High Altitude Weather Balloon Launch III for Measuring Environmental Pollution

Major Qualifying Project

A Major Qualifying Project Report Submitted to the faculty of the Electrical and Computer Engineering Department at Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science

Submitted by:

Alexander Kobsa

Madeline Kasznay

Shamiha Khan

Advised by:

Professor Maqsood A. Mughal

March 25, 2021

Abstract

The threat of climate change means monitoring greenhouse gas (GHG) concentrations at the atmospheric level is significant in maintaining the health of the planet. The high-altitude ballooning (HAB) project aims to measure GHG levels by launching a payload equipped with environmental sensors into the stratosphere. In this third iteration of the project, the team set two goals: (1) to obtain a full flight of continuous data collection and real-time transmission of environmental data during the flight. During the first successful flight in December, we met the goal of real-time data transmission mid-flight was partially successful as we were not able to receive packets of data at a continuous interval of time. Unfortunately, unexpected strong wind currents forced the payload to land in the Atlantic Ocean, where it was irretrievable. It is unknown whether improvements we made to the payload design led to a successful collection of a full flight's worth of data. The next test in March consisted of carrying the payload to the summit of Mt. Wachusett. This test was a success as the base station received data continuously from the payload transmitter when adequate line of sight was maintained. Overall, we made significant progress in the continued development of the HAB project by creating a data transmission system and improving the reliability of our data collection

Acknowledgments

We would like to thank Professor Mughal for his guidance. His professional knowledge and experience with past teams was crucial for making progress. His willingness to go above and beyond, and provide encouragement made the project a fulfilling experience. We also thank Sizhuo Li, our fourth team member during the first two terms of the project. He played a vital role in performing simulations, helping to construct the base station, being part of the launch site crew, and more. Working with him was a pleasure, and we hope he thrives in future endeavors. We want to thank Bill Appleyard for helping us source our parts and for cutting our plywood for the base station. We appreciate the insight provided from interviews with Dr. Kaveh Pahlavan and Dr. Alexander Wyglinski from WPI on wireless communication and Dr. Richard Eason from the University of Maine on his experience with high altitude ballooning. We would like to thank Andrew Kasznay for his time and support, in helping the team with programming and obtaining new sensors. We are grateful for Rayna Harter's assistance in formatting results from the test in March. We lastly want to acknowledge Jerry Rutherford from the Federal Aviation Agency and Boston, Yankee, and Albany Air Traffic Control for their assistance with our launch, and Michael's Party Rental for providing helium gas for our balloon.

Executive Summary

The High Altitude Balloon (HAB) project consists of a helium filled weather balloon launched into the upper atmosphere to collect atmospheric data and study climate change in New England. The project is a continuation of the previous MQPs with the aim to measure and monitor environmental data from at high altitudes of 20,000 to 80,000ft over a longer period. The goal is to monitor and study the changes in air quality in the atmosphere. Environmental monitoring serves a vital scientific role by revealing long-term trends that can lead to new knowledge and understanding, as well as prevent the planet from any catastrophic events. This is the third year WPI has run this project and we built upon the work done by the past two teams. Every year two launches take place, one in November and the other in March. For our contribution to this project, we had two major goals: (1) design a real-time wireless data collection system for the flight, and (2) ensure data collection during the entire duration of the flight at the ground by setting up a base station. Due to complications in the project, we did not launch in November but rather on December 5th and our March launch had to be changed into a ground test which occurred on March 19th. A further goal of the project was to also be able to transmit real-time live pictures making the livestream interesting visually to outside observers.

To achieve our first goal of collecting a full flight's worth of data, aspects of the previous team’s payload were redesigned. The payload is primarily a Styrofoam box with the sensors embedded into the wall to face the exterior of the payload and the other equipment like the Spot Tracker, a GPS tracker, are kept inside. The changes to the payload will ensure that the wires are much less likely to become disconnected mid-flight and at the time of launch when the payload experiences strong forces due to strong wind currents and lift-off. Also added to the payload was a new GPS sensor which gave the project redundancy for critical data. The new GPS sensor

allowed the transmission of GPS data from a second source other than the existing Spot Tracker. Lastly, new radio hardware which is better able to transmit flight data.

To collect data mid-flight of our balloon, we needed to establish a wireless connection with the payload. This was a difficult task because wireless communications at extended ranges suffer from the signal loss effects of radio wave propagation. Therefore, we designed a system that can operate at the lowest frequencies possible and with the largest amount of transmitting power and high antenna gains to achieve the best possible coverage area of our system. We created a base station receiver to be placed underneath our predicted flight paths to collect our live data, and this base station includes Bolton Technical's Long Ranger directional antenna, attached to a 24V stepper motor with 1.8° per step controlled by an ESP32 to maintain directional pointing at the balloon. Next, we installed a lightweight, waterproof, omnidirectional antenna operating at 900-930 MHz aboard the payload to transmit our data from the payload. Further, we acquired two identical 902-928 MHz radios which were programmed to be a Tx-Rx pair, coupled with their respective antennas, and they allowed us to encode and decode our digital data to send through the antennas. Lastly, we developed a set of Python scripts to establish an interface between the computers and the radios, enabling us to send data to a receiver on the ground in flight. We took this one step further by also using the Python scripts to display the received data live using MathWorks ThingSpeak API.

To maintain consistency with the launches from the previous two MQP projects, the team chose launch dates in mid-November and mid-March as the project runs annually. We also chose Lincoln Park in Albany, NY as our launching point to maintain consistency. We performed our November 21st launch attempt there, however, we failed to launch our payload. However, due to weather forecasts for the Dec 5th launch day, the launch site was moved further

east to Cobleskill, NY, to avoid the payload from landing in the Atlantic Ocean. This was predicted to keep our payload from landing in the ocean but was ultimately unsuccessful. For our March launch, we were unable to do a full flight test due to inclement weather on the planned day of the launch, March 19th. Therefore, we performed a ground test at Mt. Wachusett the next day, March 20th.

The team sought suitable base station locations for each launch. An ideal location is near the midpoint of the balloon's flight path, and at the highest possible point of elevation with an unobstructed view of the payload. These features reduce interference in order to receive high-quality feedback, while also remaining within range of the payload's transmitter. We conducted flight simulations based on the payload's weight, desired rate of ascent, balloon size, and desired burst altitude using Cambridge University Spaceflight Landing Predictor. However, simulated flight paths depend on the time of the year and weather conditions and could only be conducted up to a week in advance. So, we chose possible base station locations within a week of the launch date to ensure they were close to the flight path. To determine the accessibility of a site, we used Google Earth's satellite imaging for locating cleared public lands underneath the predicted flight path. The base station locations we chose were: Petersburg Pass (Nov. 21st), Mt. Greylock Visitor Center (Dec. 5th), and Mt. Wachusett Base Lodge (Mar. 20th).

During the November 27th launch, the balloon was accidentally released during the filling process, and we failed to launch our payload. Therefore, we planned another launch on December 5th. During this second launch the payload transmitted live data to the base station, allowing us to livestream the progress of our flight. While the payload was not recovered, the team did successfully receive live data during the flight at the base station. We received eight minutes and thirty-seven seconds of transmitted data from our payload, which was displayed

using the ThinkSpeak API. We gathered seven different data feeds from the following sensors: barometric pressure, low pressure, temperature, UV, CO_2, and ozone levels. The values from the sensors were in the expected range, except the NO and UV sensor which issues. With the NO sensor failure and the UV data needing more readings to make sense. This data made the launch somewhat successful as we confirmed the functionality of the radio transmission and gathered data for a portion of the flight.

For our March 20th ground test, we were able to successfully test our system and achieved our desired results. Our wireless communication system performed well, and the base station received data from the payload for the majority of the test. Our payload sensors also worked without error, providing us with a reference set of data to compare to the data recovered at the base station and evaluate the system's performance. While data collected during the ground test is insignificant to understanding GHGs in our atmosphere, we proved our system is capable of collecting it.

Authorship

Section	Primary Author(s)	Editors
Title Page	Shamiha	All
Abstract	Alex	All
Executive Summary	All	All
Acknowledgments	Shamiha	All
Authorship	All	All
List of Figures	Alex	All
List of Tables	Alex	All
Table of Contents	Alex, Shamiha	All
Introduction	Shamiha	Alex
Background	Shamiha	Alex
Methodology	Alex, Madeline	All
Results	Alex, Madeline	All
Conclusion & Recommendations	Madeline, Alex	All
References	Madeline	All
Appendices	Alex, Madeline	All

List of Figures

List of Tables

Table of Contents

Chapter 1. Introduction

1.1 Significance

Greenhouse gas (GHG) emissions are a major concern to the environment and society. They contribute to climate change by trapping heat in the atmosphere, raising the temperature of the planet. GHG emissions also pollute the air which harms the health of people and the environment. Human activity has been the main contributor in recent times due to burning fossil fuels and industrial growth. In the United States, gross GHG emissions have increased by 2 percent since 1990 and fluctuate due to multiple factors such as changes in the economy (EPA). High Altitude Air Balloons travel to the upper atmosphere where sensor readings on the GHG levels can be taken. This project aims to study the GHG emissions over New England to see how they change over the course of several years by launching a HAB semiannually.

