil when it is mixed with the soil. It is recommended that you need about 6/10 of a pound of aluminum sulfate per 0.5 drop in pH in every 10 square feet of soil.

To raise the pH of the water, it is recommended to use potassium carbonate mixed in with water. We will separate the water in the same way and have a mixture of the two so that the pH of the soil can be raised. You can use about one tablespoon per gallon of water.

5.5.4 Build Material

As previously mentioned in our Project Specifications (refer to section **2.4.3 Specifications** and **Table 1**), our planter prototype is not going to exceed the stated numbers for dimension and weight. Greenie is going to be less than or equal to a dimension of 20” x 20” x 20” and less than or equal to a weight of 15 pounds. To not exceed our previously stated specifications, we are going to utilize a lightweight but durable material to build the enclosure.

When investigating which build material we should utilize for our product, we came across several different options. The possible build materials we found were terracotta, wood, metal, plastic, fiberglass, concrete, and fabric. Each one is briefly discussed below by stating their advantages and disadvantages.

Terracotta Containers
Terracotta containers are planters made with clay. These are characteristic because of their classic warm brown color, which has maintained its popularity throughout the years. Terracotta planters are durable as long as they are being taken good care of. Although they are a staple in almost every garden, terracotta containers are extremely heavy, especially when filled with soil. Not only are they not portable, but also can break easily if they are dropped.

Wood Containers
Wood containers are one of the easiest planter choices as they can be cut and modeled after any shape and form, allowing users to custom make their pots. If quality wood is utilized and is taken proper care of, wood planters can last for a long time. Unfortunately, wood containers are quite fragile as the wood needs to be taken care of constantly by either regularly coating the wood with sealant or making sure that the soil is always dry. If not, the wood can easily decay and possibly even rot.

Metal Containers
Metal containers are often used today as pots as they can be quite stylish. Metal planters allow users to get creative with their pots as they can be easily painted over

with either a brush or spray. Metal containers are widely available as well since any metal container can become a pot as long as it is properly repurposed. Although they can easily be found, metal planters are usually not recommended for outdoor use in many hot climates since the metal can become very warm. The high temperature surrounding the plant can dry the soil quickly, requiring the user to constantly water the plant, which may result in damage.

Plastic Containers
Plastic containers are middle tier when it comes to aesthetics as they can either look nice or poorly made. Plastic pots are the most practical of our options since similar to metal, plastic containers can be found anywhere and they can be easily repurposed. Plastic pots can be fun to own as they are commonly used for many DIY projects. Plastic planters can come in any shape, size, and color, and are also extremely lightweight and resistant. They are also one of the cheapest options discussed. Care must be taken when using plastic pointers outside as the sun may gravely damage them, leading to cracks in the container.

Fiberglass Containers
Fiberglass containers are one of the most all-around flexible pots as fiberglass is extremely adaptable. Fiberglass planters can be made to look like terracotta, wood, and even concrete pots. This is due to fiberglass containers being produced with fiberglass fibers that are easily shaped into different things. Fiberglass planters are just as tin and light as their plastic competitors. Unfortunately, these containers tend to be on the pricier side and are easily worn down when exposed to severe conditions.

Concrete Containers
Concrete containers are top tier in terms of aesthetics since concrete is a very desirable material for decorating. Concrete is an extremely durable material and given its look, it is used in many houses today to create both outdoor and indoor gardens. Although concrete pots are very popular as of late, they are exceptionally heavy, making them not an ideal build material for portable planters.

Fabric Containers
Fabric containers are a new trend in the gardening world. These fabric pots claim to be beneficial to plants as they provide breathability for their roots, preventing them from rotting due to the accumulation of water in their soil. Given that these planters are made of fabric, they are thin, lightweight, and durable. Many fabric containers are being used today in the form of laundry basket lookalikes. These are used to house big plants and/or small indoor trees.

As of now, the best material for the build of our product is hard plastic. Plastic is one of the most practical materials as it is lightweight, durable, and can take on any shape, size, and color. In terms of price and availability, this material is inexpensive and easily available in stores.

The containers selected include built-in, easy-grip side handles that make it easy for the user to transport from indoor or outdoor space. The handles are also being used as access points, making it easier for the components on the bottom to reach the compartments at the top. The bins are clear, making it ideal for the user, as they can constantly observe the water and nutrient levels.

5.6 Potential Scalability

As previously stated in the scalability constraints section of this report (refer to section **3.1.11 Scale and Time**), the scale of our project has been minimized for individual use. However, Greenie is being made with the idea that it can be expanded, and if chosen to, it can be set up to irrigate large amounts of land for farmers, optimizing irrigation through the use of our sensors and software. Not only would it be able to irrigate agriculture fields, but also any other green areas such as parks and golf courses.

To show how our product can be enlarged, we are going to utilize the golf course expansion design. Given that Greenie is incorporated with IoT technology, it could support the high costs of maintaining green areas and it could ensure optimal green area conditions throughout every season. In terms of our hardware, many of our components can be purchased in wireless versions that can communicate through technologies such as Bluetooth. The soil moisture sensors included can be buried a certain distance below ground in different zones of the course to ensure that soil moisture is properly and evenly kept throughout the field. The same approach can be utilized for rain and pH sensors. A vast number of humidity and temperature sensors can be supported to make certain that the air surrounding the field is to the needs of the green areas. The data acquired from these sensors can be wirelessly sent via Bluetooth to local and/or central stations dedicated to the upkeep of courses. Most golf courses utilize recycled water to irrigate the entire area but even so, this water expenditure accounts for an estimated 312,000 gallons per day[12]. This is one of the several reasons why golf courses benefit from the use of smart irrigation since this would positively impact factors such as resource wasting.

A second area where our project could be expanded is in the amount of produce supported. For our project, we chose to only provide support for 10 herbs in our database. This is due to scalability and cost constraints as well as microcontroller storage constraints. If a microcontroller with bigger storage was utilized, the database within our system could be expanded to support more herbs or even more types of products, for instance, fruits, vegetables, and roots. If desired, support for flowers could also be included.

Finally, Greenie could also be expanded in its reachability. At the moment, our product is only accessible through the use of a web application. However, if desired, our project could be enlarged to provide support through a mobile application as well. Similar to the web app, the mobile app can be accessed by any mobile device, whether it is a phone or tablet. The mobile app would have to be downloaded through an application store as opposed to a web app, which can be accessed by simply inputting a link onto a web server’s search bar.

6. Prototype

In this section, we will be discussing the preliminary model of our product. We will use our prototype to test that our electrical components and software programming operate as they should. Given that our project is still in development, this section is not as advanced as it will be at the end of this course. First, we will discuss the facilities and equipment needed to test our product and its components. Then, we will talk about our Printed Circuit Board (PCB), the software we have decided to utilize to build it, the beginning process for the design, the layout, and the manufacturers that we will look into to construct our PCB. After, we will discuss the build of our product, meaning how the hardware components will be connected and how they will interact. We will do the same for our software components as well.

6.1 Facilities and Equipment

To create a working prototype for Greenie, we needed to ensure we had the proper facilities and equipment to test our materials. Concerning location, at the bare minimum, we needed to have an open area where all of our materials can be placed and constructed. Given our goal was to make our project as portable as possible, there were several places we could use. One example was taking a large table set up on an outdoor porch where we would place all of our materials and begin working. A benefit to utilizing a location like this was that it also allowed us to test the weather integration portion of the project. Another example was indoors within an apartment with a similar table setup. A benefit to utilizing a location like this was that it allows us to avoid the possibility of our materials getting wet from the rain. Given that we live in Florida and at the time of writing this report we're in the thunderstorm season, avoiding rain can prove to be crucial.

As for equipment, there are some key materials that we needed to ensure our product was working properly. First, a breadboard. With a breadboard, we would be able to construct samples of our circuit without having to have a fully constructed printed circuit board ready. Additionally, with the number of electrical components we have for our product, we were going to need at least a medium-sized board. Second, we needed a digital multimeter. With a digital multimeter, we would be able to verify that we are getting the necessary output voltages and currents outlined for our requirement specifications. This is important because in the case where we did not have enough power to supply our components, we had to re-evaluate not only our specifications but also our battery selection. Third, we needed a soldering iron. With a soldering iron, we would be able to attach several pieces to our printed circuit board. This was important because to have a polished final product, we needed to have several parts integrated into our design. Materials such as the breadboard were purely made for testing and were not viewed as part of the final design. Lastly, we needed an oscilloscope. With an oscilloscope, we would be able to get a visual representation to verify that our

specifications are being met. This was important because it also acted as a backup to our readings from the digital multimeter. In the case where we just used one of these tools, there was a possibility that an error could be overlooked.

On the other hand, we also took advantage of the location and materials in UCF's senior design laboratory. The senior design lab is located in Engineering building 1, room 456 on the UCF campus. This location was uniquely beneficial for our group because it was a familiar place that all of us easily have access to. Additionally, it was beneficial to us because once again it gave us the advantage of avoiding the rain. Materials-wise, the senior design lab gave us everything we needed and more. Provided by the UCF Electrical and Computer Engineering department's web page on the senior design lab, the following are some of the materials included in the lab:

- Tektronix Oscilloscopes
- Tektronix Dual Arbitrary Function Generators
- Tektronix DMM 4050 Digital Multimeters
- Keithley 2230-30-1 Triple-Channel Power Supplies
- Dell Precision 3420 Computers
- SMD Rework Station
- Soldering and Desoldering Stations
- Digital Microscope Inspection Station

Overall, we decided not to give ourselves unnecessary limits on facilities and equipment during the prototype phase of our project. The locations mentioned above were just a few of many places we thought about going to. What was most important to us was that we had an ideal environment to test our prototype. In sections **7.1.1** and **7.2.1**, we discuss more on the hardware and software test environments respectively.

6.2 Printed Circuit Board (PCB)

The Printed Circuit Board, or PCB, is the core of the vast majority of technology products today. PCBs are found in everyday items such as cellphones, computers, and car alarms. Our PCB is going to support and connect our electronic components, with copper paths or tracks, so that our circuits, and therefore, our product, work as desired. Before building our PCB using software, we are going to model it using a breadboard. Building our prototype on a breadboard first is extremely beneficial as breadboards are not permanent like PCBs, meaning that on a breadboard, we can move parts around and change circuits as needed. We are going to utilize our breadboard for investigation purposes and once our final design is complete, we will be moving on to creating the PCB.

6.2.1 Software Utilized

To successfully create the PCB that we need for our project we must utilize PCB designing software. When investigating which designing software would work best for our project, we came across several different options. The top three programs we

considered were Autodesk Eagle, DesignSpark PCB, and KiCAD. Below, we will discuss all three options.

Autodesk Eagle
Autodesk Eagle is the most popular PCB designing software. It contains a schematic editor, which allows users to design circuit diagrams, configure different components, and perform routing on circuit boards. Eagle also includes one of the most substantial libraries on any PCB designing software[3]. The program is not entirely free to use, but students and hobbyists can download the limited version for free. Autodesk Eagle is compatible with Windows, Mac, and Linux, and it is recommended for all users, whether they are beginner, intermediate, or advanced.

DesignSpark PCB
DesignSpark PCB is a free-to-use PCB designing software. It contains an easy-to-use interface that allows users to perform circuit capture as well as automatic arrangement for circuit boards and layouts[3]. DesignSpark can also perform 3D modeling of PCBs. DesignSpark PCB is only compatible with Windows and it is recommended for intermediate and advanced users only.

KiCAD
KiCAD is a cross-platform, free-to-use PCB designing software. It contains a circuit editor that allows users to easily create and build their designs. The program allows up to thirty-two layers of copper to be used and similarly to DesignSpark PCB, can also perform 3D modeling of PCBs[3]. KiCAD is compatible with Windows, Mac, and Linux, and it is recommended for intermediate and advanced users only.

For our project, we ended up going with Autodesk Eagle. The main reason why we have chosen this software is its familiarity. All team members have utilized Eagle before for one of our previous courses, Junior Design. In the course, we were introduced to the software by having to create and build a PCB for our final project.

6.2.2 Design

Given that we have chosen to utilize Autodesk Eagle to create and build our PCB, the schematic must be designed using the software's environment. To successfully create our PCB using Autodesk Eagle, we are going to follow the following steps: understanding which components need to be in the design and how they need to be connected, create the PCB's schematic by placing parts on a blank template, insert drilling holes in the schematic, add routing to traces, add labels to components in the schematic, and print the internal layers.

A few design factors have to be considered when constructing a PCB. In our case, we need to be mindful of the number of layers, the size, and the components that will be placed, as these are all factors that decide the price of constructing a PCB. As of the moment, it is assumed that two layers will be needed to be successfully printed. These two layers are the upper plane and the ground plane. In terms of size, our PCB needs to

be as small as possible while still containing all of the components that we need for our product. For the components, most of them were chosen to be through-hole since it would be easier to control what was happening when soldering ourselves, some pieces like the voltage regulator were left to be soldered by the company we sent the schematic to.

6.2.3 Layout

For the PCB, the parts will be placed around initially based on being able to connect to each of the parts as closely as they can. We will try to make this part as clean as possible in order to make it so that figuring out what ports and wires lead to where is more easily visible.

In **Figure 17,** the connections of the pins of two Atmega328Ps to the various components in the device can be observed. There is a section for the pin headers that go to various sensors our project needs, along with some leds to show the power is flowing correctly. There is also an area that has some things like the voltage regulator, the oscillators, and the usb. In Autodesk Eagle, there is an autoroute tool that can be used that is helpful as a starting point and so that it can be a guide to how the board should be routed. By cutting down the length of traces and the number of intercepts, we can reduce electromagnetic interference and reduce the cost of the board.

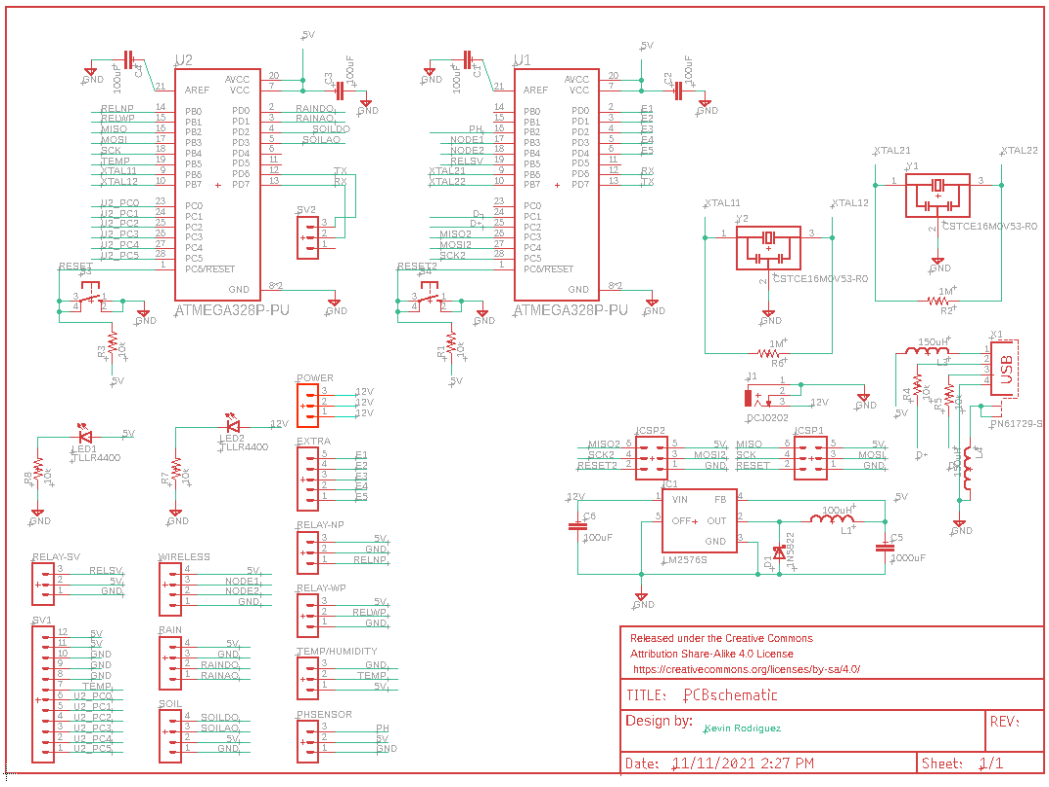


Figure 17: Printed Circuit Board (PCB)

The second iteration of the PCB that shows the top layer wiring in red and the bottom layer wiring in blue can be seen from **Figure 18** below**.** This positioning aims to decrease the total size of the board so that costs can also be reduced, we got it down to 67.31 x 69.85 mm. We took care to make sure no traces overlapped on the two layers and reduce the amount of distance all traces travel.

