Autonomous Vertical Hydroponic Drip System

Department of Electrical Engineering and Computer Science
University of Central Florida
Dr. Lei Wei, Dr. Samuel Richie

Group 13

David Babcock	Computer Engineering
Jacob Reichle	Computer Engineering
Max Gomer	Computer Engineering
Marco Bogani	Computer Engineering

Table of Contents

List of Figures

List of Tables

1.0 Executive Summary

Over the last 20 years hydroponic systems have grown increasingly popular. As climate change worsens, chaotic climate conditions such as extreme temperatures, drought, and flooding are a grave threat to the food supply chain. It is likely that people will want to grow some food in their own homes. Plants grown through hydroponics require minimal water expenditures at the cost of human attention and soil/nutrient replacement mediums for higher yields. This project attempts to free up the necessary human attention to just replacing water and nutrients to their designated receptacles. In addition, we designed this project as a household appliance with the appearance of everyday furniture to make it an attractive option for people to grow food in their own homes.

To accomplish this, we pump water vertically in an enclosed system and let gravity water the plants. We use a small computer to handle the desired behavior and display necessary information to the user. In addition, we log the state of the hydroponic system and store it in a database over the internet. The internet functionality also lets users interact with the system via a phone application.

After much research and determination, we were able to build all the components of this project to completion.

2.0 Project Description

An automated hydroponic system that functions as both a stylish in-home decoration and a space efficient way to grow leafy greens. The system is able to be controlled wirelessly, allowing for nutrient dosing amounts and schedules, light cycles, and water flow to be customized.

2.1 About Hydroponic Systems

- What is a hydroponic system?
    - Hydroponics is the art of growing plants without soil. The main advantages of hydroponics are its very low water and space consumption, higher plant yields, and its ability to grow plants in harsh environments at the cost of power and nutrient management.
    - The soil is often replaced with another physical medium to allow the roots to grow and expand as if it were growing in natural soil. Without soil, certain natural ion exchanges aren't possible in replacement mediums. More complex nutrient mixtures are necessary for the plants' health.
    - The climate is getting warmer. If sea levels rise, the availability of arable land and fresh water will decrease. In a warmer climate there will be less options for plants that need a cooler climate to grow. A small farm people can use in their own home means families will be less dependent on traditional farming techniques while conserving a diminished fresh water supply.
- Why automate it?
    - Hydroponic plants need maintenance and monitoring to ensure the plants grow properly. The roots of each plant need constant water and sufficient light to survive and thrive.
    - This means a water-nutrient solution delivery system to the plants is necessary. Doing this work manually is cumbersome and using machinery will help get the most plant growth we can out of our hydroponic system.
    - Constant lighting can be detrimental to the growth of some plants. By providing a timer, we can be sure that the plants are getting the correct amount of light each day.
    - Nutrients also have to be mixed in water before being delivered to the plants. An uneven mixture can be toxic to hydroponic plants. Using a mechanical solution, we can ensure the nutrients being delivered to plants is what is desired.
    - Lastly, we monitor the current state of the hydroponic system with a variety of sensors and keep a record of it. By having a record of pH balance in the nutrient water, amount of light the plants received, and the amount of water the plants received, we give our users the data to make informed decisions on how to take care of their plants.

- Why a wall decoration?
    - Many hydroponics units currently on the market are limited in the styles available, such as tower and horizontal rack formats. Often, these units have exposed pipes or have industrial appearances, both of which may not be attractive qualities to a customer looking for an in-home hydroponics solution. By designing a system with an enclosed vertical wall style, customers now have a more aesthetically pleasing and unique product to choose.
    - By using an upright, against-the-wall format, the customer is able to save room in their home by avoiding the large space requirement of tower and rack style hydroponics units.
- Our technical approach:
    - To achieve these goals, our system is centered around a main controller unit (MCU) fitted with appropriate sensor modules to make accurate readings of the pH and water level in the recirculation reservoir. The controller unit is able to transmit data across a WiFi connection to a database that is hooked into a user application, allowing for remote monitoring of the system. Recirculation pumps, nutrient dosers, lights, and a mixing fan in the reservoir are actuated by relays controlled by the MCU when needed. The user must top off the recirculation reservoir when it gets below a minimum fill line and completely replace the water after a couple weeks' time. The MCU makes water level readings to send alerts for replacement, and a start/stop button in the mobile application allows the user to cut power to the pump for water replacement, while the nutrient dosers and mixing fan take care of pH adjustment and nutrient dispensing when the water is completely replaced.

2.2 Project Motivations

- Goals / Objectives
    - Build a system that delivers water to plants.
    - Build a system that mixes nutrients in water.
    - Build a system that logs data collected from sensors.
    - Build an aesthetically pleasing structure for walls that plants can grow in.
    - Build an application that interfaces with the operations of the automated hydroponic system.
    - The hydroponic system should sustain the life of the plants with the correct settings.

- Personal Motivations
    - David - To start, I want to demonstrate competence that I can program a microcontroller for a system of this complexity. Secondly, I want more experience building interfaces for internet and phone

applications. Lastly, I want more experience working in a successful team setting.

- Marco - Be able to apply my knowledge I have learned over the past 4 years in a practical manner within a group. This will push me to maintain engagement and push my creativity in order to finish successfully. This will teach me the necessary skills that I can use as a supplement to knowledge in the workforce.
- Max - Acquire more hands-on experience with the software and hardware development process, further educating myself in the computer engineering field and adding to my professional marketability. Learn how to better contribute to a group engineering effort and maintain a healthy team-based workflow. Further develop interpersonal communication skills and organizational habits.
- Jacob - Equip myself with new skills working with microcontrollers and full-stack software development. Learning how to bridge the gap between connecting a complex device to the internet for management is a great challenge that will give me some solid experience as a computer engineer. I also have a vested interest in the evolution of the agricultural landscape as technology becomes more powerful.

2.3 Requirements / Specifications

The following tables include requirement specifications for the hydroponic drip tower system. Tables 1 and 2 give a broad summary of the necessary requirements for the hardware and software aspects of the system, respectively. Table 3 shows specific demonstrable requirements that will be used as guidelines to ensure that the system is working to standard.

2.3.1 Requirements Summary

Shown below is a table that summarizes the hardware requirements of the project. Following this table is Table 2, which summarizes the software requirements.

Table 1: Hardware Requirements

Product Dimensions and Structure	● The unit shall have the target dimensions of 3ft wide by 5ft tall, with a minimized thickness that will not exceed 1.5ft. Being that the thickness will have the greatest impact on the space efficiency of the unit, it is highly beneficial to design the unit to be as thin as possible. ● PVC material for framing the

	hydroponic towers ● Pipe connection and joints will have watertight grommets to hold root balls of plants ● Waterproof enclosures will hold electronic connections to prevent moisture intrusion and system failure
The power supply for the system...	● The unit shall use one primary power supply that will meet the total power requirements of the system, with several step-down voltage circuits being used to provide the individual voltage rails.
pH, and water level sensors should...	● Take accurate measurements within a 10% margin of error ● Read respective input data every hour ● Be able to withstand long amounts of time submerged in a water-based nutrient solution
When handling water flowing throughout the system...	● No water should leak anywhere in the system ● A pump will send water through each hydroponic tower evenly ● Water flow will be great enough to keep plant roots wet whenever the system is on ● Water will be dispensed to each tower from a small reservoir pipe at the top of the frame with flow restrictors directed at each tower ● The water tank will have an easy drainage valve for convenient water replacement ● Relay controlled pump actuation through the system ● Pump water 5 feet from the bottom to the top tank (~40-100 GPH) ● Water falling down the towers will drain into the main recirculation tank ● Water will be mixed every hour to ensure nutrients aren’t stagnant at the bottom of the recirculation tank

To maintain proper nutrient and pH levels in the system...	● Dosers will provide small amounts of pH adjustment fluid to the nutrient solution daily at a rate of 1 mL of balancing fluid per liter of water if it is out of proper pH range [5.5, 6.0] ● Upon replacement of water in the main tank every 2 weeks, the dosers will dispense ½ tbsp of nutrient mixture per gallon to the tank and mix the fresh water together with it ● Plants should receive at least 32 Watts of light throughout a natural day-cycle via LED strips ● Constant flow of water and nutrient mix should be maintained at all times when water tank isn't being replaced
To monitor the system locally...	● A liquid crystal display attached to the frame will display current sensor readings ● A button on the frame of the system will be able to pause / restart the pump
The microcontroller unit program should...	● Turn water pumps, lights, nutrient dosers, and mixing fan on and off. ● Balance pH of nutrient water with an accuracy of 90%. ● Collect sensor data. (Water levels, water flow, and pH balance of the water) ● Send and receive sensor data to the internet application via Wi-Fi. ● Displays water levels, water flow, and pH balance of nutrient water on an LCD screen.

Table 2: Software Requirements

There will be a mobile application to interact with the hydroponic system, which includes...	● Displays water levels, water flow, and pH balance of nutrient water. ● Controlling light schedule, pH, and water pumps. ● External database of collected

	sensor data from the hydroponic system. ● Showing graphs of collected time-series data. ● Log In / Sign up ● Should pair the user with their login credentials with their specific hydroponic system.

2.3.2 Demonstrable Requirements

Table 3 shows eight of the system's requirements quantified and made demonstrable. Any requirement highlighted it is meant for demonstration specifically to the senior design review panel.

Figure 1: System Requirements

Pump Height	5.5 feet
pH Accuracy	Within 5% error margin
Sensor reading intervals	1 hour
Flow Rate	~80-100 Gallons per Hour
Post sensor readings to database within	1 seconds of reading
Send hardware commands within	1 seconds of sending from mobile app
Mixing fan intervals and frequency	Mix for 1 minute every 45 minutes
Frame dimensions	5' x 3' x 1.5'

2.3.3 House of Quality

The following House of Quality figure is used to efficiently display the relationships between both the user's requirements and the engineering requirements for the project. Detailed goals for the engineering requirements are also listed to better define target values or tolerable performance metrics for the implementation of each engineering requirement.

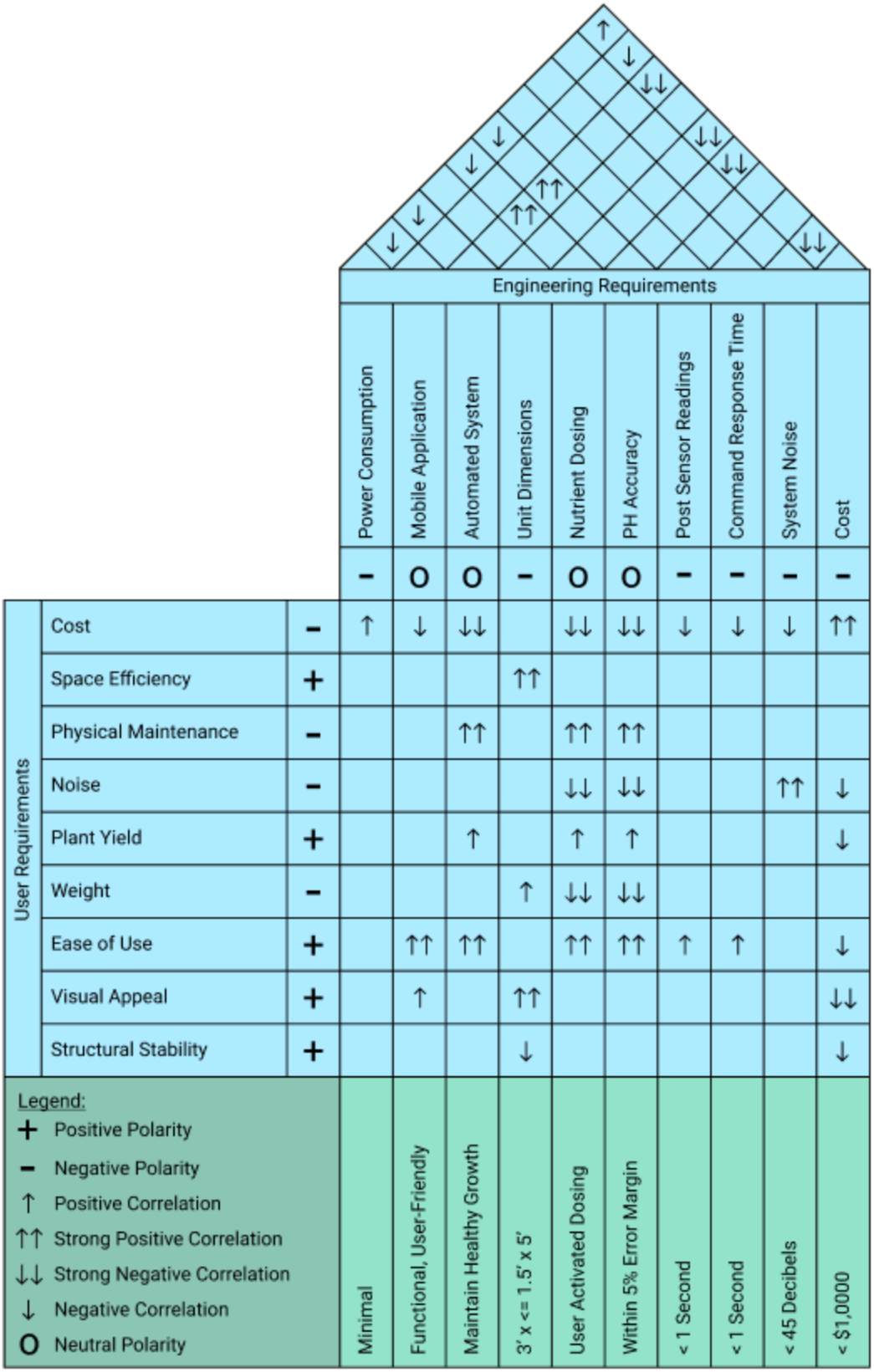


Figure 2: House of Quality

2.3.4 Block Diagrams / Illustrations

The block diagrams in the following sections show the hardware and software processes carried out by each team member. These processes also indicate what stages of development they are in (Acquired, Research, Development, Completed). All processes are currently complete.

2.3.4a Hardware Block Diagram

The diagram below depicts the main hardware components colors coded with the designated team members. In general, two team members focused on hardware and materials while the other two members focused on software development and some circuitry. As depicted by the diagram, the two main driving components are the Micro Controller Unit (MCU) and the Power Supply Unit (PSU). While the PSU supplies power to all the components, the MCU controls the functions of each component.

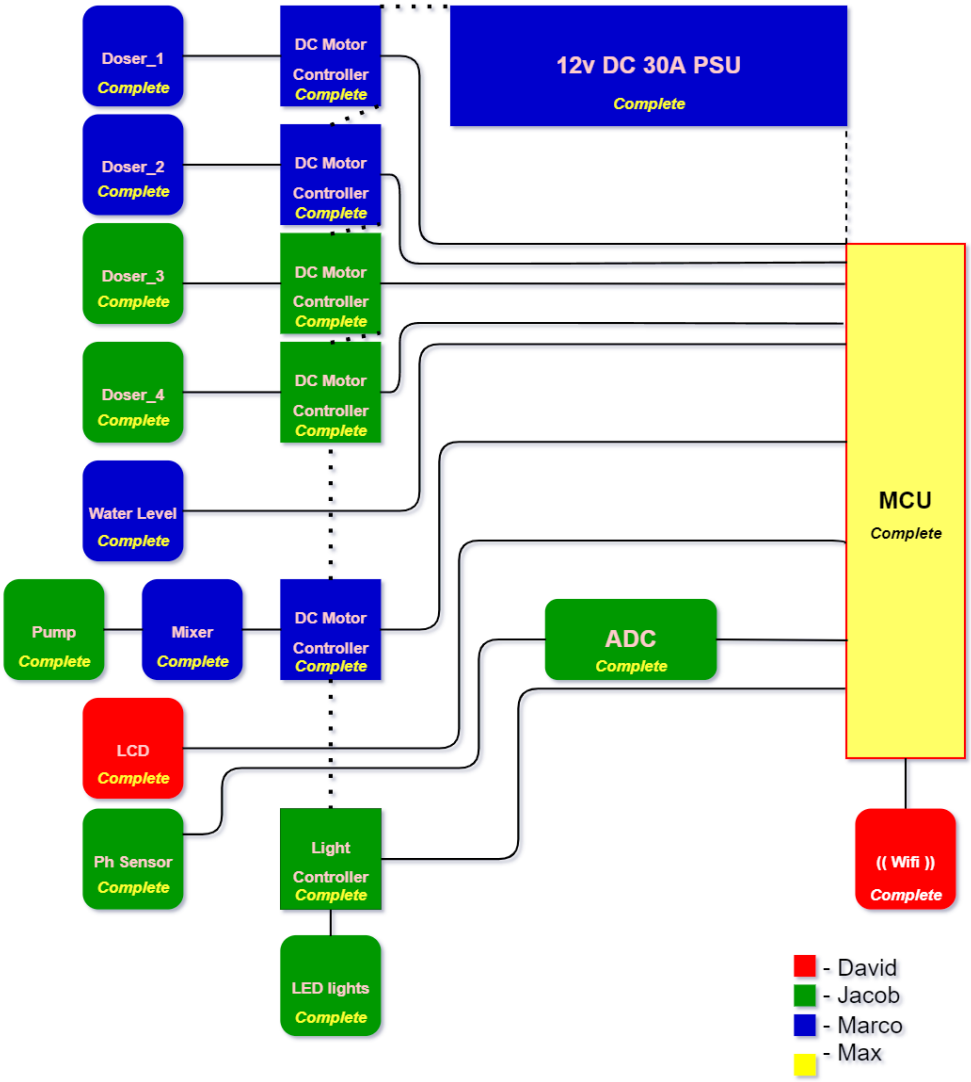
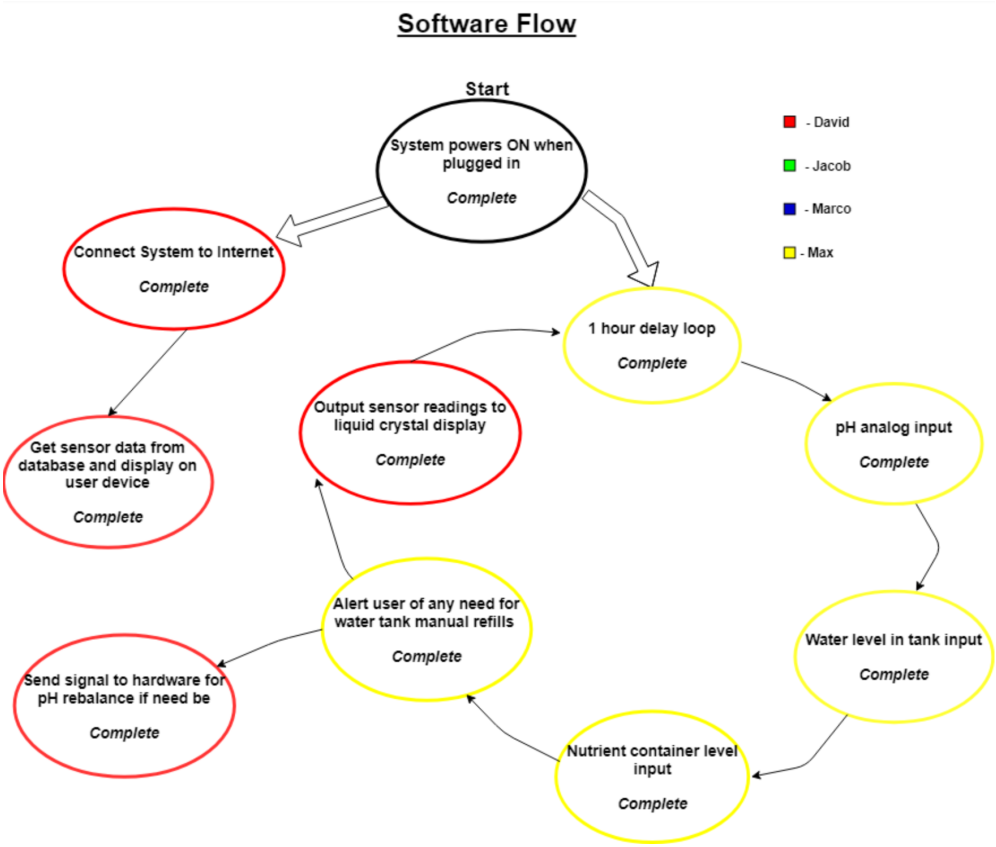


Figure 3: Hardware Block Diagram

2.3.4b Software Block Diagram

To encapsulate all system functions in the software, the loop shown in Figure 3 was followed. Once the system is powered on, it has two main branches of functionality: the first is to press a button to be able to pair a user to their system's unique ID, and the second is a sensor monitoring loop. The sensor monitoring loop will act on a 1 hour delay that will periodically send water and nutrient container fluid level and pH sensor data to the database every cycle.

Figure 4: Software Block Diagram



2.3.4c Design Sketch

From different design attempts, we decided to stay with the design below due to its simplicity and containment thus providing space savings. As shown, this is a two-tank drip system that would rely on gravity for solution delivery. Additionally, pumps are implemented for solution mixing and delivery prior to reaching the plants. Each solution concentration is maintained isolated from the rest with their own dosing pump and tank. Eventually, during build and development, we changed the design one more time, immoting some features such as the top tank, UV light and electrical water flow sensor as they were deemed unnecessary to achieve our goal. Additionally, we relocated the dosing tanks near the bottom mixer tank to reduce priming times and get a more accurate dosage. In total, this system is small enough to both be transported and powered easily for testing purposes.

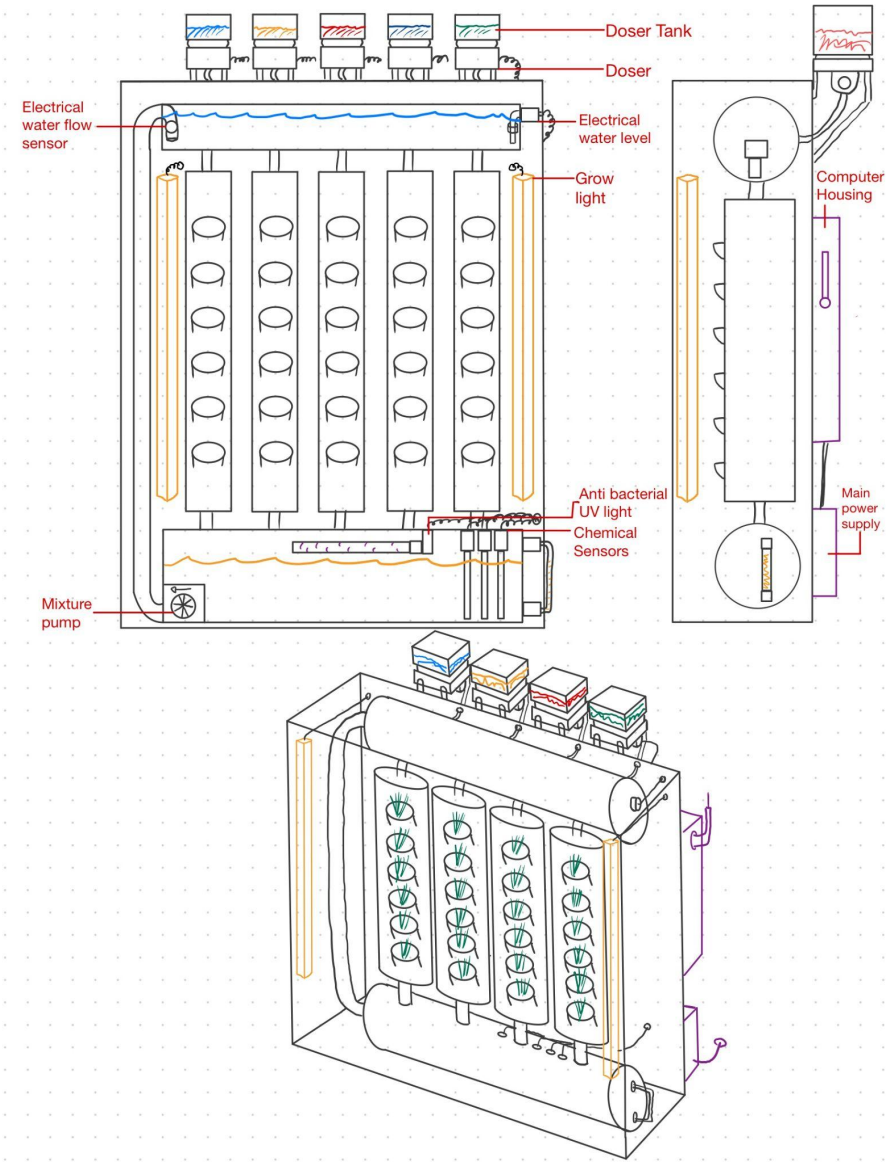


Figure 5: Design Sketch

3.0 Research

In this section, the main functional components of the software and hardware sides of the project were researched and analyzed in order to better understand the design requirements to be satisfied, as well as the benefits and drawbacks of certain component selections and what constraints and standards we had to consider when building this project.

3.1 The Microcontroller Unit

The Microcontroller Unit (MCU) is the driving force behind automating the hydroponic system. The MCU is a low-power computer with systems to allow it to interface with external devices. It must be able to control turning the lights, pumps, fans, and sensors on and off. Ideally, the MCU has multiple power settings and ways to bring it in and out of low power modes. The MCU has software libraries that make it easy to program.

3.1.1 Arduino vs Microcontroller Unit

Arduino is an open source hardware and software company that produces ready-made circuit boards to consumers. In this case, we chose a standalone MCU to build this project because using an Arduino board does not meet the complexity requirements for a passing grade. The MCU required its own circuit board design to connect to each of the peripherals.

3.1.2: Comparison of CC3220MODASx and MSP430FR6989

Shown below is a table that summarizes the comparisons of each considered MCU.

Table 2: MCU Comparison

	MSP430FR6989	CC3220MODASM2MONR
Power Requirements	3.0V	3.3V
CPU Speed (MHz)	16	80
RAM (KB)	128	256
GPIO Pins	83	63
Network Module Included	No	Yes
Price	$3.187	$9.047

3.1.2a: Advantages

The CC3220MODASM2MONR has a faster processor at 80 MHz and the MSP430 has a 16 MHz processor. Theoretically, a 16 MHz processor should be plenty but we are confident the instructions will process at a faster rate. The CC3220 has twice the FRAM of the MSP430 at 256KB. This allows us to store more data from our sensors in the event network connectivity fails.

Most importantly, the CC3220 has an integrated network processor with its own antenna. If we used the MSP430, we would have to purchase a Wi-Fi module and a compatible antenna. This saves on board space. The network processor on the CC3220 has many built-in security features for the programmer to utilize, which I will go into in detail later. Configuring the network processor was also an easier experience to connect our system to the internet. The CC3220 LaunchPad development kit has code examples showing how to write programs using the network processor, so we had references as to how to use the Bluetooth and Wi-Fi modules. Thankfully, this simplified writing the software and reduced the time spent troubleshooting problems.

3.1.2b: Disadvantages

The CC3220 has less GPIO pins at 30, but it had enough space to use the required sensors, and we had the option of adding more on later through the use of GPIO expansion chips. The average power consumption of the CC3220 is slightly higher at 3.2V-3.6V. However, the extra network module we would've had to use with the MSP430 would've required some power.

Placing the CC3220MODASM2MONR had more strict requirements when placed on the PCB, which means extra care and consideration went into our board layout so it functions properly.

In our classes, we worked with the MSP430 so some experience won't transfer over. There were some unforeseen consequences in working with different technology. However, we were able to overcome them partly because the CC3220 had a similar testing and development environment, so the growing pains were minimal.

3.1.3: Choosing the MCU

We chose the CC3220MODASM2MONR for the automated hydroponic system. The CC3220 class of microcontroller units from Texas Instruments has SimpleLink™ technology, which has two processors in one chip. One is the main processor that drives the application environment as any other MCU would do, and the second is a network processor that handles the Bluetooth and Wi-Fi connection environments. Having both processors in one chip saved on board space at the cost of more restrictive board space. The following information is taken from the CC3220MODAx data sheet [1].

The CC3220MODASM2MONR is an Arm Cortex-M4 MCU. Working in an Arm environment is a new and more current skill to have compared to our experience working with the MSP430. Working in this environment would've allowed us to expand our skill set, which would've made us more attractive to employers.

This MCU comes with a built-in antenna, which increases the MCU width by five millimeters. Because it comes with its own antenna, we didn't need to purchase one and the associated parts to make it function properly. However, we did have to adjust the placement of the MCU according to the data sheet.

3.1.3a: CC3220MODASF LaunchPad Development Kit

The LaunchPad is Texas Instruments' development kit. It contains a PCB with a functional MCU as well as a few peripherals to use to develop programs as practice. It comes with a JTAG to easily load and test programs and a USB cable to connect to a computer. Texas Instruments also sells a variety of booster packs which range from LCD screens to sensor boards that connect to the LaunchPad.

3.1.3b: Software Development Environment

The software development environment is an important thing to consider when creating software. Fortunately, Texas Instruments' Code Composer Studio (CCS) had all the necessary tools to write software for any of their MCUs as well as the software to load instructions to an external MCU.

To get a project started, open CCS and select a new project. A drop-down menu will show on the screen where the CC3220MODASM2MONR can be selected as the MCU we are developing software for. It then opens a new file with the main function as well as the header file that contains all the definitions necessary to quickly develop software for our MCU.

Testing our software on the LaunchPad was as easy as clicking the build button and connecting the USB from our computer to the LaunchPad. It's important to define variables that can be easily changed as the LaunchPad has pins connected to LEDs, temperature sensors, and an accelerometer. This means hardcoding pin numbers was the wrong way to approach testing our software. From here, we were able to observe if the desired outcomes were achieved and used the debugging software in CCS to step through our program.

3.1.3c: Loading program to stand alone MCU

All CC3x20 devices must be programmed with a gang image. The gang image is the final software product ready to be loaded to the MCU. There are two methods for programming the CC3220, one of which is Direct SPI, which is writing directly to serial flash through SPI and it requires a programming tool. It's assumed the programming tool can be Code Composer Studio or UNIFLASH. The other is by UART. This requires a PC-based utility or embedded device that can be used for

indirect programming via UART. When configuring using UNIFLASH, the MCU should be set in production mode.

When the MCU boots, it should detect the gang image and convert it to the target file system. This process is handled automatically and does not need an external interface. However, the gang image and the conversion need enough memory to exist in FLASH memory at the same time. With 256KB of FLASH memory, this should not be an issue. When a factory reset is initiated, because the gang image is retained, we can reset the MCU to its default state without reloading the program.

3.1.3d Programming via SPI

From the CC3220 Production guide [2]:

- The SPI pins must be brought out for physical contact via pin headers or test pads.
- The SPI pins must not be driven by other sources during programming.
- The MCU is held in reset to prevent I/O contention.

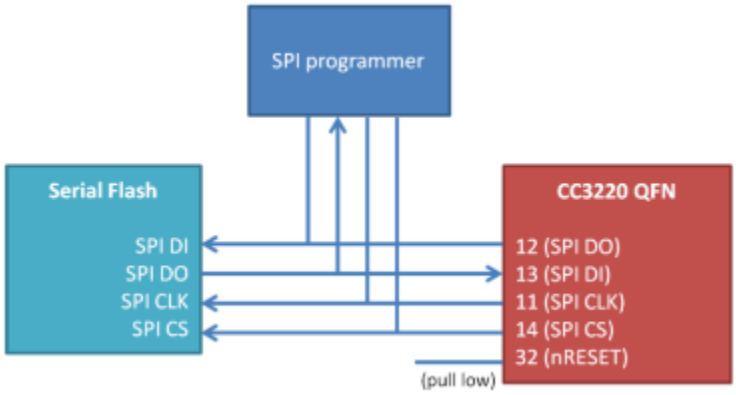


Figure 6: SPI Programming Block Diagram

3.1.3e Programming via UART

- The CC3220 must be put into UARTLOAD or UARTLOAD_FUNCTIONAL_4WJ boot using the SOP pins.
- The following pins must be used:
  - 55 - UART1 TX
  - 57 - UART1 RX
  - 32 - nRESET
  - 35, 34, 21 - SOP pins
- The UART configuration must be:
  - Baud rate: 921600
  - Data bits: 8 bits
  - Flow control: None
  - Parity: None

- Stop bits: 1
- Polarity: Positive

Table 3: Programming Over UART

Mode	SOP[2]	SOP[1]	SOP[0]	
UARTLOAD	Pullup	Pulldown	Pulldown	Factory,lab flash,and SRAM loads through the UART. The device waits indefinitely for the UART to load code.The SOP bits then must be toggled to configure the device in functional mode.Also Puts JTAG in 4-wiremode.
UARTLOAD_FUNCTIONAL_4WJ	Pulldown	Pullup	Pulldown	Supports flash and SRAM load through UART and functional mode.The MCUbootloader tries to detect a UART break on UART receive line. If the break signal is present,the device enters the UARTLOAD mode.Otherwise,the device enters the functional mode.TDI,TMS,TCK,and TDO are available for debugger connection.

3.1.4: Hardware Security

Hardware security is an important issue when choosing an MCU. It's especially important when our embedded systems are connected to the internet. The CC3220 separates the networking environment from the main processor because the networking environment executes on its own processor. The JTAG and Debug ports can be locked to prevent the loaded program from being changed.

3.1.5: Networking Environment

The CC3220 can be used as an access point or connect to networks over Wi-Fi with IEEE 802.11b/g/n standards. 802.11b/c have a maximum data rate of 11 Mbit/s and 54 Mbit/s respectively and operate in the 2.4 GHz band. 802.11n has a maximum data rate of 54-600Mbit/s and can operate in both 2.4 GHz and 5 GHz bands.

3.1.5a Security

One of the goals we had for this project was to learn skills used in industry. Security is often glossed over in our college classes, so protecting the device and our web application from bad actors was a skill worth learning.

Fortunately, the CC3220 networking processor has a multitude of security features. To start, it has an HTTPS server, which uses Secure Socket Layer (SSL) or Transport Layer Security (TLS) which is a secure way of knowing the web address accessed is authentic. This protects our device and our web application from man-in-the-middle attacks, which could corrupt our data logs and harm the intended operation of our hydroponic system. Next, Texas Instruments has a catalog of trusted root certificates which is a list of certificates of trusted entities. The CC3220 has a cryptographic engine, which uses a root of trust public key. It has 6 Secure Sockets with 256-bit AES encryption for SSLv3, TSL1.0, TSL1.1, TSL1.2 certificates. Finally, it has AES, TKIP, and WEP Wi-Fi security.

3.1.5b Interface

The MCU can be used as an access point via the phone app by connecting a phone's Wi-Fi network to the MCU (mysimplelink-XXXXX). In this mode, there are some demo applications via http://mysimplelink.net that take advantage of the LaunchPad's onboard temperature, accelerometer, and LEDs to aid in software development.

The networking module can be configured with TI's SimpleLink™ Wi-Fi Starter App. To start development with the LaunchPad, the jumpers have to be set to the Network Connection Mode, which can be found in the LAUNCHCC3220MODASF data sheet.

When the LaunchPad is in this mode, the network name and password can be entered. Press the start button on the phone, and if it is successful, there will be a positive notification and the network credentials then saved in the network processor.

3.1.6: Pin Attributes and Interfaces

When considering what the pin layout will be for a project, the totality of peripherals' interfaces and pins will be needed. In the case for this project, we needed an I2C interface for the LCD, an Analog-to-Digital Converter for the pH sensor, and GPIO pins to turn on the various motors, pumps, lights, etc. In this section, we explain our thought processes and our conclusions for the designed pin layout.