1.2 Scope

The two major goals for this iteration of the project were successful collection of a full launch's worth of data and capturing real-time information from the ground during the flight of the high-altitude balloon (HAB). Reliable data collection is important for monitoring long term the level of GHG in New England's atmosphere, but past teams had issues collecting data for most of the flights due to issues with the payload's electronics and software. Real-time data transmission made it possible for observers to follow the flight, allowed us to gain a better understanding of what is happening to the payload in flight, as well as allow for a partial data recording if the payload is not recovered. This will be accomplished by the inclusion of a newly created base station.

The base station consists of a receiving antenna that will be aided by a rotating platform with a motor attached. This motor circuit will include a motor driver, a stepper motor, boost

converters, and an ESP32 microcontroller. The payload included sensors for ultraviolet light (UV), ozone, nitrous oxide, and carbon dioxide levels. We included other sensors in our payload to monitor its position and environment with temperature, altitude, pressure, and global positioning system (GPS) sensors. A GoPro was used for taking video, and a SpotTracker to monitor GPS coordinates and trajectory of the payload in real-time for payload retrieval. A chute release was added to shorten the flight time after burst to enable easier retrieval of the payload. The Bolton Long Ranger was our chosen antenna for the base station, and the Eifagur 12 inch LoRa antenna for the payload transmitter. Data received at the base station was graphed in real-time using MathWorks ThingSpeak API.

Chapter 2. Background

2.1 Literature Review

At the beginning of this project, our priority was to understand how previous teams were unable to consistently achieve a full flight's worth of data. After reading their reports and talking with the previous team over a Zoom call, we learned that the most significant issues were sudden changes to the sensors' operation in flight and being unable to maintain a stable wireless signal with the payload in flight. Therefore, we sought to diagnose and find solutions to these problems. We also did review some online resources regarding balloon operations and long range communications systems in order to familiarize ourselves with the concepts and products we would need. Using these resources as guides, we investigated multiple products on the market that might be suitable and performed comparative analyses.

2.2 Interviews

The team interviewed three professors during the course of the project. The first interview was with ECE Professor Wyglinski on data transmission and choosing suitable components. The professor suggested LoRa for the project, but with the future goal of sending live visual information, it did not fit. The interview with ECE Professor Pahlavan centered around data transmission, antennae options, as well as the viability of a base station. The last interview conducted was with Professor Richard Eason, a professor with experience running multiple HAB launches, who provided advice on payload design and details regarding his past launches.

Chapter 3. Methodology

3.1 Real-Time Data Transmission

To collect data mid-flight of our balloon launch, we needed to establish a wireless connection with the payload. This is a difficult task because wireless communications at extended ranges suffer from the signal loss effects of radio wave propagation. We used the Friis equation to determine our ideal frequency of operation. Using the Friis Equation (Equation 1), we kept the transmit power and both antenna's directivities constant to maximize the power received at the receiver by maximizing our signal wavelength, or by minimizing our signal frequency.

$$\frac{P_r}{P_t} = D_t D_r \left(\frac{\lambda}{4\pi d}\right)^2$$

Equation 1: The Friis Equation

Therefore, we needed to design a wireless system able to operate at the lowest frequencies possible and with the largest amount of transmitting power and high antenna gains to

achieve the best possible coverage area of our system. We chose to solve this problem by creating a one-way, wireless communication system, consisting of a small, omnidirectional antenna operating at low frequencies mounted to the payload and a stationary base station on the ground, equipped with a high gain, low frequency, directional antenna which will be attached to a stepper motor rotating around the horizontal axis to maintain directional pointing at the balloon. To operate this antenna system, we needed two radios which operate at the same frequencies so we can properly modulate and demodulate our data along with two computers that interface with the radios to send and receive digital data.

3.2 Base Station

Our base station consists of a directional receiving antenna (Rx), a 360 degree, horizontally rotating platform, a high torque stepper motor and motor driver system, a receiving radio module, a 3-foot coaxial cable, a power supply with a 120VAC to 24VDC converter, an ESP32 microcontroller, and a small wooden frame. When designing this base station, we took into consideration the size, weight, stability, and ease of assembly for the overall device. This led us to create a portable, lightweight, yet functional base station.

3.2.1 Antenna

The receiving antenna that we purchased for this project was Bolton Technical's Long Ranger, which featured a large range of operating frequencies (600 MHz-6500 MHz) and high gains across all frequencies (8-15 dBi @ 600-960 MHz, 20-28 dBi @ 1700-2200 MHz, 10-26 dBi @ 3000-6500 MHz). We chose to operate in the lowest frequency band because lower radio frequencies lose less power when propagating through free space and there is an ISM band at 902-928 MHz so, we will not be violating any FCC spectrum regulations by operating in that band. In order to maintain the directional beam of the antenna focused on the balloon, we will

use the GPS tracker inside the payload to send positional updates to the base station which will then determine the correct direction and amount for the antenna to turn to track the balloon's movement. If the motor is not operational, we will still be able to manually direct the antenna using handheld compasses and some basic trigonometry.

3.2.2 Frame and Motor

When designing our frame, we focused on achieving our goals of stability and 360-degree horizontal rotation while also minimizing weight and space. We chose to make our frame out of 4-ply, 15/32" plywood because it is cheap, light, yet sturdy enough to not break under moderate stress. For the size of the frame, we needed it to large enough to be stable when supporting the antenna, but small enough to be reasonably carried if the base station location was inaccessible by road. Therefore, we chose to make the base station frame 31.25 cm by 31.25 cm by 33 cm, which can be reasonably carried by hand and the combination of its weight and size provide enough stability for our antenna.

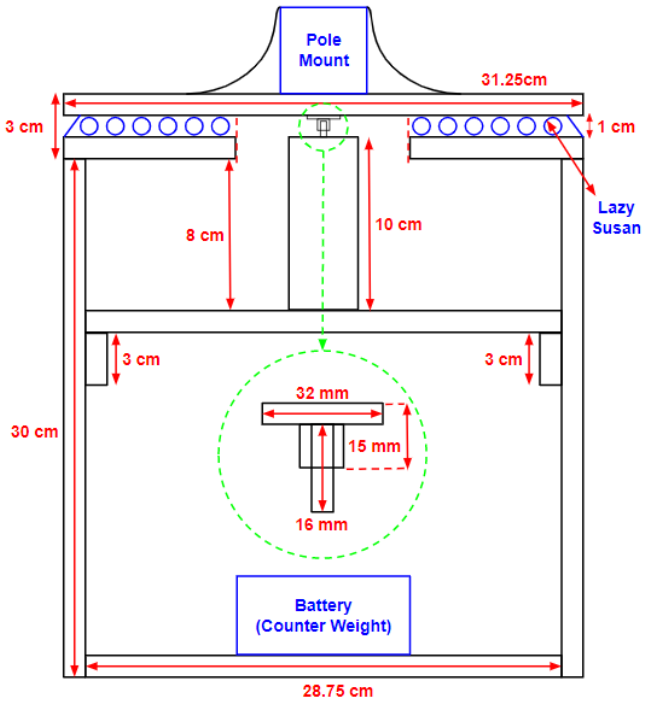

Figure 1: Schematic Drawing of our Base Station

Figure 2: Picture of the completed Base Station

The inside of the frame is hollow, except for the motor platform and the stepper motor-driver circuit. To achieve the goal of 360 degrees of horizontal rotation, we used two square pieces of plywood with a Lazy Susan sandwiched between them, with a stepper motor attached

to the center of the top plate only. A Lazy Susan is a circular metal disk that rotates over a circular ring of ball bearings, which we used to rotate the antennae. We chose to purchase a 12-inch Lazy Susan with a 1,000-pound capacity, so that it fit nearly perfectly on top of our frame while removing any concerns about the weight of the antenna affecting its rotation. In order to attach the stepper motor to only the topmost plate of the frame, we cut a hole in the lower plate beneath the Lazy Susan, so that the axle of the motor could get through. To attach the axle to the top plate, we used a flange mount to mount it to the underside of the top plate. The size of this flange mount depends on the size of the motor axle. When choosing the motor, we concluded that a stepper motor would best fit our needs. Our requirements for the motor were a slow turning radius, high angular precision, and high torque. We wanted our antenna to turn slowly and with high precision because the balloon will be far enough away from the base station that there will be a slow rate of angular change between the two objects. We also wanted a high holding and moving torque because the antenna weighs around 9 lbs. and it is two feet above the motor, so it has the potential to create high torque against the motor in windy conditions. Therefore, the stepper motor we chose was STEPPERONLINE's Low Current Nema 23 CNC Stepper Motor, with a rated current of 1.8A and a holding torque of 340oz.in/2.4Nm. It also features a 1.8-degree angle change per step, allowing us to make small changes in our antenna's pointing direction. The motor is equipped with an 8 mm wide, D-cut axle that can fit into a 5mm flange mount, which we purchased. To operate this motor, we purchased STEPPERONLINE's CNC Stepper Motor Driver, rated from 1.0-4.2A and 20-50VDC. This motor driver connects to the motor's leads and the power supply to properly regulate the power to the motor and protect our microcontroller from the dangers of excess current from the motor. The motor driver takes in 5V digital logic pulses as control signals into its circuit, so we must use 3.3V to 5V boost