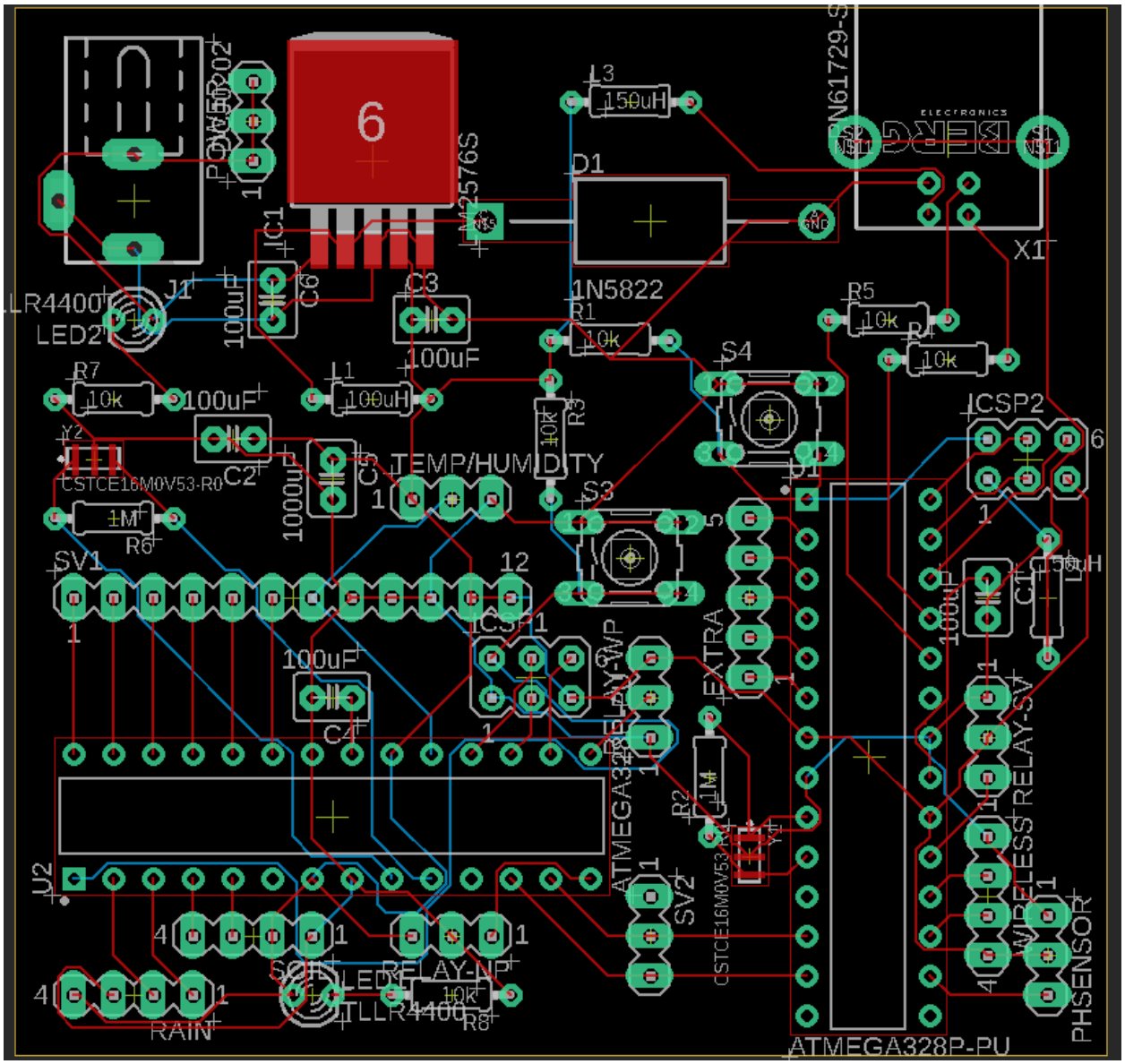


Figure 18: Printed Circuit Board (PCB)

For the design, the engineers wanted to switch to the Atmega2560 due to the larger quantity of pins, however, since this piece has been out of stock, two Atmega328Ps were chosen instead. As seen in **Figure 18**, the PCB board layout incorporates support for the LCD, all sensors included in the system, the relay module, the water and nutrient pumps, the Wi-Fi module, a port for the usb, a DC jack and the voltage regulator.

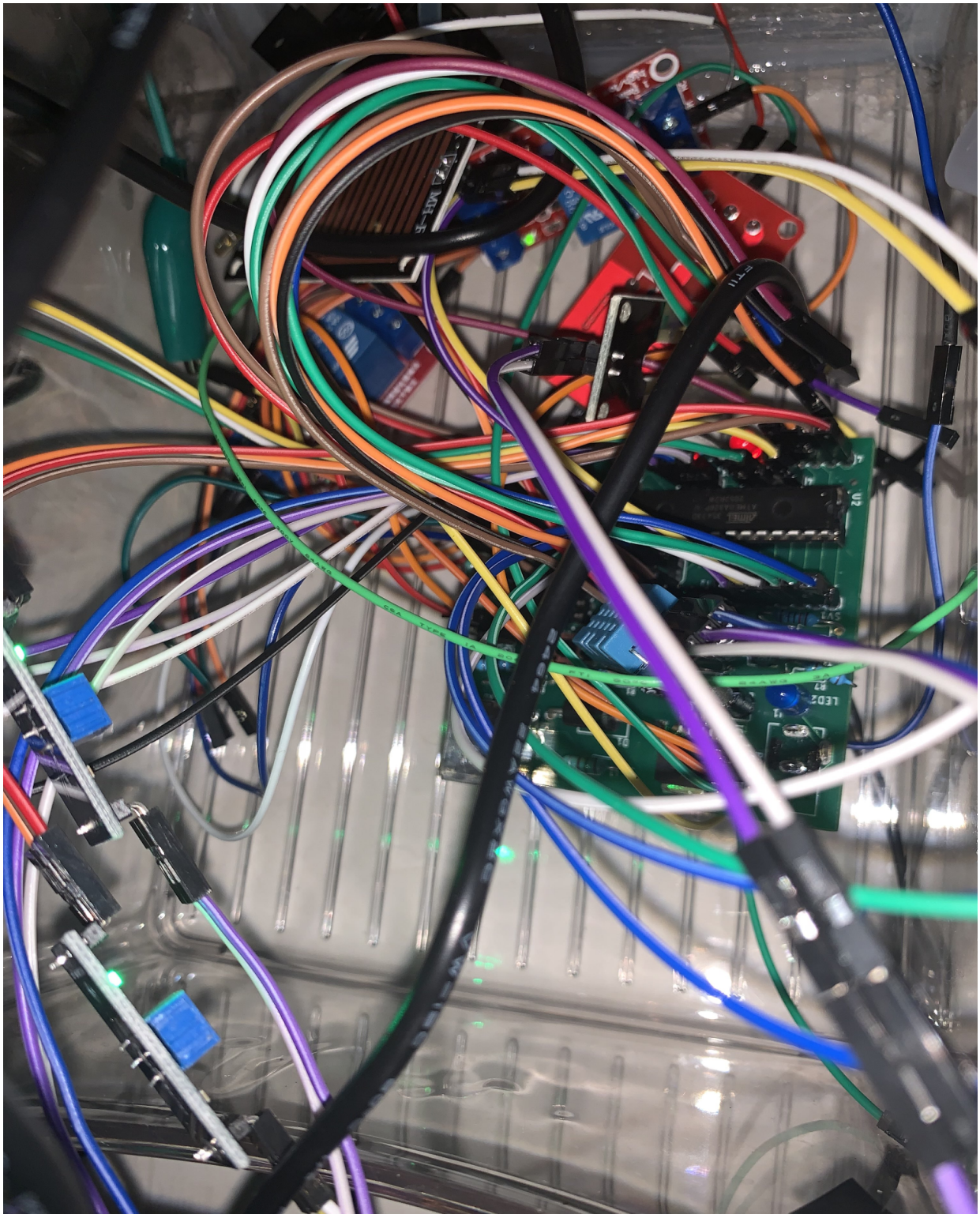


Figure 19: PCB with wiring to sensors

As seen above, **Figure 19** shows the physical version of the PCB. All of the parts were successfully placed and worked as expected with the exception of the LCD that had trouble connecting and sending data to the PCB.

6.2.4 Manufacturing

The last step of creating the PCB that we require for our product is manufacturing it. We must send our design to a company that will convert the software board to a physical board. Given the current pandemic, factors such as product availability, manufacturing, and shipping times have been severely affected. Taking these into consideration, we must choose a vendor that is preferably within the United States to reduce shipping costs and time. Once our PCB is ready to be printed, we will contact different PCB manufacturers to acquire information such as manufacturing quotes, shipping costs, and estimated delivery times. Although our goal is to keep the cost of the PCB as low as possible, we are willing to make sacrifices for the product if these affect the efficiency and reliability of the board.

6.2.4.1 Potential Vendors

When investigating which potential vendors would work best for our project, we came across a few different options. The top three vendors we are currently considering for the manufacturing of our PCB are Advanced Circuits, JLCPCB, and PCBWay.

Advanced Circuits
Advanced Circuits is the third largest printed circuit board manufacturer in the United States[7]. All orders receive a free engineering file review before going to fabrication; this prevents ending up with unusable boards. Also, they hold the best on-time shipping record, which enables good planning. Advanced Circuits offers 24-hour computer-aided manufacturing (CAM) engineer service, ensuring any questions about the product are answered. They do not have bundle purchase requirements, so we only buy what is needed. There is no tooling charge for standard specifications as well as no repeated charge for orders with custom specifications.

JLCPCB
JLCPCB (Shenzhen JIALICHUANG Electronic Technology Development Co., Ltd.) is an economic circuit board manufacturer while still offering high-quality PCB technology suitable for industrial standards. JLCPCB is the biggest PCB manufacturer in China, specializing in the fast creation of PCBs. They also specialize in the production of small PCB batches. Similar to Advanced Circuits, they offer a 99.97% on-time delivery rate and 24/7 online service[10]. Also, they have a 0.23% quality complaint rate. JLCPCB features an “easy-to-use” online ordering system.

PCBWay
PCBWay has more than a decade of experience in the field of PCB creating and manufacturing[11]. PCBWay offers services for PCB prototyping, SMD/SMT (solder paste) stencil, and PCB assembly. This manufacturer provides the best value, manufacturer

direct pricing, low minimums, and quality customer service. Similar to both competitors mentioned above, PCBWay offers a high on-time delivery rate of 99%. Even though they are located in China, PCBWay provides fast 24 hour turnarounds.

The vendor that was eventually chosen was JLCPCB. The parts were completed quickly and effectively, there were no major defects found on the boards.

6.3 Build

Here, we will discuss the build of our product, meaning how all of the components will be connected and how they will interact. It is important to understand the build of the project so that it can be implemented properly. Proper implementation of our product is crucial, as it should work perfectly once it is presented.

6.3.1 Hardware

The microcontroller unit is the component that is going to be driving every piece of the project. A steady supply of power is needed so that the MCU can be on at all times to give and take information while doing all the necessary tasks of keeping the plants watered and healthy. In this project, we have various sensors that will be collecting information and feeding it to the microcontroller. The various sensors are humidity and temperature, pH, rain, and soil. Information like this will be collected so that it can then be displayed on the LCD screen. The next component is the water pump, this piece will ensure that all the plants get the necessary amount of water to them. Finally, the voltage regulators will help make sure that each of these pieces is getting the right amount of voltage.

6.3.2 Component Mounting

The PCB is a very important part of our project, and when talking about this piece, we need to know how the PCB and its components are going to be held in place. There are three different methods to keeping the components in a safe and secured place, they are by standard mounting bosses, custom mounting standoffs, and slide-in mounting rails.

Standard Mounting Bosses
In this method, components are pre-formed along with being the easiest and quickest methods of mounting. This method makes it so that there are protrusions located along with the PCB, these holes have threading so that the mounting boss screws can be screwed until they are secured. They are about ¼ of an inch tall which makes them desirable because of how little space they take up, though there is a way to decrease this size further if need be.

Custom Mounting Standoffs

This is the second method of mounting. This method is for when you need specific locations for the mounting bosses, however, doing this would raise the price by a substantial amount for the project. The cheaper alternatives to this method are making plastic standoffs that can be used with the circuit board. There is an adhesive that allows the standoff to stay in place securely. This can be done where there would be no screws used, allowing for a smaller size.

Slide-In Mounting Rails
This is the final mounting method that could be used. On the inside of the enclosure, rails exist so that the circuit board can slide into them. This is mostly used when having multiple circuit boards and is easier since there are no screws involved.

The method that our group ultimately used is just the very standard surface mounting and through hole mounting method. The surface mounting was done by hand by two of the group members while the surface mounting was left to JLCPCB because of issues that arose when surface mounting on the first iteration.

6.3.3 Arduino Integrated Development Environment

The software that we are utilizing for most of the hardware components is the Arduino Integrated Development Environment. The Arduino IDE is an open-source programming software that allows users to write code in C or C++ and upload it to any Arduino compatible board. It is compatible with Windows, macOS, and Linux.

The Arduino IDE has several features which led our group to choose this to program our hardware components. One of its features is the board module options which allows users to choose which board they are using for their project. When another board is added or when modifications are made, the port data is automatically updated. Another feature of the Arduino IDE is direct sketching, wherein users can do sketches within the text editor. The text editor provides users an interactive experience through its additional features. It also lets users decide whether to document their project or not. This feature will help track our progress and all the modifications that are made. Even if this IDE is specifically intended for Arduino boards, it can support boards from other developers as well with the help of a third-party hardware. The Arduino IDE contains hundreds of libraries that can be used by users. This will allow our software developers to save time in programming our system's hardware components. Our system's microcontroller, sensors (soil moisture, rain, temperature and humidity, and pH), LCD screen, and relay module will be programmed using the Arduino IDE.

6.3.4 Software

6.3.4.1 Software Connecting to Hardware Overview

For connecting the software to the hardware, we used a combination of the Arduino IDE, the Atmega328P, and the ESP8266 to communicate with the other elements of the

web application. As expected with any web application there are buttons throughout the application. With each button click, we triggered events to reach the hardware. For instance, for turning on the water pump, we took a button click on the web application and sent that to the Wifi Module and from there to the microcontroller. Then from the microcontroller we went on to control the physical components and triggered it to turn on or off however we like. By setting up the connection in this fashion, we made the manual process as simple as possible for the user. In the case of automatic watering, which was what we recommended to our users, the connection is similar but a bit more controlled on the developer's end. As seen in **Figure 20**, for selecting automatic watering, additional checks needed to be performed to get something like the water pump to turn on. If the soil moisture level is already too high or rain is being detected from the rain sensor, then a notification will be sent to the user and the water pump will not turn on. In the final design, the only check that ended up being utilized was the soil moisture check, but for the sake of seeing the original intent, the figure can be observed.

As for the ESP8266, another way it came into play was with the Alexa Integration. Using the same structure with buttons throughout the application we caused an event to trigger that reached the Wi-Fi module and eventually the microcontroller. Then from the microcontroller we controlled the relay module attached to the water pump. Of course, the only substitution that needed to be made was a button click for saying the phrase "Alexa. Turn on the water pump."