3.1.6a Pin Overview

The switches and sensors work through the General Purpose In/Out pins (GPIO) and Analog to Digital Converters (ADC) respectively. The CC3220 has 30 GPIO pins. Each pin has a digital mux value that allows it to be used for different purposes. The pin attributes and multiplexing table is available on page 18 of the CC3220MODAS data sheet.

According to the CC3220MOD data sheet, we have four Analog-to-Digital converters to work with as well as 2 sets of 1^2C and UART modules. This leaves us with plenty of GPIO pins, for push buttons and turning devices on and off via voltage-controlled switches.

3.1.6b GPIO Pin Budgeting

The amount of GPIO pins available to interface with all the necessary hardware and software components of the hydroponic system had to be kept in mind while the project continued to progress. As new features were considered and hardware components were selected, it was always ensured that they did not require more GPIO pins than were available. This was a somewhat difficult task to accomplish during the research phase of the project, as many systems still required research and components that have been already selected may have required implementation methods that occupied more GPIO pins than expected. However, there were several solutions to pin shortages if the project's development encountered one, each with their own difficulty of implementation.

One straight-forward method of solving a shortage of GPIO pins was to change the selection of hardware components to use fewer pins. For example, swapping a display screen that requires ten pins to a simpler model that only requires three frees up a significant amount of pins for other systems to use in exchange for a less sophisticated display. Another possibility was to remove a feature altogether, reducing the unit's overall functionality but increasing the ease of implementation. These solutions were only possible, though, if there was flexibility in the hardware component selection at the time.

In the event that the hardware component selection was inflexible, several solutions did exist to stretch the use of the available GPIO pins. Through the use of multiplexers, it became possible to interface multiple components to a smaller number of pins. The same task can be accomplished by using shift registers as well. Being that both solutions required the use of additional hardware components, both solutions increased the PCB design complexity and real estate required, which led to higher production costs and larger labor requirements. Each of these solutions required different implementation approaches and had their own functional limitations, but were the cheapest of the hardware solutions to the shortage of GPIO pins.

Rather than stretch the use of a set number of GPIO pins, the most simple and direct way to solve a pin shortage was to expand the total number of pins. I/O expansion chips allowed for a set of external I/O pins to be added to the number of pins the MCU is able to interface with, effectively increasing the total number of GPIO pins. These chips, such as the MCP23S17 produced by Microchip, are able to be controlled by the MCU through different flavors of communication protocols, commonly using either I^2C or SPI. Using I/O expansion chips provide a slightly more expensive solution than multiplexers and shift registers, but are still considered low cost. Using multiple MCUs was also a valid method of adding GPIO pins, but was the most costly and difficult solution for several reasons. Each MCU would have to be programmed either separately or in some form of parallel format, which complicates the program flashing process and introduces more room for error. Separate programs for each MCU would have had to be written, each with their own communication routines handling how they interface with each other. Having multiple MCUs also significantly increases PCB trace routing complexity and real estate requirements, which in turn would raise manufacturing costs and development time.

An important fact that was kept in mind as the number of required GPIO pins for the project was evaluated was that several hardware communication protocols allowed for connections to be made using a common bus. Conveniently, I^2C, which the project's selected MCU supports, is capable of bus communication by using only two shared traces. SPI, another protocol supported by the selected MCU, also supports bus communication, but requires the use of at least four shared traces depending on the configuration. Through the use of a bus, multiple devices are able to interface with the MCU by sharing the same few communication pins, removing the need for each device's data transfer pins to be mapped to unique pins on the MCU. This would result in fewer overall GPIO pins required, provided that the project's implementation included shared hardware communication buses.

To aid in the budgeting of GPIO pins during the project's development process, the following table has been included. The configurations for each pin will also be provided in order to describe their required functionality.

Table 4: GPIO Pin Budget

Component/Function	**# of GPIO Pins Required**	**Pin Configurations**
pH Sensor	1	5V, GND, ADC
Water Level Sensor(s)	1	+5V, GND, In
Doser Controls	3	5V, GND, Doser
Light System	1	+V, GND, In

LCD Display	2	5V, GND, SCL, SDA
Power Relay Switches	5	+V
Total GPIO Pins Required (30 Available)	7	N/A

3.1.7: Timers

The CC3220MODASM2MONR has two 16-bit general purpose timers (Timer A and Timer B) that can be configured and/or combined to create one 32-bit timer. It has 5 operating modes. They are the one-shot timer, a periodic timer, a 16-bit general purpose timer with 8-bit prescaler, a 16-bit input-edge count or time capture modes with 8-bit prescaler, 16-bit pulse width modulation (PWM) mode with 8-bit prescaler and software-programmable output inversion of the PWM signal. Each of these can count up or down and can be stalled in debug mode with the CPU Halt Flag. Time can be determined between the timer interrupt and entry into the Interrupt Service Routine (ISR). Each timer runs from the system clock (80 MHz).

These timers would have allowed us to periodically switch between power modes when sensor data is collected. In the final iteration of the project, however, the use of different power modes were not implemented.

Table 5: Memory Addresses for General Purpose Timers

Starting	Ending	Description
0x4003 0000	0x4003 0FFFF	Timer A0
0x4003 1000	0x4003 1FFFF	Timer A1
0x4003 2000	0x4003 2FFFF	Timer A2
0x4003 3000	0x4003 3FFFF	Timer A3

3.1.8: Power Requirements

To power the CC3220, we needed to make sure we could maintain a voltage of about 2.8 - 3.3V. When choosing peripherals, we simplified our design by choosing those with a similar voltage requirement. Otherwise, we would’ve needed a way to step up or step down the voltage between the MCU pins and the peripheral sensor pins.

In the final implementation of the project, we ended up requiring a small step down circuit to lower the output voltage of the pH sensor to be compatible with the MCU's 3.3V requirement.

Table 6: Power Requirements

	MIN	TYP	MAX	UNIT
V_{BAT}	2.3	3.3	3.6	V
Operating temperature	-40	25	85	°C
Ambient Thermal Slew	-20		20	°C/minute

3.1.9: Power Modes

The CC3220MODASM2MONR has many different power modes available to the programmer. The Network MCU has its own set of power modes separate from the main MCU to allow it to idle while waiting for a signal. Current consumption is based on an average temperature of 25°C and V_{BAT} of 3.6V.

Table 9: Main MCU Power Modes

Mode	Details	Current Consumption
Active	MCU executes code at 80-Mhz state rate	190-272 mA NWP active 15.3 mA NWP idle
Sleep	MCU clocks are gated off and the last state of the device is retained. It can wake up from external events (GPIO), the internal timer, or by Real Time Clock (RTC).	56-269 mA NWP active 12.2 NWP idle
LPDS	State information is lost and only certain MCU-specific registers are retained. It can wake up from external events (GPIO), internal timer, or RTC with a wake-up time of less than 3ms. Certain parts of the memory can be retained with some configuration.	53-266 mA NWP active 135 µA NWP LPDS 710 µA NWP idle

Hibernate	Lowest power mode where all digital logic is power-gated. Supports wake-up from external events or from the RTC. Wake-up takes 15ms plus the time to load the application from serial Flash.	5 µA
Shutdown	All device logics turn off including the RTC. Wake-up time takes 1.1s. To enter this mode, the state of the nRESET line is changed (low to shut down, high to turn on).	1 µA

For our purposes, active and sleep modes seemed the most attractive. Between data collection and sending the data to the external database, we had the ability to put the MCU in sleep mode and wake up the device from the real time clock to active mode. From a power-saving standpoint, the other modes seemed attractive and were things to consider because we were saving sensor data off the device.

Table 10: Networking MCU Power Modes

Mode	Details
Active Mode with processing layers 1, 2, 3	Transmitting or receiving IP protocol packets.
Active Mode with processing layers 1, 2	Transmit or receive MAC management frames; IP processing not required
Active Listen	Power optimized mode for receiving beacon frames
Connected Idle	Composite mode that implements 802.11 infrastructure power save. This mode allows the network to wake up to beacon packets while in LPDS mode and switch to active mode if one is received. Otherwise, it will return to LPDS mode.
LPDS	Low power mode between beacons. The state of the device is retained by the network processor for rapid wake-up.
Disabled	Network processor is disabled.

For our purposes, we were able to use Active Mode to communicate between our application and this device periodically and send it into Connected Idle mode when vital processes were finished.

3.1.10: MCU Board Placement

Because this CC3220 MCU comes with its own antenna, placing it on the board required special consideration because wiring could've affected the radio frequencies of the antenna. The CC3220MODAS data sheet states the required considerations are as follows:

- The module must have an overhang of 1 mm from the PCB edge.
- The module must have a 6-mm clearance on all layers (no copper) to the left and right of the module placement.
- There must be at least one ground-reference plane under the module on the main PCB.
- The module is designed to be soldered to a thermal pad on the board.
- Wires must not run underneath the module on the same layer.

3.1.11: Conclusions

The CC3220 MCU is the control center of our project. It controls the data collection functions and the operation of the lights, water pump, and pH control. These devices have various communication schemes such as I2C, UART, and SPI or by simple GPIO. The process of loading the main driver is able to use UART or SPI with Texas Instruments software. What attracted us to this CPU is the integrated network CPU with antenna and related software interfaces. The ARM processor is fast at 80 MHz and has enough GPIO pins to complete the project. It has various internal timers we can use to control the operations of the Hydroponic Unit and power modes to be more power efficient. The only downside is the placement of the MCU has to be placed near the edge of the PCB to prevent radio frequency interference.

3.1.12 Data Transmission

Communication between the MCU and peripherals was integral to building a successful project. There are multiple standards for communication types: Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), and Universal Asynchronous Receiver Transceiver (UART). Peripherals that transmit bytes of data utilizes one of these standards of communication.

3.1.12a I2C

I2C is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial communication bus. This means that the timing between devices must be synchronized to process data. Multi-master means that multiple devices can start data transfer and multi-slave means there can be multiple devices that can be communicated with at once. Data is transferred in packets, which ensures data transfer happens via acknowledgement (ACK) or non-acknowledgement (NACK) bytes. Single-ended means it operates by measuring high and low voltage, but can be susceptible to noise.

Typically, I2C uses Serial Data Line (SDA) and Serial Clock Line (SCL) with pull up resistors with voltages being +5V or 3.3V [3]. Transfer rates are 100kbits per second to 400 kbits per second for fast mode. This assumes perfect conditions between master and slave. To start and end data transmission, a START and STOP signal is transmitted. There are four potential modes:

- Master transmit - Master is sending data to slave
- Master receive - Master is receiving data from slave
- Slave transmit - Slave is sending data to master
- Slave receive - Slave is receiving data from master

This means that communication is only between master and slave devices. The START signal is sent followed by a 7-bit address of the slave and 1-bit, 0 to read, 1 to write to the slave device. If transmission is successful, an ACK bit is sent back acknowledging the address. Then, the master can initiate data transmit or receive modes, with the slave using its opposite. When data transmission is finished, the STOP signal is sent and the process must be initiated again when transmission is required.

Because I2C is synchronous, there often needs to be a delay on the master device between the START signal and beginning data transmission. Timing diagrams are available in each device's data sheet. Timing can be simulated in the MCU via interrupts or hard-coded delays. We used I2C to communicate with the LCD and display relevant information.

3.1.12b SPI

SPI is a synchronous, 4-wire serial bus consisting of: Serial Clock (SCLK), Master Out Slave In (MOSI), Master In Slave Out (MISO), and Chip/Slave Select (CSSS). SPI can have one master with one or more slave devices. To start communication, the master configures the clock using a frequency used by the slave device. SPI works by using two shift-registers that work off the clock frequency. The master sends a bit on the MOSI line to the MOSI pin on the slave, while the bit on the MISO line of the slave sends a bit to the MISO pin on the master device.

To utilize multiple slaves, the MOSI and MISO lines are shared to the slaves and the CSSS pins are individually connected to the desired slave. To access the desired slave, the CSSS pin on the master and slave must be raised. SPI is used for communication with the networking processor.

3.1.12c UART

UART is an asynchronous communication hardware device. The speed of data transfer is configurable. Data is sent bit by bit from least significant to most significant and is framed by Start and Stop bits over a serial port. Data transfer

can be simplex, half-duplex, or full-duplex. The UART hardware is controlled by an internal clock signal which runs at 8 or 16 times the bit rate. Because it is asynchronous, the clock signals do not have to be synced but the bit rate of data transfer must be properly configured. A mistimed rate can improperly display the data. Like SPI, UART uses shift registers and parity bits to ensure the data transfer is correct. UART is essential to calibrating our sensors and testing the MCU software is functioning as intended. We used UART to help with the debugging process and to send the MCU image to the chip.

3.2 Online Application & Connectivity

The primary motivation for adding Internet connectivity to the Hydroponics Unit is user convenience. By allowing the sensor data gathered by the unit to be sent to an online database for storage, several new features become possible. The user may now monitor system data from a mobile or web app, or even send commands to the system wirelessly. With the data being stored in a database rather than locally on the unit itself, it becomes much easier to store large amounts of data, allowing for enough data to populate a graph to display to the user. This graphical display of data could be used for trend analysis for things such as power consumption, nutrient consumption, and pH fluctuation. Through an easy to digest graphical display, the user would be able to efficiently monitor the unit's performance over time. This would be more difficult to do if the user was required to only read data from a small display screen located on the unit. Due to cost and time constraints, only the pH level and status of hardware systems were shown in the graphical display of the mobile app.

Convenience is also the driving factor behind choosing wireless Internet connectivity versus a traditional cable connection. Hardwired Internet connections are often less desirable in a home setting, as the positioning of routers tend to make wiring an Ethernet cable from the router to a device difficult. Using a physical connection can lead to tripping hazards or unsightly wiring, but these difficulties are removed for the user if the Hydroponics Unit can connect to the Internet via WiFi. On the production side, it is convenient to include wireless Internet connectivity for the reason that MCUs with built-in wireless capabilities already exist in the market for reasonable prices, such as the Texas Instruments CC3220.

3.2.1 Database Engine Selection

One of the many decisions made during the development of the Hydroponics Unit was the selection of a database in which to store sensor data for each user. Most, if not all, types of databases were functional for the type of data intended to be stored, but a few types were more naturally fitting for the project's requirements. By stepping through a typical sensing cycle of the Hydroponics Unit, the more logical database choices became apparent.

Each sensing cycle is triggered according to a defined schedule, where cycles will occur at regular time intervals. Once a cycle is initiated, each sensor samples its environment and acquires data from a particular system of the Unit. The sensor data is then gathered and formatted into a table and assigned a timestamp, which allows for data samples to be organized chronologically for trend analysis.

The data format remains constant for every sample, revealing that the data naturally follows a strict-schema. Each user is able to view their Unit's data for a more focused and efficient user experience. However, a feature that was under consideration was the possibility for multiple users to view the same Unit's data if they have been authenticated as a valid owner. This would've allowed for multiple users in the same living space to interact with the Unit's internet functionality without needing to share a single validated account. These user-to-database interactions form either a one-to-many or a many-to-many relationship between each user and their Unit's data. Being that the data is relational in nature and uses a strict-schema, it follows that a relational database using SQL fits the Unit's data storage needs. The specific SQL database service selected was MySQL, as it was the most appealing option considered due to its open-source licensing and the team's familiarity with it.

3.2.2 Data Transmission Over the Internet

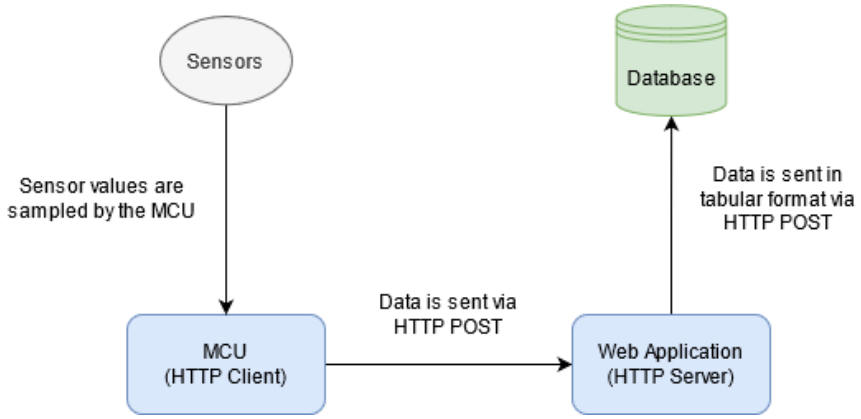
One of the main challenges of interfacing the Hydroponics Unit and the web application was in how data transmission between them was handled. As per the design requirements, the data traffic was required to be facilitated over WiFi and not rely on any cable connection. Two main types of communication needed to occur: the transmission of sensor data from the Unit to the database, and the transmission of hardware commands from the mobile application to the Unit. Each form of communication used different implementation methods, and is discussed in more detail in the following sections.

3.2.2a Transmitting Sensor Data

For the uploading of sensor data, ideally, the Unit's system controller would've been able to perform POST operations directly to the database without requiring its data to be fetched by an outside software routine. However, the CC3220 does not support PHP scripting and is unable to upload data to the MySQL database directly. The solution involved setting up an HTTP client on the MCU to make POST requests to the web application. Through the POST request, the MCU sends its sensor data payload to the web application. The web application, acting as the HTTP server, receives the requests and makes another POST request to the database to deposit the sensor data payload. Thankfully, Texas Instruments had several libraries of functions available that were compatible with the CC3220 that perform HTTP requests, as well as an example software project that demonstrates the use of these functions.

The flow of the sensor data transmission process from the MCU to the database may be better visualized through the following diagram.

Figure 7: Sensor Data Transmission Flow

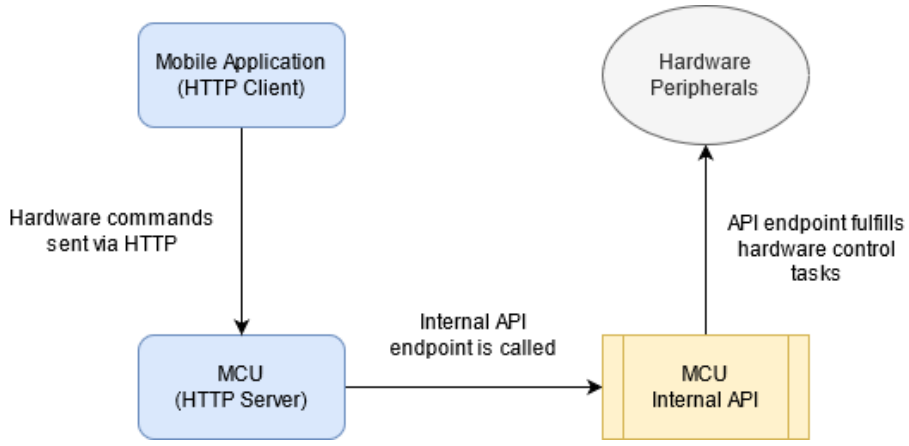


3.2.2b Transmitting Hardware Commands

The method in which hardware commands are sent from the mobile application to the Hydroponics Unit is similar to the method in which sensor data is transmitted to the database in that HTTP is used, but the roles of the MCU and mobile application are reversed. The MCU now functions as the HTTP server while the mobile application makes requests to it as an HTTP client. Naturally, this requires the setup of an HTTP server to be run on the MCU at all times in order to have it be constantly available for receiving requests. These HTTP requests made by the mobile application function much like any other HTTP request, requiring a URI to reference the API endpoint that fulfills the request as well as the setting of request headers. The requests are received by the MCU and fulfilled through calls to internal API endpoints, which interact with the Unit's hardware and complete the user's desired tasks.

The following diagram depicts the transmission flow of hardware commands from the mobile application to the MCU over HTTP.

Figure 8: Mobile App Hardware Command Transmission Flow



As with the process of making HTTP calls from the MCU, example projects from Texas Instruments were available that demonstrate proper setups for running an HTTP server from the MCU and implementing internal API endpoints. These were invaluable to the project's software team, as the concept of running a server from a microcontroller was unfamiliar territory. One small consolation found when implementing the MCU's internal API is that each supported hardware command and use case was fairly unsophisticated. Most hardware commands simply toggle a hardware system on or off, which involves setting the status of a single GPIO pin, or call an already defined software routine to be performed outside of its regular schedule, such as requesting a sensing cycle to take place immediately.

3.2.3 Web & Mobile Application Technology Stack Considerations

Quite a few different web application stacks were available to choose from, each with their own benefits and drawbacks. The choice of stack impacted many different areas of the web application itself and its development process, thus its selection was well-deliberated. To better understand what stack was more appropriate for the project's needs, several stacks were compared to each other according to how they impact certain key concepts. These key concepts are: web and mobile cross-platform compatibility, team familiarity, ease of development, functional capability, and the opportunity to learn and develop skills.

3.2.3a LAMP Stack

The LAMP stack, being composed of the Linux, Apache, MySQL, and PHP software layers, was a solid choice for building the web application upon. This open-source software stack has remained popular for many years, which has allowed it to build a large community of support. As a result, a wide variety of software modules and plug-ins have been developed to allow for extended and customizable functionality. Furthermore, the project's team has no shortage of reference material to help further our understanding of the stack and developmental capabilities with it.

Another convenient benefit of the LAMP stack's popularity was the availability of hosting options. For example, DigitalOcean, a popular hosting platform, allows you to purchase hosting and install a LAMP stack configuration on the server with a simple one-click procedure. This would simplify the setup portion of the web app development process, saving both time and effort and potentially avoids the debugging of a faulty setup.

As mentioned for the database layer, it traditionally uses MySQL, which had already been selected as the preferred database for the project. This would have avoided the need to reconfigure the stack to use a non-standard database, which in turn, would have allowed for a simpler development process. Also contributing to a simpler development process is the use of PHP, which the project team is already experienced in using.

When considering cross-platform compatibility, the LAMP stack falls short of other alternatives. The code used in LAMP stacks is not directly portable to mobile environments, raising the question of how would a mobile app based on an existing LAMP application be developed? It was possible to use a cross-platform framework, such as Ionic, during development to allow for portions of the code to be recycled across platforms, but some manual coding would still be required. Another solution involved developing a separate mobile application in parallel to the web application using the mobile platform's native coding language. This particular solution was the most labor intensive, but a small consolation was that the logic behind the web application's code likely would have still applied to the mobile environment.

Experience with the LAMP stack is something all members of the project team possess, which can be considered both a benefit and a drawback. The benefit arises from the team's familiarity with the LAMP stack development process, leading to, if the LAMP stack was selected, less time spent learning unfamiliar software technology and more time spent physically progressing the application's development. The drawback arises from the opportunity cost of forgoing the selection of a new and unfamiliar software stack. There is a current trend in the software development workforce that has resulted in a lower demand for LAMP-based applications, as other new stacks have emerged and have gained popularity. More demand for non-LAMP-based applications translates to a higher demand for future employees experienced with other stack configurations. Choosing a software stack other than LAMP would have resulted in more attractive resumes for future employers.

3.2.3b MERN Stack

Another strong contender for the selection of a stack to build the web application upon was the MERN stack, consisting of the following software layers: MongoDB, Express, React, and Node. This stack is also open-sourced, popular, and has a large community of support. Many different software plug-ins exist and

can be added to the Node server layer through the convenient Node Package Manager, providing plenty of functional customization that competes with other stacks' ability to be customized. One unique feature compared to the LAMP stack, is that it uses solely JavaScript for all software layers, which simplifies the development process by avoiding the extra tedium of managing different layers in different scripting languages.

There are also plenty of hosting options available for MERN stacks, though the performance requirements of the server tend to be higher than what LAMP stacks would require. The setup process of the MERN stack is also more sophisticated than a LAMP stack's, leading to more possible sources of a faulty setup, and potentially more time lost during the debugging process.

Though MongoDB is the usual choice of database for this stack, it is possible to substitute MySQL for the database instead through the installation of a MySQL Node Package Manager compatibility module. This is only a minor configuration change, and does not significantly hinder the development process.

The MERN stack has one very large advantage over the LAMP stack, which is its cross-platform compatibility. The use of React Native allows for nearly all of a MERN-based web application's code to be directly ported to a mobile app, though the front end scripting requires some modifications to be compatible. Another very popular framework similar to React Native that accomplished the same task is Flutter, a framework that is well-known for its ability to create well-designed user interfaces. Through the use of these cross-platform frameworks, the mobile app development process can be largely simplified and the time invested into it reduced, an advantage over the LAMP stack that makes this particular stack a very attractive option.

As with the LAMP stack, all project team members possess some experience developing applications using the MERN stack. This would have been a clear benefit if the MERN stack was chosen to build the web application upon, as this particular stack allows for more complicated scripting interactions that require a fairly solid understanding of the framework. However, there is less of a drawback to choosing this stack even if the team is familiar with it already for several reasons. There are several other stacks that are very closely related to the MERN stack, including the MEAN, MEVN, and MEFN variants, that substitute the React software layer with other frameworks. By choosing the MERN stack as the web application's foundation, the project team gained experience that directly transfers to several other stack configurations. Also, all of these stack variations have become increasingly in demand in the software development workforce, thus having additional experience with this stack format improved the team's professional marketability and had little opportunity cost.

Shown below is a table summarizing the differences between the two considered application stacks.

Table 7: Application Stack Comparisons

	LAMP	**MERN**
Programming Languages Required	PHP, JavaScript, HTML	JavaScript
Cross-Platform Compatibility	Low	High
Setup Difficulty	Low	Medium
Career Marketability	Medium	High

3.2.3c Selected Stack (MEFN) MySQL, Express, Flutter, Node

Our mobile application framework is able to:

- Connect to a database
- Connect a user with a specific microcontroller
- Display sensor information
- Display time series data of sensor information into graphs

When choosing a framework for the mobile application, we looked into frameworks we already have experience with. We didn't expect our mobile application to be too complex because we just wanted to display information from a database and leave room to interact with the Hydroponic Unit. From our software development project class, we have experience working with React.js, created by Facebook, which is a JavaScript framework that combines JavaScript, HTML, and CSS into JSX. Working with it compartmentalizes parts of web pages into components, which helps organize the code base and helps quickly build interactive web pages. Another framework we have worked with is Flutter, created by Google, which uses a programming language called Dart, which is an object-oriented programming language with strong types. Flutter, like React, has a hierarchy of widgets that are placed into views. The nice thing about Flutter is how quickly pages can be built with a small amount of code. We chose these options because these frameworks were developed to work in both IOS and Android devices with a single codebase. If we had difficulty developing for one platform, we were able to deploy to the other.

React Native uses a familiar programming language to us in JavaScript. The advantage of coding in JavaScript is there's a third-party library for virtually every common situation we foresaw ourselves getting into. However, anyone who codes in JavaScript knows the freedom that it allows can cause headaches because it is loosely-typed. Starting a project is as easy as telling the CLI to

create a new react app and then using the node package manager to install whatever external libraries are needed. However, there is no one size fits all solution for deploying a project.

Flutter does have some compelling advantages over React Native. It's a strongly typed Object-Oriented language, which can cut down on type check blocks. The advantages Dart/Flutter has over JavaScript/React Native is its rich native component/widget library. When using React, it's common to use a third-party component library, which may not work in the desired manner. Having a native set of first-party widgets is incredibly helpful in building the application because it's thoroughly tested to work across many different interfaces. Another advantage Flutter has is higher performance because it uses the Skia C++ engine. The CLI interface for Flutter can quickly build and deploy our application when properly configured. There is a first-party guide by the developers that can be used for reference. Additionally, Flutter has an integrating testing suite that can perform tests at unit, component, and integration levels.

IOS development on Windows machines can be tricky because IOS apps require an Apple computer to compile IOS code. However, there are some options available. There is an online service called CodeMagic, which connects to a GitHub repository and runs the necessary steps to create the application. Another option is to set up a virtual machine with OSX. This would require us to purchase OSX, convert it to an ISO and run it on a virtual machine. Unfortunately, Apple states that running any Apple OS on anything other than Apple hardware is against their terms of service. None of us own an Apple computer, so we would need to use CodeMagic to build our application. In addition to this hurdle, to put an IOS app on an IOS phone, a $99.69 per year subscription is required.

Because developing for IOS has so many hurdles, for this project we decided to develop for Android. All that was required was to properly set up Flutter with Android Studio, plug an Android phone into the computer via USB, choose the phone as the device and hit run. We could develop for a variety of Android devices with Android Studio's device emulator. The benefit of using Flutter for this application is getting quick feedback on code changes with hot reload where changes can be immediately viewed by the programmer.

As stated before, everyone in our group has experience working with Express, Node and MySQL. Node is a JavaScript framework for creating server applications. Express is a framework that acts as middleware between client and server. It allows for building robust APIs for web applications. The database for storing information is built with MySQL, which is a lightweight relational database system. Finally, the front-end of our application is built with Flutter.

3.2.4 Hosting Services

When choosing the hosting service that our web/mobile application would use, there were numerous options to choose from. As with the evaluation of which

software stack should be used, some of the key concepts that helped determine which hosting services were most appropriate for the project include functional capability and ease of development. Unlike the software stack evaluation, however, cost was a major factor to consider when selecting hosting services.

3.2.4a Reducing Hosting Costs

Many hosting services quickly become very expensive as the capabilities of the server increase. As such, minimizing the performance requirements of the application would help remove the need for more robust server plans, keeping hosting costs low. Certain optimization methods can be used to reduce the hardware requirements of the server, which may involve cutting features from the application, reducing project scope, and writing both efficient and effective programs. To help reduce the memory demand on the application's server, it is possible to offload certain software layers to other hosting services. The most likely candidate for this tactic is the database, as many database hosting services exist that can house data for your application outside of the main hosting service.

Another tactic to reduce server requirements is to host the database layer ourselves, which is made possible by open-source licensing of a selection of database engines. By avoiding the use of a commercial service for the database, the project's data storage is no longer bound by the limitations of the purchased database service tier, but is bound by the limitations of our local hardware. This would allow for a much larger amount of data to be stored, as local memory storage is already available in a large quantity for no cost. There are a significant number of drawbacks to hosting software ourselves, however.

One major drawback is the added responsibility of ensuring the flow of data is secure to and from the local hosting setup, a responsibility that would usually not weigh as heavily upon the project team if a commercial service would be used instead. Large companies supplying a service to a wide variety of customers have the funding and workforce available to thoroughly invest in security, resources that are not as available to a small-scale educational project team. On a related note, personally hosting software for the application negatively affects the trust a user may have in the app's security or reliability. Having a small team of students manage a user's sensitive information, such as the password they use for their app's account, is not as reassuring as having a commercial business manage it, as these businesses likely and regularly deal with many forms of customers' sensitive information with verified success.

The simplest way to reduce the financial burden of hosting is to take advantage of as many free software services as possible, searching for the most cost effective and functional zero cost product plans. Oftentimes, popular hosting companies will offer very basic levels of service for free, or grant a certain amount of free credit to new users to use on paid services.

3.2.4b DigitalOcean Evaluation

DigitalOcean was a solid choice for the project's application hosting service for a number of reasons. The number of hosting plans and customization options available is impressive, with a wide range of price points as well. Having a large variety of pricings provides some financial comfort in the event that a hosting plan would need to be upgraded as the application develops. They are also among the largest and most popular cloud hosting services.

As previously mentioned, it is common for hosting services to offer free services to newcomers, and such is the case with DigitalOcean, who gifts new users with $100 of credit that is valid for a 60-day period. Roughly two months of free service is not ideal for the project's development cycle, though, as the web application required nearly five months of hosting. Nonetheless, it is a significant amount of free funding that is capable of purchasing high performing service tiers, such as their $40 per month CPU-Optimized Droplet that includes hyper-threaded CPUs, 25GB of solid-state storage, 4GB of memory, and 4TB of transfer space.

3.2.4c Heroku Evaluation

Another large player in the cloud hosting market is Heroku, who is known for their developer-friendly app creation and deployment features. Unlike DigitalOcean, however, not as many hosting options are available, and the cost of hosting increases quickly as the server capabilities increase. The basic hosting plans, though, which are intended for newcomers to the Heroku platform and hobbyists, were robust enough for the project application's purposes, and were competitively low priced.

Heroku offers an enticing entirely free tier of hosting service, albeit with much lower performance than their paid tiers. The free tier of service allows for an application to run in their "dyno" Linux containers for up to 1,000 hours of uptime per month depending on how many personal Heroku applications are live at once. One useful feature that helps conserve runtime hours is their automatic sleep feature, which sets an application to an idle status while not in use after 30 minutes of inactivity. The amount of hours provided at no cost is more than sufficient to keep a basic application running with 100% uptime, however. The primary concern with using a free tier of service for hosting such as Heroku's is the application outgrowing the constraints of the server's capabilities. Thankfully, the Hydroponic Unit's web application does not require much sophisticated functionality and is not expected to handle high user traffic or complete intensive processing tasks. In the event that the application became unable to be handled by a free tier hosting service, it was still possible to upgrade the server's service plan to a more capable tier once the app reached the point in development where it was required to perform at higher levels. As a way to mitigate the possible cost of upgrading the hosting plan, the application may be able to be developed primarily in a local environment where hosting is not restricted by your level of service. Only once the app was sufficiently developed and was required to be in

a live, deployed state would the server's hosting plan need to be upgraded. This could have possibly saved the added cost of hosting outside of the free tier for a few months.

Shown below is a table summarizing the differences between the two considered hosting services.

Table 8: Hosting Service Comparisons

	DigitalOcean	**Heroku**
Customizability of Hosting Plans	High	Medium
Price Scaling	Gradual	Becomes Expensive Quickly
Type of Free Service	60-Day $100 Credit	Free Basic Hosting

3.2.4d Selected Hosting Service - Heroku

Heroku, as the chosen hosting service, most suitably meets the web/mobile application's needs while maintaining a very low cost, a feat that DigitalOcean would not be able to accomplish as well. Though both services provide appropriate features and hosting capability, cost is the deciding factor in this decision. By taking advantage of Heroku's free tier of hosting service during the app's development phase, which lasted over the course of several months, hosting costs are minimized. As an added comfort in the development process, the majority of the project team has more experience developing with Heroku than with DigitalOcean, which lessened the amount of time spent setting up and configuring the hosting service.

3.2.5 Software Security Features

As with any software application that interacts with sensitive data and systems, the implementation of robust security features is a necessity. To determine which security features needed to be implemented, a multi-step process was used. First, by first identifying vulnerable systems and information, the project team is able to better understand what processes require secure methods. Such vulnerabilities include: user account credentials, user network credentials, Hydroponic Unit serial numbers, database access methods, the wireless hardware command system, and API calls. Then, by listing possible methods of attack used to obtain sensitive data or sabotage operational systems, specific security measures to prevent malicious behavior can be identified.

The primary security concern with managing a user's account credentials is password protection, which can be achieved through a few features. Requiring a

certain level of complexity in the user's password when they create a new account or update an existing account will help protect against brute-force attacks, drastically reducing the chance that these attacks will succeed. This feature is simple to implement and effective, thus it was included in the mobile application. Another common attack attempting to obtain a user's information is packet sniffing, which intercepts network information as it is being transferred. This form of attack can be protected against by using password hashing algorithms, which obscure the user's correct login information before it enters the network. The complexity of a hashed password may be further increased by adding a salt, a unique string of characters appended to the end of a hashed password, allowing even identical user passwords to be stored as unique values. These security features also prevent bad actors from obtaining users' passwords even after acquiring illegal access to the database. Several different hashing algorithms exist, some more effective than others, but their executions require more processing effort and bandwidth as their efficacy increases. Of the available algorithms, SHA-256 provides an appropriate balance of complexity and performance requirements, and is selected as the mobile application's hashing algorithm. Hashed-password salting is also implemented for added security, using a cryptographically secure random value generator provided by Java.