converters with our ESP32 to generate the necessary voltage for the control signals from the ESP32 because the ESP32 is a 3.3V microcontroller. The ESP32 will be programmed to receive an input latitude and longitude coordinate from an SPI connection and then compare it to its own, known coordinates and then generate the correct compass bearing necessary to point the antenna at the payload. After this calculation, the ESP32 will determine the correct direction and amount to turn the antenna by operating the stepper motor with the ESP32's Pulse Width Modulated (PWM) signal generator. For the compass bearings to be accurate, the ESP32 will need to be aligned with true North or otherwise be told its initial direction of pointing, which will be done by our team prior to operation of the base station.

Figure 3: Close view of Base Station frame

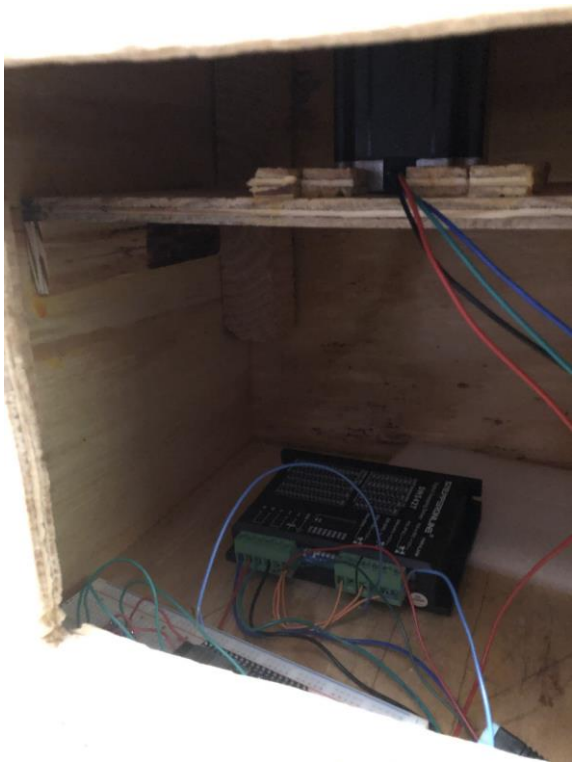

Figure 4: Inside view of the Base Station with the motor and motor driver inside

3.2.3 Location of Base Station

After finishing the base station, the team needs to pick a suitable location for our base station. An ideal base station should have two features: located around the middle of the balloon's flight path and located in an open area, maximizing the amount of visible sky, like a field or mountain top. Both features aim to avoid interference from other radio emitters and physical objects to receive high-quality feedback from the balloon.

To choose a base station site, our team used a balloon flight simulation website to simulate our balloon's flight path so that we could find a location closely under the predicted center of the flight path. This website is Cambridge University's Spaceflight Landing Predictor, which is a free software used to predict the flight of latex balloons up to a week in advance. The reason for it only being able to predict one week ahead is due to the volatility of weather and

wind patterns that change yearly. To use this software, we do need to know some details about the balloon like its launch location, launch time, ascent rate, burst altitude, and descent rate. In order to perform these calculations, we utilized the High-Altitude Science Balloon Performance Calculator, which produces the ascent rate, burst altitude, ascent time, and the amount of helium required for the balloon given the balloon's size in grams, the payload weight, and positive lift. With this information, we can produce and plot the KML files produced by the software to see both the burst location and predicted landing location. An example of this is seen in Figure 5, which shows the results of our flight simulations leading up to our first launch.

However, leading up to each launch and before accurate simulation results could be produced, we rely on simulations that were a week away from the current day and updated every few days to decide our base station location. The team initially chose Mt. Holyoke as the primary base station due to its height, moderate accessibility, and its location around the center of the flight paths of previous launches. We also identified another potential base station site at Mt. Greylock, since it had many of the same characteristics Mt. Holyoke, plus it featured an auto road and higher elevation. We initially decided to choose Mt. Holyoke as our primary base station location since it was closer to the predicted flight path of previous years' simulations and it was easy to get to, being only a 30 minute hike from a parking lot. However, as the first launch date approached and the weather can change quite quickly, Mt. Holyoke became unusable as a base station location because it would be too far from the predicted flight path. Therefore, we elected to not make a final decision for each flight's base station location until accurate predictions can be produced. Once we have consistent results for the balloon's expected path, as seen in Figure 5, we begin using satellite maps to scan the landscape beneath our expected flight path because we are looking for large, open spaces at high altitudes. Using this method, we are

able to find public parks and other public lands that we can use with permission from the local government. These potential locations must be within 2-4 miles of the expected flight path so that our base station will be in range of payload when also considering the vertical distance.

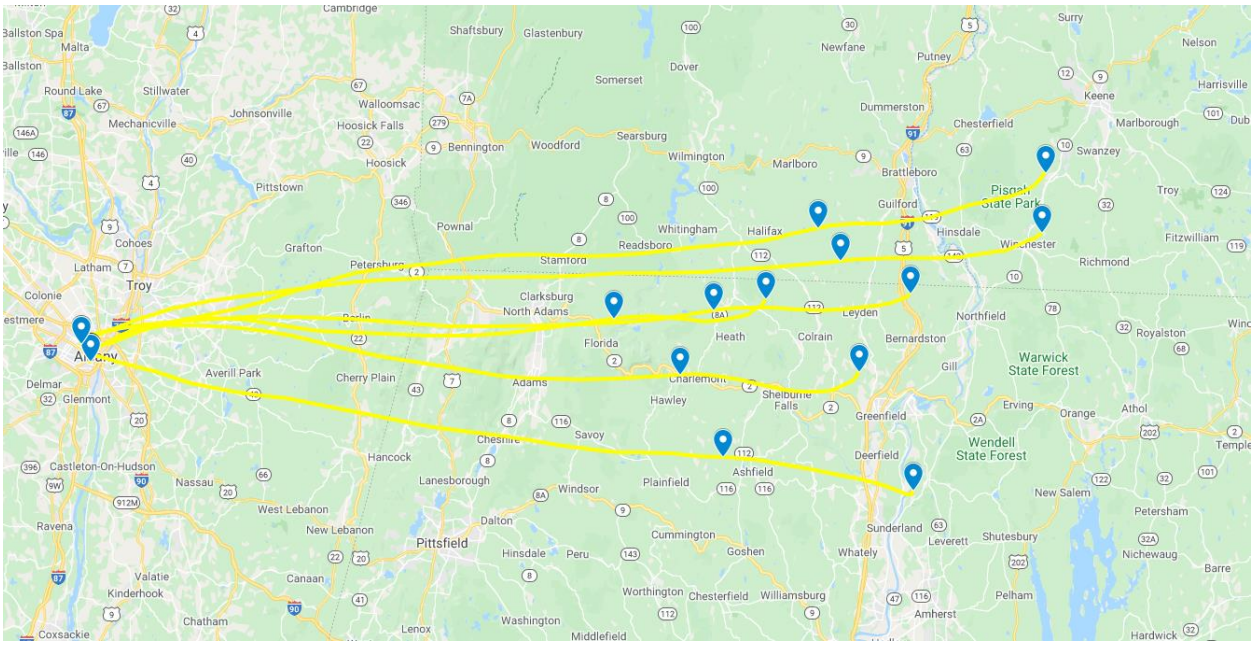

Figure 5: The 6 flight paths predicted by our simulations over multiple days leading up to our first launch attempt, with burst location and landing site

3.3 Payload Communication

To use our receiving base station, we need a functional radio transmitter operating aboard the payload. We chose a 12-inch, waterproof, omnidirectional antenna to attach to the payload. Eifagur's Waterproof 4dBi LoRa Gateway antenna operates at 900-930 MHz with 4 dBi of gain which perfectly fits in with the operating range of the Long Ranger and the ISM band. In addition, this antenna can survive upper atmospheric climates and it transmits in all directions, meaning that the orientation of the payload will not affect the reception of our signal at the base station. It has a maximum transmit power of 50W, but we cannot exceed 250mW to stay within FCC regulations.