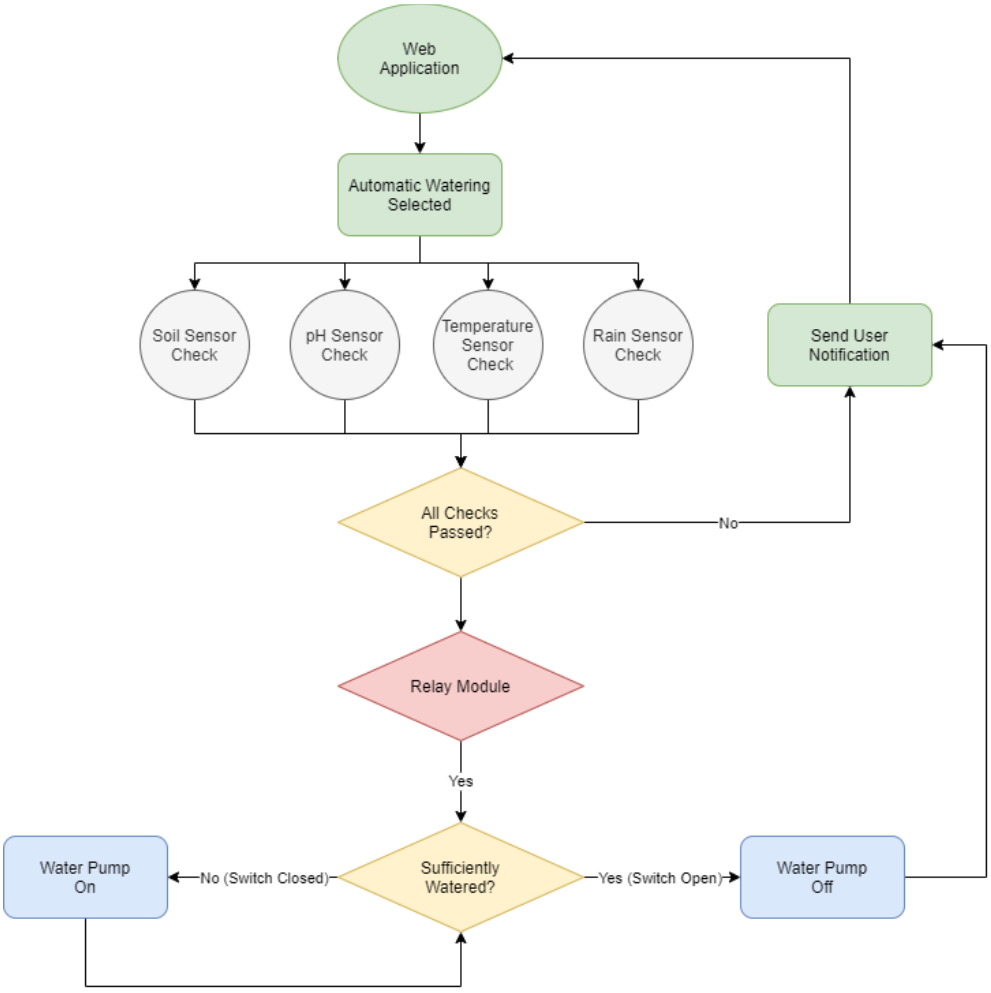


Figure 20: Automatic Watering Structure

6.3.4.2 MERN Stack Inspiration

For creating the web application, we originally planned to create it in the model of a MERN stack application. While working on the project, we decided to switch away from the MERN stack and as a specific structure and use it more as an inspiration. For our design, we ended up using pure HTML, CSS, Javascript, and Firebase. Which does contain some elements of the MERN stack, but are not all encompassing of what the MERN stack is. To gain a better understanding of the MERN stack we can look at it in parts. A MERN stack is one example of a full-stack web application where the client and server sides are fully managed. The M in MERN stands for MongoDB. MongoDB is a document-based NoSQL database that stores any data you have. So, things like users, statistics, or pictures can be stored for convenience. For Greenie, we plan to use our database to mainly store statuses and statistics. The E in MERN stands for Express(.js). Express is a server-side framework and its purpose is to make writing code for Node.js simpler. It's useful for working with APIs (application program interfaces) and handling various HTTP requests and responses. The R in MERN stands for React(.js). React is

the framework used for the front end/client side and its purpose is to present everything that the user will see. It's useful for managing all of the HTML and CSS that go into front-end programming as well as all of the error handling and events. Lastly, the N in MERN stands for Node(.js). Node.js is the JavaScript runtime environment that allows the user to build and run the application. It's useful because it is the main piece that keeps everything in the stack together.

By removing any of the NodeJS portion of the stack, the whole thing falls apart. We know this because there are other stack variations where the other components are interchangeable. Take the FERN stack and the MEAN stack for example. In the FERN stack, MongoDB is replaced with Firebase with Node.js still at the center. In the MEAN stack, React is replaced with Angular(.js), but Node.js is still at the center. When it comes to our selection though, we decided to go ahead and make that stack fall apart so that we could have a web application that is interactive with our hardware.

Looking at **Figure 21**, we can get a visual representation of how the MERN stack is connected. At the top of the stack, we see the front end of the application, in the middle section we see the server/back end of the application, and on the bottom, we see our database.

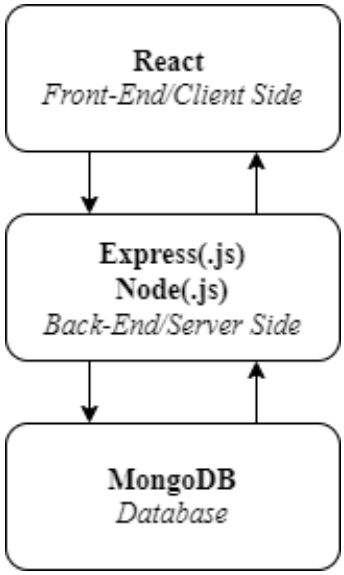


Figure 21: MERN Stack Structure

In general, some benefits that came with using the MERN stack as an inspiration for our web application are the following:

- Spent more time into learning javascript as it was the main component the stack revolved around
- The entire development cycle was covered (front-end/back-end)
- Technology was free and open-source
- Not a large learning curve
- Strong performance and speed

6.3.4.3 Sensor Checking

On a more low-level overview of how the software was utilized in the Arduino IDE, one component to address for our project was the checking process for our sensors. Given that our sensors are key to automatic watering, we needed to outline the criteria being used to decide not to turn on the water pump.

We started with the checks that involved no interaction on the user's behalf. The soil moisture check and the rain sensor check. With both of these checks, if they failed in the watering process then the user simply had to wait until further user interaction was needed. Looking at **Figure 22**, we can see that in the soil moisture check that there are specific ranges assigned to dry, well-drained, and moist and when the threshold is crossed the water pump will no longer be active. We can also see a similar structure with the rain sensor in **Figure 23**. Ultimately in the final design, only the soil check was implemented due to time constraints, but for the sake of seeing the original intent, the rain diagram can be observed.

As for the checks that did involve user interaction, we go to the temperature, humidity, and pH sensors. For both of these, neither of their checks were implemented either, but their diagrams can still be observed for our original intent. With these sensors, the type of notification sent to the user was supposed to specify that something needed to be done to allow for a proper watering process. With the temperature and humidity sensor, if the user was in a location where the temperature was 30 degrees Fahrenheit and the temperature needed for the herb is 70 to 90 degrees Fahrenheit, then the user was supposed to be prompted to move to a location where the temperature is in the desired range.

Then with the pH sensor, in the case where the soil was not in the proper range for herb growth, the user was supposed to be prompted with a notification to confirm if watering is desired. With the soil sensor and rain sensor, this type of additional confirmation was not necessary. When it came to both checks in general, we did not want to stop the user from watering the herbs if they are unable to do something like move to a new location, but we did feel it is best to let the user know the herb was not being cared for given their situation. In **Figure 24** and **Figure 25**, we outline the very nature of these sensor checks.

