# Microclimate Weather Station Array

Matthew Diehl, Anthony Testa, Hemel Debnath, Bilal Mahmood

Climate Change affects every region on the planet. We are living in times where the sea level is rising, temperatures are rising throughout the globe, and other weather-related anomalies are happening everywhere. But how can we track these effects locally across New York State? New York State is full of diverse microclimates such as massive mountain ranges, lakes of all sizes, marshlands, fields, forests, and miles of concrete cities. The impact climate change has on each of these microclimates may differ ([1]). Our mission is to create a weather monitoring system that can measure relevant humidity, temperature, and precipitation data on NY microclimates. Data will then be transmitted to the NYSDEC, providing information necessary to be able to combat potential negative effects of climate change in these regions.

**Problem Statement**

The New York State Department of Environmental Conservation needs an inexpensive and accurate weather monitoring system that is adopted across many homes and businesses so that they can better understand the impact climate change has on microclimates throughout New York State.

**Problem Introduction**

Climate change is a real problem that every person on the planet must face. Steps need to be taken to reduce negative human impact on the environment, but to effectively come up with solutions data is required. To effectively acquire data, we (a team of 3 Electrical and Computer engineering students) are looking to create a relatively cheap weather monitoring station. This station can be integrated into a larger array of stations to better understand human impact on the microclimates of New York.

***Inspiration***

**Cited Source: Raspberry Pi "Build Your Own Weather Station" Weather Monitoring System**
***https://projects.raspberrypi.org/en/projects/build-your-own-weather-station/0***

**Context**

One example of previous work examined is the raspberry pi foundations guide on building out your own weather station. Their weather station uses basic materials that can be acquired easily and has many useful sensors. For example, they have humidity and temperature sensors, air quality, wind speed and gust, wind direction, and rainfall sensors ([2]).

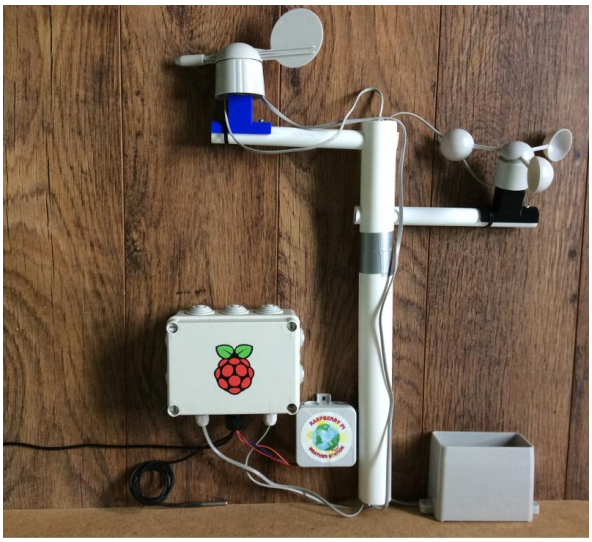


**Figure: Completed Model of Raspberry Pi "Build Your Own" Weather Monitoring System Project**

**Key Takeaways and Lessons Learned**

Although the system has good insight on the sensors needed and the interfaces and coding required to get such a weather station working the raspberry pi foundation system does not appear to be a durable system with a considerable amount of work to be done when it comes to storm proofing the device ([2]).

**Cited Source: Real Time Weather Monitoring System using IoT (link) Authors: Puja Sharma, Shiva Prakash**
**Context**

An important example of prior work includes the weather monitoring system referred to in the report "Real Time Weather Monitoring System using IoT" by Puja Sharma and Shiva Prakash. Throughout the report, descriptions are given on implementation and planning of a weather monitoring system where DHT11, BMP180, and Raindrop

sensors will measure temperature and pressure, barometric pressure, and precipitation levels, respectively. Information collected for weather using either of the three sensors is uploaded to a webpage over WiFi using a NodeMCU microcontroller, and the type of information uploaded can be toggled using 3 Modes specific to each sensor ([3]).