3.3.1 Radios

Our two antennas both need to be attached to radios that can perform the conversion and modulation of digital data into a passband radio signal. We decided to purchase two identical radios so that compatibility between the radios would not be an issue. The radio we choose will also need to be able to have a RP-SMA female coaxial cable connection so that it can connect to the antennas by N type female to RP-SMA male connector cable. The radio we selected was Digi's XBEE-PRO 900HP radio module, which has features like a frequency band of 902-928 MHz, a RP-SMA female connector, a low receiver sensitivity (-101 to -110 dBm), variable data rates (10 Kbps or 100 Kbps), an SPI interface, and multiple network topology options. This radio advertised a wireless communication range of up to 9 miles in an outdoor setting, which is the exact type of performance we want for our radio. The only downside to using this radio is that its pin connectors are designed to fit into a proprietary development board, also developed by Digi. This makes it nearly impossible to properly interface with the radio without the boards and these development boards possess micro-B USB ports, allowing us to use our computers to digitally interface with them. We purchased two XBee THT Groove Development Boards so that we could program both radios easily and they also enable our computers to send data to and receive from the radios.

To program our radio's settings, we decided to use Digi's XCTU software. This software is designed for interfacing with and managing multiple Digi radio modules, with the capability to handle complex networks. Since our network is simple, we did not use any of these advanced settings. After connecting the radio module to the development board and plugging it into a computer, the XCTU software scans and connects to our radio module using an USB serial port connection over the USB cable. The default values for this serial connection are 9600 bits per

second, 8 data bits, No Parity, one stop bit, and no flow control. For this system to work, our Windows PC also needs to know which serial port the radio is on since most Windows PCs can handle multiple serial connections at once. The correct port can be identified by plugging and unplugging the device and seeing which COM ports appear and disappear in Windows' Device Manager under the Ports list. Once the radio has been identified by XCTU, we are able to see and change every setting that the radio has, however, we only made changes to the Network ID and Node ID. We made the Network ID of both radios to be the same, unique value so that we could ensure they were on a network by themselves and not interfering with other networks. The Node ID is just a simple text name so one node was called "Payload" and the other "Base Station". Some other notable settings that we did not use are the API Mode, Tx Power level, and Channel Mask. This radio comes with an API that is useful when dealing with complex networks, but that is not something we need so we decided to choose the simplest option, which is Application Transparent Mode. In this mode, all data coming in and out of the radio is unaltered or otherwise processed; all of the data is passed through from the input to the antenna and vice versa. When choosing the Tx Power level, we want the maximum because it maximizes our transmission range which is also the default setting. The maximum transmit power of this radio is 250mW, which is the maximum allowed power by the FCC. Lastly, the Channel Mask allows us to selectively use certain channels within our radios Frequency Hopping algorithm. We left this on its default since we did not have any reason to change it since the default is using all but one of the channels available.

3.3.2 Python Code and Computer Interface

The final component of our communication system is the computer software that will perform the digital processing and formatting of our data in real-time. We achieve this by using a

Raspberry Pi 4 inside the payload and a laptop PC at the base station. The Raspberry Pi 4 handles the data from the sensors and Arduino and it also interfaces with the XBee radio through a serial communications port, which is enabled by a set of Linux drivers available on Digi's website and Windows drivers from Silicon Labs. The laptop at the base station receives the transmitted data from the XBee radio through its own serial communications port. Any Windows PC running Windows 10 or later should already have the necessary drivers installed to receive the data from the radio. To process that data, we chose to use a set of Python scripts. We created three python scripts with the names: Main.py, Plotting.py, and Bearing.py. Main.py is responsible for opening the correct COM port with the same serial connection setting as the radio, reading in the serial data as it comes in, and then it appends the data to the text file Basedata.txt.

Plotting.py first opens Basedata.txt and reads in the lines of text in the file, starting from the first line up to the most current line as they are written in. After the line is read in, a series of 12 'if' statements check the first 7-8 characters of the line to see if they are a match to the leading characters around one of our 11 fields of data: Pressure, Exterior Temperature, Altitude, Ozone, Carbon Dioxide, Ultraviolet Light, Nitrogen Oxide, Interior Temperature, Interior Pressure, Latitude, and Longitude. We included some redundant sensors to ensure the validity of our data, like an extra altitude sensor. If a match is found, the line of text has all of the leading and lagging characters stripped off and the remaining numbers are then converted from text into integers or floats. These pieces of data are then stored in their respective data lists. The data lists are kept as global lists inside the plotting.py file before we send them to our live data visualization. To display our data in real-time, we utilized MathWorks' ThingSpeak API, which is a free, online software which allows us to input up to 8 different fields of data to be displayed

live. To get our data into ThingSpeak, we needed to send it as a JSON object in the form of a URL containing our data. We send the last element of each data list, which is embedded into a single URL, to ThingSpeak every 5 minutes because that is the fastest setting for the free software.

Some changes were made to plotting.py near the end of the project to allow for it to plot data itself, because the timing limitation of ThingSpeak made it impossible to use for any data taken at a faster rate than that. These changes included importing the new libraries of Matplotlib, Numpy, and CSV; these libraries allowed us to do the necessary data analysis and plotting of our data. This included tasks like ensuring the sizes of the data lists are the same as the time index list, removing null values, and exporting the data into a CSV file which can then be further analyzed in other programs like Microsoft Excel. We

Our last Python file, bearing.py, is a simple program that allows the user to input two different latitude and longitude coordinates and it returns the compass bearing that would result from standing at point 1 and looking towards point 2. This is a vital program since it allows us to determine the direction in which to point the base station antenna to best capture the signal being sent by our payload in flight.

3.4 Payload

One of our main goals for this project was to have a full flight's worth of data recorded by the payload and sent to the base station. To achieve this goal, we made several design changes to the payload. These changes include adding more sensors to the payload, ensuring the box is well insulated, and improving the radio communication.

3.4.1 Payload Construction

When constructing the payload, we made a few changes to the previous year's design. We wanted to ensure that the inside of the payload stayed warm enough to keep the electronic components within their operating temperature. We opted to use a slightly smaller payload box so that when we put together the payload, there was less empty space needed to be kept warm. Most of the sensors had to be exposed outside the payload to the cold, but we added insulation behind the exposed sensors and used a smaller protoboard to keep the inside of the payload warm.

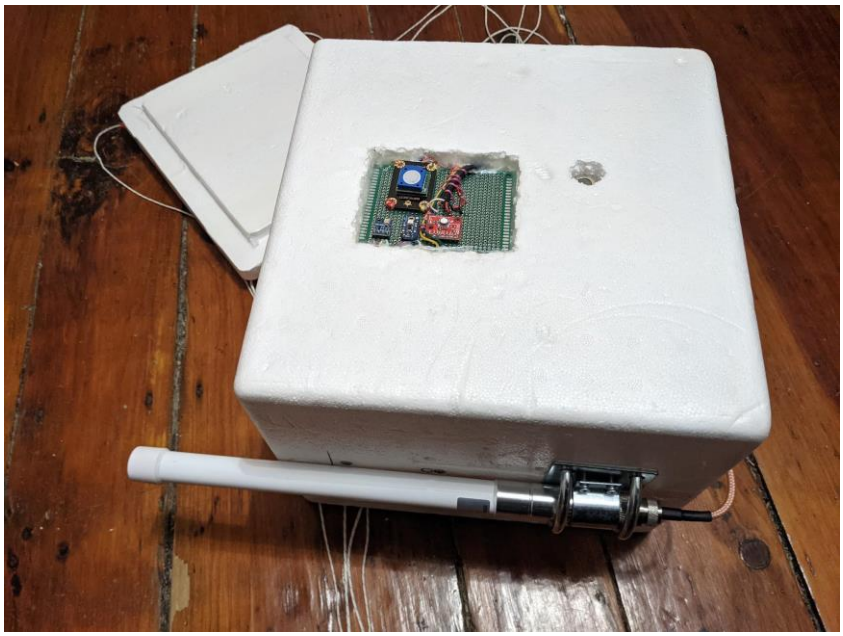

Figure 6: March Launch Payload box

Another addition was to eliminate a breadboard within the payload, replacing it with protoboards and soldered joints. As a team we were concerned about wires coming loose during flight. The previous team used a breadboard to connect many of the wires, which tend to come loose with vibration. Considering this, we changed the wire connections to be soldered onto a proto-board which is far less likely to lose connection. In addition to soldering onto the proto-board, we used crimp-on connectors for wires that would need to be frequently disconnected. We

also used hot glue to ensure that jumper wire remained connected. These connector types will ensure that the wires are much less likely to become disconnected mid-flight.

The parachute attachment system was also changed. We added a Chute release which is a device that will release the parachute a set distance above the ground. This way after the balloon pops the parachute will not open until it is 10,000 feet above the ground. This ensures that the distance between the bust site and the landing site is much smaller. We did this to try to prevent the balloon from going farther than predicted. We also included a radar reflector per FAA regulations.

