Automated Ventilation Controller

UCF Senior Design 2021

Group 15

Wendy Dominguez - Computer Engineering
Gisela Griesheimer - Electrical Engineering
Angelica Longo - Computer Engineering
David Munyon - Electrical Engineering

Sponsored By Chris Neiger

Table of Contents

List of Figures

List of Tables

1. Executive Summary

Home automation has become a vital part of our lives, from embedded smart technology in coffee makers to programming door locks, and as technology keeps advancing, so do the new things we are able to automate to make our lives easier. In our first team meeting, we discussed the different types of home automation project we could do that might benefit our sponsor, as well as meet the requirements for senior design. One of these ideas we had was creating a smart lock system. However, as we spoke to our sponsor and discussed what the environment for our project would be, we realized that it would come to a much greater benefit to create a system to control the warehouse vents. Our sponsor owns a warehouse full of machining tools used to make robotic equipment, and during the school year, also sees a lot of people coming in and out to work on projects.

The warehouse air conditioning system in place is used in conjunction with ceiling vents that control the temperature of the warehouse. This system is not only used for the comfort of the guest utilizing the warehouse, but also for the machines that are available there. Keeping these machines cool and reducing the humidity in the air is key for them to run at their optimal capacity. The air conditioning and the vents available are controlled by two different systems. The vents have an analog control system that controls each of the four vents. Each vent has three available configurations, they can be closed, open in an upward direction, or open downwards. Thus, we concluded on creating an automatic air ventilation system for our sponsor. We aimed for a cheaper, greener and more efficient alternative that wouldn't require manually opening and closing outlets. This custom system would contain features specifically requested to satisfy our sponsor and his warehouse needs. Along with being cheaper than the alternative smart home thermostats available in the market, it will also include features that will assist and inform the user in real-time. The system will include multiple indoor sensors throughout the warehouse that will read environmental data, vent motors to adjust the orientation for the vents, and a touch LCD display on the main control unit to interact with the data from the warehouse. Along with this, it will also have its own web application that will implement the same functionalities as the main controller but will give the user access to the control unit remotely. One of the most important tasks of our controller is to not only be able to read and display information given by the sensors, but to also regulate the warehouse ventilation based on the data automatically. The system should take things into account like the temperature, humidity, time schedules and other factors to control the vent motors so the user doesn't have to, and to ultimately minimize the amount of energy being used.

2. Project Narrative

This project narrative is a description of our group's project, and its goals to be achieved. The narrative section covers each proposed objective in detail, laying out all our plan methods to be used and our reasons as to why we chose them. Here, we have examined any anticipated challenges, solutions, expected results, a timeline, and each of our individual responsibilities to meet our ambitions. We have included our motivation for the project, its features, and a brief plan of action to make a clear connection from our need statement, to our capacity to solve the problem.

2.1 Motivations

Each ceiling vent in our sponsor's warehouse currently has a manual double-pole, double-throw switch that controls the vent's motor. The switch has three positions: the up position, down position, and the neutral position. When in the up position the roof vent opens. When in the down position the roof vent shuts. The neutral position is used to stop the vent part way. The current setup requires the user to manually control the ventilation when it is desired. This process can become laborious and is prone to error since the warehouse has multiple workstations and machines laid out throughout its floor plan. This process can also lead to instability and inefficiency since it can only work while someone is present at the warehouse. This is where our team's programmable ventilation controller will take charge.

2.2 Goals

The data provided by the multiple indoor sensors will be used to calculate an average indoor temperature and humidity. This average will then be displayed on the screen of the main unit, along with the data read by the outdoor sensor. Both the indoor and outdoor information will be used to determine how we anticipate automating the ventilation. The vent dampers are expected to change orientation automatically based on multiple conditions such as, a threshold temperature, an "ideal indoor temperature" set by the user, or changes in humidity. We expect to accomplish not only automate ventilation, but also optimize this ability in order to use the lowest amount of energy from the AC unit as possible. Our idea of optimizing energy usage will mainly be done by implementing a time schedule for the vents to be opened or closed. That way the user can either set the vents to be opened for only a specific duration of time, or set the vent dampers to automatically change orientation based on the environment conditions in the warehouse.

We chose to pursue this project because it was a benefit to our sponsor. This project will allow us to get more experience working with microcontroller hardware, controller programming, air circulation, as well as both the front and back end of web development. Our project will utilize Arduino parts mainly because of the high level C/C++ programming language and simple Arduino IDE. This commonly allows for more focus on the programming logic instead of worrying about the operation complexity of other microcontrollers. It's straightforward enough for beginners if all you wanna do is control electronic components (like motors, sensors, and wireless bluetooth modules) and nothing else.

We believe this programmable ventilation controller differs from all the other smart home products like Nest, because it will be built to better aid the needs of our sponsor and the personnel that utilizes the warehouse. It will, therefore, take into account more personalized energy saving approaches. It will also have a simpler user interface to make the user experience easy and intuitive. Some smart home technologies require the user to be connected to the same WI-FI network to be able to configure anything for that device, whereas our controller should allow the user to be anywhere with any WI-FI network and be able to make changes.

2.3 Features

The main objective of this project is to automatically control the roof ventilation system of the sponsor's warehouse. Currently, there are four, double pole double throw switches used to operate the ventilation motors with forward polarity to make the roof vents open and with reverse polarity to make the roof vents shut. To automate this, each switch will need to be replaced with four relays. The relays will act in pairs to align the motor's wiring for both forward and reverse operation. For the roof vents to open, the 'closing' relay pair will open first, then the 'open' relay pair will shut. For the roof vents to close, the 'open' relay pair will open first, then the 'closing' pair will shut. The sequence of the first relay pair opening, prior to the next relay pair shutting, is required to prevent shorting the wires and causing damage to the system.

Our team plans to build and program a unit ("controller") that automates air ventilation previously mentioned in the sponsor's warehouse, as well as monitor other environmental factors. The controller will include a Bluetooth module, where the Bluetooth module connects to multiple sensors inside the warehouse (one attached to each vent), and one outside. These sensors will be used to communicate both temperature and humidity to the main unit. The controller will also include a Wi-Fi module, where the Wi-Fi module will be used to connect a web application to the controller via a hosted server. This web application will be utilized as an alternative way to monitor and alter the vents and temperature whenever the user is not physically near the controller at the warehouse, this will allow them to change it from anywhere, as long as they have a WI-FI connection.

Thus, the main controller unit will need a local touch screen LCD display with a minimalistic interface, so that the user can easily monitor information and navigate through multiple features. The display is expected to show the average indoor temperature and humidity percentage read by the multiple indoor sensors, as well as the temperature and humidity percentage given by the 1 outdoor sensor. The user should also be able to manually set an ideal indoor temperature, as well as manage the orientation of the connected air vents (tilt up, tilt down, close) from the touch LCD (this can also be done through the web application, since both will have the same functionalities and interface). There will be an information icon at the top right of the display, which can inform the user of things like the battery life status for each sensor, the Bluetooth connection status, the Wi-Fi connection status, and the duration of how long the vents have been opened in order to track energy usage.

2.4 Requirement Specifications

The requirement specifications for this project are listed in the type below. The purpose of these specifications is to describe in detail the different components of this project without going into detail, into the technology that will be used to implement them. These specifications can be seen as a rough outline of the project's goals. These specifications are subject to change as the technology behind them is researched and implemented. This change is due to finding a better, more cost effective way of implementing the same goal without sacrificing the quality or the constraints given by the sponsor. The table below is an initial assessment of how to reach the sponsor’s goals for this project. The requirement specifications is also a brief outline that is used while researching to ensure this project will meet or exceed the expectations of the sponsor. At the end of the project this table will be referenced to ensure most if not all the requirement specifications have been met.

Table 2.1: Requirement Specifications

Requirements	**Specifications**
Sensors	● 4 total sensors ● Sensors will run on one 9V battery each ● Will be able to transmit a distance of 75 meters ● Will transmit data via bluetooth 4.0 (BLE) ● Expected battery lifespan will be 1.5 years
Wireless	● Bluetooth module will run off a 3.3 volt regulator ● Will connect temperature sensors to main control unit ● Expect to need 6 in total to get a good coverage of the entire warehouse ● With different bluetooth modules needed, different channels will need to be utilized to ensure proper communication ● Maximum bluetooth range is around 100 meters
Main Control Unit	● Unit will run off the power supply of the A/C controller ● Will run off a 5 volt regulator ● Will sense when A/C is running ● A/C controller is already installed at the location
Display	● Will have touch screen capabilities for local control ● Will have an easy-to-read graphics ● Larger LCD displays require more ports and can have lag when driven by an 8-bit module ● LCD Module should have pre-written drivers to ensure proper execution
Website	● Will utilize a database and an API for functionality ● Will have a similar layout as LCD Display for controlling remotely ● Will allow the user to control the main unit from anywhere, as long, as they have WIFI ● Will have to have enough security that it cannot be easily hacked
Ventilation Motors	● Position the ventilation ports ● Wiring will need to be able to operate the motors in both forward and reverse ● To prevent short circuits the relays will need to be normally open ● Motors are already installed at location
Power Supply	● Must have a reasonably small footprint ● Has to be capable of powering the display, and other accessories

2.5 Marketing and Engineering Requirements

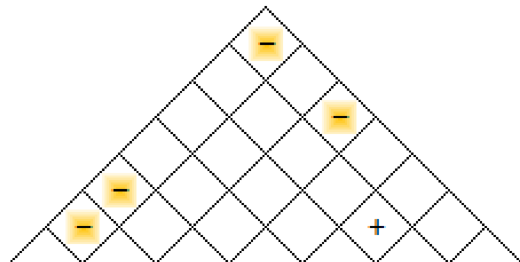
The marketing and engineering requirements table is a table that compares different aspects of our project's goals. This table looks at the fact that optimizing a certain area or goal of this project may have an unexpected impact on other areas of this project.

Table 2.2: House of Quality

Correlations	
Positive	+
Negative	−
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

Direction of Improvement	
Maximize	▲
Minimize	▼



		Column #	1	2	3	4	5	6	7
		Direction of Improvement	▲	▲	▼	▼	▼	▼	▲
Category	Weight	Engineering Requirements / Customer Requirements (Explicit and Implicit)	Efficiency	Connectivity	Cost	Maintenance	Weight	Dimensions	Accuracy
Sensor	8	Battery Powered	○	●	●	▽	●	●	▽
	9	Wireless Communication	●	●	○	▽	○	○	▽
	6	Accurate	●	○	●	●	○	●	●
Main Control Unit	9	Wireless Communication	●	●	○	▽	○	○	▽
	7	Internet Control	○	●	●	○	▽	▽	▽
	8	Local Control	○	▽	○	○	●	●	▽
	8	Local Display	●	▽	●	○	●	●	▽
	6	Wall Powered	▽	▽	●	●	●	●	▽

2.6 Key Engineering Specifications

The key engineering specifications is a table that is used to prove that we have met the design requirements of this project. All goals of this table should be met.

Table 2.3: Key Engineering Specifications

Specifications	Description	Demonstrable
Wireless Communication	The main control unit will be able to communicate wirelessly with the sensors to obtain the temperature and humidity data	Can connect and get data from a minimum of 4 sensors
Remote Control	The main control unit will have the ability to share its data and be controlled wirelessly	The ability to access the control unit remotely will be via a website with the same functionalities as local control and should allow the user to change the ventilation orientation from at least 10 feet away.
Local Control	The main control unit will have the ability to see the status and receive commands locally	A vent can be opened or closed with 1 touch of the screen
Display	The main control unit will have an integrated touch display that reads out vent status and temperature and humidity data from the wireless sensors.	The data from the wireless sensors and vent status will be updated a minimum of 1 time per minute
Website	Information from the database will be reflected on the webpage as received values from sensors	Can receive commands from at least 1 remote device chosen by the sponsor
Accuracy	The readings from the wireless sensors will be compared to a known standard ensure the requirement will be met	The temperature from the wireless sensors will be within ± .5 degrees celsius. The humidity from the wireless sensors will be within ±2%
Automated Vent Control	Vents will have an automated mode that takes control of the vents without any user input	The vent will be monitored for a 24 hour period to ensure proper automatic cycling.

2.7 Block Diagrams

The below block diagrams illustrate the high-level overview of both the hardware and software systems of our project. Our ventilation controller's main parts and functions are represented with blocks connected by lines that show the important working relationships.

For the high-level hardware diagram, our main controller's unit will consist of 5 major components: a Wi-Fi module, a Bluetooth module, a display, a microcontroller, and relays. Each of these components give support to a different functionality in order to support our project. The Wi-Fi module will provide internet connect for our website, the display will support a user interface, the microcontroller will sustain all the serial programming, the relays will replace the manual switches to control the vent motors, and the Bluetooth module will provide a way to get information from all the sensor units placed around the warehouse.

Each of these sensor units consist of their own major components too. They each will have a Bluetooth module, a temperature/humidity sensor, a microcontroller, and a battery. The battery will obviously be used to provide power to the unit, the sensor will grab temperature and humidity data from the warehouse, the microcontroller will sustain all the serial programming, and the Bluetooth module will be the means of sending the sensor data to the other Bluetooth module on the main controller unit.

For the high-level software diagram, the micocontroller's code will be the backbone to establish connections to 4 critical components. These connections include a server, internet, the database, and wireless communication. The sever and database is provided by a free hosting service that we chose, the internet is supplied by the connected Wi-Fi module, and the wireless communication is supplied by the connected Bluetooth module. Each of these connections come together to build the user interface seen on both the website and controller display.

This user interface is expected to be the same on the website and controller display. Both will have a home screen that shows the temperature and humidity, the switches for each vent, and an information menu which displays things like battery life, Wi-Fi connection status, and energy usage.

Figure 2.1: Hardware Block Diagram

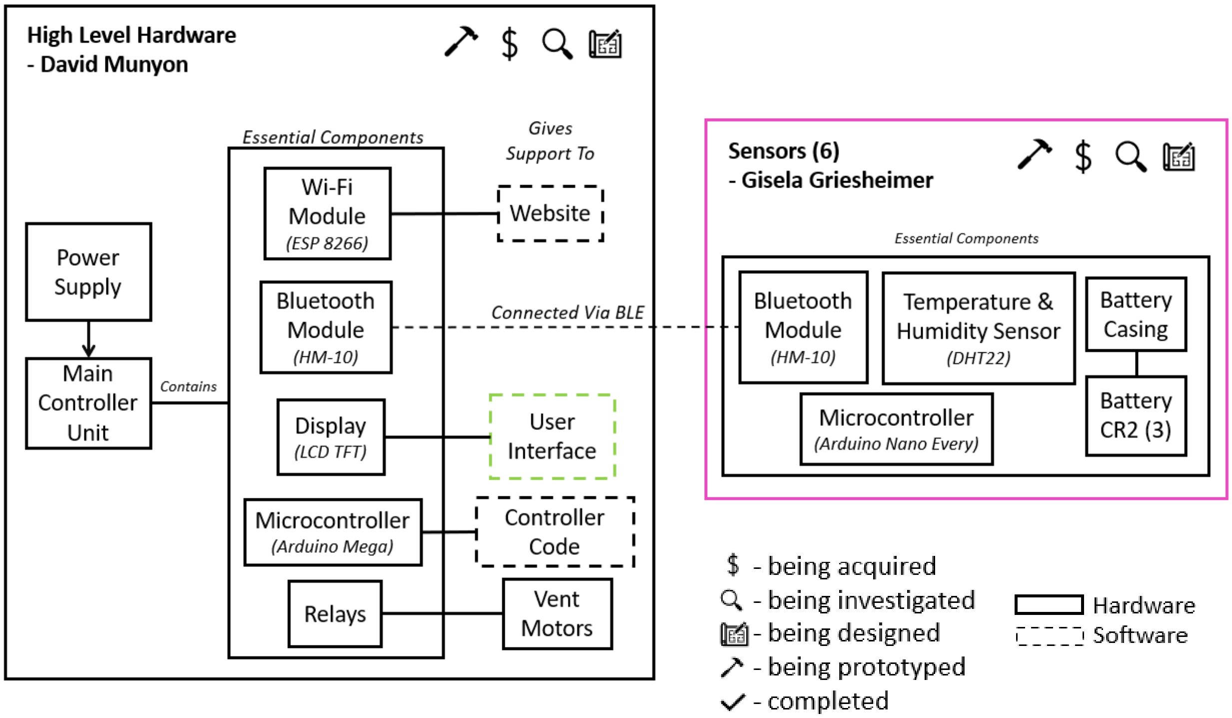
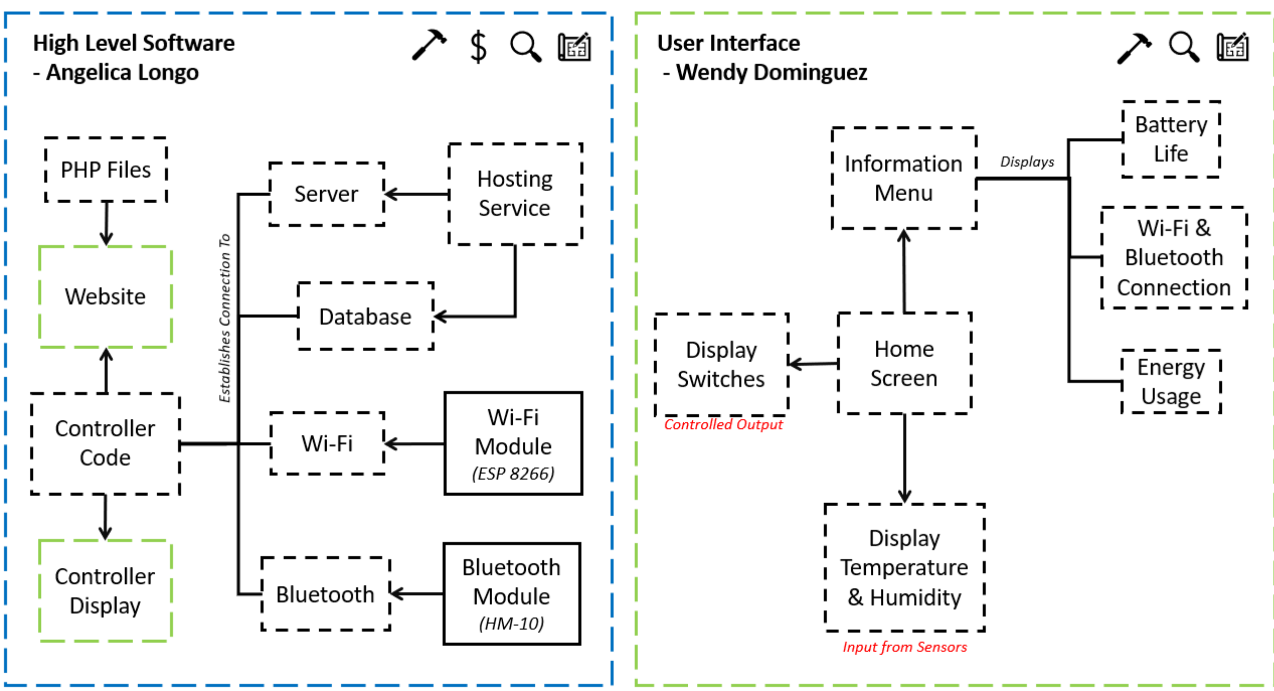


Figure 2.2: Software Block Diagram



2.8 Physical Requirements

Our project is being designed and tested four our sponsors' warehouse. To accurately incorporate our design into this space, we will need to account for the dimensions of the warehouse and the vents' placement for optimal cooling of the space.

Figure 2.3: Warehouse Space

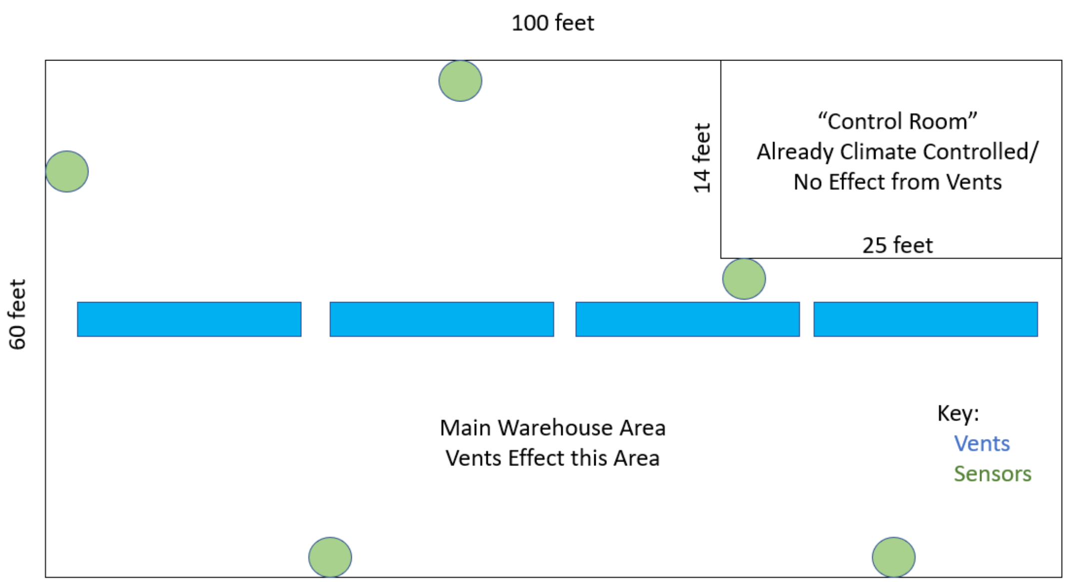
The Warehouse dimensions are 60 feet x 100 feet, with the ceiling height ranging from 18 feet by the walls to 24 feet in the lofted center. The four vents that we aim to autonomously control run down the center of the warehouse at the 24 foot center. They can each be opened and closed independently of each other.

Figure 2.4: Warehouse Vent - Open

Figure 2.5: Warehouse Vent - Closed

The warehouse also contains a "control room" space, which is 25x14 feet. The control center uses a completely separate air conditioning system from the rest of the warehouse. Therefore, the space we are designing for is not exactly rectangular, as depicted in the diagram below.

Figure 2.6: Warehouse Floor Plan

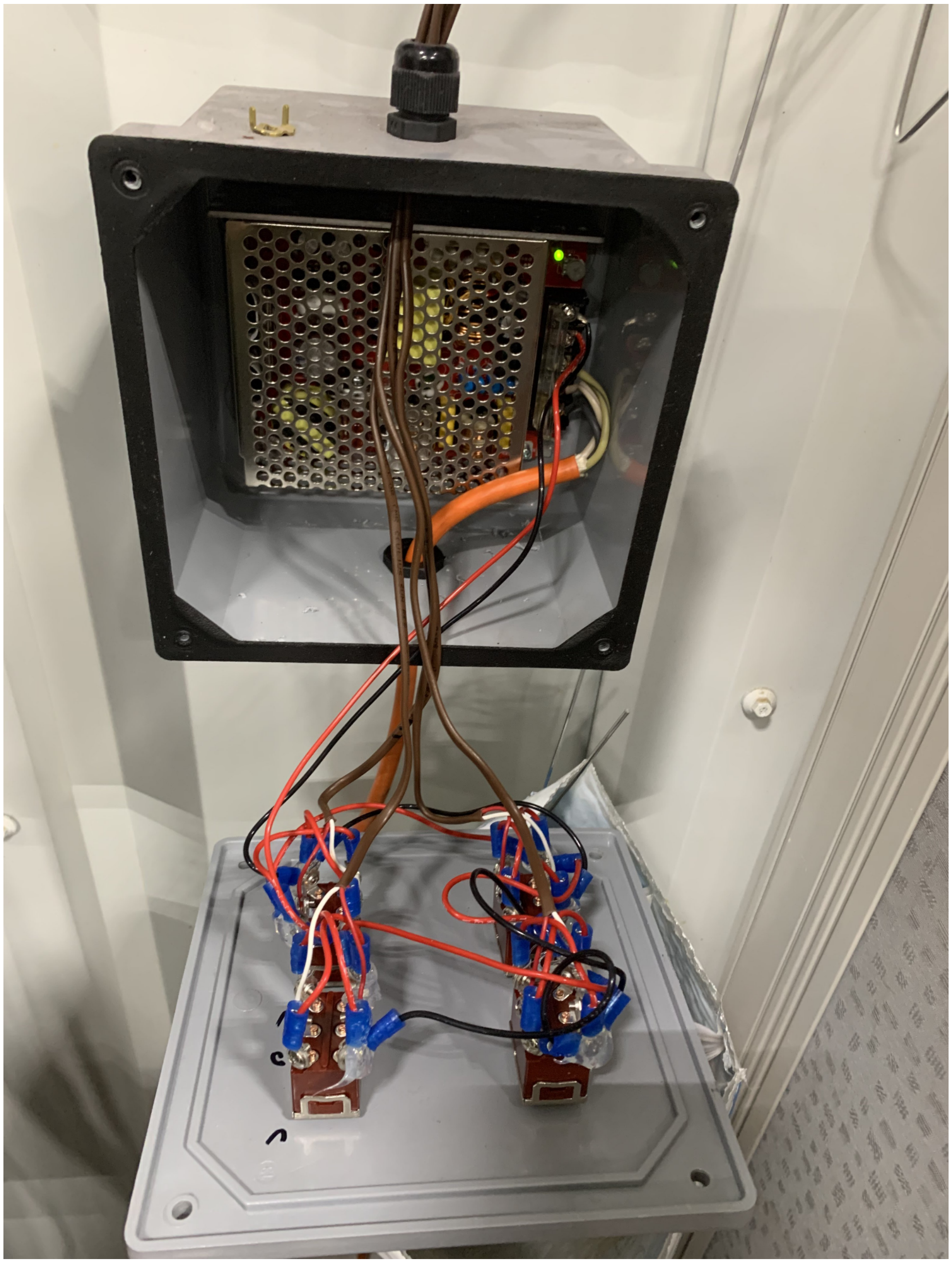


In *Figure 6* above, expected sensor placement is shown. We expect that our sensors will be wall mounted, approximately 4 feet above the ground so that they are both out of the way and easy to access for maintenance. Additionally, putting them too high would cause our temperature detection to change, since heat rises and we are trying to design for humans at ground level. Ideally, our sensors will only need to be interacted with for battery replacement, since our user should be able to obtain all the data needed to check the system from either the control panel within the warehouse, or by checking the web interface that we design from anywhere they have WiFi connection. Each of the four vents in the warehouse is currently controlled by a flip switch which needs to be hand-toggled to instigate vent movement. The system has four switches, one per vent, each of which can be set to three modes: vent open up (switch flipped high), vent travel paused (switch neutral), and vent closed (switch flipped down). This gives the option of only opening certain vents, or having some vents open fully while others open partially. While the best way to cool may be opening all vents fully to allow the most warm air to escape, this allows our group to research airflow and see if another configuration would allow us to potentially integrate windflow to better cool the space.

Figure 2.7: Current Vent Controller (Exterior)

From talking to our sponsor, we were informed that the current system uses a 12 V AC to DC converter to power the vents, with 12 V being vent open, 0 V being motion paused, and -12 V being vent closed. To account for any error, we measured the system for each vent using a voltmeter.

Figure 8: Current Vent Controller (Interior)



The above image shows the inside of the control box. The orange wire coming in through the bottom of the box is connected to the wall power, and goes into the AC to DC converter. The black wire coming out of the AC/DC converter is ground, 0V, and the red wire is Vcc, measuring at -12.01 V. Each vent had four input wires, in two pairs of two, and two output wires, sleeved in a brown coating that leaves out the top of the control box and connects to the vents.

For each vent, we measured the following voltages for various positions:

Table 2.4: Voltages For Each Vent's Position

	Vent 1 (V)	Vent 2 (V)	Vent 3 (V)	Vent 4 (V)
Vent Closed	12.01	-12.01	-12.01	-12.01
Vent Hold (after ~20 Seconds)	.006	-.006	-.004	-.003
Vent Open	-12.01	12.01	12.01	12.01
Vent Closing	11.98	-11.98	-11.98	-11.98
Vent Opening	-11.97	11.98	11.98	11.98

From these numbers, we see the values expected but that the polarity of vent 1 is consistently reversed compared to the expected polarity. This is due to assumed installation errors, and will be taken into account when coding vent 1.

We also noticed that the Voltage of the vent hold did not immediately go to 0. Rather, it disconnected both pairs of inputs, and let the voltage decrease to 0V. While flipping the switch did stop vent movement immediately, we noticed that the voltage would not go to exactly 0 V very quickly. The voltage measurements would very quickly drop to +/- 0.16 V, but it took about 20 seconds to get a value that was less than .005 V.

Additionally, we can see that there is a voltage drop of approximately .03 V when the vent is moving vs when the vent is in position. This was true for both moving from open to closed and moving from closed to open. This is low enough that it should not have a significant effect on the operation of the vent controller.

In summary,

- The warehouse space we will be implementing our design in is about 60' x100'
- The warehouse has 4 vents that can either be open, closed, or paused between states independently of each other
- We will need multiple sensors to account for the space since it have varying temperature from machine usage
- The vents currently need to be toggled from a physical switch in the warehouse

- The vents run on a 12 V system, where +12 V is vent open, 0 V is vent paused, and -12 V is vent closed
    - Due to and issue in the warehouse's wiring, vent 1 has reverse voltage values (open is -12 V; closed is 12 V)

3. Research

The purpose for this section of the paper is to introduce the research conducted for this project. The research conducted in this paper starts with a look into current products that are available on the market. Some of the items that were considered were the price point of the current products and the features that these products brought to the table. Next we looked into current technologies. This section was broken up into many parts because this project combines the use of several different technologies. First we weigh the pros and cons of several different wireless communications technologies. Then we weigh the pros and cons of different components that use the technology that we have chosen. This section then ends in component selection based on the research conducted.

3.1 Existing Products

While doing research for the design of our vent control systems some similar products were seen, but none of them were designed to accomplish what our project wants to accomplish. The first similar product was an add on to a traditional home air conditioning system. This add-on named ecovent, opened and closed the vents in a forced air flow air conditioning system. The price for a standard 1500 square foot house was around $2,750. What makes our project different is first the price our budget will come in at less than half the price of the smallest ecovent package. Secondly, Our project will not deal with forced air flow. When our vents open what will move the hotter air outside is the thermal gradient. The warm air at the top of the warehouse will try to rise and naturally leave through the vents at the top of the roof. Other similar products are found in the greenhouse automation industry.

There are several greenhouse controllers on the market that can be bought on amazon. One such controller is made by a company called autopilot and costs around $180 dollars. This controller just reads the temperature and humidity of the greenhouse and turns on and off outlets that are located on the side of the unit. These outlets can open and close vents if limit switches are installed to control the motors. These outlets can also turn on and off heaters and fans to control the temperature of the greenhouse. There are several features that separate our project from this competitor. First, our project will contain wireless sensors that can read the temperature and humidity of different areas remotely and is not limited to just the area around the main control unit. Secondly, our project wirelessly connects to the internet to allow remote control of the main control units settings and to see the current status of the vents. However, this product highlights further applications of our project. Moisture sensors can be added to the input data of the main control unit. Additional relays can be added to control fans and solenoid valves. These additions would make an extremely robust and cost effective solution to control almost every aspect of an entry level greenhouse.