One distinct vulnerability of SQL-based databases is the SQL injection attack, which attempts to pass off SQL commands as user data in order to interact with the database in some unintended manner. Many types of injection attacks exist, each of which uses a unique approach to perform the attack. When accepting input from the user, a significant number of injection attacks are able to be prevented by using a sanitization routine. The end goal of input sanitization is to ensure that user input never is treated as code to be executed. Quite a few different sanitization methods exist, some of which include checking for invalid inputs, such as null values or unwanted characters, using prepared or parameterized statements, and adding escape characters to certain input characters. At minimum, the mobile application used parameterized statements and checking for invalid inputs, as these features provide a large amount of security against injection attacks.

When server hosting from the hardware side, as discussed in section 3.1.5a, Texas Instruments has provided many built-in security layers for the project's selected MCU and the software. The MCU's network connection uses secure sockets and WPA wireless security standards. The internal server that is run in order to receive and handle hardware commands also is using HTTPS, putting the secure sockets to use in order to facilitate HTTP traffic. These two networking security features provide protection for the data being passed to and from the MCU. The file system managed by the MCU is also protected and includes a tamper detection feature. Conveniently, these features are already implemented on the MCU and are either automatically enabled or only need to be invoked by software routines. Most of the MCU's security features are naturally active while the project's code is running, as many of the SimpleLink™ library functions that

are used have built-in subroutines that enable these protective measures. The internal HTTPS server's operation may require manual configuration to ensure secure methods are used, however.

Securing the API can be done through several methods, one of which is the implementation of an authorization and authentication system. Many different versions of these systems exist, though one of the most popular and secure systems is the OAuth 2.0 protocol. By including this protocol in our software implementation, access to the API can be properly controlled by only allowing users with valid access credentials to make calls. Another way to secure the API is through the use of rate limiting, which monitors how frequently a user makes calls to the API and limits them if the call volume exceeds a specified threshold. Adding rate limiting to both the Express API and the MCU's internal API helps prevent attacks that attempt to overwhelm the server.

In the final implementation of our software, some security features were unable to be implemented due to time constraints. Features such as rate limiting and OAuth 2.0 authorization were not implemented. However, essential features such as password hashing and salting using the SHA-256 algorithm, API validation, email validation, and secure user login sessions were implemented.

3.3 Dosing Pumps

Considering the nature of the automated growth system, correct content mixture plays a key role in the plant's overall health. Additionally, another feature of the system is to provide plant type flexibility. Taking into account these two aspects we needed a mechanical chemical dosing system that could provide different amounts of mixture elements according to the user's inputs. In addition to transferring chemicals from a local chemical tank into the main mixture tank, the dosing pump also needed to provide the correct amount of liquids with small deviation due to minute mechanical constraints. This was achieved through laboratory-grade purpose-specific peristaltic dosing pumps, as they are made for accuracy and ease of integration with different types of systems. The reason behind choosing a peristaltic pump instead of another popular type of pump such as the diaphragm pump was due to the peristaltic pump's advantage for our specific situation. This was chosen by comparing advantages vs. disadvantages for each type listed below:

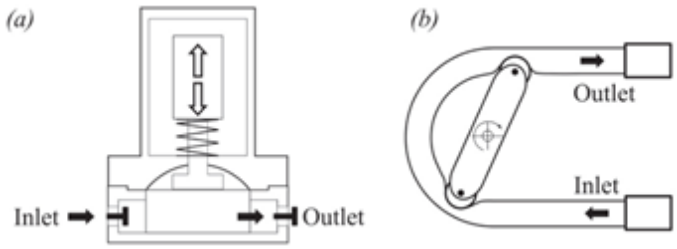


Figure 9: Pump diagram

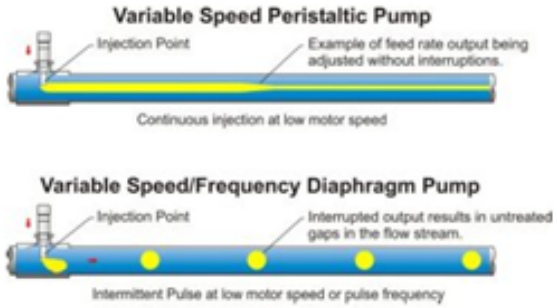


Figure 10: Pumping force differences

Table 9: Peristaltic Vs. Diaphragm pumps

	Advantages	Disadvantages
Peristaltic Pump (b)	• Can operate with dirty or thick liquids. • Handle larger back pressures and easier to prime.	• Uses more power and consumes more energy. • Pump hose has a shorter lifespan.
Diaphragm Pump (a)	• Energy efficient. • Ideal for corrosive and dangerous chemicals.	• Liquid must be particulate free. • Difficult to prime with backpressure.

The dosing pumps were needed to be driven by a 12VDC/~2 Amp variable speed motor. The motor can be connected to an interchangeable pump mechanism made up of at least 3 pump heads to maintain low deviations. The entire pump assembly provides a variable between 1 – 9999ml per day to accommodate a wide variety of chemical mixtures. Each pump has its own enclosed system containing:

- A liquid reservoir
- Inlet and outlet hoses
- 12V DC power source
- Relay/electronic on/off switching module
- Operation light
- Frame support

Individual pumps are in charge of providing their assigned chemicals at different rates, having them as separate controlled units allows for easier design,

management, end-user control/troubleshooting, and system redundancy. The pump needed to provide enough negative force pressure to draw up solutions by at least 5ft vertically. In general, each pump assembly is powered by a DC control board that is controlled by the main MCU in the systems. Due to the pump being peristaltic, interchangeability is a key feature since all peristaltic pumps only one weak point to consider shown below:

Although this is a rare occurrence, being able to quickly interchange the pump assembly will minimize down-time thus preventing critical solution starvation to the plants. Additionally, the mixture tank is placed on a higher elevation than the pump itself to ease initial priming, increase supply efficiency by using gravity and provide an easier method for finding pumping issues while troubleshooting.

3.3.1 Pump Options

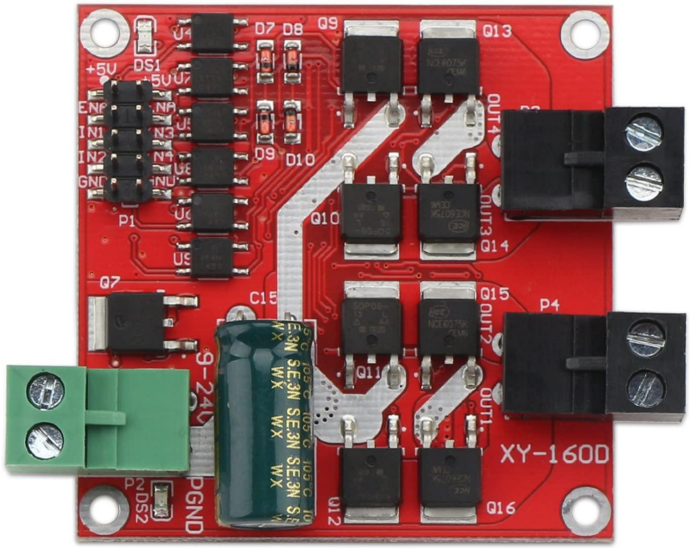
Considering peristaltic pumps, there a number of options to choose from as they come in different varieties such as:

- Single pump
- Multiple pumps within housing
- Pump with integrated controller
- Multiple pumps with a single integrated controller
- WiFi controlled pumps

Due to our design constraints and overall need, we considered using single pumps within their own housing controlled by our MCU through a controller board meant to handle voltage differentials while maintaining logic communication.

Our first option was a simple 12V peristaltic pump that can be controlled via a controller board. Being that this pump is the most basic option, more focus would have needed to be put into the controller board as it would not only need to turn it on but it would also need to control variable output voltages towards the pump to control pumping speed via PWM adjustments.

Figure 11: PWM Capable Controller Board



Ideally, we would have been able to house all the pumps in one enclosure that would also contain each of their controllers. This design plan was to maintain everything manageable for troubleshooting purposes as we had key areas isolated from each other.

We plan to follow a similar design to the one shown below:

This allowed us to place the controller board behind each pump thus making wire management more approachable.

The second option was to use a fully integrated pumping unit that would be connected to our MCU. These self-enclosed units have their own logic, power supply, housing, and communication mediums that are all used in tangent to control each pump. This unit would be able to communicate with our MCU and share simple parameters such actuation and speed commands.

For example the *Jebao Doser 2.4* is a self enclosed unit that has its own power supply and communicates through WiFi. This unit can wirelessly talk to our main unit wirelessly and execute commands as needed. All of the power regulation, actuation, calibration, and overall mechanical control would be done within the enclosed unit. One main factor considered with either design was the cost that we can see on the table below:

Table 10: Pump Price Comparisons

	Individual pump design	Jebao Doser 2.4
12V Peristaltic Pump [x4]	$11.00 [$44.00]	included
PWM controller [x4]	$15.79 [$63.16]	included
Hoses	$12.00	$12.00
Tanks [x4]	$0.00	$0.00
Housing	$40.00	included
Total	**$159.16**	$80.00+$12.00 = **$92.00**

Table 11: Pump Comparisons

	Individual pump design	Jebao Doser 2.4
Supply	1-14,000 ml/day	1-10,000 ml/day
Voltage	12V DC	12V DC
Controller	No	Yes
Self-priming	Yes	Yes
Accuracy	Up to 1ml	Up to 1ml

3.3.2 Pump controller options

For our application, an ideal way to control the pumps was through a DC Motor-Drive Controller Board that could handle 5v to 35v at 2 Amps. There were multiple boards that fulfilled these requirements, and most came with a logic input that would directly connect to the MCU unit. Additionally, an Electronic Trigger Switch and Speed Controller Board could have been used in tandem for better overall control of each pump through the use of PWM signals. Considering that our goal was high accuracy, the use of both of these boards would have provided the best control of chemical solutions pumped into the mixing tank. Considering that GPIO pins on the MCU run on lower voltage at around ~5V, this is not enough to run a pump off of. Thus we would be able to use this pin as a communication medium to control a switching board that can in-turn control the pumps.

3.4 Water Recirculation Pump

The water recirculation pump is one of the main elements of the system itself as its purpose is to move the main mix throughout the drip system. Hence, the drip

system delivers nutrients to the pants in a uniform manner. To achieve the best type of performance different pump designs and placement were researched. For best lifetime and efficient functionality we decided to place this pump within the lower tank. Placing the pump in the tank helps the pump thermally and ensure proper priming.

3.4.1 Specifications

Main factors were considered when selecting a pump for the system:

- 40-80 gallons per hour (GPH)
- 6 foot head height (maximum height the pump is capable of moving water)
- Durability
- Adjustability of pumping rate
- Cost (less than $50)
- Dimension (maximum length less than 1 foot)
- Weight
- Power consumption
- Noise (less than 40 dB)

The biggest priority of these factors was head height. This is because the pump needed to be able to reliably move water from a recirculation reservoir at the bottom of the system to the top of each hydroponic tower. If the pump was unable to do this, then water would be completely unable to circulate through the towers, the plants would not receive any water, and they would get sick or die. Since our system was going to be pumping water around 5 feet from the reservoir to the tops of the towers, the pump selected for the design was rated for a max height of at least 6 feet to make sure there are no system faults.

Gallons per hour (GPH) and the ability to adjust the flow rate of the pump needed to be taken into consideration when selecting a pump make and model for the system. If the water does not circulate through the system quickly enough, it can cause bacterial buildup from standing water in the recirculation reservoir. On the other hand, if the pump moves water too quickly, the reservoir could drain too much before the water has circulated entirely through the system. These factors make a practical pump one that has capabilities to pump more gallons per hour than we require, but also has a flow restricting functionality so that it can be optimized to pump only what is necessary to distribute nutrients through the system.

3.4.2 Types of Pumping Systems

Due to their different benefits and detriments to cost, dimension, and durability, two different styles of pumps are considered for vertical hydroponic systems: submersible pumps and transfer pumps. An example of these models are shown respectively in figures 5.1 and 5.2.

Figure 12: Submersible Pump model

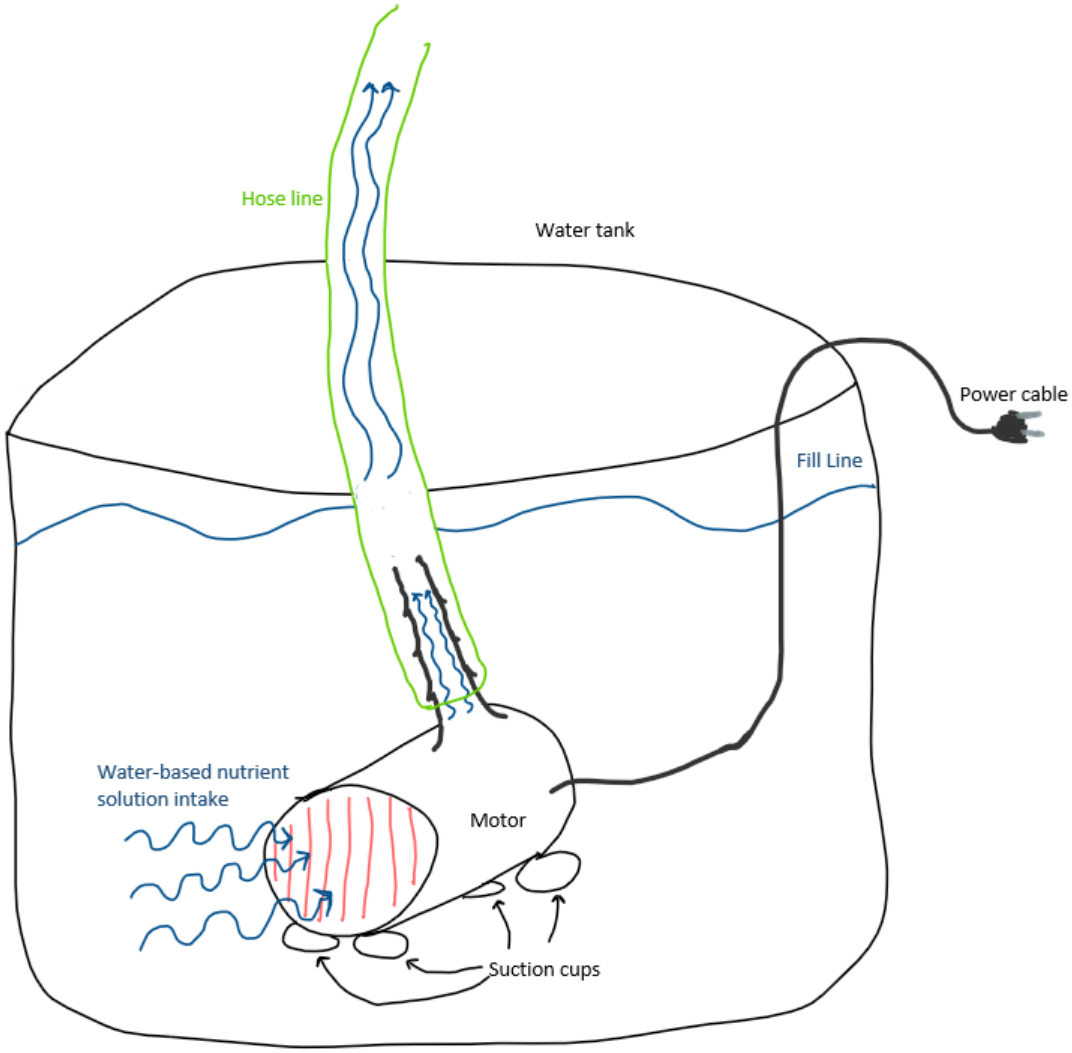
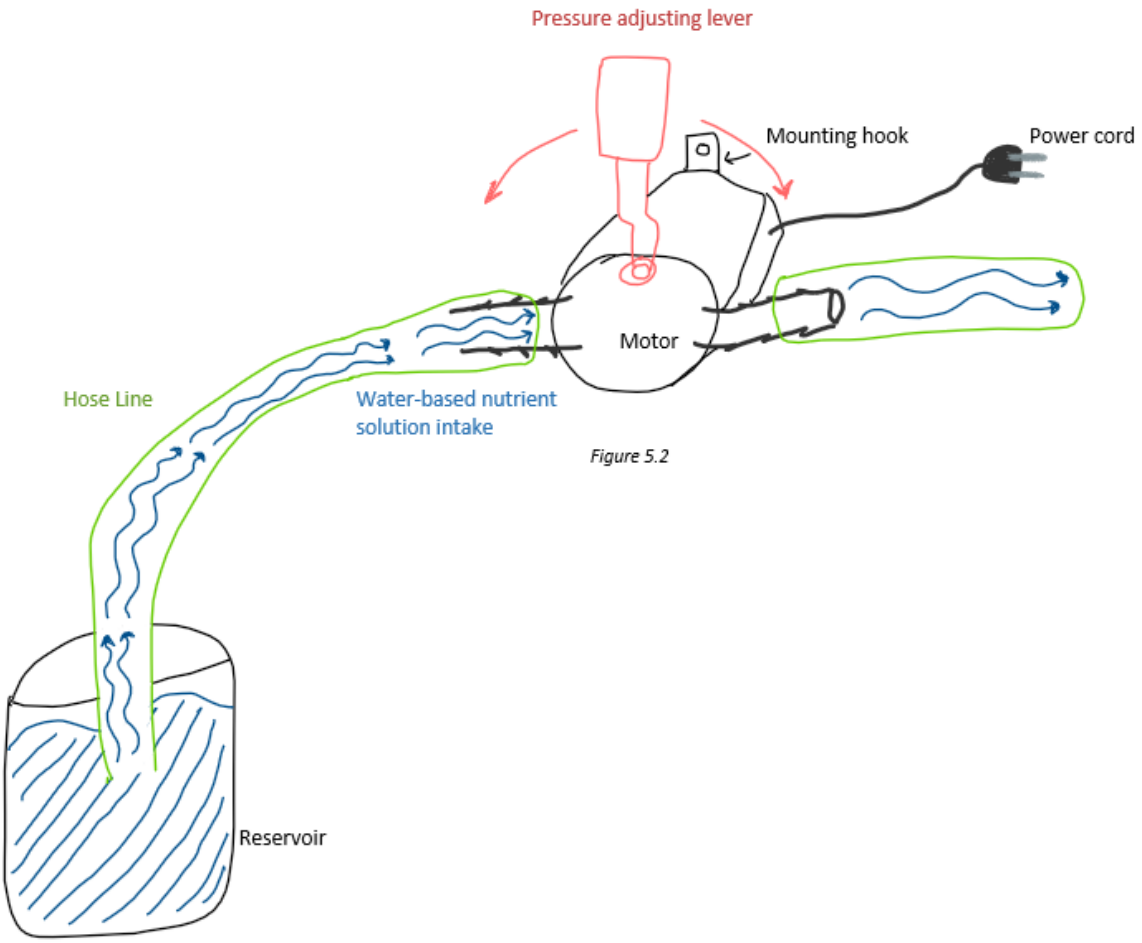


Figure 13: Transfer Pump Model



The submersible pump draws fluid from the bottom of the tank through a filter located near the motor. Suction cups at the bottom of the motor act as a base by preventing the unit from movement while powered on. The filtered fluid is pumped up through a hose line vertically and is directed to the tops of each hydroponic tower in this system. A waterproofed power cable comes out of the reservoir and plugs into a wall outlet.

The main differences of the transfer model from the submersible model is that the transfer has two different hose lines, and the motor is not located underwater. Since the motor is above the water tank, the model typically offers a better flow rate (GPH) than the submersible pump and longer durability due to the infrequency of necessary cleanings to keep the motor running optimally. Having two different hose lines adds a little more overhead, however, because there are more places where water could leak in the system (going in the motor and coming out). The last notable design difference is the mounting hook that fastens the motor to a structure.

Something that was considered was that even between these different styles of pumping, the factors like dimension, cost, noise, power consumption, adjustability of pumping rate, and head height all differed between different products. A generalized analysis of the advantages and disadvantages the different models have concerning the listed requirements is included in the next section, and one of the pumping styles was selected based on the overall cost-benefit.

3.4.3 Comparisons between Submersible and Transfer Pumps

The following table summarizes the comparisons between the two types of pumps discussed in the previous section: submersible and transfer pumps.

Table 12: Average Specifications of Submersible and Transfer Pumps

	Submersible	**Transfer**
Gallons per Hour (GPH)	- High end 1200 GPH - Low end 30 GPH - Lower flow rate due to fan controlled pumping located underwater that takes more power to force water upwards.	- High end 1500 GPH - Low end 300 GPH - Higher flow rate on average since motor is driven above ground by a high-powered impeller
Head Height	- High end 10 feet - Lower power results in lower maximum head height	- High end 50 feet - Higher power allows for much greater head height guarantee
Durability	- Durable, but can get clogged with sediment which requires cleaning	- Not submerged so highly durable - Highly durable and requires little maintenance since unit does not need to be cleaned frequently
Adjustability of flow rate	- Dependent on make and model - Dials offered on some models that accurately adjust flow rate - Difficult to adjust flow rate since unit is located inside of the fluid reservoir	- Dependent on make and model - Adjustability of flow rate offered as a lever attached to motor - Easy to adjust since unit is located above fluid reservoir

Cost	- High end models around $30-40 - Low end models as inexpensive as $15	- High-powered models around $100 - Low-end models $30-40
Weight	- Very lightweight (typical model weighs 1-3 pounds on average) - Usually made of a durable plastic frame to retain submersibility	- Typically heavier than submersible pumps (typical model weighs 5-10 pounds on average) - Weight due to motors and frames being made out of cast iron
Power Consumption	- Dependent on make and model, but on average much lower power - High-end pumps only reach up to around 24 Watts	- Dependent on make and model, but much higher power consumption on average to allow for higher flow rates and head heights - High-end pumps can reach up to 70 Watts
Noise	- Since impeller is submerged the fluid acts as a muffler to the noise produced, giving a quieter pumping sound	- Louder on average since there is no water to muffle the impeller

3.4.4 Decision to Select the Submersible Model

Considering the benefits and detriments of the two different types of pumps, the model best suited for this project's requirements is a submersible pump. While the transfer pump offers a higher flow rate, greater maximum pump height, and stronger durability, the submersible pump offers a more cost-effective way to get the appropriate amount of water-based nutrient solution to the tops of the towers for disbursement.

Since one of the main goals of our project is to design a system that can be used as a sort of wall decoration in a small room such as an apartment, noise is a large factor to consider when selecting a pump. It would be a massive design flaw to create a system that was too noisy for a consumer to want to keep in their home. Submersible pumps seem to have a decisive advantage over transfer pumps in that regard, because the nutrient solution acts as a muffler to the impeller action, making the pumping process almost noiseless.

Significantly higher power consumption in transfer pumps is another disadvantage compared to submersible pumps. This power consumption does not typically come at the cost of efficiency, but more so that transfer pumps are used to pump liquids at higher rate, whereas submersible pumps are used for smaller-scale systems such as aquariums, where 20-30 gallons of water are being recirculated which requires much less power.

Despite transfer pump models being better equipped to pump fluid at a much faster rate and greater head height than submersible pumps, the hydroponic system in consideration for this project will only be pumping water about 5 feet vertically (from the reservoir on the ground to the top of the 5' towers), which suggests a high-power transfer pump may be too heavyweight for the purposes of the system.

While the transfer pump has decisive advantages over the submersible pump when it comes to durability and flow rate, the cost-benefit of each targeted requirement implicates the submersible pump as most fitting for a 5 foot tall hydroponic drip system. Adjustable flow rate dials more available on the submersible pumps will be crucial when it comes to fine-tuning the pump flow so that enough water is getting through the system efficiently without too much pressurization in the hose line. Using a 5-10 gallon reservoir means that to recirculate the entire water supply through the system 8 times an hour, enough to ensure nutrients are efficiently cycling to each plant tower, only 40-80 GPH at a max height of 6 feet will be necessary. This can be achieved in the $15-30 range for submersible pumps, but costs around $40-60 to meet these requirements with a transfer pump.

3.4.5 Selection of a Submersible Pump Product

The ideal submersible pump for this system will match all requirements specified in section 5.2. The minimum requirements for a pump, however, are ability to pump water through the drip system at a rate of 40-80 GPH and at a max height of at least 6 feet in an optimally efficient manner. The following sections breaks down and compare product specifications between 3 different submersible pumps made for aquariums and hydroponic systems. The optimal model of the options will be selected for the system with respect to its cost-benefit.

3.4.5a Vivosun® 660 GPH Submersible Pump

The first model under consideration differs in style from the usual hydroponic pump because of its round design created to prevent swift filter clogging. The impeller takes in water from the 360 degree base and forces it through a vertical connector for tubing located above the motor.

With dimensions small enough to fit inside of a 5 gallon bucket reservoir, a maximum flow rate of 660 GPH, and a max lift height of 10 feet, this pump meets the minimum necessary requirements for the system. However, this pump does not come with a dial to adjust the flow rate, which could have resulted in a flow

too high for circulation of 5 gallons of water-based nutrients. The pump also has a high power consumption (rated at 40 Watts).

3.4.5b Vivosun® 800 GPH Submersible Pump

The second model that was considered for the system was another Vivosun product with a different design as the pump in section 2.4.4.a, the main difference being that the impeller draws water from only one small filter to the side of the motor. Specifications for this pump are included in Table 17.

Small dimensions and adjustable high flow rate with low power consumption define this Vivosun submersible pump model. The pump exceeds the minimum necessary requirements with a supple flow rate and compact design that allows for convenient use. There is also an accessible dial on the pump filter that can adjust the flow rate, which can be imperative when looking for a proper pump model since there are too many variables in the system to know pre-construction how high of a flow rate the drip towers can withstand, so the dial allows for an easy method to turn down the flow rate to calibrate it to the system's needs.

3.4.5c CWKJTOP 220 GPH Fountain Pump

This pump has an almost identical look to the Vivosun model in section 2.4.5.b. The big distinction from the other pumps that the CWJK model has is the lower power rating at 15 Watts. This pump meets the minimum necessary requirements for the system, and offers them at a lower cost and smaller size than most other submersible models.

The main appeal of this pump model is the lower power rating than the first two in consideration. It is also cheaper and smaller, which will marginally decrease the system's overhead and weight. The main concern of this model is its max lift height. The pump will be moving the nutrient solution 5 feet from the ground up, which the CWJKTOP model specifies it is capable of, but it just narrowly meets that specification. The pumping rate at 220 GPH is adequate and the dial to adjust it would allow the pump to be calibrated to an appropriate rate for the system.

3.4.6 Pump Specifications

To select the proper pump for our system, the three recirculation pumps in consideration were compared by their specifications. The ideal model that fits the system's requirements with the lowest cost available was chosen for selection.

Table 13: Submersible Pump Product Comparisons

	Vivosun® 660 GPH	***Vivosun® 800 GPH***	**CWKJTOP® 220 GPH**
Dimensions	5.5” x 3.7” x 4.9”	4.1” x 2.6” x 3.5”	2.9” x 2.7” x 1.9”
Adjustable Flow Rate	No	Yes	Yes
Gallons per Hour (GPH)	660	800	220
Max lift Height (feet)	10’	10’	5.24’
Power Consumption	40 Watts	24 Watts	15 Watts
Price	$20 on Amazon.com	$26.00 on Amazon.com	$16.00 on Amazon.com
Voltage Requirements	120V/60HZ	110-120V/60HZ	110-120V/ 50-60HZ

3.4.7 Selection of the Vivosun® 800 GPH Submersible Pump

While the CWJKTOP and Vivosun 360 degree models have certain advantages over the Vivosun 800 GPH pump, such as lower cost and power consumption from the CWJKTOP and reduced risk of pump clogging from the 360 degree model, but the Vivosun 800 GPH model ensures that flow rates required to move enough of the nutrient solution to the top of the tank are exceeded and do not border on a total fault. The CWJKTOP model would be more ideal if it had a greater maximum head height, but the model is only rated to pump water 5.24’ vertically. Since the system is aiming to pump the nutrient solution 5’ from the ground to the top of the tank, the CWJKTOP model runs a great risk of not being able to move water reliably to the top of the drip towers. The Vivosun 800 GPH model, however, has a max head height of 10 feet. This ensures that the system has no issues pumping water 5 feet vertically at a decent flow rate.

The 360 degree Vivosun model may be better equipped to handle filter clogs due to the large intake radius for water, and it is just as capable as the Vivosun 800 GPH model in terms of head height and flow rate. The main reason that the 800 GPH model was chosen over the 360 degree model is the adjustability of the flow rate. The default flow rate of the round model is 660 GPH, which is substantial to move the solution to the top of the towers, but there is concern that the rate may

be overkill in the sense of consuming too much power. Rated at 40 Watts, the 360 degree model will be consuming more power than our entire system was aiming to consume collectively, and the absence of a flow rate adjuster like a dial prevents the ability to reduce this consumption. The Vivosun 800 GPH model does have this capability and a maximum power consumption of only 24 Watts, so it is well under the power consumption goal of 32 Watts that this project is aiming to stay under.

Taking the advantages of the Vivosun 800 GPH model over the other two models into account, this model was selected as the pump to move the water-based nutrient solution from the recirculation reservoir to the tops of the drip towers. Despite a slightly higher price than the other two models, the Vivosun 800 GPH model ensures the most reliable system operation and portability.

3.5 Grow lights

In order to support 4 columns of plants indoors we needed at least 30 watts of growth specific lighting. To achieve this, we planned on using energy efficient LED strips placed on each outer edge facing into the plants. These LEDs will provide T5 full spectrum Light Wavelength: 380 - 780 nm that is needed for proper plant growth.

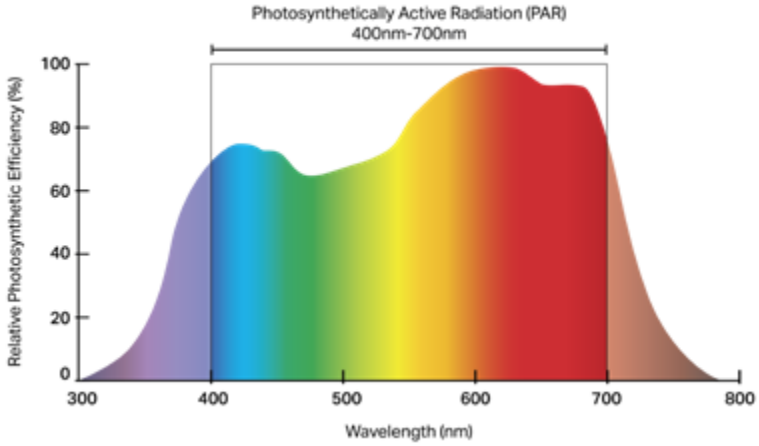


Figure 14: Growth Wavelength

According to the final vertical length of the system itself, x4 strips at about 15 inches long, containing 60 LEDs each were used to maintain uniform lighting across the grow area. By using LEDs, the power type and requirements were fairly simple, consisting of a 5v DC/2A power supply, commonly found on USB chargers hence we could use a step-down converter to convert a 12v rail down to a 5v rail. The lights were connected in parallel to prevent overall lighting failure if one strip goes down. This provides more redundancy within the lighting system and maintains a good overall health of the plants.

3.5.1 Grow light options

There are many types of indoor grow lighting types. Initially we considered incandescent bulb lighting via conventional light bulbs. This option turned out to be the least favorable type as it created large amounts of heat and the lighting provided was more towards the red side of the usable growth spectrum. Overall, our main goals in this area were:

- Provide the widest light temperature that covers most of the growth spectrum
- Provide enough energy for most types of plants (Eg ~30 Watts)
- Maintaining low heat emissions
- Maintain low energy consumption (Eg ~20Watt/unit)

Considering the above requirements, it reduces our options to a manageable amount. Thus on the table below we list the considered types of lighting alongside their pros and cons as well as their cost:

Table 14: Lighting Types Comparison

Type	Pros	Cons	Cost
Fluorescent	● Cost ● Large propagation area ● Longest lifespan	● Not useful during both vegetative & flowering stage growth ● Requires additional hardware	$35.00
HID	● Large dimmable range ● Large growth wavelength	● Cost ● Large heat emissions ● Requires additional hardware ● Short lifespan	$130.00
LED	● Cost ● Largest light/watt production ● Energy efficient ● Lowest heat emissions ● Ease of installation ● Large/customizable growth wavelength	● May not be strong enough during flowering stage	$27.00

By taking the pros and cons listed above into consideration, we confidently chose LED lighting as our growth lighting medium. Initially, LED had a large initial cost

due to its infancy in the growth department but cost has considerably dropped over time and continues to do so. Additionally, LED growth capabilities continue to grow as technology advances. Thus we have a wide variety of cost effective choices for our needed application. Coincidentally, one of the newest features of LED growth lighting is its customizable wavelength lighting. This feature was of great benefit to our system as it gave us the capability of supporting different types of plants, thus maintaining its usable flexibility. Considering all LED lighting types available, the table below depicts the best choice for our needed application:

Table 15: LED Light Comparison

Available at Amazon	Watt emission	Wavelength range	Power consumption	Cost/Unit
Seedling Plant Grow Lights Strip	30 Watts	380 - 780nm	15 Watts	$27.00
Mosthink LED Grow Light	20 Watts	460 - 630nm	10 Watts	$16.00
KingLED 1000w LED Grow Lights	600 Watts	380 - 780nm	185 Watts	$96.00
Lightimetunnel 4 pack	40 Watts	400 - 700nm	24 Watts	$36.00

We can conclude that "*Lightimetunnel 4 pack*" was the best choice for our application due to its price/cost performance.

3.6 Power management

Since we are dealing with an essentially “wet system”, great corrosion care and prevention of moisture must be taken into account to not only prevent downtime but extend the lifespan of the system. The first consideration for the power cables is gauge. Considering the rather low voltage and amperage the overall system will use, 16-gauge solid core cable similarly used in the automotive industry which closely resembles, or power demand can be used for our power supply system. For most logic driven cables an even thinner, 22-gauge cable can be efficiently used. Maintaining lower gauge, helps with cable pliability which in turn helps with design and power efficiency. Secondly, waterproofing connections are an ideal way to protect the metals from the humidity produced by the system. As a result, we initially planned to enclose the MCU and the controller boards within a waterproof box. Additionally, weatherproofing the connections through the use of butyl rubber on the connection options located on the box surface prevents humidity from entering the enclosure.

In the final implementation of the project, however, due to cost and time limitations the MCU and controller boards were not contained within a waterproof box. Though, given our final design, these components did not need to be given a waterproofing treatment as the Hydroponic Unit itself is water-tight, and the electronics were located away from any water source.

3.6.1 Main power supply Options

Due to the broad utilization of DC 12v amongst most of the devices used in our system, we considered using a MEISHILE B0781V1D7Q laboratory power supply that can adequately provide such voltage. Our main requirements for such per supply was to provide 12V DC and at least 50 Amps to give headroom for all the devices being used. Consequently, we used a 110v/220v ±15% AC to 12V ±5% 50 Amp DC power supply rated at 80% efficiency with x3 V+ and x3 V- outputs. This was not only sufficient, but it also gave us headroom in case we would like to add on more devices in the future. Considering its limiting number of outputs, a distribution block could have been added to effectively multiply the output number thus accommodating the number of devices used. Additionally, 12v DC to 5v DC reducer boards were able to be used if needed.

Another viable option was to repurpose a computer power supply. Considering that computers need constant accurate power to drive it’s sensitive components, their Power Supply Units (PSU) are made to provide plenty of accurate and stable power. They are made to be versatile due to the modular nature of most personal computers nowadays. Additionally, they provide a wide range of commonly used voltage rails ranging from 3.3V to 12V DC. Lastly, due to the popularity of personal computers, these devices can be found in a wide variety of form factors, configurations and are readily available at lower prices in general. Due to our device requirements we are considering using a power supply that provides at least 30 Amps of current within its main voltage rails. One power

supply we were considering was the MEISHILE that provides a wide variety of voltage rails that suffice our requirements.