3.4.2 Overview of Payload Systems

On Board the payload there are many electronic components that work together to gather sensor data, save that data on-board, and transmit data to the base station. Figure 6 below shows the path that the data takes on board the payload. First data from five sensors is read and processed by the Arduino. The Arduino will do the initial processing of the sensor data and then send that data to the Raspberry Pi serially. The Raspberry Pi will read the serial input from the Arduino and then do some additional data processing and read the data from additional sensors. It will then write the sensor data as well as some status data to an onboard log file. It will then process the data to be sent to the radio and send it serially to the Xbee radio module. The radio is connected to an on-board antenna which transmits to the base station. All these components need to work together to ensure that for each launch we get as much data as possible from our flight.

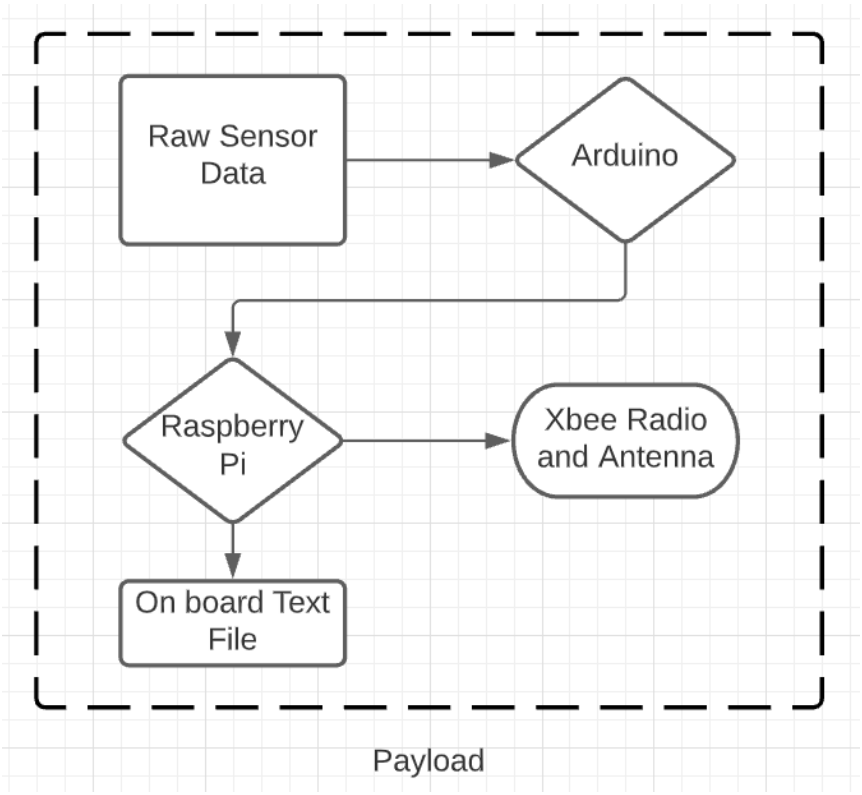

Figure 7: Data flow On-Board the Payload

3.4.3 Payload Sensors

For the payload design we had to pick sensors that were likely to continue functioning throughout the entire flight and produce accurate data. To this we had a few criteria for our sensors: they had to work with our Raspberry Pi or Arduino, they had to work in cooler temperatures, and read data within the correct range for each sensor. The primary types of data we wanted to gather from our launches were: pressure, temperature, altitude, CO_2 concentration, ozone concentration, UV index, methane concentration, and NO concentration. We were not able to find a good NO or Methane sensor that read just a NO or a Methane value and was not a combination gas sensor. Which may be an improvement that another team could make. We also added some secondary sensor data which was less critical but served either as a backup to preexisting sensor value or as an additional value. For these launches, generally, the primary

sensors were read by the Arduino and the secondary sensors were read by the Raspberry Pi. The separation between the sensor readings was done because the Arduino was unable to read the GPS values because it had to be read in UART over USB. The auxiliary temperature and pressure sensors were also read on the Pi, which can read values from only a few sensors. This was done to keep the sensor's power draw from the Arduino within specifications.

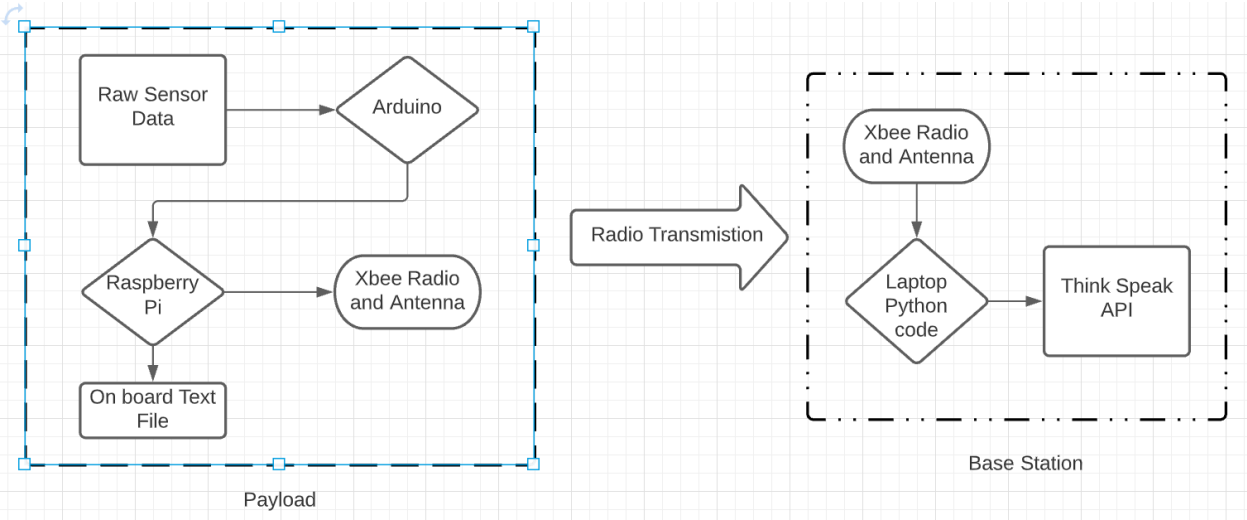

Figure 8: Function Block Diagram of our Payload-Base Station wireless communications system.

Table 1: Specifications of Payload Sensors

Sensor	Operation Range (-°C)	Voltage Input (V)	Sensor Data Range	Output Type	Selection Reason
Jtek BMP180 (Barometric, Pressure, Temp.)	-40 - 85	3.3 - 5	Pressure: 300hPa - 1100hPa Altitude: 0 - 30,000ft	Digital (I2C)	Temperature range, support libraries
DFROBOT Gravity V1.2 (CO_2)	-20 - 50	3.7 - 5	CO_2: 0-10000ppm ± 100ppm @ 400ppm	Analog	Temperature range, simple usage
GUVA-S12D (UV)	-30 - 85	2.7 - 5.5	Uses a UV index of 0-10	Analog	Temperature range, simple usage
MiCS-5524 (Methane)	-30 - 85	5	CO (~ 1 to 1000 ppm), Ammonia (~ 1 to 500 ppm), Ethanol (~ 10 to 500 ppm), H2 (~ 1 - 1000 ppm), and Methane / Propane / IsoButane (~ 1,000 ppm)	Analog	Temperature range, simple usage
Gravity I2C (O3)	-20 - 50	3.3 - 5.5	O3: 0-10ppm resolution of 10ppb	Digital (I2C)	Temperature range, simple usage
YIC 3155-YIC93030PGMFGG-U8-N-ND (GPS)	-40 - 85	2.8 - 5.5	Position accuracy: ~2.5m depends on number of reachable satellites	Digital (UART)	Use of multiple satellite networks to gain accuracy
One-Wire Temperature	-55 - 125	3 - 5.5	-10°C to +85°C ±0.5°C accuracy	Analog	Temperature range
MS5803-14BA High Resolution Pressure Sensor	-40 - 85	1.8 - 3.6	0 to 14 Bar at a resolution of 0.2mBar	Digital(I2C)	Pressure range and resolution

Table 1: Specifications of Payload Sensors

On board the payload there are two different computers for processing the sensor data. The first computer, the Arduino, inputs the raw sensor data and performs calculations to ensure the data has the correct units. We rewrote the Arduino code from previous teams to run more efficiently and utilize newly available libraries. The Arduino reads the data from the barometric pressure sensor, the UV sensor, the CO_2 sensor, the ozone sensor, and a combined gas sensor. Three of the sensors use analog input and the other two use SPI clock signals. In order to send the data to the Raspberry Pi, the Arduino adds heading info to each type of sensor data to indicate which sensor the data came from. The data then gets sent through the serial port where it is read by the Raspberry Pi.

The Raspberry Pi reads data from additional sensors, saves the sensor data onboard the computer, and sends the sensor data to the radio. The three sensors read on board the Raspberry Pi are the pressure, temperature (both CPU temperature and external temperature), and GPS sensors. The Pi saves this sensor data as well as the data from the Arduino sensors in time-stamped text files. These text files are read once the payload is retrieved to get all the sensor data. The smaller select amount of data is then serially out to the Xbee radio module.