Table 3.1: Existing Product Comparison

	Ecovent	**Autopilot**
Price Point	$2750 and up	$180
Sensor Communication	Bluetooth	Hardwired
Sensor	● Temperature ● Humidity ● Vent Pressure ● Battery life	● Temperature ● Humidity
Method of Air Flow	Forced Air Flow	Natural or Forced Air Flow
Method of Control	wirelessly open and close vents	Turn on and off outlets
Number of Vents Controlled	Not listed, but greater than 10	4 Outlets*

**Outlets can be used to control vents, but cannot be used to control other components*

3.2 Bluetooth vs Wi-Fi vs Custom EM Band

When it comes to choosing a method of wireless communication between the sensors and main control unit, there are many aspects to consider. First is range. No matter how many benefits a wireless communication system has, if the range is not far enough to communicate across the sponsor's warehouse, then it will not be suitable for our applications. Second is power consumption. The wireless sensor modules need to have a long battery life to minimize maintenance on the sensor modules. If the battery life is too short then the maintenance schedule will be an annoyance to the sponsor. The shorter battery life will also significantly increase the running cost of the system with frequent battery changes. The third aspect to consider is the ability to communicate with multiple devices. When considering this factor, it would be ideal for the wireless sensors to maintain a line of communication with the slaves. However, it is not entirely necessary. The master could only connect to one device at a time to receive the data then move on to the next device. The last aspect to consider is the ease of use. When considering the ease of use the main factors to consider are the reliability of the connection and the difficulty of programming and implementing the system. For this project, the wireless system should be very reliable and easy to implement.

The first wireless communication system in consideration is Bluetooth. Bluetooth is the first considered due to its popularity over the past decade and the vast amounts of information available on the subject. The range for a reliable Bluetooth connection is around 50 meters which is more than enough to meet the design requirements for use in the sponsor's warehouse. Bluetooth also consumes a very small amount of power which is ideal when running off battery power.

Additionally, Bluetooth is known to be easy to implement since it uses UART communication. However, the biggest drawback is that in a Bluetooth network there is one master and one slave communication at a time. To connect to multiple devices either the PCB would need to have several transfer and receiving pins each connected to their own Bluetooth module, or you would need to control the Bluetooth module with software. The software algorithm would need to connect to one module, transfer and receive the data, and disconnect from the module. This process would have to be repeated for each Bluetooth sensor.

The second method of wireless communication to consider is Wi-Fi. Wi-Fi has become the most popular option with the recent wave of Internet of Things devices (IoT) which our project will fall under. However, we will not be using the Wi-Fi to connect the sensors to the internet, we would be using it to connect the sensors back to the main control unit. The first thing to consider with Wi-Fi is the range it will reach. Wi-Fi has a range of around 75 meters which is better than Bluetooth and will more than suit the needs of supplying wireless communication within the warehouse. The second item to consider with Wi-Fi is its power consumption. This is where Wi-Fi falls short. The power consumption of wifi can be up to ten times the amount of the power consumed by Bluetooth. This will decrease the battery life of the sensor modules and increase the maintenance frequency and running cost. The third item to consider with wifi is its ability to communicate with multiple devices. Wi-Fi was designed to be able to communicate with multiple devices using multiple channels. This benefit of Wi-Fi would allow the sensors to be in constant communication with the main control unit. The final item to consider is the ease of implementation. The ease of implementation of Wi-Fi is similar to that of Bluetooth. Most of the implementation will be handled on the software level. Since both Bluetooth and Wi-fi will use UART interface to communicate with the Main Control Unit.

The third method to consider is creating a custom method of wireless communication between the sensor modules and main control unit. The first three items to consider are very similar because when creating a custom method of communication. The designer can create it to accomplish all the necessary requirements for the project. Various design features can be used to decrease the power consumption, increase the range, and increase the number of communication channels needed for the multiple sensor modules. The biggest drawback to this method is the time required to create, test, and implement a custom method of wireless communication. To accomplish the design requirements of the project a custom method of wireless communication may go past the deadline of the final deliverable. Another negative to consider in designing a custom method of wireless communication is the interference coming from the flood of EM bands already used such as AM and FM Radio, Bluetooth, Wi-Fi, and a host of other sources.

Other design criteria to consider in the decision is the price of components. When researching which method of wireless communication to use for this project the price of Bluetooth modules and Wi-fi modules seemed to be very similar and would have very little impact on the final decision. Another consideration for this project is the warehouse itself. The sponsor’s warehouse where this would be implemented is a metal building which will have a significant impact to the wireless signals if trying to send a signal across them. For this reason, all outdoor sensors should be hardwired into the main control unit to prevent any communication issues. Indoor sensors should not feel an impact from the metal building when transmitting signals inside the building.

The final decision on the method of wireless communication for the project is to use Bluetooth communication between the sensor modules and main control unit. This decision was reached because of the lower power consumption of the Bluetooth modules which will have a significant impact on the functionality of the sensor modules. The biggest drawback to utilizing Bluetooth is the communication of one device at a time. However, this can be overcome by controlling the Bluetooth modules connection with software. The software will need to have the ability to automatically connect and disconnect from the different bluetooth modules. If not then another implementation can be used that implements multiple bluetooth devices.

Table 3.2: Comparison of Bluetooth and Wi-Fi

	Bluetooth	**Wi-fi**
Price Point	$11	$16
Current Draw	8 mA	80mA
Ease of use	Simple Mater/follower communication	More complex
Connection Stability	Very Stable	Very Stable
Wireless Range	50 meters	50 meters
Size	1.2” x .6” x .1”	2” x .9” x .28”
Number of Connections at a single time	1	> 6

3.2.1 Bluetooth 2.0 vs Bluetooth 4.0 (BLE)

While researching which bluetooth module to use for the wireless communication between the main control unit and the sensors. The main differences of the modules came down to which bluetooth protocol that each module used. The two main protocols available on the most popular bluetooth modules are Bluetooth 2.0 and Bluetooth 4.0. Both of these protocols have their unique benefits as well as some drawbacks associated with their use.

The first consideration will be Bluetooth 2.0, which is an older version of bluetooth protocol. Despite its age, it still has many potential applications for its use today. First, the range of bluetooth 2.0 is around 50 meters, which will be more than adequate for use in the warehouse. Second, it has a very high rate of data transfer. Its speed can be used to transmit large amounts of data such as music and pictures and will be more than sufficient for the application of transferring temperature and humidity data. A Bluetooth 2.0 module also consumes a small amount of power, with an average current draw of 30mA. Bluetooth 2.0 also uses a master/slave communication system which is only meant to connect to transfer data to one device at a time.

The second protocol to consider is Bluetooth 4.0 also known as Bluetooth Low Energy (BLE). This newer protocol has the same range as a traditional bluetooth module which is around 50 meters. This newer protocol can also transfer a large amount of data quickly, and will be more than enough to transfer the temperature and humidity data needed for our project. The biggest advancement for this bluetooth protocol is the introduction of sleep modes. These sleep modes allow the module to draw as little as 50μA when not transferring data. This new technology also only consumes 8.5mA when actively transmitting data. This newer protocol also transmits data in a master/slave communication network. However, Bluetooth 4.0 introduced a new feature called iBeacon, which has many potential applications in the IoT realm.

For this project we decided to use the Bluetooth 4.0 technology. This decision was based on the reduced power consumption of the BLE modules. While transmitting the BLE technology will reduce the power consumption by 3.5 times that of a conventional bluetooth module. But while in sleep mode the power consumption will be reduced by more than 100 times that of a conventional bluetooth module. This lower energy consumption will greatly increase the lifespan of the batteries and potentially allow for the use of a smaller battery back than would be needed for the bluetooth 2.0 technology.

Table 3.3: Comparison of Bluetooth 2.0 and 4.0

	Bluetooth 2.0	**Bluetooth 4.0**
Range	50 meters	50 meters
Power Consumption	30 mA	8 mA
Ease of Use	Simple	Simple
Sleep Mode Capable	No	Yes
Designed for Iot Use	No	Yes

3.3 Website

We have many available options for creating the front-end and back-end of our web application. This is where the full stacks come in. These stacks allow the software developer to develop both the client and server software. Each stack has different languages and softwares that they are utilizing, however there are some overlaps, but the advantages and disadvantages will be the deciding factors.

One of the options we are considering is the LAMP stack. Which utilizes the following technologies, Linux, Apache, MySQL and PHP. Each of these softwares are open sourced which is one of the biggest advantages. In this stack, the application is hosted in a linux server, with an Apache HTTP server running on top. The database utilized to keep the information stored is MySQL. The front end of the application is connected to the back end via the PHP script, which allows the web app to run smoothly, sending the appropriate request to the database, and returning it as appropriate. Another advantage of this stack is that it is easy to interchange a component for another open source software, for example, instead of using PHP as our script we can instead use Python. This is one of the most popular stacks today.

The other contender is the MERN stack, which utilizes mongoDB, ExpressJS, ReactJS, and NodeJS as its main components. Here the database does not use an entity relationship like the MySQL does, it is more document oriented containing key-value pairs, and utilizes Javascript. This stack uses more Javascript than the LAMP stack. After the database in the stack we have NodeJs which is used to run Javascript on a machine instead of in a browser. After this, is where ExpressJS comes in, it is used to build the back end of the website using NodeJS. And finally we have ReactJS, which is used to build the UI components. In comparison to the LAMP stack which usually uses a mixture of HTML, Javascript and CSS. Like the LAMP stack, its technologies are also open sourced. One of the biggest advantages of this stack is that all the technologies involved in it are utilizing Javascript, unlike the LAMP stack which is using different languages for each technology.

The MERN is a variation of the MEAN stack, and has surpassed it in popularity. However, the MEAN also has its own advantages. The software used in the MEAN is very similar to the MERN except for the client side framework, where Angular replaces ReactJS. Like the MERN the MEAN is also composed of software that uses Javascript. The MEAN has an operating system independence that the LAMP stack lacks, as it is restricted to using Linux.

Along with choosing the correct technologies for the website, we need to look into proper security for it, as we would like to avoid just anyone from accessing it and making changes to the vent configurations. For password security, we are looking into available password hashing options. So far we have MD5 hash implementation via javascript as one of our options. The MD5 hash takes any string of any length and encodes it into a 128-bit value. This is useful because it returns the string as random values/characters.

We will also be enforcing a strong password policy, requiring certain characters to be in the password. Along with the hashing of the users credentials, we are also looking into implementing SSL certification, which will give us an extra level of encryption. Following these requirements and needs, for the best UI experience, we will also be looking into making the site as responsive as possible, so that the air vent configurations can also be done via a mobile browser, that will not have it showing as a desktop view but rather be mobile friendly.

In summary,

- There are many full stacks available to create the website, each with its own advantage.
- The three we considered were the LAMP, MERN and MEAN stack. Each of which is composed of open source software.
- We also looked into adding some security, to the password and to the website as a whole.
- The MD5 algorithm was selected for basic password security and we looked into SSL certification.

3.4 Web Scraping

In the beginning of our research process we had originally decided on adding modules to the sensors to detect the weather outside of the warehouse and based on the readings the ventilation controller would decide on opening or closing the vents. However, after researching the parts we realized that it could be much easier to utilize web scraping, which is a term used to call the process of gathering information from the internet, to get data from a weather site and then communicate that through the ventilation controller. And so, this steered us into having to figure out what type of services or languages are available to do this. Ideally it will be another functionality to the website which will then be able to transfer this information to the physical ventilation controller.

One of the key things we will have to do is decide which of the many available weather sites we will be using, to be able to study the HTML structure. Along with this, there are many different ways to go about building a web scraping script, it just depends on the languages we choose to utilize to implement it. However, before even going into which languages are up for the task, we also had to take into consideration some of the potential challenges that would arise once we implement it like how each website is different and if we were pulling data from more than one the implementation would have to be different based on how each of those websites is set up. There could also be updates to those websites that might utilize a different format or utilize different technologies altogether. These were all things we had to keep in mind when researching and coming to a conclusion on how to implement it.

The first language that came to mind as well as a result in the search engine was Python. Python is a high-level language that can be used to automate simple things, it is also used by data scientists to create machine learning algorithms. For the purpose of this project though, there are some available libraries that can be utilized to complete this, such as beautiful soup. As described in the documentation, it sits atop an HTML or XML parse, and provides Python with keywords to iterate, search and modify the parse tree. Parse in this context refers to taking the HTML code or XML code, and extracting the relevant information. In our case that would mean the weather information for the area of Niceville. Beautiful soup lets the user get information about the title of the page, as well as getting all of the available URLs found within the HTML hyperlink tags. It can also find tags by their ID. From experience looking at the HTML code of some websites, the IDs or classNames look like random characters string together, but they usually have one and this API also allows for finding the next child in the parse tree or parent so it will be possible to get the information needed with this library. In the standard Python library there is a tool available to work with URLs, called urllib, one of the methods we found when researching this was to extract the HTML from the page by using the HTTPResponse's read method available, to be able to get the HTML code and then go through the process of getting the individual information, which will require a multi step approach to be able to get the string because it will require us to set where the appropriate tag is and then slice the string from it. While this method would be tedious it would still allow us to get the data we want. Some notable mentions that also crossed our research path were XPath, which utilizes path expressions to select node(s) sets in a HTML or XML document. Scrapy which allows the user to download the web pages, processes them and saves them, and then web scrape from them. RoboBrowser which utilizes a mixture of BeautifulSoup and requests, however, it would require a lot more research since neither of our software individuals have had any experience with web scraping as of the time that this research section is being written.

Another open sourced technology is the use of NodeJS, like Python, there are tools available that are already in use for web scraping. One of the most widely used libraries, Axios, used to make HTTP requests from the Node.js environment, and provides its users with the functionality to download data removing the middleman, the json method, when getting the HTTP request result. Nightmare is another library that can also be used to get files from a website and data. It is useful for interacting with a web page or collecting the information that may only be available after the Javascript on the page is run. A downside of it is that it could take up a big amount of resources, it is also limited to visiting one URL at a time, which wouldn’t be much of an issue in our case. Though it will store cookies and cache until the process is terminated with an end command. Another library we stumbled upon was Cheerio, which utilizes a subset of jQuery to get the needed information. We can also use a combination of Cheerio and Axios to create the web scraper, using Axios to get the HTML from the website and Cheerio to get the content from the Axios result. Selenium was another library that, like Nightmare, creates a HTTP request by completing actions, so it allows the programmers to interact with the page, and like some of the other tools can locate the items based on ID, Name, or hyperlink. This library is also available in Python.

Python and Node.Js were the top results that showed up in our journey of researching the web scraping technology available. They were the ones to have more documentation available for the majority of the libraries available to do web scraping. There were also some nods into Go, which is one of the newer languages available and C/C++. However, the API available for it isn't as large are the ones available for Python and Node.Js. And there were a couple of disadvantages that were also listed for each one. There was also an example of using Javascript, jQuery and Regex to be able to get the data. jQuery is responsible for scraping the data and then Regex is used to filter the data for the relevant information. There is also an API available called the OpenWeatherMap, which requires a sign up and will give us a specific API key, which might be the easier methods as we can probably utilize a regular javascript file and send for a json package to get a response from the API with a much clearer and cleaner format that does not contain HTML tags. There are a couple of things we will have to consider before choosing the appropriate API and language for this task, as well as the site from where we will be web scraping as some could have security measures to prevent web scraping, so these are all things we will have to keep in mind.

In summary,

- There were two main technologies that have their own libraries for web scraping, these were Python and Node.JS
- Each of the two programming languages had more than a handful of libraries available to complete the task, however, some get much more difficult to implement and require more than one library to be able to implement it successfully.
- There was also a third option of potentially using a Javascript file to query an available API that already has an implementation for getting weather information neatly.

- If we do not use that API, we will have to choose one site to get the weather information and study the way they store the weather information to be able to format the file to get it.
- Since we are leaning to a version of a LAMP stack, from the available programs that have libraries, Python seems to be the biggest contender.

3.5 Natural Air Flow

For our project to work effectively, the hot air trapped at the top of the warehouse will need to leave via the roof vents and be replaced with cooler air from the outside. This will need to occur naturally because there is no fan or air vent to force cooler air into the warehouse. This natural convection or thermal driving head has been used in many areas. This natural phenomenon is very powerful and is the reason for such strong breezes coming on off the ocean. Natural convection has been utilized by engineers for years to keep large warehouses cool, and even in the operation of nuclear power plants. Natural convection will be used in two separate ways for our project.

The first way natural convection will be used in our project is to let the warm air rise naturally. The colder air will naturally be pulled down to the bottom of the warehouse due to its density. This will create a thermal gradient between the top and bottom of the warehouse. The temperature at the top of the warehouse will be hotter than the outside air temperature. When the vents open the hotter air at the top of the warehouse will naturally rise and leave the warehouse. The hot air leaving the warehouse will reduce the pressure inside the warehouse creating a low pressure vacuum. This vacuum will naturally draw in air from the outside via any available openings. To maximize the effectiveness of this process a window, door, or garage bay should be open to allow the fresh, colder air to come inside.

The second way natural convection will aid in the cooling of the warehouse is through the wind. The warehouse is located within a few miles of Florida's gulf coast. Due to the warehouse's location, naturally colder air coming off of the ocean will aid in the ventilation of the warehouse. First, the wind will ensure that there is a natural temperature difference between the hotter air trapped at top of the warehouse and the outside air. Creating the conditions for natural convection. The wind will also aid in the ventilation of the warehouse as it will act similar to a forced air flow. When the wind hits the side of the building with an open window or door. The air will naturally come into the building and slightly increase the pressure inside the warehouse. This increased pressure will force air out of the vents at the top of the warehouse creating airflow.

For both of the methods of natural air flow listed above an open door or window will be required to maximize the airflow through the warehouse. This is due to the pressure differences created by airflow. However, opening a door or window is not always safe to do especially when the warehouse is unstaffed during the night hours. A practical solution for this would be to open a garage bay door only three to four inches and locking it in that slightly open position. This would prevent any unauthorized access to the warehouse while allowing for colder, outside air to flow in naturally into the building. Another effective solution would be to install a vent into the side wall of the warehouse. The side vent could be easily wired to open when the roof vents open which would allow for natural air circulation in the warehouse.

In summary,

- This project will use natural convection or thermal driving head to exchange hot in in the warehouse with cooler air
- Natural Convection can be optimized by leaving a window or door open to allow cooler air from ground level to enter the warehouse
- The breezes coming off the gulf can also assist in the natural convection process by supplying ocean cooled air.
- Ocean breezes will also make the hot air leaving the warehouse more effectively by forcing colder in through a window or door opening
- The addition of a vent located on the wall facing the ocean which can open and close with the roof vents may improve performance of the vent system.

3.6 Extensibility

In this section we will cover potential applications for our project outside of the warehouse. We will also cover potential additions that could easily be added to our project to make it suitable for use in other areas. These additions will seek to be minimally intrusive to the design and easy to implement once the project is completed.

The first potential addition for this project would be to increase the efficiency of the natural circulation by adding a vent to let air from outside into the warehouse when the vents open. To accomplish this the additional vent should be installed on the side of the warehouse where the air should be cooler than on the roof. When the hot air leaves the vents on the roof, this vent would allow cooler air to enter the building. This could be implemented easily with the addition of another relay. The relay would be coded to close the contacts which would open the side vent when any of the roof vents open. This would ensure that air could easily flow into the building without the inherent risk of leaving a door or window open. This would also be a hands-free option that would not require any operator action.

The second possible addition to this project would involve installing a duct with a blower to create forced air circulation. This option would increase the operational cost due to the additional energy consumed by the blower or fan. However, this option would maximize the airflow into and out of the warehouse. The implementation of this would be very similar to that of the side vent. An additional relay would be installed and coded to close the contacts of the relay when any of the roof vents open. Most relay options that would be suitable for this project can handle up to 250V AC with a current of around 10A. These relays would be more than suitable to drive a single blower or fan installed in a duct.

Another application for this project would be for residential houses. This roof vent system is a very popular option for residential houses to help remove hot air out of attics which would aid in keeping the house cooler. The implementation of our project would be simple since most houses have ridge vents installed on the roof already, so there is no need to to open and close vents on the roof. Our project would monitor the attic's temperature. When the attic becomes hotter than the outside air temperature the main control unit could use its relays to turn on ventilation fans to force colder air into the attic which would remove the hotter air. This implementation would reduce the runtime of the air conditioning system and reduce the overall energy usage of the house.

The fourth potential application for this project would be for its use in a greenhouse. Residential greenhouses are becoming more popular due to the increase in demand for healthy and organic foods. Greenhouses are a popular option for growing plants for use as food since they provide shelter for the plants from extreme weather and invasive pests. To implement our project in a greenhouse there are several key items to consider. First of which is the environment. Greenhouses tend to be humid and wet, which may create an environment that is prone to damage sensitive electronics. Due to this the main control unit should be housed in a watertight case. Further waterproofing can be added to the circuits in the form of a silicone compound that will repel water away from the printed circuit boards.

Another item to consider when utilizing our project for a greenhouse would be the power system. Many greenhouses found in residential areas will not have a dedicated energy source to power the PCB and motors required to open and close the vents. A great alternative power source would be a marine, deep-cycle 12V battery. This power source is ideal due to the main control unit is already designed to run off 12V DC. Furthermore many parts can be found such as motors to open and close the vents that already work off 12V DC. The downside to using a battery to power all automated functions of the greenhouse is its limited charge. Eventually, the battery will have to be charged back up. This problem can be minimized with the installation of a solar panel. Most consumer grade solar panels come in a 100W size which would be more than enough to keep the battery topped off. Along with the solar panel a solar charge controller can be added to ensure that the battery is being charged safely. This option should be an almost maintenance free approach to power the greenhouse automation system.

Another potential add-on to the greenhouse would be to implement a fan to blow fresh air into the greenhouse while the vents are open to reduce the humidity and temperature. This fan would be implemented in a similar fashion to the vent fans listed above. However, the fan should run 12V DC to work with the battery-powered system. The fan would be controlled through a relay to turn on when the greenhouse vents open and turn off when the vents shut.

Another potential add-on for this project would be to add additional sensors to read the soil's moisture level. This data is crucial to ensure the plants are not receiving too little or too much water. These sensors could be programmed to control solenoid valves that would run on 12V DC. These solenoid valves would be used to control the watering of the plants and to maintain a soil moisture level to ensure proper hydration of the plants, in the greenhouse. Additional software features can be added to the system in the form of calendars and reminders to allow the user to keep track and stay up-to-date on the maintenance of the greenhouse.

This section briefly touched on the potential uses of our project both inside and outside the warehouse. This project has shown to have many potential applications for its use and can only be limited by the limits of our own creative ability. Electrical components can be added by increasing the number of relays and coded to open and close their contacts based on almost any data provided to the main control unit. Additional sensors can be added via bluetooth connection or hardwired to an input/output pin or ADC pin to transmit their data to the main control unit.

In summary,

- Potential additions
    - Vent installed on the wall of the building that faces the ocean to allow colder air to come inside
        - Vent would open and close with the roof vents and be controlled by the main control unit

    - Installation of a duct with a blower to create forced air flow
        - Blower would turn on when roof vents open and off when roof vents shut. Controlled by the main control unit
- Potential applications outside of the warehouse
    - Residential use when installed in the attic
        - Main control unit would turn on and off a blower to move colder air into the attic. Hot air would escape out of preinstalled ridge vent
        - Cooling the attic could reduce the amount and cost of A/C
    - Greenhouse Control
        - Could be used to automatically control the vents
        - Addition of a blower to create forced air flow
            - Controlled with an additional relay
        - Solar Powered
            - Can run off a 12 V battery, and be recharged with a Solar Panel.
        - Addition of a soil moisture sensor to sense when watering is required.
        - Addition of a 12V solenoid valve to automate the control of watering.
            - Controlled with an additional relay
        - Would need a waterproof and weather resistant casing to prevent electronic component damage

3.7 API Testing Software

When building a website, there are many components that must get done for it to function successfully. One of the most important components that is in charge of the functionality is the API. Which for the LAMP stack, is composed of PHP files. For organization purposes there is one file for each functionality. These APIs oversee the establishment of communication between the frontend and the back end. It enables the frontend to get information from the database as well as add information to it. It is good practice to test each API out and make sure it is working as it is supposed to. This way we can use the Agile methodology, since to test the API we only need to make sure that the database is properly set up, the frontend and backend can be done at the same time and the only thing they will have to keep in mind is the name of the components that will be utilized to get the information. There are various technologies readily available to do this, some of the most popular are Advanced REST Client, or ARC for short. As well as Swaggerhub and Postman.

ARC is the simplest and easiest one of the list, as it is only used for testing. It will send a request to the API, depending on whether it is sending or receiving information the method will be different. It can be used to get information like a person's name, or it can be used to delete information too. It is really simple to use, there is only a requirement of the URL, and the header and body of the payload. For our purposes it will probably be using a JSON payload, we will be selecting that as the content-type so that the information gets sent the correct way and the API knows how it should expect it. Like the other testing software, it will also provide codes when something failed or if it was successful. While it is a quick way to check if the endpoints work, ARC does not provide a way to document them.

Swaggerhub is also an open-source software that has a free tier that will be more than sufficient for the testing and documentation of our APIs. It also provides a way to have the whole team see the documentation and see how everything is working, that will be good to make sure we are all on the same page. To test it we will have to write YAML blocks to describe the endpoints we wish to test. This will include tags, a summary, the operation ID, a description, as well as what it will consume and produce. It will also have to have the parameters specified. Each of the endpoints will require this YAML block. For testing, we will have to write the JSON package the way we decided it to be set up, so it can test the endpoint from the website. And provide the URL that holds the PHP file to be tested. When it gets tested, if it is successful, it will provide a code 200 with a description of it being OK. On the other hand, to let you know and if it is unsuccessful, it will also give you an error code which will be represented by a 404 with a URL not found description. In the documentation, there will also be examples of what each JSON is called and the types that are used, as well as any examples that are already logged into the system. Swaggerhubs free tier, comes with the ability to create multiple APIs, however they are not unlimited. Once you have reached a certain point, it will not allow you to create any more, however, usually if you delete some of the APIs you can create some more.

Lastly, we have Postman, which like ARC and Swaggerhub, can test the API endpoints and determine if they are working as they were intended to. It’s free tier comes with the ability to have up to three share requests from team members, which will encourage teamwork. It also has unlimited public API documentation. It has a choice of recovering deleted collections only up to a day after. It also comes with an unlimited number of allowed API, which is not something Swaggerhub provides.

While Postman has some very good benefits, since it provides collaboration apart from all the other benefits that Swaggerhub has, we will try all three, ARC for quick testing, and Postman or Swaggerhub for documentation. We will compare Postman and Swaggerhub together in real time to see which one is much simpler to implement and to document, and will hold everything we want it to.

In summary,

- API testing is essential to make sure everything is working as intended
- There are many technologies available that can do this, the main three we will consider are ARC, Postman, and Swaggerhub
- ARC is an easy to use API testing that does not a provide documentation option
- Postman has a free tier with many benefits that include documentation and collaboration with up to 3 team members. And unlimited API documentation option.
- Swaggerhub provides a testing and documentation feature, but there is a limited number of APIs that a user can make and provides a read only shared version of the documentation.

3.8 Wireframe Technology

Wireframes are very beneficial, they can facilitate the frontend process. They serve to layout the content and functionality on a page keeping in mind the user's needs. Wireframes are much easier to adapt than a concept design. These have to get done during the design phase to be of the most advantage in the project. To get these wireframes done, there is an abundant amount of free open-source software that is specially built for wireframing. The technologies we considered were Figma, Lucidchart and Canva.

Figma provides a great workspace to collaborate within teams, there is a slack integration for team communication. There is no lag on any changes made to the design, everything is kept up to real time. It is also free and can run in any browser available. There is also a desktop app available to make the design process much easier. It also comes ready with templates and themes for inspiration and to get started. It can also generate CSS code to make the design phase much easier. Figma has many features, however for beginners it is somewhat difficult to understand all the components and the best way to use them in one go. However, Figma is one of the best designing softwares currently out there.

Lucidchart has a free tier that restricts users to three designs, along with having an ability to create wireframes there, it can also help facilitate the design of Entity Relationship Diagrams, Use Cases, and all other UML diagrams. Lucidchart also has templates to get started on creating the wireframes but the free tier is limited, it does not include Android, IOS or UI shape libraries. However, compared to Figma their styling is much more limited as it tries to just do the simpler design of wireframes, without having to worry too much about the IU aspect of it. While it does not have a feature to concurrently update the design, users are allowed to share the wireframes as view only files.

Canva is a free graphic design platform. It uses drag and drop for its components. Canva's free tier is restrictive to its developers. However, for simple wireframing it works great. Canva is usually used for creative interactive presentations, which can include video presentations too. It has great available graphics and it also has collaboration features on projects. As well as a ton of available templates to use and get inspiration from. There are video tutorials available as well to get more familiar with the interface and, while it is not marketed as a wireframing tool, it can be used as one. For this project, we will be learning how to use Figma, utilizing everything it has to offer, to be able to use it to create the wireframes for our web application.

In summary,

- Wireframing is a very beneficial tool for developers, that gets done in the design phase of any project
- There are many wireframing tools available today. But the main ones we considered were Figma, Lucidchart and Canva.
- Figma is the tool with the most features available to for free
- Lucidchart has a restrictive access to the amount of graphs, diagrams, or wireframes available for free to one user.
- Canva has simple tools available for wireframing at no cost to the user, and has many good graphics available.

3.9 Core Technology Selection

This section will describe and compare the key benefits and drawbacks of popular technologies used for similar projects. These technologies are Raspberry Pi and Arduino. There are many areas to consider when selecting the brain that will control all the functions of the vent control system. One of main items to consider is computing power, will the selected technology be able to accomplish the task of controlling the vent system. Another item to consider is the compatibility of the technology with other components that will be used in this project. The third consideration is the complexity of programming required to implement the project. The last item to consider is the ability for the technology to meet the requirements of the Senior Design project.

Raspberry Pi was selected first due to its massive popularity in recent years for implementing Iot projects. The Raspberry Pi is a mini computer that utilizes a microprocessor to complete its tasks. Many of the functions of our project come standard on a Raspberry Pi such as a wi-fi module and bluetooth module. Raspberry Pi's also come standard with a graphics driver that is capable of driving a large touch screen which will be utilized in this project. The Raspberry Pi operates at a high frequency of 1.2GH and will be more than capable of processing the data from the sensors and controlling the vent positions. Most Raspberry Pi models come with 40 GPIO pins that can be used for additional sensors and relays. The Raspberry Pi is mainly programmed with linux commands. However, since it is a computer on its own, additional IDEs can be downloaded to allow programming in other languages such as Python. The Biggest drawback to utilizing a Raspberry Pi is it is not friendly to PCB design. Meeting the design requirements of substantial PCB design for electrical engineering would be extremely difficult.

The next technology in consideration is the Arduino. An arduino based microcontroller was selected also due to its popularity for similar IoT projects. The arduino differs from the Raspberry Pi because the arduino is a microcontroller not a standalone computer. With arduino being a microcontroller many of the key features of the Raspberry Pi are not standard such as Wifi, Bluetooth, and a Graphics Driver. This will make the project more difficult because all of these technologies will need to be integrated into the microcontroller. Also with the lack of a graphics driver the arduino will not be able to support driving a larger touchscreen which may reduce the overall quality of the project. Arduino offers many different microcontrollers with various numbers of GPIO pins ranging from 20 to 70 pins. These pins can be utilized to integrate the different components needed to implement the relays, bluetooth, wi-fi, and other functions. The arduino controller operates off a much slower 16MH crystal oscillator. However, it should still have more than enough computing power to implement the vent controls needed for this project. The arduino is programmed with the Arduino IDE that was specifically developed for user friendliness. This IDE should simplify the programming required for this project. Lastly, since this arduino is a microcontroller, PCB design for it should be easier to implement than the Raspberry Pi. The PCB design requirement for Senior Design can be implemented.

Based on the research conducted the Arduino microcontroller was selected as the core technology that will drive this project. Arduino has more drawbacks when compared with the Raspberry Pi, but should still be more than sufficient to complete the task required of this project. The main reason for the selection of the arduino microcontroller was the requirement for significant PCB design for Senior Design. The Raspberry Pi would be plug and play, and too easy to implement for a good project. A Raspberry Pi would not showcase the skills and knowledge gained while at UCF.