Table 16: Power Supply Comparison

	XION AXP-850K14XE 850W	MEISHILE B0781V1D7Q
Voltage rails	+3.3@28A +5V@30A +12V1@20A +12V2@20A +12V3@30A +12V4@30A -12V@0.3A +5VSB@3.0A	+12V1@30A +12V2@30A +12V3@30A
Total Power	850W	360W
Efficiency	>82%	>80%
Size	150*86*140mm	215*115*50mm
Certifications	CUL, FCC, TUV, CE, CB	FCC, CE
Active Cooling	Yes	Yes
Switchable	Yes	Yes
Price	$30.00	$41.00

3.6.2 Power connection switching

Considering that our MCU is controlling most mechanical devices through switching, a viable option for our system was to use a relay module that can handle upwards of AC250V 10A and DC30V 10A currents via 5v DC signal. Considering that a relay is a mechanical switch, applying this module gives the MCU the capability to safely switch on or off devices that run on high voltages and currents. The number of relays to be used is established through the planning of the entire system as devices might be added or removed accordingly; planned 3. Similarly, we used transistor triggered switching for devices requiring lower current to maintain overall efficiency. In any case a relay can be used on the entire system if needed. In addition to switching these modules provide LED status lights that can help troubleshoot possible burnouts during use.

3.6.3 Step-up DC-AC Power Inverters

During our parts research we have noticed that some components require 120V 60Hz AC voltage. This proved to be a challenge as our main power supply only provides 12V and 5V rails as source. Thus to overcome this we used a DC-AC power inverter commonly marketed as power interverters for vehicles. This device helps not only convert the voltage from 12V DC to 120V 60Hz AC but as seen, it also regulates and step-up the voltage depending on the need of our components. Such devices are readily available at prices ranging from $20.00 to $120.00 and can be implemented with ease. In specific, we used:

The Ridgid 100 Watts Power Inverter

- Continuous 100W
- Input: 12-13.8V DC
- Output: 120V 60Hz AC, 0.83 A

This specific power inverter provides enough power to our AC voltage driven mixing pump as tested. Due to possibly changing specifications, we had other alternatives in mind that could better help our system design:

Table 17: Power Inverter Comparison

	Input	Output	Outlets	Remote	Price
Ridgid 100W	12-13.8V DC	120V/0.83A AC	1	No	$35.58
Cobra CPI 500W	10-15V DC	115V/4.3A AC	2	Yes	$49.95
Ampeak 100W	11-15V DC	120C/3.1A AC	2	No	$39.99

3.6.4 Relay switching

Due to the variety of voltages used in the main system, we needed an electronic means to control the switching of these devices through the MCU. Considering that the IO pins of the MCU can handle up to 3.3V DC, our relay also needed to share such a requirement within it's communication rail. Additionally, the relay would have to have supported a variety of load voltages ranging from 5V/1A DC to 250V/10A AC. For this, our best choice would be to use an arduino oriented relay as the Arduino board has similar power requirements compared to our MCU. Thus we decided to use the - DC 3V Relay High Level Driver Module Optocoupler Relay Module Isolated Drive Control Board with the following specifications:

- Detection input: 0 - 25V DC
- Work load: 3 - 3.3V
- Control load: 30V DC/10A & 250V/10A AC
- Driver Chip: EL817, SRD-DC03V-SL-C
- Isolated
- Diode protected

This specific device is highly beneficial to our final design as it can handle the exact required parameters in addition to being both isolated and efficient. Overall this is used to control our different types of pumps, the lights, and some sensors.

3.8 LCD Screen

The Liquid Crystal Display (LCD) screen is a way for us to monitor the status of our sensors and display them to the user. The LCD also helps us debug the operations of the hydroponic system when properly configured. The following section is information about LCDs and the factors we considered when choosing an LCD screen.

3.8.1 About LCD Screens

LCDs are flat-panel displays that use liquid crystals, polarizers, and a backlight to display an image onto a screen. The LCD we chose for this project is small in size and can be used to display characters for the user to read.

The LCD screen should have enough space to display all the desired information. This includes the pH sensor, which includes an integer from 0 to 14, and an empty indicator for the two water tanks. Optionally, we could display other information such as whether the mixer, water pump, lights, and the individual nutrient dosers are on. This means we need five characters for displaying the pH and five to seven characters for each water tank with the option of five to seven for the water pump, five to seven for the water pump, six to eight for lights, and each of the three dosers could be four to eight characters. To meet our specifications, we needed a display with ten to sixty-eight characters.

The pin requirements the LCD needs to function is the next attribute for consideration. The CC3220 MCU we are using only has 30 GPIO pins. Some interfaces can use up to eleven pins, which could put us in a tough spot if we were looking at expanding our project with additional devices. Depending on the interface type, we could get as low as 3 pins for the LCD to function. To send a character to the LCD, there needs to be a pin(s) reserved for the character as well as two or three pins to synchronize the display and the timing of data being sent with the enable pin, read/write pin, and reset pin.

Power requirements are next for consideration. If the voltage required for the pins to register as active is higher than the microcontroller's output, we need a way to step up the voltage which would complicate the design. Additionally, the current draw determines how much power the LCD needs to function and since this device remains active at all times, we want to minimize this as much as possible.

Some other attributes to consider are operating tolerances. These include, high and low temperatures, humidity, shock, vibration and static electricity. Of these, humidity is one we'll have to keep in mind in testing because our water tanks are open.

There are other less important attributes for this project. For example, the time it takes to send a character and have it displayed on the screen. I feel this is not important because this LCD screen is just for monitoring sensor data once every five minutes.

3.8.2 Choosing an Appropriate LCD

After doing some research, I chose four contenders to discuss in this report. They are the NHD-C0220BIZ-FS(RGB)-FBW-3VM, NHD-C0216CZ-NSW-BBW-3V3, NHD-0420AZ-FSW-GBW-33V3, and NHD-0420D3Z-FL-GBW-V3. All of these displays are from NewHaven Display International. These displays came up in the parts search because it meets the needs of our supply voltage or came up at a reasonable price. The attributes are as follows:

Table 18: LCD Comparisons

	NHD-C0220 BIZ-FS(RGB)-FBW-3VM	NHD-C0216 CZ-NSW-BB W-3V3	NHD-0420AZ -FSW-GBW-3 3V3	NHD-0420D3Z-FL -GBW-V3
Display Format	20 x 2 Characters	16 x 2 Characters	20 x 4 Characters	20 x 4 Characters
Backlight	LED – RGB	LED - White	LED – White	LED - Yellow/Green
Supply-Cur rent	0.2mA – 1.5mA	0.2mA – 1.5mA	0.5mA – 2.5mA	N/A
Backlight Supply Voltage	2.7V – 3.1V	2.7V – 3.1V	2.8V – 3.2V	4.7V - 5.5V

Backlight Supply Current	30mA – 40mA	30mA – 40mA	10mA – 30mA	21mA - 44mA
Interface	I²C	Serial	Parallel	I²C, RS-232, SPI
Number of Characters	40	32	80	80
Price	$12.89	$11.52	$20.48	$22.45

Of these options, the NHD-0420D3Z-FL-GBW-V3 was a great option for our project. It may be more expensive, but the 1²C interface would keep our pin requirements low at three to five pins. It has a sufficient amount of character space, but we need a way to step up the voltage to 5V. The NHD-0420AZ has a parallel interface which uses eleven pins and is not much less expensive. The upside to using the NHD-0420AZ would be the lower current draw with the same character space. The NHD-C0216CZ has a serial interface with five pins. It is cheaper and has similar power requirements, but thirty-two characters is going to be on the edge of our minimum requirements and we want to make sure we display all the necessary information. This option also requires the least amount of external components, which reduces the number of points of failure, so it was the easiest to work with.

3.8.3 LCD Implementation

The NHD-04203DZ-FL-GBW-V3 can use I²C, RS-232, SPI interfaces for data transfer. To use I²C, a jumper must be placed on R1 on the back of the LCD. I²C uses a maximum clock of 50 KHz. To use SPI, a jumper must be placed on R2. SPI uses a maximum clock of 100 KHz. Timing this device is easy because it has a PIC16F690 microcontroller board that handles serial communication. All that is required is a 100 millisecond delay when powered on to initialize the display. The datasheet contains a list of commands to operate the LCD such as clearing the screen, writing and reading data from the device's internal memory, moving the cursor, etc. There is also a table of fonts on page 14 of the data sheet, which we required to process characters and display them to the LCD screen [4].

3.8.3a Software Setup and Usage for I²C

The default slave address for this I²C device is 0x50. It can be adjusted to address 0x00 by sending the command 0x00 0x62. The address must be an even number and the change takes 20 milliseconds to take effect. If SPI or RS-232 is selected, the default address is restored.

3.9 Water Level Sensor

To effectively monitor water-based nutrient quantity inside of the recirculation reservoir, a sensor that is able to detect if the solution is below a minimum level is required to alert the user when to refill the system back to its fill line. Either an analog or a digital sensor would suffice for the purposes of the project since the only requirement of this sensor is that it can tell when the water is below the minimum fill line, which happens when the tank is only at 60% capacity; however, a digital sensor was chosen to save costs and simplify this component's application.

The ideal water level sensor selected for the system should be able to meet the following requirements and constraints:

- The sensor shall be able to reliably detect the presence and absence of water at a given level
- The sensor should require no more than one GPIO pin for reading values
- The sensor should be submersible for long periods of time
- The sensor shall be sized appropriately to fit in the nutrient solution reservoir while taking up negligible room

3.9.1 Analog or Digital Sensor

When a hydroponic system that relies on a recirculation reservoir begins to get low in fluid level, it only needs to be topped off every couple of days or so due to a combination of the solution being consumed by the plants and evaporation. Despite seeming like a straightforward issue needing a simple solution, Water level deterioration rates and thus need to refill the reservoir can depend on a variety of system conditions: the humidity in the room where the system is located, the ages of the plants in the system and how much water they consume daily, and water spillage in the system.

While an analog sensor could be converted to a digital input that can be used to monitor water level at a certain percentage, a digital sensor would be ideal to meet the requirements for this project. Since the water-based nutrient solution will need to be topped off with fresh water at highly infrequent intervals, a sensor that can send one signal if water is below a minimum fill line to alert the user when to refill the reservoir is the simplest and most cost-effective way to ensure the user does not allow the system to run dry, causing severe pump damage and the plants in the drip towers to die.

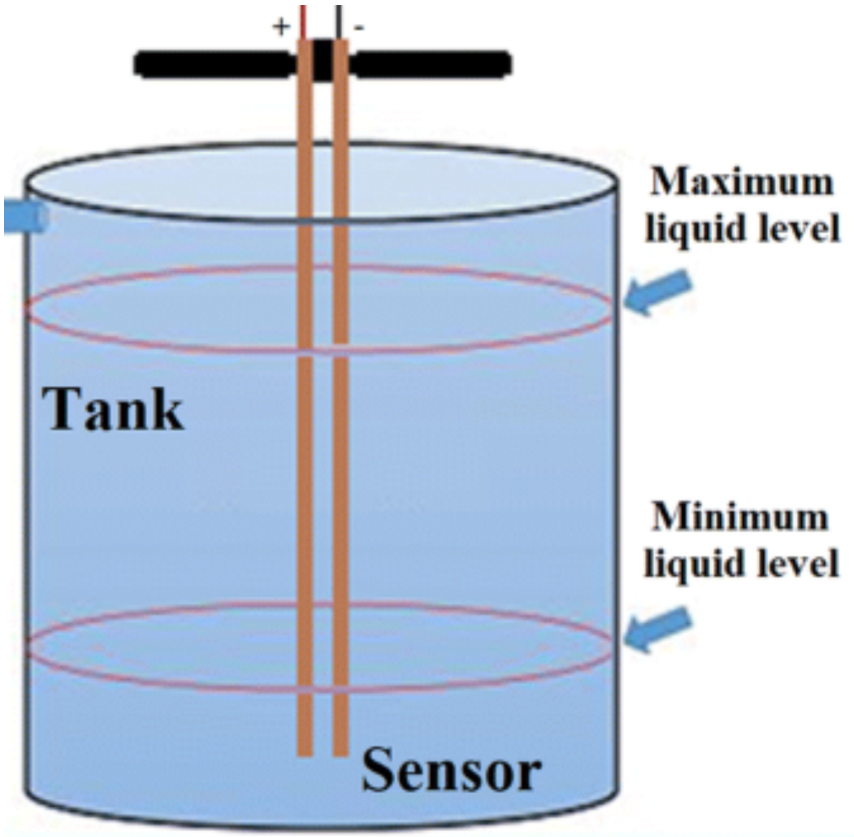
3.9.2 Selecting a Digital Fluid Level Sensor Type

There are a few different types of digital fluid level sensors that can be used to tell when liquid is present in a container or not: capacitive, float, and optical liquid level sensors. Each of these sensors has advantages and disadvantages, and the simplest, most cost-efficient sensor was chosen to meet the requirements and fit within the constraints outlined in the previous sections.

3.9.2a Capacitive Liquid Level Sensors

Capacitive sensors typically use 2 electrodes with a small distance between them. When these electrodes are submerged in fluid, a small current is able to run through the sensor and the capacitor in the sensor reads a high capacitance; however, if they are not submerged, then it creates a short circuit and a low capacitance is read. These sensors are designed for analog inputs, as most of them have long metal electrodes that, depending on how much of the electrodes are submerged, charges the capacitor to different levels. This means that it is possible for the capacitive sensor to be calibrated to read fluid levels on a percentage-basis (0% - 100% capacity). While this could be useful, it was in the project's best interest to keep the fluid level readings straightforward and digital. A diagram of a typical capacitive sensor is included in Figure 22.

Figure 15: Capacitive Liquid Level Sensor Functionality

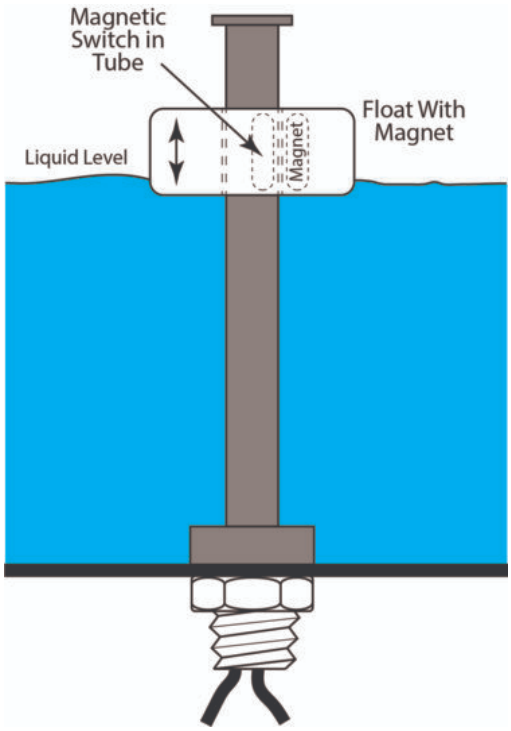


3.9.2b Float Liquid Level Sensors

This model of sensor centers around the idea of having a magnet encased in a buoyant material attached to a rod that will float when sitting on top of water, and when the magnet is sitting at a the top of the rod, a circuit is closed inside of the sensor and it reads high voltage as an output. This sensor model is simple to use and cost-effective, with great options for less than $10.00. A detriment to this type of liquid level sensor is the dimensional overhead; the 5 gallon bucket being

used as the system's recirculation reservoir will already have a pump, pH sensor, and a mixing fan inside of it, and it is important not to overcrowd the system with more moving parts. With the water moving around so much through pump recirculation and the mixing fan, it had a chance of causing unforeseen problems when attempting to make accurate measurements if the magnet was moving up and down with the fluid level. However, this model would have provided for a cost-effective way to make liquid level readings.

Figure 16: Liquid Level Float Sensor Functionality [5]

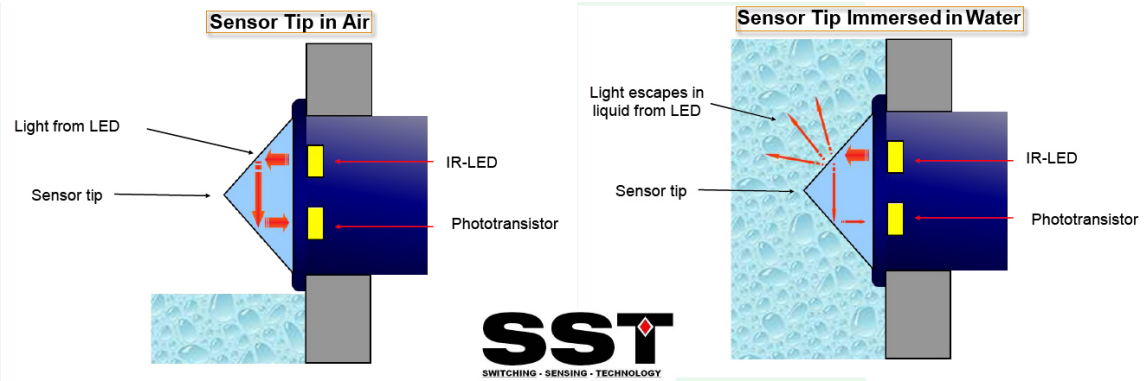


3.9.2c Optical Liquid Level Sensors

The final model of liquid level sensor that was considered for the system was optical sensors. An infrared LED and a phototransistor are optically aligned when the sensor is only exposed to air, but when the sensor is immersed in water, the infrared LED's rays are scattered and no longer align with the phototransistor. When the rays do align with the phototransistor, the sensor outputs a high voltage, so a signal can be interpreted appropriately to know if the sensor is submerged in liquid or not. This sensor is slightly more expensive than the previous models mentioned ($15-$25), but offers a compact design around the size of a bottle cap. The sensor is reliable and has a highly desired capability of sending a digital signal that does not need to be converted from an analog input, since the phototransistor is either receiving the infrared light rays or it is not. The setup allows for a swift and simple calibration, and small dimensions of the typical optical sensor make it easy to cut down on the volume of sensors being

placed in the small reservoir. A diagram of how optical sensors function in air versus submerged in liquid is included in Figure 24 [6].

Figure 17: Optical Sensors' Behavior in Air vs. Water



3.9.3 Selection of the Optical Liquid Level Sensor

Despite the slightly more inexpensive costs of the float and capacitive liquid level sensors, the optical sensor outweighs both of them in cost-benefit. Since a goal of the system was to create a sort of "wall decoration" out of the hydroponic towers, it was crucial to cut down on the amount of moving parts that could break or need to be cleaned regularly. The float sensor offered one of the cheaper methods to monitor fluid level, but there was concern about the reliability of the product in such a small system. While the capacitive form of liquid level sensing has a wide variety of styles and sizes, most of them output an analog output that would need to be converted to a digital reading, which leaves room for error when maintaining the system as a whole because of the possibility to make an incorrect reading.

These issues with the float and capacitive models pave way for the optical sensor's benefits to outshine them. Despite being limited in companies that have this type of sensor available, the models offered are revered as highly accurate and low maintenance. The $25.00 price tag on the more expensive models is significantly more than the other two models being considered, but it hardly puts a dent in the budget of $1000.00. Since the optical sensor was fiscally possible, it was the clear front-runner for a liquid level sensor in the recirculation reservoir. The tight design allows for it to be drilled into the side of the 5 gallon reservoir at the minimum fill line, so it is out of the way of the pump and mixing fan to get accurate liquid-presence readings. The digital input of the sensor also eliminated the need for more coding that would be needed to convert analog inputs to digital, since the sensor reads the liquid as present or not present.

3.9.4 Selecting an Optical Liquid Level Sensor

Since optical liquid level sensor technology is relatively new, only a couple companies offer a model, so the options for one are limited. In the interest of cost-efficiency, the cheapest model was selected for the system considering the effectiveness of the sensor at detecting liquid. While the company SST Sensing offers a large variety of models of optical liquid level sensors, they offer a basic model for a price that is half of the cost of their more advanced models that fits the needs of detecting the presence of water-based nutrient solution in our reservoir. Due to this, only the basic model was considered for the purposes of the hydroponic system, since the lightweight sensor is durable, accurate, and helps keep costs in the system low.

3.9.5 Selection of the Optomax LLC200D3SH

The basic model selected from SST Sensing is a standard digital sensor that can accurately detect the presence of liquid. It was chosen over all other models because of its low cost and straightforward application. The sensor met all requirements because it is digital, low cost, and highly durable due to its plastic design. Specifications for the sensor are included in Table 24. An image of the sensor is shown in Figure 25.

Table 19: Optomax LLC200D3SH Liquid Sensor Technical Specifications [7]

Supply Voltage (V_s)	$4.5V_{DC}$ to $15.4V_{DC}$ OR $4.5V_{DC}$ to $5.5V_{DC}$ (PWM output)
Supply Current (I_s)	2.5mA max. ($V_s = 15.4V_{DC}$)
Source Current (I_{out})	100mA
Operating temperatures	Standard: -25℃ to +80℃ Extended: -40℃ to +125℃
Storage Temperatures	Standard: -30℃ to +85℃ Extended: -40℃ to +125℃
Housing material	Polysulfone or Trogamid®
Sensor termination	24AWG, 250mm PTFE wires, 8mm tinned
Output Voltage (V_{out})	Output High: $V_{out} = V_s$ - 1.5V max Output Low: V_{out} = 0V + 0.5V max
Pulse Width Modulation	Duty cycle in air: 25% ± 10% Duty cycle in liquid: 75% ± 10% Frequency: 2kHz ± 10%

Power consumption (P)	P_{max} = 38.5mW
Cost	$25.00

3.10 Liquid pH sensor

The pH of a hydroponic nutrient solution is the essential component to ensuring liberal nutrient intake of the plants contained in the system. If a solution is overly alkaline or acidic, the plants have a difficult time absorbing the nutrients provided in the water-based solution. This is why a cornerstone of the hydroponic system needed to be accurate and have rapid pH adjustments, which required a reliable sensor that can relay information to the database efficiently and accurately.

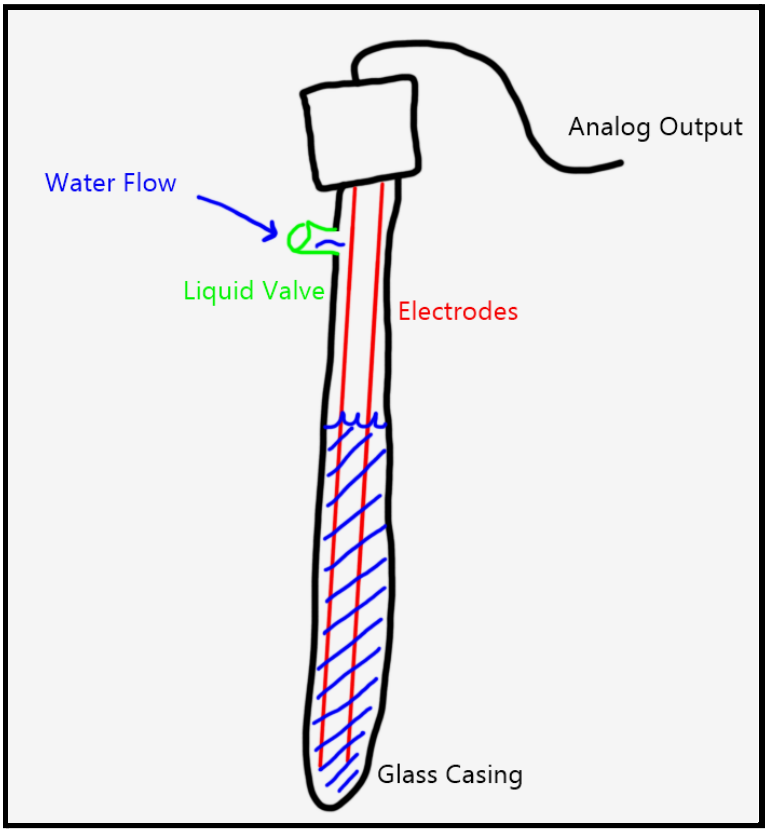
3.10.1 Types of combination liquid pH sensors

An ideal pH sensor is one that is accurate and durable. Since our system revolves around the idea of using it in a confined space as a wall decoration on the consumer-end, a sensor that can withstand long amounts of time without needing maintenance or replacement is a large benefit to the overall cost-efficiency of the hydroponic system. Since pH is read on a scale from [0, 14], 0 being the most acidic and 14 being the most alkaline, a sensor that reads in inputs as analog is required.

3.10.1a Combination Sensors

Combination sensors are a two pronged electrode module that measure the electrical signal between them when submerged underwater. Since a change in pH is a change can be measured by the change in electrical conductivity of the water (i.e. the number of ions in between the electrodes), this type of sensor can be calibrated to translate an electrical signal from the probes into an accurate pH reading of the solution they are submerged in. Figure 26 shows a diagram of the standard combination sensor. The device is two space-efficient small electrodes encased in glass, one a reference electrode and the other a polling electrode. Liquid fills the container when submerged through its liquid valve at the top of the casing. The electrical signal received through the electrodes is sent through the wire and can be read as analog data.

Figure 18: Combination Sensor Diagram



3.10.1b Differential pH Sensors

Similarly to the combinational pH sensor mentioned in the previous section, the differential sensor uses electrodes that read a voltage between them to send an analog signal through the wire. The main difference in differential pH sensing is the presence of a third electrode connected to ground. This electrode elevates the pH sensing process to be more heavy duty, because it helps with an event known as “reference fouling”, which is when the heat of a liquid between the two electrodes or the presence of a foreign substance causes the pH reading to get scrambled and be inaccurate. The third electrode mostly negates this phenomenon because it acts as a buffer between the two combinational electrodes.

3.10.1c Laboratory pH sensors

Laboratory pH sensors also use combination pH sensor electrode technology, but are encased in a thin 12mm glass and are a thin body style coated in plastic

material. These sensors are much more lightweight than any of the other styles being considered. They are typically used for testing water pH infrequently, but offer results for a much more inexpensive price than a lot of the other models. The thin design is the primary reason for this model was considered for this project, because the system's recirculation reservoir is a 5 gallon bucket with a 12" diameter; with a pump, water level sensor, and pH sensor going into the bucket, it was important to try to minimize the space taken up by the sensing technology in the reservoir.

3.10.1d Industrial pH sensors

Similarly to Laboratory pH sensors, Industrial pH sensors use electrodes encased in glass to measure the voltage between them that can change depending on the amount of ions in the liquid surrounding them. The biggest advantages of the industrial sensor over the laboratory sensor is durability. These sensors typically come with a small reference electrode encased in a polytetrafluoroethylene membrane surrounded by a protective metal casing. Industrial models are also typically made of stronger materials like carbon fiber or rust-proof metal composite. This upgrade in electrode encasing prevents clogging much better than the laboratory pH sensors since liquid doesn't have to physically get between two electrodes to get an accurate pH measurement from it. Figure 27 shows a reference electrode on an industrial pH sensor encased in the polytetrafluoroethylene membrane.

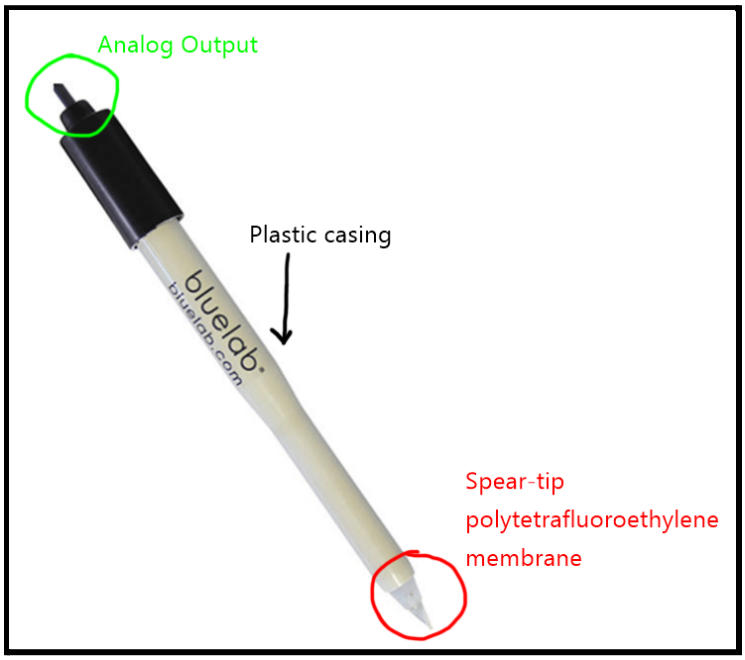
Figure 19: Industrial pH Probe [8]

3.10.1e Spear Tip pH Meter

Spear tip pH meters are a different classification of industrial pH sensors, the difference being that instead of a heavyweight body around the combinational electrodes, it has a sleek design with a probe encased in a polytetrafluoroethylene membrane shaped like a pencil. Typically this sensor is

used to read the pH of semisolid material such as soil, fruit, and other plant matter, but the probe comes into consideration for this system because of its high durability and slim figure. As mentioned, it was important to cut down on the space taken up by the pH sensor in the recirculation reservoir, so that the amount of liquid in the reservoir could be maximized. Figure 28 shows an example of the slim body and spear tip of the pH meter:

Figure 20: Spear Tip pH Meter from Bluelab™



3.10.2 Requirements of System pH Sensor

The factors that were considered when selecting a pH probe for the purposes of a hydroponic system with a recirculation reservoir were prioritized as follows:

1. *Durability*
2. *Accuracy*
3. *Price*
4. *Size*
5. *Compatibility*

The system needs frequent pH monitoring (every hour), because keeping the acidity balanced in the water-based nutrient solution optimizes nutrient intake by any plants receiving water from the system. This in addition to the system's aim to be a "wall decoration" for small spaces such as apartments with users not needing to frequently monitor and clean their system called for a sensor that was durable enough to withstand long amounts of time submerged in liquid. With available pH sensing technology, there aren't many sensors within a price range of this project that can stay submerged indefinitely, but there are options such as

the industrial sensors that have durable cases and electrode probes that do not get clogged easily. The laboratory sensors are the most cost-efficient way to measure pH of a liquid, but these sensors just don't have the capabilities to stay submerged underwater for 2-3 weeks at a time until the user replaces water in the system. The plastic material encasing the electrodes can be prone to bacterial buildup, and the two-pronged electrodes can get clogged with sediments easily, which undoubtedly causes reference fouling when trying to make an accurate pH measurement.

Accuracy in pH readings is a given with most available pH sensors today, but accuracy over time is a large factor to consider in selection of a sensor, because even short-term submersion underwater can begin to alter any output read from electrodes due to sediment buildup mentioned previously. That being said, a meter that is rated to make accurate measurements (preferably <2% error) and is not susceptible to easy electrode clogging will keep accurate measurements going through the system.

Compatibility is a highly necessary requirement for a pH sensor. A large amount of commercially available pH sensors are converted to analog output through a BNC adapter and a signal conversion board. The consequence of this is the sensors are sometimes only compatible with Arduino and Raspberry Pi controller units, so a sensor that is compatible with the TI CC3220 is required or the system would not be able to make accurate pH measurements if it were to be able to make any at all.

3.10.3 Industrial or Laboratory Sensor

Table 25 shows some comparisons between two different affordable DFRobot® sensors. One is their cheaper laboratory model while the more expensive one is their industrial model. One of these models is narrowed down as the final selection.

Table 20: Industrial and Laboratory pH Sensor Comparison [9]

	SEN0169-V2 Industrial Sensor	SEN0161-V2 Laboratory Sensor
Probe Life	**24 hours a day 7 days a week for >0.5 years**	**>0.5 years depending on frequency of use**
Cost	**$64.90**	**$39.50**
Supply Voltage	**3.3~5.5V**	**3.3~5.5V**
Dimension	**1.70” x 1.26”**	**1.66” x 1.26”**
Temperature Range	**0~60**℃	**5~60**℃
Cable Length	**500cm**	**100cm**

3.10.4 Selection of the Industrial pH Sensor

Despite a bulkier outer casing of the electrical probes compared to the much thinner spear-tip and laboratory probes, the Industrial pH category of sensors was selected for use in the recirculation reservoir. Differential pH sensors would be a way to get much more accurate pH measurements because of the third reference electrode, but these models can cost hundreds of dollars minimum, which is outside of the project’s budget. A laboratory sensor was highly desired due to its affordability and accuracy, but there was concern that it was not heavy-duty enough to withstand long amounts of time submerged underwater. While it would be up to the user to clean their sensors when need-be, it was pivotal to extend the period between necessary maintenance times as much as possible to optimize the “automated” feeling of the system. This pointed the needs of the project in the direction of the Industrial Sensor, because it offers the same accuracy as the other sensors with highly extended durability, and the spherical polytetrafluoroethylene membrane encasing the electrodes prevents sediment clogging and therefore reference fouling, while making bacterial buildup easy to clean off.

3.10.5 Selecting an Industrial pH sensor

Since an Industrial pH sensor is classified as a much more durable laboratory sensor with combinational electrodes for pH sensing and a

polytetrafluoroethylene membrane encasing them, most upgraded laboratory sensors meet the requirements of the system. DFRobot® offers two versions of their industrial pH sensor, their original model and an upgraded version. The upgraded version added capabilities to be used with an external ADC module instead of being constrained to only using Arduino libraries, so the DFRobot Industrial V2 probe is the only selection of the two options that work with the system.

3.11 Framing Support

A good support material that serves as the main support structure for the system would have to resist weight, humidity, and mechanical vibrations. In turn out top two choices were treated wood or PVC, below we can see the advantages and disadvantages from either material.

Table 21: Wood Vs. PVC

	Advantages	Disadvantages
Wood	• Cost efficient • Availability • Easy of fabrication • Thermal properties • Acoustic response	• Weaker material • Humidity susceptibility • More flammable • Weaker Sturdiness
PVC	• Sturdiness • Waterproof • Fire resistance • Acoustic response	• Costly • Tougher fabrication • Availability • Thermally inefficient

Due to these attributes, we decided to go with PVC as it gave us the needed flexibility while being largely available. Additionally, we took acoustic response into consideration due to having mechanical moving parts driving within the system. These parts such as electric motors, relays and solenoids could be a cause of excessive noise. In turn, we strove to maintain the noise pollution down to below 36 dB which is the normal talking volume. We have considered using a mix of the two materials in parts where their strengths can be more beneficial than others. Regarding the control box that houses the electronics such as the MCU, it was possible for it to be made out metal by using a repurposed electrical junction box and fitted with proper weather tight through-hole connections nodes. The table below depicts the proposed material for each section of the overall system:

Table 22: Support Materials

Section	Material	Cost/Unit
Main tank support structure	PVC (2”X 6’)	$8.20
Dosing pump supports	Expanded PVC (¼”)	$22.00
Window	Acrylic (18”x24”)	$14.40
MCU enclosure	Metal	$20.00
Light supports	Expanded PVC(¼”)	$22.00
Main power supply unit	Metal	$20.00
Plant housing	PVC (3”x10”)	$13.00
Mixture tanks	PVC (4”x10”)	$13.00

3.12 Pump Line

To mainline the water-based nutrient solution from the recirculation reservoir to the tops of the individual drip towers, some sort of tubing is necessary that can attach firmly to the 1/2” pump nozzle. As mentioned in the requirements, it is necessary to have opaque tube lines to prevent bacterial buildup that happens when water is exposed to too much light. The tubing must also be malleable, because it needs to be able to make gradual turns to distribute water in the system.

3.12.1 Vinyl Tubing

Since any vinyl tubing fits the necessary requirements, the flexible and inexpensive material was selected to use as the main pump line. The required diameter for the line is ½” to fit around the spout of the Vivosun® 800 GPH pump.