Due to both the GPS module and the Arduino module having their own input clocks, threading was used in the Raspberry Pi code so they could both input data simultaneously. The code runs in three threads, one for the Arduino input, one for reading the Pi sensors, and one to transmit the data. The Arduino data thread reads in the input from the Arduino and saves it to a text file and a global variable. The Raspberry Pi sensor thread does something similar reading the sensor data and writing it to a text file and global variable. The Arduino thread reads in new data every 5 seconds and the Pi sensor data reads in new data once a second. The transmit thread reads the global variables and transmits them once a second. This allows for some redundancy in sending our Arduino data to ensure it is received.

A few tests were run on each sensor before integrating the sensors into the payload. We tested if each sensor was working within expected values for the ground level. This included checking for the expected values over several days and comparing them to our measured values. Next, we checked if the measured values adjusted when we changed the conditions the sensor was in, such as blowing on the CO_2 sensor or putting the temperature sensor in the fridge. We also ran tests all the sensors working simultaneously. Lastly, we tested if the sensors could operate in reduced temperature environments by putting the sensors in the freezer and running them for about an hour. After fully testing the sensors, they were integrated with the rest of the payload.

3.4.4 On-Board Data Storage and Transmission

Onboard our payload we had a couple of different ways we stored and transmitted data. We wanted to ensure that we had the best chance to get as much data as possible from each flight. To do this we stored data onboard in the form of log files and transmitted a shortened version of that data to the radio. The path that the data took can be seen in Figure 7: below.

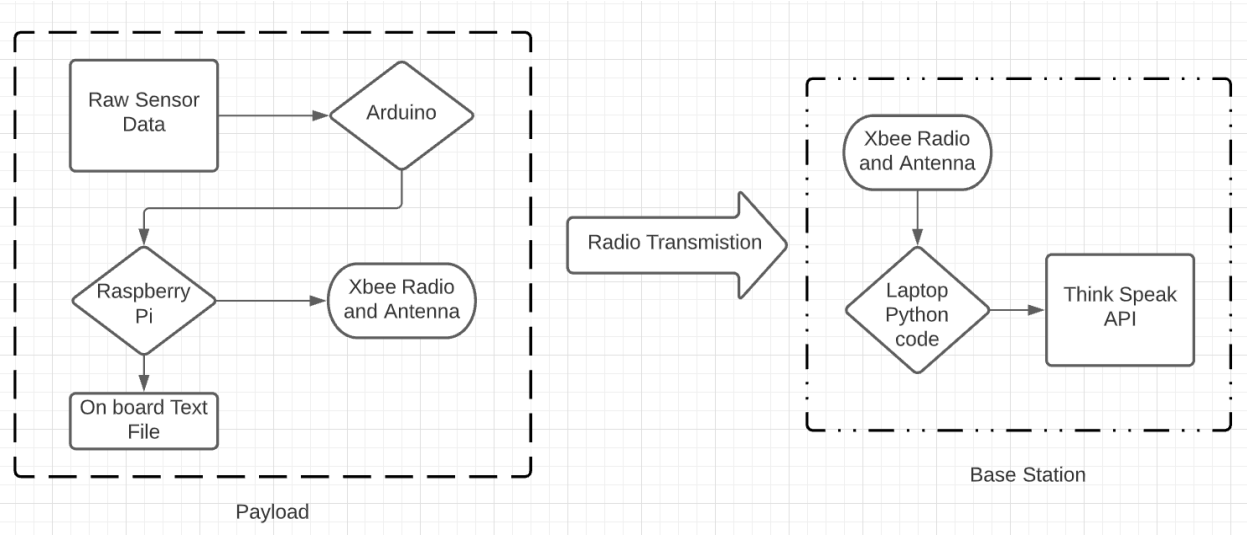

Figure 9: Diagram of Data Transmission from the Payload to the Base Station

Our Payload code creates three log files that we saved on board the Raspberry Pi. One stores all the Arduino data as it comes into the Pi. This was done to keep a record of the data

coming from the Arduino and not just what was transmitted and to diagnose any failures coming from the Arduino. The next text file stores all the sensor data from the sensors connected to the Pi. This file is very similar to the Arduino sensor file but was kept separate. The last text file stores all the data that was transmitted by the Pi. This will help keep a record of what exactly was sent to the radio. All three files are time-stamped using the Raspberry Pi's internal clock.

3.5 Flight Logistics

Table 2: List of Pre- and Post-launch Objectives per launch attempt

Pre-Launch	**Post-Launch**
1. Weigh the payload 2. Test communications system and sensors by using distance, time, and temperature tests 3. Simulate HAB flight path 4. Locate and confirm a suitable base station site 5. Retrieve the necessary amount of helium for the HAB 6. Assemble or acquire a radar reflector 7. Select launch site and date and call ahead to reserve the space 8. Activate Spot Tracker Account 9. Send completed HIBAL form and simulated flight paths to Jerry.CTR.Rutherford@faa.gov or current FAA contact a week before launch 10. Comply with instructions given by the FAA and call appropriate personnel at the discretion of the FAA. Previous contact was Jerry Rutherford (Cell:470-925-8277) 11. Test payload electronics a final time prior to launch, especially the SPOT tracker 12. Inflate the balloon and seal it 13. Ensure the balloon is securely fastened or otherwise tied down during inflation	1. Track payload's location using the Spot Tracker app 2. Let Base Station Team know the balloon is in the air 3. Base Station team starts monitoring for a signal 4. Base Station teams stays at the location until data no longer comes in 5. Launch team and Base Station team drives to the approximate landing site 6. Monitor the location of the payload on the Spot Tracker app to see burst point and landing 7. Wait for zero movements between positional updates to confirm landing location 8. Retrieve payload

Table 2: List of Pre- and Post-launch Objectives per launch attempt

Chapter 4. Results

4.1 November Launch

Our first planned launch of our balloon and payload was scheduled for November 21st, 2021. When choosing our launch site, we decided to stick with the historical precedent of launching from Lincoln Park in Albany, NY. However, when we got in contact with Albany's Parks and Recreations Department, they informed us that the park would be closed due to construction. Therefore, we located another park in Albany, Beverwyck Park, which was located about one mile to the North of Lincoln Park.

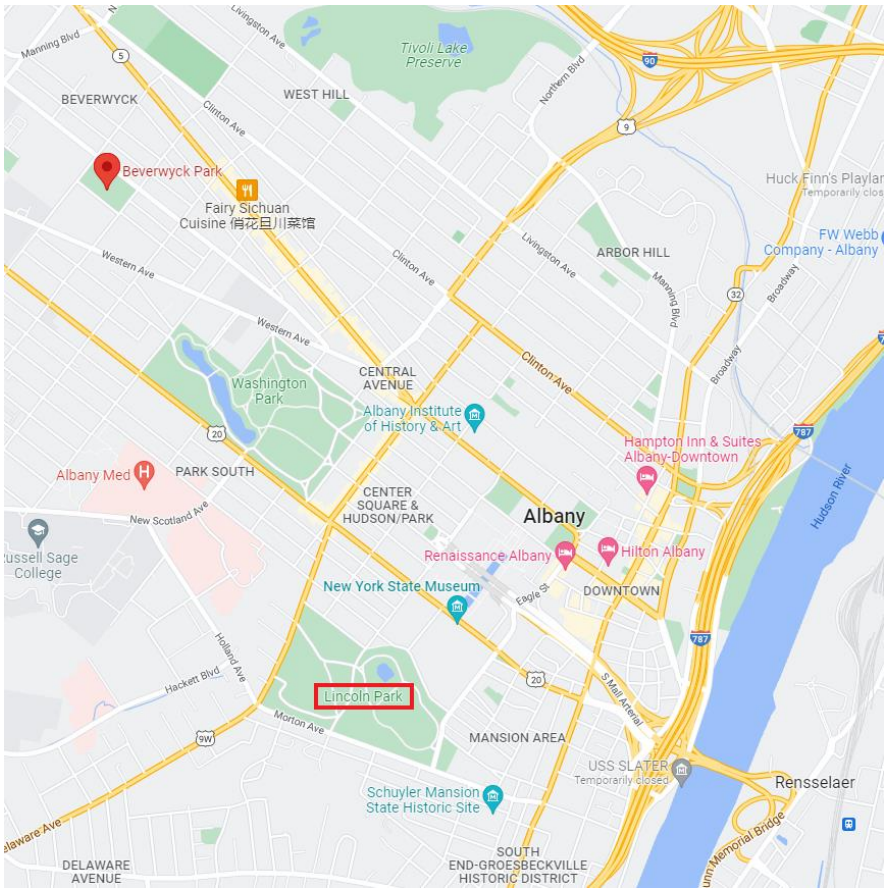

Figure 10: Map of Albany showing both Lincoln Park and Beverwyck Park

For our base station location, we decided to go with Petersburg Pass in Petersburg, NY. Petersburg Pass is an outdoors recreational area, located on a mountain pass in the Berkshire Mountains. From this pass, we have clear, unobstructed lines of sight on both the East and West

sides. Our sight lines to the North and South are obstructed by mountains so we would only be able to make a connection to the payload if the payload was significantly high enough to rise above the ridgeline. We thought that this would not be a significant issue since our flight simulations had the balloon traveling nearly directly overhead of the pass. We also did not need to contact any authorities about using Petersburg Pass since it is simply a dirt parking lot on top of the mountain pass.