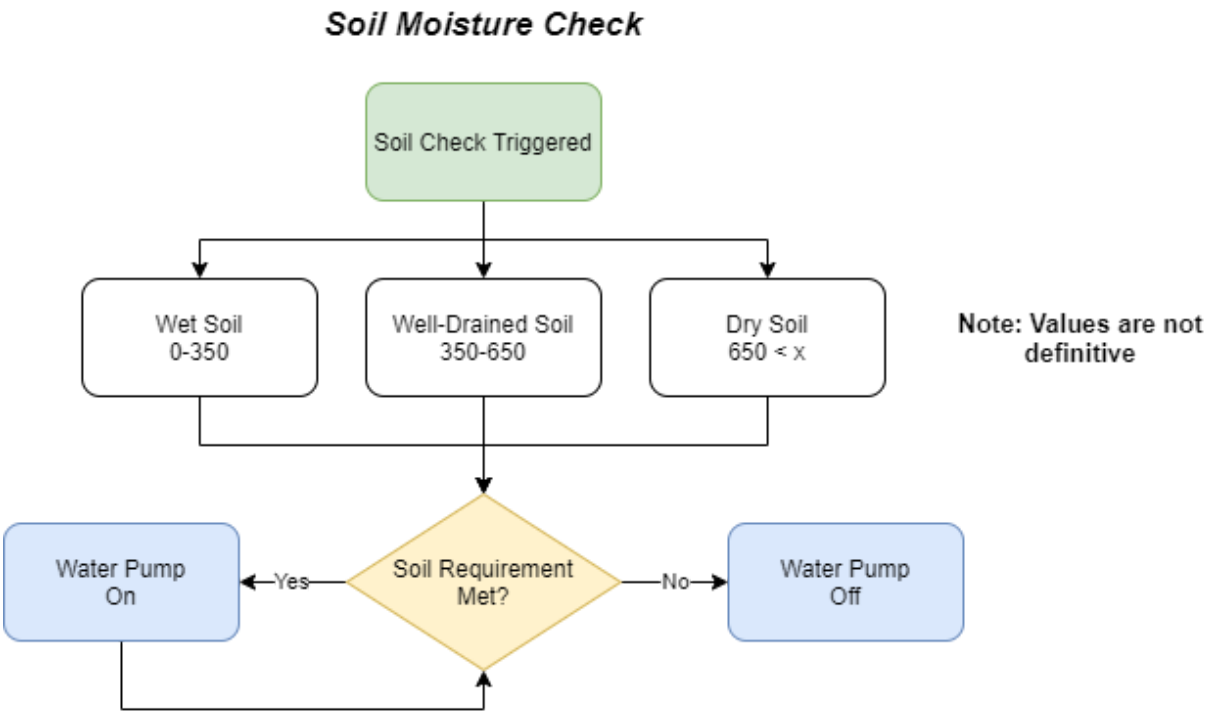


Figure 22: Soil Sensor Check

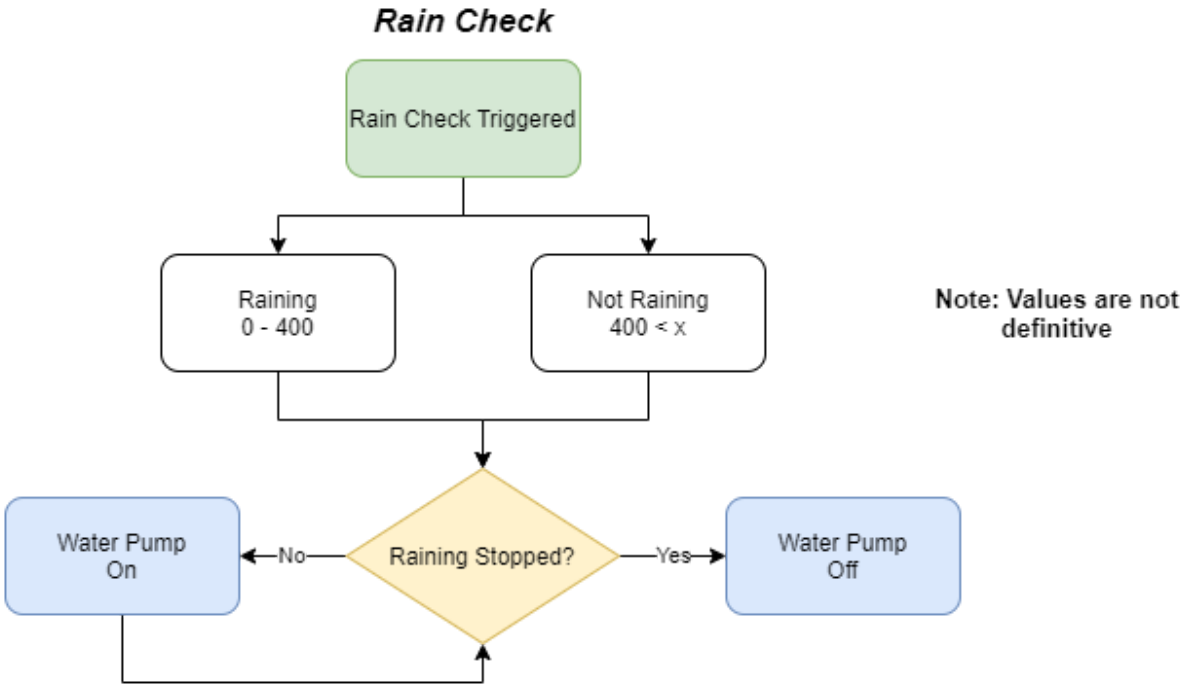


Figure 23: Rain Sensor Check

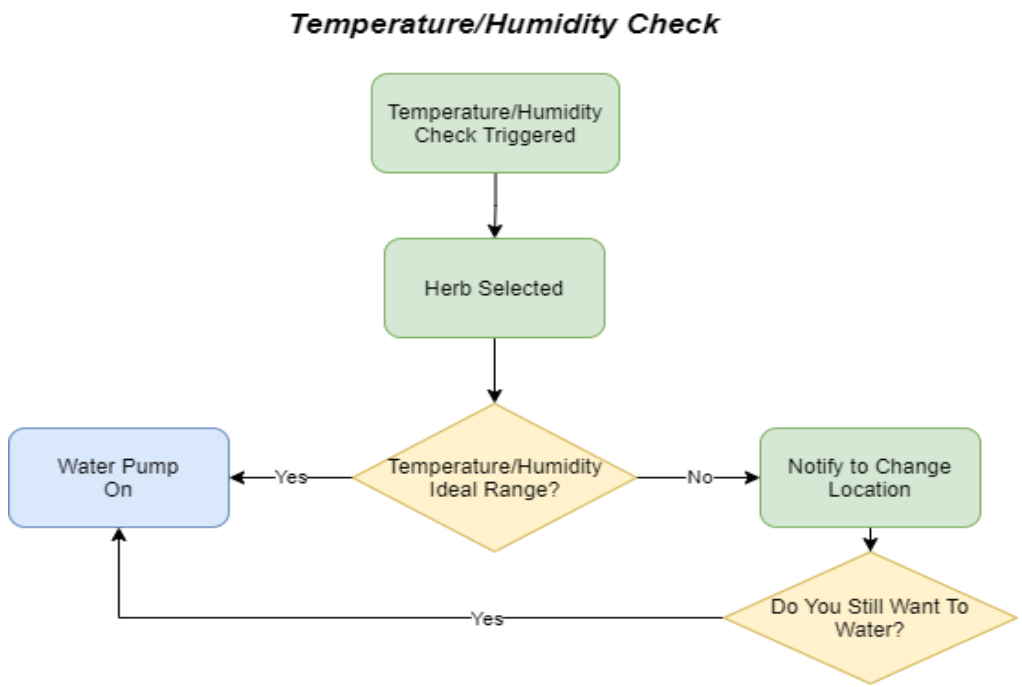


Figure 24: Temperature and Humidity Sensor Check

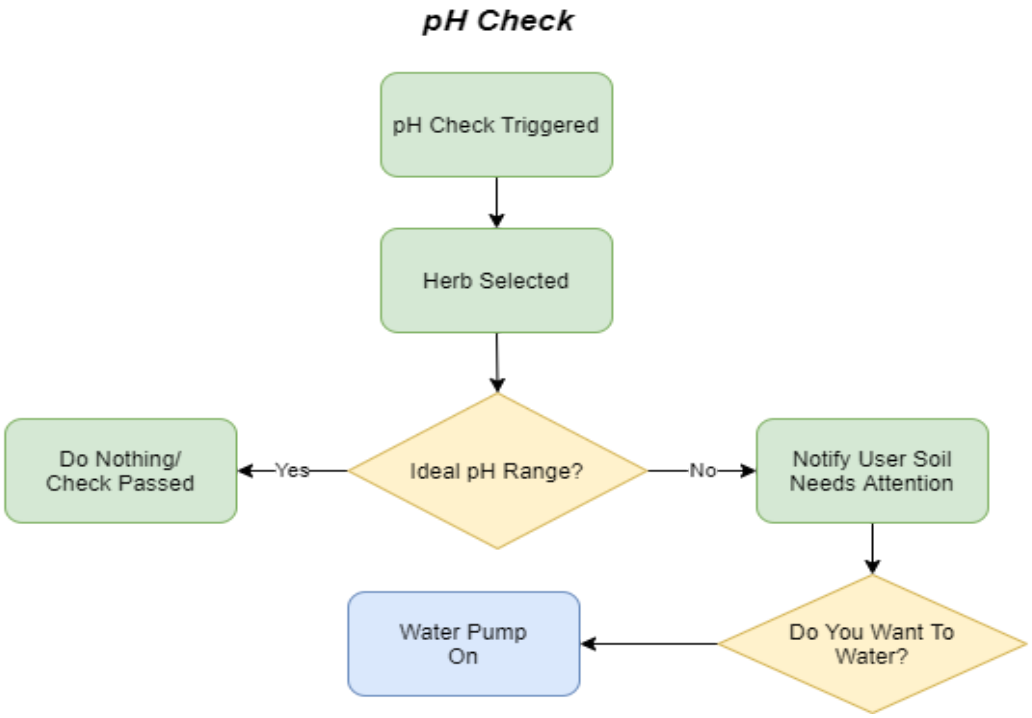


Figure 25: pH Sensor Check

6.3.4.4 LCD (Software to Hardware)

Another component to look at on a low-level is our LCD. For our prototype, we were orignally looking to include a large variety of information. Our goal was to get the data from our sensors working and save the other elements for later. Ultimately, this did not end up happening. The only functionality that we decided to get working with the LCD were the Alexa commands. This decision was made simply because the time constraint was too great. Given more time, we would've implemented these plans into our LCD. Looking at **Figure 26** we can see an overview of what original intent was in utilizing the LCD. In the figure, we can see that for each sensor that we were using, we were going to assign a value to it and check its value to determine what message that should be displayed. From there we loop what's on the screen so the data is continually being updated in real time.

For elements like the number of times water has been dispersed in a day, we can see in the figure that counters were going to be used. So with every iteration the water pump switches on, the counter would increase and we could reflect that value on the LCD. The value would then reset at the end of the day. Lastly, for the elements like the amount of water dispersed, we can see that there was a bit more involved.

In the case of determining the appropriate amount of fluid ounces dispersed, we were going to use time and in conjunction with a counter to reach a value. We were going to start by finding out how much water was dispersed for having the water pump on for one second. Then we take that value and use it to correlate to how much time the water pump is on. So starting with a value of zero, the moment the water pump turns on, we are counting the seconds and adding up the number of fluid ounces with each second. This process would then go on throughout the day and the user would be able to see the total amount of water dispersed at different times of the day. At the end of the day, the counter value that was being used for display on the LCD would be set back to zero. For the information we decided to implement with the LCD in the finished product, we were going to use a format similar to this structure and make modifications as time goes on.

In **Figure 26** below, the LCD software to hardware configuration can be observed. We begin with the welcome screen that introduces the users to Greenie. After staying on the welcome screen for a certain period of time, the text would automatically scroll to the first screen where the users can see the soil moisture sensor readings. Here the user could verify that the herb is getting the water it needs and that the automatic watering process is going smoothly. Next, the text on the LCD would scroll to the second screen where they could see the number of times their herb has been watered. Here the user can then verify that their herbs are being watered to the correct amount. Finally, the text on the LCD would scroll to the third screen where they can see how much water has been dispersed from the water pump. Here the user could verify that Greenie is not pumping out excess water. After iterating through all of the screens, the LCD would loop back to the welcome screen and begin the process again. This way the user can continually observe the LCD throughout the day if they desire. In **Figure 26**

below, the process explained in this paragraph can be observed in a flowchart manner, making it easier for the reader and/or user to follow.

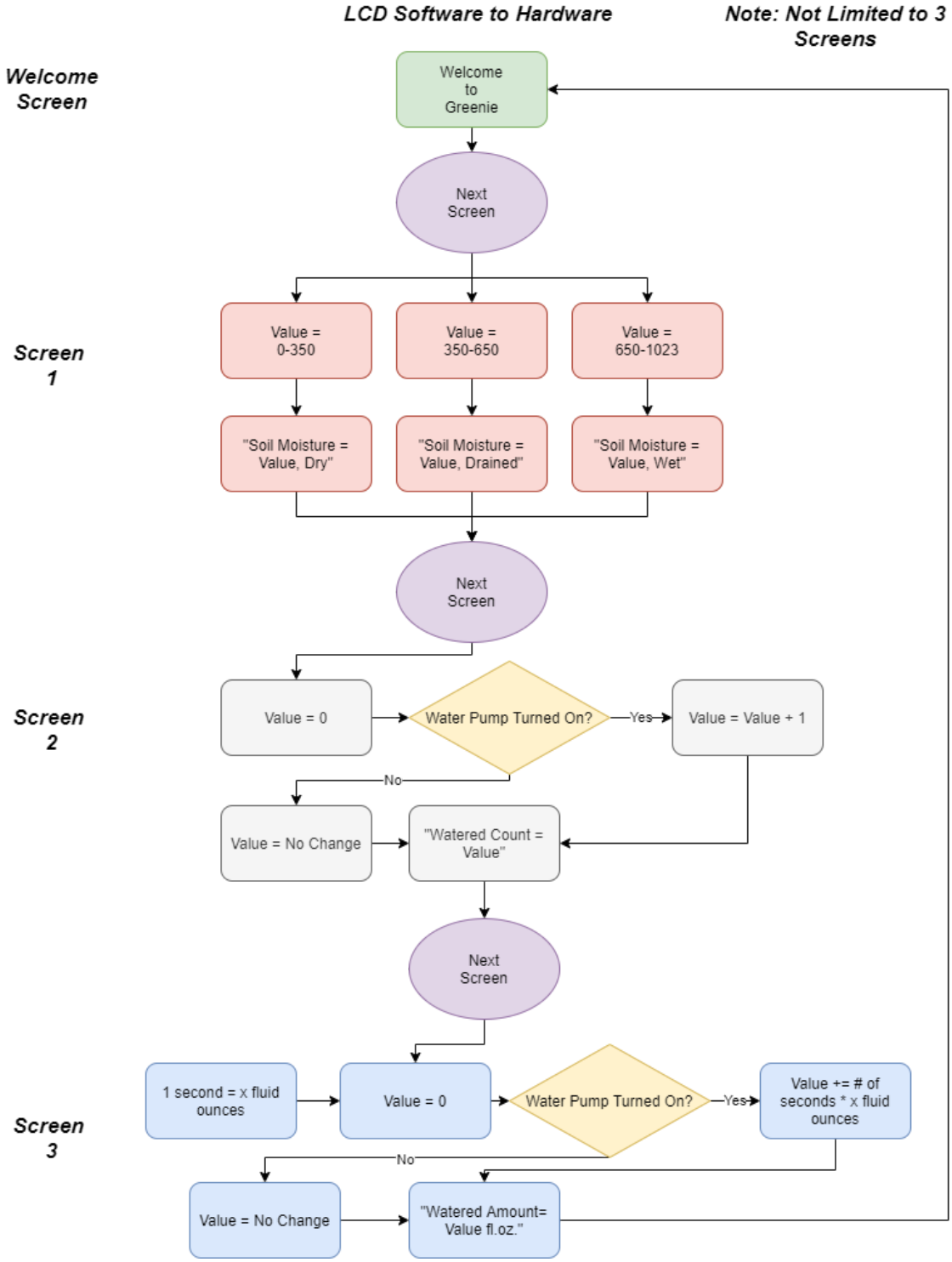


Figure 26: LCD Software to Hardware Configuration

6.3.4.5 Relay Module (Software to Hardware)

For working with our relay modules on a low-level, we needed to select pins on our Atmega328p microcontroller and use those to control the state of the devices connected to the other ends of the modules. Taking the water pump as an example, say we use pin 6 on our microcontroller. Connecting all 3 components together, we could control the water pump by altering the state of pin 6 and the relay module will act as a middle man. As for how the state is altered in this example, we needed to set the pin 6 value to either “high” or “low”. By doing this, we switched the circuit open and closed respectively.

To ensure the relay modules operated properly in our prototype as part of our automatic watering structure, we needed to ensure that designated sections and time delays were set up. With designated sections and time delays, we gained the ability to isolate different watering patterns for the ten herbs we specified. So in the case where we have an herb that needs the water pump to be on for 5 seconds at a time repeatedly, we could create a small section within the main loop that has a delay value of 5000 milliseconds. Of course with an herb like this though, the soil would eventually become flooded. For our herbs, our main use for sections and time delays were to itemize how long the water pump should stay on. So, when the water pump had been on for 5 seconds, the relay module would be set to high and it should not be closed again until one of our conditions has been met (i.e soil moisture is dry or enough time has passed). Most of the time for our project, the condition would be dry soil, but in the case where more frequent watering is needed, we recognized that having a timer for frequency can act as a good back up.

6.3.4.6 Wi-Fi Module (Software to Hardware)

To obtain a better understanding of how the Wi-Fi module will work with the other components on a low level, then we need to expand on the usage of libraries in the Arduino IDE. So, in the Arduino IDE given that we coded in a structure similar to c we started by obtaining libraries by using the standard “#include”. From there we went into the different options available for creating our prototype. To clarify, in working on our prototype, our plan was to go through the different libraries we’ve collected and try out each one to see which yields the best results. Based on our research, we know that with different libraries comes different methodological approaches and with each approach typically comes some undesired outcomes. Our goal was to sort out each of those outcomes and find the ones that reduce the amount of code that needs to be written, and that meet our requirement specifications. So for instance, in the case where there’s a library that accomplishes some of our tasks but it's alexa capabilities only work for certain devices or works for 2 commands, then we had to put that library lower on our priority list. Or in the case where there’s a single library that can do things that would normally require 2, we would put that single higher on our priority list.

For our Wi-Fi module, there is a library available specifically catered to it called “ESP8266Wifi.h”. By using this library in conjunction with libraries like the “WiFi.h” , “SoftwareSerial.h”, and “FirebaseESP8266.h”, we were able to secure a connection and ensure that data was being properly transferred to and from the web application. So in the case of our sensors for example, if we wanted to ensure that our data is readily available on our web application then we would establish a connection, send information from our atmega 328p microcontroller to the ESP8266, and then that have the data posted in Firebase. By using the libraries in this fashion we were able to get the best use of hardware and imitate a stack-like software structure.

For implementing Alexa integration, one approach available was using the Wi-Fi libraries mentioned above along with some external tools. One example of an external tool was with the “skills” service that Amazon provides to their customers. With the “skills” service, Amazon gives their users the ability to create custom commands for alexa that make the user experience more enjoyable. With these custom commands, they can be paired with different kinds of devices and they can all be mapped to web services like Heroku. Ultimately, we did not end up using this approach, but using an external tool such as this would have been beneficial in the sense that it’s advanced capabilities went beyond our desired specifications. Given our goal was to have at least 3 unique Alexa commands. The downside to using this methodology would have been that there was a large time commitment needed for setup.

An alternative library that could be used was the “espalexa” library. With this library, the user would essentially be able to work specifically with alexa and get our devices connected. The library works in its own fashion and can do several unique things. The benefits that come from using this library would be that it’s a convenient single library for getting Alexa set up with our devices. The downside to using it would be that it is not as comprehensive as other Alexa setup approaches. In the end, we ended up using this approach due to time constraints. However, we do encourage any future projects to look at the skills development portion of the Amazon website so even more in-depth commands could be made.

6.4 Web Application Prototype

Software prototyping is important because it enables the developers to understand the user’s requirements during the early stage of development. The feedback from the users will be helpful to the software developers as it provides them details on what is expected from the product. These prototypes are utilized during the analysis and design process of web application development. Software prototyping allows for more user involvement during the development process. It will also help us to save time as the errors and missing functionalities can be detected much earlier. When it comes to web application prototyping, horizontal prototypes and vertical prototypes are used to verify and elaborate the requirements of the application.

Horizontal prototypes provide a broad view of the web app and are used during the early stages of development. This includes the sample screen, main menu, and buttons.

These features do not necessarily need to be fully implemented. For example, a user may just navigate through the screen and once a button is clicked, they could be redirected to another page that contains sample data. These prototypes give users the idea of what will be included in the web app without the actual implementation of the features when integrated with the hardware components.

In contrast, vertical prototypes are used in the later stages of web application development. This type of prototyping focuses on the technical side of the web app. It includes connecting to the database and interacting with the subsystems and hardware components. Vertical prototypes also allow users to navigate through the screen, click on buttons, and see the actual functions of the buttons. It is useful in demonstrating that the web app's features are actually feasible.

The users of Greenie have many options to interact with the system. They can view the current statistics acquired from the following sensors: soil moisture, rain, pH, temperature, and humidity through the LCD screen that is attached to the plant enclosure. They can also use voice commands to control the system through Amazon Alexa. Lastly, the users can manage and control the system using the web application. This section describes the possible design of Greenie's web app. As we moved forward in developing the web app, the logo and colors were added to the interface. The design of the web app is very user-friendly, simple, and straightforward. The color and the theme is eye-catching but not too bright. The colors that we used on the web app were appropriate with the theme of our project. Throughout the web app we included different hues of green.

The main page of our web app displays the name of our project, Greenie: The Smart Irrigator. On the left-hand side of the web app are four clickable buttons that will take the user to the home page, the herbs page, the metrics page, or the about page. Once the user has clicked any of the buttons, the response time should not exceed five seconds. The default page is the home page or the dashboard which is divided into different sections as seen in **Figure 27**.

The sections on the dashboard/home page are classified as soil moisture level, rain level, temperature, humidity, and pH level. Each section contains an up-to-date reading from the sensors as well as the ideal range for the selected herb. We originally wanted to incorporate a bell button on the left side of the page that lists the notifications when clicked by the users. Users will receive a notification from the system when the following actions have been triggered. When the water pump disperses water to the plant, it will inform the user how much water was dispersed and when the system watered the plant. When rain is detected, the system will alert the user so that the user can decide whether to turn off the system or delay the watering cycle. When the sensors are not working, the user will be alerted which sensor is not working. When an error is detected on the system, a notification will pop up on the user's end containing an error message. Due to time constraints, we decided to not include the notification features mentioned above.

Since we are gathering weather information from a weather app called OpenWeatherMap, the daily weather forecast was initially placed on every page but we decided to just put it on the home page along with the sensor readings. The users will receive a pop-up notification that asks them to share their current location for accurate weather information that will be updated every minute. Users also have the option to enter their city and country on the search bar to view the weather information at their location. The watering schedule is based on the combined data from all the sensors. Originally, there is only one on/off button placed at the top of the main page where users can easily access it. However, since we added a nutrient pump, we decided to have a start/stop button for the water pump and for the nutrient pump. This buttons exist so that users can turn on/off the water pump and the nutrient pump manually. The diagram below is one of the prototype designs of Greenie's web app.

