**Oscillation 300**

****

**Final Capstone Document**

**April 16th, 2021**

**Anthony Galdi**

[**ajg425@nau.edu**](mailto:ajg425@nau.edu)

**Brendan Smith**

[**bs873@nau.edu**](mailto:bs873@nau.edu)

**Bryce Vegh**

[**brv43@nau.edu**](mailto:brv43@nau.edu)

**Luke McGowan**

[**ldm225@nau.edu**](mailto:ldm225@nau.edu)

**Table of Contents**

1. **Introduction………………………………………………………..………..………..Pg 3**
2. **Design Process.…………………………………………………..………..…………..Pg 4**
   1. **Overall design description……..………………………………………..…...Pg 4**
   2. **Functional decomposition..…………………………...…………...…..……..Pg 5**
   3. **Prototype Findings………….…………………………...………...………..Pg 10**
3. **Final Design.…………………………………………………...…………...………..Pg 14**
   1. **System Architecture………...………………………...…………...………..Pg 14**
   2. **Major Components …………………………………...…………...………..Pg 15**
4. **Results………………………………………………………………..………...…….Pg 16**
   1. **Requirements Spreadsheet…....……………………...…………...………..Pg 16**
   2. **Important Test Results …………………………….....…………...………..Pg 17**
   3. **Analysis of Results………………..…………………...…………...………..Pg 17**
5. **Conclusion of Capstone Report…………………………………..…...……………Pg 18**
   1. **Most Important Requirements………..……………...…………...………..Pg 18**
   2. **Lessons Learned……………..…………………...…………...……….…....Pg 18**
6. **User Manual………………………………………………………………………..Pg 20**
   1. **Introduction…………………………………………………………………Pg 20**
   2. **Installation…………………………………………………………………..Pg 21**
   3. **Configuration and Use……………………………………………………...Pg 24**
   4. **Maintenance…………………………………………………………………Pg 27**
   5. **Troubleshooting……………………………………………………………..Pg 28**
   6. **Conclusion…………………………………………………………………...Pg 31**
7. **Appendices.………………………………………………….……………………....Pg 32**
   1. **Appendix A………………………………………………………………….Pg 32**
   2. **Appendix B………………………………………………………………….Pg 33**
   3. **Appendix C………………………………………………………………….Pg 34**

**1. Introduction**

Our Client General Dynamics Mission Systems is a technology integrator and original equipment manufacturer who provides various mission-critical solutions to civil government, defense, intelligence and cybersecurity customers. These services help those who serve, from soldiers to sailors, to complete their missions safely and efficiently. This project specifically aids the General Dynamics Rescue 21 system, which assists the U.S Coast Guard advanced command with Search and Rescue operations (SAR).

Our project is to develop a device to measure power transmission as well as verify site weather conditions at an unmanned facility owned by our client. The goal of this is to prevent our client from having to send technicians out to the site in person as the device remotely verify the radios are transmitting. More specifically, this device will be measuring the VHF radio output power in order to determine if the radio is transmitting properly as well as be measuring the immediate local weather conditions. The benefits of such a device will provide our customer with the ability to securely view the current weather conditions of a remote location, as well as determine important information about local VHF signals regarding power and channel frequency. Secure access to such information allows for the customer to make important decisions without having to be on site themselves.

Our design incorporates an Arduino, a laptop, Hamshield Mini an additional module that works with the Arduino , attenuators, and weather sensors that are used to gather the data. The weather portion of our device will use wind, temperature/humidity, and rain sensors to gather the weather data. The power portion of our device will use a Hamshield Mini and attenuators to detect the power of the radio signal that is being transmitted from the radio. Once the weather and power data is gathered it will be sent to the laptop. Finally, a program on the laptop will be used to take in the data and record the data with a timestamp in a text file that can be accessed on the laptop at a later time.

**2. Design Process**

**2.1. Overall design description**

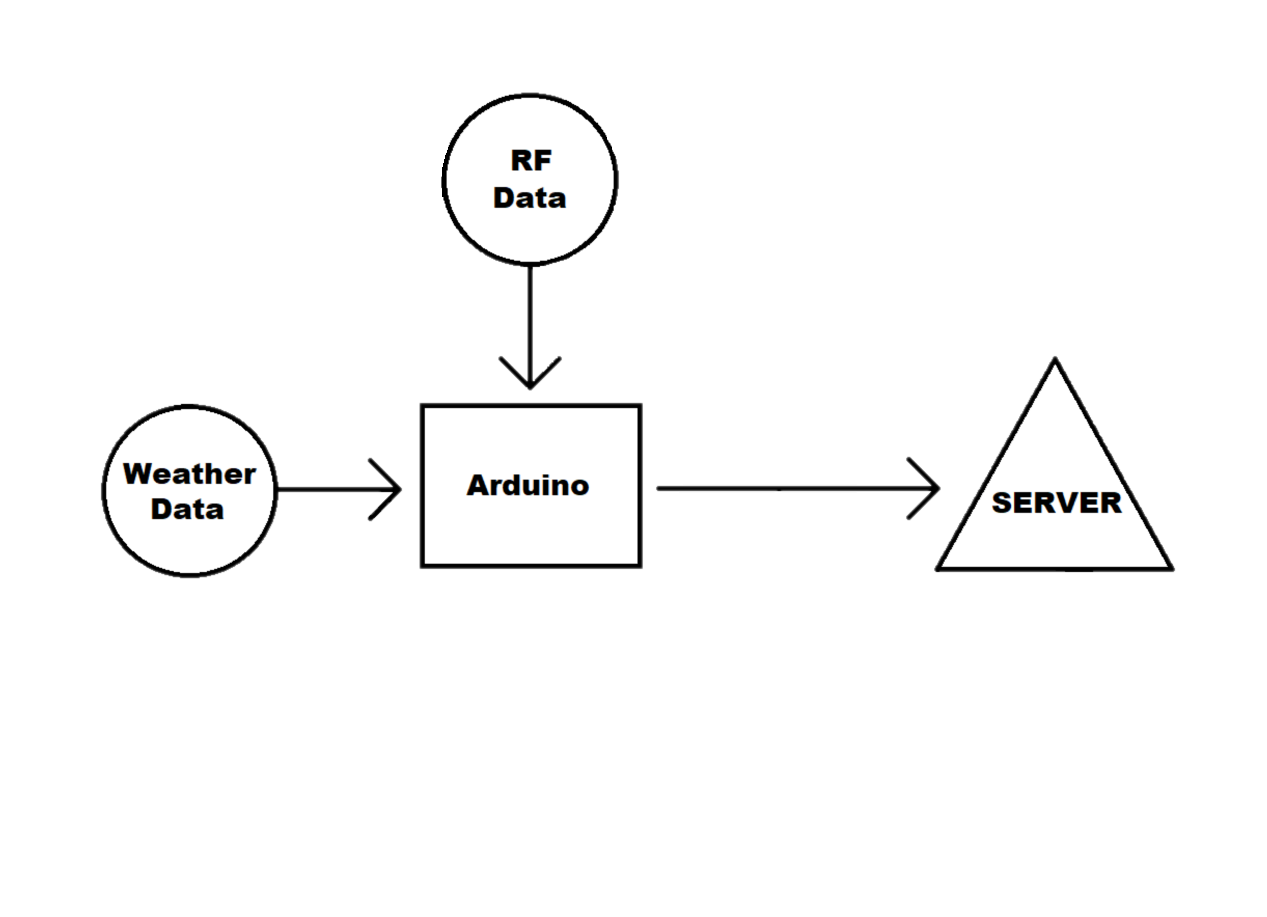
The SWAPR device is designed to measure site weather conditions and power transmission of VHF radio signals at General Dynamic’s Remote Fixed Facilities (RFF). The intention is to provide live information and site conditions to General Dynamics to avoid the need of sending a technician to the RFF if issues arise with the facility. Our design incorporates an Arduino, a laptop, a hamshield mini, attenuators, and weather sensors that are used to gather data from our weather and power station.

The SWAPR design is broken into three main modules, the weather station, the power recording station, and the Arduino processing portion which collects the data and records it to the server; Figure 1 below shows this simplified design overview. The weather station senses condition at the RFF and then relays the sensed information to the Arduino. The weather station is further broken down into three submodules which are the temperature/humidity sensor, the rain sensor and the wind anemometer. Each submodule consists of its separate sensor and circuit to maximize the modularity of the device.

The site power recorder module consists of a VHF handheld radio, a Hamshield Mini, attenuators, and SMA connection adapters. The attenuators are required to lower the input power of the radio into the Hamshield Mini to ensure that the input is within the Hamshield Mini’s limits. The Hamshield Mini then analyzes the received radio signal and relays the power transmission to the Arduino.

The third module, the Arduino recording and processing portion, is what integrates the first and second modules together, as well as, records the sensed data and sends it to the server. The power recording station and each sub-module of the weather station are connected to the Arduino, which processes the data and records it. Each of these modules and submodules are further broken down and detailed in the following section, 2.2 Function Decomposition.

**2.2. Functional Decomposition**

****

**Figure 1: Flowchart of complete SWAPR system**

The SWAPR project’s system design is broken down into individual components including the Arduino, radio frequency station and the weather station which is composed of multiple sensors.