**Key Takeaways and Lessons Learned**

The proposed subsystem nicely integrates the data acquisition and communication aspects of a weather monitoring system using sensors to measure data, modes to select what data to measure, and a microcontroller that uploads the data to a public webpage over one's IP address ([3]). Nevertheless, the subsystem lacks core functionalities needed for a data acquisition unit as no mention is made of whether the unit is placed in a durable enclosure, which is important to consider if the unit is placed outside and exposed to potential weather damage ([3]). In terms of its communications features, the system's reliance on using WiFi to upload data to a public webpage over one's IP address does not consider potential issues such as internet outages and crashes/expected website maintenance on the server where weather is displayed which could affect proper upload of data to the webpage ([3]).

**Cited Source: How I Made a Fully Functional Arduino Weather Station by Francisco Claria**

**Context**

In "How I Made a Fully Functional Arduino Weather Station", author and engineer Francisco Claria attempts to solve an issue of receiving data on wind from kitesurfing areas in Argentina with a low-cost Arduino Uno based weather monitoring system. ([4]) The design of the system is based on the use of an Arduino Uno (and later an Arduino Mega due to limited storage capabilities of the Arduino Uno) storing data on temperature, pressure, wind speed, wind direction, wind gust, and rain through sensors to an SD card and server ([4]).

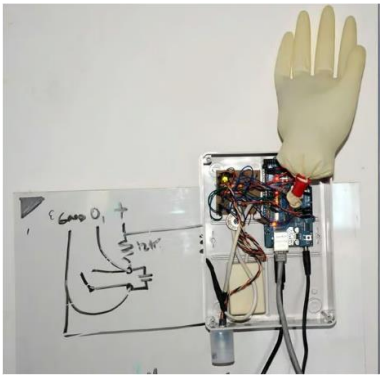


**Figure 2: Completed Model of Francisco Claria's Arduino Uno Weather Monitoring System**

**Key Takeaways and Lessons Learned**

The system implemented by Francisco is well thought out and considers various limitations which were either directly addressed when making the system or never addressed but referred to in his report on how and why the system was made ([4]). It is noted that the created system will face several software issues including the bouncing effect which leads to some interrupts in receiving information from a sensor due to physical opening and closing of contacts between the sensors and the microcontroller ([4]). Since the system was designed to have zero maintenance, it also included a "watchdog" feature in the Arduino code which would check if the entire system worked correctly at regular time intervals but would reset the entire Arduino board after checking the status of the system. Besides this, the system was initially implemented using an Arduino Uno which has limited RAM storage (2 KB of storage) and limited storage for compiled software (around 32 KB of storage) ([4]).