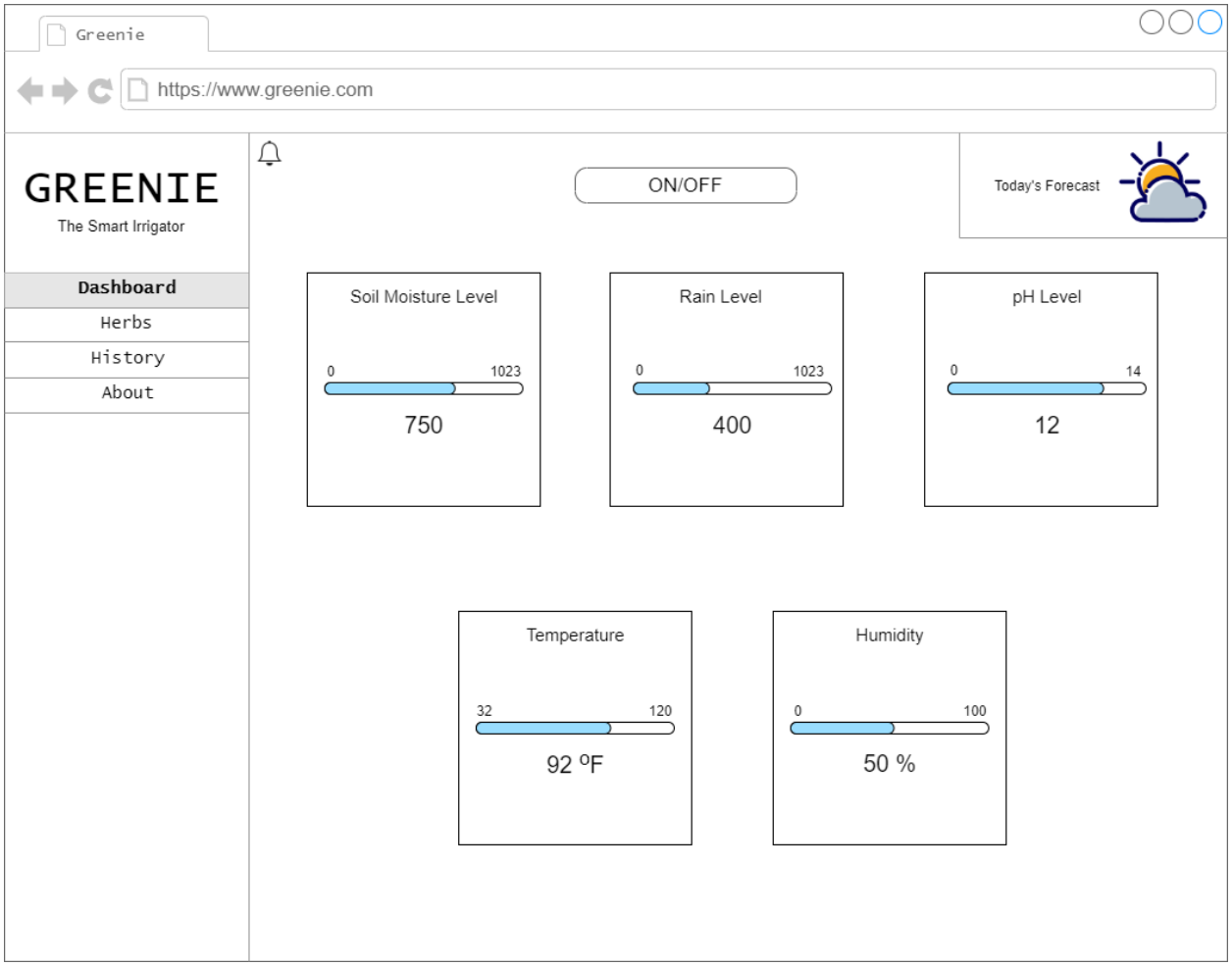
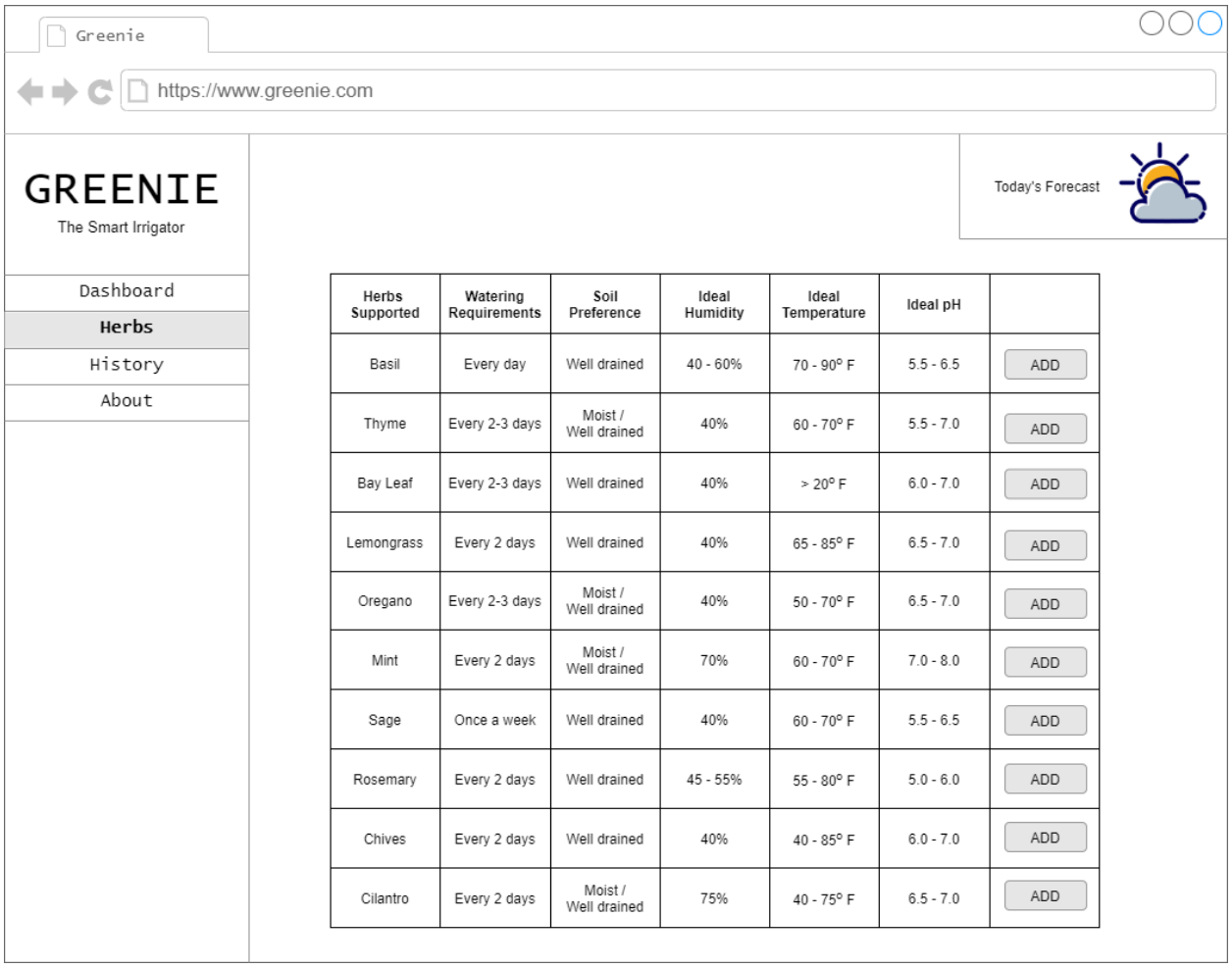


Figure 27: Prototype of the Web App's Dashboard

When the users click on the "Herbs" tab, they will be redirected to a page where they can view the list of ten herbs that are supported by our system. The herbs are organized

in a table that describes the watering requirement, soil preference, ideal humidity, ideal temperature, and ideal pH level of each herb.

In **Figure 28**, every herb has an add button so users can add the herb that they want to put into the plant enclosure. If they click the “add” button, the system will gather the information about that herb, so that the sensors within the system can function based on the needs of the selected herb. Instead of the “add” button, we used “start/stop” button for automatic watering. For example, if the user decided to put lemongrass inside Greenie, then all its information will be added to the system. All of the herb’s requirements will be used to water the herb and the ideal range of values for each sensor will be displayed on the dashboard page so that the user has reference when viewing the current data of their plant. The current reading of the sensors will then be displayed on the web app’s dashboard which will be updated every hour.



Herbs Supported	Watering Requirements	Soil Preference	Ideal Humidity	Ideal Temperature	Ideal pH	
Basil	Every day	Well drained	40 - 60%	70 - 90° F	5.5 - 6.5	ADD
Thyme	Every 2-3 days	Moist / Well drained	40%	60 - 70° F	5.5 - 7.0	ADD
Bay Leaf	Every 2-3 days	Well drained	40%	> 20° F	6.0 - 7.0	ADD
Lemongrass	Every 2 days	Well drained	40%	65 - 85° F	6.5 - 7.0	ADD
Oregano	Every 2-3 days	Moist / Well drained	40%	50 - 70° F	6.5 - 7.0	ADD
Mint	Every 2 days	Moist / Well drained	70%	60 - 70° F	7.0 - 8.0	ADD
Sage	Once a week	Well drained	40%	60 - 70° F	5.5 - 6.5	ADD
Rosemary	Every 2 days	Well drained	45 - 55%	55 - 80° F	5.0 - 6.0	ADD
Chives	Every 2 days	Well drained	40%	40 - 85° F	6.0 - 7.0	ADD
Cilantro	Every 2 days	Moist / Well drained	75%	40 - 75° F	6.5 - 7.0	ADD

Figure 28: Prototype of the Web App’s Herb page

The “About” page on **Figure 29** will contain a short description of the Greenie system including the goals and objectives of our project. A copy of the device’s manual will also

be placed on this page so that the users can access it anytime via the web app. Instead of a pdf link of the user's manual, we included a section on how to use Greenie on the about page. We also provided answers to possible questions that the users might have regarding the system and the web app.

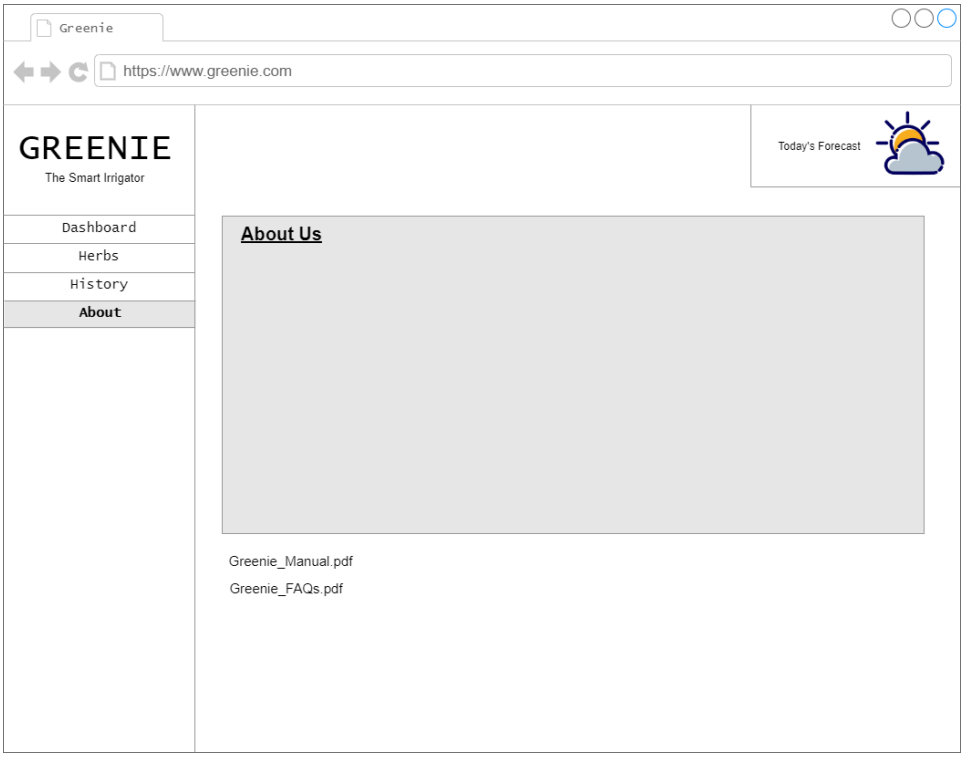


Figure 29: Prototype of the Web App's About page

The History page on **Figure 30** will provide a brief overview of Greenie's activities over the past seven days. It will contain the watering history and soil moisture level of the herb so that users are able to review the condition of their plant and see how it changes over time. The weekly history will be represented by a graph that plots the amount of water that has been dispersed by the water pump and the average soil moisture level that was collected for the past seven days. The x-axis of the graph represents the time and the y-axis represents the average value. A blue line will be used for the water that has been dispersed and a green line will be used for the average soil moisture level. The users will also be able to download the data of their plant.

Instead of a History page, we decided to implement a Metrics page. This page contains a bar graph of the total water and nutrient dispensed, as well as the water and nutrient saved while the system is running. Another graph that is included in this page is the average sensor readings from five automated readings. Lastly, this page provides users the percentage of their manual usage and automatic usage of Greenie.

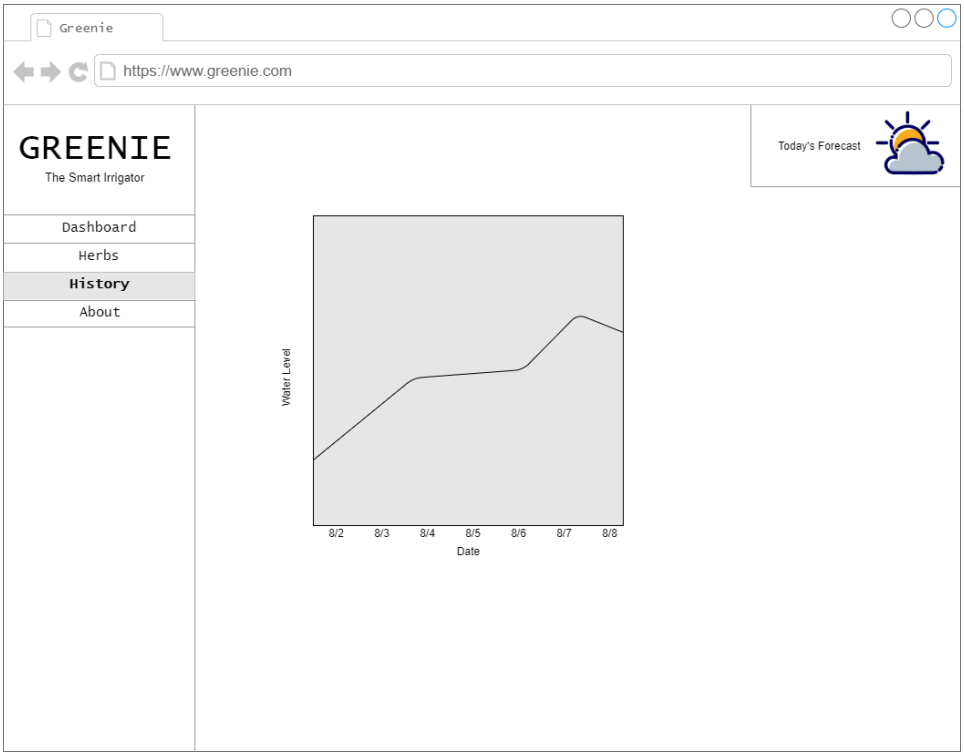


Figure 30: Prototype of the Web App's History page

7. Testing

In this section, we will be discussing the preliminary tests that we either plan on performing or have already performed on the components that we have acquired so far. Testing our components is one of the most crucial steps to making our product work. Given that our product needs to perfectly function during our final presentation, it is crucial that testing on parts is performed so that malfunctions can be observed and fixed before this deadline.

7.1 Hardware

Here, we will be discussing the preliminary tests that we either plan on performing or have already performed on the hardware parts that we have acquired so far.