**Design Part 1: Power Station using Software Defined Radio (SDR)**

The Power station will take a transmission from a handheld radio via coaxial cable connection, will need to be attenuated before being passed to the Hamshield Mini, where the signal will again be passed along to the Arduino:

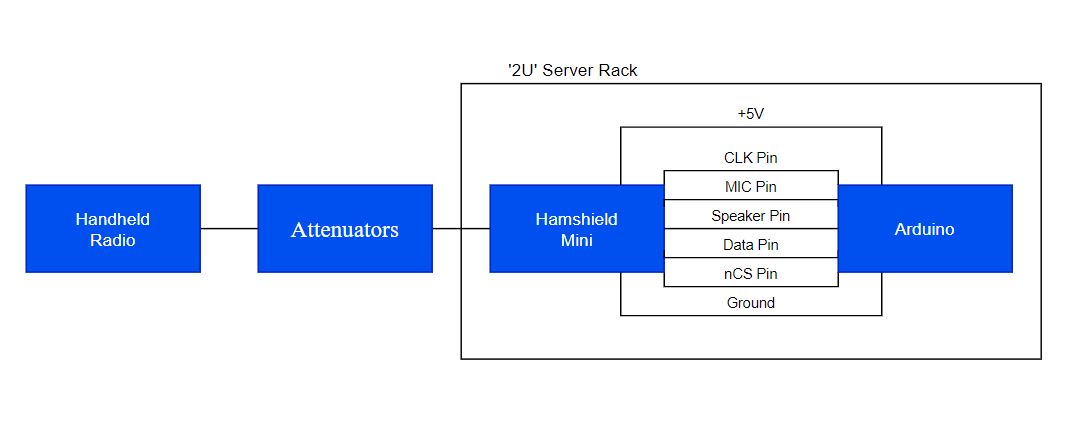
**Function:** Obtain, attenuate and transmit Radio signals

**Component:** Hamshield Mini, Attenuators

**Input:** VHF Frequency RF(140-172MHz)

**Output:** Power of Transmission in dBm

**Connections:** Coaxial

****

**Figure 2: RF Station Diagram**

**Design Part 2: Weather Station using three different weather sensors**

This section goes over the weather station which will consist of multiple sensors that each individually will detect one or two variables relating to the weather. The exact sensors that make up our weather station are:

1. Wind sensor: Davis 6410 Anemometer
2. Rain sensor: HiLetgo Rain Sensor
3. Temperature/humidity sensor: DHT11

All sensor data will be sent directly to the arduino and we also expect to power each sensor using the Arduino.

***Sensor #1 Wind Sensor:***

The wind sensor is one of the three main weather sensors required for our project’s weather station. The wind sensor’s purpose is to detect both wind speed and wind direction. The wind speed will be outputted in one mile per hour increments and the direction will be outputted in sixteen different compass rose directions. The Davis 6410 Anemometer will provide both wind speed and direction and will be powered by the Arduino, as seen in Figure 4.

**Function:** Sense wind speed & direction

**Component:** Davis 6410 Anemometer

**Input:** 5V, wind

**Outputs:**

1 mph increments for wind speed

16 different values on compass rose for wind direction

**Conversion Formulas:**

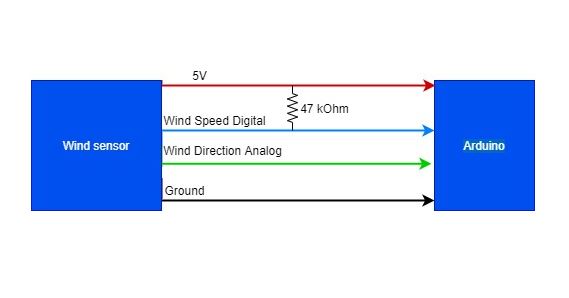
Speed (mph) = P(2.25/T)

P = Pulses/sample period

T = Sample period (sec)



**Figure 3: Image of Wind Sensor**



**Figure 4: Wind sensor diagram**

***Sensor #2 Rain Sensor:***

The rain sensor subsystem is the second of the three main weather sensors required for our project’s weather station. The rain sensor will work in unison with the temperature/humidity sensor and the wind sensor to create a complete weather station. The project requirement is to sense if rain is present or not, our team will provide additional information on how much rain is present using this sensor. Rather than simply detecting rain or no rain, we will provide a range of approximately five increments ranging from no rain to heavy rain.

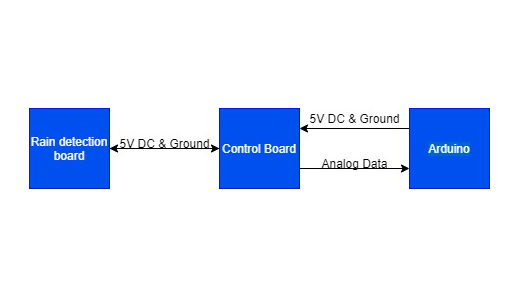
The rain sensor is made up of two components, the control board and the detection board. The detection board will be placed in a location exposed to the weather, as required to do its job. The detection board is essentially a highly resistive circuit with a resistance of 2 MOhm. When water comes in contact with the detection board, it completes the circuit and lowers the resistance. The more water that is on the board, the lower the resistance gets. When completely soaked, the board has a resistance of 100 kOhm. The lower resistance allows current flow to increase which is recognized by the control board and then converted by the code to an output on the Arduino serial monitor.

**Function:** Sense amount of rain present

**Input:** 5V, rain on board

**Output:** Analog value between 0 and 1024

Component: HiLetgo Rain Sensor



**Figure 5: Rain sensor diagram**

***Sensor #3 Temp/Humidity Sensor:***

The Temperature/ Humidity sensor is last of the three main weather sensors required for our project’s weather station. The Temperature/ Humidity sensor’s purpose is to detect both temperature and humidity. The temperature will be outputted as a numerical value and the humidity will be outputted as a numerical value also. The DHT11 Temperature and Humidity sensor will provide both temperature and humidity and will be powered by the Arduino, as seen in Figure 6.

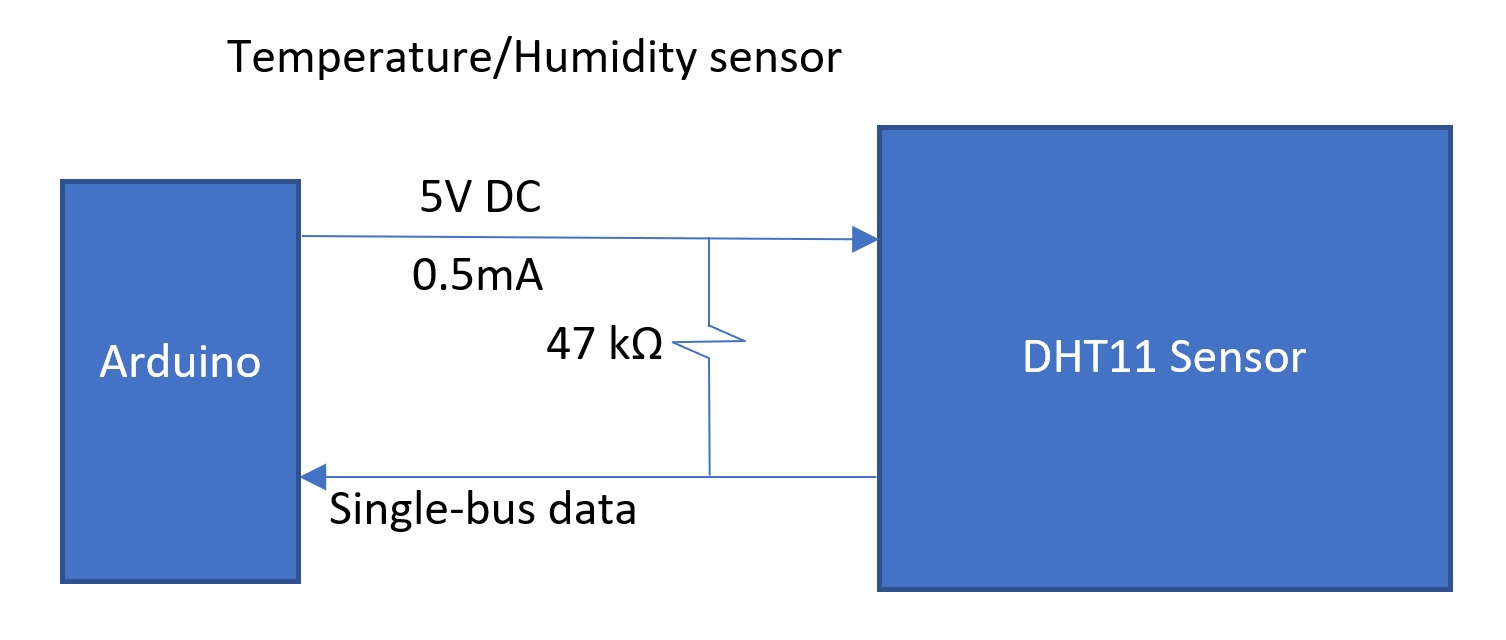
**Function:** Measure local air temp. & humidity

**Input:** 5V, surrounding air

**Output:** Temperature (°F), Humidity (%)

**Equation** °C to °F: 1.8\*°C + 32 = °F

**Component:** DHT11 Temperature and Humidity sensor



**Figure 6: Temperature/Humidity sensor diagram**

**Design Part 3: Recording and Processing Station using Arduino**

The recording station is the last stage of our design, which is where our device takes all of the data that has been recorded from the sensors/SDR and stores it on the laptop. The storing part of this station uses a python program that decodes the data and converts the data into a string type so that the program can then write the data to a .txt file on the laptop.

**Function:** Records data collected from other portions to laptop

**Components:** Arduino, Laptop, and Python program

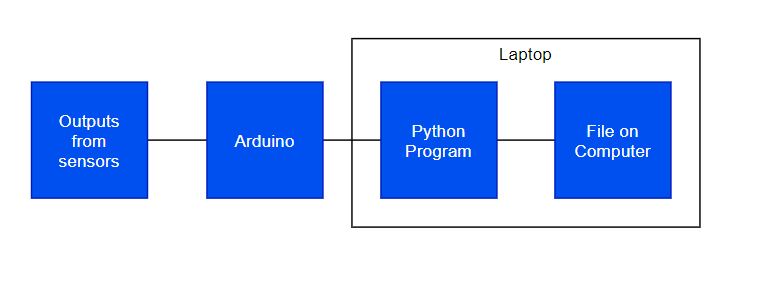
**Input:** Outputs of sensors and Hamshield Mini, COM Port

**Output:** String of text to .txt file

**Conversion:** decode function

**Baud Rate:** 9600

**Storage:** Clear text file after two weeks (~500KB)

****

**Figure 7: Recording & Processing diagram**

**2.3. Prototype findings**

During the prototyping phase, we each chose a small portion of the project to prototype. This section will include an overview of each of our individual prototyping, broken into 4 parts accordingly.