### *System Requirements*

**Use Case Diagram**

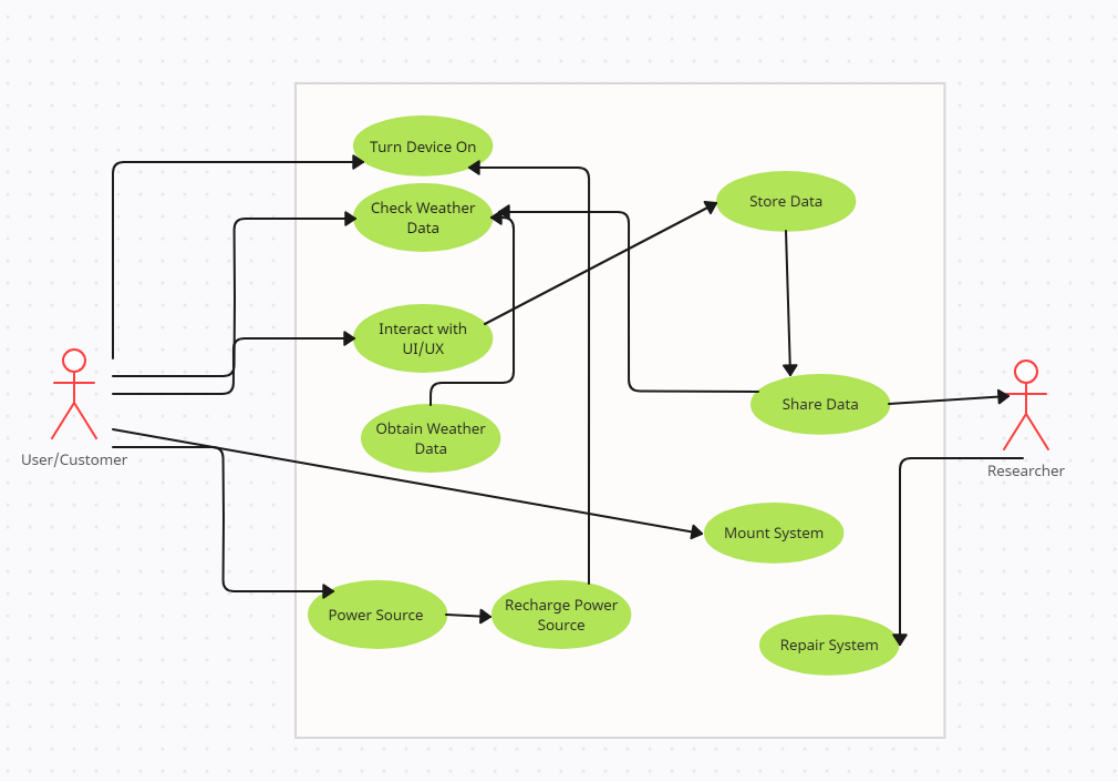



**Figure 3: Use Case Diagram of a Complete Weather Monitoring System employed at User Locations by NYSDEC**

**Functional Requirements**

- **Ability to Collect Weather Data:** The system should be able to collect data on ambient temperature, humidity, and precipitation as a means of measuring the impact of climate change on microclimates in New

York. According to the Environmental Protection Agency, climate change in the U.S. has been defined by a rise in average temperature and 0.2 inch per decade increase in precipitation across the 48 states on the U.S. mainland ([5]).

- **Accuracy of Collected Data:** All data collected from the weather monitoring system should be accurate and undergo numerical and graphical error analysis to ensure accuracy and precision. This can be done through performance of percent error calculations and graphical comparisons of hourly temperature, precipitation, and humidity data collected from the center to the same data available through the National Weather Service and local weather stations. In this case, "accurate" and "precise" data will include any data collected through the system which has a maximum percent error of 5 – 10% from corresponding data collected for that hour through a local or national weather station. For example, temperature data collected at 12:00 pm on a certain day is only "accurate" and "precise" if the data has a 5 – 10% percent error difference from temperature data collected by a local weather station at 12:00 pm.

- **Data Transmission and Communication Abilities:** As the New York State Department of Environmental Conservation (NYSDEC) intends for the system to be able to collect and share relevant weather data, it is important to ensure that the data can be shared with NYSDEC. As a result, the system should be able to convert the collected data to a machine-readable format that allows for easy collection of data by the system, transmission of the data to NYSDEC, and easy analysis of data by analysts at NYSDEC.

- **Speed and Efficiency:** The system should be able to perform data acquisition, processing, and transmission simultaneously without significant delays in time. This means that the system should be able to multitask doing various operations like taking in data, processing the data, and sending the data to NYSDEC without facing significant performance or efficiency issues. An acceptable latency time for acquisition and processing of the data starting from data collection to data storage in an Excel file should be no more than the maximum latency time of the Arduino Uno (i.e, around 5.1 milliseconds).

**Non-Functional Requirements**

- **Safety:** The system should operate and perform data acquisition, processing, and transmission in a way that doesn't pose threat of injury or death to end users of the system.

- **User Friendliness:** The system should be easy to use for end users who have the system installed in their home and/or business. To promote its widespread adoption, the system should have a nice display which can be changed to provide information on different features of the weather that are most important to end users.
- **Durability to Weather Extremities:** Since the weather monitoring system will be implemented throughout New York State, it is important to consider the range of extreme weather conditions such as blizzards, snowstorms, and thunderstorms that could pose a threat to any outdoor components of the system. Hence, it is important to consider that the system's design should incorporate some element that allows it to be protected from damage by severe weather.
- **Scalability:** Design and implementation of the weather monitoring system should ideally be done in a way that allows the system to be mass produced by NYSDEC at a very large scale. Any components used to create the system and procedures required to build the system should be considered such that NYSDEC may have the ability to reproduce the system in large quantities.