3.12.2 Tube Fitting and Splitting

Since the system has 3 separate drip towers, there needed to be a way for the main line to be split the same amount of ways with equal water distribution. The Vivosun® 800 GPH has no issue with water pressure getting to each drip tower since it far exceeds the flow rate requirement of 120 GPH for the system, so only three barbed ½" tee joints were required to split the main line down to each tower. Commercially available barbed Tee Joints do not differ too much in price, so a cost effective model was chosen appropriately. These tee joints only cost $0.87 at local hardware stores, so buying 3 of them amounts to a negligible cost in the total bill of materials.

To fit each T-Joint to separate sections of tubing, the vinyl tube had to be fitted and secured using adjustable ½" pipe clamps. These are standardized as stainless steel clamps with a flathead screw in them used to tighten or loosen them. These clamps are also highly inexpensive. Only 10 are necessary to secure each tubing end to its respective tee, and a bulk pack of 10 is available at local hardware stores for only around $7.00.

3.13 Growing Medium

The plants in the system need a medium for their roots to take hold before their roots begin to expand throughout the drip towers. The standard for hydroponic systems and a favorite medium of most growers is rockwool. Rockwool is a combination of basalt and volcanic slag that is spun into a fibrous material. It is known for its use in insulating homes, but the rockwool that is used in hydroponics serves as a base for the roots to take hold [10]. Its rough, sturdy, yet fibrous body makes it the perfect selection for any hydroponic system. Rockwool also has great water retention abilities, so young sprouts don't have to worry about lack of moisture in their adolescent and fragile state.

The rockwool cannot hold plants on its own, however, and it needs a cup-like structure to support it inside of the PVC wye. Mesh plastic cups are another hydroponic standard, because they are affordable and offer effective stability for a plant's root ball, while giving it enough oxygen and space for its roots to branch out into the hydroponic tower. 20-30 generic mesh plastic cups are used for this system from a local hydroponic store, and they only cost around $0.20 per cup, which is another negligible price added to the bill of materials.

3.14 Mixing Fan

The mixing fan located at the bottom of the reservoir serves a great purpose for this project, because it is responsible for equal distribution of nutrients and stimulating pH readings. The ideal mixing fan is small in size and low in power consumption, because nothing heavy-duty is needed to give a mild stir to 5 gallons of water; however, it was also desirable to get a mixing fan with a

negligible cost that would be able to withstand long periods of time submerged in water.

3.14.1 Mixing Fan Control System

To control the mixing fan, the same relay system used for the submersible recirculation and dosing pumps is used to switch it on and off. A signal is sent to a GPIO pin that activates the relay which in turn allows current to flow to the mixing fan. This makes it simple to turn the fan on or off without risk of connecting a high-power device to a microcontroller that could be harmed by any short-circuitry.

3.14.2 SunSun JVP-101A or FFXTW Wavemaker

Most inexpensive submersible mixing fans have standardized power consumption and sizes, so most models do not differ much from one another, but two different fans, the SunSun JVP-101A and the FFXTW Wavemaker both were considered for use in the project. Table 28 shows some specification comparisons of these two products. Not included in the table are specifications which the two fans both share, including power consumption at 6 Watts each and water flow at 800 GPH at full power.

Table 23: Submersible Mixing Fan Comparisons

	SunSun JVP-101A	FFXTW Wavemaker
Dimensions	6” x 4” x 4”	6.41” x 4.64” x 4.33”
Cost	$10.99 on Amazon	$14.99 on Amazon
Weight	9.6 Ounces	11.53 Ounces

3.14.3 Selection of the SunSun JVP-101A

Since there are not many detrimental differences between the two mixing fans in consideration, the cheapest and smallest option was selected, which was the SunSun JVP-101A model. This model provides exactly what is needed for the hydroponic system for a negligible cost of only $10.99. This model is also slightly lighter and smaller than the FFXTW Wavemaker, which isn’t a huge determining factor in selection, but it is better for the system to minimize external part weight and size in the reservoir so that the maximum amount of fluid can fit.

3.15 Nutrient Solution and pH Adjustment Fluids

The fluids contained in the dosing pumps are vital to proper plant upkeep. pH Adjustment fluids are standardized as “pH up/down” and just contain acidic or basic compounds that, with small doses, can drastically shove the pH of a liquid in a certain direction. Standardized pH up/down fluid for the system is provided by Atlas Scientific™. The nutrient solution on the other hand can be swapped out depending on the type of plants being grown in the system, but since the system is demonstrated with spinach, a nutrient that is high in Nitrogen and Phosphorous (which promotes vegetation) [11] is ideal. Dyna-Grow’s 7-9-5 solution, available at https://www.amazon.com/Dyna-Gro-Gro-008-7-9-5-Plant-8-Ounce/dp/B0001XGPIM/ref=pd_lpo_1?pd_rd_i=B0001XGPIM&psc=1 is the ideal solution for all-around plant growth and vegetation, and is available for a cost-effective price of only $20.00 per Quart. Dyna-Grow recommends 2 tsps of nutrients per gallon of water, so with full nutrient replacement every 2 weeks that means the user only needs new nutrients every 24 weeks, making this highly cost-effective.

4.0 Design Constraints and Standards

Before starting any project, it's important to understand limiting factors when making design decisions. These factors are the standards to conform to for implementation and other external factors called design constraints.

4.1 Standards

Standards help engineers conform common situations with a specific implementation. Doing this often helps build a safe solution everyone can use. For example, the IEC 60035-1 standard covered later specifies safety electrical guidelines for building a household appliance. Ignoring standards can lead to unorganized implementations and that is something to be avoided.

Standards for building software are a way of conforming software development styles into a single standard. Conforming to a single style helps other programmers use and maintain the software. Often these include naming conventions, spacing, and file structure.

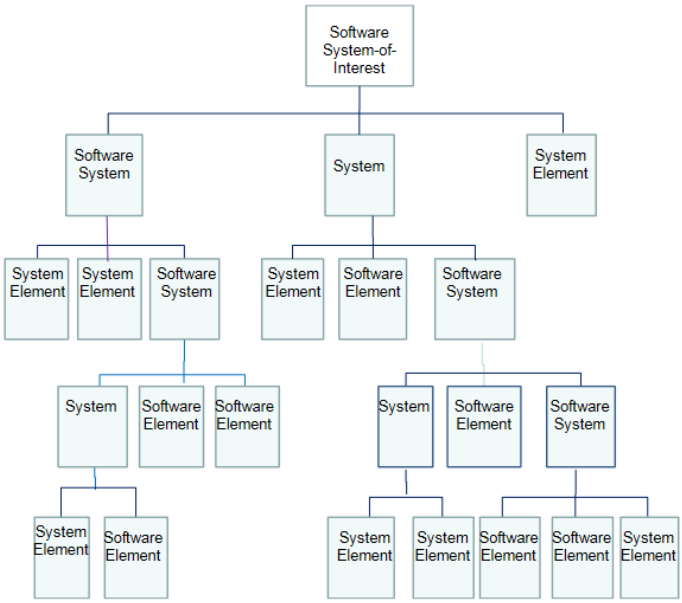
4.1.1 ISO/IEC/IEEE 12207:2017

The ISO/IEC/IEEE 12207 standard is a software development standard or framework for building and maintaining software. These processes are achieved through constant feedback through stakeholders, which would be everyone involved in this project. The goal of the 12207 standard is to provide general processes for defining, controlling, and improving the software life cycle processes within an organization or project [12]. The 12207 standard does not prescribe a specific process to develop software, but it does provide models, methodologies, modelling approaches, and techniques. It is up to us to choose which processes helped us succeed in this project, which we will go into further into this section.

Section 5.2.2 of the 12207 standard manual provides a diagram standard for visually organizing the project. Figure 29 shows at the top should be the overall system and branching off of it should be systems, software systems, and system elements, which in turn can include subsystems of those types. Organizing the project in this way helps us stay focused on the tasks at hand.

This document helps identify processes, which can be technical or administrative in nature as well as various modeling architectures for defining the project functions or tasks.

Figure 21: Standardized Software Flow Visualization



4.1.2 IEEE 830-1998 Software Requirements Specifications (SRS)

The IEEE 830-1998 standards is about putting together clear software requirements. It outlines techniques for communication between customer and client, which in this case would be this project's group members, to put in writing what the expectations for the software should accomplish. The SRS is meant to discuss the product and not the project. This means that costs, delivery schedules, etc. should not be mentioned in the SRS. The basic issues it should address are [13]:

- *Functionality*: What is the software supposed to do?
- *External Interfaces*: How does the software interact with people, the system's hardware, other hardware, and other software?
- *Performance*: What is the speed, availability, response time, recovery time of various software functions?
- *Attributes*: What are the portability, correctness, maintainability, security considerations?
- *Design constraints imposed on an implementation*: Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

Following these guidelines gives us clear directions in developing a functional product. Overall, the SRS should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, and traceable.

4.1.3 IEEE 802.11 b/g/n

IEEE 802.11 is a subset of the 802 Local Area Network standard which covers Wi-Fi. This standard was created by IEEE LAN/MAN Standards Committee in

1997. The b/g/n denotes frequencies 2.4 GHz at 11Mbit/s, 2.4 GHz at 22 Mbits, and 2.4 or 5 GHz at 54 Mbit/s. This standard is important because it standardizes the flow of information over Wi-Fi networks, so that devices can be plug and play for consumers without deeper configuration.

4.1.4 IEC 60335-1:2020

IEC 60335 covers household appliances. Section 1 covers safety of electrical appliances for households. The rated voltage should not be more than 250V for single-phase appliances and 480V for other appliances including direct current supplied appliances and battery-operated appliances [14]. Using this standard ensures we create a safe household appliance.

4.1.5 ISO/IEC 9899

ISO/IEC 9899 is a standard for programming in the C programming language. Some of the features of this standard include specifications for:

- The representation for C programs
- The syntax and constraints of the C language
- The semantic rules for interpreting C programs
- The representation of input and output data processed/produced by C programs

Mostly this standard is for implementing your own compiler, but it can be used as a reference for proper syntax and program structure [15].

4.1.6 Effective Dart

This standard shows how to write Dart programs with a clear style. The first part shows how to name identifiers, classes, libraries, etc. The second part shows how to write effective documentation. The third part, importantly, shows the proper usage of Dart programming tools and what pitfalls to avoid. The final part of this document shows how to write a clean, consistent program design [16].

4.1.7 Google JavaScript Style Guide

The JavaScript programming language is notorious for its loose style. Because of this, it’s really easy to write incomprehensible code. Using Google’s styling guide [17] helps us conform to a single, clean style when programming in JavaScript.

4.1.8 ATX Power Standard

The “ATX” in DTK Computer ATX SMPS Power Source stands for “Advanced Technology eXtended”. This was a new power source standard created by Intel in 1995 to replace the Baby-AT standard that their power sources adhered to prior to the invention of ATX. Its improvements include doubling the height of the I/O panel, and relocation of the drive I/O, processor, and memory. The table shown in Table 29 gives a feature summary of ATX power sources and how the improvements from Baby-AT standards correlate to the features.

Table 24: ATX Feature Summary

Feature	Benefit
Double height flexible I/O panel allows higher integration	● Lower cost ● Fewer cables ● Improved reliability ● Shorter assembly time ● Support for connectivity and I/O standards like USB, TV in/out, ISDN, etc. ● Integrated graphics allows use of unified frame buffer architecture
Relocated drive I/O means shorter cables	● Reduced cost ● Support for faster drives such as PIO Mode ⅘ IDE drives
Relocated processor and memory	● All full-length expansion slots ● Easier to upgrade processor ● Easier to upgrade memory ● Easier to add cards ● Relocated processor allows easier use of bulk capacitance and voltage regulation circuitry

4.1.9 SHA-256 Hashing Algorithm

The SHA-256 secure hashing standard specifies the algorithm that performs a one-way operation on a value to transform it into a 256-bit output known as a message digest. This standard, along with many other hashing standards, allows for the added security of obscuring the true values of sensitive information when being stored in databases. Commonly hashed values include user passwords and social security numbers. This particular standard provides a sufficient amount of security for the project's intended small user population without requiring as much processing power as more secure algorithms. Implementing this standardized hashing algorithm, combined with a salting algorithm, in our software helps provide secure methods in which to handle sensitive user information.

4.2 Design Constraints

Before starting a project, it's important to consider the external factors that will affect our design decisions. To start, there are economic and time constraints,

how big our budget to build the project, and how much time is there to build it in. There are manufacturing and sustainability constraints, what factors are there in the manufacturing of the project to consider and how sustainable can the project be expected to last. Finally, most importantly, there are environmental, health, and safety constraints. Environmental constraints deal with factors that affect the environment, such as proper disposal of the product or the proper environment the product should operate in. Health constraints are areas to study where the health of the user can be an issue. Safety constraints cover any issue that can cause harm to the user or others around them. For example, electrical shock or light damage while using the product. Doing proper research into all of these areas helps us build a better product.

4.2.1 Economic and Time Constraints

Our goal was to get a working automated hydroponic system prototype under $1000. Being self-sponsored, we want to have a reasonable, self-imposed budget, but we can be flexible if needed. During our preliminary research, building the frame costs $150 to $250 as initially designed.

There are many different lighting products that easily exceed the totality of our budget. Those products are metal halide lights, which have environmental issues that we will get into in environmental constraints. For this reason, we chose to go with LED lights. LED lights have become more economical and power efficient in recent years and can serve as a reasonable light source for growing plants.

4.2.2 Manufacturing and Sustainability Constraints

Working in a wet environment, our sensors and parts can be prone to corrosion. The parts we choose should be able to survive for at least 6 months of continued use, and their lifespans are extended drastically when routine cleanings are done to the small moving parts in the system, such as the impeller in the recirculation pump and the reflective material on the outside of the optical water-level sensor.

When it comes to sustainability, hydroponic systems hold the future of sustainable farming practice. On average, hydroponic systems utilize almost 10 times less water than a soil-based grow [18]. Increasingly negative effects of climate change such as less available fresh water combined with overusing fresh water for animal and botanical agriculture, it is critical that society makes a vast shift in how we consume water in our industries or the planet could run out of usable water by 2040 [19]. This system aims to make it simple for the common consumer to maintain a hydroponic garden of their own to cut down on water waste in the agriculture industry sector not only through use of hydroponic technology, but also by keeping food local to the consumer, effectively cutting the agricultural supply chain (and its waste) out of the picture for certain vegetables.

4.2.3 Environmental, Health, and Safety Constraints

Having a closed loop of nutrient water, it is important to research the health impacts of having an open or closed water tank for the user as well as the plants being grown. It is important to choose a plant nutrient mix that avoids adverse effects of its user because old nutrient water can grow undesirable bacteria and fungi. The purpose of a hydroponic system is to grow plants, so we need to know how often the water must be changed to prevent adverse effects on the plants and its users. It was found that the water must be changed every two weeks.

With that in mind, the water delivery system should be closed to prevent damage to external parts, such as the power supply, peripheral devices, and the microcontroller units. More importantly, dry electrical parts will prevent electric shock for the user.

It is important to use lights that have a wavelength that is safe to the user. There are lighting systems that can produce enough heat to require an HVAC system. For this reason, we chose LED lighting.

4.2.4 Ethical, Social, and Political Constraints

Ethically, we have a duty to design a system safe to its users. This means building a system that does not cause harm. This does not only include physical harm. Because we built an online application, we have a duty to protect the user's data and information. We included encryption algorithms for storing the user's login information to prevent bad actors from obtaining it.

Our power system is designed using US standards. This means conforming to IEC 60335-1 electrical requirements of 120 volts. Doing so allows us to create a safe product using the US electrical grid.

We have a duty to build a functional system that sustains plant life. Knowingly building a subpar product would be unethical. It is important to us to build a hydroponic system that lasts and properly document the life expectancy of our parts.

5.0 Project Hardware and Software Design

In the following sections, the design specifics of both the hardware and software sides of the project is discussed. Each major hardware system of the Hydroponic Unit has its implementation specified in their own sections. The coverage of software system design is separated into three parts, as three major software systems are implemented. These software systems are the MCU's code, the mobile application, and the Node server.

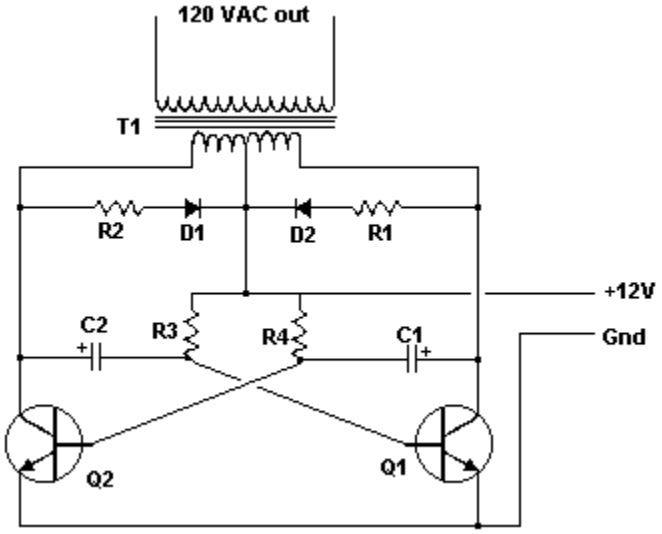
5.1 Power Supply System

Due the wide range of sensors and pumps that are used in the system, there are differing power supply requirements for each individual component. These requirements range from ~3.3V-5V, so separate busses are necessary to provide the respective voltage. The DTK Computer ATX SMPS power supply has supply rails of ±3.3V, ±5V, and ±12V.

5.1.1 DC/AC Inverter

Due to design constraints both our main supply pump and mixer runs on AC voltage. Considering that our main power supply can only supply DC voltage, an inverter would be needed to both step-up and convert the 12V DC voltage to 120V/60Hz AC voltage. The schematic that our inverter is based is shown below:

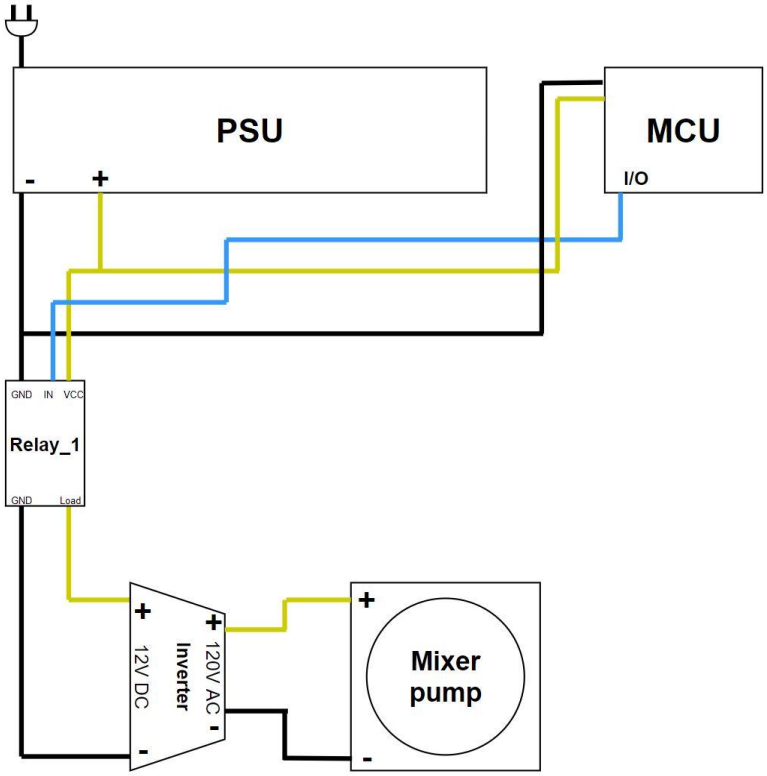
Figure 22: Inverter Schematic



As shown, this inverter design is made to work within a 12V DC system commonly found in vehicles. By implementing this device in the main power source rail, we can easily convert the voltage and thus be able to drive our AC

voltage dependent pumps with no issues. The function schematic below shows how the inverter is implemented in our system.

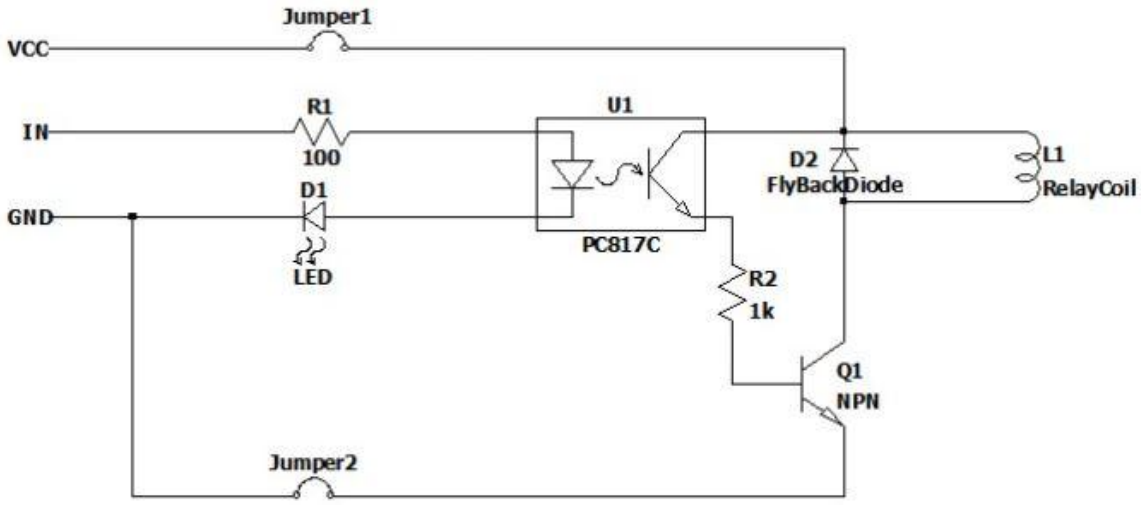
Figure 23: Inverter Use Schematic



5.2 Relay Controls

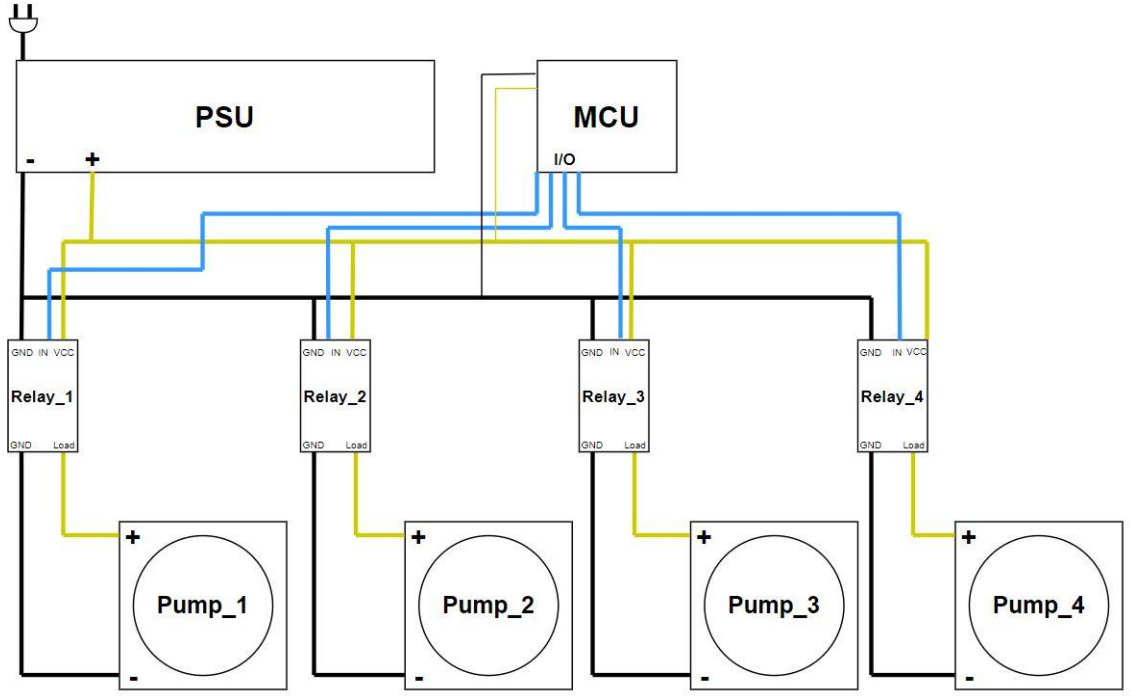
Most of the high current device loads in our system is managed through 3.3V driven relays. This was due to our MCU design as it only has 3.3V communication outlets. The relays act as a power gateway that allows 12V DC into each element whenever needed. As mentioned, our relay choice is the SRD-DC03V-SL-C based unit as our requirements fit within its function range. Below is the schematic for the relay component and it's driving elements:

Figure 24: SRD-DC03V-SL-C Based Unit Schematic



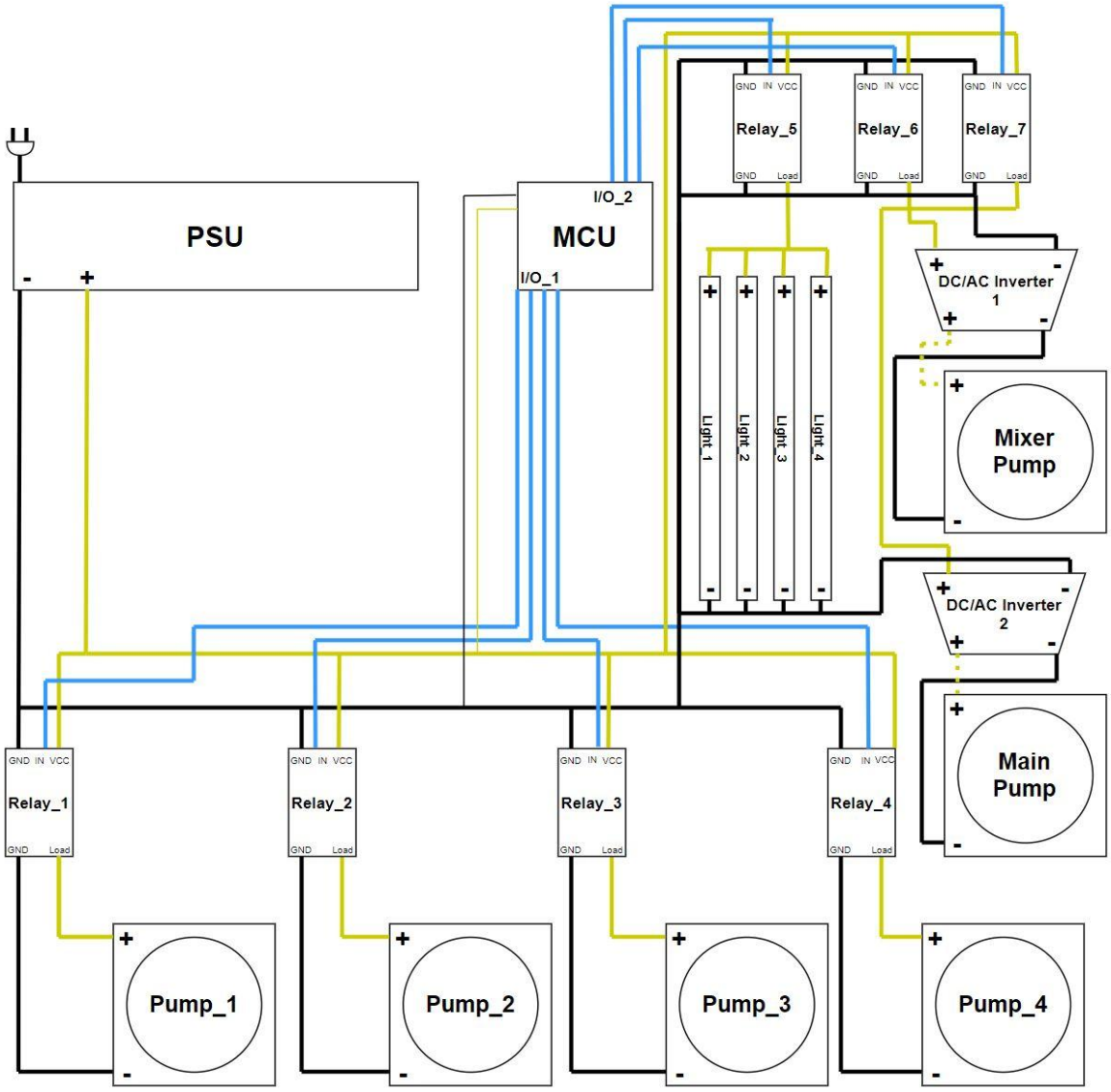
As we can see, the relay element uses the 3.3V signal through its IN port in order to either close or open the relay. Additionally, the relay can be wired in either actuate when it detects 3.3V or when there is an absence of 3.3V on the IN rail. Below its function schematic with the dosing system:

Figure 25: Dosing Pump Relay Schematic



Each pump is independently controlled by the MCU via the 3.3V (I/O) line. Lastly below we can see the entire relay system in the function schematic below:

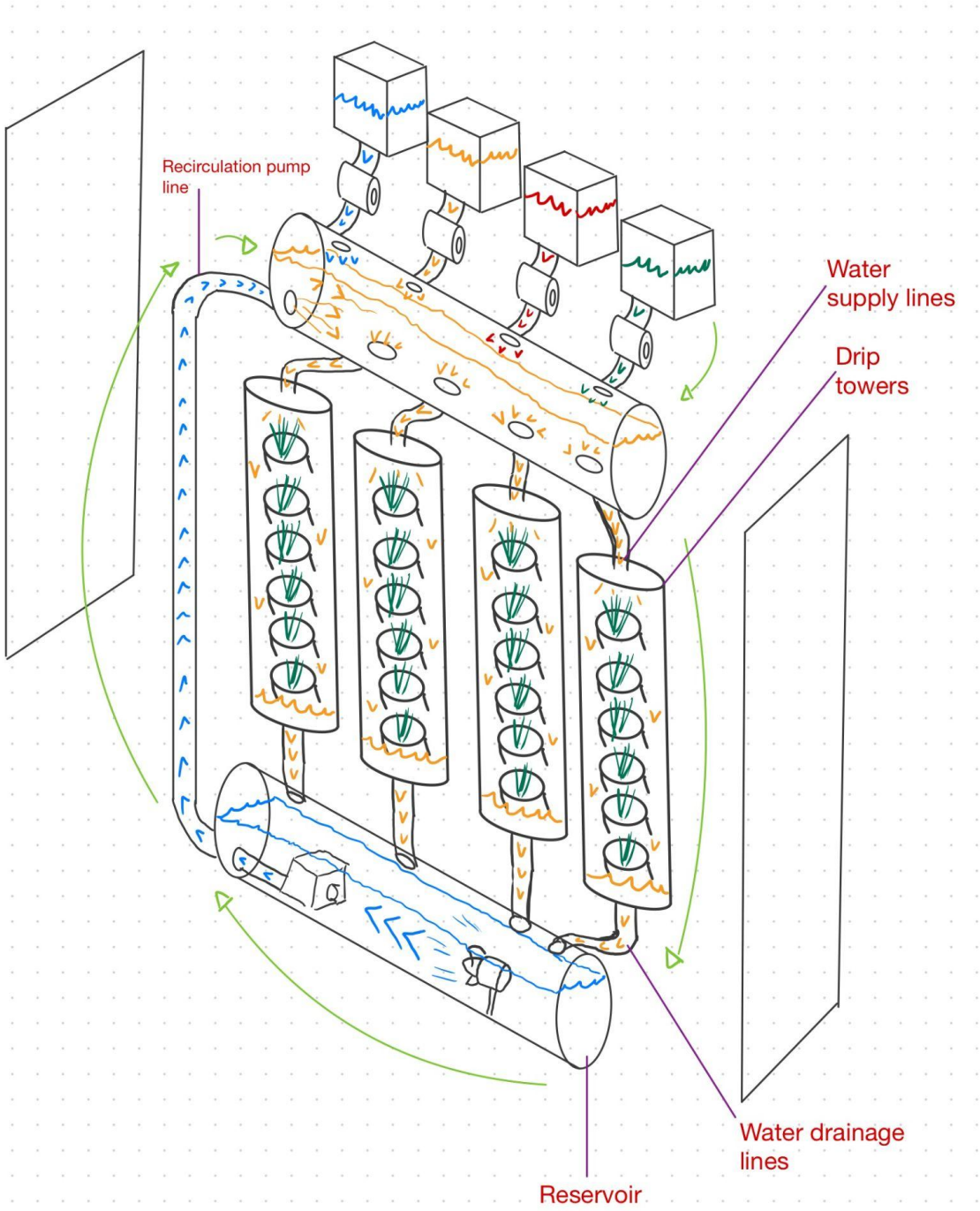
Figure 26: Relay System Schematic



5.3 Water Flow System

The water recirculation in this system is dictated by the recirculation pump that moves the fluid to the top of the hydroponic drip towers where gravity is used to simply allow the water-based nutrient solution to trickle down through the root systems of the plants where it will fall into a drainage line towards the recirculation reservoir. Figure 35 shows a sketch of how water will flow from the recirculation reservoir through the towers and back to the reservoir, with water flow direction represented with blue arrows. Upon further testing, it was decided that we would use one tank to store all the water.

Figure 27: Water Flow Diagram

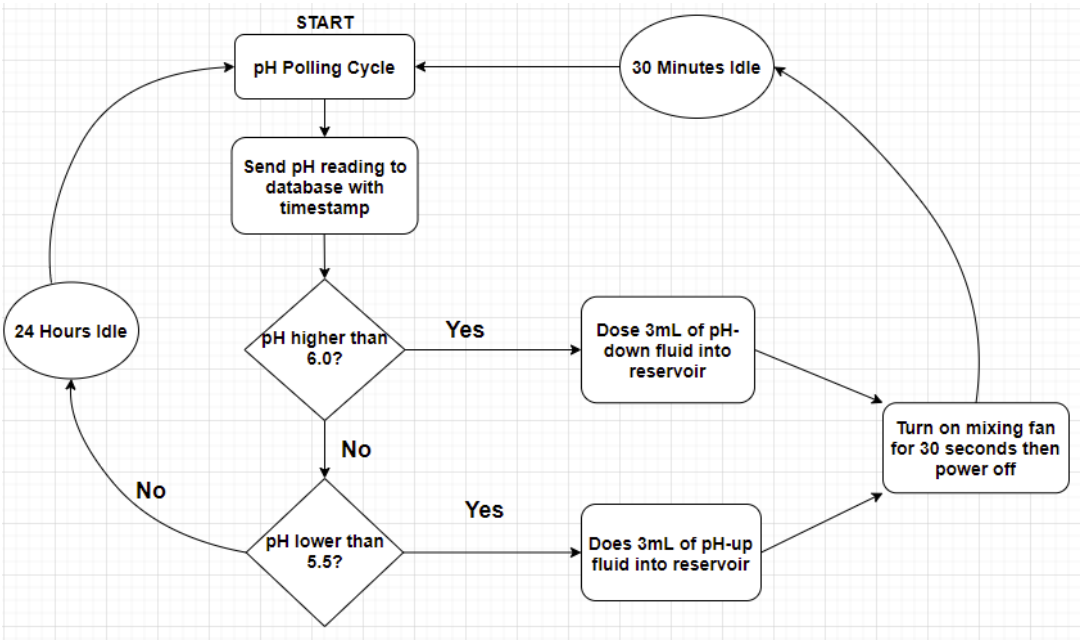


5.4 pH Control System

The approach for adjusting and maintaining proper pH balance in the recirculation reservoir will be straightforward; when a pH reading is made by the sensor, it will see if that digitally converted analog reading is higher or lower than the target range of [5.5, 6.0]. Depending on if it is higher or lower, a fixed amount of the pH up or pH down fluid is injected into the reservoir, then mixed for 30 seconds. The pH reading is sent to the database and the sensor remains idle for 30 minutes, then repeat this process. Only once the pH has been rebalanced within range will it go completely idle for 24 hours. Figure 36 outlines the process of pH adjustment that is implemented.