***Prototype #1 Recording and Processing Station:***

The part of the system that I am prototyping is the system that the team is using to record the data from the sensors and then save that data to the laptop on a separate file. This part of the system fits into the system architecture by being the final stage of our system that takes all the data from the sensors and software-defined radio and makes it available to be seen by the user. The team is hoping to learn from this prototype if we are able to develop software that will be able to take the data that is shown in the Arduino IDE and copy that data to a file that will be saved onto the computer.

The last part of saving the data into another file is going to be the most challenging part of the prototype because the team will have to develop software in another language that will be able to take the information from the Arduino IDE and save it on the laptop. Lastly, this prototype will reduce the total risk for the entire project because it will show the team whether it is possible to develop the software that will allow the team to save the data to the laptop or if we need to have a conversation with General Dynamics and let them know that it is not possible with the time we have to develop the needed software.

The demonstration for taking data from the serial ports of the Arduino and saving that data to the laptop in a file was successful. The program that was written for the Arduino was made simpler by just using two analog pins to take in a value and print that value out through the serial.print function. This value was created by using two potentiometers to increase or decrease the voltage from 5 volts. The other half that was figured out was the portion of the program that would be written in python. This program was used to open the same COM port as the Arduino and have the program running on the Arduino so that the python program could read the data from the COM port.

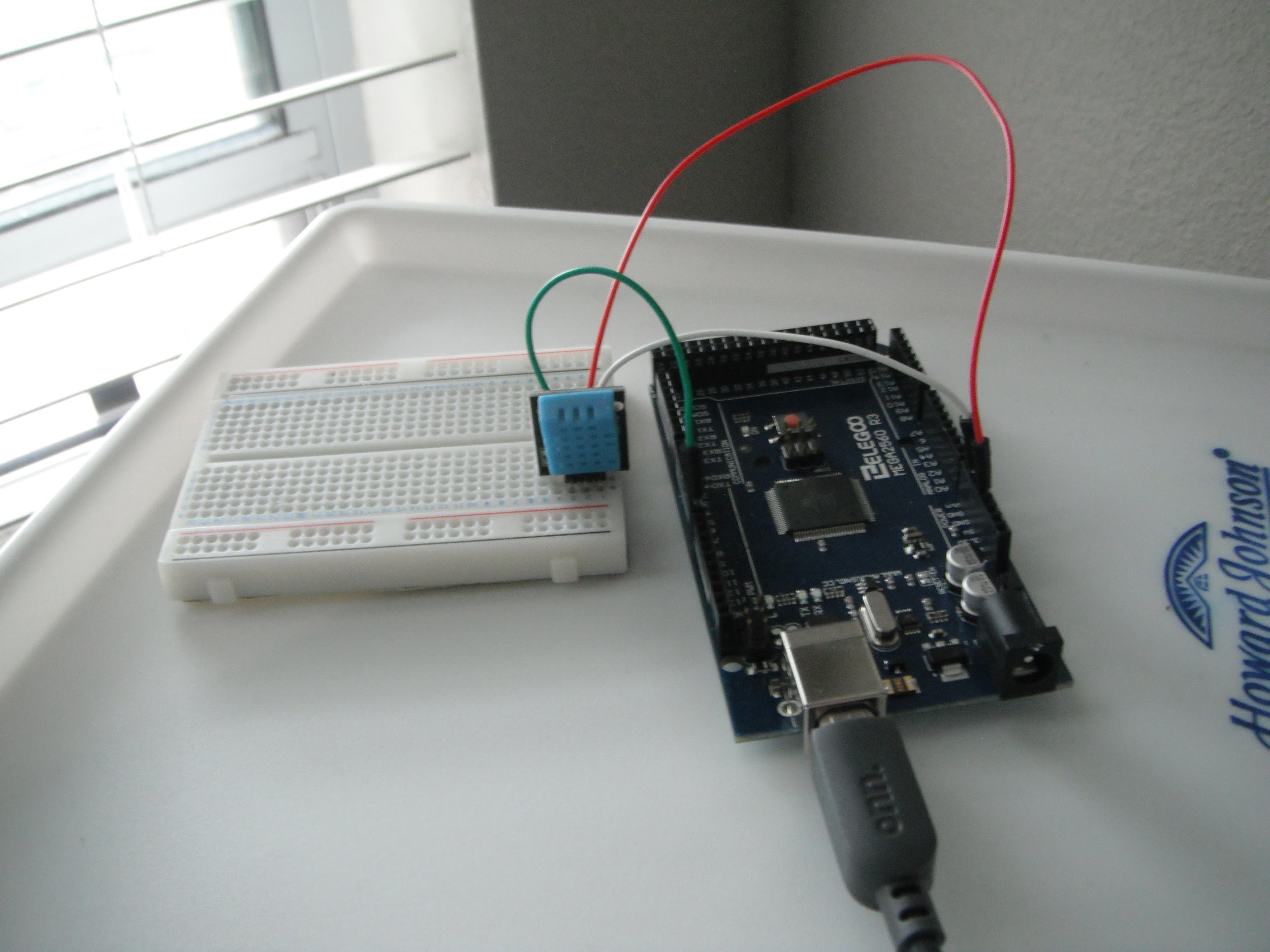
Then, use the data that the program was reading from the COM port to print the data out into a premade file by using the appending function in python. I feel that the demonstration went as I expected because a lot of time was spent on debugging the program to work with saving the data to the file on the laptop. Now that we have this more basic style that works, our team can modify the program to be able to take the data from the Arduino and store it on the laptop for later reference.

***Prototype #2 Temperature/ Humidity sensor:***

One of the sensors on the weather station that we decided to go with was the temperature and humidity sensor. This part is essential for determining the temperature of the surrounding air, which is one of the requirements from our customer. I expected to learn how to write the code for the temperature sensor that we would be using in the final project. I also thought that it would be challenging to build the model since I didn’t have access to the sensor, but it turned out that one of my teammates had a sensor I could use to build a prototype. By building this part of the sensor and learning the code, we reduced the risk of running into problems with the sensor and I can familiarize myself with the code necessary to record the temperature and humidity data.

In the demonstration, I presented a complete model that could measure the surrounding temperature and humidity and was able to keep a record of those data points. I used a DHT11 temperature and humidity sensor with an Arduino Mega to build the model. The demonstration was somewhat what I expected, the temperature sensor didn’t respond as fast as I expected and the readings seemed high so we need to make sure the sensor is calibrated correctly in the final design. We also need to consult our customer about the response time of the sensors. Aside from these small issues, I would say it was a successful demo. The humidity worked as expected and reported the humidity as a percentage.

The biggest challenge was finding the right library to use with the DHT11 to get the device to work. Since the library was not already installed, I had to find it online which took some time to find the correct one. We will have to find the correct library for our weather station to work properly. The project took about as much time as I expected, however the research beforehand took more time than I expected. If I were to do this prototype again, I would try to make the prototype more closely resemble the final product. This will change how we design the weather station based on the feedback we get from our customer.



**Figure 8: Photo of the prototype of the temperature/humidity sensor**

***Prototype #3 RF Signal Filter:***

This prototype portion of the project is the filtration of incoming Radio Frequency (RF). The RF signal received will need to be filtered as well as potentially be amplified before finally being passed to the Software Defined Radio (SDR) and this simulation prototype seeks to prototype the filtration process by using a Band pass filter.

The prototype fits within the functional decomposition or system architecture as the front end of the RF station. The entire system has only 2 inputs and one output, and the RF station is one of those two inputs. The expected lesson from doing this prototype is to learn how to receive and control the incoming high frequency RF signal before passing it along to SDR. The challenge expected was to find the proper band pass filter as well as calculate the resistance and capacitance values for that specific circuit based on the expected frequencies coming in.

Many filter options were available and had to be tested in simulation, such as a ‘sallen-key’ band pass filter circuit, cascading op amp circuits and more. This reduces the risks of the entire project because we cannot simply pass the incoming RF signal straight to the SDR as it is received since the signal will most likely be significantly too low to be picked up as well as the fact we need to prevent certain frequencies from being passed to the SDR.

The simulation was mostly successful as the cascaded OP AMP filter simulation was able to create a band pass filter that successfully filtered out the undesired low and high frequencies without attenuating the Signal for lower frequency inputs. Not entirely successful due to the fact this filter results need to work for high frequency inputs as well. Despite this, the recorded results and graph showed it still acted as a proper band pass filter.

The biggest challenge to making the prototype functional was the calculations of the resistor and capacitors as well as making this whole thing work for higher frequencies. The equations for gain and cut off frequencies need to balance in order to get the desired results. The results from this prototype influence the project by creating a basis for our front end to look like.

***Prototype #4 Rain Sensor:***

The last subsystem that was prototyped by our team was the rain sensor module for the weather station. The weather station portion of our project is made up of three sensors: wind, temperature/humidity, and rain. Each sensor is vital to the project, but the rain sensor was chosen to prototype because no team members had previous experience with or knowledge of this type of sensor.

Our team had many questions about the rain sensor, so the decision to prototype it was motivated by answering these questions. We also believe the weather station is a simpler system than the SDR, so working on the weather station early will allow us more time later on to work on the SDR as a team.

The demonstration of the rain sensor prototype went well and as expected. Unfortunately, the rain sensor part was not delivered in time for the demonstration, but we were able to find a substitute. Through research, it was found that the rain sensor is essentially a variable resistor controlled by the amount of water present on the sensor plate through research. As the amount of water on the plate increases, the sensor's resistance decreases which increases the current flow.

When completely dry, the sensor's resistance is 2 MOhms, and when completely wet is only 100 kOhms. So, a potentiometer resistor was implemented in place of the rain sensor which allowed the demonstration to be completed.

To simplify the transition when the part came in, the code was written in a way that accommodates both the potentiometer and the actual rain sensor, so that no changes in software are required. The sensor is made up of two parts, the detection board and the control board. When water touches the detection board, its resistance decreases causing an increase in current.