**Design Constraints**

- **Cost of System:** Design of the weather monitoring system must be based around a maximum budget of $150 - $200 contributed to $50 amounts by teams of 3 - 4 individuals responsible for creating, testing, and implementing the weather monitoring system. Limiting the budget that must be spent on the weather monitoring system to $50 significantly reduces the amount and type of tools available at each team's disposal, meaning comparison of design tradeoffs for the different tools that could be used for designing the system should be done knowing that only $50 can be spent per person in designing the project.

- **Time Needed to Build System:** The process of fully designing, implementing, and testing the system should occur within 3 months (or 1 college semester) and must be completed by April 30th, 2024. Confining the amount of time available to create the system to 3 months (from February 2024 to April 30th, 2024), in turn, limits the choice of various tools/designs for the system to only those which are easily implemented.

*System Design*

**Design Overview & Justification**

**Decision Matrix for Possible Microcontroller Choices**

| Criteria | Arduino UNO | ESP32-C3 | MSP430 |
|---|---|---|---|
| Power Consumption | 2 | 4 | 4 |
| Library access | 5 | 3 | 2 |
| Range of Functionality | 5 | 2 | 2 |
| Performance | 3 | 4 | 2 |
| **Total Score** | **15** | **13** | **10** |

**Decision Matrix for Possible Weatherproof Enclosures**

| Criteria | Weatherproof Chassis | Plastic Box (3D printed) |
|---|---|---|
| Durability | 5 | 2 |
| Ability to Integrate and Fit Hardware Components | 5 | 3 |
| Ability to Mount on Surfaces | 5 | 3 |
| **Total Score** | 15 | 8 |

Our design works by acquiring temperature, humidity, and precipitation data from via Adafruit HTS221 and soldered electronics, respectively. This data is then processed through an Arduino Uno microcontroller, organized into an Excel file, and stored on a local machine.

**Format of Stored Data**

Storage of acquired data can be done either on a Microsoft Excel file, or a removable SD card contained within the subsystem. However, as our subsystem is focused purely on data acquisition, the format of storing data should be chosen for ease of use by engineers responsible for designing data transmission/communication systems and data analysts at NYSDEC. In this regard, Microsoft Excel was chosen as the primary form of data storage due to the high portability of Excel Files compared to SD cards (which must be manually obtained to acquire data). Additionally, data contained in Excel files is easily readable to humans, which is an optimal design choice especially if the collected data is to be analyzed manually and not with algorithms.

**Choice of Microcontroller**

Through our research of microcontrollers, we landed on three possible microcontroller options. ESP32-C3, Arduino UNO, and MSP430. ESP32-C3 was considered due to its single core high processing power between 80Mhz – 160Mhz and low power consumption at 5μA when in deep sleep mode. We didn't implement it into our final design due to its lack of library support to integrate our sensors with it in a timely manner. The MSP430 provides a

low power consumption option with its low power mode idling at 1μA and between 40 μA/MHz and 400 μA/MHz when processing (this depends on the chipset). It has the same issue as the ESP32-C3 as it lacks the library support to implement the sensors and would take longer to implement, limiting our time for other areas of the project.

**Chosen Microcontroller**

We went with the Arduino UNO as our microcontroller of choice. It has the necessary library support to integrate our sensors and the Arduino UNO has a much wider range of functionalities. It performs slower when compared to the ESP23-C3 and consumes more power than both other options. Our justification for this is that our project doesn't require the extra processing power provided by the ESP32-C3. Having the extra low power consumption from the MSP430 would be beneficial in that respect but with put without the library support, establishing a working design in our limited time frame would prove more difficult. Nevertheless, the Arduino Uno microcontroller is well equipped.

**Enclosures**


**Figure 4: QILIPSU Weatherproof Enclosure with Clear Cover & Steel Latches**

We needed an enclosure to hold all our electronics with the added aspect of being weatherproof. There were two options for this, a 3D-printed box in which we would weatherproof, or a bow we can purchase that fills all our requirements. We went with a junction box made for holding electronics. This box was chosen due to its hinge operated gasketed clear door making the enclosure waterproof, durable hard plastic housing, and easy mounting system capable of being mounded anywhere. The junction bow also has a plate on the inside specifically designed to mount electronics and can be oriented in any direction. This choice was more expensive than 3D-printing a box but gaining all these properties for a relatively small portion of the budget outweighed the other option.