Table 3.4. Core Technology Comparison Chart

	Raspberry Pi	**Arduino**
Core Technology	Microprocessor	Microcontroller
Wi-fi implementation	Standard	Needs Wi-fi module
Bluetooth Implementation	Standard	Needs Bluetooth module
Processor Speed	1.2GH	16MH
GPIO Pins	40	Up to 70
Available Display Size	Large	Small (up to 4”)
Programming	Linux, Python	Arduino IDE
Substantial PCB Design	No	Yes

4. Part Selection

This section covers the parts selection required to implement this project. The technology that drives the following parts was researched and completed in the previous Section 3. Further research is conducted in this section, with a basis on cost and quality of the chosen technologies. This section is split up into two main categories. The first category looks at the parts required to implement the Bluetooth Sensors. The second category looks at the parts required to implement the main control unit. These parts will need to be able to work together as a whole to fully implement the goals of this project.

4.1 Sensor Unit

To complete this project, we will need to aggregate temperature and humidity data from across the warehouse. This requires the use of multiple sensor units. The section below discusses the build for each sensor unit, including the selection of each of the components we plan on using (battery, temperature sensor, microcontroller, and wireless communicator), and the technology we plan to use in the data sending between the sensors and the main control unit.

4.1.1 Bluetooth Connection

Each of our sensors will need its own bluetooth connection to the controller. These will have to be able to send data, but do not need to be able to receive data. When looking at which part to use, we compared the HC 05, HC 06, HC 07 and the HC 08. All of these would allow our sensors to connect to the control unit and support arduinos.

The HC 05 and HC 08 both support leader/follower configuration, which we will be using. Originally we had anticipated not needing the sensors to have a leader configuration, but when choosing batteries it was determined that if each sensor was able to initiate the connection to the controller, the sensor could turn off for the majority of time when it was not sending data. This would allow battery savings of approximately 10x vs using a follower Bluetooth device that would need to be on constantly to wait for the controller to request data.

We also considered using BLE with the HM10. It has the same pinout as the HC 05, so there would not be any physical ramifications of using one part over the other for the microcontroller. Additionally, BLE uses significantly less current than bluetooth and would therefore drain battery power at a much lower rate. It also uses a standard UART connection which will make integrating this piece into both our software and hardware designs easier.

The HM10 is only bluetooth 4.0, so it cannot connect to bluetooth 2.0 or 2.1, which the HC 05 and HC 06 use. This means we need to be consistent with our bluetooth vs BLE choice between not only the sensors but also the controller. After discussing price difference vs power drain, our team decided the lower power drain was preferable for our design, and chose to use BLE with the HM 10. Since the arduino uses 5V serial and the HM10 uses 3.3V, we will need to add a simple voltage regulator so that they can properly communicate the data. This is displayed in the pinout figure below, where the left resistor is 1k ohm and the right resistor is 2kohm.

Figure 4.1: HM10 Pinout

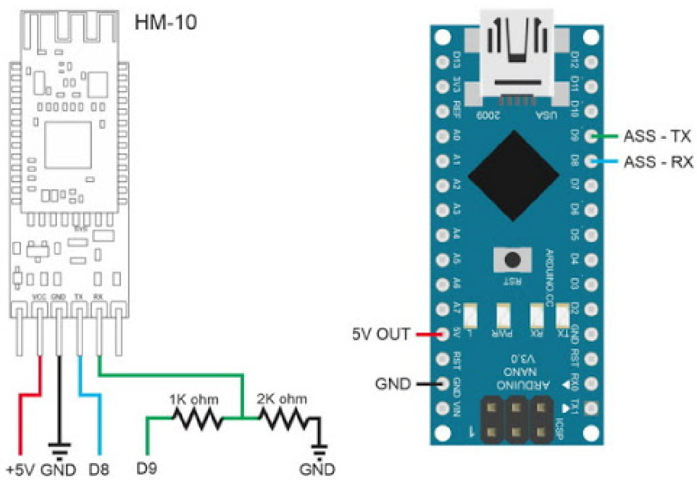


Figure 4.2: Position of Voltage Regulator for HC 06

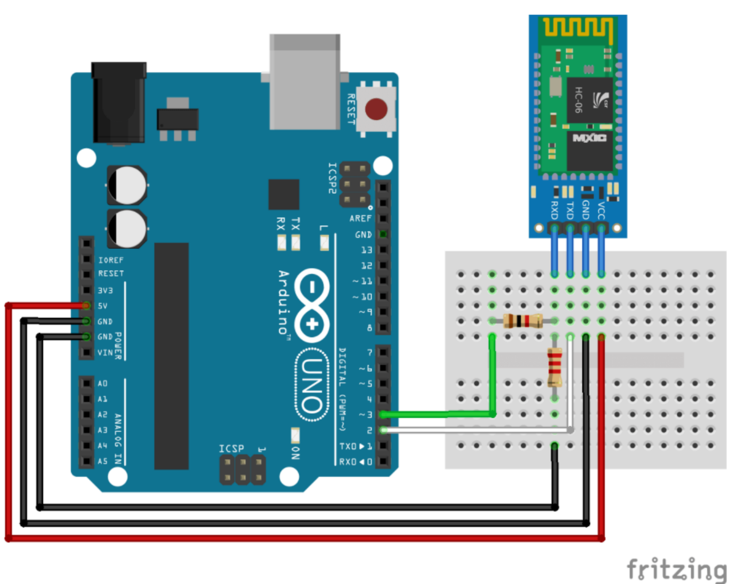


Table 4.1: Comparison of Arduino Bluetooth Module

Bluetooth Module	**Configuration**	**Regular Bluetooth /BLE (Bluetooth 4.0)**	**Well Documented?**	**Price**
HC 05	Master/minion	Bluetooth	yes	$7.99
HC 06	Minion only	Bluetooth	yes	$8.49
HM 10	Master/minion	BLE (Bluetooth 4.0)	yes	$10.99
HC 08	Master/minion	BLE (Bluetooth 4.0)	no	$7.99

In summary,

- After doing research on Wifi Vs Bluetooth, we decided on Bluetooth
- When looking at Standard Bluetooth vs BLE, we concluded BLE would be best for saving battery life, but compared BLE and standard Bluetooth in case of major price or feature differences
- HC 05 and HM 10 were considered the two best options when looking at documentation, configuration, and price
- The HM 10 was ultimately selected since expected battery life would be significantly increased

4.1.2 Temperature and Humidity Module

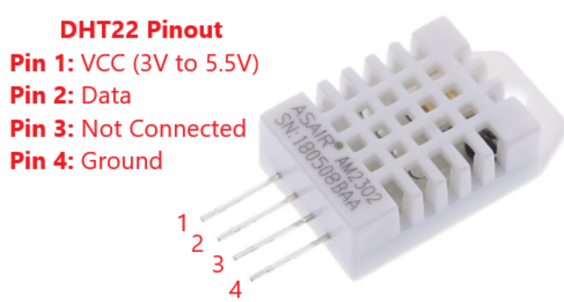
In looking at which temperature and humidity sensors to use in our design, we firstly wanted one component that could measure both temperature and humidity, so that we could save space in our PCB design. We also wanted our sensor to be compatible with Arduino. Both the DHT11 and DHT22 met these initial constraints. Upon comparing the two, the DHT22 has higher temperature and humidity range and a lower margin or error for both readings. The DHT11's only advantages are price and its sampling period of one second, while the DHT22 only samples every 2 seconds and is about twice as expensive. Since the upper limit of the DHT11's humidity range is 90% humidity, we can expect that the humidity in and around our sponsor's warehouse, located on the Florida Coast, will regularly be above 90%. Since we only plan on taking one reading from each sensor per minute, the difference in sampling times doesn't affect our project.

Table 4.2: Comparison of DHT11 and DHT22 Sensors

Parameter	**DHT11**	**DHT22**
Temp Range	0-50(C)	-40-80(C)
Temp Accuracy	+/-2(C)	+/-0.5(C)
Humidity Range	20%-90%	0%-100%
Humidity Accuracy	+/-5%	+/-2%
Sampling Time	1s	2s
Price	~$5/each	~$10/each

Despite needing multiple sensors and therefore increasing the total difference in cost, the DHT22's parameters far exceed those of the DHT11 in all viable areas of concern for our project, so we will use the DHT22 in our design. The DHT22 has four connection pins, but only three of them will be connected in our build, as shown in the pinout diagram below.

Figure 4.3: DHT22 Pinout



In summary,

- The DHT11 and the DHT22 were looking at over other temperature and humidity sensors since it senses both in one device and is well stocked and documented
- The DHT11 is better in terms of price and data collection rate
- The DHT22 is better in terms of sample range and accuracy, which were determined to be more significant constraints in our design
- The DHT22 was chosen since the goal is to design for human comfort and energy efficiency when cooling the warehouse, and since Florida humidity is regularly over 90%

4.1.3 Microcontroller

Each sensor unit will require a microcontroller to be able to read data from the temperature/humidity sensor, and then forward the information to our main control unit via wireless communication from the connected Bluetooth module. Since the main control unit will be using an Arduino Mega for the microcontroller, we will also use an Arduino Nano to make the programming compatibility easier. Looking at the Arduino library, the Arduino Nano Every is both the cheapest microcontroller option, and has all the pins needed for the HM 10 and the DHT22. Since we are planning on using a separate bluetooth device to better connect to the controller's bluetooth and avoid software conflicts, we don't need to worry about the sensor's microcontroller having built-in bluetooth.

Between the Nano Every and the Nano every with Pin Headers, all the specifications are the same. The only difference is that the Nano Every would need to be soldered to its pin headers and is $2 less per unit. Additionally, Arduino sells the Nano Every in multipacks, so we can purchase multiple for a reduced bundle price. Finally, soldering is an important skill for electrical engineering, and with only 30 pins per board, the time commitment in soldering enough for each sensor is not limited. For these reasons we chose the Nano Every as our board, with the understanding that the Every with Pin headers is acceptable if needed (eg, the Nano Every is on backorder or soldering supplies cannot be obtained.

Figure 4.4: Arduino Nano Every Pinout

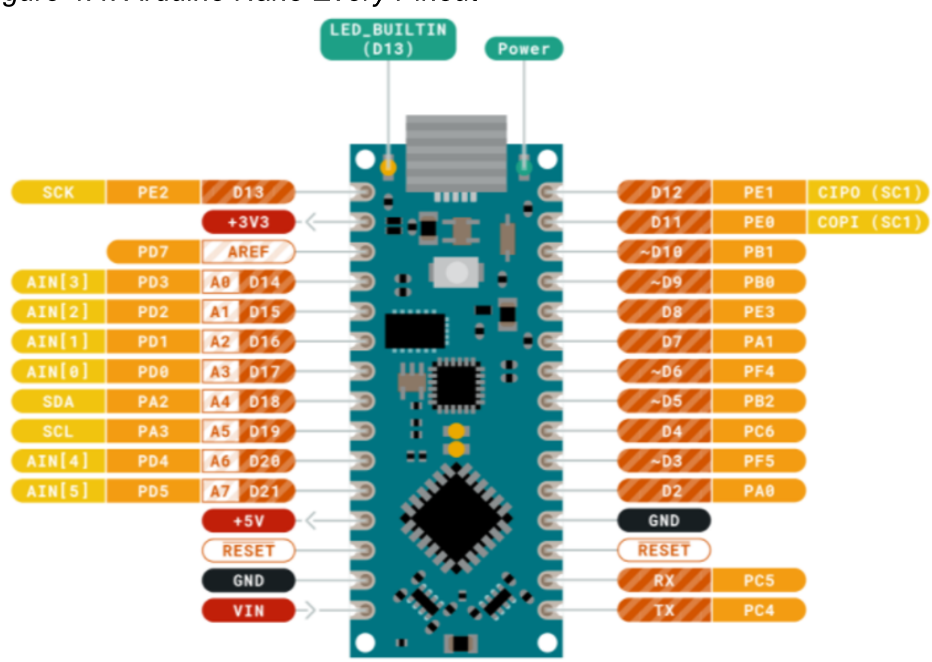


Table 4.3: Comparison of Microcontrollers

Arduino Microcontroller	Contains all Pins required?	Comes with pin headers?	Bluetooth /BLE embedded	Price
Nano Every	yes	Not soldered	no	$10.90
Nano Every with Headers	yes	yes	no	$12.90
Nano 33 IOT	yes	Not soldered	Bluetooth	$14.70
Nano 33 BLE	yes	Not soldered	BLE	$20.20

In summary,

- As discussed in our research section, Arduino products are the only microcontrollers considered in our design.
- Looking at the cheapest arduino products, all of then had enough pins for our sensor design, so all were viable options
- Since we have already chosen for the sensors to use the HM 10, the Nano 33 IOT and the Nano 33 BLE do not provide any significant advantages.
- Between the Nano Every and the Nano Every with Pin headers, we liked that our soldering skills would be improved and we can bundle the Nano Every, so the Nano every was selected.
- In the event that the Nano Every becomes unavailable, the Nano Every with Pin Headers can be used at a slightly higher price point.

4.1.4 Battery and Battery Housing

Each sensor module requires power. The sensors need to be battery powered so that they are easy to move around the warehouse. Since we are planning on taking temperature readings approximately once per minute, we don’t need a battery meant for a high-drain device. We are instead looking for the longest lasting battery that will give the required input voltage for the arduino nano every that the sensors will be using (7-21V).

In looking at batteries, we studied the specifications of Google’s Nest Home sensor and found that it used CR2 batteries. Looking into these further, we found that CR2 batteries can hold charge for up to 10 years in storage, and had a practical life of 800mAh (milliamp hours), which means they should last about two years with a constant .05mA drain. Since our sensors will be off the majority of the time and will only be sending data once every minute or so, and the BLE connection we have picked only uses 9mA when online and .05mA when asleep, we can expect that our batteries will need to be replaced approximately every year and a half since the majority of the time they will be in sleep mode. Unfortunately, CR2 batteries only have a 3 V charge, and the arduino nano every that we plan on using requires a power input of 7-21V this means that we would need 3 CR2 batteries in series to provide the needed voltage, and all the batteries and their housing would require us to have a larger PCB design, making the overall sensor bigger.

We then looked at using a 9V battery. We know that a single 9V battery would be within the range of our microcontroller, so our most important factor in choosing a 9V is practical life. Since the practical life of an alkaline 9V battery is about 500mAh, that would be giving us worse battery life than the CR2 batteries. However, lithium 9 V batteries have a practical life of approximately 1200mAh, making them a better alternative to the CR2 in space and lifetime. Additionally, since we would have needed 3 CR2 batteries per sensor, the price of the batteries is about equivalent. We will also need fewer battery housing stations, which will decrease the price further, by about $4 per sensor.

Table 4.4: Battery Comparison Table

Battery	**CR2**	**Lithium 9V**
Voltage (V)	3	9
Number needed to meet Voltage requirement	3	1
Practical life (mAh)	800	1200
System Cost	$13.50	$6.89

For our battery casing, our two biggest options were a simple clip connector or a case to connect into our PCB design. The case option is preferred since our sensors will be wall mounted, so we cannot rely on gravity to keep the batteries in place. The cases cost slightly more, at about $2 each, but will allow battery connection without straining the connection wires.

In summary,

- CR2 Batteries were initially considered since they are used in other low energy smart home devices
- The voltage requirement for the Arduino Nano Every would necessitate a minimum of 3 CR2 batteries, which would take of the majority of the space on our PCB, and would be expensive
- A single lithium 9V battery can be used to produce similar battery life, and would be smaller and cheaper than using 3 CR2 Batteries
- Ultimately, the 9V battery was the clear choice for our sensor units and was selected

4.1.5 Casing for PCB

Each sensor module, once fully assembled, will need a protective casing to keep the parts from being damaged. The casing for the sensor module should ideally be lightweight, sturdy, and subtle to the eye. Additionally, the whole sensor should be lightweight enough to be wall mounted with a Command Strip, so that our users can put the sensors wherever they find them to be most useful and have them out of the way. Finally, the DHT22 will need to accurately measure the temperature and humidity, so it requires open airflow. We plan to use our sponsor’s 3D printer to create a snap-together case with a solid back to mount the PCB on, and a raised lattice cover, protecting the sensor from accidental harm and making the sensor more appealing to look at. Each sensor should aim to be no more than 4” x 6”,about the size of a large postcard.

4.1.6 Sensor Placement

In order for our sensors to be most efficient, they will need to be placed in such a way where they can accurately detect the temperature and humidity of the area. The warehouse space we are trying to alter the temperature of is approximately 5,500 square feet. We expect to be placing 4-5 indoor sensors. This should give a better temperature and humidity spectrum reading of the full warehouse space as opposed to only getting one indoor reading for the entire warehouse.

Wall placement for the sensors is preferred, since a large portion of the warehouse’s floor space is in use for local robotics practice zones and therefore is sectioned off. Wall placement keeps the sensors out of the way and also at a height that easily measures the effective temperature of the space from a human perspective. Since heat rises, placing the sensor on the ceiling near the vent would give a reading a few degrees warmer than the warehouse floor. Wall placement at human height also allows easy battery replacement.

Additionally, various machinery is in use in the warehouse and creates heat pockets. Because of this, our team needs to make sure we can account for both general floor temperature and pockets of heat from machine use. Sensors should be placed around the warehouse, far enough apart that they can account for the full space. One sensor should be placed in the corner where the machinery is to detect if it is currently creating a pocket of heat. Another sensor should be placed on the outermost wall of the command room, since that is the closest to the center of the warehouse we can place a sensor while still wall mounting it.

The remaining sensors should be placed evenly between these points on the wall. Both the front and back of the warehouse have entrance points that could influence data, a human door in the front for regular entrance, and a garage door in the back to allow the transfer of machinery. We cannot place our sensors on the doors, and placing them too near could give off data influenced by outside temperatures. For the front door, we expect entrances and exits to be quick, while we expect that the garage door will be open for longer periods of time, but opened less frequently overall. The garage door is very close to the machinery corner, and having it open would help cool the space even more due to the chimney effect.

In summary,

- Four or five temperature and humidity sensors will be used to allow data collections from multiple areas
- Sensors should be evenly spaced throughout the warehouse.
- Wall placement is preferred to keep the sensors out of the way
- Sensors should be placed at human height
    - Gives temperature readings where people will be located and Simplifies battery replacement

4.2 Main Control Unit

This section covers research directly related to the main control unit and the parts selection based on the research. This section begins with the easier and more generic parts research such as the bluetooth and wi-fi modules. This section then moves on to cover research on more complex components that have not been discussed yet in this paper, such as the central processing unit.

4.2.1 Wi-Fi Module

The main control unit will also need to connect to the internet to allow for remote control of the unit while the sponsor is not inside the warehouse. In order to accomplish this, the main control unit will need a WI-FI module. Due to its popularity among Arduino users, an integrated ESP8266 ESP-01 Wi-Fi Module chip will be the link between our main control unit and the internet using Wi-Fi radio signals. One of the key benefits of the ESP-01 is the plethora of information available for its use and programming which will make it a very useful tool to make the main control unit part of the Internet of Things.

The biggest drawback to this Wi-fi module is its power consumption. During normal operation this module draws around 80mA. However, since the main control unit will draw its power from a hard wired line and not from a battery, the power consumption will have a minimal effect. The ESP-01 is powered up using a 3.3V power source, and an Arduino Mega will be able to power up the ESP through its regulated 3.3V power pin.

Table 4.5: ESP-01 Wi-fi Module

	ESP-01 Wifi Module
Price	$16
Power Consumption	80 mA
Ability to connect to the internet	Yes
Operating Voltage	5 V
Data Transfer Voltage	3.3 V
Data Transfer	UART

4.2.2 Bluetooth Module

The main control unit will communicate with multiple sensors by using its own bluetooth module. The bluetooth module that was selected for the sensors was the HM10 due to the power saving benefits of the module. The HM10 is a serial BLE module (Bluetooth-Low-Energy) which is intended to be used for the low power consumption applications and can last long even with a coin-sized battery. The HM10 is a popular Bluetooth 4.0 based module that comes with a serial/UART layer which makes the device able to interface with different microcontrollers. Like the Wi-Fi module, the HM10 can also be programmed using AT commands sent over the serial UART connection, which makes the HM10 ideal for creating simple connections.

The HM10 module can be set in both master and slave mode. In order to communicate with the slave mode wireless sensors, the main control unit will need its bluetooth module set as Master to control communications. For this reason the HM10 bluetooth module was selected for the main control unit. The HM10 placed in the main control unit will use the same bluetooth protocol that the wireless sensors will use. The HM10 bluetooth module also draws a minimal amount of current with a normal value of 8.5mA. The reduced power consumption makes a nice opportunity for future expansion if the reliability of the power supply needs to be increased with a battery backup option. The only downside to using the HM10 bluetooth module is the fact that it uses Bluetooth 4.0 protocol to send and receive data. Bluetooth 4.0 is not compatible with earlier versions of Bluetooth and all future bluetooth connections for any additional add-ons will need to also use the HM10 bluetooth module.

Table 4.6: HM-10 Bluetooth Module

	HM-10 Bluetooth Module
Price	$11
Range	50 meters
Power Consumption	8 mA
Bluetooth 4.0 compatible	yes
Sleep Mode Compatible	yes
Data Transfer	UART
Wireless Connection Modes	Master/Follower

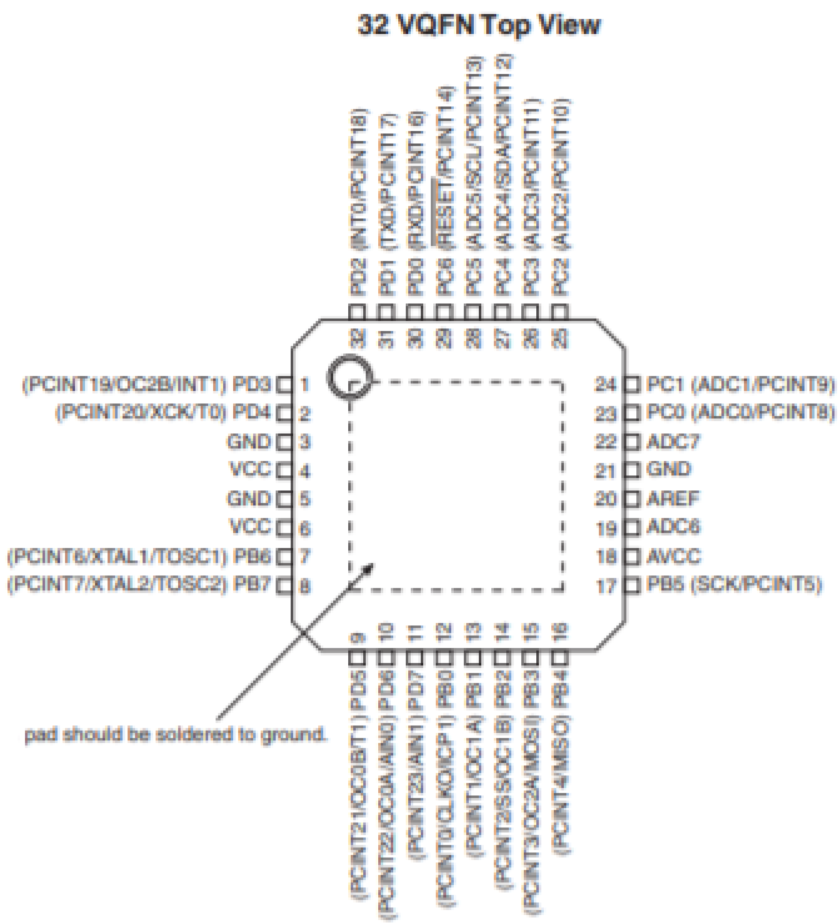
4.2.3 CPU Module

The brains behind the main control unit comes down to the CPUs inside arduinos 2 most popular Circuit Boards, the ATmega328P and the ATmega2560. These two microcontrollers have gained popularity over the years due to their lower cost and easy to use integrated development environment (IDE). Arduino microcontrollers were chosen for this project due to their increase in use by popular tech companies, and due to their huge library of projects that will aid in the completion of this project. Arduino also has a massive host of third party companies that can supply parts cheaply and come with the necessary drivers to interface with an arduino board.

The first microprocessor to consider is the ATmega328P CPU which is featured in the Arduino Uno series. The ATmega328P is a small CPU that features 32 pins. It's first in consideration due to its popularity among circuit designers and is great for simple tasks with IoT devices. However, the limited number of pins will be a huge barrier when it comes to the ability to drive the larger touch display along with the Wi-fi and Bluetooth Modules. The limited number of pins would also create difficulty when trying to connect the four, four channel relays needed to control the vent motors.

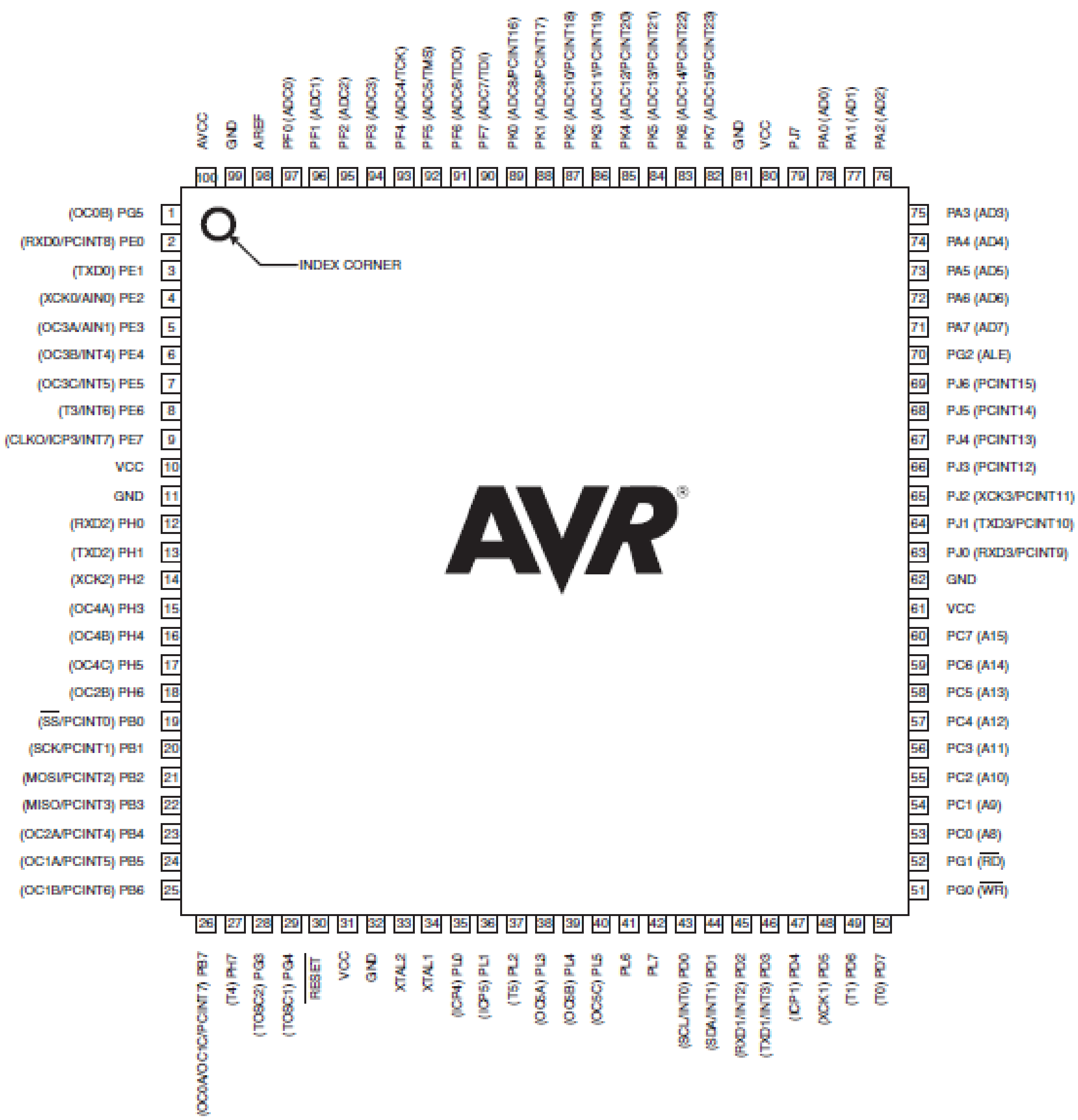
The second CPU to consider is the ATmega2560. The ATmega2560 has 100 pins. This robust chip will be well suited for driving the larger touch display along with having four transfer and receiving pins that will simplify the UART communication needed for the bluetooth and wi-fi modules. The extra input/output pins would be ideal for driving the four, four channel relays, and would leave extra room for future extensibility if needed by the sponsor. For these reasons the ATmega2560 was chosen as the CPU for this project.

Figure 4.5: ATmega328P Pinout Diagram



As seen in the figure of the pinout diagram for the ATmega328P. This is a very popular microchip due to its more simple design, and is great for the execution of simple tasks. However, this microchip only has thirty two pins. Of those thirty two pins, only twenty three pins are designed for Input/Output operations. The twenty three pins will not be enough to drive the larger touchscreen display as well as receive inputs from Bluetooth and Wi-fi modules. For this reason the ATmega328P was not selected for use in our project.

Figure 4.6: ATmega2560 Pinout Diagram



Above is the figure of the pinout diagram for the ATmega2560. From first glance this CPU clearly is more robust and offers one hundred pins. Of these one hundred pins, eighty six of them can be used for input/output operations. These eighty six pins will be more than enough for driving the large touch display for this project as well as sending signals to open and close the relays, and receive data through the wi-fi and bluetooth modules. This microchip offers four USART connections which will allow for the addition of another bluetooth module in the future if it is needed. It is for these reasons the chosen microchip for the project will be the ATmega 2560.

Table 4.7: Comparison of ATmega328 and ATmega2560

	ATmega328	**ATmega2560**
Number of Pins	32 Pins	100 Pins
Operating Voltage	5 V	5 V
Arduino IDE Compatible	Yes	Yes
Number of RX/TX pins	1	4
Satisfy Design Requirements?	No, not enough pins	Yes

4.2.4 Display Interface

The concept of a GUI is essential to not only present results from the indoor sensors to a display, but to also allow the user to manage some controls in a user-friendly way. We propose to practice software design on an *Arduino Mega 2560* in conjunction with a *TFT 3.5" Color Touch Screen Display*. Evidently, we will take advantage of the Arduino IDE to work on controlling Arduino. The TFT display screens utilize the UTFT and UTouch libraries. These libraries work with many different TFT screens sizes, shields and controllers, and thus provide the easiest use of programming the user interface. These libraries can be downloaded from RinkyDinkElectronics.com, in addition to demo examples and detailed documentation of how to use them. After including the libraries, UTFT and UTouch objects will need to be created along with parameters that depend on the model of the TFT Screen and Shield and these details can be also found in the documentation of the libraries.

Fonts and variables will need to be defined, and the program will need a setup section. This is where we would initiate the screen, the touch, define the pin modes for the connected sensors, and some custom functions to create the drawing of the program's home screen for the controller.

Table 4.8: Touch Screen Display Summary

	3.5" TFT Touch Screen Display
Size	3.5"
Screen Type	TFT
Touch Compatible	Yes
Touch Type	Resistive
Operating Voltage	5 V
Data Transfer Method	8-bit
Backlight Control Method	Pulse Width Modulation

5. Design Constraints and Standards

In order to successfully implement our design, this section describes technical and logistical challenges presented to our group, and considers constraints that are associated with the design and production of the ventilation controller. This section also describes any standard we have to consider that could have an impact on this project.