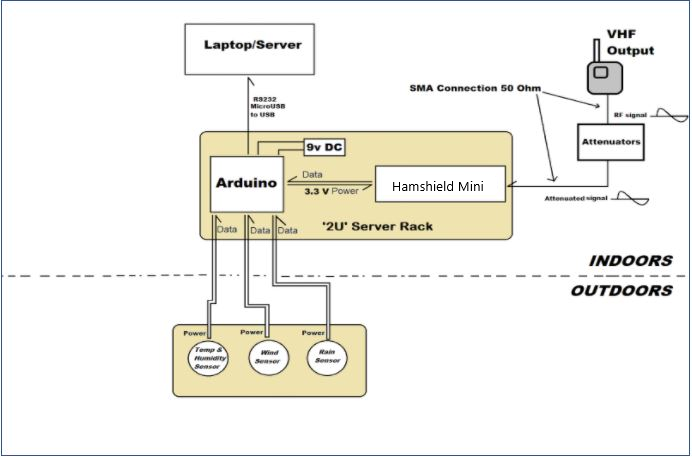
The control board takes the incoming current and maps it to a value between 0 and 1024. The Arduino code then takes that numerical value and divides it into three increments: no rain, light rain, and heavy rain; more increments may be added in the future. So, the software displays in the Arduino serial monitor either no rain, light rain, or heavy rain based upon the resistance of the potentiometer or rain sensor.

Prototyping the rain sensor taught the team a lot about both the hardware and software needed for the rain sensor to operate. The team now understands exactly how the rain sensor functions. We know that the detection board must be exposed to weather and that when water contacts the board, the circuit is completed, and current flow is increased.

We learned that the control board must be protected from weather and that its job is to convert the current flowing from the detection board into a value between 0 and 1024. We used this information to create Arduino code that converted the value sent from the control board and mapped it to a rain level increment that is outputted on the serial monitor.

**3. Final Design**

**3.1. System architecture**

****

**Figure 9: System Architecture Overview**

The entire system is centered around the Arduino, which organizes the data from the weather sensors and power station before storing it on the laptop, as previously mentioned. This system is distinctly separated into outdoor and indoor sections as reflected in the above Figure 9. The Arduino and the entire power recording station are inside of the 8 in. x 8 in. x 4 in. junction box, which will be placed inside the RFF onto a ‘2U’ server rack as requested by the client.

The weather station is located outside of the RFF in two portions. The first is the wind sensor, which is built onto its own stand, and the second is the rain and temperature/humidity sensor which is located inside of the smaller 4in x 4in x 2in junction box.

When the SWAPR is used at the RFF, the device will receive radio signals from a hardwired connection and output the power/weather information to a General Dynamics server, both of which will be next to the server rack inside of the RFF. In our implementation, we used a handheld radio hardwired using SMA to simulate the radios inside the RFF and a laptop to simulate their server. To best match real-life application, we placed both the laptop and handheld radio next to the SWAPR device when testing.

**3.2. Major components**

The SWAPR system consists of two major components that each fulfill a different function specified by our client. These components are the weather station and the power recording station. The weather station's main function is to detect and send the recorded weather conditions to the laptop. The Hamshield Mini uses a handheld radio to detect power transmissions, this component also uses attenuators to drop the radio signal down in power to record and send that to the laptop as well. Once the data is sent to the laptop, it is recorded and uploaded to a server.

**Weather Station**

The weather station is designed to detect the weather conditions at the remote facility. The weather station consists of three sensors capable of sensing five different variables. The sensors used are a rain sensor, anemometer, and humidity/temperature sensor. The different variables that we are able to measure are wind speed, wind direction, temperature, humidity, and the level of rain. The sensors are held in a weather resistant casing and the wiring follows the key provided in the user manual. Each sensor is individually wired into the arduino and shares the same 5V power. It has 25ft of wiring to allow for outdoor installation and the wind sensor is mounted to the top of the remote fixed facility.

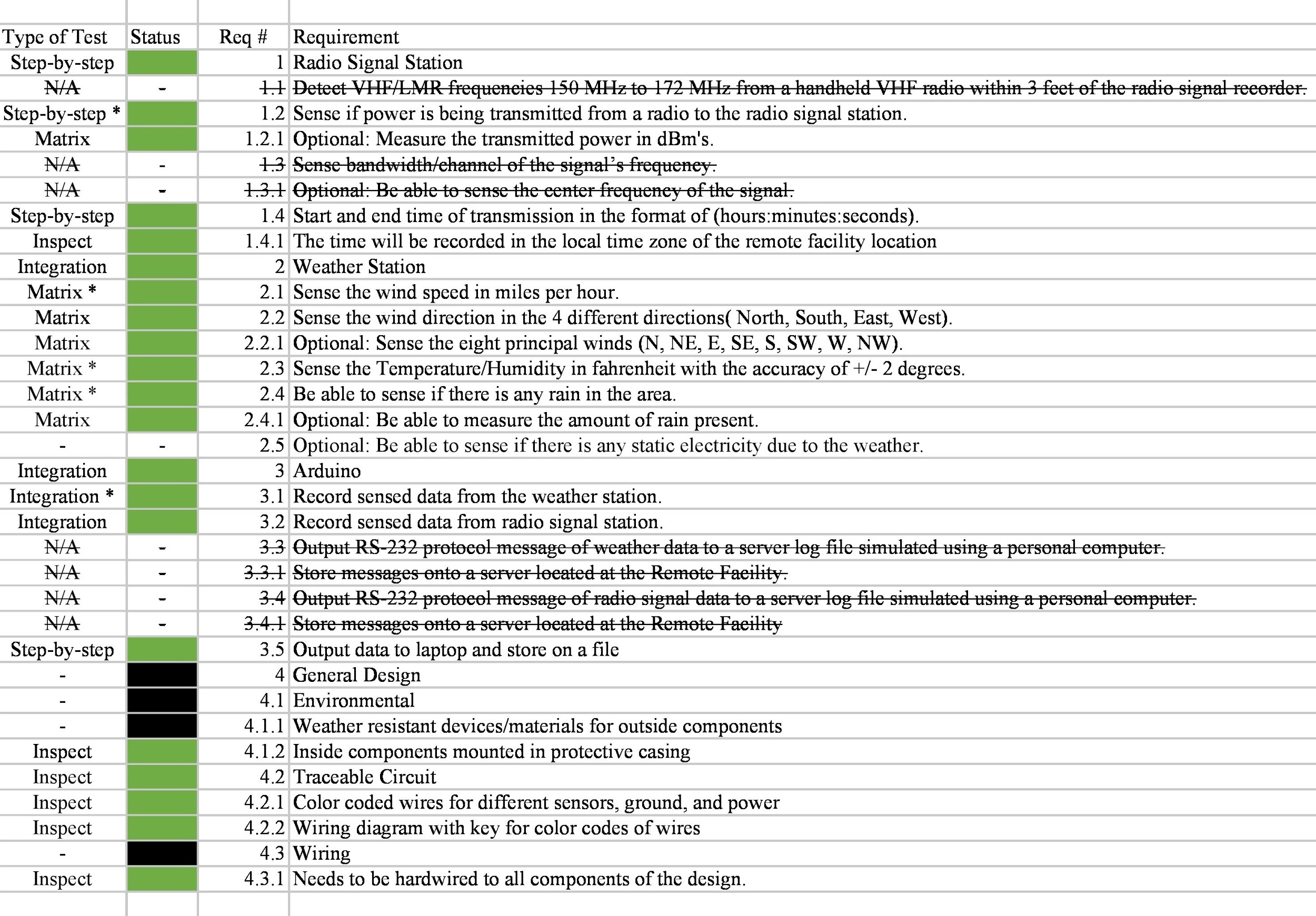
**Hamshield Mini**

The Hamshield Mini power station will take a transmission from a handheld radio via coaxial cable wired connection. Its function is to receive and filter radio transmissions. The handheld radio can read high frequency transmissions in the 140-172MHz range. We use attenuators to lower the power of the signal before it is passed through to the Hamshield Mini. The output of the Hamshield Mini is the power of the transmission in dBm.

**Laptop/Server/Arduino**

The laptop receives the data from the weather station and the Hamshield Mini and records the outputs from each component. Before the data is passed to the laptop, the data is passed through the Arduino. The Arduino is designed to be installed inside to prevent any weather damage. The laptop which is acting as the General Dynamics server uses a Python program to convert the data to a .txt file..

**4. Results**

**4.1. Requirements spreadsheet**

**Figure 10: Requirements Spreadsheet from Completed Test Workbook**

**4.2. Important test results**

Our team thoroughly tested the SWAPR device by executing six different tests to ensure that it meets all of the requirements specified by our customer. We performed three unit matrix tests, one on each of the individual weather station sensors (wind, temperature and humidity, and rain). We performed one unit step-by-step test on the Hamshield power recorder. The final two tests were integration tests. One integration test was on the completed weather station, and the other was on the weather station integrated with the signal power recorder. Each unit matrix test was completed successfully prior to performing the weather station integration test, which was also successful.

The unit step-by-step test for the power recorder was successful at all tested frequencies within the 140 to 172 Mhz ranges. The final integration test of the completed project has been executed and is successful throughout. The most important test result for our team is arguably the issue we encountered during the weather station integration test, which was actually the only major issue we encountered. After the weather station submodules were integrated, the temperature/humidity sensor began outputting invalid values.

After debugging, our team realized that an interrupt in the wind speed code was causing this error, so we altered the code so that the interrupt was both attached and detached inside of a function of the wind speed code so that it no longer had an effect on the temperature/humidity sensor. These results are likely the most important test results we received because the results lead us to find a simple error in the code that was causing an entire submodule of our weather station to malfunction.

**4.3. Analysis of Results**

Our project performed very well under the test conditions that we subjected the device to. Each individual unit matrix test on the weather station sensors was entirely successful. Each sensor’s code was debugged and any hardware issues were resolved during prototyping and early white box testing, so that when it was time for the actual testing, everything went smoothly and as expected. The weather station integration test initially failed because the temperature/humidity sensor was outputting invalid values.