**Black Box Diagram**

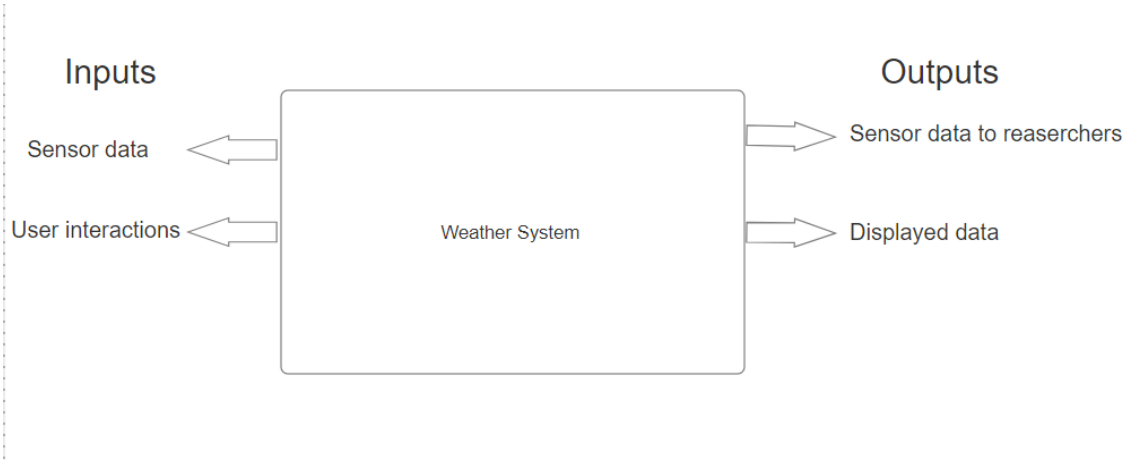



**Figure 5: Black Box Diagram Description of Weather Monitoring System**

**Logical Design**

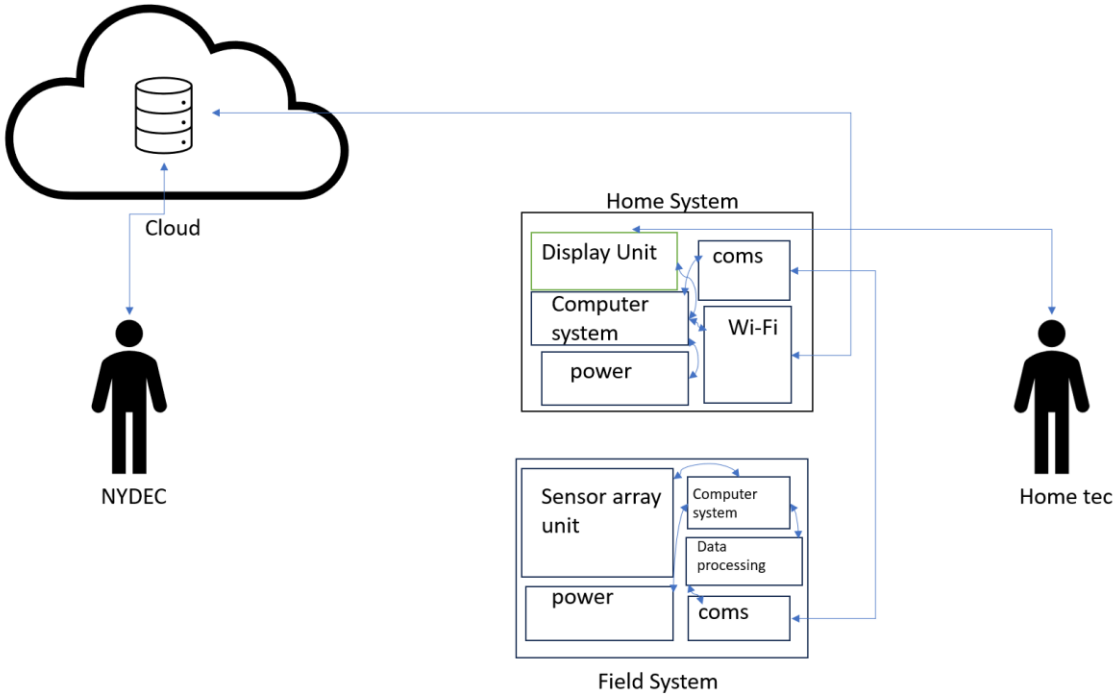



**Figure 6: System Architecture Diagram of Weather Monitoring System**

Figure 3 outlines the design of a weather monitoring system deployed by the NYSDEC to individual homes for collecting weather data in terms of three main components: an outdoor field system, indoor home system, and cloud for data storage and transmission.