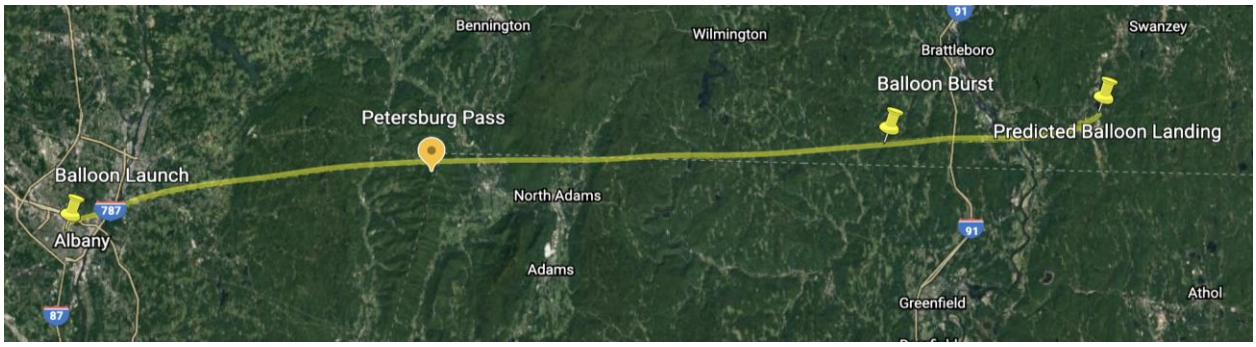

Figure 11: The predicted flight path of our balloon on Nov. 21st with Petersburg Pass marked

On the day of the launch, we successfully arrived at the launch and base station sites in our respective teams. As the launch team was preparing the balloon and payload for launch, the balloon was accidentally let go while it was being filled. Since we did not have a spare balloon or enough helium left to fill another one, the launch was a failure and we decided to plan another launch for the near future.

Following the failure of this launch, we held meetings to discuss what went wrong and what could be improved for the next launch. In these meetings, we discussed potential solutions to the failure of the first launch. This included an improved tie off mechanism and tethering the balloon before the intended release; these methods were then used in the later launches.

4.2 December Launch

Due to the failure of the first launch, we decided to do another launch two weeks after the original date, making our second launch date December 5th. For this launch date, we were better prepared, and we made a few changes. Those changes were changing our launch site, our base

station site, and improved payload code. When we went to do our second launch attempt, we started by running the flight simulations to better understand how the weather and wind patterns would change over the two weeks. We found that the winds would be much stronger in the Eastward direction, causing us to worry about the balloon ending up too far East and landing in the ocean.

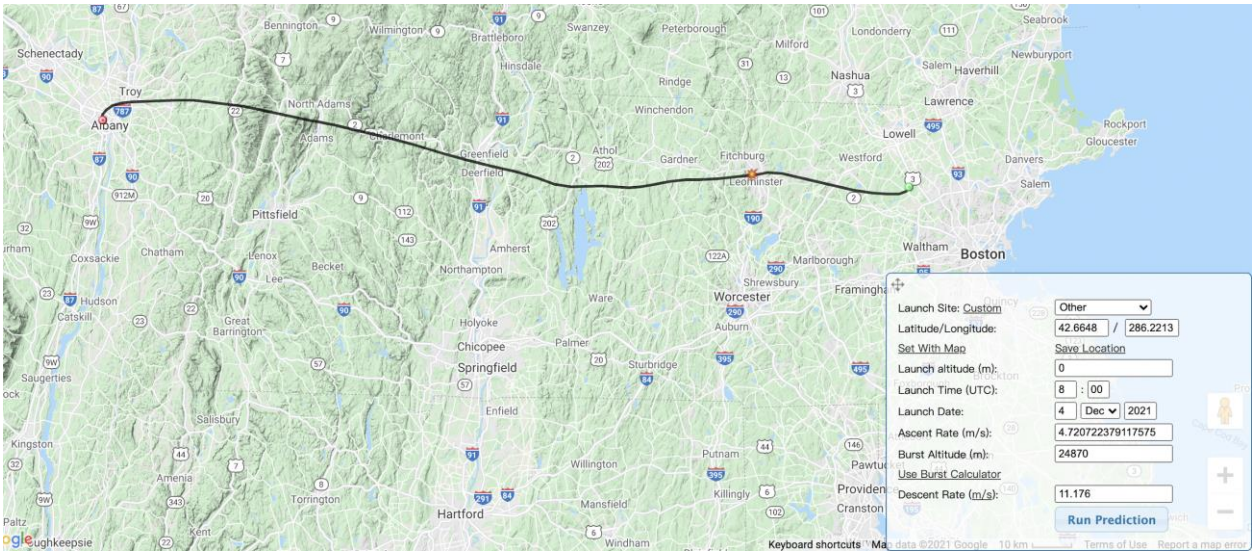

Figure 12: The predicted flight path of our balloon on Dec. 5th when launched from Albany

Due to these potentially troubling predictions, we thought it was best to move the launch site to the West, further inland, to hopefully have the balloon land on the land. However, we could not travel too far West because it had to be close enough to Worcester so we could get there in about 2-3 hours of driving so that we could launch and drive back without much difficulty. With this in mind, we chose a new launch site, Iorio Park in Cobleskill, NY. We were able to call the town and get permission to use the park with short notice, so we decided to choose this as our launch site. Using our new launch site in our flight simulator, we were able to determine a new base station site. We chose to use a new base station site at the Mt. Greylock Visitor Center. We moved the base station because from our previous launch, we saw that Petersburg Pass does not have good views to its North or South and therefore would likely miss the balloon because the newly projected flight path had the balloon traveling significantly farther

North or South of the pass. The Mt. Greylock Visitor Center has a large parking lot at the edge of an open field that looks West and North. The base station team decided, upon arriving, that the lower parking lot at the Visitor Center's maintenance building had a better view North, which was where the balloon was projected to travel.

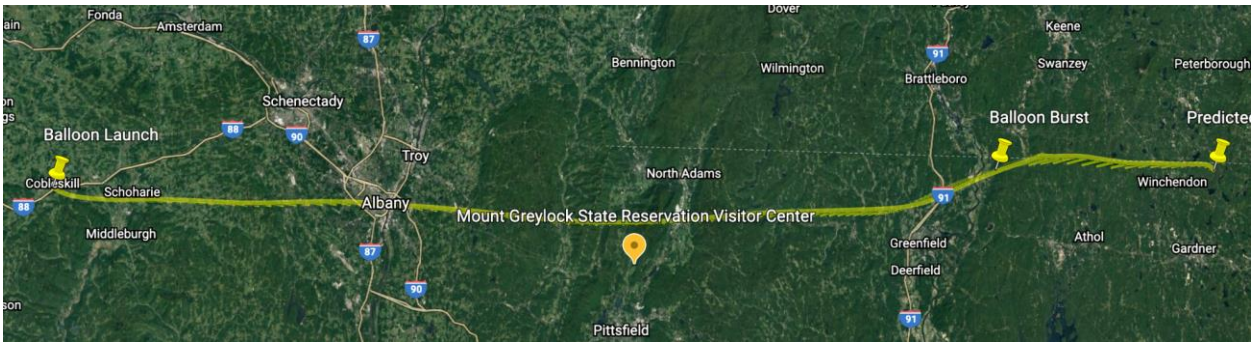

Figure 13: The predicted flight path of our balloon on Dec. 5th when launched from Cobleskill

Despite our precautions in moving the launch site our balloon still traveled further than intended. Throughout the flight we were unable to get GPS data from our onboard GPS tracker and got our first GPS signal when the payload landed. In Figure 14 below the payload landing site is marked by a red dot. Due to the payload landing in the Atlantic Ocean we were unable to retrieve the payload and the sensor data contained in it. We were however able to get data transmitted down from the payload during the middle of the flight. This data can be seen in Figures 15-21 below.