Through debugging, we found that the attached interrupts from the wind sensor code were causing this issue. We resolved the problem by attaching the interrupt inside of the findSpeed function for the wind sensor and then detaching it at the end of the function. After this fix, the integrated weather station performed successfully and as expected. The unit step-by-step test for the Hamshield did behave as expected and was able to detect power transmission as desired. The final integration test of the entire system was successful and performed as expected.

All weather station sensors were being sensed and recorded correctly and the Hamshield recorded power in dBm’s for the tested frequencies. A few of our client’s requirements have changed over the course of this project and these changes are shown using strikethrough in Figure 10. Our project meets or exceeds all of the current requirements from our client and also meets a few of the optional requirements.

**5. Conclusion**

**5.1. Most important requirements and their results**

1. Record sensed data from the weather station

The requirements for the weather station were to sense if rain is present, humidity, temperature, wind speed, and wind direction. The weather station successfully senses all required weather conditions, meeting this requirement. The weather station exceeds requirements by meeting some optional requirements such as sensing the amount of rain present in increments and sensing wind direction up to the eight principle winds.

1. Record sensed data from power station

The requirement for the power station was to sense if power is being transmitted from a radio to the station and an optional requirement of measuring the transmitted power in dBm’s. The power station meets both of these conditions, exceeding the requirements.

1. Output data to the laptop and store on a file

The requirement for outputting the data to a file on a laptop was met by sending the sensed data as JSON variables using a Python program to store them on a text file.

1. Weather resistant devices/materials for outside components

This requirement was met. Weather resistant junction boxes and strain relief connectors, found in Appendix A, were used to protect the vulnerable components of the device from harsh weather conditions.

1. Needs to be hardwired to all components of the design

This requirement was met. All components of our device are hardwired. No wireless communication, including Bluetooth, Wi-Fi, or wireless radio signals were used.

**5.2. Lessons Learned**

Overall, our project went well but of course it was not perfect. The one major challenge we ran into during this project was with our selection of a software defined radio (SDR) for the power recording portion of the project. We had chosen and purchased parts for a SDR and then built and prototyped that SDR, but later discovered that our SDR selection could not be used because it required a laptop or computer for processing, which was not allowed for this project. So, our team had to quickly find a new solution to record power for this project. Due to this late change, we had to adapt our work schedule and stray from the Gantt chart some to ensure the project would be completed on time.

So, a lesson learned is to develop a thorough understanding of all requirements and to ensure that our selected parts meet those requirements and constraints. In the future, our team will be sure to spend more time on the selection of parts and will verify with our client and supervisor if we are unsure about any component. Other than that, our team only experienced minor issues during testing that were quickly resolved through debugging. For example, when the weather station was integrated, the temperature/humidity sensor began to malfunction. We used the test-fix-test procedure to find and resolve this issue. The test-fix-test was also used during white box testing of the rain sensor to calibrate the sensor and its output. Our team also used regression testing often, usually before integrating a new module or when returning to work after an extended break.

The regression test allowed us to verify that certain components were working so that if something goes wrong, debugging would be quicker and easier. A few requirements were changed throughout the project because as we researched and worked on the project, we developed a better understanding of what was actually required. For example, the struck-through requirements in Figure 10 like requirement 3.3.1, *Store messages onto a server located at the Remote Facility,* was changed because we learned that we would be storing the data to a text file on our personal laptop rather than a server.

Our team did not come across any requirements that were too difficult to test, we believe that our requirements met AVUT (abstract, verifiable, unambiguous, testable) which made our requirements easy to test. Our team learned that testing is more about the preparation and previous work than it is the actual test. We learned that a majority of the work must be completed prior to completing the test such as writing strong and AVUT requirements, prototyping the device, building the device, and designing a quality test that will actually tell if the device meets requirements.

If all of this work is completed correctly before the test, then the test itself is very simple and easy. Our project’s performance could have been improved by meeting the remaining optional requirements listed. These requirements included sensing how much static electricity is in the air and a higher degree of accuracy for the amount of rain present in the area.

**6. User Manual**

**6.1 Introduction**

We are pleased that you have chosen Oscillation 300 for your business needs. There is a strong need for a Site Weather and Power Recorder (SWAPR) device, as evidenced by the specific problem description given by General Dynamics describing the need for their Remote Fixed Facility locations to be able to determine if transmissions are being sent within the 140-172MHz VHF band as well as knowing the immediate surrounding weather conditions as the weather, most notably heavy rainfall, can severely impact the power transmission data collection by causing RF interference.

The system itself must be hardwired, as wireless transmission of data would be considered a security vulnerability. Additionally, the system also needs to be fitted to be able to withstand the varied weather conditions so as not to have any excess water or heat damaging the components and thus rendering it useless. The final and major function of the system is that it needs to log the previously mentioned data onto a laptop acting as a substitute for their server and must be RS-232 protocol compliant, which we have successfully done.

Our team’s design provides for you here a powerful system that is able to cover all of the previous stated requirements such as measuring the severity of rainfall, humidity, temperature, wind speed, wind direction, and the power of a radio transmission in the VHF radio frequency range that has been custom designed to meet your needs. Some of the key highlights include:

1. Accurately measure temperature, humidity, wind speed, wind direction, and if rain is present anywhere you want.
2. Monitor incoming radio signal and their transmitted power
3. View all of this data in real time as the values are recorded to a text file located on your personal computer.

Due to varied nature of the project, each subsystem has unique features and thus vary greatly in their data collection and communication with the Arduino microcontroller that is processing the data before handing it over to be logged onto the server:

The weather subsystem consists of three distinct sensors that combine to help paint a picture of the environmental weather conditions, which are the temperature and humidity measurement sensor, the wind speed and direction sensor, and finally the moisture sensor. Although each sensor is unique, they successfully combine to get the entire job of indicating the weather conditions as expected. All sensors power and data are run to and from the Arduino Microcontroller via approx 25’ jump-wiring cables.

The Power Recording subsystem includes a Handheld VHF radio to act as the simulated RF source and is designed to connect to the Hamshield Mini via a coaxial connection. Our design includes attenuators between the Handheld Radio and the Hamshield Mini so as to significantly step down the power being received by the Hamshield Mini to not damage it while still being able to detect transmission within the desired frequency range. The Hamshield Mini itself is specifically designed to be compatible with Arduino microcontroller directly for ease of integration of the data collection while simultaneously bypassing any use of wireless transmission that other device choices often utilize. Finally, the power recording station has been coded by our team in order to test specific set frequencies falling within the previously mentioned 140-172 Mhz band on regular intervals, pausing briefly only when the microcontroller needs to check the weather data. This code is able to be modified by the customer in order to fit their needs if so desired.

The laptop portion of the project that acts as the server where all the data collected and managed by the microcontroller is sent to to be logged and later viewed by the client as needed, while being RS-232 protocol compliant. This is done by utilizing JSON variables within a Python script and will be logged every 5 minutes.

Finally, the entire system itself was built with the ability to be protected by the varied weather conditions it is expected to perform in, and we achieved this by using polycarbonate framing that is used to safely house our sensitive components. The microcontroller itself is housed with the power station for convenience while the weather stations components have their own shared housing and have the 25’ cable reach so as to be able to be placed far out from the other components in order to reach wherever the client prefers the weather data to be collected. Lastly, the laptop itself which is acting as the server our client would be logging data is not protected, as it as well the actual server is expected to be indoors and not exposed to any outside weather conditions.

As a side note, the wiring itself has extra features such as being color coded for ease of maintenance or repair as well as terminal blocks being used to prevent any wires from easily being dislodged. The setup of all these components as well as their code implementations will be discussed further in this manual.

Many of the systems design choices were decided upon by our team based on not only the function of the device but more notably by its costs, efficiency, and compatibility for integration within the entirety of the system itself.

The purpose of this user manual is to help you, the client, successfully use and maintain the SWAPR product for your business application going forward. Our goal is to make sure that you are able to benefit from our product for many years to come and we have made sure that the system is flexible enough to be suited to the clients need by allowing for modification of our code as well as ease of altering the hardware components as desired by maintaining organized and color coded connections.

**6.2 Installation**

**Software**

There are two types of software that are needed for the SWAPR device to function correctly, these softwares are Arduino IDE and Python. Starting off with setting up the Arduino IDE software, this software is available at the following link <https://www.arduino.cc/en/software>. At this link, you will choose the download option that fits the computer that is being used to program the Arduino with the Arduino code that our team has developed. When you choose the correct download option it will download a .exe file that will allow you to download the Arduino software onto the computer that is being used. Click this .exe file, this will start the download process for the Arduino IDE and follow through the installation windows keeping all of the default settings selected in the installation windows.

Once the installation process is complete, there are now libraries that need to be installed for the Arduino code that our team has developed to function correctly. The libraries that need to be installed are **DHT sensor library**, **Arduino\_JSON**, and **HamShield**. To install these libraries, you will need to open up the Arduino software. Once opened go to the upper left corner of the Arduino software and select the tools drop down menu, in this tools drop down menu select Manage Libraries. When you click Manage Libraries, it will open up the Library Manager window and this is where you will search for the three libraries that were previously listed. You are able to search for libraries by copying and pasting in the library name for this user manual into the white box that is in the upper left corner of the Library Manager window.

Once the correct library has been found, you will hover over the library that matches the library name being searched for and an install button will show up in the lower right corner. Click this button, this button will install the library and you will repeat this process for each of the other libraries that are needed. Now that these libraries have been installed, the installation of the Arduino IDE has been completed and this software is ready to be configured for use. But, this will be done later in **Section 6.3 Configuration and Use** and now we will go to installing Python to be used for the SWAPR device.

To install Python please follow the following link <https://www.python.org/downloads/>. This link will take you to the website that will allow you to find the latest version of Python that will work for the computer that is being used to run the Python program our team has developed. On this website page, you will find a yellow button that says Download Python and this is the button that you will click to download the .exe file for Python. This .exe file that will be used to download Python, click on this .exe file and it will bring up the installation window for Python. Go through the installation windows without changing any of the default settings for the installation process.