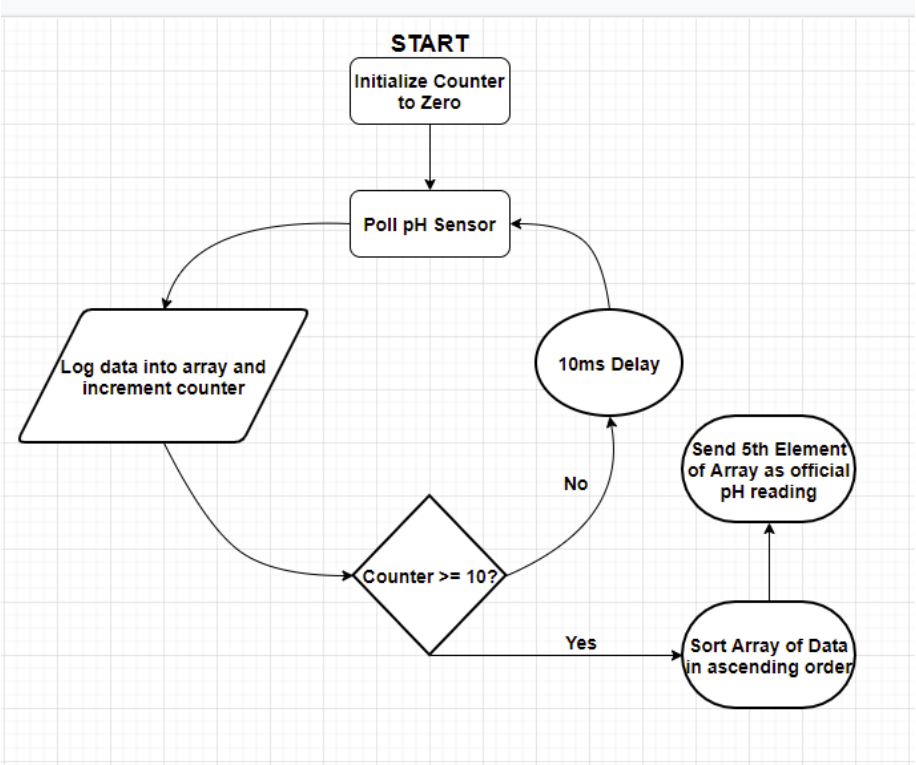
To avoid overjustification of the pH (i.e. too much adjustment fluid added that will put pH out of balance in the other direction), a small amount of adjustment fluid is added to the reservoir and mixed at a time. This slows down the rebalancing process, ensuring that too much solution isn't used, and cost-efficiency is optimized for the user when it comes to resupplying on pH up/down.

Figure 28: pH Adjustment and Monitoring Cycle



The pH sensor selected for the project has an analog reading drawn from it. To avoid one ill-timed pH reading that could initiate an adjustment cycle, many polls are taken with the sensor, and the median number of that data is the official pH approximation of that time. Figure 37 shows the software flow of how this process is performed.

Figure 29: pH Polling Cycle



5.5 Lighting System

As per figure 39, the lighting system is controlled by the MCU through relays for either ON or OFF operation. There is no need to control light intensity through potentiometers since the plants will need full capacity when the lights are engaged. The main control requirements consist of enabling the lights through certain periods of time depending on the plants within the system. Additionally, to provide the plants with as much UV light exposure as possible, we opted to place four light elements, one on each corner of the main system within proper distance, thus covering as many angles as possible. The system is simple yet effective for our use case.

5.6 MCU System

The MCU controller system used for the system, TI's CC3220MODASM2MONR handles all GPIO inputs and outputs, as well as controlling the relay system for toggling the mixing fan, dosing pumps, and recirculation pump to an on or off state. The system also comes equipped with reset capabilities, UART functionality for programming, and protection circuits to avoid current leakage or amperage overloading. The MCU is the core of the system.

5.7 Software Design

Software is the key component for controlling the behaviors of the hydroponic system. There are two separate systems we developed. The first is the microcontroller. This system is focused on the operations which will sustain the life of the plants. This includes monitoring the water levels, pumping water, and balancing the pH of the water as well as the systems for sending its data over the internet. The second is our mobile application. This system focuses on storing data collected from the microcontroller into an online database. It is able to collect and display time series data of the hydroponic system as well as send commands back to the MCU.

The figure below shows our software systems hierarchy in the ISO/IEC/IEEE 12207 standard. At the top of this hierarchy is the system we are creating, which is the automated hydroponic system. This system has three parts: the MCU application, which controls the direct functions which will sustain plant life via the microcontroller unit, the mobile application, which consists of a graphics user interface to log sensor data and interact with the MCU over the internet, and the Node.js server, which houses our database management system and Express API. The following sections will go more in depth about the technologies and diagrams of how we expect in each respective part of the hydroponic system.

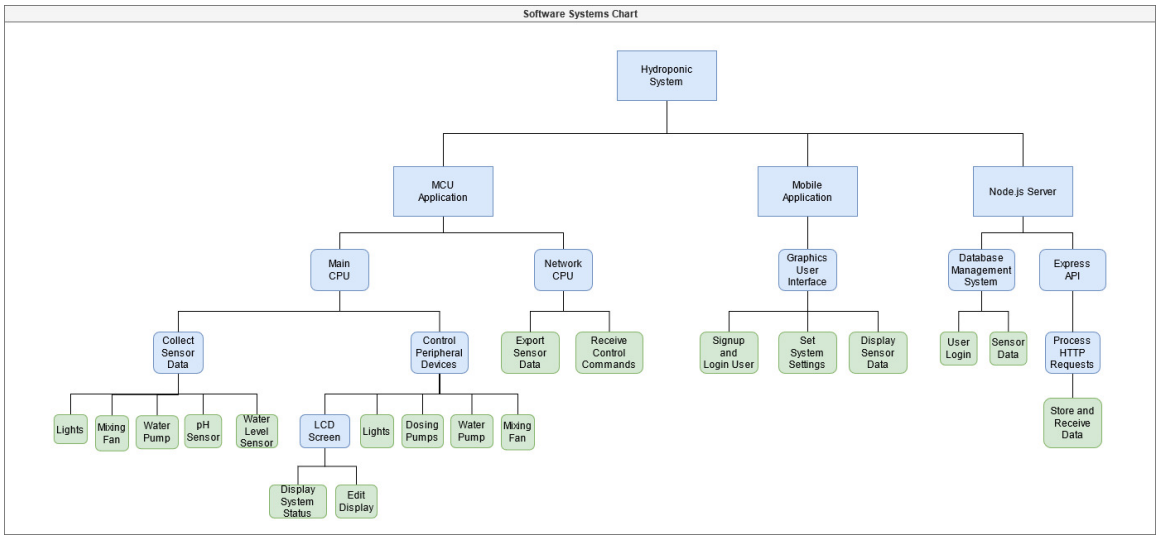


Figure 30: Software Systems Chart

5.7.1 MCU Software Development

Developing software for the CC3220 initially seemed very daunting. The user's manual included 1100 pages of register and bit information for accessing the microcontroller's various functionalities. Fortunately, there are software tools

developed by Texas Instruments that make these processes fairly straightforward.

5.7.1a Code Composer Studio

Code Composer Studio (CCS) is an Integrated Development Environment (IDE). An IDE is a software development tool that can handle building large software projects. CCS handles linking the TI operating system and its libraries in the action of building the project. Importing libraries, which are files with name definitions and functions that can be used in projects, are also an important tool when building software.

An important part of an IDE is its text editor. This portion has code completion, checking for proper syntax, and displaying build errors to help the user resolve. Using and creating keyboard macros for common tasks, such as renaming variables or commenting out blocks of code is a helpful tool when using an IDE to build software.

5.7.1b Software Drivers

The driver for the hydroponic system is created using Code Composer Studio's MCU based libraries. CCS has software for building the operating system, linking libraries and software tools for writing clean code. To start, we import the empty project from the drivers folder. This empty project contains definitions for a project that toggles an LED. It shows that GPIO definitions are controlled by a SysConfig file. SysConfig is software that creates GPIO definitions and configurations based on the MCU. Some of these configurations specify which pins are used for I2C, SPI, UART, or GPIO. When the file is saved, the configuration file is created and can be used in our project.

CCS has special drivers for controlling our many peripherals. These drivers take care of event handling and manipulating configuration of registers behind the scenes. These register names have their own library definitions connected to special memory addresses. Without the driver libraries, we would need to use the MCU's user manual to create our own functions. The user manual for the CC3220MODAS is over 1100 pages of register information, with each register's individual bits having their own functionality, so using TI's special libraries help speed up development and keep the software tidy.

The CCS drivers we'll need for this project are as follows:

5.7.1b (i) GPIO Driver

General Purpose Input Output pins control most of the functions of the MCU. This driver is the basis for easily configuring the various states of the GPIO pins.

- GPIO_Init configures the GPIO pins for the specified board in the SysConfig configuration file.

- GPIO_setConfig sets the pin specified with the SysConfig generated configuration file. GPIO.h has simple definitions for every configuration.

5.7.1b (ii) Display Driver

Display.h contains an interface for working with an LCD, UART, and more. For the LCD and UART, we need to print sensor data and status of our peripheral devices to the user. For this project, we can use this driver to:

- Initialize display based on type
- Clear data by line or entire screen
- Print text on screen by row and column.

5.7.1b (iii) HTTP Communications Library

Httpclient.h is a functional library provided by Texas Instruments that provides easy to use methods to make and fulfill HTTP requests.

- Will provide most of the MCU's database communication functionality
- Creates a connection to an HTTP server
- Implements and makes available HTTP request methods, such as GET and POST

5.7.1b (iv) Simplelink Basic Library

Simplelink.h is a multipurpose library intended for use with TI's SimpleLink™ line of products that provides the following functionality to projects:

- Includes a number of other libraries to allow for making and managing a connection to a WiFi network
- Error and event tracking for WiFi connection events
- Client & Server socket controls
- Configure device IP & MAC addresses

5.7.1c MCU Application Design

The MCU application controls the functions closest to automating sustaining plant life and managing network functions. This application is split into two parts, the Main CPU thread which handles the collection of sensor data and controlling peripheral devices, and the Network CPU which acts as both a client and a server to communicate with the Node.js server. It is written in the C programming language using Code Composer Studio's CC3220 MCU software development kit.

5.7.1c (i) Main CPU Thread

This program collects sensor data regularly and directly controls the LCD screen, LED lights, dosing pumps, water pump, mixing fan, pH sensor, and water level sensor. The purpose of this program is to automate the functions of the

hydroponic system by controlling the peripheral devices and collecting sensor data.

Each peripheral has its own header file. The LCD header file has operations editing the state of the LCD screen. The Lights, Water Pump, and Mixing Fan header files have operations turning those devices on and off. Similarly, the Dosing Pump header file has operations turning the pH up, pH down, and nutrient dosers on and off.

The program starts by initializing all the pins with their respective configurations that are generated by the Code Composer Studio Sysconfig tool. Next, the sensor cycle begins. The sensor data being collected are the status of the lights, mixing fan, water pump, and water level sensor and the values generated by the pH sensor. If the pH is out of balance by more than 5%, the correct pH dosing chemical pump will drop the chemical into the water tank. This process will repeat every hour.

There is an interrupt that resolves new settings sent from the mobile application. These settings include the state of the lights, water pump, and mixing fan.

There is an interrupt when the water level sensor detects open air that will stop the water pump and display to the user that water needs to be added.

5.7.1c (ii) Network CPU Thread

The network CPU is responsible for handling incoming and outgoing network traffic to and from the Hydroponics Unit. Three main tasks are carried out by the network CPU core: the establishment and maintenance of a Wi-Fi connection, the creation and handling of an internal HTTP server to receive hardware commands, and the making of outgoing HTTP requests.

Of the three tasks, the Wi-Fi connection must be accomplished first, as the remaining networking tasks require this connection. Using the SimpleLink™ libraries provided by Texas Instruments, a connection to the Wi-Fi network with the specified credentials is attempted. If successful, an IP address is assigned to the MCU and network operations may now be used. If unsuccessful, an error message is displayed to the user, and another attempt to connect to the network will occur.

The next task is to set up the internal HTTP server, which acts as the receiver for hardware commands from the mobile application in the form of HTTP requests. Once the server is active, it remains active while it waits for requests to be received. When a request is received, the server uses the URI in the request to locate and call the proper internal API endpoint to fulfill it. The endpoint's software routine will then perform the necessary hardware actions to execute the command.

Outgoing HTTP requests are performed whenever sensor data is to be sent to the database, either occurring according to the defined schedule in the automatic maintenance procedure, or being requested manually by the user through the mobile application. After the necessary data has been sampled by the sensors, it is formatted into a JSON package to be sent in a POST request. Using the HTTP client functions included in the SimpleLink™ library, the request will be sent to the Node server using the specified headers and URI. The request is then processed by the Node server, which will result in the sensor data being deposited into the database. The MCU will then receive a response code from the Node server, indicating whether or not the HTTP request was successfully fulfilled or if any errors were encountered.

5.7.1d Security Features

For the software that is hosted on the MCU, built-in security features provided by Texas Instruments are used, such as HTTPS server hosting, WPA wireless security standards, and secure sockets. The internal API, which houses the endpoints that execute hardware commands sent by the mobile application, will also be protected through the Node server's API security. This is possible due to the fact that the Node server's API will be the only method used to call the MCU's API. The received API security will sanitize user inputs and require valid authorizations and session tokens, protecting the MCU from invalid API calls.

5.7.2 Mobile Application Software Development

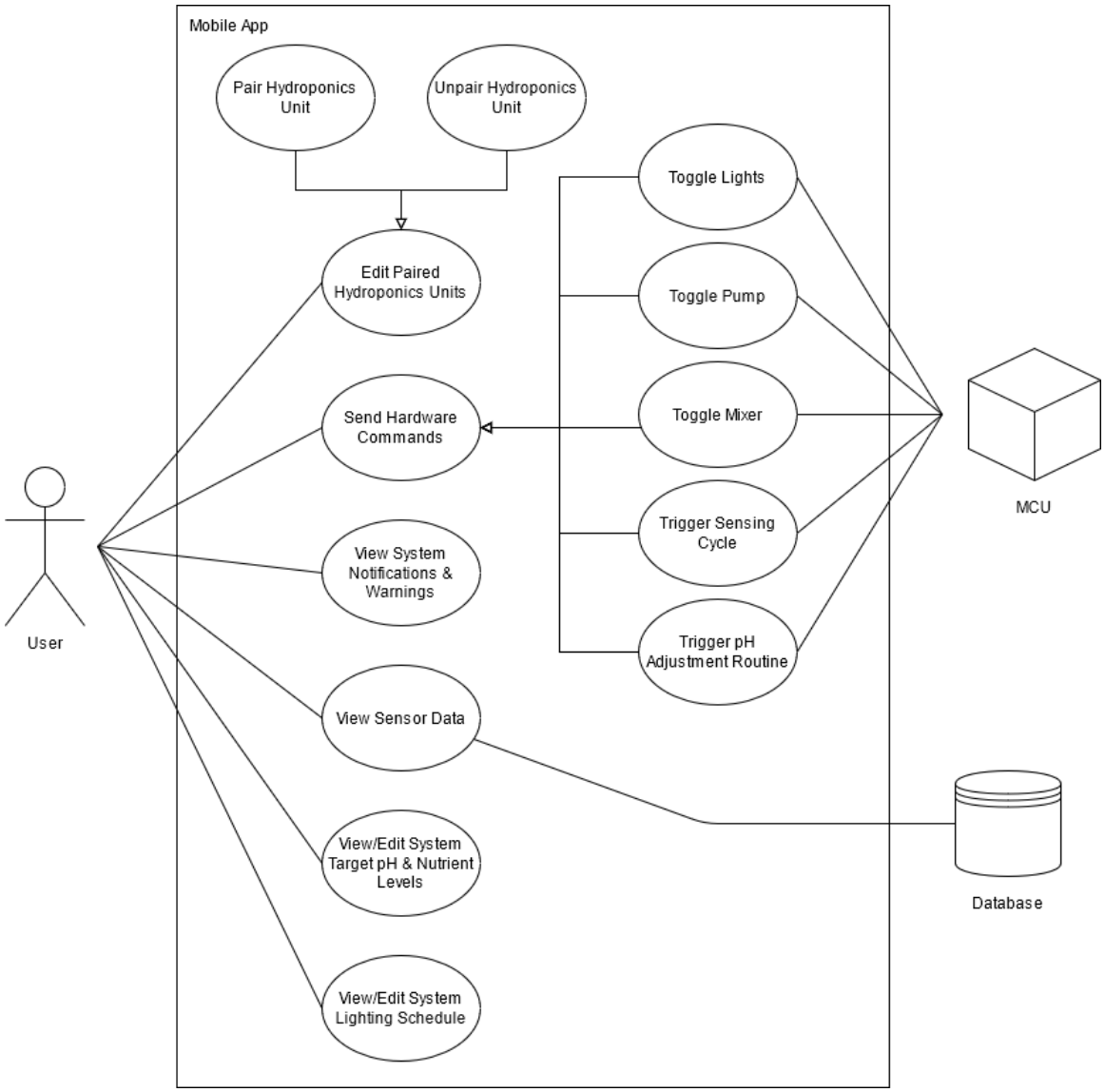
The mobile application, serving as the user's primary method of interacting with the Hydroponics Unit, must be designed with several key concepts in mind. The app must: be easy to use, display sensor data both efficiently and effectively, provide the user with control over the Unit's functionality, and have a professional and attractive user interface. To describe how these design concepts will be implemented, the following sections will discuss the various features, algorithms, and design philosophies used in the mobile application's conception.

The mobile application is created using Flutter using Android Studio IDE. This section includes the totality of the mobile application's user interface design and its operations.

5.7.2a User Functionality

The mobile application must be sufficiently functional to the user, allowing them to easily view sensor data gathered over time and send various hardware commands to the Unit. To better visualize the possible actions the user may perform with the mobile application, the following use case diagram has been included. For the sake of simplicity, common use cases such as account creation, deletion, and logging in have been excluded from the diagram.

Figure 31: Mobile Application Use Case Diagram



As it can be seen, the user has a large amount of control over the Unit's operation through the mobile application. Each major hardware system is able to be toggled manually, and the target parameters of the Unit's automatic maintenance system, such as the pH and nutrient levels, are editable. However, the final iteration of the project may not include all use cases, as a number of them are considered either stretch goals or quality of life features. For example, the ability to edit target pH and nutrient levels are not features that are strictly necessary for the Unit to support in order for it to successfully operate. As such, by only allowing the Unit to use hard-coded target values, it is possible to exclude the editing feature if the project's success deems it necessary. The use cases that are considered stretch goals include hardware toggling commands, editing target pH and nutrient levels, and editing the automatic lighting schedule. Through the use of hardcoded software values or physical switches on the Unit,

these features become unnecessary, but their inclusion can still be justified by the provided convenience and flexibility to the user.

Certain use cases must also have constraints in place for the user so that they are unable to create unstable conditions for the system's plants. One such use case requiring constraints would be the editing of pH and nutrient levels. The user should not be allowed to set extreme pH target levels, as this would most likely result in the death of the plants, which is not a desirable outcome for the user. The same holds true for the setting of nutrient levels, which must have its allowed range of values limited.

5.7.2b UI Design

This section contains the user interface design. These mockups were created in Figma and are not meant to be one-to-one comparisons, but are meant to serve as a guide when creating the UI with Flutter.

5.7.2b (i) Login View

The login view is the first thing that appears when the application is started. On this view, the user must enter their credentials. If successful, a positive message appears, which will then take the user to the status view. If unsuccessful, a message appears that the login process was unsuccessful. Pressing okay will keep the user on this view. If the user does not have credentials, clicking the signup link will take the user to the signup view.

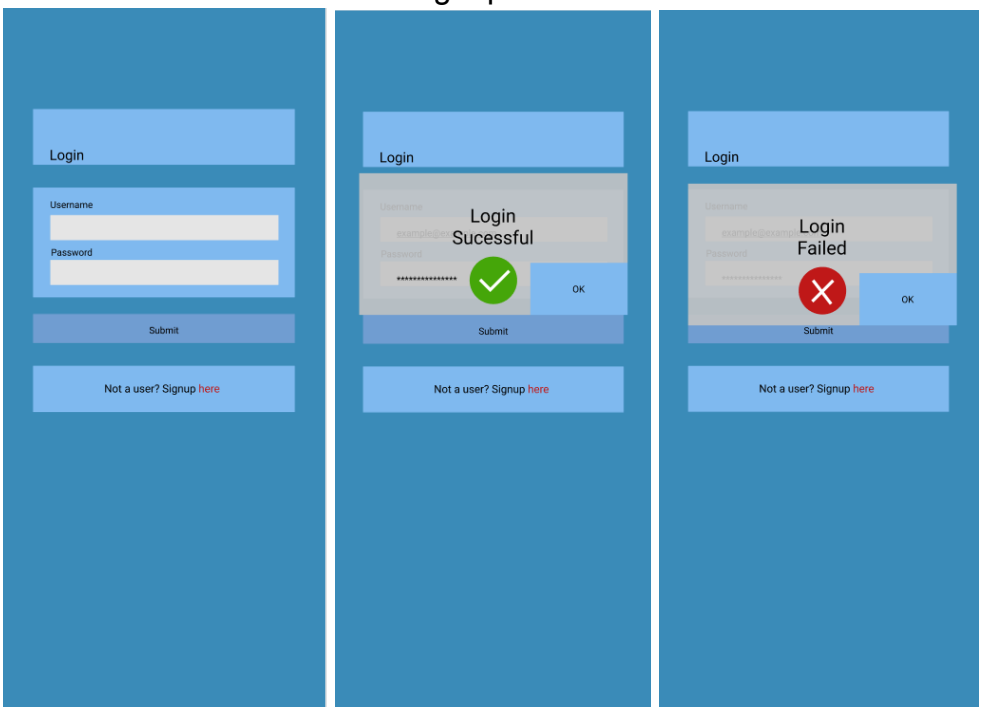


Figure 32: Login Views

5.7.2b (ii) Signup View

The signup view is the view where the user can create a username and password, and unit number. The unit number's location is on the LCD of the hydroponics unit. Similar to the login view, if the signup process is a success, a positive message is displayed to the user, taking them to the login screen. If the signup process is unsuccessful, a negative message with the error is displayed to the user and will keep the user on the signup view.

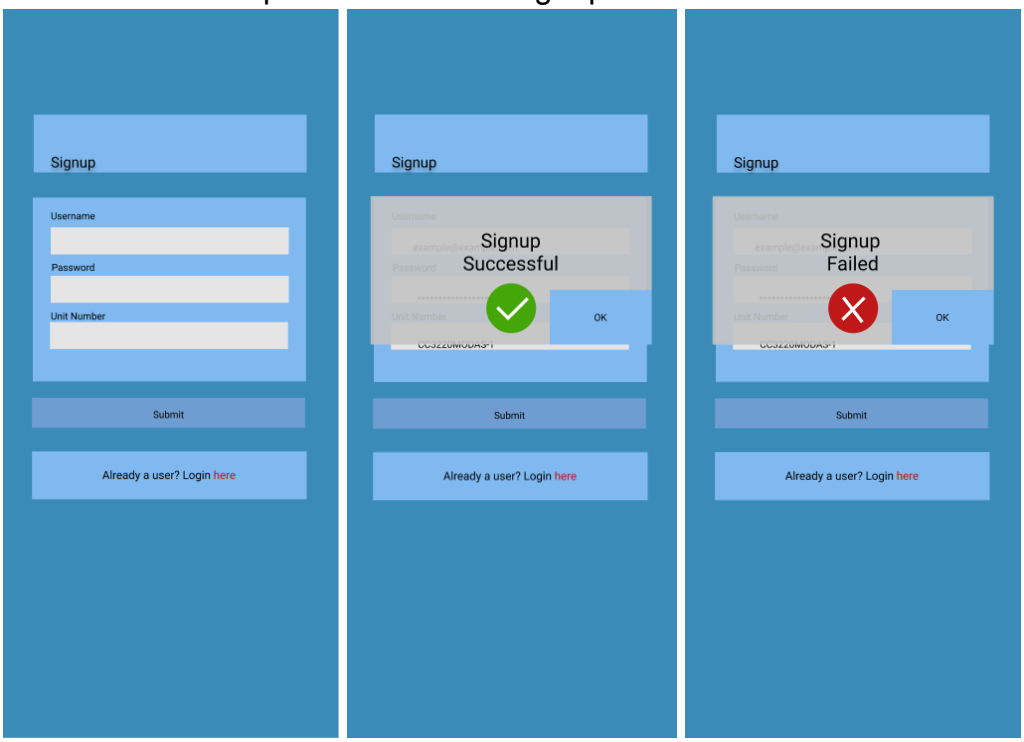


Figure 33: Signup Views

5.7.2b (iii) Status View

The status view is where the user is displayed the last known statuses of the hydroponic system. The user can see the status of the pH sensor, water pump, lights, mixing fan, and water level. The views outside of login and signup have a header showing the user what view they are on, a logout button that takes the user back to the login screen when pressed, and a navigation bar on the button that can take the user between the status view, system settings view, and graphs view.

5.7.2b (iv) System Settings View

The system settings view allows the user to send commands to the hydroponics unit. In this view, the user can control the lights, water pump, and mixing fan through a toggle display. The last status known in the database is the value displayed in this view. When the save button is pressed, the commands are sent to the hydroponics unit. On success or failure, a positive or negative message is displayed.

5.7.2b (v) Graphs View

In the graphs view, the time series data of the tracked categories is displayed. Each category has its own graph.

Figure 34: Status View, Systems Settings View, and Graphs View

5.7.3 Node.js Server

The Node.js server houses the database management system and the Express-based API. The database management system stores sensor data and the Express API is used as an interface used for interacting with the hydroponics unit and the database. The following sections describe the contents of each of these systems.

5.7.3a Database Management System

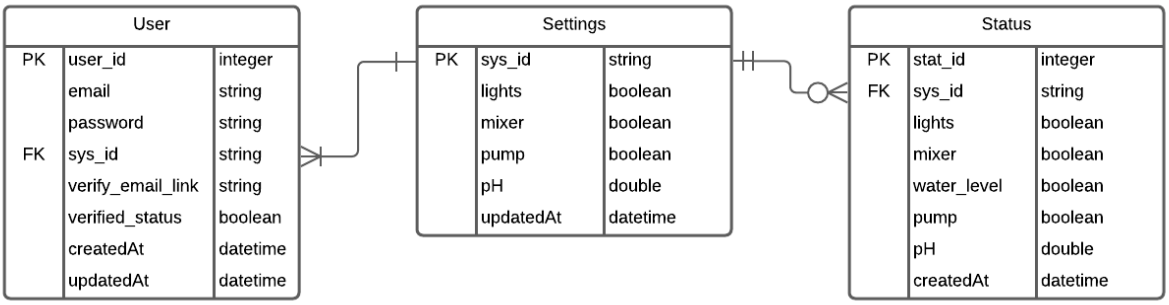
Through Heroku, the Node server used ClearDB as a hosting service for the MySQL-based database, which contained a table to store mobile application user accounts and their paired hydroponic units, and another table to store each unit's sensor data. To better show how the table for a unit's sensor data is constructed, a sample format of each data entry is shown below.

Table 25: Sample Data Entry

Sample No.	Unit ID No.	pH	Tank Level	Light Status	Mixer Status	Pump Status	Timestamp
1	1	5.7	True	True	False	True	10:20 AM 6/20/21

As for the overall organization of the database, the following entity relationship diagram serves as a model for implementing how each table interacts with each other, allowing Hydroponic Units and their sensor data to be referenced by a specified user.

Figure 35: Software Entity Relationship Diagram



5.7.3b Express API

The Express API system, hosted on the Node server, serves as the software interface between the mobile application, the database, and the Hydroponic Unit. The following API endpoints are implemented:

Signup User:

- Creates a new user account with the given credentials after hashing and salting their password in the database
- Success Conditions:
    - The username does not exist in the database
    - There is a valid hydroponics unit id
        - The unit id does not exist in the database
        - Follows the unit id convention

Login User:

- Securely signs the user in with a valid session token
- Success Conditions:
    - Username and Password match database entry

Update User Information:

- Modifies a user's credentials in the database to match their new information
- Success Conditions:
    - User has a currently valid session token

Delete User:

- Removes a user's database entry, effectively deleting their account
- Success Conditions:
    - User has a currently valid session token

Get Most Recent Sensor Data:

- Retrieves the most recent sensor data entry for a specified unit
- Success Conditions:
    - User has a currently valid session token

Get All Time-Series Sensor Data:

- Retrieves a set of sensor data entries for a given amount of time for use in a graph for trend analysis
- Success Conditions:
    - User has a currently valid session token

Send Set of Hardware Commands to Unit:

- Sends a list of new hardware states and target maintenance levels to the specified hydroponics unit
- Possible Commands:
    - Toggle: pump, lights, mixer
    - Trigger: sensing cycle, pH adjustment cycle
    - Edit: pH target value, nutrient target value, lighting schedule
- Success Conditions:
    - pH and nutrient target values are within safe tolerances for healthy plant growth
    - User has a currently valid session token

5.7.3c Security Features

As previously discussed, the Node server implements various security features in order to protect the different software systems involved in the project. Password protection is achieved through the use of the SHA-256 hashing algorithm combined with salting, as well as requiring a minimum level of complexity in the user's password to help protect against brute-force attacks. Anti-SQL injection security measures are included in the form of input sanitization routines, which checks for invalid characters and parameterize SQL queries in order to prevent user inputted information from being executed as code. Securing the API is done by requiring users to have valid session tokens and proper authorizations, which is provided through the use of the OAuth 2.0 protocol.

6.0 Project Prototype Construction

To ensure proper construction for the final system, hardware and software prototypes are constructed to initiate error testing and debugging. The following sections include these prototypes with details about how the iterations of the system are aiming to be functional and meet system requirements.

6.1 Integrated Schematics

Figures 44, 45, and 46 show the integrated schematics for the system hardware. These schematics respectively depict the logical pinout diagram for all system hardware elements and the more specific circuitry diagrams for reset buttons, power supplies, and pin jumpers.

Figure 36: Pinout Design Diagram

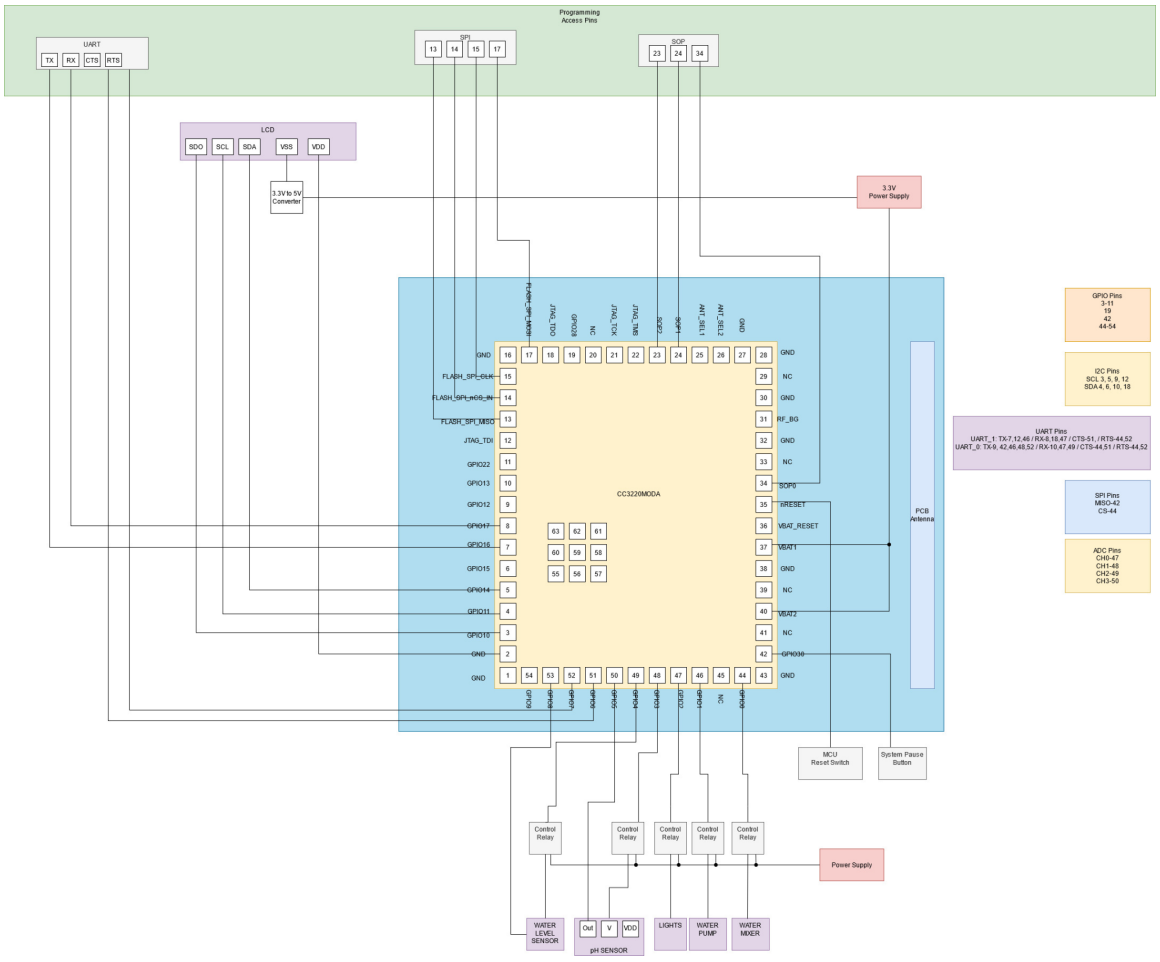


Figure 37 details the schematic for the final PCB design to be used in the system. This PCB includes pin headers for the MCU pins into pullup resistors

where needed. The PCB design also includes a step-up voltage converter into 5V DC current, which is drawn from the main MCU power rail, and delivers 5V DC current to the sensor components to be soldered directly onto the board that needs them, such as the pH sensor and water-level sensor.