7.1.1 Hardware Test Environment

There are many parts in our project that need to be tested so that we can make sure that they are performing well. All the parts that need to be tested are the voltage regulators, the battery, the microcontroller unit, the LCD screen, the relay module, the water pump, and the various sensors in the project. The main pieces of equipment that

we will be using to test these components are a digital multimeter (DMM) and an oscilloscope.

For the battery, we need to make sure that the output voltage of the battery and the number of Amps coming out of the battery are at acceptable values. To see if the water pump works as intended, we need to turn it on and make sure that it can pump out 240L/H. For the LCD1602 module, we use the multimeter to make sure that it is using the right amount of voltage of 5 V. Next, we check to see if the voltage regulators are providing the correct voltage for each part by measuring the power with a multimeter. We need to check to see if 12 V goes into the pump, about 3 V for the sensors, and 5 V for the MCU. When looking at the MCU, we need to simply make sure that it powers on and executes code the way that it should. Finally, for the sensors, we look at whether they are capturing information and sending it to the MCU.

7.1.2 Sensor Testing

The sensors must provide accurate values since the system relies entirely on the data collected from the sensors. Preliminary testing and additional testing were conducted on the soil moisture sensor, rain sensor, humidity and temperature sensor, and pH sensor. For the preliminary testing, the sensors were tested individually to make sure that each sensor works properly and provides reasonable values. The values were compared to the ideal values stored in the database (as seen in **Table 14**). After the initial testing, the sensors were tested together to verify that they are compatible with each other and with the microcontroller. The preliminary tests were done by connecting each sensor to the microcontroller and then running a code on the Arduino IDE. The code was able to read data from the sensors and output values to the LCD screen. Additional tests were done during and after the development of the system. Different methods of testing the sensors are described below.

Possible issues arose while testing the sensors. One issue that we encountered is when the LCD screen is not displaying the correct format of the output. Another issue is when the readings are completely off compared to the expected values. To fix these issues, we first had to make sure that there are no connection issues between the microcontroller, the sensors, and the LCD screen. We then checked if the pins are properly connected. Next, we had to check the code that we used to implement the test and make sure that we are using the proper functions and variables. In a case where the LCD froze during the sensor testing, we restarted the system.

7.1.2.1 Soil Sensor Testing

7.1.2.1.1 Preliminary

Testing the soil moisture sensors verify if they measure the correct volumetric content of water in the soil. This specific sensor has a potentiometer that can be used to adjust the threshold level. It also has both digital and analog outputs, however, the analog output will be utilized for our project.

The following tests were applied to the soil moisture sensor. The first step is to set up the connection between the microcontroller, LCD screen, and sensor in a breadboard. We connected the VCC pin from the module to the Arduino board and the GND pin to the ground on the Arduino. Since we are using analog output, we connected the analog data pin to A0 on Arduino. Then, the probe was inserted into two different soil samples. Before putting the probe in the soil, we have to make sure that the probes are dry and clean so that the reading will be accurate. The first soil sample was completely dry while the other soil sample was wet. A code was executed on the Arduino IDE that outputs the current moisture level of the soil. It contains a loop function that reads data from the sensor and displays a value on the LCD screen. It reads analog values from 0 to 1023. The LCD screen displayed a value within the range and the value varied when placed in different types of soil, therefore we were able to confirm that the soil moisture sensor was working.

7.1.2.1.2 Additional

The soil moisture sensor was tested on our system similar to the preliminary test, but this time we used one of the herbs from the collection of herbs that we have predefined. An herb was chosen to test the sensor under two conditions. For the first condition, we inserted the probe into the soil of the herb when it was dry. The second condition includes watering the herb and then inserting the probes in the soil. To confirm that the sensor works with Greenie, the system should be able to read data from the sensors and display the reading on the LCD screen and the web app. The reading was observed so that we can see how the values change under different conditions. The output values were compared to the values on the table from our database that contains information about the water needs of different herbs. If the output value is between 0 and 350, then the soil is dry and needs to be watered. If the output value is greater than 650, then the soil is wet and does not need to be watered. The ideal moisture level for most herbs would be between 350 and 650. This triggers the system to water the plant when the user chooses automatic watering via the web app.

7.1.2.2 Rain Sensor Testing

7.1.2.2.1 Preliminary

The rain sensor was tested to ensure that it can detect raindrops accurately. Before testing the rain sensor, the microcontroller, LCD screen, and sensor were connected properly. We connected the VCC pin from the module to the Arduino board and the GND pin to the ground on the Arduino. Since we are using analog output, we connected the analog data pin to A0 on Arduino. An initial test was done by placing a few droplets of water on the sensor board. If the D0-LED lights up, then the rain sensor is working. For further testing, we placed the rain sensor under three conditions. The first condition was when the rain sensor was completely dry. For the second condition, we sprinkled a few drops of water on the rain sensor board. For the last condition, we submerged the sensor board in water. A code was executed on the Arduino IDE which contains a loop that reads analog values from the sensor and maps the values. Originally, the LCD

screen should display “No Rain” when the sensor board is dry, “Moderate Rain” when the sensor has a few drops of water, and “Heavy Rain” when the sensor board is completely soaked with water. Instead, we had the LCD screen display the amount of water drops. We observed how the output on the LCD screen changes under the three conditions.

7.1.2.2.2 Additional

The same conditions above were used to test if the rain sensor works with our system. In addition to displaying the output on the LCD screen, Greenie should be able to delay the watering schedule when rain is detected. When there is enough rain detected, the system will be triggered to turn off the water pump to avoid water waste and overwatering of the herb.

7.1.2.3 Humidity and Temperature Sensor Testing

7.1.2.3.1 Preliminary

The humidity and temperature sensor was tested to ensure that it provides the current temperature and humidity accurately. This sensor was tested in two different environments, indoor and outdoor. Before doing the tests, we made sure that the wiring between the microcontroller, LCD, and sensor is connected properly. We connected the VCC pin from the module to the Arduino board and the GND pin to the ground on the Arduino. Since we are using analog output, we connected the analog data pin to A0 on Arduino. We then placed the sensor indoor where it is colder and less humid. We executed a code on the Arduino IDE that reads the data from the sensor, maps the humidity value, and displays the output on the LCD screen. The output was the current temperature in Celsius which was then converted to Fahrenheit and the current humidity in percentage. The readings were observed and compared to the current temperature that can be obtained from a thermostat. The readings were around the same value, therefore we were able to confirm that the sensor is functioning properly. Next, we tested the humidity and temperature sensor by placing it outdoors where it is hotter and more humid. The readings were observed and compared to the current temperature and humidity provided by a weather app. The readings were within boundaries, therefore the sensor is functioning outdoors as well.

7.1.2.3.2 Additional

To verify that the humidity and temperature sensor is functioning within our system, we placed Greenie indoor first and observed if the reading changes over time. We then transferred Greenie outside and observed the changes in temperature and humidity. The outputs shown on the LCD screen and the web app varied based on the current environment that the sensor is in. The observed values were around the values that a thermostat and weather app provided. If the temperature and humidity reading is within the ideal humidity and ideal temperature of a particular herb, then the system will turn

the water pump on.

7.1.2.4 pH Sensor Testing

7.1.2.4.1 Preliminary

To test that our pH sensor is working, we gathered several substances to see if data was correctly being obtained through the Arduino IDE. First, we started by getting the wiring setup for our circuit. We took our sensor, microcontroller, and breadboard and made sure the pins were properly connected. Once the wiring was complete, we set up the Arduino IDE and wrote some code that retrieves the information from the sensor. Then taking our pH sensor, we took it and placed it in a cup of water. If the pH value we see returned is between 6.5 and 8.5, then we confirmed that the sensor is partially working. To complete the preliminary test for the sensor, we took substances such as window cleaner, vinegar, and milk and verified that they were getting accurate results as well. In the case where all of our tests returned the same pH value, we would have confirmed that as a sign of potentially having faulty equipment.

7.1.2.4.2 Additional

For testing our pH sensor specifically with Greenie, we looked at the 10 herbs we've identified and observed how they changed over time. Take basil, for instance, we started with the soil when it was dry and see the value we get. After watering the herb, we did a check immediately to see how the values have changed. Then after waiting for an extended time, we checked the pH again to see how it was responding again. As for why we tested in this fashion, we have highlighted that for our 10 herbs that each one has an ideal pH range for the water to be in when dispensing. By checking how the pH changes over time after watering, we identified a good pattern of automatic watering for each type of herb.

7.1.2.5 Piezoelectric Sensor Testing

7.1.2.5.1 Preliminary

To test that our piezoelectric sensor is working as intended, we first made sure that it senses vibrations. To accomplish this, we connected the sensor to the MCU and connected the power to the sensor such that it receives the necessary voltage to turn on. This was set up on a breadboard where we can connect all these wires together to the Arduino IDE. A code that allows us to see if a reading can be gleaned from the sensor was utilized. After connecting everything together, this is where we can then move the sensor and see that a response emits from the sensor to the MCU. Once we make sure of this, we can see what sort of information is being passed so that it can be read by the user. We then observe if water is passed through the tube, the sensor accurately picks it up and lets the user know that this is going well. We have to place the sensor along various points on the tube to make sure that it can read water flowing in any place. If these things did not work, we would have to double check the

piezoelectric sensor to see if there is any glaring issue and if it needs to be replaced with another piece.

7.1.2.5.2 Additional

When we were testing the piezoelectric sensor with Greenie, we looked at all the sensors that are placed and made sure that only the correct sensors are turning on when we are watering that herb. Each herb has its own sensor on the tube that leads to the soil. The test made sure that we are noticing that the tube is actually being watered. There was also a tube where there is a hole that allows water to seep out of it, so that the piezoelectric sensor can detect that the flow is not normal. If the sensor can accurately tell that and let the user know, then we can confirm it is working as intended.

7.1.3 LCD Testing

7.1.3.1 Preliminary

For our project, we are going to utilize the LCD1602. This LCD was donated by one of our team members to cut the cost related to an LCD screen. The team member acquired this LCD screen as part of an electronics kit required for a project in a previous course, Junior Design. The project was created as an ultrasonic-based distance meter with an LCD screen. The same LCD1602 that was utilized for that project will be used in our product. In **Figure 31** below, we can see that the LCD1602 being utilized for this project works perfectly fine. As mentioned in the Project Specifications (refer to **Table 1**), the LCD1602 screen consists of two lines where sixteen characters fit in each. This can be observed in **Figure 31** as well, where the LCD screen is displaying the word “Distance”, which consists of eight characters, in the first line, and the message “2 cm”, which consists of four spaces (including the blank space), in the second line.

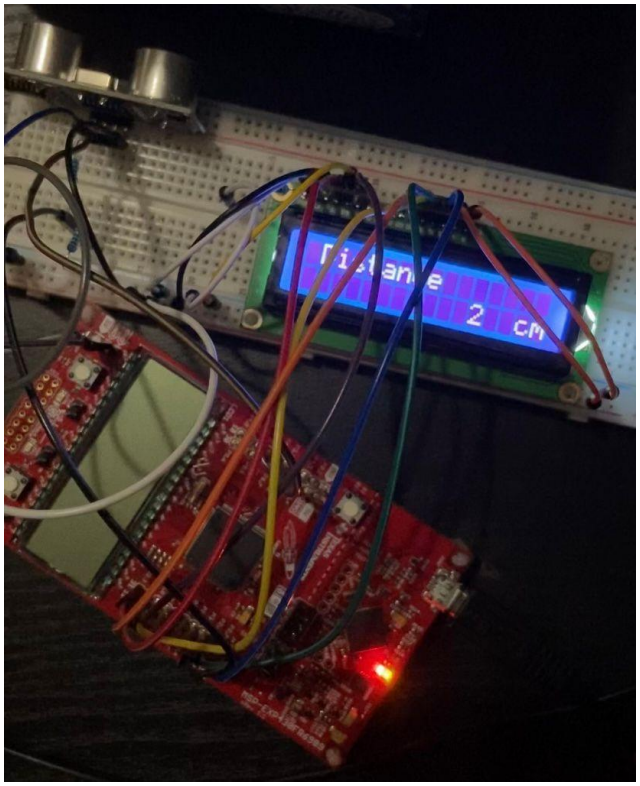


Figure 31: Preliminary Test of LCD1602 Screen

This figure is being presented as a preliminary test. As our project advances, we will be performing more tests on our LCD that are relevant to our project. We plan on utilizing our LCD screen to show data such as the soil moisture levels on the current plant, the pH levels in the soil, whether the plant has been watered that day by rain or not, current humidity and temperature in the room, and weather prediction data. When our software starts being developed, we are going to test our LCD to check if our data is being printed correctly.

7.1.3.2 Additional

Although a preliminary test was conducted on our LCD screen, more tests need to be planned to ensure that it works for the needs of our project. We are going to utilize our LCD screen to show data regarding the plant's health as well as weather prediction. Data such as the soil moisture levels on the current plant, the pH levels in the soil, and the current humidity and temperature in the room. Through the combined use of our weather prediction application, the LCD screen is also going to be able to show whether the plant has been watered that day by rain or not and if it will be on a day in the foreseeable future.

To make sure that the LCD screen is properly set up, we are going to appropriately plug its pin headers onto a solderless prototyping board by following the steps required. The LCD1602 is composed of 16 pins. The first pin, VSS, is where the ground (GND) is

going to be connected. The second pin, VDD, is connected to the 2.5 V to 5 V power supply. The third pin, VO, is in charge of controlling the contrast of the screen. The fourth pin, RS, is the register select pin. The fifth pin is used to select between reading or writing mode while the sixth pin is used as the enabling pin. The rest of the pins, pins 7 to 15 (D0 to D7), are used as the terminals to send and receive the data that will be displayed on the LCD. These pins can be configured in two ways, 8-bit or 4-bit. If the 8-bit configuration wants to be utilized, pins D0 to D7 must be used, otherwise, only pins D4 to D7 are used. This is due to the most significant bits being found in these last 4.

These pins are going to be appropriately connected with wires to the terminals in our microcontroller, the Miuzei Board with Atmel ATmega328P. Once the LCD screen and the microcontroller are properly wired, a trial code is going to be used to test that text and numbers are being printed on the screen by utilizing the Arduino IDE. Given that our screen can print 16 characters over 2 lines of text, all 26 letters of the alphabet are going to be printed on the screen over the two lines of text as the first test. For the second test, the numbers “0123456789” are going to be printed on both lines. Once we confirm that our LCD screen is correctly printing these characters, we are going to continue to test it with our sensors.

The LCD screen needs to read and print data from the sensors. Given that our soil moisture sensors are going to be tested by inserting their probes into soil samples with different moisture levels, the LCD has to print the numerical results. If our soil sensor is inserted into fully dry soil, the LCD screen needs to print “Soil moisture level of 1023” while if it is inserted into fully soaked soil, the LCD needs to print “Soil moisture level of 0.” Our pH sensor is going to be tested by measuring different liquids, so if the probe is inserted into a neutral liquid, the LCD should print “pH level of 7.” Lastly, as we are testing our humidity and temperature sensor, the LCD needs to print the resulting values for humidity and temperature that are recorded by the sensor’s probes, in the format “Humidity reading of XX” and “Temperature reading of XX.”

Our product is going to include an integrated weather application. The LCD screen is also going to be tested through the use of the weather application. Our screen needs to display information regarding the weather forecast as well as if the plant has been watered by rain or not through the use of our rain sensor. This is going to be tested by performing two test cases utilizing our rain sensor. In the first test case, the rain sensor is going to be sprayed with a few drops of water. If the sensor detects water, the LCD screen should print “Plant was watered by rain.” The second case is going to be when the sensor is completely dry. If the sensor detects no water, the LCD screen should print “Plant was not watered by rain.”

7.1.4 Microcontroller Testing

7.1.4.1 Preliminary

To test that our Atmega328P microcontroller was working, we performed several tasks. Given that the microcontroller was at the center of our project, ensuring that it worked

properly was crucial. First, connecting the microcontroller to a computer and the Arduino IDE we began with a basic blink test. We used the LEDs built into the board, loaded a sample program provided, and ran it as is. The LED returned with the result we were expecting and we noted it as a good sign. Then we modified the code so that the LEDs blinked for different scenarios. One example was the LED blinking faster. After seeing it worked according to how we programmed it, we confirmed that a portion of the microcontroller was working.