Once Python has been installed, there are three libraries that need to be installed which are **Pyserial**, **jsonlib**, and **times** libraries for the Python program to function correctly. To install these libraries, you will need to open the command prompt on your computer and enter in specific commands to install the libraries. You are able to open the command prompt by going to the search bar in the lower right corner of your computer desktop. Once you have opened up the search bar, enter in Command Prompt and click on the app that says Command Prompt. This will open up a command prompt window that you will enter in specific commands that will be given to you shortly. To download the first library **Pyserial**, you will type in the following pip install pyserial into the command prompt and hit enter to start the installation of this library. This will take about thirty to forty-five seconds, once this finishes the **Pyserial** has been successfully installed. Now to install the next library **jsonlib**, you will type in the following **pip install jsonlib** into the command prompt and hit enter. This will take about thirty to forty-five seconds, once this finishes the **jsonlib** has been successfully installed.

Finally to install the last library **times**, you will type in the following **pip install times** into the command prompt and hit enter. This will take about thirty to forty-five seconds, once this finishes the **times** library has been successfully installed. After this has been completed, all the software that is needed for the SWAPR device has been installed with the correct libraries that are needed to function correctly. The next section will go over how to correctly install the hardware portion of the SWAPR device.

**Hardware**

For the hardware installation process, please reference the bill of materials in Appendix A to purchase all of the hardware that is needed to build the SWAPR device. After these parts have been ordered and have arrived, the process of putting all of the hardware together can be started. The building process can be started by taking the first 8in x 8in x 4in junction box and using a drill to drill holes into the 3 sides of the junction box to be able to fix strain relief connectors to these holes. For these holes, there will be one hole that will have a ⅜” strain relief connector so a ⅜” drill bit will be used to drill this hole. The last two holes will be for ¼” strain relief connectors, a ¼” drill bit will be used to drill these last holes in the junction box.

Once that these holes have been made, you can go ahead and fit the ⅜” strain relief connector to the ⅜” hole, and the two ¼” strain relief connectors to the ¼” holes. This junction box is done for now and the focus will be moved to the smaller 4 in x 4in x 2 in junction box. For this smaller junction box, you will drill a ¼” hole on one of the sides that does not have the screw mount piece that sticks out. Once this hole has been drilled, you will take the last ¼” strain relief connector and screw it into the hole that was just made.

That junction box is now finished and can now have the rain sensor and temperature/humidity sensor installed into the smaller junction box. To install these sensors, you will need to drill a ⅛” hole in the side of the junction box that is opposite of the strain relief connector. This hole will be used to run the wires of the rain sensor through to the bread board that will be put into the junction box. Once this hole is drilled, feed the wires of the rain sensor through and use silicone to fill in the hole where the wires are being fed through this will allow the rain sensor to be glued in place and to seal the hole from allowing water into the junction box.

Now you will place the small breadboard inside of the small junction box, with this small breadboard you will place in the temperature/humidity sensor and wire the rain sensor into the breadboard. You will need to follow the circuit schematic that is provided in Appendix B. After this has been done, the small junction box is complete and should have twenty-five feet of wire that is coming out of the strain relief connector. We can now focus on the last junction box, the first step is to place in the terminal block and pass through the four wires from the small junction box through one of the ¼” strain relief connectors.

Once those wires have been passed through, screw each wire into their own terminal on the terminal block. There will be one terminal for both ground and power that is shared for all of the sensors and devices that are used. Next, place in the large breadboard and place the Arduino Nano onto this board and wire the terminal block wires from the rain and temperature/humidity sensors to the Arduino Nano using the circuit schematic in Appendix B. The wind sensor uses a RJ11 connector board, this will be the next step to wire this board into the Arduino, use the circuit schematic in Appendix B to complete this task. Once this has been done, you can feed the wire from the wind sensor through the ⅜” strain relief connector and plug the RJ45 cable from the wind sensor wire into the RJ11 connector board, this will complete the installation of the wind sensor. The final step of the installation process is to connect in the radio to the Hamshield Mini and the Hamshield Mini to the Arduino Nano. This can be done by first taking the antenna off of the radio and attaching the first SMA cable to the radio first followed by all of the attenuators.

When attaching the attenuators you want to make sure that they are tightly screwed on because if they are not, it can cause issues with the readings for the power of a transmission. Then, you will attach the second SMA cable that will go from the attenuators to the Hamshield Mini, this SMA cable will be fed through the ⅜” strain relief connector because the Hamshield Mini will be stored inside of the large junction box. Next, you will place the Hamshield Mini onto the opposite side of the large bread board and attach the SMA cable to the Hamshield Mini. Now, you will follow the circuit schematic in Appendix B to wire the Hamshield Mini into the Arduino Nano. The final step is to feed a micro USB cable through the last ¼” strain relief connector to connect the Arduino Nano to the laptop to be able to power and program the Arduino Nano.

**6.3 Configuration and Use**

**Updating the Arduino Code**

Once all of the installation has been completed, there is now software that needs to be configured for the device to be able to be used correctly. The first step is to set up the Arduino software for sensing the weather and power stations data, you will want to move towards the bottom of the source code for the Arduino. In the bottom portion of the Arduino code, there are four findPower functions that are used to find the power of the four radio frequencies that are going to be scanned to see if there is a radio transmission being transmitted. In each of these findPower functions there are variables named **freg** which is short for frequency, this is the variable that is used to set the frequency that the Hamshield Mini is using to find the power of the transmission in dBm. Once you have found the **four freg** variables in the Arduino code, you will need to configure these variables to the four radio frequencies that you are wanting to scan with the Hamshield Mini for radio transmissions. These variables need to be put into the Arduino code as their equivalent radio frequency in kHz instead of MHz. For example, if the radio frequency that is trying to be scanned by the Hamshield Mini is 150 MHz; you will need to input the radio frequency into the Arduino code as the value of 150000 which is 150 kHz. This is due to the fact that the Hamshield Mini and Arduino are not able handle the full power of the radio output , so attenuators are used to bring the radio output power down from 4 watts to a lower power of 1 watt which is the maximum power that the Hamshield Mini is able to handle.

**Compiling Arduino Code**

Now that the desired four frequencies to be scanned have been changed, you will need to save the Arduino code by going to the left corner of the Arduino software and selecting the file drop down bar and selecting save in the file drop down menu. This will save the changes that you have made to the Arduino code and allow for the code to be compiled to check for any errors that may have occurred in the code while editing the Arduino code.

To compile the Arduino code, in the upper left corner there is a circular button that has a check mark. Once you find this button, you will click on it and this will start the compiling of the Arduino code. If for some reason there are errors that do come up when the Arduino code is compiled, please reference **Section 6.5 Troubleshooting** of this user manual to help correct the issue and get the Arduino code to compile without errors.

**Programing the Arduino Code**

Now that the Arduino has been compiled and no errors have occurred in the code, it is time to program the Arduino to run the written code. The first thing that must be done is to make sure the correct Arduino is selected and a COM port is selected. This is done by going to the upper left corner of the Arduino software and selecting the drop down menu called tools. Once you have selected the tools drop down menu, you will go down to the board selection option in the tools drop down menu and make sure that the Arduino Nano board is selected.

Now that the correct board is selected, you will now make sure that a COM port is selected for the Arduino Nano. You will do this by going to the tools drop down menu again and selecting the port option in the tool drop down menu. Then you will select the COM port that has Arduino Nano in parentheses next to it. Once this is done, you will go to the upper left corner of the Arduino software and look for a circular button that has an arrow that is pointing to the right. This is the button that is used to program the written code to the Arduino. Click this programming button, this will start the process of programming the Arduino and you will see a green status bar in the lower right corner and on the bottom left corner there will be text that will go from compiling sketch to uploading to done uploading. Once the Arduino software says in the bottom left corner done uploading the Arduino has been programmed and will start sensing and sending data to the COM port. If this doesn’t happen please reference **Section 6.5 Troubleshooting** of this user manual to help correct the issue.

**Verifying Data is Being Sent**

The final setup to configuring the Arduino to use is to open up the COM port and verify that data is being sent to the COM port. You can find the window to open up the COM port by going to the Arduino software. Once you are in the Arduino software, if you look to the upper right corner and look for a square button that has a magnifying glass with a dotted line going through the magnifying glass. Click this button and it will open up a new window that will show data that is being outputted to the COM port. Now, you will watch the data and you should see data for humidity, temperature, rain, wind speed, wind direction, powerF1, powerF2, powerF3, and powerF4. If there are numeric values for humidity, temperature, wind speed, powerF1, powerF2, powerF3, and powerF4 that means the data is being correctly outputted by the device. If there are null values, please reference **Section 6.5 Troubleshooting** of this user manual.

If there is data that says any of the following values “No Rain, Dry Conditions”, “Light Rain”, “Moderate Rain”, or “Heavy Rain” for the rain data then the device is outputting correctly. If there are any other values than the previous listed, please reference **Section 6.5 Troubleshooting** of this user manual. If there is data that says any of the following values “N”, “NE”, “E”, “SE”, “S”, “SW”, “W”, or “NW” for the wind direction data then the device is outputting correctly. If there are any other values than the previous listed, please reference **Section 6.5 Troubleshooting** of this user manual. Finally all of the data has been verified to be working correctly, you can now close the COM port window that you opened to be able to see the data that was being sent out and move to the next section below.

**Python Program**

Finally, you will open up the python program that was written to record the data from the Arduino software to a text file on the laptop. Once this python program is opened in the python editing software, there is only one value that needs to be checked to make sure that the python program will work correctly. This value is the line that starts with serPort or line 5 of the python program. To be able to change this line you will need to reference the Arduino software to find the correct COM port that the Arduino is using, open up the Arduino software. Now that the Arduino software is open, to find the correct COM port you will go to the upper left corner of the Arduino software and find the tools drop down menu.