Figure 37: MCU Prototype Schematic

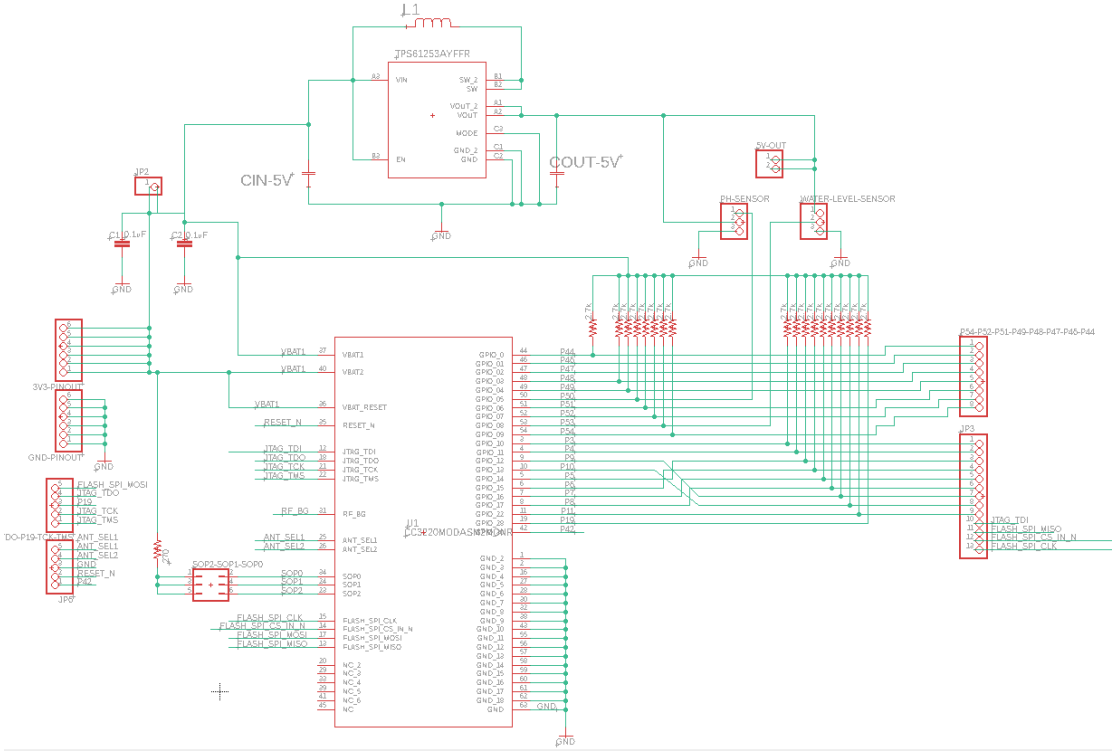
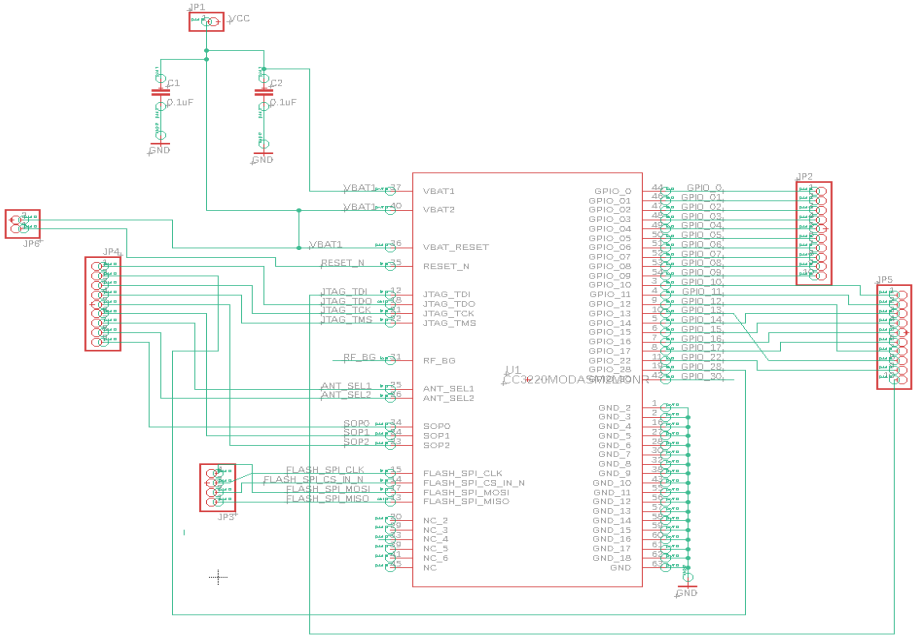


Figure 38: MCU Breakout Board Prototype



The second schematic detailed in figure 38 is the overview of the “breakout board” the group used for testing components individually as well as ensuring proper functionality of the WiFi capabilities of the board. It is a much simpler design, with no pullup resistors nor a voltage step-up circuit, but still ensures that the system can properly function in its entirety, and it is perfect for prototyping and testing, as the board still adheres to TI’s recommended design considerations by including capacitors to regulate current intake when powering the MCU.

6.2 PCB Vendor, Assembly, and Design Configurations

The system’s PCB design is done entirely through Autodesk Eagle®. The PCB’s components are compiled through different libraries found through Ultra Librarian®, and ordered through Texas Instruments™. The information in the following section is provided by the Texas Instruments CC3220MODx family data sheet [20].

6.2.1 Design Considerations

As with any PCB design, there are certain considerations that need to be taken into account when designing the final board. The next subsections break down the vital limits that are necessary to work into the design.

6.2.1a Power Supply Decoupling and Bulk Capacitors

It is highly recommended that the power supply drawn from the MCU be given 100µF ceramic decoupling capacitors. This will allow for peak current draw from the MCU.

6.2.1b Reset

To reset the MCU, the nReset pin must be held below 0.6V for at least 5ms. To perform this action a pull-up resistor configuration is included to pull the voltage source to the nReset and VBAT reset pins to ~0V while the button is pressed.

6.2.1c Unused Pins

It is not necessary to connect the NC pins to anything. There are built-in modules to prevent any current leakage.

6.2.2 PCB Layout Guidelines

The CC3220 MCU comes equipped with a plethora of functionalities, but this makes it necessary to consider a multitude of guidelines to avoid interference to any antennas or input pins.

6.2.2a General Layout Recommendations

The following guidelines must be followed to ensure there are not any faults with the MCU:

- A solid ground plane and ground vias under the module must be included for stable system and thermal dissipation
- Signal traces cannot be run underneath the module on a layer where the module is mounted.

6.2.2b RF Layout Recommendations

The RF section of the module must have the highest priority in the final PCB layout. The RF section must be laid out correctly to ensure optimum performance from the module. If it is not positioned correctly, there will be sensitivity and EVM degradation as well as mask violations and low-output power. RF recommendations are provided in figure 47.

Figure 39: RF Layout Requirements [21]

- RF traces must have 50 Ohm Impedance
- RF Trace bends must be made with gradual curves, and 90 degree bends must be avoided
- RF traces must not have sharp corners.
- There must be no traces or ground under the antenna section.
- RF traces must have via stitching on the ground plane beside the RF trace on both sides.
- RF traces must be as short as possible. The antenna, RF traces, and the module must be on the edge of the PCB product in consideration of the product enclosure material and proximity.

6.3 Final Software Coding Plan

In terms of priority, getting the MCU software to function as intended is highest on our list because it controls the functions of the hydroponic system. Next, we will work on the API for the Node.js server to prove we can issue commands from the internet. The mobile application is last on our list because it is a graphics user interface to interact with the hydroponics system and requires the previous parts of the project to be functional. When developing each system's software routines, an iterative approach will be used to first create small, simple, and functional programs that will be built upon and modified in future iterations. By focusing on simpler working iterations first, some form of each system's software will always exist as an operational fallback. This approach also allows for natural pauses after each iteration is completed to evaluate the current software's performance and identify weak points, organize plans for the modifications needed to complete the next iteration, and confirm what features should be prioritized for implementation next. Each section has its own Github repository for proper version control and sharing the project files between team members. The

following sections explain in more detail how we plan to work on each part of the project.

6.3.1 MCU Coding Plan

Using the CC3220MODASF Launchpad, we plan to split each component of the MCU software into separate driver files that can easily be modified to fit when porting over to the CC3220MODASM2MONR MCU. As explained before, the Sysconfig tool allows the programmer to easily configure any of the GPIO pins with aliases into a generated header file. As long as the naming conventions are kept to a standard and the pins have a valid configuration, the driver files used in the software tests will be able to be integrated into the final product. There will be two threads to work on, the main MCU which will take care of controlling the relays and sensors, and the network MCU to take care of communications with the internet hosted server and database.

6.3.2 Node.js Server and Database Coding Plan

Using the class diagrams and systems overview as a guide, we will implement each API endpoint built with Express and test that it is functional on a live, barebones application deployed on Heroku.

The MySQL database deployed on the Node.js server can be configured with the user interface panel to match the specifications. Each API endpoint will test that the database is properly storing the data.

6.3.3 Mobile Application Coding Plan

Android Studio has all the components needed to develop the mobile application. It has a phone emulator in which the project changes can be viewed in near real-time. It has processes for exporting the project to an Android phone. Our plan is to use resources available on the internet to build the draft views in this report.

7.0 Project Prototype Testing

This section covers the testing of the project prototype. The tests covered in this section are the environments of the hardware and software as well as the specific tests for each component of the hardware and software systems.

7.1 Hardware Test Environment

Due to the system requirements and overall functionality, the requirements for our testing environment will be simple and practical. Considering both the size and testing requirements of the main system, we would need a large enough room to allow proper testing. In general the main requirement would need to be a power source and space. Thus, the main requirements would be composed of:

- X2 electrical outlets providing at least 120V AC 60Hz
- Available internet connectivity
- Proper ventilation
- Space to accommodate at least 3 people, the main system, and a test bench.
- Fire prevention equipment
- Entry size large enough to let the system in and out

7.2 Hardware Specific Testing

For each element of the hardware, various testing methods will be applied to verify that all acquired parts are working properly and meet the necessary requirements specified for the system. These testing methods also include calibration for sensors and verification that they are accurate.

7.2.1 Dosing Pump Testing

To test pump performance we select the Jebao Doser 2.4 as the pump elements it uses are very similar to the individual pumps available in the market. In this test we test the strength of the pump head itself through three different types of reservoir placements that affect the negative pressure (vacuum) produced by the pump head. This test provides us with data that will help with the design of the overall system. By minimizing pump stress we will have better:

- Working lifetime
- Parts lifetime
- Energy efficiency
- Switching response

This helps our overall system's functioning lifetime as this is important due to the fact that its main function is to keep plants alive continuously thus we ought to

prevent as much downtime as possible. Once these measurements are taken we used the design that provided the fastest transfer rate as it will indicate the pump has less resistance. The testing methodology will be:

- Main the pump stationary at the center throughout testing
- Ensure all moving parts are well lubricated
- Use different set amounts of water for each configuration
- Time priming transfer rate first
- Time liquid transfer rate
- Change reservoirs and destination enclosure after each test
- Maintain distance between each tank at an average of 16.5”

Test 1: Reservoir at the bottom and destination at the top (19” apart):

Table 26: Pump Test 1 Results

	Priming Speed(s)	Transfer Speed (m)
500 (ml)	11 s	12 min
300 (ml)	10.2 s	10.4 min
250 (ml)	10 s	7.37 min
150 (ml)	10 s	3.53 min
100 (ml)	10 s	1.60 min

Test 2: Reservoir at the top and destination at the bottom (19” apart):

Table 27: Pump Test 2 Results

	Priming Speed(s)	Transfer Speed (m)
500 (ml)	8 s	11.22 min
300 (ml)	7.8 s	8.4 min
250 (ml)	11 s	5.58 min
150 (ml)	11 s	4.51 min
100 (ml)	11 s	3.44 min

Test 3: Reservoir at level with destination (14” apart):

Table 28: Pump Test 3 Results

	Priming Speed(s)	Transfer Speed (m)
500 (ml)	11 s	11.46 min
300 (ml)	11s	8.76 min
250 (ml)	11 s	6.07 min
150 (ml)	11 s	4.75 min
100 (ml)	11 s	3.46 min

From this testing we can conclude that, Test 2 - a reservoir at a higher elevation of the pump proves to be the most beneficial as it has the fastest transfer times. We can expeculates this from the added additional pressure provided by gravity. Due to this we designed our doser location in such a way that it has its reservoir tank above it.

7.2.2 LCD Power

This test is to make sure the NHD-0420D3Z-FL-GBW-V3 LCD can be powered on with the correct conditions. The NHD-0420D3Z-FL-GBW-V3 requires a 5V power supply. This test was performed using the CC3220MODASF Launchpad as a power source and a breadboard with assorted wires and pin headers. The 5V source was connected to pin 6 on the LCD, which is named VDD and the ground was connected to pin 5, which is named VSS. Jumpers were placed on the R1 and R2 resistors on the back on the LCD to put it into test mode. In test mode, two screens alternate: *Newhaven Display Firmware 3.00* and *Baud rate 9600 I2C address 0x50*. Below are two pictures taken of this test.

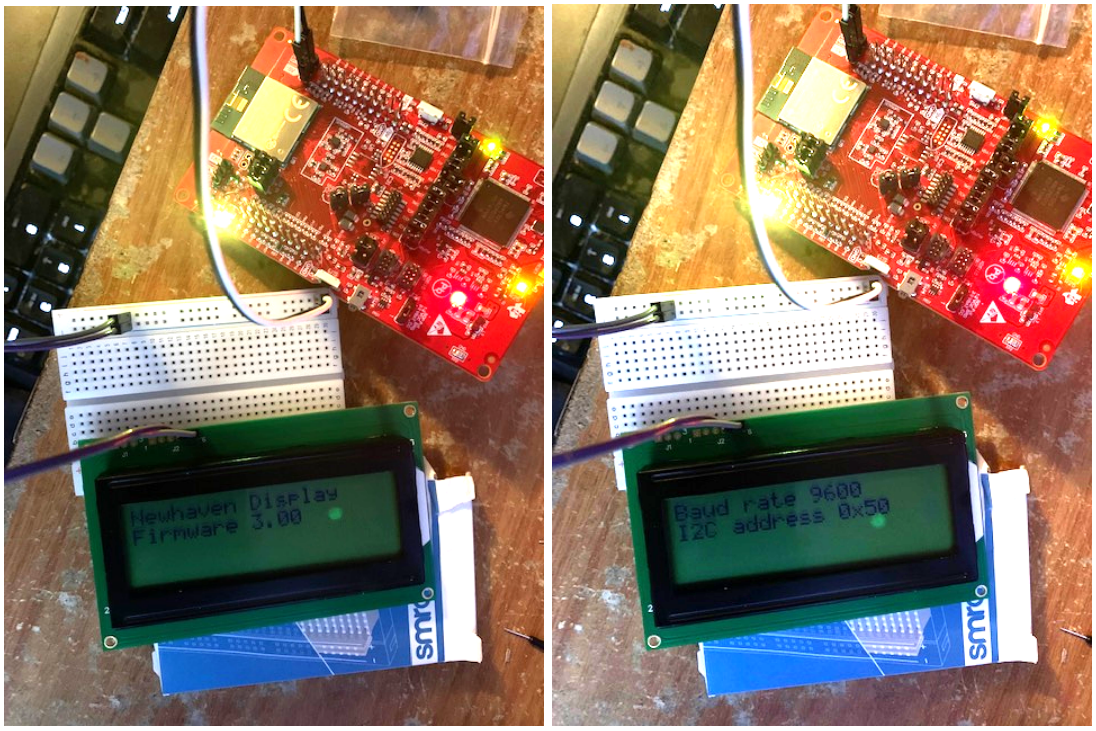


Figure 40: LCD Power Test

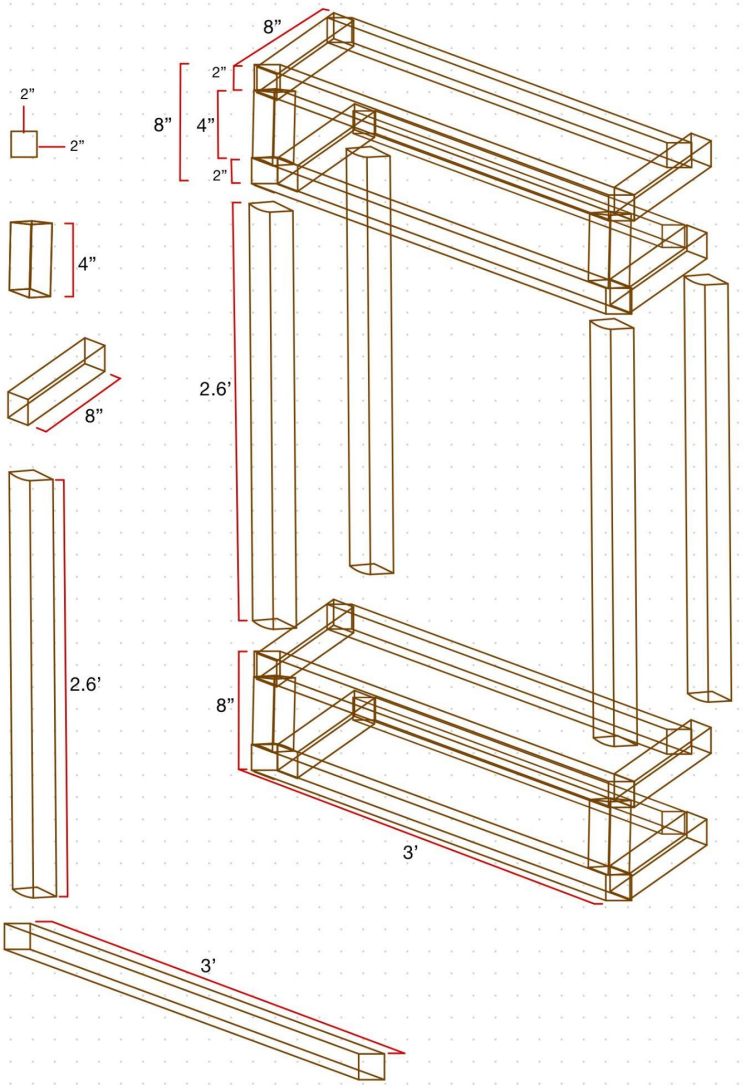
7.2.3 Power Supply

The power supply was tested for both voltage and current to make sure it meets our requirements. Since the power supply contains different voltage rails, each rail was tested for compliance. Additionally, functions of the power supply itself were tested such as electronic switching and thermal management. Due to the power supply being designed as its own component for a computer system, it had its own self managing mechanism such as thermal management and voltage linearity. The power supply proved to be practical and accurate enough for the system's needs.

7.2.4 Support and Frame

PVC was chosen for the main support frame due to cost and availability. To optimize the size and maintain construction practicality, we decided to use 2" PVC beams. This size provides enough structural integrity to sustain the top weight while providing enough room clearance for the other components. Additionally, this would need commercially available tools instead of specialized tools that would increase overall cost.

Figure 41: PVC based supporting frame



7.2.5 Water Level Sensor

To test the water level sensor, the TI MSP430FR6989 development board was used with the Optomax digital liquid level sensor hooked into one of the GPIO pins available on the board. Positive voltage line of the liquid level sensor was hooked to the 5V supply on the MSP430, while negative voltage was hooked to GND. The red user LED wired on the board was turned on when the sensor was submerged in water, and off when it wasn’t. Figures 53 and 54 show the different states of the sensor, successfully meeting the conditions described previously.

Figure 42: Water-Level Sensor in Air

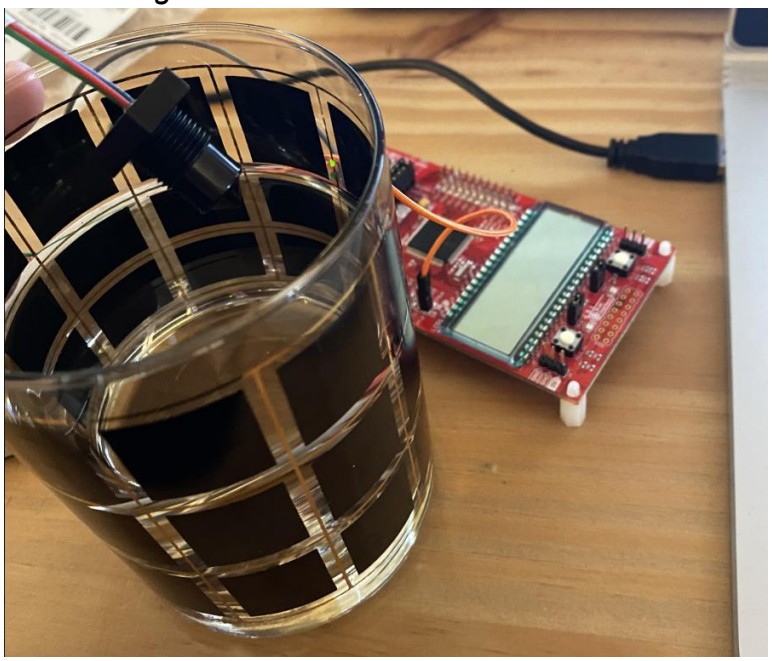
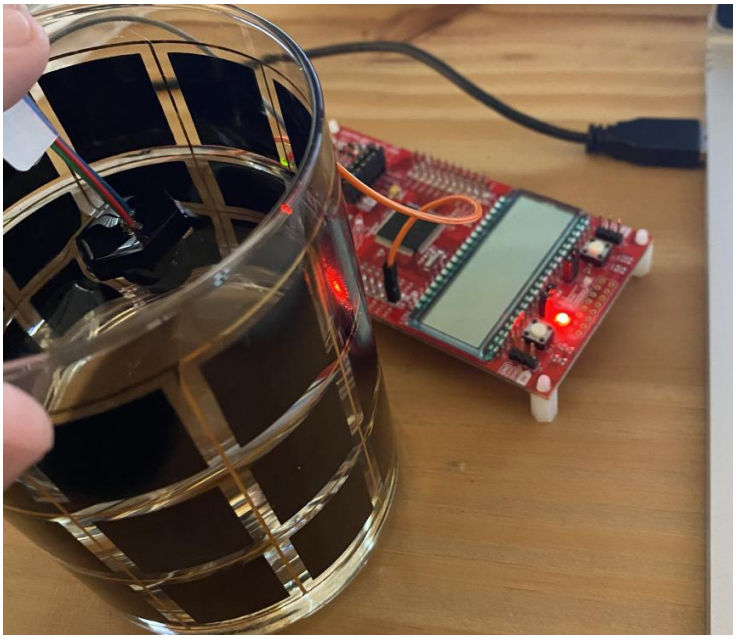


Figure 43: Water-Level Sensor in Water



7.2.6 pH Sensor Calibration and Testing

To calibrate the DFRobot Industrial Liquid pH Sensor V2, a 3-point calibration method using a linear regression calculator is used to accurately estimate pH. Three cups of water were set aside; one with a low pH, one with a basic pH, and one with a high pH. A digital pH sensor made the respective pH readings of these glasses of water and the Industrial sensor made a corresponding analog

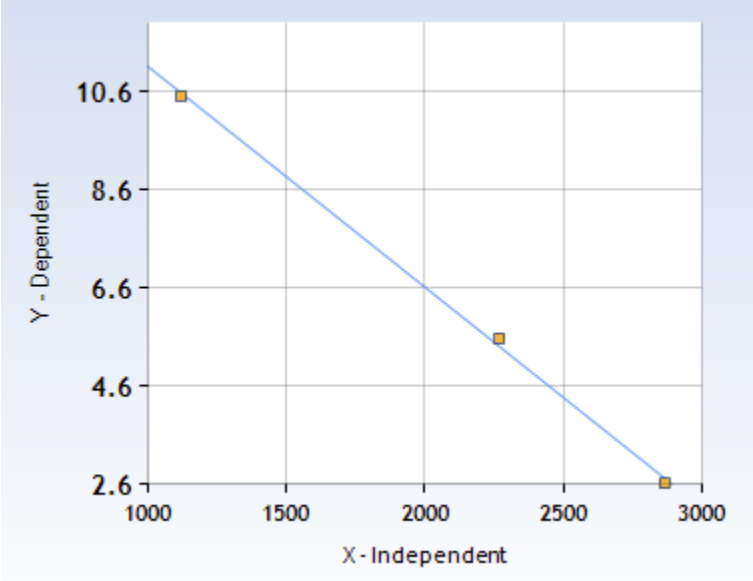
reading of the same liquid using the MSP430FR6989's digital input and an analog to digital conversion. Table 34 shows the 3 points of data taken for calibration.

Table 29: 3-Points of pH Calibration

Digital pH Reading	Analog Reading
2.62	2863
5.58	2270
10.52	1121

From here, the data was plugged into a linear regression tool [22] that calculates the formula for finding data along the linear regression line. The following graph in figure 53 was provided with pH being the Y-coordinate and the analog reading from the sensor being the X-coordinate.

Figure 44: Linear Regression of pH to Analog Reading



The equation of that line gives an accurate estimate of the pH of a liquid given the analog reading from the Industrial pH sensor. The linear regression formula for finding the pH is as follows: **pH = (-0.0045 * A) + 15.62732**, where A is the given analog reading from the industrial sensor.

7.3 Software Testing Environment

Testing software is important to knowing whether or not the written programs function as they should. As stated before, our software is split into three parts: The MCU, server, and mobile application. Each part is written in a different programming language and has different testing software.

7.3.1 MCU Software

The MCU Software is written in the C programming language using Texas Instruments' Code Composer Studio. Each test uses a Windows 10 operating system PC and a CC3220MODASF LaunchPad development kit. A breadboard with the required components for each test and the required power source is utilized.

The software that is used to program the MCU to control the Hydroponics Unit and communicate with the web/mobile application consists of many functional components and routines, each with their own testing phases. As more simple components are individually tested, developed, and made functional, new testing phases will begin interfacing multiple components together to form more complete software routines. Each testing phase is described in the following sections.

7.3.1a MCU to Mobile Application Connection

Once the basic testing mobile application is operational, setup of the testing environment for verifying the MCU's connection to the mobile application becomes possible. This environment allows for the development and testing of functions that will facilitate the flow of information from the MCU to the mobile application, with the database acting as an intermediary destination.

Using Code Composer Studio, the CC3220 LaunchPad, and Tera Term, a basic testing environment for running and observing the related code can be set up. The use of TeraTerm allows the team to view console log statements as the LaunchPad executes its program, letting the contents of variables, inputs, and outputs of functions be monitored in real time. To test whether the database is properly being updated with information from the MCU, MySQL Workbench is used for its convenience and intuitive user interface. The final destination in the MCU to mobile app connection is naturally the mobile app, which will use HTTP requests to pull information from the database. These requests are tested using both Postman and the mobile app itself, ensuring that the requests themselves are functional and that the app properly displays the information gathered from these requests to the user.

7.3.1b MCU Hosted HTTP Server

To verify the functionality of the MCU's HTTP server hosting code, a basic testing environment is set up using several software programs. Using Code Composer Studio and Tera Term, the written server program is run on the CC3220 LaunchPad during the first testing phase, then on the standalone MCU once the program is confirmed to be operational on the LaunchPad. The use of a web browser will also allow for testing access to any internal HTML pages, as well as testing any functionality these pages may have, such as software testing buttons to call internal API endpoints to control hardware systems. The internal API is also tested through the use of an API testing program, the selected program

being Postman. Calls to each endpoint are made from Postman to ensure each operates properly, using console printing functions throughout each call's handling routines to track their progress. If a call is successful, the corresponding hardware action is executed and is able to be verified by testing for changes in GPIO pin output values, which is confirmed through software by sampling the status of a pin and printing it in a console over UART and also through hardware by using a multimeter.

7.3.2 Mobile Application

The mobile application's testing environment can be separated into two different iterations: the first being the initial developmental environment where the first simple functionionality is written, tested, and built upon to form a more complete app; and the second environment where the final iteration of the app is tested.

7.3.2a Initial App's Testing Development

The testing and development environment for the mobile application began as a barebones connection of several different hardware and software layers. In a local environment, a basic Node.js server was created and placed in a newly initialized GitHub repository. The server was then loaded with installations of various software packages using the NPM package manager, which included Express to serve as the main framework for the Node application, and MySQL2 to serve as the functional library for MySQL database interactions. The main Javascript file of the Node server was then populated with some basic code that would allow for a local host to run in a web browser and display a simple "Hello World" message. After confirming the program ran successfully in a local environment, the application thus far was then deployed to a server container using Heroku's free tier hosting service.

Once the basic application was live and verified as functional in the server environment, a few configuration steps were performed within Heroku. The container was set to automatically deploy from the application's GitHub repository, which conveniently prevented the need to manually push the local app to the server whenever deployment was desired. Another task completed was the addition of the ClearDB add-on for the server container, which is a popular database service supported by Heroku that is capable of creating MySQL databases. ClearDB was selected to function as the database service for its ease of use and its inclusion of a free tier of service.

Using MySQL Workbench, a basic MySQL table was created in the provided ClearDB database. The table's purpose was only to test database communication from the web/mobile application and the MCU, therefore it was simple in its construction. Only two columns were created, one with a "message_id" variable as a primary key and the other with a "message_content" variable, which would contain a short string. Several entries containing messages were manually added to the database using MySQL Workbench's query functionality, messages that stated where they were inserted from.

On the web application side, a few basic API endpoints were written to perform GET and POST requests to the database, each with their own Express callback routes. After adding the database access credentials to a secure envelope file and configuring the API calls to use them in an equally secure manner, a basic test using a GET request was performed to confirm that the application was able to communicate with the database successfully. This was verified by displaying the results of the GET request in the console. At this point, no front end framework had been set up outside of the existing Node and Express framework, so the next step was to add Flutter to the application, which would complete the assembly of the MEFN stack. Using Flutter's Dart programming language, a simple user interface was written to allow for the making of API calls and displaying their results in the browser. The application thus far was confirmed to function properly in a local environment, successfully being able to retrieve and present the contents of the database message entries. The Flutter framework was then built for deployment, and the application was pushed to the live Heroku server where it was once more confirmed to be operational. The basic web/mobile testing application framework was considered set up at this point, and was ready for further development and testing.

7.3.2b Final App's Testing Environment

After the mobile application's development is completed, the final testing environment can be set up. This environment uses the same programs as the initial testing app's environment, but is used to test more complete processes and systems rather than basic functions in their infancy. For example, the initial environment may test API calls for successful basic execution, but the final environment includes tests for security and full functionality. This environment uses the following tools for testing: MySQL Workbench to verify proper database functionality, Postman to test the API for both functionality and security, Visual Studio Code to run Flutter's front-end framework in an emulated environment, and an Android device with an installation of the mobile app to test the app's functionality on its intended platform.

7.4 Software Specific Testing

Software tests are necessary to ensure the system's software behaves as intended. There are three systems to test: the MCU software, the mobile application, and the server. Each section has its own components with its own procedures to test.

7.4.1 MCU Tests

These tests determine whether or not the MCU software functions as intended. The tests are conducted on the CC3220MODASF Launchpad development kit, the CC3220MODASM2MONR breakout board, bread board with the associated components, and debugged with Code Composer Studio on a Windows 10 operating system.

7.4.1a Peripherals Connected to Relays

This driver program includes the lights, water pump, water mixer, and dosing pumps. This test will test the functionality of turning on these peripheral devices via the MCU. A driver program toggles the pin high and low every five seconds. If each device toggled also turns on and off at the same rate, the test will be a success.

7.4.1b LCD Screen

This driver program will test the functionality of the LCD Screen. Providing a 5V power source to the VDD pin, a greeting message will display on the screen. The driver for this program will show the test layout for sensor data and other required information below. If the information shown matches the test layout, the test will be a success.

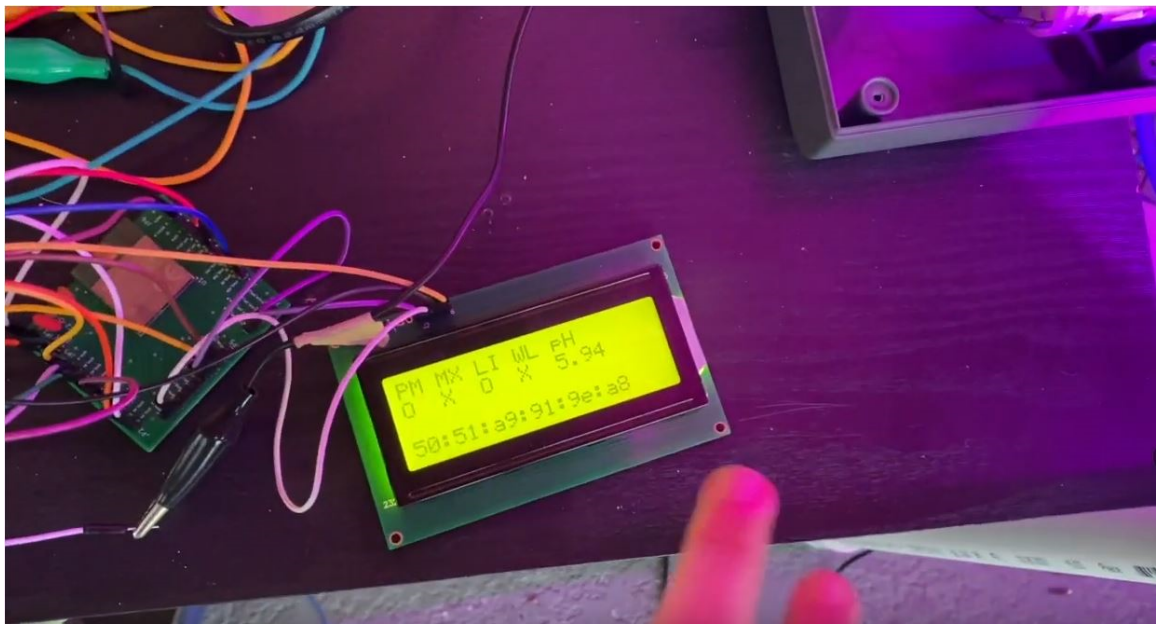


Figure 45: Example LCD Output

7.4.1c pH Balancing

This driver program will test the ability of the hydroponic system to automatically balance the water being fed to the plants. The test will require a functional pH sensor, dosing pumps, and a water source. Tap water has a pH content of 6.5 to 8.5. The driver program will make an initial measurement of the tap water. We will set the desired pH to be above 8.5, wait 60 seconds, and take a second measurement. If the pH went up but did not reach 8.5, apply doses of pH balancing solution until it gets within 0.1 of the desired level. Then the test will repeat for lowering the pH to 7. If the pH is within 0.1 of the desired level, the driver program will be considered a success.

7.4.1d Pairing MCU to Specific Mobile Application User

Each CC3220 has its own MAC address stored in a specific address in FLASH memory. When the user signs up in the mobile application, they will need this information to tie the user's account to their specific hydroponic system. To test the desired behavior, we can use the CC3220 Development Kit's number in the test account. Upon creation, we can display that the device has been successfully paired by displaying a success message on the LCD screen.

7.4.1e Connecting the MCU to Wi-Fi

After completing the provisioning process for the CC3220, which specifies the access credentials for the Wi-Fi network the MCU will connect to, the MCU's connection to the wireless network can be tested. The test is fairly simple, as it involves calling existing functions provided by Texas Instruments' SimpleLink™ libraries to initiate the connection and verify whether the connection is successful. Through a UART connection to a computer terminal, the results of each step of the Wi-Fi connection process can be displayed. Upon successfully connecting to a nearby wireless network, the terminal will display a success message as well as the acquired IP address for the MCU. If the connection process fails, an error message will be displayed instead.

7.4.1f Uploading Sensor Data to Database

To verify that the MCU correctly uploads sensor data to the database, software outside of Code Composer Studio will need to be used to access the tables in the database. One of such programs able to do this is MySQL Workbench, which allows for SQL queries to be made within its easy to use GUI.

Being that the sensor data uploading process occurs in three main sections, three testing phases are required. The first section involves the making of a POST request from the MCU to the database, attempting to deposit the sensor data payload into the correct table. For the execution of the POST request to be considered successful, several smaller tasks must be performed correctly as well, beginning with a successful connection to a Wi-Fi network. The testing procedure for ensuring the MCU properly connects to a wireless network may be seen in Section 7.4.1e. The MCU must properly format the payload sensor data into a JSON package that the database will accept. This step may be tested by comparing the JSON package used to make a successful API call from an API testing program, such as Postman, to the printed out JSON package being used by the MCU's program. So long as the testing program's API call is successful when using the same JSON package formatting as the MCU's program, it can be confirmed that the formatting is correct. The MCU must also properly call the Node server's API endpoint necessary to make the POST request, which involves obtaining the valid URI to reference the API endpoint. Testing this task is done much like the previous task, which involves first making a successful API call from an API testing program using a particular URI to an API endpoint in the Node server. By then comparing the URI used in the testing program's successful

call to the URI being used by the MCU's software, which will be displayed in the terminal, it can be confirmed that the URI is properly formed if they match.