The Field System is a subsystem that is to be deployed outside and used for collecting key weather data including data on temperature, humidity, and precipitation through an array of sensors. Collected data is then sent to a computing system (such as a microcontroller) which will process the collected data.

The data will then be sent to an indoor home system and stored in a computing system, which is responsible for displaying any relevant information from the data (including temperature and humidity levels) to a user in whose home the system has been deployed. Using a Wi-Fi connection, data that was collected and processed between the field and home system will then be sent to a cloud which manages transmission of the data from the system to analysts at NYSDEC. In case the system malfunctions or faces technical issues, a home technician may be deployed by NYSDEC to diagnose and solve errors.

**Wireframe Model**

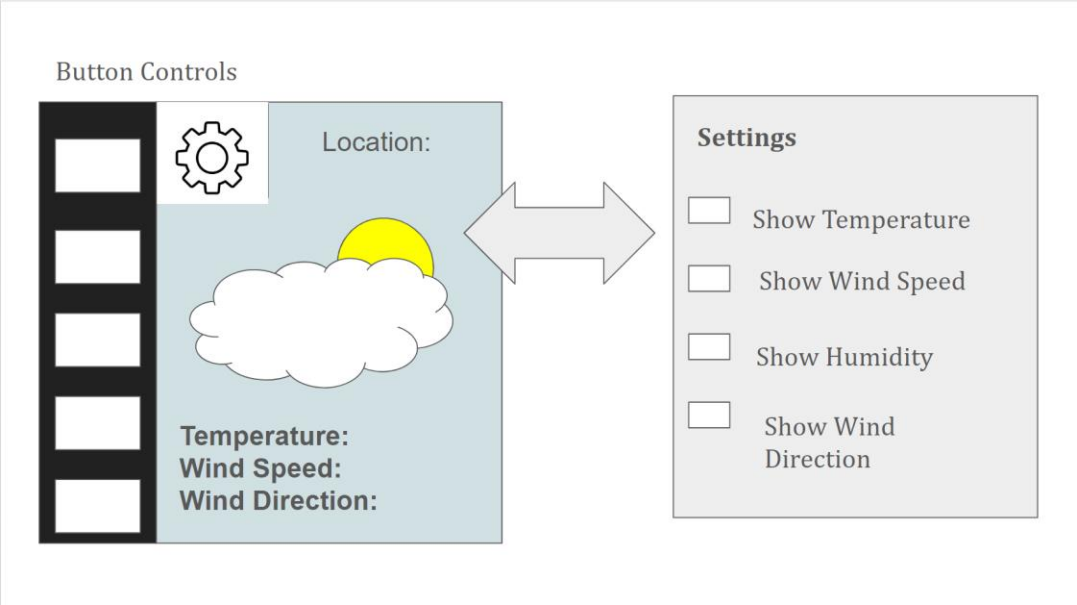



**Figure 7: Wireframe Model for User Interface of Weather Monitoring System**

The Wireframe Model for the user interface of the weather monitoring system (see Figure 4) is designed to be able to display important details of weather to an end user. The user should have the capability to pick and choose which aspects of whether they want displayed to the screen through a Settings page. Accessing the settings page and toggling which details of weather should be displayed/hidden can be controlled through buttons as opposed to a touchscreen interface for easy implementation of user interface by engineers as implementing touchscreen involves more complexity in terms of design than implementation of button controls. However, the

design of the wireframe model is meant to be a key aspect of the indoor aspect of the subsystem referred to in the Logical Design Diagram. As our system solely implements the outdoor field unit responsible for collecting data and transmitting the data to either NYSDEC analysts or the indoor home system, it does not implement the wireframe model or an interface that will display information to the user.