5.1 Remote Constraints

The warehouse we will be implementing our design in is in Niceville, Florida. However, the majority of our team currently resides in Orlando, Florida. We anticipate that all our team members will still be living in Orlando during the Fall 2021 semester, which is when we intend to build and install the system for automating the vents. This will require the team to make at least one trip to Niceville during Fall 2021 to install the system, though the team is aware that more trips may be required. Additionally, Covid-19 has given us a logistical challenge in that we cannot physically meet to work on this project until Fall 2021. We plan to address this by ordering select parts early so that we can individually familiarize ourselves with the coding and assembly process as early as possible.

5.2 Environmental Constraints

We face the challenge of the warehouse being made of metal. This causes some difficulty when it comes to working with Bluetooth connectivity from the indoor controller to the outdoor sensor. In order to account for this, we had intended to hardwire the outdoor sensor into the building, but instead decided to scrape local weather data instead.

5.3 Sustainability Constraints

Regarding software, we predict to encounter complexities dealing with the implementation of Wi-Fi and Bluetooth connectivity. Thus, we propose to begin familiarizing ourselves with the application of both models as early as possible to avoid any bottlenecks in the fall. In addition, we also need to have an automated programmed system that tells the connected vent dampers when to open, when to close, and what orientation they should adjust to, which ultimately requires an understanding of thermodynamics and air circulation. Consequently, we expect to do a significant amount of investigation to be able to program the most efficient logic commands for the vents.

Another sustainability constraint this project faces is due to semiconductor shortage that the world is facing. Recent news reports have shown that many industries are facing a shortage of silicon based parts due to a production shortage caused by the COVID-19 pandemic. These shortages have and will continue to impact the testing of our project. While designing the power supply for this project texas instruments has sold out of all buck converters that can be used to reduce the 12V supply voltage to a usable 5 volts. If these shortages continue DC voltage reduction may not be possible for implementation in this project. A back up method of using 120 V AC as the supply voltage may be necessary to keep this project on time.

5.4 ISO/IEC/IEEE 29148-2011 Standards

The Institute of Electrical and Electronics Engineers (IEEE) is a professional association responsible for developing standards in a broad range of technologies. The ISO/IEC/IEEE 29148-2011, known as Systems and software engineering - Life cycle processes - Requirements engineering, which specifies the required processes that are to be implemented for the engineering of requirements for systems software products throughout the life cycle. In use of this standard, we have to acknowledge and gather the appropriate requirements as well as document them, validate them and provide a basis of verifying designs and accepting solutions.

This standard provides us with a guideline to define well formed requirements for our software requirements, which allows us to coherently provide a way to distinguish between the requirement and its attributes. It also provides a list of characteristics that each individual requirement should have, like it being necessary, which means that if this particular requirement is removed, then there will be a deficiency that will exist. It also provides a list of characteristics to consider when evaluating a set of requirements, like it being consistent, meaning that there are no duplicates, and the same terminology is used for the same item throughout all the requirements. After gathering all the requirements and distinguishing between actual requirements and requirement attributes, the standard also provides an iterative application of processes, which is a way to apply and check to see if there were any other requirements that were missed and be able to transform the requirements into a technical view of the product, that will be able to fulfill all the requirements. We will make sure that our sponsor is involved in making sure all of his requirements are met, in a similar format as the ones detailed in this standard.

After defining the analysis requirement process, we are introduced to the requirements in architectural design, which gives guidelines on how to analyze and evaluate the architecture as well as defining what the architecture will be. In regards to this project, I believe there will be two architectures, the PCB and the website framework. Following this, the standard defines what a good verification method consists of and what it should address, it also specifies four standard verification methods used to obtain evidence that the requirements have been met. These four methods are: inspection, analysis (including modelling and simulation), demonstration, and test. Inspection examines the architecture using sight, hearing, smell, touch, and taste. It is more of a visual compare and contrast on what was designed as a requirement and what it ended up being. Analysis utilizes simulation and analytical data as a verification method. Demonstration deals with the functional performance and what it is able to accomplish, so in regards to our project it will probably look like testing the website functionality and then testing the hardware to be able to do the same functionality as the website based on the touch of the user on the user interface attached to the PCB. Finally, the last method, test, is checking the performance capability of an item in controlled conditions that are simulated and real world conditions too. Which are all good methods that our team will have the ability to use once our design is created and completed.

Finally, in the standard there are different outlines provided to bring the document all together, with the example outline for the software requirement specification being at the end. And also provides examples of organizational approaches to the requirements in the software requirement specification. Which will be a great outline for us to utilize once we start developing the system and documenting it.

6. Hardware Design Details

This section covers the design methodology and wiring requirements of the hardware components on both the main control unit and wireless sensors. This section is crucial in deciding the requirements for each component, since many of these components need different voltage levels for different pins. In this section, we will lay out the basic connections needed for each component, and how they will be laid out to form a cohesive unit that works in unison with one another.

6.1 Main Control Unit Design

The main control unit is responsible for collecting all the data from the sensors as well as commands from the local display and website interface. The main control unit then processes all the data to turn the relays on and off which in turn opens and closes the roof vents. The main control unit will have the benefit of getting its power supplied from a hard wired line. The hardwired line will come from the 12V DC power supply that is used to drive the vent motors that are already installed at the location. This line was chosen to reduce the overall number of components needed for the Main Control Unit. Since the voltage source is already 12V no full bridge rectifier or any other frequency correcting components is required. The 12V only needs to be reduced to 5V to make it usable for the main power source of the main control unit.

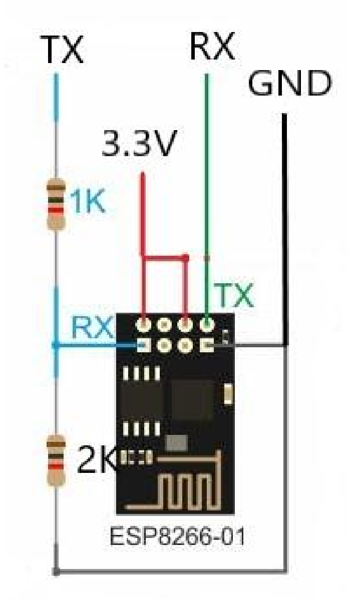
6.1.1 Wi-Fi Module Wiring

The ESP8266 can't connect to the Arduino through TXD, RXD, GND, and the VCC pins to the RX, TX, GND and 3.3 V pins, respectively. The RXD of the ESP-01 needs to receive a signal of 3.3V logic and not 5V. If this were to happen, the ESP-01 received signal will malfunction. Thus, the TX received from the Arduino can be connected to a voltage divider. The voltage divider will produce an output voltage that is a fraction of the input voltage. We will use the voltage divider between the TX of the Arduino and RXD of the ESP-01 to produce an output voltage of 3.3 V. THe ESP8266 can draw a lot of current during its normal operation. To supply the power required for this module the 3.3V will come directly from the power supply via a voltage divider to drop its voltage to the required 3.3V.

In summary,

- The ESP8266 transmits and receives its data via the TX and RX pins.
- The ESP8266 requires a 3.3V logic level therefore it will need a voltage divider to reduce the voltage on its RX pin.
- The ESP8266 requires 3.3V for power.
    - During prototype testing it will get its power from a 3.3V pin
    - Final design will power it straight from the power supply via a voltage divider
- Prototype testing still continues, Once it is completed the final pin layout will be decided

Figure 6.1: Wiring for Wi-fi Module



6.1.2 Bluetooth Module Wiring

One of the main drawbacks to all of the Bluetooth modules we researched is their logic voltage of 3.3 V. Directly plugging them into the arduino board may cause serious damage to the bluetooth module itself due to the five volts the arduino board puts out. This damage could burn out the board completely making the bluetooth communication impossible. To ensure this does not happen a voltage divider will be implemented to reduce the input voltage to the board. The voltage divider will consist of a one kilo-ohm resistor placed between the five volt power supply and the board and a two kilo-ohm resistor placed between the board and a ground pin. This will reduce the voltage supplied to the bluetooth module to

3.3V allowing for safe operation of the bluetooth module. However, the voltage divider is not required for all connections. The only connections that require a reduced voltage are the receiving pins of the UART data communication and the VCC pin.

$$Vout = Vin \cdot \frac{R2}{R1+R2} \Rightarrow 5V \frac{2k\Omega}{1k\Omega+2k\Omega} = 3.33V$$

Another potential method for connecting the bluetooth module to the main control unit will be through input/output pins. This may be necessary due to the fact that the bluetooth connection is one to one. This connection makes it difficult when trying to read data from multiple sensors. To connect a bluetooth module to a different sensor the first connection must be severed before any new connection can be established. The only way to get a bluetooth module to connect to a new device is through ATcommands. To get the bluetooth module into ATcommand mode the Master device must be shut off completely and restarted. Once the device is restarted the code can send ATcommands to the bluetooth module to get it to pair with another sensor. However, If the bluetooth module is connected to a 5V VCC connection then the power cannot be shut off to the module and the new connection cannot be established. This is where the input/output pins come into play. Instead of connecting the bluetooth module to a VCC pin we can connect it to a I/O pin to supply its power. Since we chose to use a module that utilizes bluetooth low energy protocol, an input/output pin should be more than enough to supply its power. When the bluetooth module needs to turn on we can simply drive the pin high to VCC. When the bluetooth module needs to be turned off we can simply drive the pin low to ground. This can also be done to the transfer pin on the printed circuit board to completely electrically isolate the bluetooth module. The wiring for these pins will still contain the voltage divider required to reduce the voltage to a usable level for the bluetooth module.

In summary,

- This bluetooth module draws a small amount of power, and can be powered off a normal pin during prototyping and final design.
- A 3.3V logic level is required when transmitting data to its RX pin. A voltage divider will be implemented to reduce that voltage.
- The bluetooth module may need to be shut down prior to establishing new connections with the different sensors.
    - This can be accomplished by driving a digital pin low to ground.
- Once the bluetooth module has been shut off and turned back on it will respond to AT commands to establish a new connection.
- Prototype testing still continues on the bluetooth module.
- Once prototyping is completed the final pin locations will be decided.

6.1.3 Relay Module Wiring

The backbone of the main control unit will be the bank of relays. These will be wired to control the 12V DC vent motors that are already installed inside the warehouse. The relays will need to be wired in a way that allows the DC motor to run forward and reverse. The relays chosen for this will be the Elegoo 4-channels relay unit. These relays were chosen due to their popularity and price point. The 4 channels will be run in pairs and each pair will control a direction of travel for the DC motor. The current warehouse contains four vents each with their own motors. A four-channel relay module will be required for each motor, making a total of 16 relays. Each four-channel relay module requires six wires, one for 5V VCC, one for ground, and four wires to be connected to input/output pins which will control the operation of the relays.

Figure 6.2: Double Pole Double Throw (DPDT) Switch Wiring

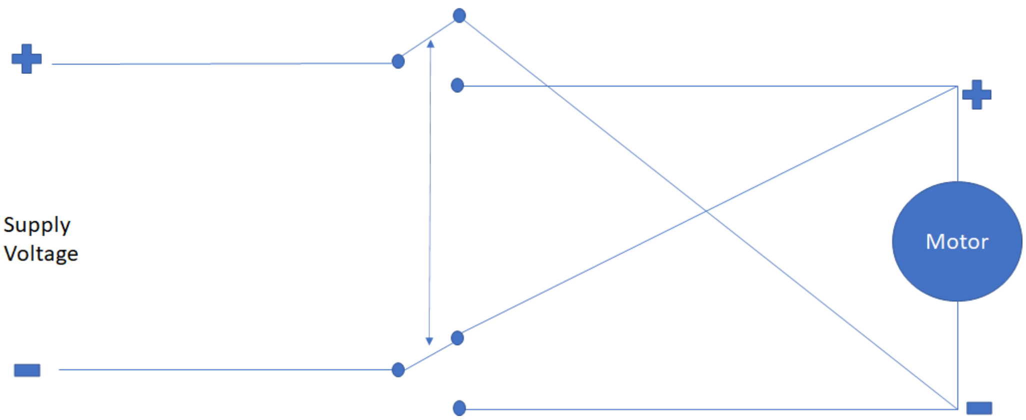
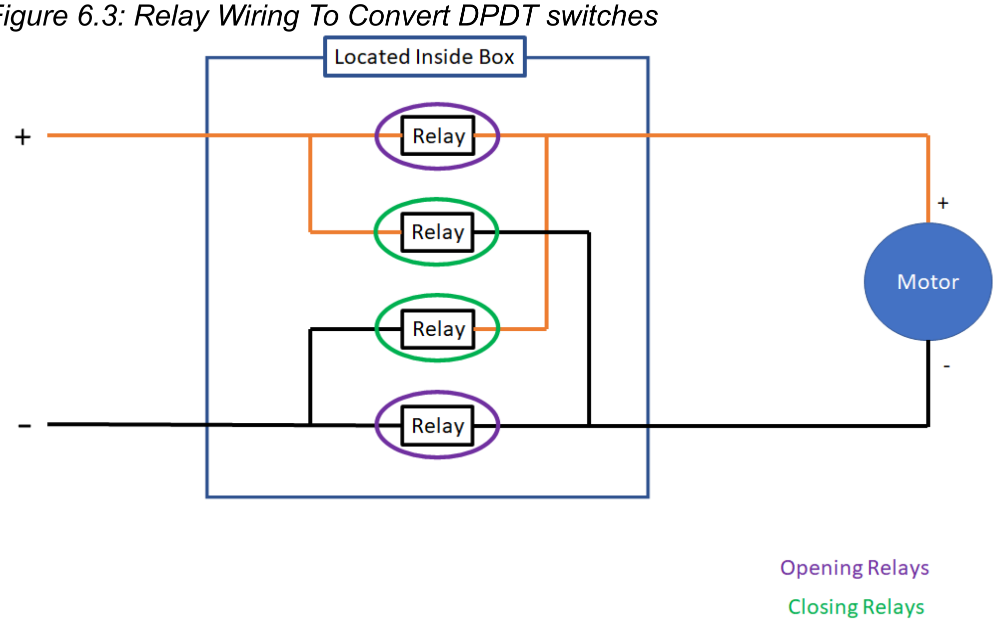


Figure 6.3: Relay Wiring To Convert DPDT switches



In summary,

- Relay module will be implemented to replace the double pole double throw (DPDT) switches that are still currently installed.
- The relay modules will isolate the main control unit from the 12V source required to drive the vent motors.
- Four relays will be required to replace a single DPDT switch.
- Our implementation will require 16 relay modules to replace all four DPDT switches.
- Testing on the relay modules has been completed.

Table 6.1: Relay Pinout Connections

	Relay Pin	Prototype Pin	CPU Pin
Relay module 1	IN1	30	60 (PC7)
	IN2	31	59 (PC6)
	IN3	32	58 (PC5)
	IN4	33	57 (PC4)
Relay module 2	IN1	34	56 (PC3)
	IN2	35	55 (PC2)
	IN3	36	54 (PC10
	IN4	37	53 (PC0)
Relay module 3	IN1	38	50 (PD7)
	IN2	39	70 (PG2)
	IN3	40	52 (PG1)
	IN4	41	51 (PG0)
Relay module 4	IN1	42	42 (PL7)
	IN2	43	41 (PL6)
	IN3	44	40 (PL5)
	IN4	45	39 (PL4)

6.1.4 Touch Display Wiring

The touch display will be connected to the main control unit via wires. This was done due to the fact that the touch display will be mounted to the cover of the electrical box. The touch display will require sixteen wires to transfer the data, receive power, and receive signals from the touch screen. These sixteen wires will have a disconnect in them which will allow the cover to be removed without any strain in the wires or pins. The screen can transfer and receive its signal via a SPI connection or eight bit connection. The eight bit connection was chosen due to the faster transfer speeds. The faster transfer speeds will enable the screen to have a faster response time than with the SPI connection. The downside of using the eight-bit connection is the increased number of input/output ports required to transfer data to the touch screen. However, this issue has been resolved due to the selection of the ATmega2560 CPU which will have more than enough pins to make all the necessary connections.

Table 6.2: Display Pinout Connections

TFT Screen Pins	**Prototype Pins**	**CPU Pins**
3-5V	5 V	5 V
GND	GND	GND
D0	22	78 (PA0)
D1	23	77 (PA1)
D2	24	76 (PA2)
D3	25	75 (PA3)
D4	26	74 (PA4)
D5	27	73 (PA5)
D6	28	72 (PA6)
D7	29	71 (PA7)
CS	A3	94 (PF3)
C/D	A2	95 (PF2)

WR	A1	96 (PF1)
RD	A0	97 (PF0)
RST	RESET	30 (RST)
Y-	A4	93 (PF4)
Y+	A6	91 (PF6)
X-	A5	92 (PF5)
X+	A7	90 (PF7)

In Summary,

- The connection between the main control unit and the screen will be made with wires.
    - This was done to facilitate removing the box cover when required.
- The connection for the screen will be 8-bit data transfer.
    - This was done due the increased data rates for the 8-bit connection.
- Display testing has been mostly completed.
- Touch screen testing is still underway.

6.1.5 Main Control Unit Housing

The main control unit will be housed in the Cantex box that is already installed onsite, at the warehouse. The Box has 120V AC coming into the box through the bottom port hole. Inside the box a 12V DC converter is already installed. The 12V DC electrical cables then connect to the double pole, double throw switches which are mounted to the box cover. The 12V DC power lines that control supply power for the vent motors leave the cantex box out of the top port holes.

For our installation the 12V DC power converter will remain in place. The front cover will be removed and replaced with a new front cover. This will be done because of the holes from the double pole, double throw switches. Those holes will not be electrically safe and will create an eyesore that will detract from the overall project. A rectangular hole will be cut in the center of the new cantex cover to allow for the touch screen to be surface mounted to the cover of the cantex box. The printed circuit board of the main control unit will be mounted behind the touch screen display. There will be wires, with a disconnect, that will connect the PCB to the touch screen display to allow for the cover to be removed and replaced with ease. The relay module will sit above the PCB to provide easy access for thicker wires that are needed to control the vent motors.

In summary,

- The housing for the main control unit will be the Cantex box that is located on site
    - This was done because of the permanent connections and power source that are located inside the box
- The location of the power supply inside the box will remain unchanged
- A new front cover will be used due the holes from the double pole double throw switches.
- A square hole will be cut in the new cover and the screen will be mounted to the front cover
- The PCB of the main control unit and the relay modules will be mounted inside the box.

6.1.6 Power Supply

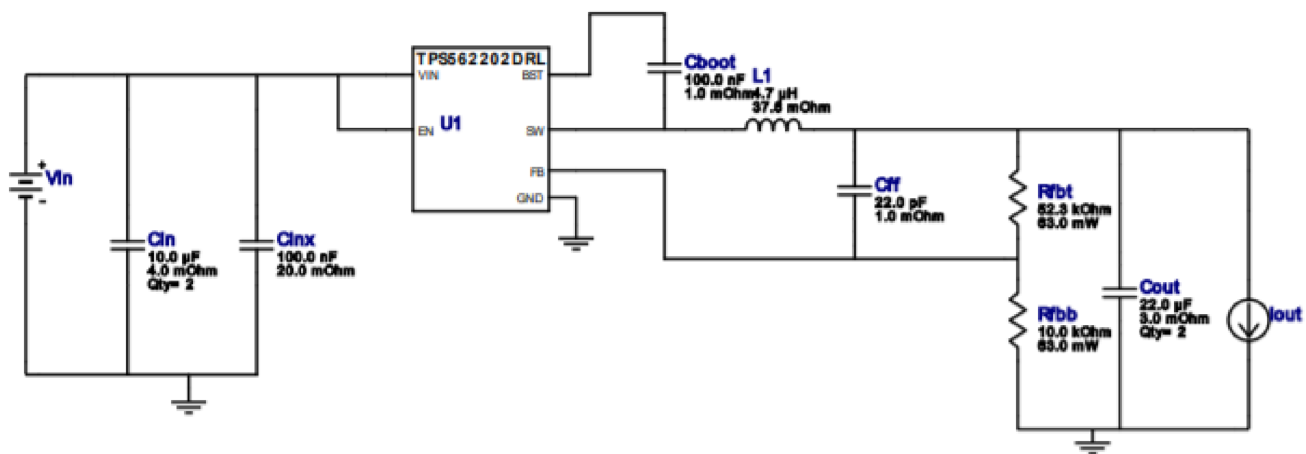
The power for the main control unit will be provided from the power source for the motors. This was decided to reduce the cost and plug availability for the area. The switch box that is currently installed in the warehouse is plugged into a standard north american 120V AC plug. The AC was stepped down to 12V then converted to DC using a basic full bridge rectifier. This 12V DC source that is provided in the box provides a perfect power source. To make this power source usable the 12V DC will need to be stepped down to 5V DC which can be accomplished with the Texas Instrument TPS562202 and using a voltage divider circuit to achieve the required voltage. The power supply can run on voltages varying from 7V to 17V which will account for any power drops that may occur during the motor operation. Capacitors will be located at the input to reduce any noise from the power supply and filter out any voltage spikes or drops that may occur. The output will likewise have a capacitor to filter out any noise along with an inductor to reduce any sudden changes in the current.

The main power supply may need to be revised in the future due to supply chain shortages. Currently all Texas Instrument buck converters are sold out due to the semiconductor shortage. However, There are many alternative options to the buck converter. One such option is using a linear voltage regulator to power reduce the voltage to a usable 5V. The linear voltage regulator is not as efficient as the buck converter and will require the addition of a heat sink to ensure it will not fail prematurely.

In summary,

- The source of power to the power supply will be 12V DC supplied from the vent motor power supply.
  - This was chosen to reduce the number of parts since there is no need for rectification.
- The power supply was designed using Texas Instrument Webench.
- The current power supply uses a buck converter to reduce the 12V DC to 5V DC.
- An alternate power Supply was designed due to supply constraints of buck converters.
  - Alternate supply uses a linear voltage regulator to reduce the 12V DC to 5V DC.

Figure 6.4: 5V DC Power Supply Designed On T.I. Webench

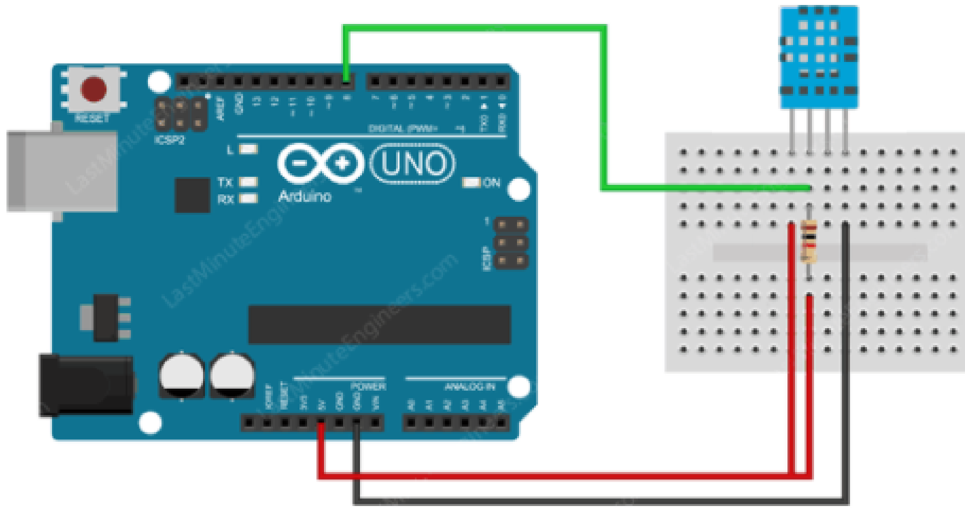


6.2 Sensor Unit Design

To complete the wiring for the sensor units, connections were needed to connect the DHT22, the HM 10, and the battery to the arduino nano. These are the only major components of the sensor unit design.

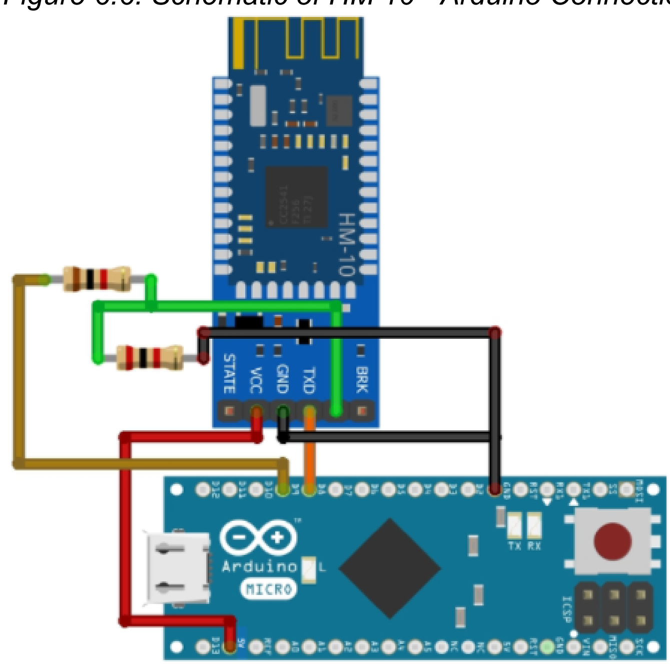
Firstly, we began connecting the DHT22/ The DHT22, which has four pins, only connects three of them to the arduino. Pin 1 is connected to Vcc. Pin 2 goes to D2, a digital data pin. Pin 2 also has a 10k ohm resistor connecting to the 3V3 on the arduino. This acts as a pull-up resistor for when the pins switch form output to input. Pin 3 is unused, and pin 4 is connected to ground.

Figure 6.5: Schematic of DHT22 - Arduino Nano Connections



The HM 10 Bluetooth Low Energy also has four pins, but unlike the DHT22 all four pins are connected. Pin 1 goes to the 5V output pin on the arduino. Pin 2 is grounded, and Pin 3 was connected to D8, since it's a digital input. Finally pin 4 went to D9, but had to be split by a 1k ohm and 2k ohm resistor, with the 1k ohm resistor connecting to D9 and the 2k ohm resistor going to ground, since the serial connection of the arduino expects 5V and the HM 10 expects 3.3 V. A schematic of the HM 10's connection to the Arduino is shown in the figure below.

Figure 6.6: Schematic of HM 10 - Arduino Connections



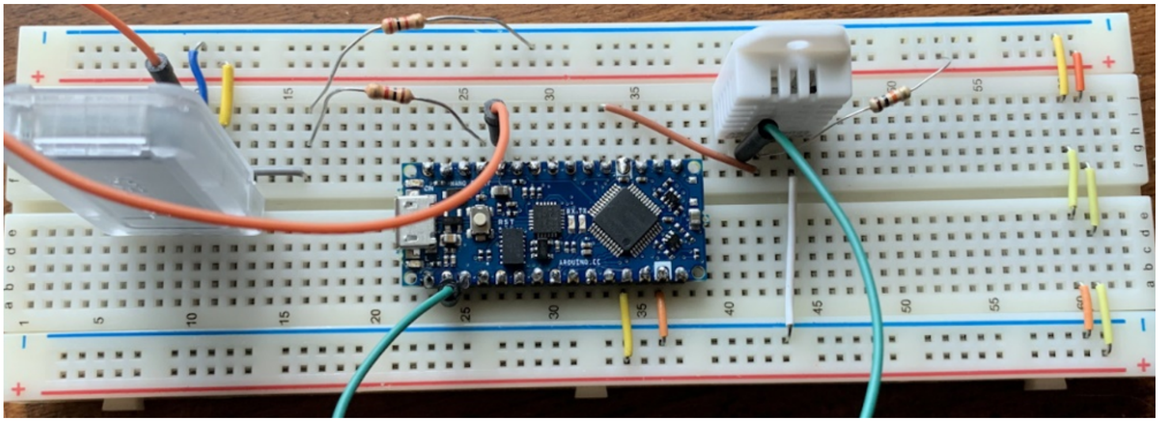
The arduino nano every microcontroller came with the header pins unattached. After they were soldered to the board, it was connected to the HM 10, DHT22, and the battery. Since multiple components used ground and Vin, the arduino is only connected at a few pins: Ground, 5V out, Vin, 3v3, D8, D9, and D2. This means that if we were to later amend our design to add more sensors (e.g. wind speed) we would likely have plenty of pins available. The image below shows a close up of the fully assembled sensor unit, without the battery installed. Since the circuit is entirely on the breadboard, no casing has been needed at this stage.

The breadboard connections for testing are as follows:

Table 6.3: Sensor Breadboard connections

Module	**Pin**	**Connection**	**Resistor**
DHT22	1	3v3 (arduino, 2nd from top left)	N/a
	2	D2 (arduino, 5th from bottom right)	10k Ohm resistor to 3v3
	3	Unused	N/a
	4	Ground (arduino, 2nd from bottom left)	N/a
HM-10	1	5V out (arduino, 4th from bottom left)	N/a
	2	Ground (arduino, 2nd from bottom left)	N/a
	3	D8 (arduino, 5th from bottom right)	N/a
	4	D9 via 1k Ohm resistor Ground via 2 k Ohm resistor	1k ohm to D9 2k Ohm to Ground
Battery	Red	Vin	N/a
	Black	Ground	N/a

Figure 6.7: Assembled Breadboard for a Sensor Unit



In summary,

- The sensor unit was built on the breadboard, using all of the components selected during part selection
- The DHT22 has a 10k pull up resistor
- The HM 10 uses a 1k and a 2k resistor to slipt voltage
- Our next step is to test on the breadboard before moving to a PCB design

7. Software Design Details

In this section we will discuss what we will be utilizing to complete the software aspect of this project. This will consist of our design methodology, as well as software we will be using to code the different components we proposed in the hardware design details.

7.1 Wi-Fi Module Connection

Once the connection of the Wi-Fi module to the Arduino has been set up, code needs to be uploaded to the Arduino to enable serial communication between the serial monitor and the ESP-01 through the Arduino. As a result, when a command is entered into the Arduino's Serial Monitor on the computer, the Arduino will relay it to the ESP-01. The following AT commands can be written in the Arduino's Serial Monitor to connect the ESP-01 to a Wi-Fi network:

Table 7.1: Basic AT Commands

Command	Function & Response
AT	This will check if the module is connected properly and its working, the module will reply with an OK acknowledgment.
AT+RST	This will reset the Wi-Fi module since it is good practice to reset it before or after being programmed, the module will reply with a system ready acknowledgment.
AT+GMR	This command is good practice to declare the firmware version installed on the ESP8266, however, it is an optional command.
AT+CWLAP	This will list the Access Points (available networks) and their available signal strengths, in which the module will reply with an OK acknowledgment.
AT+CWJAP =”SSID”, ”PASSWORD”	This will join an Access Point (available network) by connecting the ESP8266 to the specified Wi-Fi SSID and PASSWORD.
AT+CWJAP= “”,””	Contrary from the previous command, this will disconnect from any Access Points.
AT+CIFSR	This obtains and displays the ESP8266’s IP address.
AT+CWMODE=*?*	This sets the Wi-Fi mode. It should be set to Mode 1 if the module is going to be used in station mode (client), or 2 to be used as a Wi-Fi router (host), or 3 to be used as both
AT+CIPMUX=*?*	This enables the multiplex mode. It should be set to 1 for multiple connections and 0 for single connection.
AT+CIPSERVER =1,*port_number*	This configures the ESP8266 as a Wi-Fi server. Here ‘1’ is used to create the server and ‘0’ to delete the server.

Alternatively, it is possible to program the ESP8266 using Arduino IDE which is a lot easier. You no longer need to learn instruction documentation of the ESP module or have to program it using AT commands. This is done by pasting the link: *“http://arduino.esp8266.com/stable/package_esp8266com_index.json”* into the *Additional Board managers URL* and then installing the “esp8266 by ESP8266 Community” from *Board Managers* within the Arduino’s IDE. Note that after this, if you try to use the AT commands of the module it will not work. you will have to flash the ESP8266 module with the firmware so that we can get it back to working with AT commands again.