Once you select the tools drop down menu, you will need to go down to the port option in the tools drop down menu. What this port option in the tools drop down menu will tell you is what COM port number that the Arduino is using, this is the number that you will need to remember to be able to change the COM port value in the Python program. Now that you have this value, go to the Python program and go to the line that starts with serPort or line 5. The value you will be changing is the text that is green and you will only change the numeric value at the end of the green text to the number that you referenced from the Arduino software. For example, if the Arduino is using COM port number 1 you will change the line from serPort = serial.Serial('COM7') to serPort = serial.Serial('COM1'). Once this has been changed, you will need to save the changes that were made in the Python program. This can be done by going to the upper left corner of the Python program window and selecting the file drop down menu.

Then, going down to the save option in the file drop down menu to save the Python program. Finally now that the program has been saved, it is time to run the Python program. You can run the Python program by going to the upper left corner and selecting the run drop down menu and selecting the run module option in the run drop down menu. This will run the Python program as it was intended, if you receive any error message when you run the Python program please reference **Section 6.5 Troubleshooting** to correct this issue.

**Device is Ready**

Now that each of these steps have been completed and any errors that may have occurred have been corrected, the device is ready to be left to do the work it was designed to do.

**6.4 Maintenance**

**Wind Sensor Maintenance**

The wind sensor will need to be checked and maintained due to how it has an anemometer that is used to find the weed speed and wind directions. The anemometer needs to be able to spin freely to be able to take accurate measurements for both the wind speed and wind directions. Which means that every month it is recommended that the data for the wind speed and wind direction should be checked to verify that it has changed and there is no data that seems off. The data should also be checked versus wind speed and wind direction data that is available on the internet for areas that are close to the remote fixed facilities. Doing this verification monthly will allow you to make sure that the data that is being recorded is valid and useful data.

If a problem in the data is found, a technician will need to be sent out to first try to clean and remove debris from the anemometer to see if this will correct the problem. If the cleaning does not correct the issue, then the whole sensor has been compromised and will need to be replaced. But, if no problem is fine then no action is needed. The last bit of maintenance that is recommended is that every six months a technician should be sent out to the remote fixed facility to check the wind sensor and clean the anemometer from any debris that may be inside of it. This will allow the sensor to stay clean and function as it was designed.

**Reuploading of the Arduino Software**

Our team recommends that the Arduino software be reuploaded to the Arduino every six months in case any type of corruption that may happen due to the weather conditions at the remote fixed facilities. This can be done by recompiling the Arduino code and program that code to the Arduino. This can be done by going to the upper left corner of the Arduino software, in this location there is a circular button that has a check mark. Once you find this button, you will click on it and this will start the compiling of the Arduino code. This has compiled the Arduino code, and now it is ready to be programmed to the Arduino. To program the Arduino you will go to the upper left corner of the Arduino software and look for a circular button that has an arrow that is pointing to the right. This is the button that is used to program the written code to the Arduino. Click this programming button, this will start the process of programming the Arduino and you will see a green status bar in the lower right corner and on the bottom left corner there will be text that will go from compiling sketch to uploading to done uploading. Once the Arduino software says in the bottom left corner done uploading the Arduino has been programmed. After this has been completed, the reuploading of the Arduino has been completed but you will need to start the Python program back up. Reference the next section to start the Python program back up so that the SWAPR device can be put back into service.

**Starting up Python Again after Reupload of Arduino Software**

To start up the Python program again you will follow the following steps. You will have to start by opening up the python program. In the python editing software, there is one value that needs to be changed. This line is the line that starts with serPort or line 5 of the python program. To be able to change this line you will need to reference the Arduino software to find the correct COM port that the Arduino is using, if you are unsure on how to do this reference **Section 6.3 Configure and use**. Now that you have this value for the COM port, go to the Python program and go to the line that starts with serPort or line 5 and changes this COM port value. You will save the Python program, if you are unsure on how to do this reference **Section 6.3 Configure and use**. Now that the program has been saved, it is time to run the Python program. You can run the Python program by going to the upper left corner and selecting the run drop down menu and selecting the run module option in the run drop down menu.

**Cleaning up the Server from Day Text Files**

Due to how the Python program was developed, it creates a .txt file for everyday that it is running. Which because of that the SWAPR device is going to be creating a mass amount of .txt files that will need to be sorted and erased once the files are no longer needed. This maintenance will depend on how long General Dynamics wants files to be saved, but our team recommends that you keep one month of data on the server to be able to see weather conditions and power transmissions trends. With this being said, we recommend that every three months a technician opens the server up and erases the oldest files until there is only one month worth of .txt files. This will help from slowing down the server and make it easier to find .txt files that are needed for reference.

**6.5 Troubleshooting**

**Compilation Errors**

If the error “expected ';' before 'radio'” occurs during compilation after you have changed the value for the freq variable, it is because the semicolon was deleted by accident when the value of the variable was changed. This can be corrected by adding a semicolon after the freg variable line that is highlighted in red. Once this has been corrected, you can go back to **Section 6.3 Configuration and Use** and the subsection **Compiling Arduino code**.

**Arduino Programing Errors**

If an error occurs when you click on the program but the first thing that needs to be checked is that the Arduino is plugged into the computer. If this was the issue please return back to **Section 6.3 Configuration and Use** to step up the Arduino for use. If this does not correct the issue, the next step is to check that the correct Arduino board has been selected in the Arduino software. This is done by going to the upper left corner of the Arduino software and selecting the drop down menu called tools. Once you have selected the tools drop down menu, you will go down to the board selection option in the tools drop down menu and make sure that the Arduino Nano board is selected. If this was the issue please return back to **Section 6.3 Configuration and Use** to step up the Arduino for use. If not, the last step is to check to make sure that the correct COM port is selected in the Arduino software. To do this you will do this by going to the tools drop down menu again and selecting the port option in the tool drop down menu. Then you will select the COM port that has Arduino Nano in parentheses next to it. Now all of these things have been checked and verified you can go back to **Section 6.3 Configuration and Use** to set up the device for use.

**Null Values for Data**

If null is being outputted by one of the weather sensors that normally outputs a numerical value, the first step of troubleshooting that you will need to take is to reference the full circuit schematic in appendix B and the wire color table in appendix C for the specific wire color that matches the sensor you are troubleshooting. You will want to reference this schematic and wire color table because most likely a wire has come loose or broken at some point in the connection. First using the listed reference above, you will want to check the wiring at the Arduino and double check that the correct wire color is going into the correct pin of the Arduino. If this is wrong, correct this issue and go back to **Section 6.3 Configuration and Use** to step up the Arduino again. But if this is not the issue, you will want to track the wire all the way back to the sensor that is being troubleshot and while you are doing this you want to check for any type of breaks or damage to the wire.

If there is any kind of damage to the wire, it is recommended that the wire be completely replaced due to the wire being compromised. Once the full wire has been replaced, you can go back to **Section 6.3 Configuration and Use** to step up the Arduino for use. If there was no damage or breaks in the wire, the last step of troubleshooting is to completely replace the sensor with a brand new sensor. To find where a new sensor can be bought from please reference the bill of materials in appendix A to find the name, part number, and where the sensor can be bought from. Once the new sensor has arrived, using the schematic from early in the troubleshooting wire the new sensor into the circuit of the device. Now that the sensor has been replaced, you can go back to **Section 6.3 Configuration and Use** to set up the device for use.

**Incorrect Text Output for Data**

If the incorrect or error text is being outputted by one of the weather sensors that normally outputs a text value, the first step of troubleshooting that you will need to take is to reference the full circuit schematic in appendix B and the wire color table in appendix C for the specific wire color that matches the sensor you are troubleshooting. You will want to reference this schematic and wire color table because most likely a wire has come loose or broken at some point in the connection.

First using the listed reference above, you will want to check the wiring at the Arduino and double check that the correct wire color is going into the correct pin of the Arduino. If this is wrong, correct this issue and go back to **Section 6.3 Configuration and Use** to step up the Arduino again. But if this is not the issue, you will want to track the wire all the way back to the sensor that is being troubleshot and while you are doing this you want to check for any type of breaks or damage to the wire.

If there is any kind of damage to the wire, it is recommended that the wire be completely replaced due to the wire being compromised. Once the full wire has been replaced, you can go back to **Section 6.3 Configuration and Use** to step up the Arduino for use. If there was no damage or breaks in the wire, the last step of troubleshooting is to completely replace the sensor with a brand new sensor.

To find where a new sensor can be bought from please reference the bill of materials in appendix A to find the name, part number, and where the sensor can be bought from. Once the new sensor has arrived, using the schematic from early in the troubleshooting wire the new sensor into the circuit of the device. Now that the sensor has been replaced, you can go back to **Section 6.3 Configuration and Use** to set up the device for use.

**Python Run Errors**

If you receive an error when you select the run module button in the run drop down menu, the first set of troubleshooting is to check that the COM port text has been changed in the Python program. This can be done by referencing the Arduino software to find the correct COM port that the Arduino is using. To find the correct COM port you will go to the upper left corner of the Arduino software and find the tools drop down menu. Once you select the tools drop down menu, you will need to go down to the port option in the tools drop down menu. What this port option in the tools drop down menu will tell you is what COM port number that the Arduino is using, this is the number that you will need to check for in the COM port text value in the python program.

If these do not match, this was the issue that was causing the error and you need to change the Python program to match with the COM port that the Arduino is using. Once this has been corrected, you can go back to **Section 6.3 Configuration and Use** to finish setting up the Python program to use. If this is not the issue, the next step is to check what Arduino software windows you have open. If you still have the COM port window open for the Arduino software it will not allow the Python program to run. The COM port window is the window that was opened up when you clicked the square button in the upper right corner of the Arduino software that had the magnifying glass with a dotted line through it. Click the exit button in the upper right corner of the COM port window to close this window out, now this is down please return back to **Section 6.3 Configuration and Use** to finish setting up the Python program for use.

**6.6 Conclusion**

Our team, Oscillation 300, would like to thank General Dynamics for giving us the opportunity to work on this project and design this device for General Dynamics and the Coast Guard. Each member of this team has learned valuable skills and knowledge from working on this project for the last eight months that we will be able to use in our professional career. We would also like to extend our best wishes to the deployment of this device at your remote fixed facilities and the continuation of this project to the next capstone team.