**Physical Design**

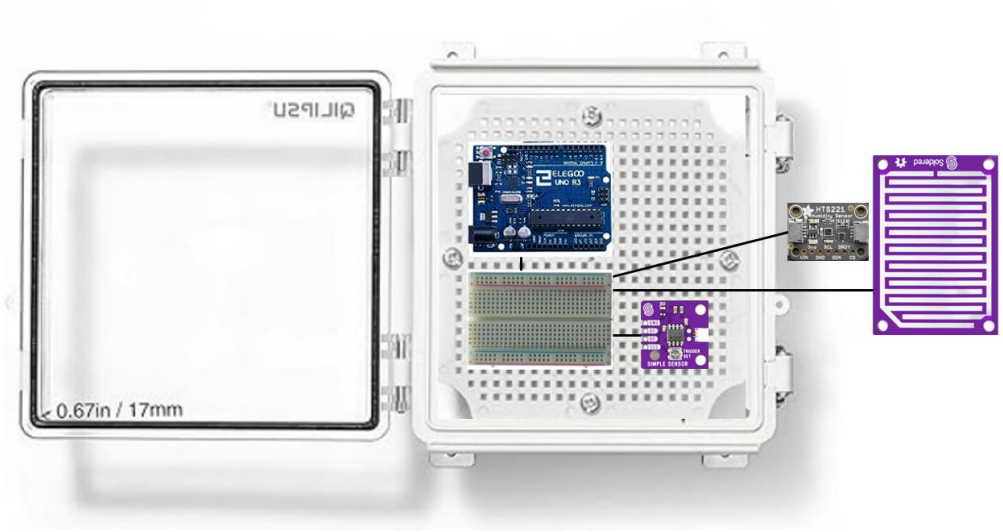



**Figure 8: Physical Design Diagram of Data Acquisition Unit**

Figure 8 provides a physical display of our design for a Data Acquisition Unit used to collect ambient temperature, humidity, and precipitation data. The entirety of the unit is enclosed within a 150 mm x 150 mm x 90 mm waterproof chassis with the Adafruit HTS221 sensor (shown to the right in gray) used to measure temperature and humidity and soldered precipitation sensor (shown also to the right in purple) used to measure precipitation. Data read through each sensor connects to an Arduino Uno microcontroller contained within the chassis for further data processing

**Bill of Materials:**

| **Component** | **Price** |
|---|---|
| Adafruit HTS221 Temperature and Humidity Sensor Breakout Board | **$9.95** |
| UNO R3 Board with USB Cable | **$29.95** |

| | |
|---|---|
| Soldered Electronics 333044 (Precipitation Sensor) | **$7.39** |
| QILIPSU Clear Hinged Cover Stainless Steel Latch 150x150x90mm | **$18.88** |
| Half size breadboard | **$7.99** |
| | **TOTAL: $74.16** |

Component Links:

*Adafruit HTS221 Temperature and Humidity Sensor Breakout Board*

https://www.adafruit.com/product/4535

*Arduino Uno R3 Microcontroller with USB Cable*

https://www.firgelliauto.com/products/arduino-uno-r3-microcontroller?variant=7692979587&currency=USD&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gad_source=1&gclid=CjwKCAjwrIixBhBbEiwACEqDJf3gjAfJxJRdSEWQjrPKu2CzbSZqUv90go9MBeCpRMLops3hF5K1WxoCWYIQAvD_BwE