In summary,

- The first step is to connect the wi-fi module to the arduino based on the wiring diagram.
- For the next step we will establish a connection between the wi-fi module and serial communication port through the arduino.
  - The serial monitor enables communication between the computer and ESP8266.
  - An example code provided in the Arduino IDE to aid in establishing the connection.
- Next, use the serial monitor to ensure a connection is established with the ESP8266.
- Next setup the wi-fi module to enable wireless communications
  - This can be done using the AT commands that are listed in the table above.
- Finally, test the wireless connection for the ESP8266.
  - This can be accomplished with a phone or computer.

7.2 Bluetooth Module Setup

Bluetooth networks use a master/slave model to control when and where devices can send data. The master device must run an inquiry to try to discover a slave device. The sent inquiry request, and any slave listening for such a request, will respond with its address, and possibly its name and other information. Once a master device knows the address of a slave device, a connection between the two Bluetooth devices is formed through a process called *paging*. A single master device can be connected to several different slave devices, but any slave device in the network can only be connected to a single master.

After a slave has completed the paging process, it enters the connection state. While connected, a slave can be in regular active mode, it is actively transmitting or receiving data. Likewise, a slave can also be put into different low power modes while connected. These include a *power-saving mode*, where the slave is less active and only listens for transmissions at a set interval (e.g., every 100ms). A *temporary hold mode*, where the slave sleeps and returns to active mode when that defined period has passed (the master can command a slave device to hold). And lastly, a *park mode* where a master can command a slave to become inactive until the master tells it to wake back up.

Mainly, the master coordinates communication throughout the network, it can send and request data to any of its slaves. However, slaves cannot talk to other slaves in the network, they are only allowed to transmit to and receive from their master. Thus, Bluetooth can only have bi-directional communication. In Bluetooth

Point-to-Point protocols, the slave device is the one with data from sensor readings and the master device is the one that connects to the slave to gather that data.

When multiple modules are set to slave, they will be waiting for a connection, and the one module operating as master will initiate the connection with the other devices. We can essentially use a loop to instigate a bi-directional communication with each slave’s IP address. Assuming we know all the IP addresses for each of the slave modules, we could break communications with one slave by cutting off power, restore power and go into AT mode to configure the next slave, then cut off power, and then connect with the other slave, all in a trip round the loop. Alternatively, it is possible to put the slave modules into one of the low-power modes mentioned previously to allow the preservation of more power while they are waiting for a connection.

In summary,

- Bluetooth networks use a master/slave relationship for communication.
- The master must run an inquiry to establish a connection with the slave device.
    - The slave responds with its address and other information required for the connection.
- The connection is established through a process known as paging.
- A master can be connected to several slaves, but a slave can only be connected to one master.
- After the paging process is completed the slave can enter the connection mode.
    - When in connection mode the slave can actively send and receive data.
- The slave can be put into power saving mode where it only listens for requests for short time intervals.
    - Power saving mode is useful when running off a battery.
- The slave can also be put into park mode.
    - The slave sleeps until it receives a command from the master module to wake back up.
- Point to point protocol is used where the slave will send the data and the master receives the data.
- Communication can be established with multiple services by cutting power to the master device and sending AT commands to connect it with another device.

7.2.1 Master and Slave Module Configurations

There will be multiple Arduino Nanos around the warehouse that measure temperature and humidity using a DHT22 sensor. The data from the sensors will be sent through Bluetooth via an HM10 to another HM10 on an Arduino Mega, which is the main control unit that will show the data on a TFT display. The HM10 connected to the Arduino Mega will be set as master, and the other HM10s connected to Arduino Nanos will be set as slaves.

Configuring the Bluetooth modules will be done by entering some AT commands into the Arduino's Serial Monitor. Usually, setting up the slave device is done before the master. Thus, for each slave module, the command **AT+UART?** will return the baud rate of the module, the command **AT+ROLE=0** will set the module to slave, and lastly, every single HM10 has a unique address so the command **AT+ADDR?** will return the IP address of the module (which will be needed later).

After arranging all the slave modules, the master module can now be configured using the same steps. The master's baud rate should be the same as the slaves, and the command **AT+ROLE=1** will set the module to master. After that, **AT+CMODE=0** can be used to set the master's connection mode to a fixed address. This allows us to set the IP address of a specific slave module by doing **AT+BIND=*someAddress*** since our project is dealing with multiple modules set as slave.

The Bluetooth modules can also have user-friendly names given to them. These are usually presented to the user in place of the address to help identify which device it is. Using AT commands, setting a module name is done by the command **AT+NAME=*modulename***, where *modulename* will be the newly set name.

In summary,

- There will be multiple sensors spread throughout the warehouse to collect data from multiple points.
- The data collected will be sent via a bluetooth connection provided by the HM-10 modules.
- The HM-10 module connected to the main control unit will be set as master.
- The bluetooth modules connected to the sensors will be set as slaves.
- Setting the mode of the bluetooth module can be achieved by using AT commands.

    - These commands are typed into the serial monitor inside the Arduino IDE.
- The bluetooth modules can also be connected to each other using AT commands directed through the serial monitor
- Finally, user-friendly names can be given to the bluetooth modules to distinguish them from one another.

7.3 Unified Modeling Language

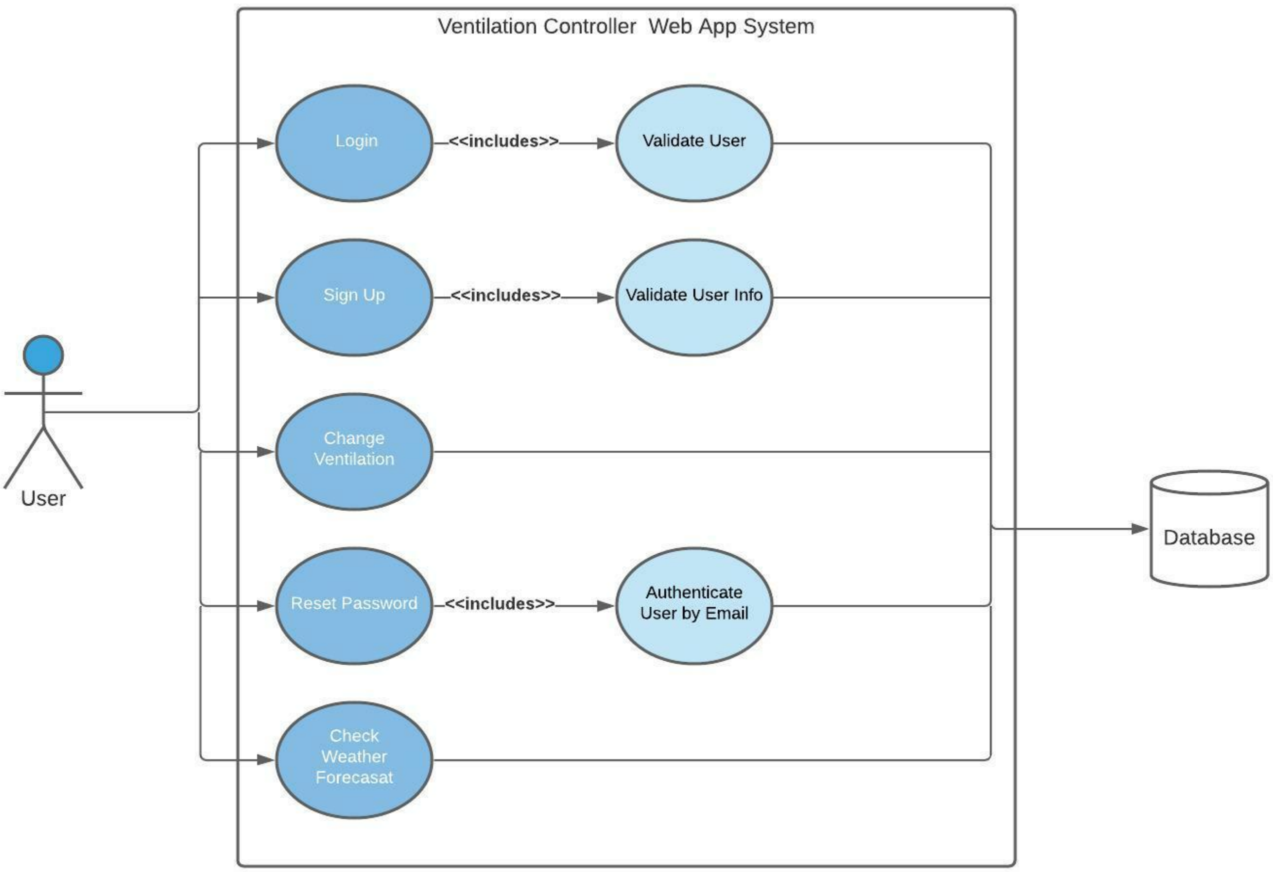
Unified Modeling Language, also referred to as UML, is utilized by software engineers to provide a way to visualize the design of the system. It is used to model object-oriented software engineering, but it can also be used to model the structure of an application as well as its behavior. The UML documents provide an excellent way of organizing thoughts during the planning stages of development as well as potentially preventing delays caused by miscommunication. There are many available UML documentation options, in this section we will describe two of them, Use Case diagram and the Entity Relationship Diagram. Both of them will be essential in the development of our web application.

7.3.1 Use Case Diagram

Use case diagrams are used to map out the different ways that a user, or also referred to as actors, will be interacting with the system. It is used to represent the different scenarios that the system will be interacting with users. And it will also map out the goals that the system should be able to do, at a high-level. Therefore it will not contain many specific events, but instead will be able to show the general flow.

As displayed in the diagram below, there are going to be 5 primary choices that the user will be able to make. First and foremost, they will be able to Login into the site, to be able to change any of the ventilation configurations. When they log in, the API will check that the credentials exist to give them access to the dashboard. We thought it would be wise to include a sign-up process, since there might be other users apart from our sponsor who would also need access to the website. Along with these two we can see that the user can change the ventilation settings, there will be three options, upward, downward or close. And there will be 4 vents that will have this same configuration. Lastly, the user will also have the option of checking the weather forecast for the city, which will be stored in the database too, because this will be communicated to the board, so that the user can also have access to it there.

Figure 7.1: Use Case Diagram of Web Application



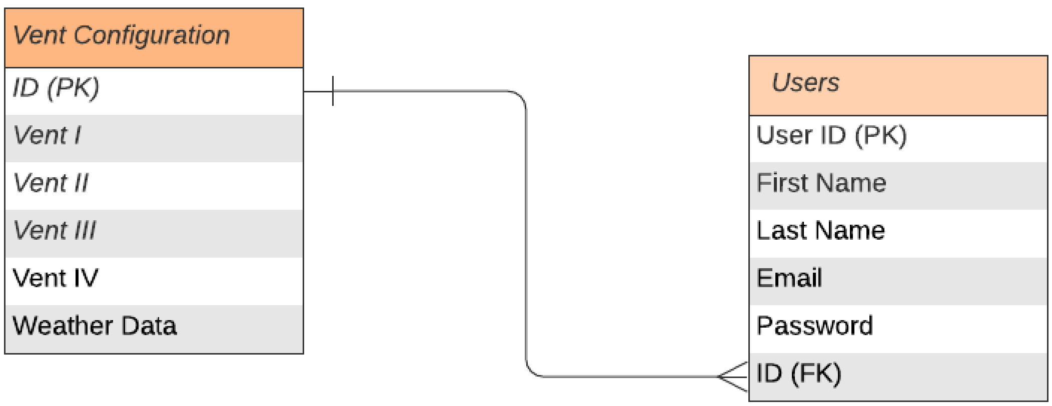
In summary,

- Use case diagrams are used to visualize the applications the software will have as well as show the end user or in this case sponsor what the functionalities will be for the software.
- Each available choice is broken down into a bubble, which contains a high level description on what the user will be able to do.
- For this project, there are approximately five different options that the user will be able to do, which will then be sent into the database to be able to make the appropriate changes.

7.3.2 Entity Relationship Diagram

An Entity Relationship Diagram, or most commonly referred to as an ERD, is used to demonstrate relationships that exist between the major entities in a database. An entity is usually a person, or role, but it can also be objects, like products. Each entity has its own attributes, like a person's first name and has what type of attribute it is, for a person's first name we would have a varchar type and specify the limit of characters that can be used. Along with attributes, the most important attribute in an entity will be the primary key, which uniquely defines each data in the database table. When we connect the entities, we utilize a foreign key which is a reference to a primary in another table, and this is how we connect each entity. However, in the diagrams, we specify the foreign keys and use special connectors to specify the sort of relationship that they have. The available relationships are one-to-one, one-to-many, and many-to-many. For the purposes of our database, we will not be using a many-to-many relationship, instead we will be doing a one-to-many relationship. Which happens when one record in a table has a relationship with many records in another database.

Figure 7.2: Entity Relationship Diagram



As we can see from this diagram, the modifications to the vents installed in the warehouse will have a one-to-many relationship with each user. Since there is only one system that can be modified and there can be many users that can have access to the system. The system will have the following attributes that will be in charge of deciding what changes will be made to the physical interface in the warehouse as well as to the vents. Since each vent has three available configurations, each will get its own attribute labeled after it, and the weather data collected from the web scraping will also get stored.

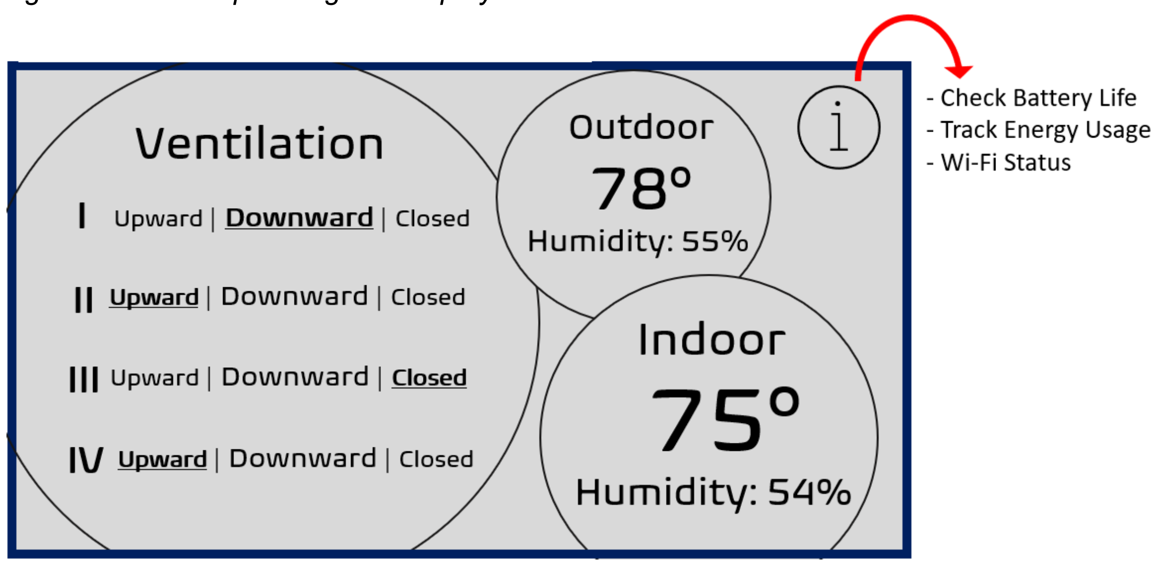
In summary,

- The entity relationship diagram is a common Unified Markup Language diagram used to visualize the relationships between MySQL tables.
- For this project, there will be two tables, with a one-to-many relationship. Where the first one will be the table to control the board, and to where the website data will be sent to, and the second table or entity, will be the users that will be able to access the website and make changes as they choose to the ventilations or check for the temperature, both indoor and outdoor.

7.4 TFT Touch Screen Design Concept

The conceptual design of the home screen shown in the below *Figure X*, is expected to have toggling buttons for each of the four vents. Even though the system is intended to adjust the vents automatically, these buttons will be implemented to give the user the ability to override the automation whenever they choose to. When the user toggles these buttons, they would alter the orientation of air flow to either be upward, downward, or closed. Additionally, there will be an information icon placed in the top right of the screen which will take the user to an information page when pressed. This page will display data to the user such as the battery life status for each sensor, the Bluetooth connection status, the Wi-Fi connection status, and the duration of how long the vents have been opened to track energy usage.

Figure 7.3: Concept Design of Display's Main Screen



The user interface for the controller's display is expected to only have two pages. One page being the display's main home screen, and the second page being the information screen accessed by the information button at the top right corner of the home screen. Regarding the code logic, by having an integer representing the current page, we can indicate what page/screen the user is on. For example, this can be done by checking whether the user is currently on the home screen (0) or the information screen (1). The appropriate screen for the user will then be displayed depending on which condition is met.

In summary,

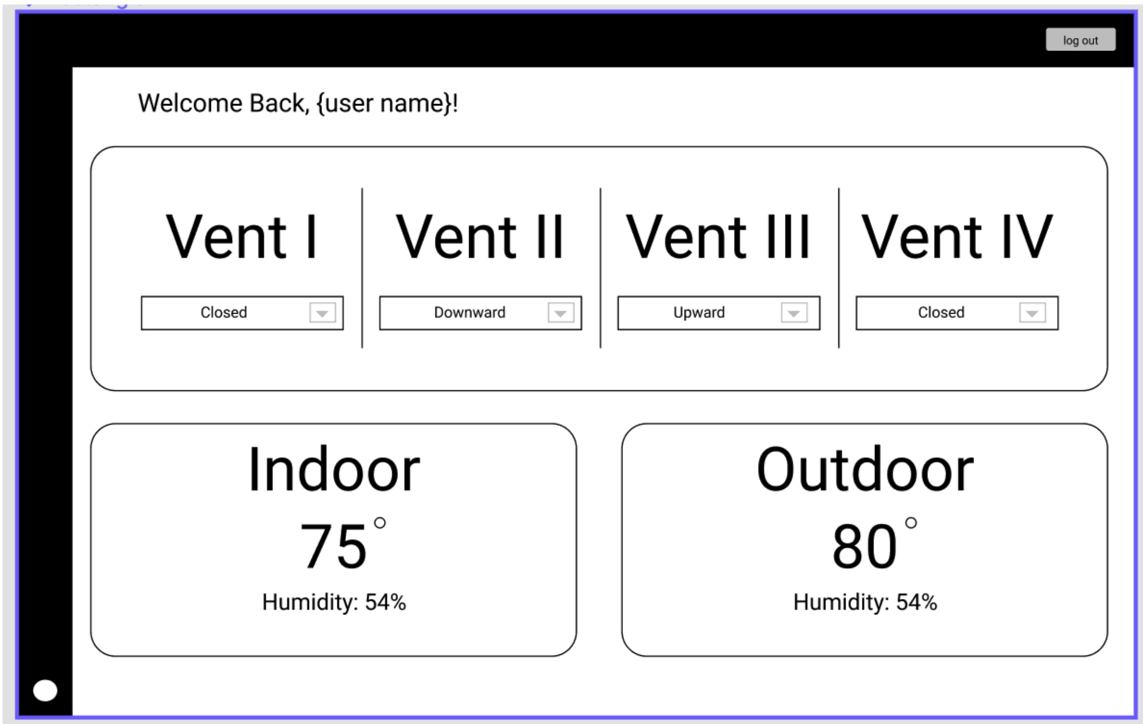
- The display will have toggle buttons that utilize the touch screen system.
- These toggle buttons will allow for manual control of the vent system.
    - Manual control will override the automation commands
- The display will also show the current indoor and outdoor temperatures.
- An informational page will be displayed when the “information” button is pushed.
    - This information page will show the battery life of the sensors, the wi-fi status, and the bluetooth status
- The display will have two screens that can be toggled between.
    - Homepage and informational page

7.5 Web Application Wireframes

In this section we will have the web application’s wireframe for the dashboard. Created using Figma, which mentioned in section 3.8, is a very good design software. This wireframe will facilitate the design process for the dashboard. Like this one, there will also be a simple wireframe for the login page too, and account page, so that we may be able to utilize most of our time creating and implementing the APIs, as well as potentially adding more features if there is enough time. To make the most of using Figma, we had to take a dive into UI/UX design, which would help prioritize the needed functionalities as well as optimize the layout and look of our website. The UI design, or user interface design, focuses on making the interface attractive. While the UX design, or user experience design, focuses on making the interface useful to the user.

Since the UX design focuses on enhancing the user satisfaction with the product or interface, by improving its usability, accessibility, and pleasure provided in each interaction between the user and the product. This is where our focus will be before moving to the UI design, so all of the early preliminary wireframe designs for the web application will be focused on utilizing UX for the most optimal design. To get started on our UX design, we had to take into account the existing principles. The main ones we focused on was making sure that the interface was usable, meaning that it would be able to fulfill the users needs. We also needed to make the interface self-explanatory, meaning that as the user makes their way through our application they can learn what each and every button does and won’t be confused or think the application is complicated. UX design also considers how to be able to successfully combine the UI design with any other visual design that will be added with no issues. There were other principles that emphasized keeping the interface as simple as possible. Along with these principles there were also others that we kept at the back of our minds like accessibility, and memorability.

Figure 7.4: Early Wireframe for Web Application



The dashboard above, will have drop-down menus for each vent available in the warehouse. Since there are only three existing configurations for each one, then each will have three options, Downward, Upward, and Closed. There will also be an estimated temperature for inside the warehouse. The outdoor temperature will be determined by the web scraping feature we will be implementing, the goal for this feature is to refresh and reload on its own to be able to get any changes to the temperature after certain time, however, in the worst case scenario, it would be refreshed via a click of a button.

For the other pages, the steps to creating the wireframes will be similar to the ones utilized in this dashboard, however they will be much simpler than the ones here as they will mostly be forms. There will however be an account page, where the user will be able to update their password and even change their profile picture. After these preliminary wireframes are completed we will shift our focus to the UI design. Similarly to UX design, UI design also has a couple of principles that are used in the industry, these of course vary depending on the source, but the main ones we encountered and considered were four principles. The first principle states that the user should be placed in control, in simple terms this refers to letting the user be able to gain a fast sense of mastery. This could mean including reversibility for actions, like if something is deleted on accident, providing an undo action, or a popup that will ask the user if they truly want to delete a record or data before actually deleting it. We have to have some sort of response to be able to show the user which action they are performing, so they aren't stuck in limbo waiting for a sign that an action is truly being completed. We also have to account for users of many different skill levels. The second principle stated that we had to make sure to give the user comfortability whenever they used the interface, this would be done by getting rid of any of the elements that would not be helping the user, items that are irrelevant or rarely used. The right terminology would also have to be used and be clear to any user interacting with the interface. We would also have to account for any errors that could occur, like forgetting the password and having an action to either send it to the user or reset it. Another golden principle stated that we should make sure to reduce cognitive load, meaning we would have to reduce things like the number of actions to complete a task, or add visual aids like tooltips to be able to support users in recognizing information. This also applied to using symbols or icons that are already recognizable to users from everyday exposure. The last main principle expressed making the user interface consistent, meaning that the style, things like colors, fonts and icons would be used throughout the interface. The behavior would also have to be consistent throughout it, this referred to having consistent buttons and menu items that do not change throughout the navigation of the interface. Along with these principles, we will also be considering good color schemes to apply that will comply with existing UI design principles.

In summary,

- The wireframes we will be creating will be done using Figma, and will facilitate the process of bringing this product to life.
- Some of the tools we will have to consider using to make the best user-friendly interface will be UI/UX design.
- UI design and UX design each have their own principles that are followed by industry professionals and these will be also followed by us.
- We will first focus on the UX design aspect to create the first wireframes, after completing them we will consider the UI design aspect to make sure we have everything we truly need and that it makes sense to the user.

7.6 Server Setup

By using a web hosting provider, we will be given a place on a web server to store all of our website's files, which will then be delivered as soon as a browser makes a request by typing in the domain name or our created website. With a free hosting service, we can utilize the opportunity to publish our own website and share them with the world completely free. We will consider using *000webhost* as our free hosting platform because it provides 300MB of disk space, 3GB of bandwidth, and the support of the latest PHP and MySQL versions. Additionally, we will be able to own dedicated resources, separate FTP details, and use any tool provided within the Web Host Manger's control panel.

If we outgrow the limits of free web hosting and need something more powerful, we can consider upgrading to premium web hosting options for more resources and extra features. These extra features include larger/unlimited bandwidth, frequent backups, extra disk space, additional MySQL databases, and more for $1.39 to $3.99 a month depending on the premium plan chosen.

After signing up for a free hosting account, we can begin working with this platform by uploading our existing website's code to FTP via the built-in *File Manager* where we can upload our application's PHP scripts. For easier development, however, we can use a CMS such as WordPress, or use the provided *Website Builder Tool* which has drag and drop features to streamline the entire process. These options allow for a "Mobile Ready" application which will be flexible for screens of all shapes and sizes. However, for the purpose of this project, we will be providing our own files that will have everything enclosed.

In summary,

- Our web hosting provider will allow us a place to store our files, as well as personalize our domain name, though it will have the webhostings name attached to it as well unless we pay for a domain name.
- The free hosting already comes with storage that should work and be enough for our web application, however if there were ever to be any issues that would require us to upgrade, our web hosting provider has some very affordable options.
- 000webhost has its own file manager that will facilitate storing each of our files.

7.7 Website Files Setup

The front end of our website will consist of a mixture of HTML and CSS, each page to our website will have its own HTML file, to keep everything organized. The CSS will be utilized to stylize the website, as well as make it responsive to different screen sizes, we will utilize libraries like bootstrap to make the responsiveness even smoother. There will be at least 4 HTML files utilized, these will be the login and sign up pages, as well as the dashboard where the user will be able to see the status of the vents as well as the temperature outside of the warehouse. There will also be an account page in case any of the users information needs to be configured. Following the HTML and CSS files, the functionality will be contained in JavaScript files. Each of which will be in charge of one functionality to make it more organized as well. These JavaScript files will get the information inputted by the user and format it into JSON packages. Then they will be sent by referencing the PHP files, which will be connecting the database with the frontend. It is the API of the system. These will also be a file per request. And they will be in charge of translating the request, and returning anything back to the user that might be required. In between here, there will be a space for the API to do the web scraping, and collect our desired data of the weather outside the warehouse.

7.8 Database Setup and Connection

It's important to have our information saved to a website's database in order to control our system from anywhere via internet connection. Thus, we can utilize the *Database Manager Tool* supplied by the web host to create a new SQL database by providing a database name, username, and password. The new database can then be managed using *PhpMyAdmin* which is one of the most popular MySQL administration tools for web hosting services. Through this service, we can either manually create a new table to store values from our website, or we can import an already existing SQL file which creates objects such as tables and indexes automatically. The PHP files for our project's website will need to specify the same information that was used when establishing the new database. This includes the created username, password, database name, and the server name. By doing this, our website will now have a connection to a database so that our application can store received values.

7.9 Arduino Code Setup

To tie it all together, the code that will be uploaded to our Arduino Mega must include a file that connects to the ESP 8266 Wi-Fi Module. It needs to specify a domain in order to fully make a connection with both the website and database. The Arduino's code must state the user's exact Wi-Fi name, Wi-Fi password, the name of the host website, the database identification, and the database password. Ultimately, we want the information from the sensors that are connected to the Arduino to be stored in the database, and then use the website's connection to the internet to read (and possibly alter) that information anywhere and anytime. The decision to create our project's web application using the building blocks of HTML, CSS, and PHP was reached due to the team's interest in learning a deeper understanding of web development and wanting to gain more appreciation of the basics.

7.10 Design Methodologies

When creating any project, it is helpful to have a good software design methodology in place to make sure that the tasks that need to get done are done on time and are functioning correctly. Which is why the software team will be taking advantage of the agile methodology, while developing the website and programming the PCB.

Before agile was developed, the way projects were completed was hectic to say the least. This traditional method consisted of a staircase approach, called the waterfall development process. While in theory it looked like a good way to develop software and general development things, in actual practice it proved to be inefficient. Going through this waterfall process was also very time consuming, even more so whenever any errors and issues were discovered. Sometimes it would even require starting from the beginning which would lead to a late delivery of the project or not even being delivered at all. And this is where agile really came in.

The term agile refers to the methods and best practices for organizing projects. It involves discovering requirements and developing solutions through collaborative efforts and involving the end user. The values and principles are documented in the Agile Manifesto. Agile really emphasizes four major principles in its project management, and these are: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. Agile uses a circular schedule, rather than a waterfall, where the testing leads right back to the beginning, requirements. And this process is repeated over and over until it is ready for deployment. This scheduling process allows the developers to discover any potential issues early as well as be able to release some features early. It also gives room for requirement changes, something the waterfall approach wouldn’t be able to handle without starting over from the beginning.

To be able to implement agile into our project, we will have to utilize the scrum framework. Under this framework, the process of implementing agile is based on 15 to 30 day sprints. For the website and PCB these sprints might have to be smaller, something ranging between one to two weeks. These sprints are fundamental development periods in which the team should be able to produce a part of the product that can be ready to deploy. Something like the functionality of a login page. There can also be daily scrums that are less than 30 minutes and serve as a meeting to check that progress is being made and to voice any problems that may have occurred. We might not be able to do a daily scrum based on the availability of the team members but we could probably have some sort of variation of this daily scrum.

In summary,

- Agile is a methodology widely used in software development.
- It is used to manage projects, time-saving and keeps the end-user involved.
- Agile uses scrum for implementing it into projects.
- Scrum helps teams work together and allows for good communication.

7.11 Version Control

For this project, we will have more than one person working on the software aspect. There are many situations we had to consider to make sure that the software production went smoothly. This is where version control came in, version control allows developers to track and manage changes in software code. It also provides a space for multiple people to be able to work on the same project together. It also provides a way to get a previous version back. Nowadays there are many options available that implement version control like Git, Apache Subversion, and Mercurial. However, the one we truly considered was Git.

Git, like the other version control softwares, can keep track of any and all changes made to the repository. A repository is stored in a cloud and contains all of the files of the software. With Git, the only way to have these files locally is to clone them. Any changes made to a local file will then have to go through a process of being added to be tracked and then committed. And even then Git has a feature called branches, that lets developers make changes to small parts of the software and still have a working copy, called the master. These branches can be created to test the software, make changes and test them out, as well as fixing any errors and testing out the solution. Another advantage of these branches is that it allows multiple people to work on different features at the same time without it leading to errors. And once the feature is at a point where it is tested and error free it can be merged into the main branch or master, to have it ready for production. After any changes are made to any of the files the changes have to be added to be tracked which gives developers another chance to review and check that the changes are what they want and contain no typos or errors.

Git also has its own version tracking system set up, that allows developers to go back to any of the versions that require a commit like merging a branch or creating one. Making any change to the master branch can also get tracked. And developers can go back to any of these if they choose. Or they can look at the changes that were made. Git does not only provide many resources to utilize for project version control but also has resources to be able to learn it and utilize it to its full potential. Git is open source software like the rest of the software we considered for this project. This tool is going to be extremely beneficial for our project, it will also provide us with experience in using it.

One of the most used tools for implementing Git is Github, which not only provides a student pack to college students, but is also free and for the purposes of this project also allows the usage of keeping projects private. Github is a repository and has a copy of your work that you can clone to any machine. It also makes it easy to add other contributors to the project, and keep anyone that isn't a contributor from making any changes. For private repositories however, only the people that are invited as contributors are the only ones that can make any changes. This is why we chose to use Github to implement version control on our software.

In summary,

- Version control is important when working in teams, it also provides a way to return to a previous working version if anything goes wrong.
- Git is one of the most widely used version control technologies and it is free and opened sourced
- Git also has branches as an advantage over other version control softwares.
- Github, which is a place to store the files and have access to previous versions or commits, it also allows users to collaborate on projects and even has a student pack with credits to technologies like web hosting and domain names.