**7. Appendices**

**Appendix A**

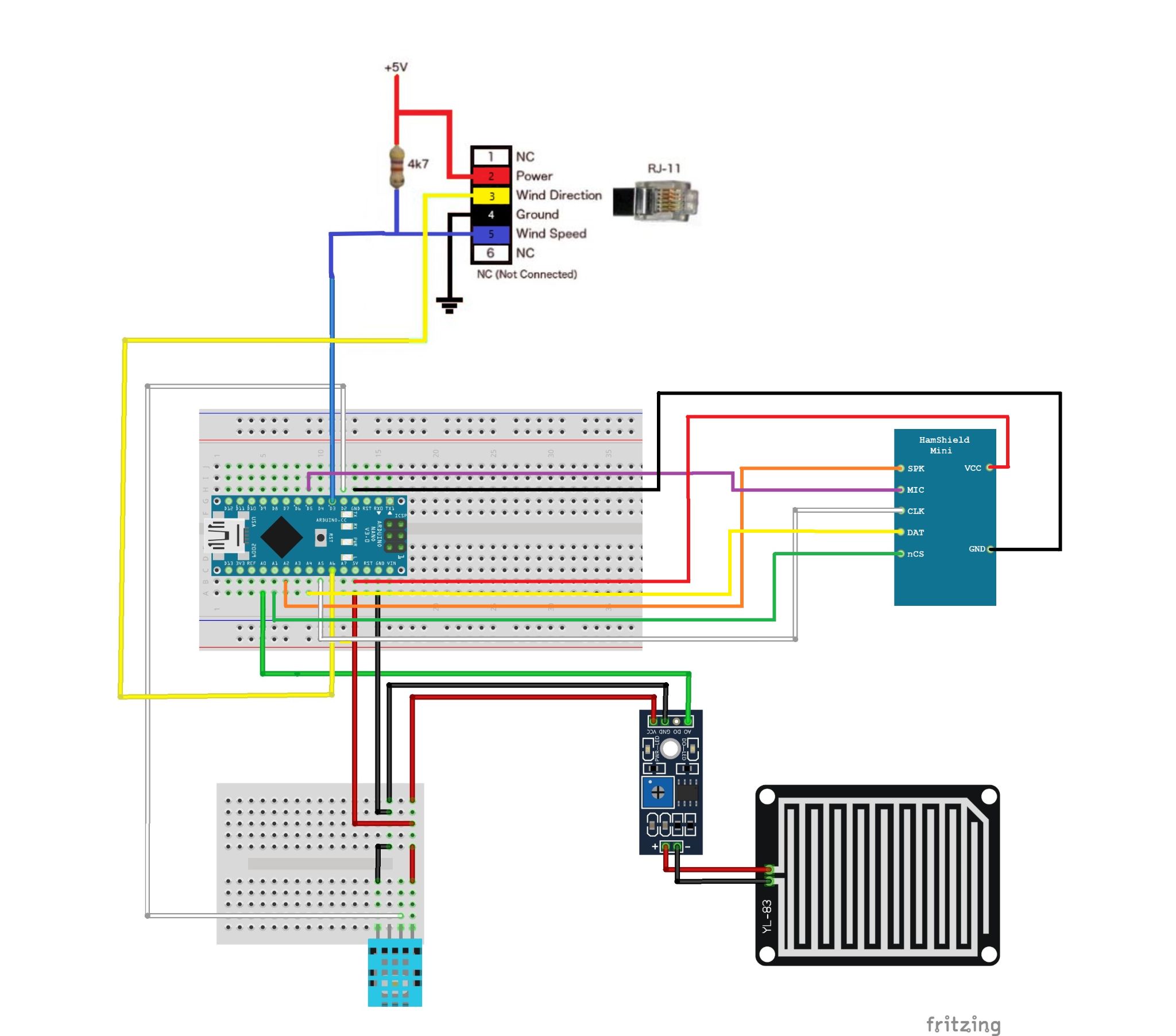
The below table is a table that represents the bill of materials to build this SWAPR device. The bill of materials lists the name of the part that was used, the quantity of parts needed, the price of the part per unit, and the website that the part is able to be purchased at.

**Table 1: Bill of Materials**

|  |  |  |  |
| --- | --- | --- | --- |
| Part Name | Quantity | Price per Unit | Website to Order |
| Arduino Nano | 1 | $20.70 | [store.arduino.cc/](https://store.arduino.cc/usa/arduino-nano) |
| Rain Sensor | 1 | $5.99 | [www.amazon.com/gp](https://www.amazon.com/gp/product/B01DK29K28/ref=ox_sc_act_title_2?smid=A30QSGOJR8LMXA&amp;psc=1) |
| Temp/Hum Sensor | 1 | $5.99 | [www.amazon.com/Temp-Hum](https://www.amazon.com/Comidox-Temperature-Humidity-Raspberry-Included/dp/B07KD6VZCM/ref=sr_1_7?dchild=1&amp;keywords=dht11&amp;qid=1603823266&amp;sr=8-7) |
| Wind Sensor | 1 | $121.61 | [www.amazon.com/Wind](https://www.amazon.com/gp/product/B004GK9MFO/ref=ox_sc_act_title_2?smid=A3SQ9YFE6CSCS0&psc=1) |
| HamShield Mini | 1 | $50.00 | [inductivetwig.com/hamshield](https://inductivetwig.com/collections/hamshield-and-accessories/products/hamshield-mini) |
| Attenuator 40dB | 1 | $12.58 | [www.amazon.com/attenuator](https://www.amazon.com/angwang-Attenuator-Attenuators-Frequency-Connectors/dp/B08H1R62JQ/ref=sr_1_15?dchild=1&keywords=40db%2Battenuator%2Bsma&qid=1614804943&sr=8-15&th=1) |
| Attenuator Pack of 6 (42dB) | 1 | $39.95 | [www.amazon.com/Attenuator](https://www.amazon.com/Nooelec-SMA-Attenuator-Kit-Line/dp/B07YYMT65T/ref=sr_1_19?dchild=1&keywords=attenuator+sma&qid=1614730873&sr=8-19) |
| 4in x 4in x 2in Junction Box | 1 | $8.98 | [www.homedepot.com/4x4x2-Jun](https://www.homedepot.com/p/4-in-x-4-in-x-2-in-Junction-Box-R5133705/202043434) |
| 8in x 8in x 4in Junction Box | 1 | $29.95 | [www.homedepot.com/8x8x4-Jun](https://www.homedepot.com/p/8-in-x-8-in-x-4-in-Junction-Box-R5133712/202043419) |
| ⅜” Strain Relief Connector | 1 | $1.99 | [www.amazon.com/Strain-Relief](https://www.amazon.com/Cable-Gland-Strain-Relief-Locknut/dp/B00HBJ6FNE) |
| ¼” Strain Relief Connector | 3 | $1.79 | [www.amazon.com/Strain-Relief](https://www.amazon.com/Cable-Gland-Strain-Relief-Locknut/dp/B00HBIYSSY) |
| 8-Circuit Terminal Block | 1 | $7.86 | [www.homedepot.com/Term-Block](https://www.homedepot.com/p/Gardner-Bender-22-10-AWG-8-Circuit-Terminal-Block-1-Pack-GTB-408/202522560) |
| Small breadboard | 1 | $4.00 | [www.adafruit.com/BreadBoard](https://www.adafruit.com/product/65?gclid=Cj0KCQjwpdqDBhCSARIsAEUJ0hP0ABdAIo4YTukYUVgYvFVJoXB-YKbWtPtidUfMrbmCCbhXWU7Sqm4aAj4-EALw_wcB) |
| Large Breadboard | 1 | $5.95 | [www.adafruit.com/BreadBoard](https://www.adafruit.com/product/239?gclid=Cj0KCQjwpdqDBhCSARIsAEUJ0hMKzkdPQB2c5NblO5ZVOfXttlo-Laskpa5BbPtIQeGaR1mBi_fd9MkaAqMVEALw_wcB) |
| RJ11 Connector Board | 1 | $3.65 | [www.amazon.com/Connector](https://www.amazon.com/gp/product/B01CWQZ1JO/ref=ppx_yo_dt_b_asin_title_o05_s00?ie=UTF8&psc=1) |
| Radio | 1 | $62.89 | [baofengtech.com/bf-f8hp](https://baofengtech.com/product/bf-f8hp/) |
| SMA Cable | 1 | $8.27 | [www.amazon.com/SMA-Cable](https://www.amazon.com/MWRF-Source-Male-RG316-Cable/dp/B074Y9RN6J/ref=sr_1_2_sspa?dchild=1&keywords=sma&qid=1618421742&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyTFBNVkhSWDFKM0hWJmVuY3J5cHRlZElkPUEwNjg1NDEyMVhERVI2OENCUTlVQyZlbmNyeXB0ZWRBZElkPUEwMDk4OTUyMVZQUU1MNDQ2RTU5MCZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=) |
| Total Cost | | $392.15 | |

**Appendix B**

The below image is the full schematic of the SWAPR device that our team has developed for General Dynamics. The image should be referenced below when building the SWAPR device initially and also when a technician is troubleshooting an issue with the device.



**Figure 11: Full Schematic of the SWAPR Device**

**Appendix C**

The following table is used to tell what color of wire is used for which sensor. This color coding allows for a technician to easily troubleshoot a sensor due to each sensor having different color wire that is attached to it. Meaning when they are looking for a specific sensor, they can reference this and easily find the wire that is associated with that specific sensor.

**Table 2: Wire Color Chart**

|  |  |
| --- | --- |
| Color of wire | Wire Destination |
| Red | +5V |
| Black | Ground |
| Green | Rain Output (Pin A0) |
| White | Temperature/Humidity Output (Pin D2) |
| Yellow | Wind Direction (Pin A6) |
| Blue | Wind Speed (Pin D3) |
| White | HAMShield CLK (CLK pin to A5) |
| Purple | Hamshield Mini (MIC pin to Pin D5) |
| Orange | Hamshield Speaker (Speaker pin to pin A2) |
| Yellow | Hamshield Data (Data pin to pin A4) |
| Green | Hamshield nCS (nCS pin to pin A1) |