*Soldered Electronics 333044 (Precipitation Sensor)*

https://www.digikey.com/en/products/detail/soldered-electronics/333044/21720441Q

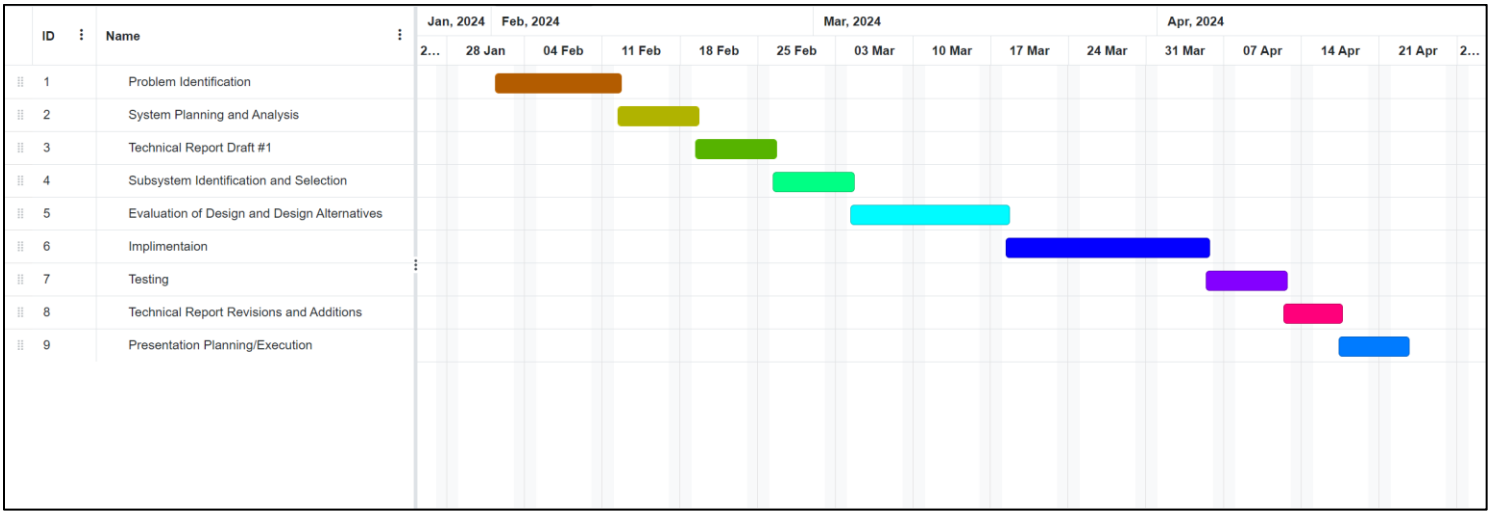
*QILIPSU Hinged Cover Stainless Steel Latch 150 mm x 150 mm x 90 mm*

https://www.amazon.com/QILIPSU-Stainless-150x150x90mm-Waterproof-Electrical/dp/B089K6Z567?ref_=ast_sto_dp&th=1

https://www.sciencepurchase.com/products/03mb801?variant=39970685583533&currency=USD&utm_source=google&utm_medium=organic&utm_campaign=SP%20-%20Google%20Shopping&utm_content=Solderless%20Breadboard%2C%20400%20Tie-Points%2C%202%20Distribution%20Strip

## *Semester Planning*

**Figure 9: Gannt Chart**


ID
Name
Jan, 2024
Feb, 2024
Mar, 2024
Apr, 2024
2...
28 Jan
04 Feb
11 Feb
18 Feb
25 Feb
03 Mar
10 Mar
17 Mar
24 Mar
31 Mar
07 Apr
14 Apr
21 Apr
2...
1 Problem Identification
2 System Planning and Analysis
3 Technical Report Draft #1
4 Subsystem Identification and Selection
5 Evaluation of Design and Design Alternatives
6 Implimentaion
7 Testing
8 Technical Report Revisions and Additions
9 Presentation Planning/Execution

Works Cited

[1]
"Observed and Projected Climate Change in New York State: An Overview AUGUST 2021 Basil Seggos, Commissioner." Available: https://extapps.dec.ny.gov/docs/administration_pdf/ccnys2021.pdf

[2]
"Build Your Own Weather Station," *Raspberrypi.org*, 2017. https://projects.raspberrypi.org/en/projects/build-your-own-weather-station/0 (accessed Apr. 21, 2024).

[3]
P. Sharma and S. Prakash, "Real Time Weather Monitoring System Using Iot," *ITM Web of Conferences*, vol. 40, p. 01006, 2021, doi: https://doi.org/10.1051/itmconf/20214001006.

[4]
F. Claria, "How I Made a Fully Functional Arduino Weather Station," *Toptal Engineering Blog*. https://www.toptal.com/c/how-i-made-a-fully-functional-arduino-weather-station-for-300

[5]
United States Environmental Protection Agency, "Climate Change Indicators: Weather and Climate | US EPA," *US EPA*, Jul. 26, 2023. https://www.epa.gov/climate-indicators/weather-climate