Next, we went through other examples that were provided by the Arduino IDE. Some of the built-in examples included a digital serial test, an analog serial test, and an LED fade test. By going through these types of examples, we confirmed that the pins on the board are working and that the LEDs are working beyond blinking. Lastly, we tested the microcontroller by adding libraries like the SPI library and ensured that those were working with the microcontroller as well. With libraries such as those working, then we anticipated that libraries for a separate hardware component like the wi-fi module would work as well.

After going through all the tests in the IDE, we can say that the microcontroller's base functionality is fully working. Our next steps from that point on were to create a more complex circuit.

7.1.4.2 Additional

To test our microcontroller for Greenie, we verified that a connection was established from the web application down to the hardware components. Similar to the software to hardware process mentioned in section **6.3.4**, we took buttons that we created for the web application and tested if the microcontroller is being properly integrated.

Looking at the water pump as an example, we started at the web level application level and pressed a button to turn the water pump on. On the button press, the code that we wrote reached the microcontroller and as a result, reached the relay module which has the water pump connected to it. The connection was properly made and we noticed the water pump turned on. Another example we looked at was the LCD screen. Going through the same process, at the press of a button, we reached the hardware and displayed any piece of information we wanted. What was important to note in this test was that for however many devices we needed to connect to the microcontroller we were able to control them on a web application level.

7.1.5 Power Supply Testing

7.1.5.1 Preliminary

The power supply will need to go through various tests so that we can make sure that the system works as intended. The first thing we need to do is make sure that the power supply holds the correct amount of voltage. This will be verified with a multimeter The

power supply unit (PSU) will be measured using a multimeter with no load connected to make sure that the full voltage of the power supply is being transmitted through the system. We measured to see if the correct amount of voltage was being drawn from the power supply while also making sure that minimal variation occurs. If there is too much variation in the system, this can cause issues when trying to draw power from the different pieces.

When the first test passes, the next test will be done with maximum load instead of minimum load. So we would be testing the ampere at the highest load it is supposed to work in, thus we will try to simulate the full load. We will also have to use the multimeter to find how the device performs, if it were to not perform correctly, we will need to find the cause and do our best to remedy the situation.

The last test is to look at the ripple voltage of the output. To check for this, we looked at the outputs using an oscilloscope, and we ensured that there was as little ripple voltage as possible.

7.1.5.2 Additional

To see if the system works as intended with our power supply, we need to connect everything to make sure that all of it can be powered. This will be done by connecting all of the pieces together, from the regulators and sensors to the MCU and pump, once all the pieces are in place, then we will hook in the power supply and check to make sure that every piece comes on and is powered appropriately.

7.1.6 Relay Module Testing

7.1.6.1 Preliminary

To test that our relay modules were working, we checked that the switching functionality works as expected. To do this we performed a blink test with the realy's built-in LED. We started by getting our Microcontroller, some wires, and a breadboard. Then we wired the circuit so the relay module was properly connected on one end and the other end was connected to the microcontroller. Once the circuit was assembled, we set up the Arduino IDE and wrote the code to get the switch to flip and the LED to glow. Starting with the switch closed and leaving it closed, we verified that the LED turned red to indicate it was on. Then with the switch open and leaving it open, we verified that the LED turned off. When both actions work as intended, we confirmed that our code was being recognized and being sent to the module. In addition, we recognized that our modules had a good chance of working in our manual watering process.

Next, we verified that the module can do the switching independently. To do this, we modified the code so that the LED turns on and off in a loop. This worked as intended and we confirmed that the automatic watering component has a partially good chance of succeeding. Lastly, we verified that the module can switch from open to closed over

different periods. To do this, we modified the code and put in different time periods for the LED to turn on. We then compared the results with a stopwatch for accuracy.

The results we obtained show a response time that was in line with our requirement specifications and confirmed that our modules are fully working. Also, we confirmed that the modules have a great chance of working in our automatic watering process.

7.1.6.2 Additional

For testing our relay modules with Greenie, we moved on from the built-in LED to our actual components. So, similarly wiring the circuit to the preliminary setup, we put in the water pump and see if it will turn off and on in response to the switches being opened and closed respectively. We went on to perform all the tests mentioned in the preliminary section and verified that the water pump works in the same manner as the LED. Doing so gave the most clarity in seeing if our manual and automatic watering processes work or not. During the final test, we checked once again with a stopwatch to see if our results were in line with our requirement specifications. In the case where our tests were not yielding the results we were looking for, we would have reevaluated the components we are using as well as the specifications we laid out. Something we recognized that was not ideal if it came to that, but also was something that we viewed as necessary to keep our project within the realistic bounds of completion.

7.1.7 Water Pump Testing

7.1.7.1 Preliminary

So that we can check that the water pump works well we will need to test how it operates. To do this, we must pass the correct operating voltage of about 12 V and the current of the pump with an appropriate power source to see and make sure that the pump works well. The pump must drive water through at the correct rate of 240L/H. To see if the rate is correct, we will fill a bucket with water using the pump for 10 minutes, and then make sure that the rate that the bucket filled is in line with the specifications. If it is not going through at the right rate we must find out what the problem is, if there is something wrong with the circuitry it could be opened and corrected. Otherwise, if it does not function at all, then we would have to order a new piece since ours would be considered defective.

7.1.7.2 Additional

The next part to check is to see if the water pump works with the code from the microcontroller. We will need to use a breadboard, wires, the pump, and the MCU for this. To do this, we must connect the various pieces together and make sure that the code can run the pump, turning it off and on so that it can provide water to the pumps when needed by the various sensors. Once we confirm that it works as intended, we

can see if the MCU can do things like pumping water at different rates. After looking through this, we then need to see if it can operate for long periods of time.

7.1.8 Voltage Regulator Testing

7.1.8.1 Preliminary

For the voltage regulator, we need to make sure that it works as intended. To do this, we must check that the proper voltage enters the device and that the proper voltage also exits the device. The regulators we will be looking at are the 12 V, 3 V, and 5 V, so we must make sure that the proper values are being recorded with our digital multimeter (DMM). Since many voltage regulators are being used, we need to go through them and make sure that each one of them is working as intended.

For our project, we plan on utilizing the LM2576 voltage regulator. This regulator has a five-pin design. The first pin is used for the input voltage, the second pin is used for the output, the third pin is used as the ground, the fourth pin is used as the feedback, and the fifth pin is used as the on/off. The pins can be checked using our DMM. To ensure that the input voltage pin is working, we are going to check the voltage from the input pin to the ground. We will be setting the DMM to the DC voltage setting and through the use of the DMM probes, we are going to place them on the input pin of the voltage regulator and the ground pin, pin three. The same process is going to be repeated for the output pin, pin two. However, for this test, the positive probe of the DMM is going to be placed on the output pin and the negative probe on the ground pin.

Another test that can be run is a digital test that Texas Instruments provides for people to use that is called the Webench Power Architect tool. This tool can allow us to try out samples of voltage regulators that perform well to see how this piece should be working as intended.

7.1.8.2 Additional

When testing it with Greenie, we need to make sure that the proper values are flowing through the system once everything is hooked up. The regulator on the pump should read 12 V, the regulator at the MCU should read about 5 V and the regulator around the various sensors should read about 3 V. A multimeter will be used to determine if the values are correct when all of the pieces are put together. We also need to make sure with an oscilloscope to check the ripple voltage of the various voltage regulators that are going to be hooked in the system.

7.1.9 Continuity Test

7.1.9.1 Preliminary

The first thing that we have to do in the greenhouse system for testing, is having to do what is called a continuity test. This test is done so that we can make sure that all the connections that are located on the PCB and the various sensors and microcontrollers were soldered correctly and that the paths that the current takes are moving correctly and match the schematics that are made. So that this test is done correctly, team members need to look at each piece individually using the schematics and we need to perform the continuity test with a multimeter using the continuity function on said device. Testing to make sure that there is a correct current path is important because if there is a break, this can result in there being an incorrect signal being sent out and the MCU collecting the wrong information making our device give incorrect data back to the user.

When we are checking the various ports in the system, we need to make note of any discontinuities that are found within the system. Once these are found, we need to take steps to ensure that we can fix that part of the system, either by soldering the connection that is not working or by affixing a wire so that the connection can be completed. In the case where there are many connections in the system that do not work, we would have to reorder the part from the vendor, so it is important to this procedure first.

7.1.9.2 Additional

Once the preliminary tests are done, additional testing will need to be considered to ensure that the PCB works correctly. To ensure that everything is right, the project needs to be put together and we must ensure that all the data is being transmitted successfully. If everything is working as expected, there will be no need to do anything else, however, if there is an error in some part of the project, we need to perform more tests on the PCB to make sure that no component becomes fried while powering everything together.

7.2 Software

7.2.1 Software Test Environment

Software testing is an essential part of project development to ensure that the overall system is working as intended. We ran several test cases to ensure that every component of our project is functioning as expected. The web application and all the software components of Greenie were tested on a computer using a web browser. The software testing for the microcontroller were conducted on a computer using the Arduino IDE. The testing process were documented to keep track of all the tests that have been done on the software. This allowed us to determine what changes need to be done to meet the specific requirements. Our software developers came up with test cases that we used to improve the quality of our system’s software. These test cases helped ensure that we meet all the requirements of our project. Using a test case was beneficial when deciding how to execute the different software tests. A test case includes the objective, steps, expected output, actual output, and pass/fail. The

objective is where we are going to describe our goal for that particular test. The steps list the instructions that need to be done to achieve our objective. For the expected output and actual output, we described the output that we want to achieve and the actual output that we got after completing all the steps. The last section of the test case was based on our actual output. Pass was used if we are able to achieve the expected output and fail if we are not able to achieve the expected output.

In addition to test cases, alpha testing and beta testing were considered when testing the software of our project. Alpha testing was performed by the developers to identify all possible issues and bugs before presenting our project. After the development of our project, our group members tested Greenie and executed several tasks that a user might do in our system. This allowed us to fix all the issues that arise to provide a better system before the beta testing. Beta testing is done by several users to obtain feedback on the quality and efficiency of the project. We let other people test our project and asked them how well our project functions and some suggestions on how we could improve our project. Alpha testing and beta testing were executed after performing all the other tests and before presenting our project.

Other software testing methods that we can use in testing the software components of our project include black-box testing and white-box testing. In black-box testing, we can test the functionalities of the software without prior knowledge of the internal structure of the system such as the code structure and the implementation process. It focuses on providing input and observing the generated output. Black-box testing can help us identify the response time, possible usability issues, and how our system will respond with user interaction. It can be applied during unit testing, integration testing, system testing, and acceptance testing. The first step in carrying out black-box testing in our system is to identify all the requirements and specifications of the system. Next, we can come up with valid inputs to check whether the system will respond as expected. We can also choose invalid inputs to check whether the system is able to detect them. The next step is to create test cases with the selected inputs and determine the expected output for the inputs. We will then execute the test case and compare the actual outputs with the expected outputs. For the last step of black-box testing, we can fix all the defects and test the system again.

In white-box testing, we can test the functionalities with the knowledge of the internal structure and design of the software. We are able to see the code as we conduct this test. White-box testing is mainly applied to unit testing, but it can also be applied to integration testing and system testing. It focuses on testing the code. Each statement and functions are tested including the functionality of the loops. When performing this kind of test, the tester must learn and understand the code. We can then create test cases with both valid and invalid inputs. After that, we can execute the test cases and compare the actual outputs with the expected outputs. The advantage of white-box testing over the black-box testing is that testing will be more precise and detailed as it aims to cover most parts of the code.

When it comes to testing the software components of our project, the code is tested frequently to ensure that every hardware and software component and the overall

system will run smoothly and without errors. The software components of Greenie were tested at different levels to ensure that we get the best results and reduce any potential bugs or errors. The first level of testing is called unit testing wherein small parts of code are tested independently. This was done throughout the development of our project to ensure that every part of the code is functioning as expected. Conducting this type of testing allowed our software developers to determine the errors before they proceeded in testing the other parts of the code and the entire code. The next level of testing is integration testing where individual components were integrated and tested as a group. This testing ensures that all components are compatible with each other. It also detects errors in the interaction between the combined components. The third level of testing is system testing where the overall, integrated system were tested as a whole. This testing was very helpful in evaluating and verifying if the system has met all the requirements and specifications. The final level of software testing is called acceptance testing where the system was tested for acceptability. Acceptance testing was performed by actual users who determine whether the system meets all the requirements and specifications.

These levels of software testing were applied in our project throughout the development process. For instance, the code for each sensor was tested individually during the unit testing. When an error was detected, we were able to easily identify the part where there is a fault and fixed it right away. As for the integration testing, we tested the sensors together. We then observed how the sensors work together and how one sensor impacts another sensor. The system testing involves testing the system and the web as a whole. We observed whether Greenie is functioning properly considering all the requirements and the specifications that we have identified at the beginning. We made sure that the web app is responding to the system and displaying the correct output. For the acceptance testing, we asked other people to test out our project and gathered some feedback from them.

7.2.2 Wi-Fi Module Testing

7.2.2.1 Preliminary

To test that our ESP8266 ws working properly, we performed two tasks. First, we used the module’s micro-USB port to connect to a computer and the Arduino IDE and bring up some sample code. After establishing the connection, we initiated a blink test on the module’s built-in LED. From there, similar to the Atmega328p testing, we ran it as is and then verified that it was blinking correctly. If it ran as expected then we noted that as a good sign. Then, once again we modified the code to work for different scenarios. After making these modifications, we verified that it worked for how we programmed it. If the LED responded in the way we expected, then we could say that the module was partially working. For the second task, we tested to see if the module can do any kind of network connection. To do this task, we used the module as an access point for other devices to look for. So, going back to the Arduino IDE, we wrote some code that utilized the ESP8266 Wi-Fi library and set up the required components for the access point. Once the access point was set up, we searched for the module on a computer and then connected to it. Finally, we opened a terminal on our computer and pinged to

192.168.4.1. The module responded to the ping, and we confirmed that the module was fully working.

7.2.2.2 Additional

For testing the ESP8266 with Grennie, we took our devices and verified that they were connected over the internet. Take our smart speaker, for example, to ensure that it worked with something like our water pump, we needed to verify that the speaker and the Wi-Fi module were connected to the same network. To accomplish this, we set up a username and password for the Wi-Fi module while adding the necessary libraries to the IDE. Once we've verified that both devices were on the same network, we verified that the Wi-Fi module was properly connected to the water pump. To do this we programmed the module like our Atmega328p microcontroller and caused the water pump to trigger as needed. All the connections were correctly made, we confirmed by performing more tests like in our next section "**7.2.3 Alexa Integration**."

7.2.3 Alexa Integration Testing

7.2.3.1 Preliminary

For performing a preliminary test of Alexa, we verified that our selected smart speaker is working properly. For this, we asked simple questions such as: "What's today's date", "What's the weather like", or "What's 1 + 1". The responses given are what we expected and we confirmed that our smart speaker was working and so was Alexa.

7.2.3.2 Additional

For testing Alexa specifically with Greenie, we started by going to the Arduino IDE, setting up the connection with the Wi-Fi module, and adding in our desired libraries. We wired the circuit similarly to **Figure 9** in section **5.3.5** with the Wi-Fi module acting in place of the microcontroller. This setup was important for Alexa testing because the relay module was still acting as the controlling force for deciding whether to activate a device or not. We then programmed the required actions needed for turning our water pump on and off. Once that was complete and the circuit was constructed, we went to the Amazon Alexa app and registered our water pump as a new device. With the water pump added, we issued commands for Alexa to perform. The following commands were executed:

- "Alexa. Turn on the water pump"
- "Alexa. Turn off the water pump"
- "Alexa. Turn on temperature display
- "Alexa. Turn on soil display"

7.2.4 Web Application Testing

For the web application testing, we followed the standard web testing requirements including functionality, usability, security, compatibility, performance, interface, and crowd testing. This testing was done during and after the development of our project. As part of the web application testing, we made sure to check the following requirements that are outlined below.