7.12 Integrated Development Environment

When developing software, one of the most important tools before getting started is to have the right workspace to be able to comfortably and efficiently be able to code. This is even more important for large projects, and are solved by Integrated Development Environments, these are software’s that provide an environment with tools, like a text editor with debuggers and sometimes even has automation. Our Automated Ventilation Controller will be utilizing an Arduino board that will also be accessing a database to be able to communicate with the web application and get information about the weather outside of the warehouse. For this reason, our team will be using at least two IDE’s, one of which will be the Arduino IDE 1.8.15 and VSCode. Both of which are open sourced.

The Arduino products have their own built-in support for programming it, it has both online and offline support available, with their own language that is similar to C/C++. This support is achieved by simply connecting the board to the appropriate PC, which can support Windows, MacOS, and Linux operating systems. And since one of our teammates is working on MacOS, this is extremely helpful. Connecting the board to the PC will not only allow us to program it but it will also serve as its power source. For this IDE, the files are referred to as sketches. And it comes ready with example files that will give you not only a look to the format but will also allow you to check that certain parts of the board work, like it comes with an LED blink example. Since this IDE is used for all of the boards manufactured by Arduino, whenever creating a sketch, we will have to select the board type and port. After this is selected and the code is complete, we can upload the program to the board. Which is the only way to be able to upload programs to the Arduino boards. One of the advantages of this software is that, as previously mentioned, it is open-sourced, which means that there will be many free and available resources whenever we run into a problem.

The other software we will be using to complete this project will be Visual Studio Code or VSCode, which is a widely used code editor. It comes with built-in Git, as well as has settings for running and debugging code. Not only that but it also comes with a tool called IntelliSense that will provide useful autocompletion of function definition types, variable types and imported modules. And one of its biggest advantages is its extensibility and customizability. There are countless extensions available to facilitate the development of this project, especially when it comes to the web application. There are extensions that can auto rename HTML/XML tags when any changes are being made to it, even some to keep comments in code even more organized. In cases where there might be a spelling error when naming a variable or making a comment, there are extensions that will check for spelling errors too. There is also an extension called CSS Peek, which gives a look in html files to the CSS references for IDs and class strings, without having to look into or open the CSS file. Apart from the ones mentioned there are also many more that will be utilized to successfully complete this project. Its customizability will also come in handy to make an even better user experience and will be useful to add colors that can be looked at for multiple hours.

In summary,

- IDEs are important tools when developing software, they can facilitate the process of programming as well as testing the code.
- The software for the hardware of this project will utilize the Arduino IDE which will use C++ as its primary language.

- For the website, VScode will be used to create each file of the LAMP stack, since it has extensions to support each of those languages.

7.13 UART

The UART protocol, which stands for Universal Asynchronous Receiver/Transmitter, is widely used for device-to-device communication. The signals used to communicate are Transmitter (Tx) and Receiver (Rx), which are used to transmit and receive serial data. UART transmits this data in a form of a packet that consists of a start bit, data frame, a parity bit and stop bits. This technology will be utilized for both the Bluetooth and Wi-Fi modules on this project. For our project, serial communication is used between the Arduino board, computer, and other devices. All Arduino boards have at least one hardware serial port (also known as a UART), and some such as our Arduino Mega, have several.

Figure 7.5: Serial Pin Comparison of Arduino Boards

BOARD	USB CDC NAME	SERIAL PINS	SERIAL1 PINS	SERIAL2 PINS	SERIAL3 PINS
Uno, Nano, Mini		0(RX), 1(TX)			
Mega		0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)

On Uno, Nano, Mini, and Mega, pins 0 and 1 are used for communication with the computer. Connecting anything to these pins can interfere with that communication, including causing failed uploads to the board. The *ATmega2560* on the Arduino Mega provides four hardware UARTs in order to receive (RX) and transmit (TX) TTL (5V) serial data communication.
These serial pins are listed as the following:
Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Note that pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

Additionally, the *ATmega8U2* on the Arduino Mega board channels one of the four UARTs over USB and provides a virtual COM port to software on the computer. (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically). The Arduino's software environment includes a built-in *Serial Monitor* which allows simple textual data to be sent to and from the board in order to communicate with an Arduino board. This is done by clicking the serial monitor button in the toolbar and selecting the same baud rate used in the call to begin(). The RX and TX LEDs on the board will flash when data is being transmitted through the ATmega8U2 chip via USB connection to the computer (but not for serial communication on pins 0 and 1).

8. Hardware Testing

This section discusses the methods of testing the hardware parts and validating their selected purpose as discussed in the research section. This section will cover many different tests for the different components. These tests will ensure first that the hardware component is working within its design parameters, and to ensure that it will work as intended with the other selected parts. The tests that require software to complete will be done with a pre manufactured arduino board. This is done to ensure that parts are arduino friendly, and to establish a baseline of proper operation. This baseline will allow us to troubleshoot and verify proper operation for the final PCB design of the main control unit. The use of the pre-manufactured arduino board will also allow code to be written and tested well before the final printed circuit board is designed to aid in streamlining all areas of the project development.

The prototype chosen for this project is the Arduino Mega2560. Building a prototype from this pre-manufactured board is essential for the testing phase of this project. Prototyping based on this design will enable us to verify that specific parts will work with the processor that is chosen for this project. It will also allow us to select the pins that the specific components will use. This will enable us to design the printed circuit board throughout the testing phase of the project. Prototyping with a pre-manufactured board also has some other advantages. First, it has its own independent power supply. This power supply will allow us to run tests on the prototype board without the need to build a dedicated power supply, since most of the power supply parts are in short supply.

Similarly, the Mega2560 board has a dedicated USB connection. This USB connection allows for quick uploading of sketches from the arduino IDE. This will enable the project testing to run smoother since programming can be done quickly, and small changes to the code can be accommodated faster. Lastly, The Arduino Mega2560 will allow us to run a full prototype test with all components connected to it. This will ensure that the goals of this project can be met prior to completing the final PCB design.

8.1 Relay Testing

The first component to test for this project will be the relays. The test for these relays can be simplified since the relays do not need a program and printed circuit board to ensure their proper operation. The initial test of the relays will ensure that all relays on a module are working properly. To accomplish this test the 5V VCC line will be connected to a 5V DC voltage source. The ground of the relay module will be connected to the ground of the voltage source. Next a multimeter will be used to test the resistance across the relays. The first resistance to measure will be between the normally closed contact and the neutral.

The resistance across these should be very low since it is normally closed. The next resistance to measure will be between the normally open contact and the neutral contact. The resistance across these terminals should be extremely high since the contacts should be open. Next, to ensure proper operation of the relay module the control pins will be driven low to ground. Once the pins are connected to ground the associated relay should open and the associated LED should turn on.

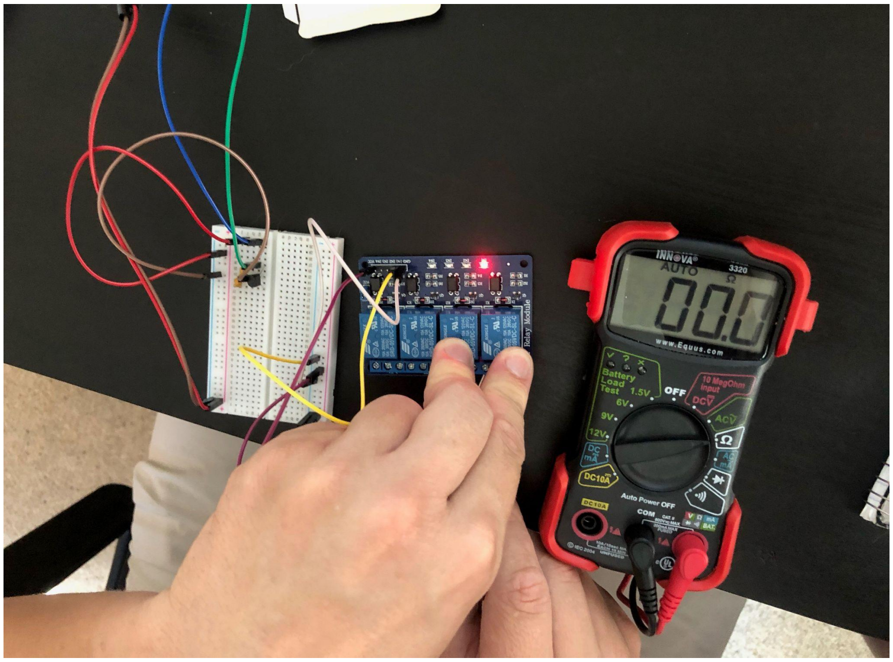
To accomplish this, a wire will connect the control pin of the relay under test to ground. A multimeter will again be used to measure the resistance between the contacts. The resistance between the normally closed contact and ground should be high since these contacts should be open since control power was applied. Next the resistance between the normally open contacts and neutral will be measured. This resistance should be very low since the contacts should be controlled due to control power being applied.

For this test the most important measurements will be between the normally open contacts and neutral since these are the contacts that will be used to control the vent motors. All resistance measurements of the normally open contacts will be recorded in the table below. Resistance values do not need to be measured to the decimal place since this test is just to verify continuity of the relay.

Table 8.1 Measuring the resistance of the relays when open and closed

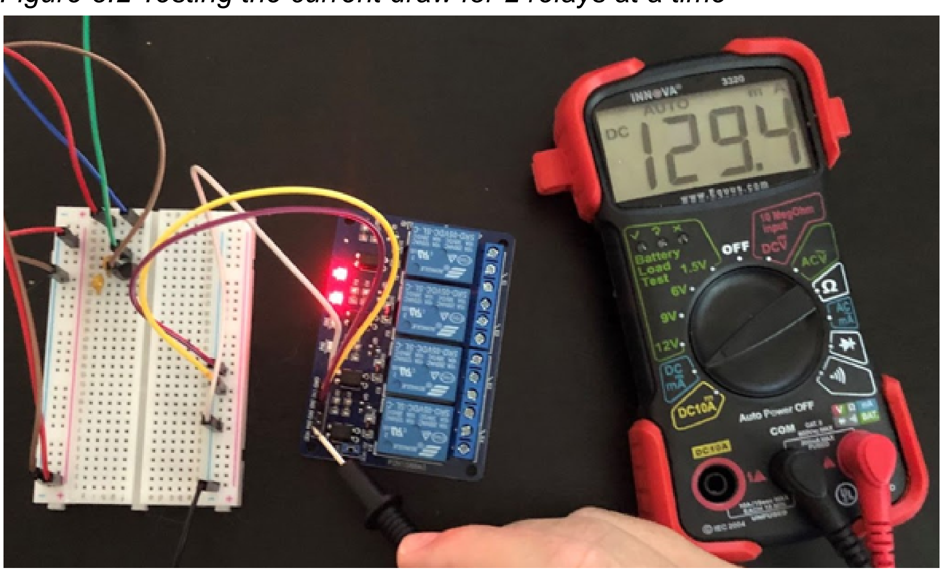
	Relay Number	**NO contact without control power (Ω)**	**NO contact with control power (MΩ)**
Relay Module 1	1	<1 Ω	>1MΩ
	2	<1 Ω	>1MΩ
	3	<1 Ω	>1MΩ
	4	<1 Ω	>1MΩ
Relay Module 2	1	<1 Ω	>1MΩ
	2	<1 Ω	>1MΩ
	3	<1 Ω	>1MΩ
	4	<1 Ω	>1MΩ
Relay Module 3	1	<1 Ω	>1MΩ
	2	<1 Ω	>1MΩ
	3	<1 Ω	>1MΩ
	4	<1 Ω	>1MΩ
Relay Module 4	1	<1 Ω	>1MΩ
	2	<1 Ω	>1MΩ
	3	<1 Ω	>1MΩ
	4	<1 Ω	>1MΩ

Figure 8.1: Testing the relay for circuit continuity



After the relay modules have been found to all work properly, next is a test on how much current each relay draws. This is an important test because it will determine how much current the main control unit should be supplied with. The relays will be the largest current drains for this project. The relays will act in pairs so it is important to note how much current is drawn for two relays at a time. This can be done by measuring each relay's current draw and adding the two relays together that will work in a pair.

Figure 8.2 Testing the current draw for 2 relays at a time



This testing needs to be more precise than the continuity test to ensure the proper values are found. The current draw for each relay will be recorded in the table below.

Table 8.2 Measuring the current draw for each relay

	Relay Number	Current Draw (mA)
Relay Module 1	1	62.1
	2	61.9
	3	62.5
	4	62.4
Relay Module 2	1	62.3
	2	62.0
	3	61.8
	4	60.8
Relay Module 3	1	63.3
	2	64.2
	3	63.6
	4	62.8
Relay Module 4	1	63.8
	2	63.3
	3	63.3
	4	63.6

Further relay testing will be conducted with the use of the Arduino Mega to ensure the relay modules performance when controlled by software. To complete this testing, the relay modules will be connected to the prototype board. Next, a code will be uploaded that will drive a control pin low for a period of ten seconds.

Next, the test code will drive that pin high and move to the next control pin where it will again drive that pin low for a period of ten seconds. This will be repeated for each relay on the relay module. When the pin is driven low a loud click sound should be heard from the board. A continuity test can also be performed to ensure that the switch closes and opens properly for each relay. Next, the DHT22 temperature sensor will be connected to the prototype board. A code will be uploaded to the arduino to shut a relay switch when a temperature above 70 degrees is sensed and open the relay switch when a temperature below 70 degrees is sensed. Once the code has been uploaded to the prototype the test can begin. Initially, the relay should shut since the ambient room temperature should be greater than 70 degrees. Next, using a frozen cup, place the frozen cup over the sensor. Within a few seconds the DHT22 should sense the temperature drop and open the relay switch. If a frozen cup is not available then a bag of ice placed close to the sensor should also work. While performing this test ensure that condensation does not drop onto the like electronic components. A continuity test should be performed when the switch is open and closed to ensure proper operation. This test will be the first test to integrate two of the chosen components for this project. This test is also very similar to the operational principle of our project, and as such will be the basis for our proof of concept testing that will be performed once all component testing is completed.

Figure 8.3: Relay Terminal Diagram

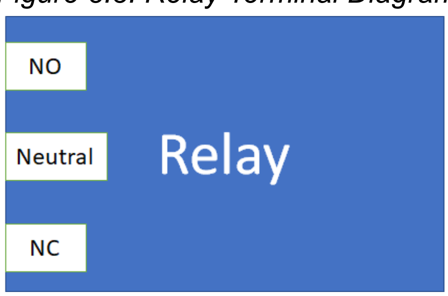
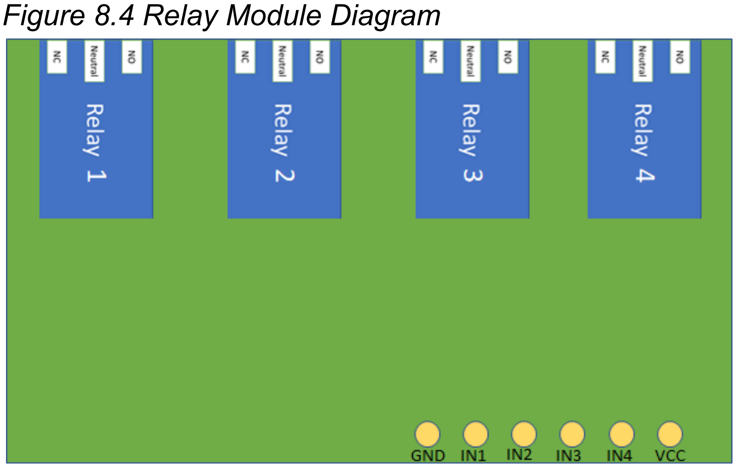


Figure 8.4 Relay Module Diagram



8.1.1 Continuity Testing

Below is the test procedure for the continuity testing. The purpose of this test is to verify that the relays are opening and closing as they should when their control line is driven low to ground. The testing for this procedure will be done with a handheld multimeter connected to the relay that is being tested. Precise ohm values are not required since the test is only to verify proper operation.

1. Connect the VCC terminal on the relay module to the positive wire of a 5V DC source.
2. Connect the GND terminal of the relay module to the negative wire of the same 5V DC source.
3. Connect the terminal IN1 to the positive side of the 5V DC source to simulate driving the pin to HIGH. No light or clicking sound should be heard.
4. Place the multimeter in "Ω" mode
5. Connect the probes of the multimeter to the NC and Neutral terminal of the first relay. Record the resistance value between these terminals. The resistance should be very high since these are open.
6. Repeat Step 3 and 4 for the other relays on this module. Ensure to move the "simulated HIGH pin" for each relay.
7. Connect the terminal IN1 to the negative side of the 5V DC source to simulate driving the pin LOW. The corresponding relay light should turn on and a clicking sound should be heard from the relay.
8. Connect the probes of the multimeter to the NC and Neutral terminals of the first relay. Record the resistance value between these terminals. The resistance should be very low since these terminals are now closed.
9. Repeat steps 6 and 7 for the other relays on this module. Ensure to move the "simulated LOW pin" for each relay.

8.1.2 Current Load Testing

Below is a step by step procedure for measuring the current each relay will require. The purpose of this test is to measure the current draw for each relay. This is an important test because the relay modules will be the largest load on the power supply. The values that are measured will be used to ensure the power supply that is designed for this project will be able to supply the required current. This test will use a handheld multimeter connected between a 5V DC source and the VCC pin of the relay module. Precise measurements are required here to ensure accurate current draw calculations. In this project, relay modules will operate in pairs. In order to get an accurate estimate, the current draw of the paired relays should be added together.

1. Connect the negative terminal of the 5V DC power supply to the GND pin on the relay module
2. Connect the VCC pin to the negative terminal of the multimeter
3. Place the multimeter in “DC mA” mode.
4. Connect the positive terminal of the multimeter to the positive terminal of the 5V power supply.
5. Connect pin IN1 to the negative terminal of the 5V power supply to simulate driving the pin to LOW.
6. Read and record the current displayed on the multimeter.
7. Move the “Simulated Low Pin” to the other control pins located on the relay module. Repeat steps 5 and 6 for each relay.

8.1.3 Operational Testing

Below is a step by step procedure for the operational test of the relay modules. This test will be used to ensure that the relay modules will respond to inputs from the microcontroller. This will be the final test for the relay modules apart from the final prototype testing. While performing this testing ensure that only one relay switch is operated at a time. This is necessary due to the large amount of current that the relay modules draw. Failure to do this could result in a faulty test due to overloading the prototype microcontroller.

1. Connect the pins of the relay module to the microcontroller in accordance with the wiring table provided in section 6.1.3.
2. Connect the Arduino Mega2560 to the computer using the USB connection.
3. Open the arduino IDE and upload the first test code that cycles through the relay switched by periodically opening and closing the switches.
4. When a switch is open perform a continuity test in accordance with section 8.1.1 to ensure proper operation of the relay switch, repeat for other switches.
5. Once this test has been properly completed unplug the arduino from the USB connection.
6. Connect the DHT22 Sensor to the Arduino Mega2560 in accordance with Table 8.5.
7. Reconnect the Arduino Mega2560 to the computer using the USB connection.
8. Upload the second test code to the Arduino Mega 2560. Once it is uploaded the selected relay switch should shut.

9. Perform a continuity test on the selected relay switch to verify proper operation.
10. Using a frozen cup, place the cup over the DHT22 sensor. The sensor should sense the colder temperature and open the relay switch.
11. Verify proper operation of the relay using a continuity test.

8.2 Power Supply Testing

The power supply testing will be one of the most difficult tests to perform and is also one of the most important. The power supply needs to be able to provide a constant 5V DC to the main control unit, and be able to supply a current that can power all the necessary components. Some of these components draw a significant amount of power especially when run simultaneously. It is important that the voltage during these periods of high current does not drop excessively or the main control unit could shut off in mid operation. It is also important that when a component that draws a lot of current turns off there is not a large voltage transient that occurs which could burn out the main control unit.

To test the power supply from *figure 16,* a 12V DC we will need to connect it to a source of power that is around 12V DC. To accomplish this the power supply will be connected to a deep cycle battery to simulate the 12V DC provided at the warehouse. The deep cycle battery was chosen due to its voltage level and ability to supply a large amount of current. After connecting the power supply to a voltage source, we will connect various loads to it to measure the current and voltage drop. The loads will be resistors of various values to simulate various loads placed on the power supply. The first test will be a 100Ω resistor to draw 50 mA of current. Next five 100Ω resistors will be placed in parallel to simulate a larger load that would draw 250mA.

The final test will consist of placing ten 100Ω resistors place in parallel to simulate a load that will draw .5 amps. The resistors are placed in parallel to limit the current going through any single resistor. Drawing too much current through a resistor will result in overheating and cause premature failure. The resistors that will be used are rated for use up to one quarter watts. These load tests will drive these resistors to their limits, so caution should be taken during these tests. During the power supply testing the voltage will be monitored to ensure the voltage stays constant with different loading. Current will also be tested to ensure that the power supply is providing adequate current for testing. All values will be recorded in the table below.

Table 8.3: Power Supply Testing

Loads	**Voltage (V)**	**Expected Voltage (V)**	**Current (mA)**	**Expected Current (mA)**
100 Ω		5 V		50 mA
20 Ω		5 V		250 mA
10 Ω		5 V		500 mA

The data collected on the above table will be used to verify the ability of the power supply to fully power the main control unit and all of the accessories connected to it. The data collected from testing the power supply can be used in conjunction with the data collected from the relay module testing. These tables should be compared since the relay modules will be the biggest source of power consumption for the main control unit. If the power supply can deliver a .5 amps of current then it should be more than sufficient to power the system as a whole. An oversized buck converter was chosen for this job. From the data sheet of the buck converter it can handle up to 2 amps. This is important because our project should not draw much more than 25% of the rated current for this design. This leaves a large margin of error. Which will lead to a longer lasting power supply, and prevent premature failure.

An alternate design for the power supply has also been made due to the supply constraints of the buck converters. Currently, all buck converters are sold out from many online stores. The alternate design uses a linear voltage regulator that is rated for two amps. The two amp margin for a linear voltage regulator is very important due to the nature of these regulators heating up which can lead to premature failure. If needed a heat sink can be added to the linear voltage regulator to aid in heat dissipation. The location of the power supply will be inside the main control unit casing which will not be ventilated. The dissipation of heat will be important. A larger heat sink may be necessary due to limited cooling available inside the main control unit casing. Also linear voltage regulators are sensitive to reverse polarization. To prevent failure of the voltage regulator due to this, a diode will be used to bypass the voltage regulator. The cathode of the diode will be connected to the input side of the voltage regulator. The anode will be connected to the output of the voltage regulator. If the power supply output receives a higher voltage than the input, the diode will direct current around the voltage regulator, saving it from the damaging reverse current

Figure 8.5: Current Simulation for 100 Ω load

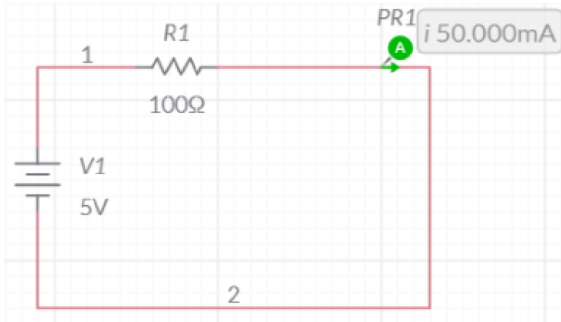


Figure 8.6: Current Simulation for 20.5 Ω load

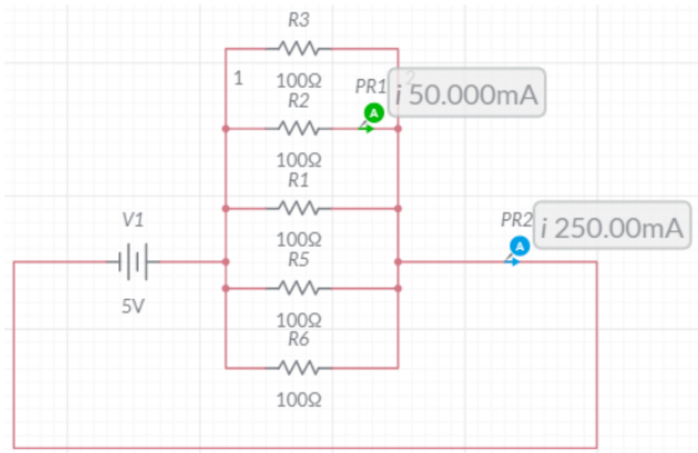
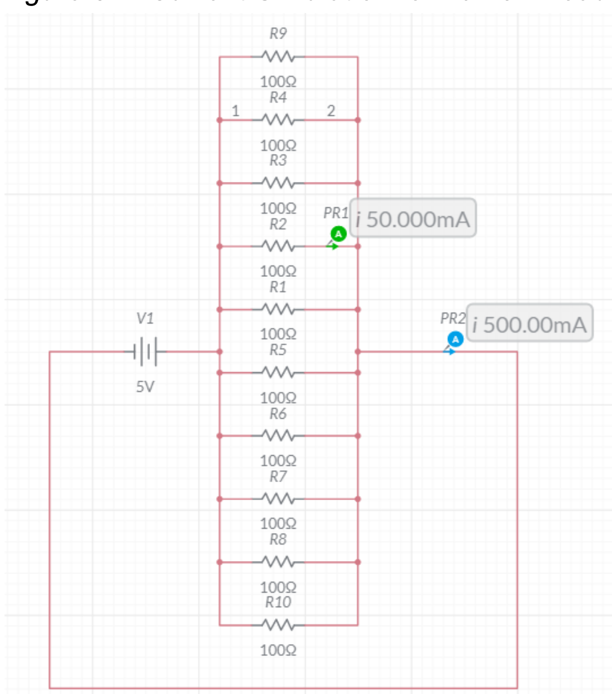


Figure 8.7: Current Simulation for 10.25 Ω load



8.2.1 Voltage Testing

Below is the procedure used to test the voltage output of the power supply. This test is important to ensure that the power supply will maintain a steady 5V DC output voltage for the different loads that will be placed on it. The loads are designed to simulate the different operational loading requirements of the main control unit. This test can be performed with either a handheld or bench top multimeter. The voltage levels should not fluctuate more than 5% (5V ± .25V).

1. Build the Power Supply based on the design proposed in figure 16.
2. Replace Iout with a 100 Ω resistor
3. Connect the positive terminal of the 12V battery to the positive side of Vin.
4. Connect the negative terminal to the ground side of Vin
5. Connect other grounds to the negative terminal of Vin
6. Place the multimeter in “DCV” mode
7. Read the voltage across the 100 Ω resistor.
8. Unplug the positive terminal of Vin from the 12V Battery
9. Repeat steps 2 through 8 for the simulated 20 Ω load and 10 Ω load

8.2.2 Current Testing

Below is the procedure used to test the current output of the power supply. This test is important to ensure the output current of the power supply will be suitable for use in this project. The original design of the power supply uses a buck converter. A back up design may also be used that is centered around a linear voltage regulator. If the linear voltage regulator is used, periodically check the temperature. If the temperature increases drastically, stop the test and install a heatsink on the linear voltage regulator before resuming.

1. Build the Power Supply based on the design proposed in figure 16.
2. Connect the positive terminal of the multimeter to the positive terminal of Iout
3. Connect the negative terminal of the multimeter to the 100 Ω resistor
4. Connect the other side of the resistor to ground. At this stage, the multimeter should be in series with resistor and replacing Iout.
5. Connect the positive terminal of the 12V battery to the positive side of Vin.
6. Connect the negative terminal to the ground side of Vin
7. Connect other grounds to the negative terminal of Vin
8. Place the multimeter in “DC mA” mode. Read and record the output current of the power supply
9. Repeat steps 2 through 8 for the simulated 20 Ω load and 10 Ω load

8.3 Wi-fi Module Testing

To test the connection of the Wi-Fi module, we will configure the ESP8266 using AT commands. The AT commands are sent from the Arduino IDE, not the microcontroller, thus the Arduino microcontroller used for testing should be in reset mode.

Below are the connections to place the microcontroller in reset mode:

1. The ESP8266 RX pin to the Arduino RX0 pin.
    - *Voltage divider for RX. with 1K and 2.2K resistor*
2. The ESP8266 TX pin to the Arduino TX0 pin.
3. The ESP8266 ground pin to the Arduino ground pin.
4. The ESP8266 VCC pin to the Arduino 3.3v pin.
    - *Note: Since the 3.3v power supply of the Arduino cannot provide sufficient current to the* ESP8266*, it might be necessary to connect the VCC pin of the* ESP8266 *to an external 3.3v power supply.*
5. The Arduino reset pin to the Arduino ground pin.

Table 8.4: Wi-fi Module Wiring

Wi-fi Module Pins	**MCU Pins**
VCC	5V
GND	GND
TX	RX1
RX	TX1

Now connect Arduino to PC through USB cable. Open Arduino IDE software > File > New (blank sketch). Before uploading the blank sketch to the Arduino, unplug the TX, RX, and reset wires from Arduino. Once uploaded, reconnect the wires. Open the Serial Monitor within the Arduino IDE to begin using AT commands to communicate with the Wi-Fi module.

Below are the AT commands to configure the module:

1. "**AT**" - Enter into AT command mode; receive OK response
2. "**AT+RST**" - resetting the module
3. "**AT+GMR**" - lists version of module
4. "**AT+CIFSR**"- shows the IP address of module
5. "**AT+CWMODE=2**" - sets the module's mode; OK response

    - *1 is client, can request URL from server*
    - *2 is a host, in which other devices can connect to*
    - *3 is both*
6. "**AT+CWSAP?**" - displays module's default name and password
7. "**AT+CWSAP="SOME_NAME","SOME_PASSWORD"**" – will set a custom ssid and password to the module
    - *Note that password must be 8 or more characters long*

Now test the Wi-Fi module by using a phone or another compatible device to search for the name of the module and connect to it using the set password.

8.4 DHT22 Sensor Testing

The DHT22 sensors have four pins, VCC, GND, data pin and a not connected pin which has no usage. To test the communication between the sensor and the Arduino Board, a pull-up resistor from 5K-10K Ohms is required to keep the data line high. We will use the DHT library which takes care of the precise timing and the timing diagrams for getting the data from the sensors.