The second testing section for the sensor data upload process verifies whether or not the fulfillment of the API call made from the MCU is successful. As with the first section, the fulfillment process involves several steps. The API call from the MCU must be received by the Node server and be handled by the proper API endpoint. This task can be verified by inserting console log statements in the Node server's code to visually display when requests have been received and when certain API endpoints have been called. Another beneficial use of console logs during this task is to display what payload information has been received from the MCU, allowing for confirmation that the sensor data correctly reached the Node server for further processing. After the correct API endpoint on the Node server has been called, the proper use of SQL queries to deposit the sensor data payload into the database must be accomplished. By using MySQL Workbench or a command line interface once more to query the contents of the database, it can be verified whether the sensor data successfully was deposited into the proper table.

The final task to be tested is the HTTP response handling, which must account for successful and erroneous requests and return the proper status codes. Testing for this task is accomplished by creating test cases that simulate conditions for each type of request response. If any test case fails, the request handling code must be modified and troubleshooted in order to solve the error. After passing all test cases, only then can the HTTP response handling procedure be considered correct.

Once all testing phases are complete, verifying that: the sensor data is correctly being sent through a properly formatted POST request, the request is being fulfilled correctly by the Node server, and the HTTP response is handled correctly; the process of uploading sensor data to the database may be considered fully tested.

7.4.2 Mobile Application

In the following section, the testing procedures used during the mobile application's development will be described. Being that the mobile app consists of several software layers, each layer will require testing in order to ensure that the entire application functions properly. These layers include each layer of the selected MEFN software stack. The communication between the mobile app and the Hydroponic Unit's MCU must be tested as well, as hardware commands will be sent to the MCU without the use of a database. Thus, the mobile app's software components that must be tested include: the MySQL database, the Express framework, Flutter's front-end framework, the Node.js server itself, and the communication between the app and the MCU.

7.4.2a Sending Mobile Application Commands to Unit

The mobile application's ability to change the hardware settings of the Hydroponic Unit through commands will be tested in several steps, the first of which involves testing the HTTP server hosted on the MCU that will be receiving the hardware commands. The server must first be confirmed to be active, which can be achieved by having the MCU serve a simple HTML page while the server is active. If this page is able to be visited in a web browser without issue, then the server is being hosted successfully.

Once the server is verified to be live, the MCU's internal API must be tested. Using API testing software, such as Postman, the API's functionality can be tested by calling a specific endpoint by its corresponding URI. Each API endpoint's purpose is to control a particular hardware system of the Unit, resulting in either toggling a system on or off by supplying a voltage to a relay, or adjusting target values for the system to automatically balance around. Thus, by calling an endpoint through testing software and observing the correct change in the hardware's state, the API can be tested successfully. Also, the use of console logs throughout the API's code is a useful technique to help verify correct functionality and troubleshoot.

The third phase of testing involves ensuring that the mobile application is able to properly send hardware commands to the MCU's server through HTTP requests. One method to help achieve this is comparing the URIs and headers of successful API testing software requests to the URIs and headers used in the mobile application. If both URIs and headers match, then the mobile application should have no errors when calling the MCU's API endpoints. Another way of testing the mobile application's sending of hardware commands is to use console logs in the MCU's server coding to display the contents of each request in a terminal. Inserting console log statements throughout the processing path of each HTTP request will also allow for the tracking of each request's progress as it is handled. If all tests up to this point are passed, the MCU should be able to correctly fulfill the hardware request from the mobile application and execute the proper changes to the hardware. After the command is executed, the state of each hardware system listed in the mobile application should update to match the actual state of the hardware.

After all phases of testing have been completed and passed, the process of sending hardware commands from the mobile application to the Hydroponics unit may be considered fully tested and functional.

7.4.2b Database

Testing the MySQL database begins with ensuring the setup process using ClearDB, the selected hosting service for the database, and integrating it into Heroku is successful. This procedure is simple, as a successful setup will result in the creation of a free "Ignite" tier database that is visible and accessed through the Heroku container manager. If this database cannot be located or accessed

through the container manager, then the setup process was not performed correctly and should be redone.

The ability to access and modify the database should be tested as well, either using a command line interface or MySQL database managing software, such as MySQL Workbench, to do so. For its ease of use and GUI, MySQL Workbench is used instead of a CLI to carry out the database testing procedures. Using the given credentials, which are provided during the setup process, a connection to the database should be initiated by setting up a new connection within MySQL Workbench. Conveniently, this software includes a connection testing feature to confirm whether the access credentials are correct. After performing the connection test successfully and continuing to log in, the ClearDB database that was created in the initial setup process should be visible in MySQL Workbench's GUI as an active connection. Various other database details can be viewed within the GUI as well, such as whether the ClearDB MySQL server is active and how much traffic it is handling. These details are useful to confirm that there are no issues with the database hosting service itself in the event the database must be troubleshooted.

Now that the database access has been tested, several SQL queries should be performed to ensure that modifications to the database are possible. Creating a table with some basic columns with the "CREATE TABLE" command should result in said table being displayed when a "SHOW TABLES" command is executed. In similar fashion, other CRUD operations should be verified as functional using queries to modify the created table. Each operation is able to be verified by performing other queries that return the contents of the table, where the changes made by each query should be visible.

After confirming the database is operational, accessible, and able to be modified through the discussed testing procedures, the database may be considered fully tested and ready for integration with other software components.

7.4.2c Account Creation & Management

The user's ability to create and manage their mobile application account should be thoroughly tested in order to ensure these features function properly and contribute positively to the user's app experience. In the following procedures, all app functions related to account management are tested. These functions to be tested include: account creation, email verification, login, updating account credentials, and account deletion.

After installing the app on an Android device, a new account should be created by following the on-screen prompts and filling out the forms for the user's credentials. Submitting the forms should result in a new user being added to the user's table in the database with an unverified status variable, a process which can be verified through the use of SQL management software such as MySQL Workbench. If a new user with the correct credentials has been added to the

database, the app should display a success message and prompt the user to check their email to verify their account. If unsuccessful, the database should remain unmodified and the app should display the proper error message to the user, prompting them to try creating their account again with the proper form contents.

Upon creating a new account, an automated email should be sent to the specified user's email address with a link to verify their account. Using either a dummy or a personal email during the test account creation process allow for the team to check if this email is correctly sent, formatted, and contains a working link for account verification. Clicking this link should, without any additional input from the user, update the user's account status in the database as verified and redirect them to a page with a "Successfully Verified!" message. If verification fails, the user's account entry should remain unmodified, and the user should be redirected to a page with an error message.

Logging in is a simple task to test, as it only requires entering an account's credentials on the login page, submitting them, and confirming that each possible result is handled correctly. If the correct login information is submitted, the user should be granted access to the app's features and be redirected to a home page. If the incorrect information is submitted, an appropriate error message should be displayed. Also, if an unverified account's login information is submitted, an error message stating that the account has not yet been verified should be displayed. The case where two users attempt to log in with the same account information simultaneously on two separate devices should be tested as well. Only one instance of each account should be allowed to be logged in at a time, thus a successful handling of this case would have the app log out the currently logged in account when attempting to log in with a second instance. Testing a user's ability to update their account information is another simple task, only requiring the database to correctly update their account's entry with the user's new information. The usual success and error messages apply here as well. Account deletion may be tested by verifying that, after confirming with the user that they wish to have their account deleted, the database no longer contains the entry for that user's account. The user should also no longer be able to login with this account's credentials if this process has been correctly performed.

7.4.2d Security

In order to test the mobile application's security features, the app's framework should be tested against multiple simulated attacks of many different types from within a safe environment before it contains any sensitive information. The main areas of security to be tested are: password protection, SQL injection attack prevention, and the prevention of unauthorized hardware commands.

Testing how securely passwords are handled, ensuring that they are realistically unable to be obtained in the event of a security breach, is accomplished through

several smaller tests. Verifying that the password hashing and salting algorithms are properly producing unique and sufficiently complicated strings is the first test. One case that demonstrates whether this test is passed is by creating two user accounts with the same password. If the password algorithms are functioning properly, each user account should have unique passwords generated from the same strings. Using packet sniffing software, such as Wireshark, as developers is another effective way of testing if the app's password protection routines have vulnerabilities. Through this software, which intercepts network packets and allows them to be inspected, the team can test whether any passwords have been properly obscured or not when being sent over the network. The point at which passwords are hashed should be tested in order to ensure that the hashing process is not simply replacing the user's password with a hash value, which would defeat the purpose of hashing to begin with. The password should be hashed server-side in order to benefit security. To test when hashing is occurring, console log statements can be used in both the client and server codes to print the current value of a password as it is processed by both environments. If the password is being hashed at the correct time, only the server-side console logs will display a hashed value while the client-side logs show the original password's value.
In general, all error messages, aside from ones that provide necessary feedback to the user, should be hidden from the user for security purposes. Testing whether each error message is hidden or displayed properly can be done by simulating use cases that throw these errors. If all errors intended for the user are shown and all others are hidden, these tests are then considered successful.

Performing different types of SQL injection attacks on mock database tables allows for testing how well the app is able to prevent unintended and malicious queries from being executed as code. For each test performed that fails, resulting in injected code being executed, modifications to the software routines that interact with the database must be performed until the test successfully passes and no injected code is executed.

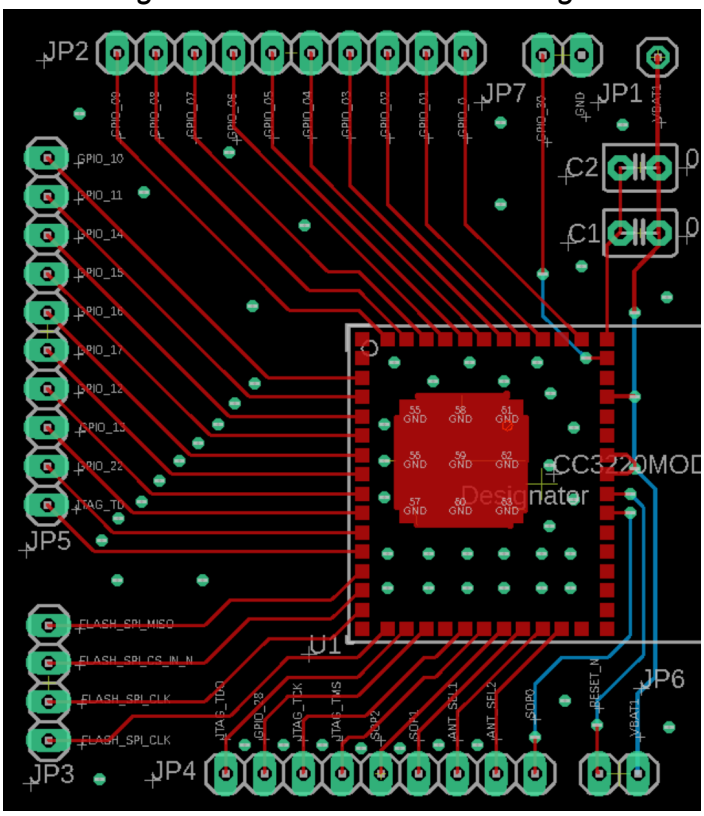
In order to test how well the software prevents unauthorized hardware commands, simulated malicious calls to the API can be made through API testing software. If the calls are correctly refused, such as when calls without authorization or authentication are made, then the test has been successfully passed. Stress testing the API should be done as well in order to determine how well the software can limit the rate of calls made by each user, as very high numbers of calls indicate suspicious activity. This test is passed if high volumes of calls, above a defined rate, are refused in the proper manner.

7.5 PCB Testing

For the prototype testing of this project, the team decided to make a preliminary board for testing components individually and WiFi connectivity. This board is

what is known as a “breakout board” and consisted of only our CC3220MODA MCU, capacitors to regulate current intake when powering the board, and pin headers for each pin. Figure 44 shows the Breakout Board’s design.

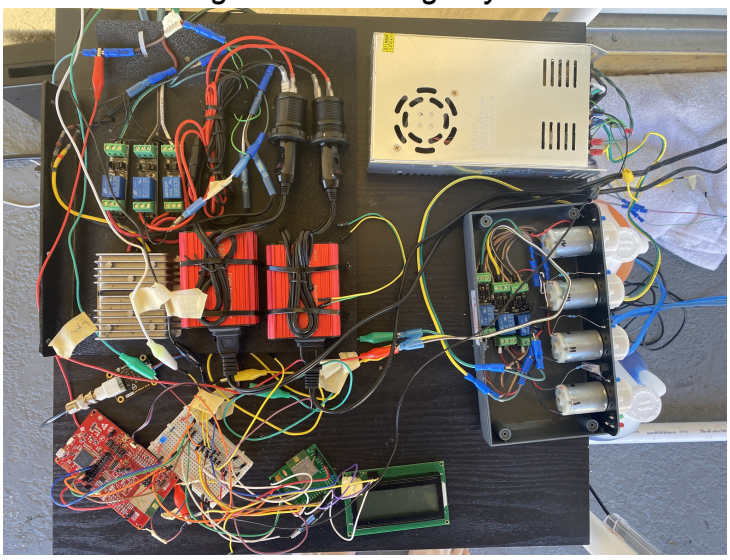
Figure 46: Breakout Board Design



7.5.1 PCB Testing with Power

To ensure that each component was functional and compatible with the CC series MCU selected for our project, programs were flashed over UART connection through a CC3220MODASF Development Kit using Code Composer Studio by Texas Instruments. Despite some issues programming the board early on in development, this PCB proved to be a reliable device for testing and development throughout the project. Figure 58 shows the setup of the breakout board connected to all components through breadboard connections when used for testing the system in its entirety. The aforementioned development kit is also connected to the correct UART receiving and transmitting pins

Figure 47: Testing Layout



7.5.2 Requirements Testing

For each demonstrable requirement in the requirements table located in figure 1, the requirement was tested and demonstrated on the breakout board: pH Accuracy, Response time of Hardware Commands, and time to post sensor data to the database from the microcontroller.

7.5.2.a pH Accuracy

As a baseline for measuring pH accuracy, a store bought sensor was used to compare pH readings from the DFRobot SEN0169-V2 Industrial Sensor. When testing the nutrient reservoir with the sensor used as a baseline, the pH reading was 4.9. When running 10 pH readings of the same reservoir using the SEN0169-V2, the results averaged only a 0.97% error margin. The results of these tests are shown in Table 34 below.

Table 30: pH Accuracy

pH
4.85
4.80
4.95
4.85
4.88
4.92
4.87

4.83
4.79
4.78
Average = **4.852**; Error Margin = 100*(4.9-4.852)/4.9 = **0.97%**

7.5.2.b Sensor Data Posting to Database

To ensure that sensor data from the microcontroller is posting in a reasonable amount of time, the group set a goal of achieving system data posting within 1 second of reading device statuses. To measure the time elapsed, the time difference between reading sensor data and receiving an HTTP response of 200 from the backend server was taken and printed on a UART terminal.

On the UART terminal, sensor data is included along with system component statuses for the lights, mixer, and pump. For the purposes of the test the system was configured to take sensor readings every ten seconds and send them to the database. The mobile application was referenced during the testing so that the data can be verified on the client-end once the screen is refreshed. To verify that the sensor data postings were reliably meeting system standards, this test was run ten times in a row and their response times were averaged. Table 35 displays these results and their average response time was well within the goal of 1 second set in the requirements table in Figure 1.

Table 31: Posting Sensor Data Response Time

Response Time (Seconds)
0.3513
0.1323
0.1429
0.1545
0.2608
0.2564
0.2366
0.2354
0.2527

0.2602
Average = 0.2636 Seconds

7.5.2.c Hardware Command Response Time

In a similar fashion as the group tested sensor posting response time, the system was expected to be able to send commands to the microcontroller from the mobile application and receive a change in hardware status within one second. To do this, the mobile application recorded pressing switches to alter system lights, while video software recorded the time between a button being pushed and lights altering state. A table of these recorded events is included in table 36.

Table 32: Hardware Command Response Time

Start Time (m)	Start Time (s)	End Time (m)	End Time (s)	Elapsed Time (m)	Elapsed Time (s)
10	16.469	10	16.72	0	0.251
10	18.93	10	19.231	0	0.301
10	21.642	10	21.968	0	0.326
10	24.303	10	24.856	0	0.553
10	26.965	10	27.116	0	0.151
10	29.401	10	29.601	0	0.2
10	31.786	10	32.238	0	0.452
10	34.347	10	34.498	0	0.151
10	36.708	10	37.185	0	0.477
10	39.445	10	39.992	0	0.547
			Average Response Time:		0.3409

7.6 Facilities and Equipment

While working on the system's development, the group used a couple different facilities: the senior design lab located in the Engineering 1 building on the UCF campus, and the back patio at a group member's home. In the initial stages of development, the group utilized the senior design lab heavily, because of its easy access to soldering equipment, multimeters, and oscilloscopes. Once the PCB designs were both soldered and tested for crossed connections in the senior design lab, they were brought back to the location of the constructed hydroponic towers to be hooked up into the system's power and relays.

The patio that the group spent a majority of its time working on the project was used because the frame built could not be moved easily, and it needed a semi-permanent home while its software was being developed. This outdoor area allowed for the group to work with less worry of water damage on anything, as we sometimes experience water leaks or spills when testing flow throughout the frame.

The equipment used to construct the system was limited to what was readily available to the group; soldering irons, digital multimeters, a tape measure, and a jigsaw were all that was required in the construction of the frame and soldering of parts to the PCBs. Proper safety precautions like wearing eyeglasses and close-toed shoes during construction were taken to ensure no group members were put at risk during development.

8.0 Administrative Content

The contents of this chapter are related to our expected rate of work and the budget we have allocated for ourselves.

8.1 Milestones

These milestones are the expectations we have set for ourselves to complete this project. By August 3, 2021 we expected to have completed all design sketches and schematics related to this project as well as tested each individual component to see that they are functional. Unfortunately due to the time it took to troubleshoot some LCD and pH sensor problems, we did not meet the October 4th deadline to have our second prototype finished. The second prototype was finished on November 5th. The final deadline was met when we did the final tests on November 22nd.

Table 33: Project Milestones

Date	**Description**
8/3	- Complete design sketch and project requirements. - Calibrate sensors and test individual components.
8/10	- Finish PCB prototype and Breakout board
8/17	- Complete interfacing MCU with relay peripherals (water pump, fan, etc)
8/21	- Acquire all components and sensors - Begin construction of 1st prototype
8/24	- Complete LCD software - Finish PCB testing and rework into second prototype
8/31	- Complete pH balance control loop software - Complete relay system prototype
9/7	- Complete sending MCU data to external database
9/14	- Start mobile application
9/21	- Complete login, signup, and sensor data view
9/28	- Complete sending commands from mobile application to MCU - Complete graph view - Add Relays to each doser, recirculation pump, and mixing fan
10/1	- Complete 1st prototype design with minimal flaws

	- Have database models set up and connected to prototype
10/4	- Second prototype PCB testing completed - Rework PCB into 3rd and final product if necessary
10/11	- Run complete system tests and begin full system debug
10/20	- Complete quality assurance and testing
10/25	- Add software security features
11/1	- Continue calibrating pH rebalancing process for higher accuracy
11/8	- Solder all parts together for final product
11/15	- Calibrate system sensors a final time
11/22	- Complete final project iteration

8.2 Budget Analysis

The project is self-funded with a target budget of no more than $1,000. The budget for our project is as follows:

Table 34: Budget Analysis

Item	**Quantity**	**Price Estimate**
PCB	1	$30 - $80
System Controller & Development Board	1	$70
Power Supply	1	$20 - $30
Water Pump	1	$25 - $40
PH Sensor	1	$80
Water Level Sensor	2	$50
LED Grow Lights	2-4	$30 - $80
Dosers	4	$60
Underwater Mixer	1	$10 - $20

Display Panel	1	$15
PVC Frame & Piping	Variable	$50 - $150
Water Tanks	2	$30 - $60
Plants/Seeds	Variable	$5 - $10
Nutrient Solutions	2	$20 - $50
PH Control Solutions	2	$20 - $50
Enclosure Paneling	Variable	$20 - $100
Miscellaneous	Variable	$50 - $150
Total Cost		$595 - $1120

8.3 Bill of Materials

This section includes the final bill of materials to complete this project. This list does not include shipping costs or sales tax.

Table 35: Bill of Materials

Item	Part Name	Quantity	Cost
Printed Circuit Board	PCB	2-3	$30.00
System Controller	CC3220M2MONR	2	$18.094
Development Board	CC3220MODASF LaunchPad	1	$60.00
3V Relay Power Switch Board	ICSTATION 1CH-DC3V	6	$36.00
Power Supply	ALITOVE B06XJVYDDW	1	$25.50
3-Prong Power Cord	Middleway B07MCBPG5V	1	$12.99
22 AWG Wire	BNTECHGO 22 Gauge Silicone wire	1	$9.06
Alligator Clips	WGGE WG-026 Alligator Clips	1 Pack	$7.00

Power Inverter	UEIUA 150W	2	$35.58
12V Socket Power Outlet	ZHSMS CL-S-101	2	$10.29
Step-Up Converter	Dkplnt 10A 240W 12V to 24V	1	$19.20
Step-Down	eBoot Mini MP1584EN 12V to 5V	3	$9.99
Water Pump	Vivosun 800 GPH	1	$26.00
PH Sensor	DFRobot Pro V2 Industrial Liquid pH Sensor	1	$86.00
Water Level Sensor	Optomax LLC2003DSH	1	$25.00
LED Grow Lights	Lightimetunnel 4 Pack	1	$36.62
Dosers	Jebao Doser 2.4	1	$92.00
Silicon Tubing	Deep Blue Professional ADB12296	1	$12.99
Underwater Mixer	SunSun JVP-101A	1	$10.99
Display Panel	NHD-0420D3Z-FL-GBW-V3	1	$22.45
PVC Frame & Piping	various	27 pcs	$300.60
Zip Ties	Bolt Dropper Zip Ties	1 Pack	$10.00
Water Tanks	5-Gallon Bucket	1	$5.00
Plants/Seeds	Spinach Seeds	1 Pack	$4.50
Nutrient Solutions	(not acquired)		(not acquired)
PH Control Solutions	pH up/down	16 oz	$18.00
Enclosure Paneling	Acrylic sheet	1	$14.40
Miscellaneous Components	MCU plastic enclosure	1	$7.00
Total			$1,270.25

8.4 User Manual

This section explains how to use the Hydroponic Unit. The controls are split in two parts: the controls on the physical unit and the mobile application.

8.4.1 Hardware Manual

This system aims to have a user-friendly hardware application; turn the system on or off, and give the user a code to pair with the system.

8.4.1a System Power-On Procedures

Before powering on the system the user must assure:

- An a available outlet that provides 120V AC 60Hz
- A clear line of sight between the system and the controlling device to assure proper communication
- Look for any cracks or structural integrity deficiencies within the tanks and overall support frame.

After making sure everything looks safe and proper to operate the user can now:

1) Connect the power plug of the device to the wall outlet
2) Switch ON the main power switch
3) Verify power-on LED from the MCU is ON
4) Turn ON the MCU via the switch power switch
5) Verify from the LCD that the MCU is functioning
    a) In the event of a blank screen, check the connection from LCD to the MCU pinout.

8.4.1b Plant Support Procedures

The system's main focus is to support the required needs for the type of plants used. Since the main goal of the system is flexibility, the system is able to adapt itself through a combination of user settings and automated functions in order to provide the necessary nutrients in a cyclical timed manner. To achieve this, we have separate sub-systems that work in tandem with each other controlled by the main MCU. Hence in order to operate the device the user will:

1) Obtain information about the plant’s specific nutrient and sun exposure needs
2) Load the doser tanks with the necessary nutrients
3) Load the lower tank to ~80% capacity with fresh water
4) Test communication between the system and software
5) Input the proper parameter in the software
6) Run the system

8.4.1c LCD Indicators

Even though the system is mainly monitored through software, an LCD is implemented on the system itself to quickly check certain functioning parameters. The LCD shows:

- The sensor labels
- The sensor data
- System status
- MAC address to use to pair the mobile application and server to the hydroponics system.

Additionally the user can quickly verify certain function statuses of the system via The LCD. The third line of the LCD displays the status of the system. Some examples are:

- Main power detection
- Communication status
- Pump status
- Light status
- Wifi connection
- MCU self-check

8.4.2 Software Manual

This section explains how to use the Hydroponics System's software. It is split into two parts, the MCU software that pairs the MCU to a Wi-Fi connection and the mobile application.

8.4.2a Pair MCU to Wi-Fi connection

1. When the device powers up, the WiFi credentials encoded in the program obtains the MAC address for the MCU.
    a. If there is an error connecting to WiFi, make sure the credentials are correct and restart the system.
2. The MAC address is displayed on the fourth line of the LCD.

8.4.2b How to use the Mobile Application

1. Launch mobile application on Android phone
2. Signup with a username, password, and MAC address displayed on LCD
    a. If there is an error in signing up, make sure the phone is connected to the internet and try again.
3. Login with signup credentials.
    a. If there is an error stating the user does not exist when logging in, make sure the credentials are correct and try again.
    b. If there is a general error, try again in a few seconds. This error occurs because the server needs time to come online after being idle.

4. When logged in, use the navigation bar at the bottom to switch between the status, graphs, and settings views.
    a. In the event the status view is empty, please wait for the MCU to send its first round of sensor data to the server.
5. The Status view displays the last known entry of the hydroponic system's status in the database.
6. The Settings view shows the current settings. These settings can be changed by clicking the toggle buttons or changing the pH value with a slider. Clicking save saves the settings in the database. Clicking the system pause button turns off all the peripherals and enters a dormant state. Clicking the system resume button returns to the previous state.
7. The Graphs view shows the hydroponic system's time-series data for pH, water level, lights, mixing fan, and water pump.

8.5 Consultants, Subcontractors, and Suppliers

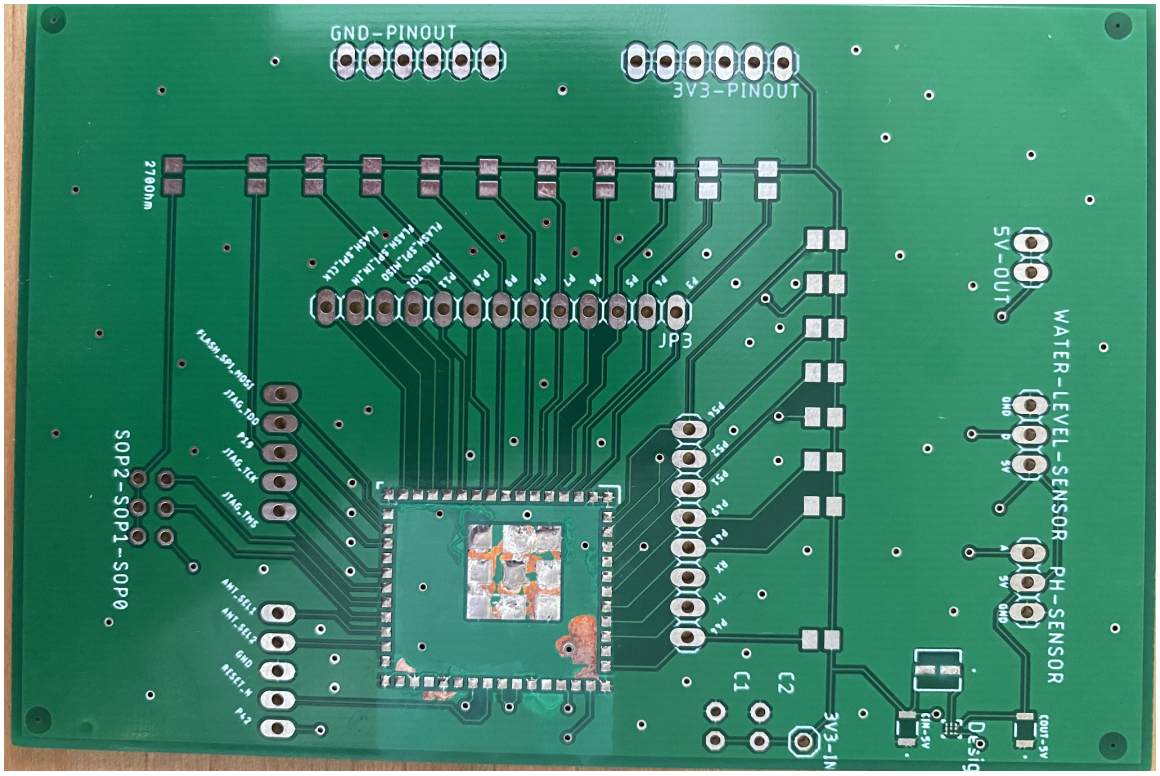
The project team consulted with several different parties throughout the project's development. One such consultant was Dr. Richie, who we asked on several occasions for guidance in solving some of our challenges. In particular, Dr. Richie aided us in the initial PCB manufacturing process, recommending online board manufacturers that have treated previous students well and providing us with a social connection to Quality Manufacturing Services, a local electronics assembly company that was kind enough to assemble our first PCB without charge.

Another valuable resource we consulted with many times was the Texas Instruments E2E forum, in which we posted many questions to and received quality responses from. This resource resolved much of our initial confusion about the proper programming and flashing process of our CC3220MOD MCU, as well as addressing some concerns about its internal hardware.

Very few subcontractors were used during the project's development, one of which being Quality Manufacturing Services, who assembled our first PCB, and the other being JLCPCB, who manufactured our first and second PCBs. Unfortunately, we encountered an issue with our second PCB assembly, which was also performed by Quality Manufacturing Services. The second PCB produced by them appeared to be successfully completed, but we were unable to flash any software to the MCU, leading us to believe the MCU was damaged in the assembly process. QMS had attempted one mounting of our MCU onto a board, but was unsuccessful in their first attempt and had to remove the soldered MCU from the PCB it was mounted on. During this removal process, heat is used to remelt the solder beneath the MCU to remove it from the board. However, the remelting point of solder is higher than the temperature used to perform the initial soldering. Due to this fact, we believe that too high of a temperature was used to remelt the solder and extract the MCU from the unsuccessful first board, which would damage the MCU internally. It was also our misfortune that both sets of

parts used in our step-up circuit were wasted during this process, leaving us without a successful final PCB and no parts available to attempt another assembly, as online suppliers were out of stock of these parts. The PCB that QMS originally has soldered the MCU onto can be seen in Figure XXXX.

Figure 48: Damaged Final PCB



Our team used several suppliers for both our construction materials and hardware components. For our MCUs and development boards, we ordered these directly from Texas Instruments, who designed them. Our PCB components were ordered from Digikey, and the board itself was manufactured by JLCPCB, who was recommended to us by Dr. Richie. From Amazon, a few minor tools and supplies were ordered, which included an electronic pH tester and pH adjustment solutions. The pH solutions were consumed quickly, however, and the pH tester became unreliable after several weeks of use. Our second and much more reliable electronic pH tester came from Root Grow Bloom Organic & Hydroponic Gardening Center, who we also purchased more pH adjustment solutions from. The pH sensor used in the project was purchased from and manufactured by DFRobot, and our physical construction materials (PVC, water tubing) were purchased from Lowes.

9.0 Conclusion

Much of the inspiration for the project came from the team's personal interest in growing plants in a home environment, as well as the realization that the field of hydroponics allows for the marrying of these interests to the fields of computer and electrical engineering. This made for the creation of a project idea that each team member felt passionate about and motivated to successfully implement, while also serving as an excellent way to demonstrate our knowledge, experience, and research ability.

After performing market research, we felt that the current market of available hydroponic systems was lacking in variety and user functionality. Most units required the manual adjustment of nutrition content and pH levels, were restricted to either tower or rack formats, and lacked software connectivity. By designing a hydroponics unit for in-home use with automated maintenance systems, a mobile application to control and monitor the unit, and a unique physical format, the project team believes this product is able to offer an attractive alternative to the current available units and fills a market void.

Throughout the conception and design process of the Hydroponic Unit and its mobile application, the project team has encountered many new obstacles. As a group of four computer engineering majors, there was less hardware knowledge available to draw upon to solve these challenges, but we believed this presented a unique learning opportunity where the team was able to collaboratively research and solve hardware-related design problems. This opportunity provided us with a more well-rounded education by reinforcing our software-oriented backgrounds with additional hardware experience, and strengthened our team dynamic and interpersonal skills.

Appendix A - References

1. CC3220MOD/CC3220MODASx Data Sheet. https://www.ti.com/lit/ds/symlink/cc3220mod.pdf?ts=1625368494598&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed June 19, 2021.
2. CC3220 Production Guide. https://www.ti.com/lit/an/swra568/swra568.pdf?ts=1624842815347&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed July 10, 2021.
3. "I^{2}C-bus specification Rev 6" (PDF). NXP Semiconductors. April 4, 2014. https://www.nxp.com/docs/en/user-guide/UM10204.pdf. Accessed July 2, 2021.
4. NHD-0420D3Z-FL-V3 Data sheet. https://www.newhavendisplay.com/specs/NHD-0420D3Z-FL-GBW-V3.pdf. Accessed June 21, 2021.
5. https://techblog.ctgclean.com/2020/02/liquid-level-sensors-floats/. Accessed July 1, 2021
6. https://www.electroschematics.com/optical-liquid-level-sensor/. Accessed July 1, 2021
7. Optomax Digital Liquid Level Switch Datasheet https://sstsensing.com/product/optomax-digital-range-of-liquid-level-switches-2/. Accessed July 6, 2021
8. Industrial pH sensor example with picture of electrode encased in polytetrafluoroethylene membrane: https://www.dfrobot.com/product-2069.html. Accessed July 16, 2021.
9. pH sensor specifications retrieved from DFRobot's liquid sensor selection guide. https://www.dfrobot.com/blog-1138.html. Accessed July 24, 2021
10. Rockwool FAQ guide https://www.rockwool.com/north-america/advice-and-inspiration/faq/. Accessed July 26, 2021.
11. Information about choosing a hydroponic nutrient solution https://www.trees.com/gardening-and-landscaping/hydroponic-nutrient-guide. Accessed July 26, 2021.
12. IEEE/ISO/IEC 12207-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes. https://standards.ieee.org/standard/12207-2017.html. Accessed July 2, 2021.
13. International Standard IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. https://standards.ieee.org/standard/830-1998.html. Accessed July 3, 2021.
14. IEC 60335-1:2020. Accessed July 5, 2021.
15. ISO/IEC 9899:1999. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf Accessed July 10, 2021.

16. Effective Dart. https://dart.dev/guides/language/effective-dart Accessed July 10, 2021.
17. Google JavaScript Style Guide. https://google.github.io/styleguide/jsguide.html. Accessed July 10, 2021.
18. University of Arizona study on the use of water in soil-based grows versus hydroponic grows: https://cals.arizona.edu/swes/environmental_writing/stories/2011/merrill2.html. Accessed July 15, 2021
19. Remarks from a Danish professor about time we have left until all water on earth is unusable unless drastic changes are made https://www.theworldcounts.com/challenges/planet-earth/freshwater/when-will-the-world-run-out-of-water/story. Accessed July 15, 2021
20. CC3220MODx data sheet page 67 design considerations https://www.ti.com/lit/ds/symlink/cc3220mod.pdf?ts=1625368494598&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed August 2, 2021.
21. RF layout requirements provided by the CC3220MODx Data sheet on page 68. https://www.ti.com/lit/ds/symlink/cc3220mod.pdf?ts=1625368494598&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed August 2, 2021.
22. Linear regression line calculator https://www.socscistatistics.com/tests/regression/default.aspx . Accessed July 28, 2021.