Functionality Testing checks if each feature on our web app is functioning as expected.

- All links must be working properly using a web browser.
- Cookies should be deleted when the cache is cleared.
- Possible syntax errors will be checked.
- The ‘start’ button on the ‘Herbs’ page should add the selected herb to the system and start the automated watering.
- The ‘stop’ button on the ‘Herbs’ page should stop the automated watering.
- The ‘start/stop’ button on the main page should turn on or off the water pump and the nutrient pump.
- The dashboard should display updated readings from the sensors and updated weather forecast.
- The Metrics page should include a graph of the amount of water and nutrient solution that has been dispersed and how much was saved.

Usability Testing checks if our web app is responsive and accessible.

- Menus, buttons, and the herb’s daily statistics should be visible and accessible on the web page.
- The format of the web app should be consistent and easy to navigate.
- Texts should be clear and understandable.
- Texts and images must have the right proportion.
- The theme and colors should not be too bright.
- The tabs on the left side of the web app should redirect users to the correct page.

Security Testing is important since our web app asks for the user’s personal information such as name and address.

- When users enter an invalid address, the weather information will not be displayed on the web app and integrated into the system.
- The session should expire after 30 minutes of inactivity.
- The herb’s daily statistics and watering history should not be accessible to other users.

Compatibility Testing checks the compatibility of our web app on various operating systems, web browsers, and devices.

- The web app should work on Windows, MAC, and Linux.

- The web app should work on Google Chrome, Microsoft Edge, Mozilla Firefox, and Safari.
- The web app should work on desktops, laptops, tablets, and mobile phones.

Performance Testing checks how well our web app performs under different circumstances.

- The response time should be less than 5 seconds.
- The response time will be checked under different internet conditions.
- The website must handle multiple users at the same time.

Interface Testing checks if communication between the web server, application server, and the database server is working properly. Crowd Testing allows different users to test our web application and provide feedback.

7.2.5 Weather Integration Testing

To test the weather integration for Greenie, we used the API keys provided to us to verify that they work for our web application. Starting with the web application, we started by using our current location and getting it to appear. Once that was confirmed we moved to the forecast information. We checked to see if the temperature is accurately displayed as well as the forecast status (i.e rain, sunny, cloudy, etc.). To do the accuracy check, we looked at weather stations on a separate device and noted the similarities. With all the checks going well, we verified that the weather integration was working on our web application.

As for the notification for alerting the user of rain, we decided to use the forecast status as a key point. So in the case of rain, we went back to the web application and configured it so that it tells the user that it would be best to use the water pump at another time given its status. To ensure that this feature was tested on time, rather than waiting for it to rain where we are currently located, we searched for a location where the forecast was already set to rain and sent the notification based on that. After performing the rain forecast test, we found a location where it is sunny to verify that the notification no longer appeared.

7.2.6 Database Testing

7.2.6.1 Preliminary

Given that we have decided to implement Firebase for our project and we are looking to create an application that involves Arduino hardware materials, we have decided to perform tests on our database to ensure that it works well in our project. In the event where we assumed the database would work well with our hardware components and they did not, we thought it would be best to catch these mistakes early. So, to perform a preliminary test on our database we started by making sure that when working with the web application, that the data retrieved was stored properly. We began by creating a

table and filling it with random pieces of information. We then took the information and displayed it on our web application. If the data seen on our web application was accurate, then we confirmed that our database is storing information correctly and working with our web application. Next, we took that random information stored in our table and then got it to display on our LCD. By doing so, we were not only able to see that our microcontroller was set up properly with our LCD, but also that our stored information was capable of reaching our hardware components. We recognized this as an important check because it also played a big part in making sure our sensors were reachable. With our sensors, because we were looking to do several checks based off of the 10 types of herbs in our database, we wanted to ensure that our sensors were capable of utilizing that information.

7.2.6.2 Additional

For testing our database with Greenie, we expanded on our preliminary testing by using our 10 herbs in place of the random information. Taking basil as an example, we took the status we've stored about basil in the database and then got that data to display on the LCD. If the display showed the expected status, then we could say that our Database is capable of reaching the hardware. To verify that the status showed up correctly for basil, we also decided to do checks for the other 10 herbs. Given that the status should return a certain value according to a certain herb, we confirmed that hardware was working very well with our database when we saw that results were displaying correctly for each herb on the LCD.

Then moving onto the sensors, for testing the database with them we took the data normally used for sensor checks and verified that they could be altered. So, using basil again but this time with our various sensors (soil moisture, temperature/humidity, pH) we started by taking the values assigned in the database, sending them over to the Arduino IDE and verifying that it has been captured in the output screen. The values in the output screen matched the values in the database tables and we confirmed the data transfer was a success. Then proceeding on with the alter check, we verified that the values we've obtained were usable and that they did not yield any errors. So, we took a value like temperature and added/subtract from it. After making the change and seeing that the values updated accordingly on the hardware, we confirmed that the transfer was a success and that there were no errors.

7.2.7 Web Service Testing

7.2.7.1 Preliminary

Given that we have several tests and components in our project that revolve around the web application working properly, we recognize that testing the web service itself is equally as important. So, for performing a preliminary test with Drive to Web decided to test the responsiveness of their service as well as the it's accuracy. In terms of responsiveness, we started by creating a gmail account for the group and creating some simple code to upload. After creating a simple setup with HTML, CSS, and

Javascript, we created a folder on Google Drive and uploaded the files to it. We then deployed the code using their service and waited for the response. After seeing the code update instantly, we confirmed that the service was responsive and that the Drive to Web servers are running fine. Then in terms of accuracy, we tested this by uploading code to it that involved external sources from the internet. We determined this was crucial to test so we could verify that our weather integration could be properly deployed online. After confirming that using external internet works in the code deployment, we confirmed that the Drive to Web does work for our web application.

7.2.7.2 Additional

For testing our web service with Greenie, we performed a button interface test. With the water pump for example, we created our button for manually turning the water pump on and off and uploaded the code for it to the Google Drive folder. After making all of the necessary connections, we attempted to turn on the water pump on and off and see if we got our expected outcome. If neither button worked, then we concluded that we have an issue with establishing a connection between the software and hardware. And to remedy problems such as these, we would go back to our code and continually modify it until the connection can be made. Ultimately, after making a successful connection between the hardware and the software, we were able to confirm that Drive to Web does work well specifically for our project.

8. Administrative Content

In this section, we will be discussing the administrative content behind our project. We will discuss relevant information such as our budgeting and financing, our Bill of Materials (BOM) as of now, our milestones for Senior Design 1 and 2, and how our team members have divided the content of our project.

8.1 Budget and Financing

Greenie is financed by the members of Group 10. Throughout our research, we looked around for the most inexpensive items that would ultimately lead to the completion of our objectives. We have also decided to place a cap in our budget of up to $500.00. The table shown below (refer to **Table 18**) gives a rough estimate of how much money we intend to spend on the components we believe we need so far. The components and their cost have changed throughout the development of our project as they are acquired and tested. Given that the project is being self-funded by us, we have decided to utilize parts that we have already purchased for other projects, whether they were personal or for other classes.

Item	Price	Quantity	Total
LCD1602 Module (already owned)	$0.00	1	$0.00
Microcontroller - Atmega328P (already owned)	$0.00	1	$0.00
Power Supply - Portable Battery	$25.00	1	$25.00
Soil Sensor	$10.00	3	$10.00
Rain Sensor	$10.00	1	$10.00
Temperature and Humidity Sensor - DHT-11 (already owned)	$0.00	1	$0.00
pH Sensor	$40.00	1	$40.00
Piezoelectric Sensor	$10.00	1	$10.00
Relay Module	$10.00	1	$10.00
Water Pump	$15.00	1	$15.00
Wi-Fi Module	$15.00	1	$15.00
Tester Herbs	$5.00	3	$15.00
Wires (already owned)	$0.00	20	$0.00
Breadboard (already owned)	$0.00	1	$0.00
Voltage Regulator 12V	$1.50	1	$1.50
Voltage Regulator 5V	$0.95	1	$0.95
Final Total = $152.45			

Table 18: Estimated Budget

8.2 Bill of Materials (BOM)

Our Bill of Materials (BOM) can be observed below in **Table 19**. In it, all of the components that we have purchased for our project can be observed. The components have the same numerical label and name as in **Figure 11** and **Table 18**.

Number	Part Name	Qty	Unit Cost	Cost
1	HiLetgo ESP8266 Development Board	2	$6.49	$12.98
2	Miuzei Board with Atmel ATmega328P	1	$0.00	$0.00
3	LCD1602 Module	1	$0.00	$0.00
4	KeeYees LM393 Soil Moisture Sensor	1	$7.99 for 3	$7.99
5	Teyleten Robot LM393 Rain Sensor	1	$5.88 for 2	$5.88
6	DHT-11 Temperature & Humidity Sensor	1	$0.00	$0.00
7	Solar Power Bank 10,000mAh	1	$28.99	$28.99
8	Bojack Voltage Regulator Assortment Kit	1	$13.99	$13.99
9	Quickun Pure Silicone Tubing	1	$12.99	$12.99
10	LEDGLE Mini Submersible Water Pump	1	$8.99	$8.99
11	GAOHOU PH0-14 Sensor Module	1	$35.59	$35.59
12	HiLetgo 5V Relay Module	2	$6.19 for 2	$12.38
13	Gikfun 3m Silicone Tube 2mm	1	$7.88	$7.88
14	Gikfun 12V DC Dosing Pump	1	$10.98	$10.98
15	PCB	1	$35.51	$35.51
16	PCB Components	1	$39.18	$39.18
17	Enclosure	1	$31.99	$31.99
18	Nutrient Solution	1	$6.89	$6.89
19	HFS Solenoid Valve	1	$13.99	$13.99
	Final Total:			**$286.20**

Table 19: Bill of Materials (BOM)

8.3 Senior Design 1 Milestones

Given that this Senior Design course is split into two semesters, in **Table 20** and **Table 21** seen below, the milestones for each course and its corresponding dates can be observed.

Milestone	Task	Completion Date	Completion Status
1	Submit initial project document (D&C 1.0)	June 11, 2021	Complete
2	Have the project approved by Dr. Riche	June 16, 2021	Complete
3	Submit initial project document (D&C 2.0)	June 25, 2021	Complete
4	Order and test parts	July 2, 2021	Complete
5	Design and create PCB	July 7, 2021	Complete
6	Submit 60-page draft	July 9, 2021	Complete
7	Have a 60-page meeting with Dr. Richie	July 14, 2021	Complete
8	Perform more in-depth research for 100-page submission	July 12 - 22, 2021	Complete
9	Submit 100-page report	July 23, 2021	Complete
10	Submit SD1 final document	August 3, 2021	Complete

Table 20: Senior Design 1 Milestones

8.4 Senior Design 2 Milestones

Milestone	Task	Completion Date	Completion Status
1	Finalize ordering correct parts for the project	August	Complete
2	Finish testing project	August	Complete
3	Get web app working	August	Complete
4	Connect web app to hardware	September	Complete
5	Purchase materials necessary to build the project design	September	Complete
6	Build product design	September / October	Complete
7	Verify that the build works properly with the hardware and software built	October	Complete
8	Refine project	October	Complete
9	Edit and submit SD2 Final Document	End of SD2	Complete
10	Prepare final presentation	End of SD2	Complete
11	Build and prepare a website to store all of our project's information	End of SD2	Complete
12	Final presentation	End of SD2	Complete

Table 21: Senior Design 2 Milestones

8.5 Content Distribution

Senior Design courses are team-based so the work behind our project has been split between our team members. Given that there are two Computer Engineering (CPE) students and two Electrical Engineering (EE) students, the areas of focus were assigned accordingly. Our CPE members were assigned the software part of our project and the EE members were assigned the hardware parts. This was done to ensure that all members were equally contributing to our joint document while working on the areas that they enjoy the most. Below, the content distribution between our team members can be observed.

1. **Angelica**:
    a. Microcontroller
    b. Sensors (soil moisture, rain, pH, humidity and temperature)
    c. AC/DC Converter
    d. DC/DC Converter
    e. Water pump
    f. Relay Module

2. **Elliott**:
    a. Database
    b. Wi-Fi module
    c. Alexa integration
    d. Web application

3. **Kevin**:
    a. LCD
    b. Power supply
    c. Voltage Regulator
    d. PCB
    e. Relay Module
    f. Solenoid Valve

4. **Patricia**:
    a. Communication
    b. Interface
    c. Web application

5. **Everyone**:
    a. Documentation
    b. Building
    c. Testing
    d. Refining

9. Appendices

9.1 Appendix A: References

[1] https://www.ers.usda.gov/topics/farm-practices-management/irrigation-water-use/#:~:text=Agriculture%20is%20a%20major%20user,percent%20in%20many%20Western%20States.

[2] https://www.intellias.com/smart-irrigation-in-agriculture/

[3] https://www.ourpcb.com/pcb-design-softwares.html

[4] https://www.oaktreeproducts.com/img/product/description/List%20of%20Worldwide%20AC%20Voltages.pdf

[5] https://www.digikey.com/en/maker/blogs/introduction-to-dc-dc-converters

[6] https://hetpro-store.com/PDFs/lm317.pdf

[7] https://www.4pcb.com/about_us/

[8] https://www.ti.com/lit/ds/symlink/lm2576.pdf?ts=1626712929011

[9] https://www.onsemi.com/pdf/datasheet/lm2596-d.pdf

[10] https://jlcpcb.com/

[11] https://www.pcbway.com/

[12] https://www.npr.org/templates/story/story.php?storyId=91363837

9.2 Appendix B: Purchase Links

Solar Power Bank
https://www.amazon.com/gp/product/B07FDXDB3W/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

LEDGLE Mini Submersible Water Pump
https://www.amazon.com/gp/product/B085NQ5VVJ/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

Teyleten Robot 3pcs LM393 Rain Drops Sensor

https://www.amazon.com/gp/product/B088FXM2JG/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

KeeYees 5 Pcs High Sensitivity Soil Moisture Sensor Module
https://www.amazon.com/gp/product/B07QXZC8TQ/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

HiLetgo 1PC ESP8266 Development Board
https://www.amazon.com/gp/product/B010O1G1ES/ref=ppx_yo_dt_b_asin_title_o00_s01?ie=UTF8&psc=1

BOJACK Package High Current Positive Voltage Regulator Assortment Kit
https://www.amazon.com/gp/product/B07T5ZHY63/ref=ppx_yo_dt_b_asin_title_o00_s01?ie=UTF8&psc=1

Quickun Pure Silicone Tubing, 8mm ID x 12mm
https://www.amazon.com/gp/product/B08BR8BB5D/ref=ppx_yo_dt_b_asin_title_o00_s02?ie=UTF8&psc=1

HiLetgo 5V Relay Module
https://www.amazon.com/dp/B00LW15A4W/?coliid=I1HRPDCZ693E0I&colid=3I8HKT9170ODE&psc=1&ref_=lv_ov_lig_dp_it

GAOHOU PH0-14 Sensor Module
https://www.amazon.com/GAOHOU-PH0-14-Detect-Electrode-Arduino/dp/B0799BXMVJ/ref=sr_1_1_sspa?dchild=1&keywords=ph+sensor&qid=1625522607&sr=8-1-spons&psc=1&smid=A2NM95757809ZA&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzNFBNMVRSSVIwVEw0JmVuY3J5cHRlZElkPUExMDQ0ODg5M0pDNklJSkc3Tlk0MyZlbmNyeXB0ZWRBZElkPUEwNTU2NDUxMlJCVUdVWEdES0lWUyZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=

Gikfun 3m Silicone Tube 2mm
https://www.amazon.com/gp/product/B08H1ZD5VZ/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

Gikfun 12V DC Dosing Pump Peristaltic Dosing Head
https://www.amazon.com/gp/product/B01IUVHB8E/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1