Figure 8.8: Wiring Diagram for DHT22 to Arduino Mega

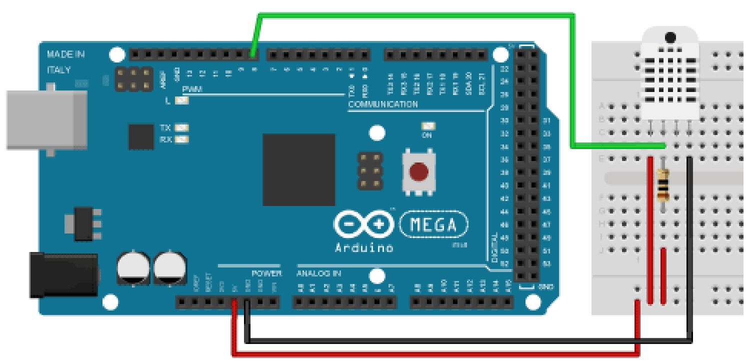
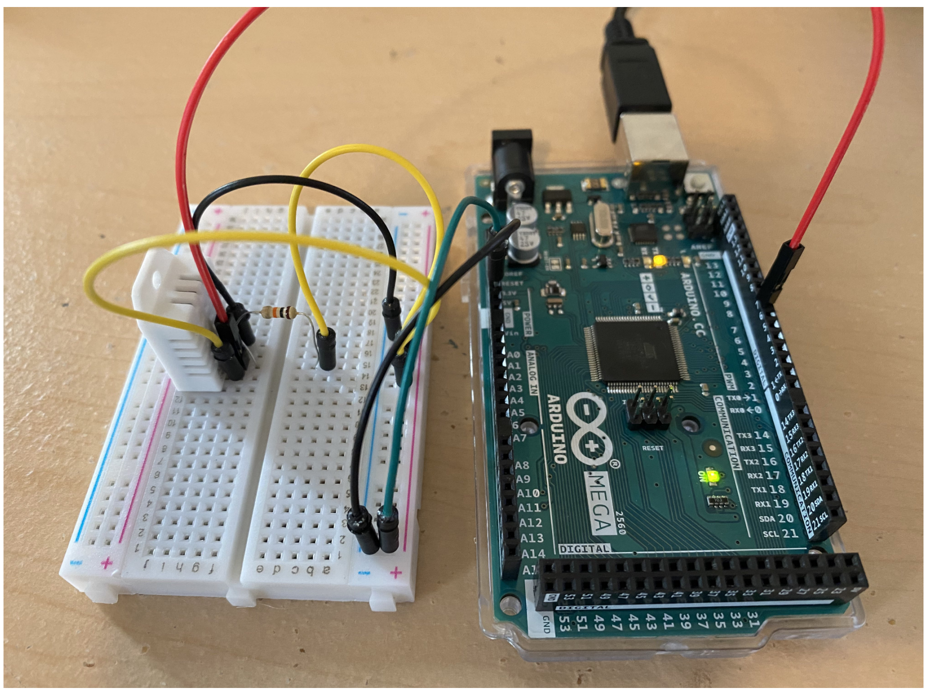


Table 8.5: Connections for DHT22 to Mega

DHT22	**Arduino Mega**
Pin 1 (VCC)	5v
Pin 2 (Data)	Digital pin 7 and 5v via 10k resistor
Pin 3 (NC)	Not Connected
Pin 4 (GND)	Ground

Once the sensor is connected to the Arduino Mega 2560 the test code can be uploaded to the board. Some adjustments to the driver will be necessary to establish proper communication between the prototype board and the sensor. Once the driver has been adjusted, the test code can be uploaded to the Mega2560. Once uploaded, the sensor temperature and humidity reading can be read through the serial port interface that is provided in the Arduino IDE. To ensure the sensor is taking proper measurements, the displayed temperature and humidity data can be compared to a known standard. If the temperature readings are correct no further correction is necessary. If the readings are incorrect, a calibration algorithm may be necessary to bring the reading into an acceptable range. For our purposes, the temperature should be within ±1 degree fahrenheit of the standard. Humidity should be within ± 2% of the standard.

Figure 8.9: DHT22 Connected to Arduino Mega



Regarding the source code that will be uploaded to test the DHT22, we first include the mentioned DHT library which can be found on Arduino's official website, define the pin number to which our sensor is connected (in this case, it is digital pin 7), and create a DHT object. Within the setup section of the code, we need to initiate the serial communication in order to use the Arduino's IDE Serial Monitor to read the results. Within the continuously looping section of the code, we create floating point variables to hold the temperature and humidity values separately:

```
float temp = DHT.temperature;
float hum = DHT.humidity;
```

Once the data is gathered, we can print this test data on the Serial Monitor:

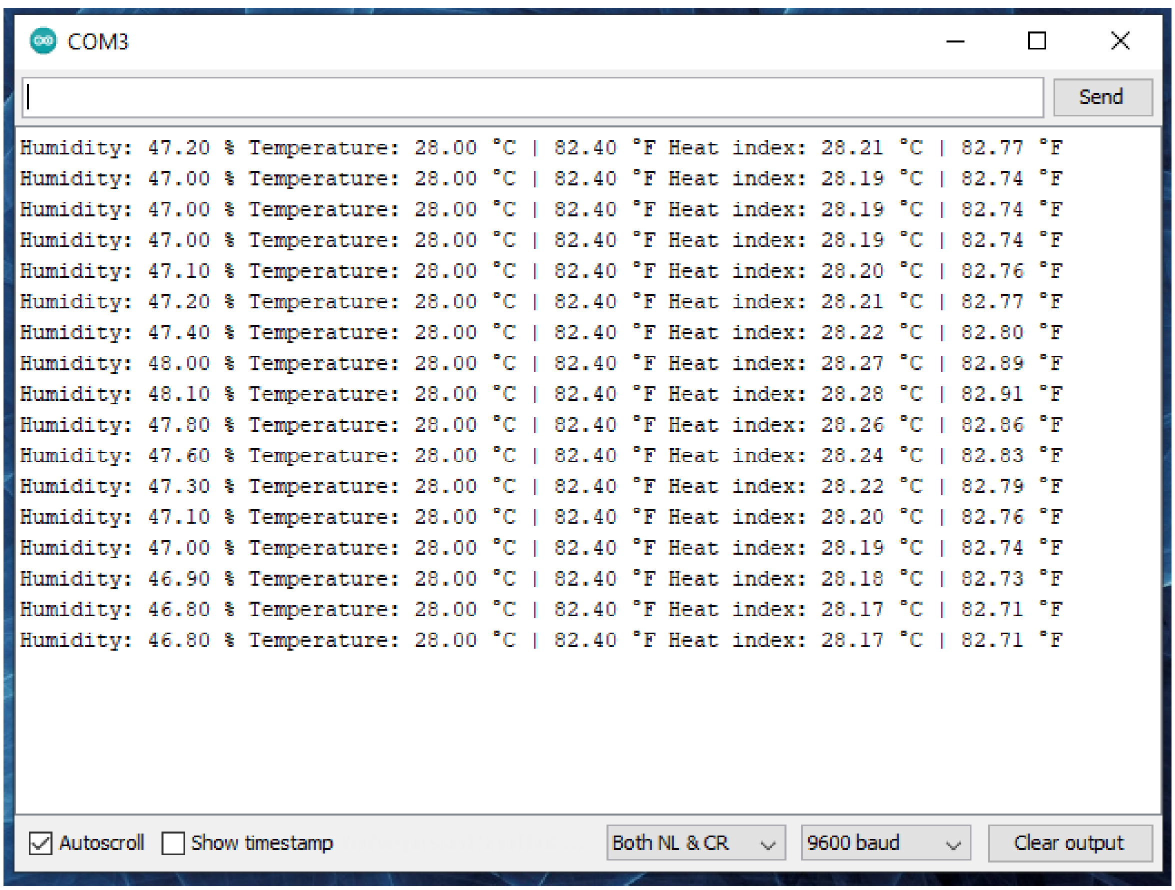
```
Serial.print(temp);
Serial.print(hum);
```

Lastly, we include a delay so our looping code will grab data and display the results every 2 seconds:

```
delay(2000);
```

Once the code is uploaded to the Arduino board, the temperature and humidity results from the sensor can be seen on the Serial Monitor:

Figure 8.10: Screenshot of Sensor Results



8.5 Bluetooth Module Testing

The Bluetooth module will be a key component of the main control unit. It is what will enable communication between the main control unit and the various sensors around the warehouse. This component will need substantial testing to ensure its reliability in receiving the required data from the sensors.

8.5.1 Testing Setback

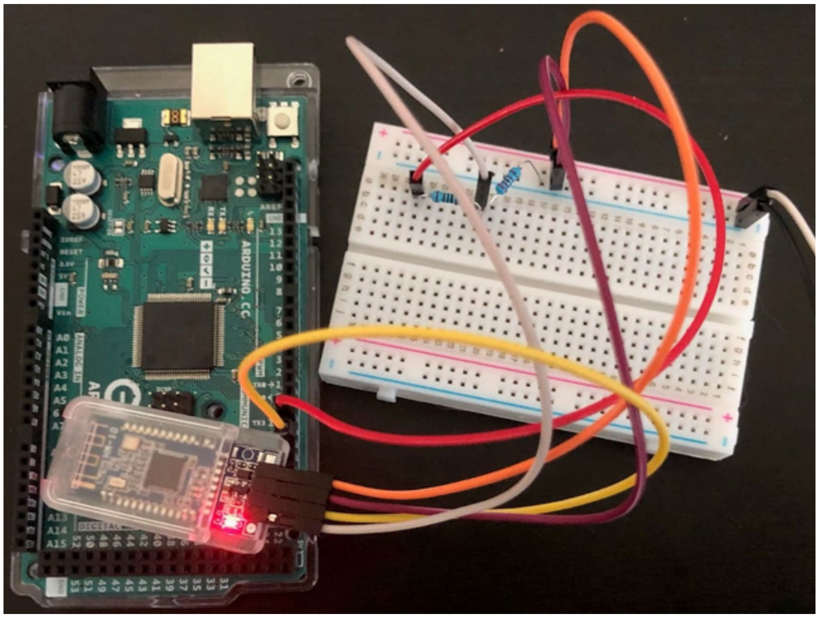
Our initial testing did not go as planned since the HM10 we ordered turned out to be a clone of the original module. Thus, we were left with the following options: use another module, or try the painful process to flash a different working available firmware. Attempting to “fix” the cloned module by re-flashing to the HM-10 original firmware, comes with some advantages and disadvantages.

The advantage of flashing the HM-10 original firmware to our clone device would mean we wouldn’t have to buy a different module; we would be able to move on with our testing rather than wait a few days for shipping. The disadvantages would be that we would need to solder and flash each of our 5 modules which would be very tedious and complex to do. Since our attempt to first flash one of our clone modules didn’t seem to work anyway, we decided that the disadvantages outweigh the advantages, and we would place an order for the original HM-10 modules.

8.5.2 Wiring and Voltage Testing

The first test to perform will be to ensure voltage of the receiving line is dropped to 3.3V. The RX and TX pins of the HM10 use 3.3V logic levels, so we use a simple voltage divider at the RX pin of the module, and we do not connect it directly to the Arduino board. This is done because the Arduino Mega will output 5V DC to its transfer line. The HM10 will need that voltage dropped to 3.3V to prevent burning out the Bluetooth module. Also, note that the RX pin of the module goes to the RX Pin of the Arduino and the TX Pin of the module goes to the TX pin of the Arduino. Once the wiring is set up, we are now ready to power up the Arduino, and test a successful connection of the Bluetooth module to the Arduino Mega. Power is supplied to the HM10 to guarantee the module powers on as it should. Before plugging in the USB cable, we press the button on the Bluetooth module to enter the command mode. The LED on the Bluetooth module is monitored, the flashing status ensures that it is working correctly. When the LEDs on the module start flashing slowly, that means that we have entered the command mode successfully.

Figure 8.11: Arduino Mega Connected To HM10



Once the wiring is set up, we are now ready to power up the Arduino, and test a successful connection of the Bluetooth module to the Arduino Mega. Power is supplied to the HM10 to guarantee the module powers on as it should. Before plugging in the USB cable, we press the button on the Bluetooth module to enter the command mode. The LED on the Bluetooth module is monitored, the flashing status ensures that it is working correctly. When the LEDs on the module start flashing slowly, that means that we have entered the command mode successfully. Now, all we have to do is to upload an empty sketch to the Arduino board. Before pressing upload remove the wires from the serial port of the Arduino board and connect them again after the sketch has been uploaded. The next step is to open the Serial monitor to send some AT commands to the Bluetooth module.

Table 8.6: HM10 Connections to Arduino

Bluetooth Module Pins	**MCU pins**
VCC	5V
GND	GND
TX	RX2
RX	TX2

8.5.3 Stable Connection Testing (Pairing)

To test the pairing of two Bluetooth modules, we will establish one Arduino Nano to have an HM10 module set as master (to receive data), and another Nano to have a DHT22 sensor and an HM10 module set as slave (to send data). This is done by plugging the Arduinos to a computer and configuring the HM10 to the correct mode. We will set up the slave module first, then the master module using AT commands within the Arduino's Serial Monitor.

While setting up the master, we establish a connection to the other module by providing it the IP address of the slave. Once this is done, the two Bluetooth modules will now be paired together, and are ready to begin transferring data to each other.

8.5.4 Data Communication Testing (Code Logic)

To get data wirelessly from sensors within the Arduino code, we can use the *EasyTransfer* library which allows for sending data structures to the serial port. By utilizing this, we can send structured data that contains both temperature and humidity via the connected Bluetooth modules. Two EasyTransfer objects are necessary to achieve bi-directional communication. One for sending data, and the other for receiving data. Additionally, an Arduino library for the DHT series of low-cost temperature/humidity sensors is needed to be able to read the data from the sensors. For example:

```
struct EX_DATA_STRUCT {
     float temperature;
     float humidity;
};
EX_DATA_STRUCT data;

DHT dht(DHTPIN,DHTTYPE);  //Sensor object
EasyTransfer ETin, ETout; //2 EasyTransfer objects

// Save readings from sensors to struct variables
data.temperature = dht.readTemperature();
data.humidity = dht.readHumidity();

//then send structure to Bluetooth module
ETout.sendData();
```

The code logic to be uploaded for each Nano goes as follows:

The slave module will have two data structures, one to contain temperature and humidity data, and the other to contain a boolean acknowledgement for when the data has been received (initially declared as false). There will be a counter to ensure that data is being sent to the master module every 2 seconds, a DHT type object to read data from the sensor, and two EasyTransfer objects (ETin and ETout) to send and receive data to the other Bluetooth module.

Within the initial set-up section of the code, the baud rate is set to the same rate listed from the AT command for serial data transmission. Additionally, we begin the serial data transmission for the DHT sensor object and establish the two EasyTransfer objects (ETin and ETout). ETin is for receiving data regarding the acknowledgement data structure, and ETout is for sending data regarding the temperature and humidity data structure.

Within the continuously looping section of the code, we check if the ETin EasyTransfer object has received any acknowledgement data from the master module, as well as if the acknowledgement boolean returns true from the master module. If both conditions are true, we can assume that the data from the sensor has been sent to the master module successfully. Otherwise, if no acknowledgement data is being received from the master module, we increment the counter. Once the counter reaches about 2 seconds, we read from the DHT sensor data pin (temperature and humidity) and send the structure via ETout to the master module. Immediately after sending sensor data to the master module, the counter is set back to zero, and the acknowledgement boolean gets set to false.

Using similar logic, the master module will also have the same two data structures, one to contain temperature and humidity data (weather), and the other to contain a boolean acknowledgement for when data has been received (initially declared as false). Like the slave, there will also be two EasyTransfer objects (ETin and ETout) to send and receive data to the slave module.

Within the initial set-up section of the code, the baud rate is set to the same rate listed from the AT command for serial data transmission. Additionally, we establish the two EasyTransfer objects (ETin and ETout). ETin is for receiving data regarding the weather (temperature and humidity) data structure, and ETout is for sending data regarding the acknowledgement data structure.

Within the continuously looping section of the code, we check if the ETin EasyTransfer object has received any sensor data from the slave module. If this condition meets true, set the boolean acknowledgement to true, and send the structure via ETout to the slave module. Otherwise, if no sensor data is being received from the slave module, we set the boolean acknowledgement to false.

In summary,

- Include the EasyTransfer library and create two EasyTransfer objects
- Create one data structure to send temperature and humidity data, and another data structure to receive boolean acknowledgments
- Code for each slave module will read data from the DHT22 sensor every 2 seconds if the data acknowledgement received from the master module returns false, and if no EasyTransfer object was received from master
- in order to send a boolean acknowledgement to the slave modules, the code for the master module will continuously check if an EasyTransfer object was received from any slave module

8.5.5 Multiple Communication Testing

To test that one Bluetooth module is able to connect to several other Bluetooth modules, we will use a looping algorithm. The logic for the algorithm basically continuously loops the communication of one master module to multiple slave modules by first connecting to one test slave module, retrieving data from the connected DHT22 sensor, disconnecting, and then repeating the process with another test slave module. Once the algorithm proves to establish a stable connection between one test master module and multiple test slave modules, then the communication range of the Bluetooth modules will subsequently be tested.

8.5.6 Distance Testing

To test the required long-distance communication between the Bluetooth modules, we will measure out a distance that will be similar to the warehouse conditions. This will be done by placing each HM10 with a DHT22 sensor in various locations that have atmosphere differences such as a garage and an air-conditioned room. Each tested wireless module with sensor will be moved to a specified distance, and the communication quality will be assessed by reading the temperature and humidity data from each module's sensor.

8.6 TFT Touch Screen Configuration Testing

Below is the section that discusses the TFT display and touch screen testing. These sections are split apart because the display and touch screen are two separate components that are brought together in one package. The display is a

3.5” color display with 320x480 pixels. This information will be important during the display testing since the drivers for the screen use the pixel information to control the height and width of the display window and where to place the objects on the screen. The touch screen uses resistive touch technology. The resistive touch technology will be used in conjunction with the screen to allow for local control of the main control unit. The resistive touch technology should be accurate enough to sense when a small area of the screen is touched and use that signal to control what the overall system is doing.

8.6.1 Display Testing

The first part of testing the screen is soldering the pins that will be used for data transfer, control, and touch screen signals. This can be done by counting the number of pins and cutting the pin header to fit all the holes. Since 8-bit data transfer is used, soldering is only required on the side that uses 8-bit. The side used SPI data transfer can be left alone. Place the pins to be soldered in the selected holes with the long side of the pins facing down. Once this is completed, the soldering can be accomplished in the Senior Design lab at the provided soldering stations.

After the soldering is completed, the display testing for the screen can begin. The display should be connected to arduino mega based on the wiring table shown in section 6.2.4. This display will use 8-bit data transfer to send the display data to the screen. There are four other wires connected that will send control signals to the screen and power the backlight of the screen. The last wire for the display is the reset wire which can be used to reset the screen. Once all the wires are connected successfully the arduino can be plugged into the computer, and programming can begin.

To begin the programming the Arduino IDE should be opened. Once opened, the drivers for the 3.5” TFT display should be downloaded. Once downloaded the drivers will need to be modified since the provided drivers are for both a 2.8” display and a 3.5” display. To change the drivers for the 3.5” screen size the driver should be modified by changing the values entered in to TFTWIDTH and TFTHEIGHT field. Once this is completed the test code provided for this screen can be uploaded onto the arduino. Once successfully uploaded onto the arduino the screen should run through the test code and the screen should display various shapes and words.

Figure 8.12:. Modifying the Display Driver

Adafruit_TFTLCD.h | Adafruit_TFTLCD.cpp

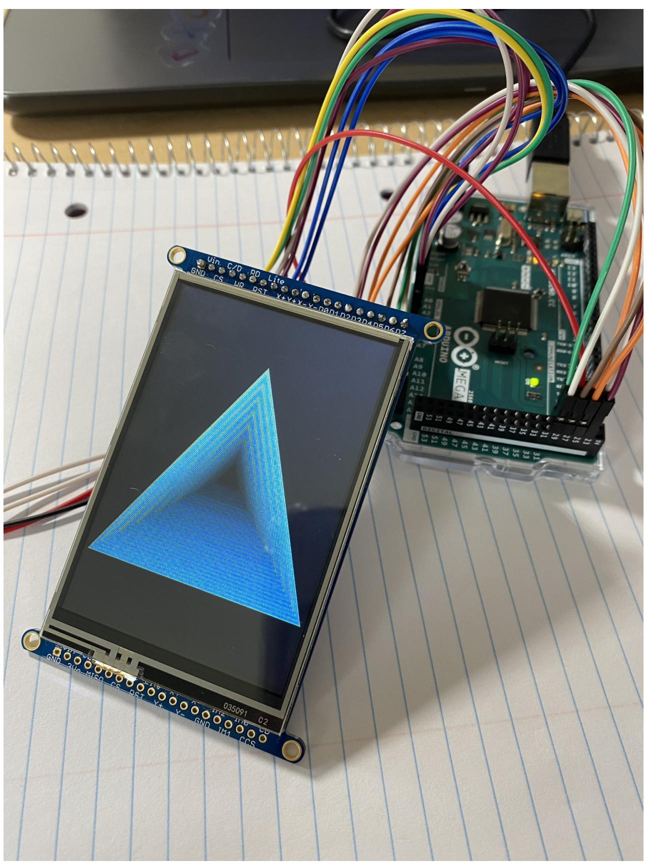
```
#if defined(__SAM3X8E__)
#include <include/pio.h>
#define PROGMEM
#define pgm_read_byte(addr) (*(const unsigned char *)(addr))
#define pgm_read_word(addr) (*(const unsigned short *)(addr))
#endif
#ifdef __AVR__
#include <avr/pgmspace.h>
#endif
#include "Adafruit_TFTLCD.h"
#include "pin_magic.h"
#include "pins_arduino.h"
#include "wiring_private.h"

//#define TFTWIDTH   320
//#define TFTHEIGHT  480

#define TFTWIDTH 240
#define TFTHEIGHT 320

```

Figure 8.13: Successful Execution of the Test Code



In summary,

- For data transfer this screen uses 8-bit data communication.
- The first step is soldering the pins to the 8-bit data transfer points, screen control points, and touch screen control points.
- The next step is to connect the screen to the arduino in accordance with the wiring table.
- Next, Download the provided drivers for the display.
    - THe driver will need to be modified for the 3.5” display
- Finally, upload the test code to the arduino. The display should run through the test code by displaying different shapes and words.

8.6.2 Touch Screen Testing

Once the screen has been tested successfully, testing the touch screen functionality can begin. The touch screen testing must be completed after the initial display testing because the drivers for the touch screen utilize the drivers for the display to know screen size and locations on the screen. To begin the touch screen testing, the wires for the touchscreen must be connected to the Arduino. These wires must use the analog inputs on the Arduino to send an analog signal that is generated by the resistive touch screen. The touch screen wires can be connected based upon the table shown in section 6.2.4. Once the wires are connected the programming of the touch screen can begin.

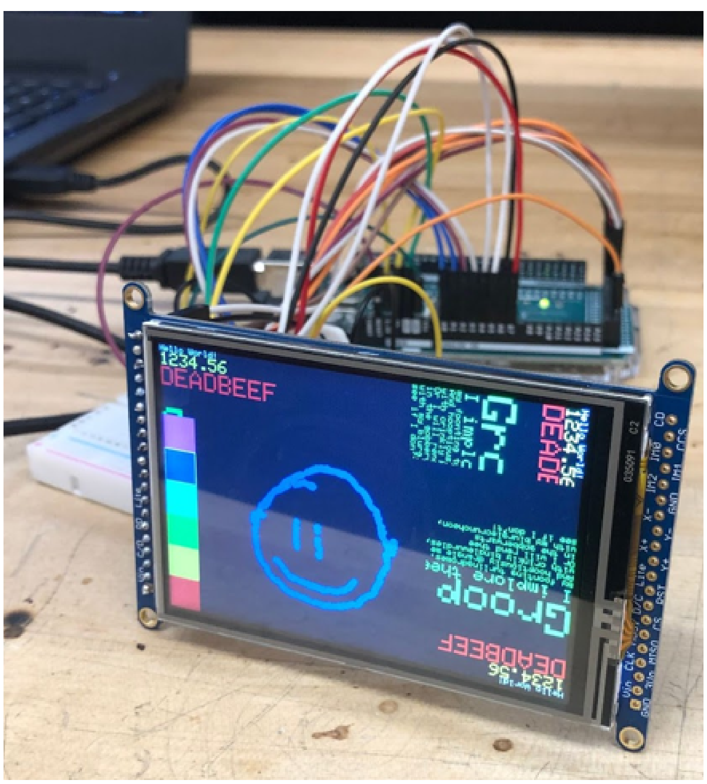
The first step of programming the touchscreen begins with opening the Arduino IDE. We will use two programs to test our touch screen. One program will test the setup of using buttons, and the second program will allow paint drawing on our touchscreen.

Some adjustments need to be made for programming the touch screen functionality. The source code for the touch screen should include the following libraries: Adafruit_TFTLCD, Adafruit_GFX, and TouchScreen. We then define the CS, CD, WR, RD and RESET pins used by the LCD display. In order to activate the touch screen, we identify the YP, XM, YM, XP pins that these ports are plugged in to. We then need to define the minimum and maximum X and Y values, which is done by calibrating the screen. So, keeping this code aside for now, we can upload the “touchscreendemo” example within the Arduino IDE that comes with the library for calibration (File -> Examples -> TouchScreen -> touchscreendemo). After uploading, we open the Serial Monitor from the IDE to begin calibration. We calibrate by touching the top left corner of the display and get the X and Y values displayed on the Serial Monitor. These values will then be entered as the maximum X and Y variables in the original testing source code. The same process will be done to get the minimum X and Y values by touching the bottom right corner of the display. Once the minimum and maximum X and Y values have been entered, the testing program can be uploaded to the Arduino.

To further explain the source code for testing buttons, a boolean variable “buttonEnabled” is initialized to true in order to enable the button. The map() function converts the values we get from the TouchScreen library into pixels, and this is the resolution of the screen we use in pixels. Thus, we can use these X and Y values of the touch point in pixels to make clickable buttons. If the screen is touched within specific coordinates assigned on top of a button, we then activate (enable) the button. If we touch the button shown on the screen, the program will display a message and deactivate the button coordinates (assign buttonEnabled to false).

The source code for testing drawing on the screen is like the previous testing code. In a similar way, the painting test code also uses the map() function to convert values from the TouchScreen library into pixels. Thus, these X and Y values of the touch point in pixels are used to draw on the screen. When the screen is touched within specific coordinates assigned on top of the colored buttons, that specific color is then selected to paint with.

Figure 8.14: Successful operation of the Touchscreen



In summary,

- Connect the wires that control the touch screen to the arduino using the displays wiring table as a reference
- Download the provided drivers for the touchscreen.
  - The touchscreen drivers are based on the drivers for the display.
    - Display testing should be completed prior to testing the touchscreen.
- Modify the drivers to use the analog pins that the touch screen is connected to.
- Upload the test code for the touch screen onto the arduino.
  - The test code allows you to draw images on the screen.
- The screen may require calibration to ensure the accurate touch positions.

9. Software Testing

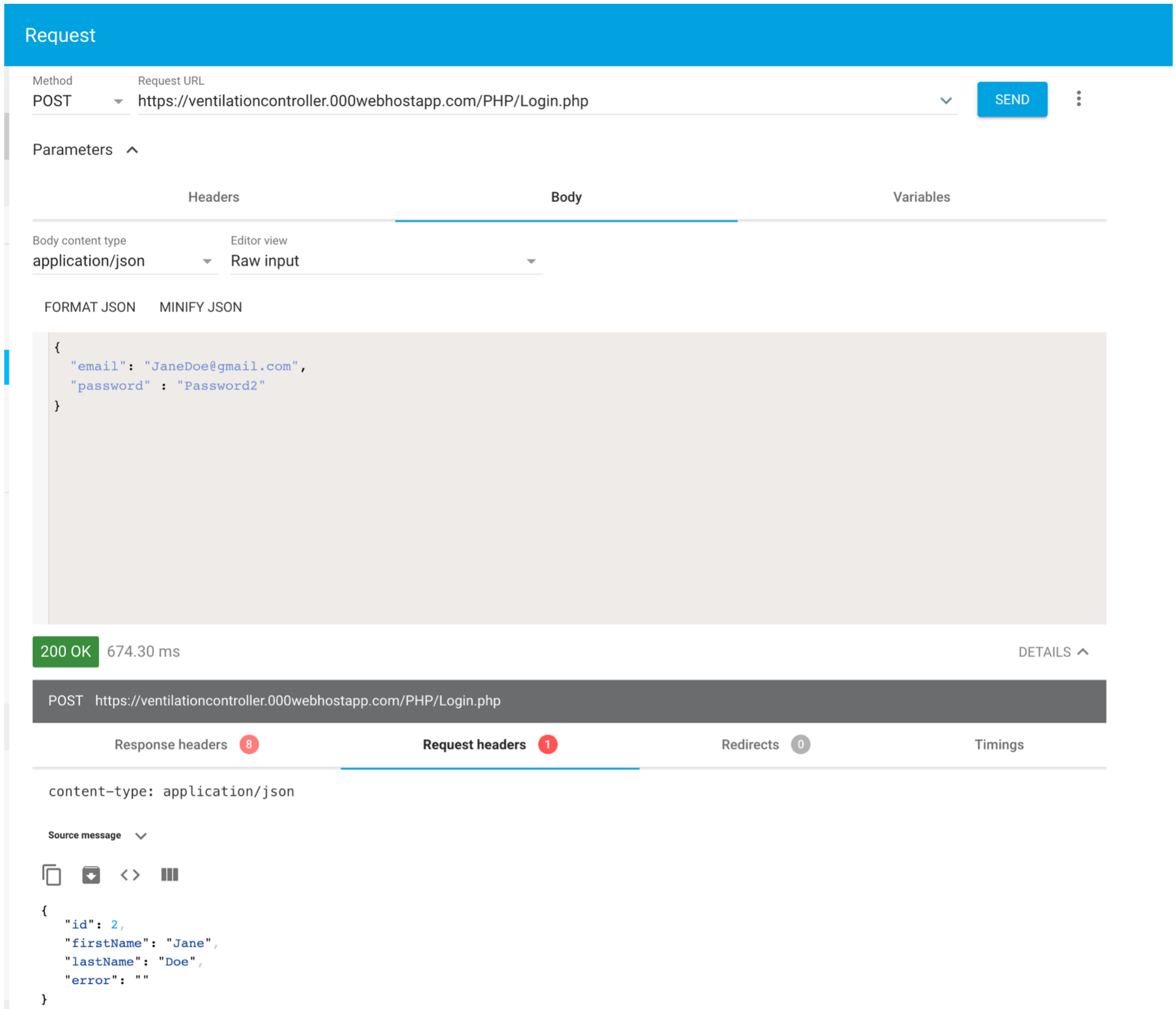
This section will go over the various methods that we utilized to test the web application. These tests will ensure that the website components are working within its design parameters, and all the parts of our development stack are communicating with each other.

9.1 API Endpoints

Here we will test the API endpoints that are used to login a user, and make sure they already exist in our database before letting them access the rest of the information available. We will also be testing that the endpoints for the API we will be using to scrape the weather data also works. To test these endpoints, we will first have to write the PHP files, these files will connect the front end, which is what the user will see, with the back end, and will make this communication as smooth as possible. Once the PHP files are written we will use ARC for some quick tests.

To use and test these APIs, we will be using JSON, which stands for JavaScript Object Notation, it is used to send formatted data from a server to a web page. This data is formatted in key/value pairs, each pair is separated by commas. A key is only one type, a string, enclosed in quotation marks, on the other side, the value can be any of the common types. The JSON syntax is obtained from JavaScript object notation syntax, because of this JavaScript programs are able to easily convert JSON data into native JavaScript objects. Each JSON object is written inside curly braces, and can contain many key/value pairs. While the syntax was made similar to JavaScript objects, any of the available popular programming languages can write JSON objects.

Figure 9.1. Successful Testing of the Login Endpoint



The first endpoint that was tested was the login API, which is in charge of giving the user access to the dashboard. When testing this out there were a couple of issues we ran into. First and foremost, since 000webhost is new to us, we were not too sure about how the URL would work to reach each of the API endpoints, so there were many trials and errors to get the right one working. Once we were finally able to pin down the correct URL, the next task was defining the JSON package.

Since the login will require its user to use the email and password that was provided in the sign up form, we had to send that formatted with the existing users we inserted at the beginning, during our database set up. This JSON package contains the email and password, as seen in the grey, before sending it though we had to define the header to JSON to be able to send it and properly test it. Once all three fields were defined we were able to send the package to our API, and as seen in the screenshot, because it was successful we got a 200 Ok code, and in the response and request header tabs we can see the users information. For this test, the user is Jane Doe and they are the second person in our users table, which is pointed out by their id in the response. For the other endpoints, the testing format will be the same. The only difference will be the method utilized for the request.

In summary,

- To test the API endpoints we have to actually know where the file is in the server to be able to write the correct URL to send the JSON package.
- The testing steps for each API will be similarly formatted, the only difference will be the method used to test it and whether or not it will have a JSON response.
- The first endpoint tested was the Login endpoint, which required a POST method and returned the users information.

9.2 TFT Touch Screen GUI Implementation

UTFT and *UTouch* libraries enable easy use for TFT screens. After including these libraries in the Arduino code, UTFT and UTouch objects will need to be created. The parameters of these objects will define the model of the TFT screen and shield, which can be found in the documentation of the libraries. Next, fonts contained in the libraries need to be defined, along with any variables needed for the programming of the graphical user interface.

Within the setup section of the code, we need to initiate the screen and the touch capabilities, as well as define pin modes for any input and output. Afterwards, a custom function (such as *drawHomeScreen*) will initialize the home screen of the program for the controller. The UTFT library offers a couple of very useful functions such as. setBackColor(), .setColor(), .setFont(), .print(), .drawLine(), and .fillRoundRect(), which are important for creating a graphical user interface. These functions can be used within the *drawHomeScreen* custom function to begin constructing the layout design and touch buttons on the home screen.

Within the loop section of the code, we can make the buttons on the home screen functional. Using the .read(), .getX(), and .getY() functions from the UTouch library, we can get the x and y coordinates of where the screen has been pressed by the user. We can then check if the coordinates are in a certain area of the screen or fall on top of a button. If that condition is met, it then makes the button useful to do whatever we need, like call other custom functions or change some data.

Another custom function, *drawInfoScreen*, will be called only once when the user presses the information button on the top right corner of the screen. In this function, we would draw all the graphics on the screen in a similar way in which we did for the home screen. Additionally, each screen will repeatedly need to call other methods that grab information from other modules in order to display their data.

In summary,

- There are libraries available to use for the touchscreen lcd screen we will be using, this includes commands like setting color, etc.
- There are commands for getting the coordinates where the user touches the touchscreen to have it do what they need it to do.

10. PCB Design

For the PCB, or Printed circuit board, our project is going to use a software called Eagle which is made by Autodesk. This software allows users to import the selected parts for their printed circuit board design and lay them out in a logical format. The software then allows the users to connect the different components with wires to give a general schematic of how everything on the printed circuit board is connected.

After all the components are layed down and all of the necessary connections are made the software takes your inputs and lays them into a PCB. The wires can then be turned into traces to connect all the components on the printed circuit board to complete the design.

In our PCB design, we used both Altium and Eagle software. Both allow us to import individual parts and connect them within the schematic, and then to import them to the PCB. Both also have an autorouting feature that allows The subsequent PCB to become cleanly wired very quickly. While the group had experience with Eagle, Altium had not been used prior by our group, and was suggested as by our mentor Chris Neiger.

Since both Eagle and Altium provided free student licenses, price was not an issue, and we decided to expand our knowledge base and use Altium for our sensor unit PCB design and Eagle for our control unit PCB design. Each of the two platforms provides the positives listed below:

Altium Notable Features:

- Cloud-based collaborative working, so multiple team members can update and edit designs
- Advanced routing features including walk around, differential pairs, obstacle ignore, and hug
- Design simplification converting multi-channel designs into stepwise designs
- Easy to create real time Bill of Materials

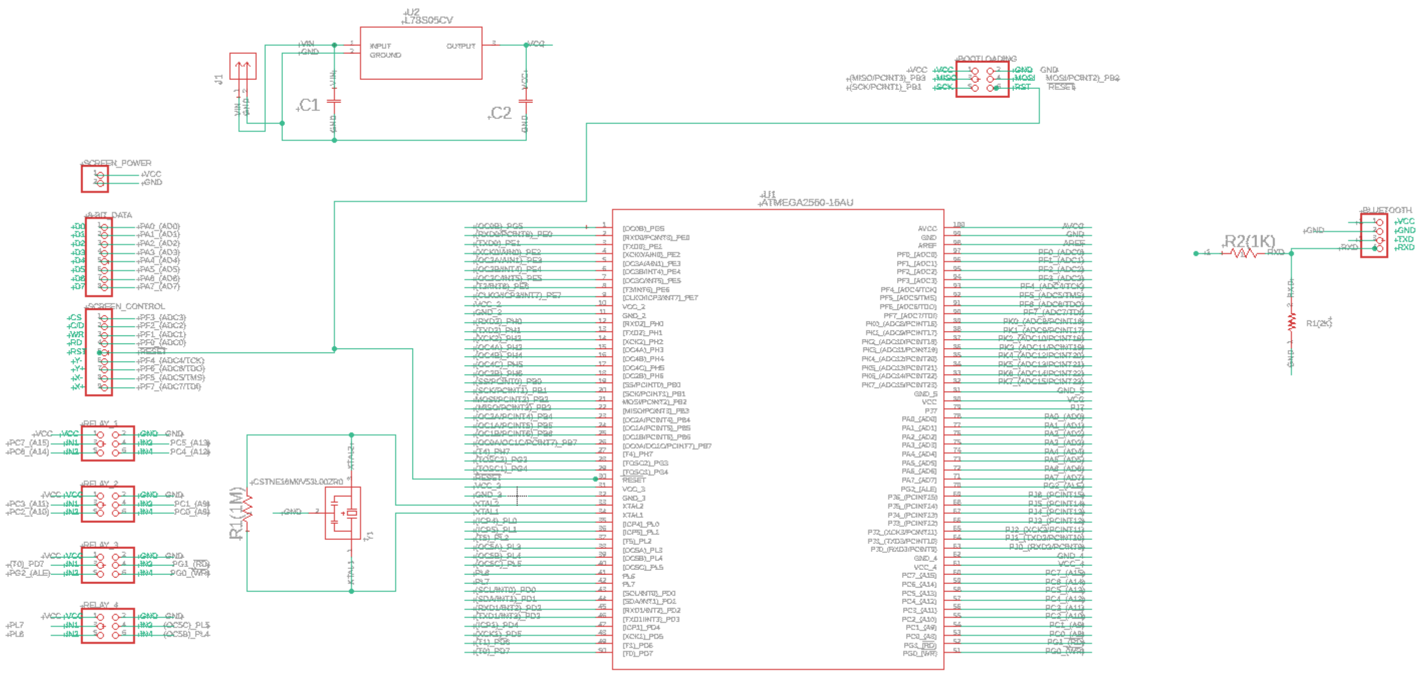
Eagle Notable Features:

- Backward/Forward annotation between PCB and Schematic
- More powerful simulation tools with spice integrated
- Group has some experience and familiarity with Eagle

10.1 PCB for Main Controller Unit

The design of the main control unit will start with the ATmega2560 which will be centrally located on the board. One of the sides of the board will consist of the components for the voltage regulator that will reduce the voltage to make it usable for the CPU. On the opposite side of the board a header of male pins will be connected to various input/output pins on the CPU. This header will be used to connect and drive the touch screen display to the main control unit. On one of the adjacent sides there will be holes to through mount both the bluetooth module and wi-fi module. These through hole connections will be connected to two of the four USART connections on the CPU. These connections will contain the voltage dividing circuit to obtain the 3.3V power level required for the communication modules. On the side of the board opposite to the communication modules there will be a double row of pin headers to connect the relay modules to the main control unit. Six male pins will be required for each relay module. Of these six pins, one pin will connect to 5V VCC, one pin to ground, and the other four pins will connect to input/output pins on the CPU. There will be four relay modules to connect, which will create a total of 24 male pins. If room is available on the PCB, male pins may be added to allow the unit to be expanded for future use. All male pins, through hole connections, and surface mounted components will be soldered to the board to ensure permanent connections are created. For our design, we expect each of our sensors to be able to utilize identical PCB designs.

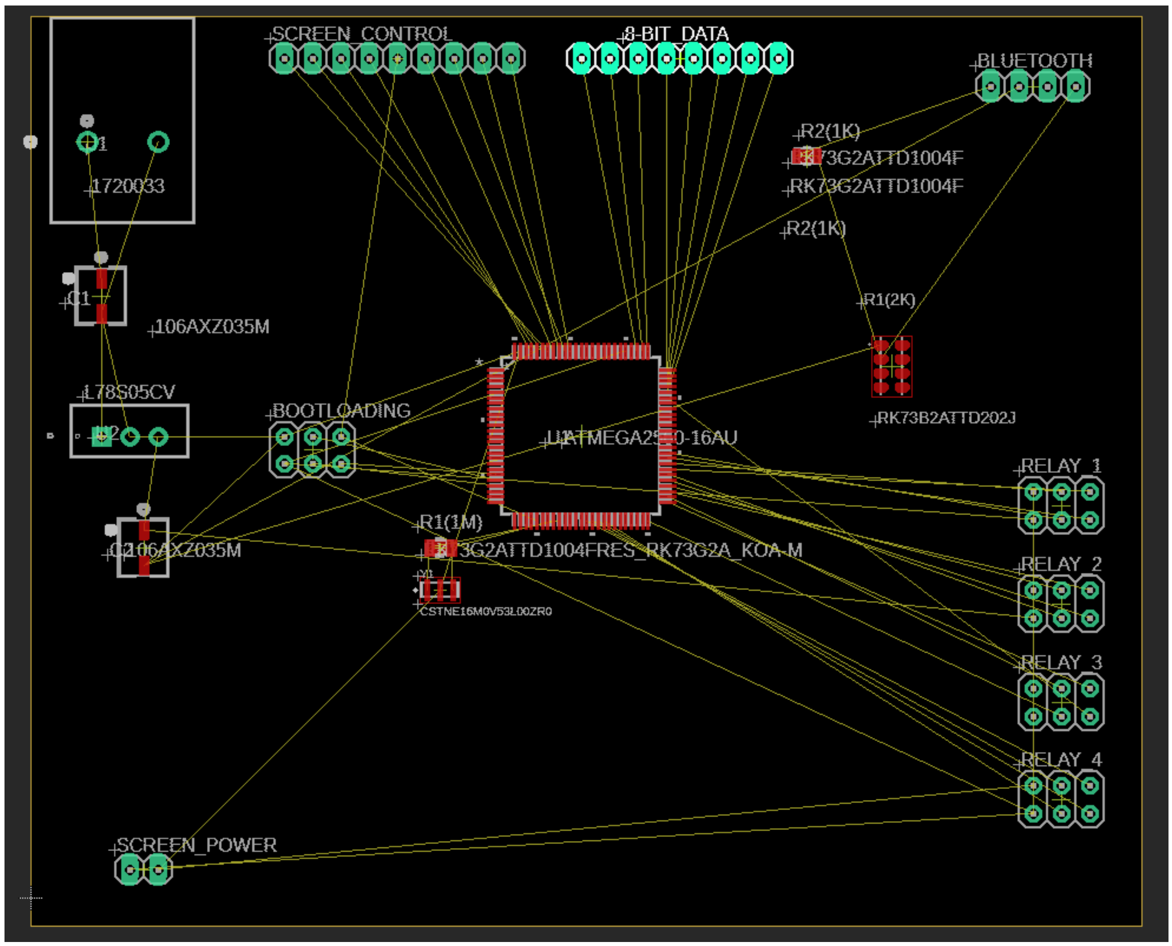
Figure 10.1: Eagle PCB design schematic



This initial design was created using the eagle software. Many of these parts were selected from the research conducted earlier in this paper. The schematics for many of these parts had to be downloaded to ensure the right foot print and layout of the PCB would match the parts that are planned on being ordered. The pin selection for many of these parts came directly from component testing. The testing for many components such as the wifi and bluetooth module have not been completed yet. Upon successful testing of the parts the pins for these components will be added to the Eagle schematic to complete the design. Currently the initial design of the power supply for the PCB was based upon a buck converter that was designed using Texas Instruments Webench. The buck converter design is still the main plan for powering our PCB, but semiconductor shortages have limited the ability to purchase any of the buck converters from Texas Instruments.

A back up design was implemented in this Eagle schematic that consists of a 5V linear voltage regulator. This linear voltage regulator is rated for 2A which is more than enough power for the PCB. However, The Linear voltage regulator is in a 7805 format. This design will require a heatsink to be attached to the linear voltage regulator to prevent it from overheating. During normal operation the power demands for this PCB will be low, but when the relays open to turn the vents on, the power demand will increase drastically. The increase in demand will require the heatsink to prevent overheating and premature failure of the PCB. Basic pins have been used to implement the screen and relay modules. This was done to help decrease the footprint of the PCB to allow for more flexibility of the installation.

Figure 10.2: Initial PCB layout



Above is the initial footprint of the PCB. THe locations for the various pins and components were chosen to minimize crossover of the air wires. This will aid in the trace routing process for the final design. The component locations are likely to change as the Bluetooth and Wi-fi module testing continues.

In summary,

- Eagle has been chosen as the design software for the controller PCB.
- The ATmega2560 is centrally located due to the large number of connections made to it.
- The PCB sides are divided up to group similar components for ease of understanding the layout.
- The sides were chosen to minimize the number of crossover connections for the PCB.

- Due to supply constraints a linear voltage regulator was implemented as a back up design instead of the buck converter.
- Bluetooth and WI-fi module testing is the limiting factor for finalizing the first draft of the PCB design.

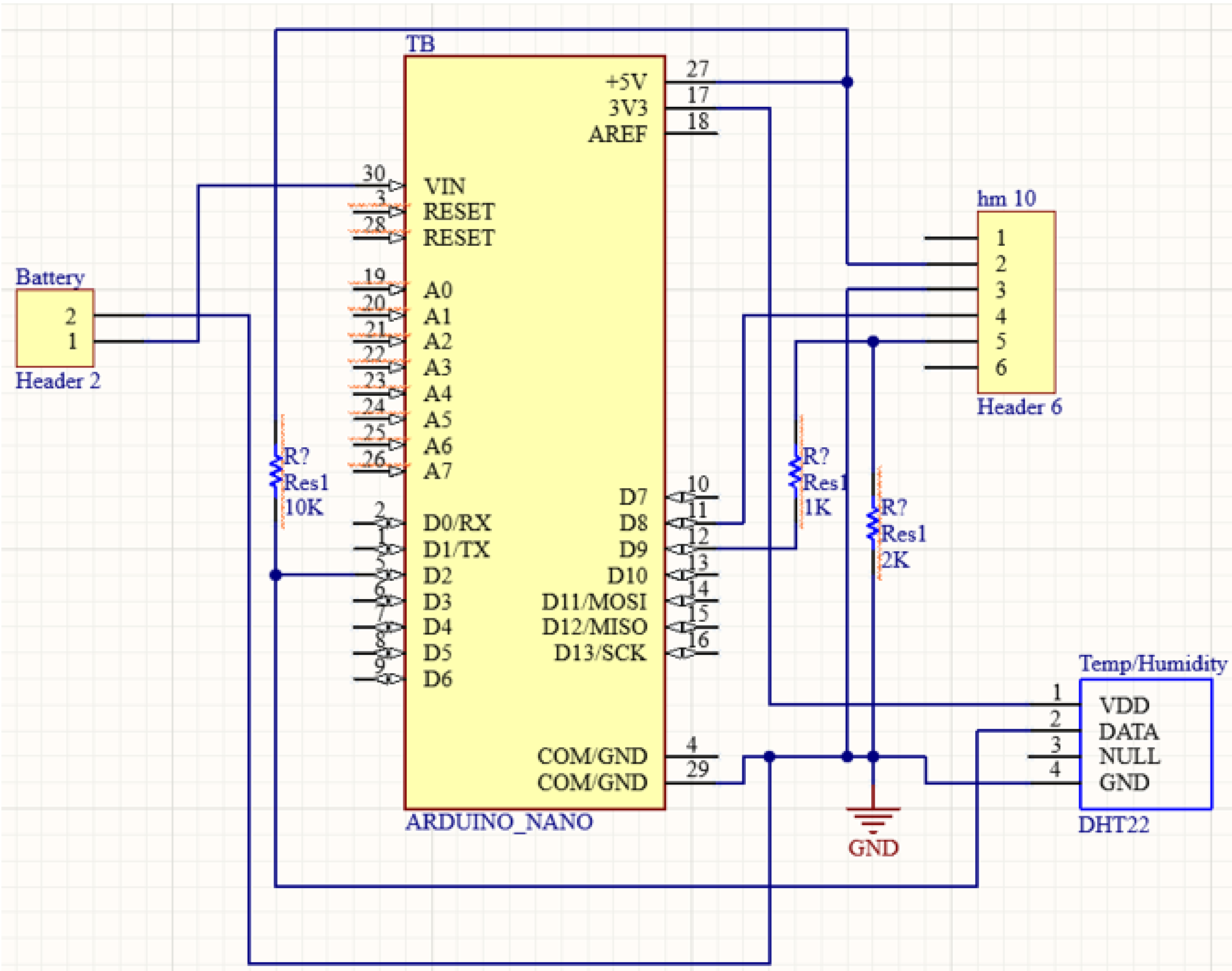
10.2 PCB for Sensor Unit

For the Sensor Unit's PCB design, we discussed software options with our sponsor Chris Neiger and decided to use Altium. He recommended this software so that we could learn a new system and use him as a support resource as needed. Altium also has a resources and support page that provide help and tutorials on things like creating a PCB layout from a schematic. After looking into Altium, we saw that there was a free student license that lasts through the project's duration, and that Altium has a schematic wizard that allows us to easily move between Eagle and Altium, so that going between the two software options would be possible. Altium's PCB design software is used from automotive and aerospace designs to consumer electronics and medical devices design. There also exists Altium 365 that allows designers to collaborate in the design process for the entire PCB, this software also has the ability of embedding the PCB design into a website along with being able to share it, there is also no CAD tools or experience required in using it. There is also a free ECAD software called circuitmaker that is used to create PCB designs however it does have some limitations and won't be very helpful for us. During the design process, schematic components were needed for the Arduino Nano Every, DHT22, HM 10, Battery, and 1k, 2k, and 10k resistors, as selected earlier.

While the schematics for the resistors were included, everything else needed to be downloaded and imported. The DHT22 was imported, while the battery was set as a 2 pin header, with pin 1 being Vin and pin 2 being ground. Ample space was left on the PCB so that the battery casing can be attached to the board securely. One side of the battery casing has the on/off toggle, while the other has the screw that allows battery replacement. Since both will need to be accessed, the battery was placed on the edge of the board, so that the on/off toggle is slightly off the edge of the board and can be accessed.

Due to the testing issues with the HM 10, a 6 pin header was used as a placeholder. While the HM 10 module we originally purchased has 4 pins, other modules use 6, with pins 1 and 6 not connected. In case switching parts is needed, we decided to use the 6 pin so that the PCB design would not need to be altered later on if a 6 pin HM 10 is purchased. Space was left around the 6 pin header to allow for the size difference of the HM 10.

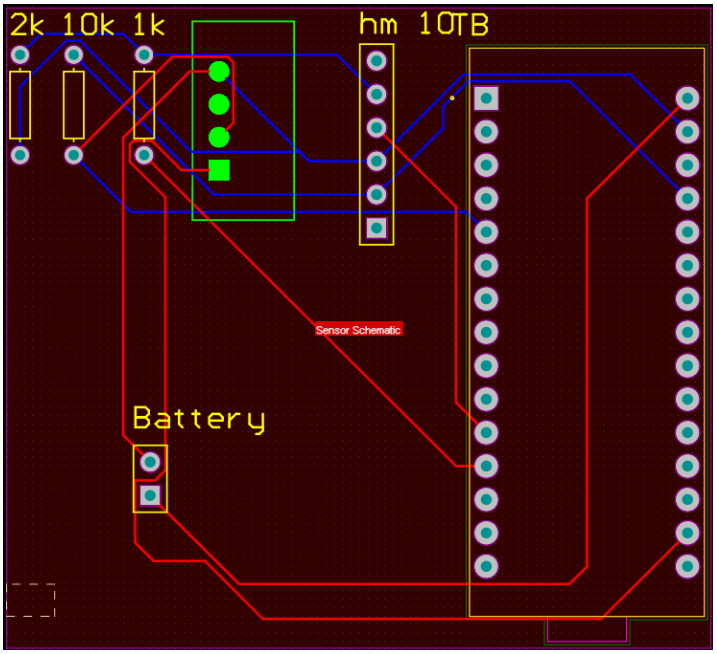
Figure 10.3 Altium Design Schematic



The figure above shows the wiring that was created for the Schematic in Altium. All of the sensor components are wired as they were on the breadboard during testing, since the only issues with testing were from the HM 10 part not working correctly, as opposed to connection issues.

The schematic was then imported to the PCB. Autorouting was used to give a clean schematic, the 2 pin header representing the battery was placed in the space the battery casing is expected to sit for picture reference, and will be moved closer to the edge of the board before printing.

Figure 10.4 Altium Design Schematic



As shown above, the excess room for when the HM 10 module is added and the battery compartment. Care was also taken to have the battery and the Arduino Nano Every on corners, so that any heat they produce affects the other components less. This also lets us plug into the Nano Every to alter code if needed. Additionally, the battery needs to be placed so that it is slightly over the edge of the PCB so that the on/ off switch can be toggled. Overall the size of the PCB is much smaller than the maximum acceptable size of 4” x 6” that had been discussed in planning the sensor. This means that even with the casing that will protect the final sensor PCBs, it will still allow the sensor to be within size restraints.

In summary,

- Altium has been chosen as the design software for the Sensor PCB.
- Altium has an import wizard that lets us import from Eagle, which may be necessary for later PCB designing/printing
- The Arduino Nano Every is located on the edge of the board to avoid heat affecting other components and to allow the nano to be reprogrammed even when on the board.
- The battery casing is modeled with a 2 pin header on the edge of the board to allow for both battery on/off toggle and battery replacement.
- A 6 pin header is being used for the HM 10 until we can find a brand of HM 10 that works.

11. Administrative Content

This section of the document will be to show how well the team can manage their time and budget on this project. The administrative portion of our responsibilities includes finances and timeline management. The budget will be decided by our sponsor and will be listed to show each expense that makes up our ventilation controller.

11.1 Budget

For this project we are being financed by Chris Neiger, the estimated Budget is currently $1,588.60.

Budget Break Down

Description	Price	Quantity	Total
Arduino Mega 2560	40.3	2	80.6
Breakout Board, ESP8266 Wifi module	7.93	2	15.86
HM-10 Wireless Bluetooth Transmitter	7.99	4	31.96
Jbtek 4 Channel DC 5V Relay Module	9.59	4	19.18
TFT 3.5" Color Touch Screen Display- ILI9341	26.98	1	26.98
DHT22 Temperature and humidity Sensor	9.95	4	39.8
Breadboard jumper wires, 120 piece	5.99	1	5.99
Breadboard- 4 piece	5.99	1	5.99
Software-Price per month	30	6	180
Assorted discrete electronic component kit	70	1	70
Final PCB build	50	1	50
Travel to/from Niceville, FL- 419 miles- 16 cents/mile	67.04	6	402.24
Hotel in Niceville, FL- $110/room/night	110	6	660
		Total	1588.6

11.2 Finance

Our Sponsor is providing us with a lump-sum of two thousand dollars to complete this project. In using these funds, our group is responsible for opening a bank account and keeping track of our budget and spending. Any unused money will be returned to our sponsor at the end of this project.

We looked into the creation of a corporate account, but after talking with the bank, we found that this type of account would require us to incorporate or form an LLC, get tax numbers, and would in general be more time consuming and complicated than our group and our sponsor felt necessary.

A personal account would allow us to use money under one member's name, and would not require us to do anything special legally. Gisela Griesheimer was chosen since she is our sponsor connection, physically closest to the sponsor's location, and our sponsor was most comfortable with this arrangement. We will be keeping track of purchases as a team, though Gisela Griesheimer will be in charge of keeping up with receipts and online ordering for the group.

Account statements will be checked regularly and shared with our sponsor on a monthly basis, and whenever requested. Most purchases will be made online, so we do not anticipate any issues with having the account tied to a single person. Our only anticipated costs that will not be purchased online are gas and accommodations for when the team travels to Niceville to install the system in the sponsor's warehouse. Since the entire team will be traveling together, this should also not be an issue.

At our sponsor's recommendation, we decided to open an account with Synovus Bank. This bank provides us with a free account and a debit card, so that purchases can be made by Gisela without any service fees. During the summer semester while the group members are communicating virtually, any parts needed can be shipped directly to the group member who needs them. In the fall, all members will reside in the Orlando area, and parts can be shipped to one location and distributed in person. Any large purchases outside of our initially proposed budget will be discussed with our sponsor before purchase. Fluctuation in part prices will be discussed with the sponsor if they are significantly different than anticipated in the budget report, but small changes in price are expected and acceptable.

11.3 Current Spending

The current funds that have been expended are shown in the chart below:

Table 11.1 Chart of Current Spending

Item(s)	Website	Date	Shipped to	Order Price	Shipping & Tax Price	Total
Arduino Mega	Arduino	7.15.21	David /Orlando	$40.30	$13.82	$54.12
Arduino Nano Every	Arduino	7.15.21	Gisela /Niceville	$10.90	$13.82	$24.72
HXD8357D (LCD Screen), DHT22 (Temp Sensor), ESP8266 (Wifi)	Adafruit	7.15.21	David /Orlando	$59.85	$15.75	$75.60
DHT22 (Temp Sensor) x4	Adafruit	7.15.21	Gisela /Niceville	$39.80	$14.72	$54.52
Relay, HM 10 (BLE)	Amazon	7.15.21	David /Orlando	$18.98	$1.33	$20.31
HM 10 (BLE), Lithium 9V Battery, Battery Case 3pack	Amazon	7.15.21	Gisela /Niceville	$24.87	$1.74	$26.61
Relay x3	Amazon	7.20.21	David /Orlando	$23.49	$1.65	$25.14
Ship Sensor for Testing	USPS	7.23.21	Angelica /Orlando	N/a	$8.45	$8.45
Lithium 9V Battery	Amazon	7.25.21	Angelica /Orlando	$6.89	$0.45	$7.34
Wifi Module	Amazon	7.28.21	David /Orlando	$10.99	$0.77	$11.76
HM 10	Amazon	7.28.21	David /Orlando	$10.99	$0.77	$11.76
Return Both Broken HM 10s	Amazon	7.28.21	David /Orlando	-$21.98	-$1.54	-$23.52
Running Total						$296.81

11.4 Milestone Discussion

The project milestones is a brief summary of the submission deadlines for this project, and the goals of when each section should be completed. This section covers both the submissions for senior design 1 and senior design 2. Many of the deadlines chosen are hard dates that cannot be violated in order to complete this project in a timely manner.

11.4.1 Project Milestones Summer

Below is a table of the submissions and the deadlines for the first part of this project which covers the first semester of Senior Design 1. The deadlines in this semester are tighter due it being a summer semester which is the shortest semester of the academic school year. With this tight schedule there is little room for missing deadlines.

Table 11.2: Summer Semester Milestones

Summer Semester 2021	
June 2	Begin writing document and researching parts
June 11	Submission Due for D&C 1.0
June 14	Project Set and approved - Meeting with Dr. Wei
June 25	Submission Due for D&C 2.0
July 9	At least half of the final document written
July 23	Submission Due for 100 Page Draft
July 31	Individual document portions complete
August 1	Edit, format, and assemble document
August 3	Submit Final Paper

11.4.2 Project Milestones Fall

Below is a brief table that summarizes the milestones that cover Senior Design 2. The fall semester is the longest semester of the academic calendar and will provide more room for prototyping and building this project. However, The schedule will still be tight and have little room for error. The below schedule is likely to be edited once Senior Design 2 begins and the submission schedule for each part of the project is released.

Table 11.3: Spring Semester Milestones

Fall Semester 2021	
August 8	Order parts
September 6	Begin building design and writing software
October 18	Begin integrating software and hardware
November 1	Project fully built
November 8	Begin Practicing Presentation
December 3	Final Presentation

11.5 Completed and In-Progress Work

This section contains an up-to-date checklist of the work completed along with the in-progress work that is being done. The section containing research has been removed because it is completely finished, any further research for this project will be in prompt to research which by definition cannot be foreseen, and therefore cannot go into this section that is used for planning work. Any work beyond testing is also not in this section because that work is neither in-progress nor completed as of this date.

Table 11.4: Testing Status

Status	**Work Description**	**Notes**
IP	Bluetooth Testing	Faulty Bluetooth Module- New module just recieved
IP	Wi-fi Testing	Wrong module ordered- New Module just recieved
✔	Relay Module Testing	No Issues
✔	Display Testing	No Issues
IP	Touchscreen Testing	Needs Calibration
✔	DHT-22 Testing	No Issues
NS	Power Supply Testing	All Buck Converters sold out- changing design to LVR
IP	Main Control Unit PCB design	Waiting on pin selection for Bluetooth and Wi-fi modules
IP	Sensor PCB Design	Waiting for confirmation of bluetooth pins
IP	API Design	No Issues
IP	API Testing	Issues with accessing API files

✔ - Completed
NS - Not Started
IP - In Progress

12. Online Sources

In the following section, we share the sources used for researching our paper. We also include several datasheets referenced.

12.1 Research Sources

Below is a list of resources that were used while researching the project. These resources provided valuable information that aided in the implementation of the project goals, and gave us direction in the selection of the components that are being tested to implement the project.

3.1 Existing Products
https://www.veluxusa.com/products/smart-home#Getasmarthome
https://www.energy.gov/energysaver/natural-ventilation

3.2.2 Bluetooth Connection for Sensors
https://www.aranacorp.com/fr/arduino-et-le-module-bluetooth-hc-06/

3.2.3 Temperature and Humidity Sensors
https://randomnerdtutorials.com/9-arduino-compatible-temperature-sensors-for-your-electronics-projects/
https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-ds18b20-vs-bme280-vs-bmp180/

3.2.5 Battery
http://techlib.com/reference/batteries.html
https://www.panasonicbatteryproducts.com/wp-content/uploads/2017/05/Battery-Cross-Reference-Chart-5.22.17.pdf
https://store.google.com/us/product/nest_temperature_sensor_specs?hl=en-US
https://www.batteryequivalents.com/

3.2.6 Casing for PCB
https://www.thingiverse.com/thing:3103773

5.3 Sensor Design
https://raspberrypi.stackexchange.com/questions/12161/do-i-have-to-connect-a-resistor-to-my-dht22-humidity-sensor

6.1 Wi-Fi Module Connection
https://www.instructables.com/Get-Started-With-ESP8266-Using-AT-Commands-Via-Ard/
http://woodsgood.ca/projects/wp-content/uploads/ESP8266ATCommandsSet.pdf

6.2.2 Code Setup and Logic
https://forum.arduino.cc/t/can-1-bluetooth-master-have-multiple-slaves/431055
https://www.saibatudomt.com.br/2018/01/conectando-3-dispositivos-arduino-utilizando-o-modulo-bluetooth-hc-05/
https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works
http://educ8s.tv/arduino-two-way-bluetooth-communication/

6.9 UART
http://www.farnell.com/datasheets/810077.pdf
https://www.arduino.cc/reference/en/language/functions/communication/serial/
7.4 DHT22 Sensor Testing
https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/

7.6 TFT Touch Screen Configuration Testing
http://educ8s.tv/arduino-3-5-color-tft-display-ili9481-arduino-uno-mega-tutorial/
https://learn.adafruit.com/adafruit-3-5-color-320x480-tft-touchscreen-breakout/8-bit-wiring-and-test

8.2 TFT Touch Screen GUI Implementation
https://howtomechatronics.com/tutorials/arduino/arduino-tft-lcd-touch-screen-tutorial/
https://www.youtube.com/watch?v=9Ms59ofSJIY&list=PLr46zQhJdqMulhmitZh1FqT2BqjKtLKJh&index=8

10. PCB Design
https://www.softwareradius.com/altium-designer-vs-eagle/

12.2 Datasheets

Below are the data sheets used for reference when wiring our hardwardware for testing.

ATmega48A
https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

ATmega640
https://www.mouser.com/datasheet/2/268/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-128-1315286.pdf

ATmega2560
https://www.microchip.com/en-us/product/ATmega2560

Arduino Nano Every
https://store.arduino.cc/usa/nano-every