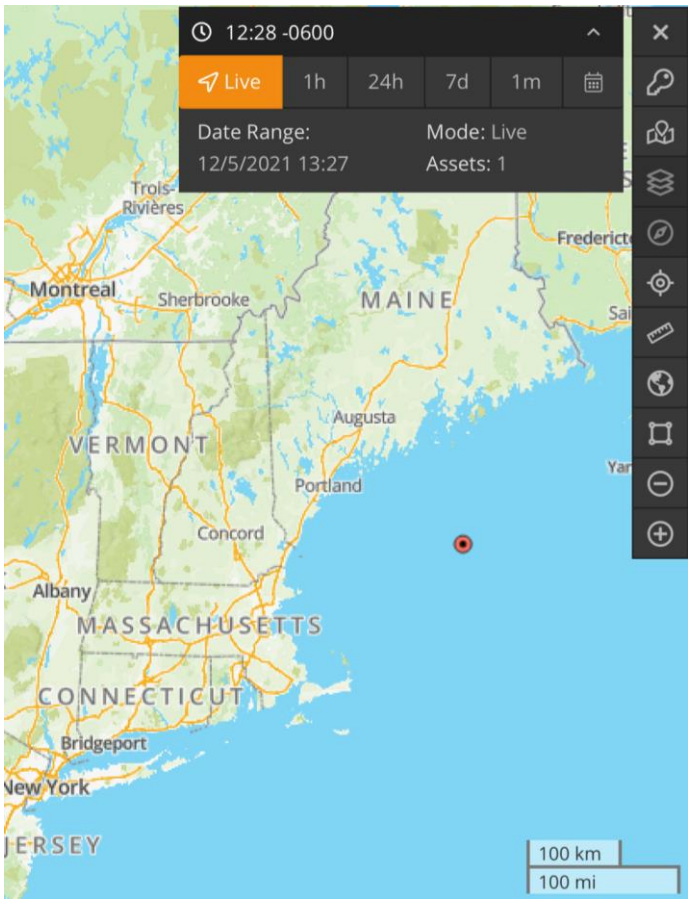

Figure 14: December Payload Landing Site

We got a total of 9 minutes of data transmitted down from the payload to our base station site. We got this data from 9:41:25 - 9:48:51 AM and 10:02:38 - 10:04:11 AM and at 6869 - 7955m and 9301 - 9630m. The gap in receiving data was due to the antenna not being pointed directly at the payload because of the failure of our GPS tracker. We got altitude, temperature, pressure, CO_2, Ozone, NO, and UV data. All but the NO and UV data lines up with the expected values for that altitude.

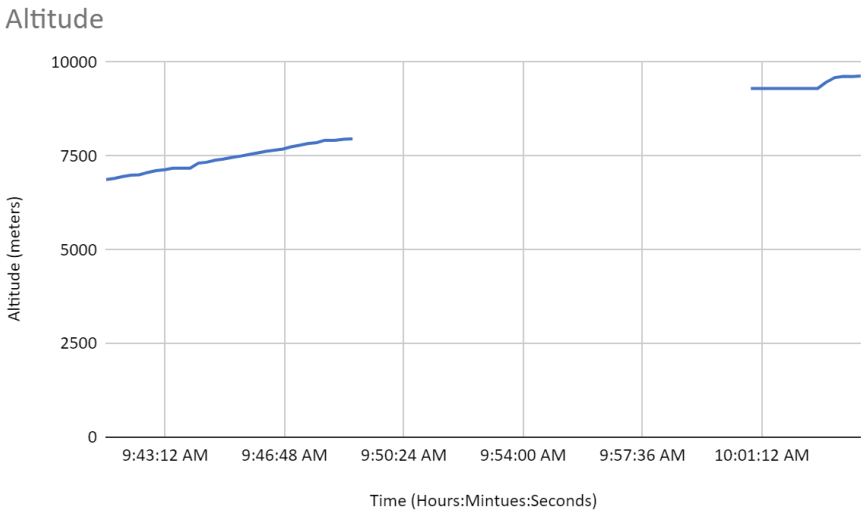

Figure 15: Transmitted Altitude Data

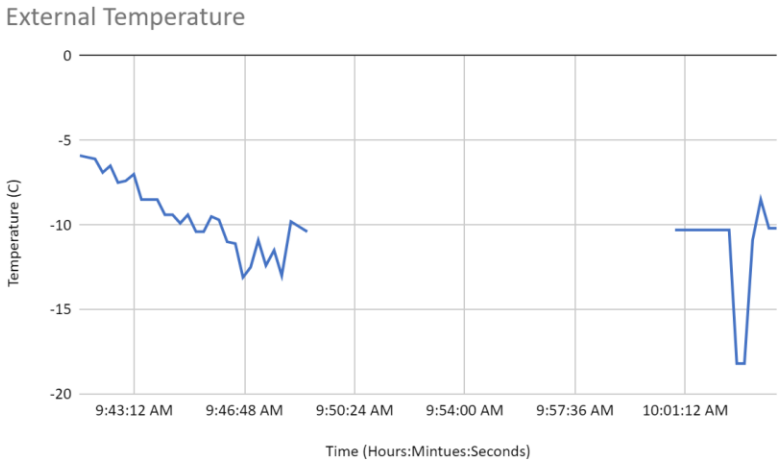

Figure 16: Transmitted External Temperature Data

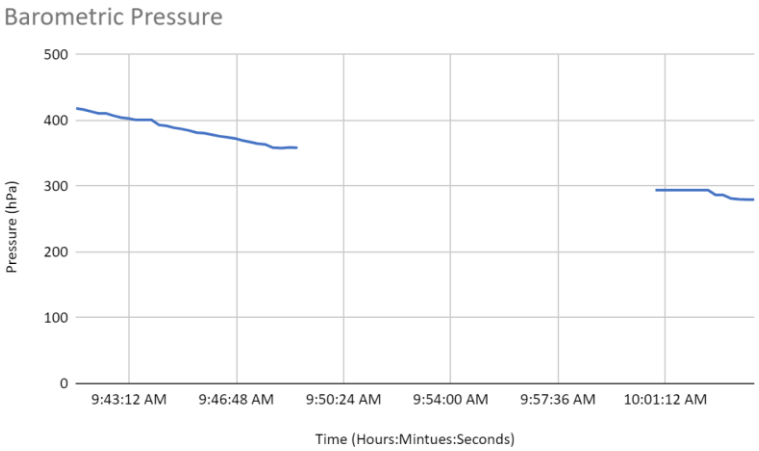

Figure 17: Transmitted Barometric Pressure Data

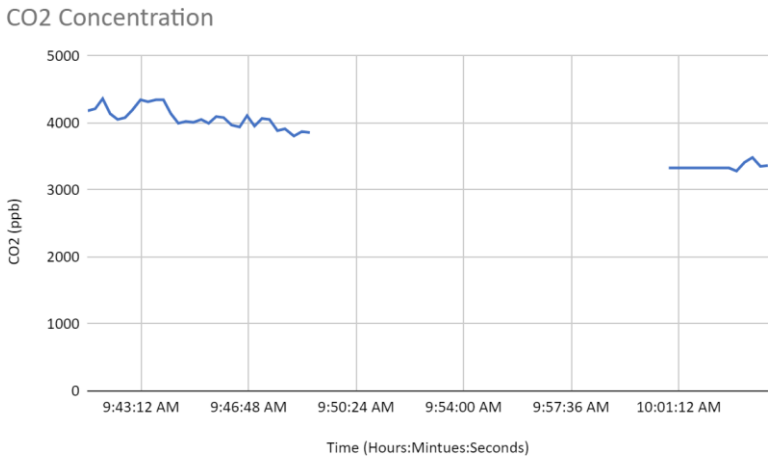

Figure 18: Transmitted CO2 Data

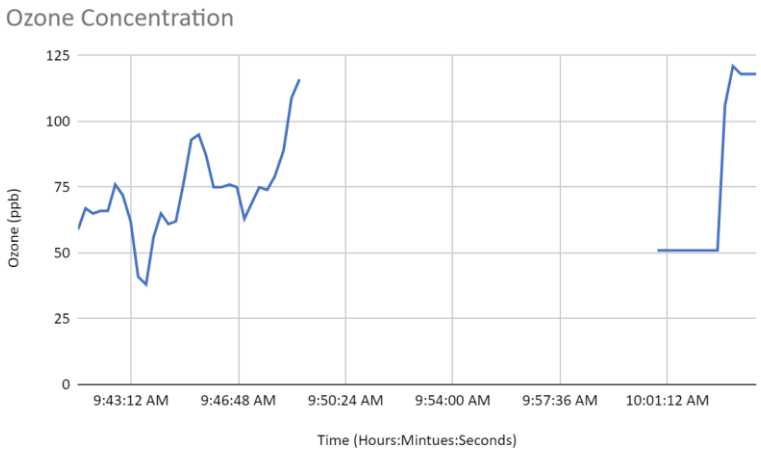

Figure 19: Transmitted Ozone Data

The NO data remained constant, and this is possibly due to a sensor failure which can be seen in Figure 20 below. We were unable to retrieve the payload, we could not perform further tests on the sensor.

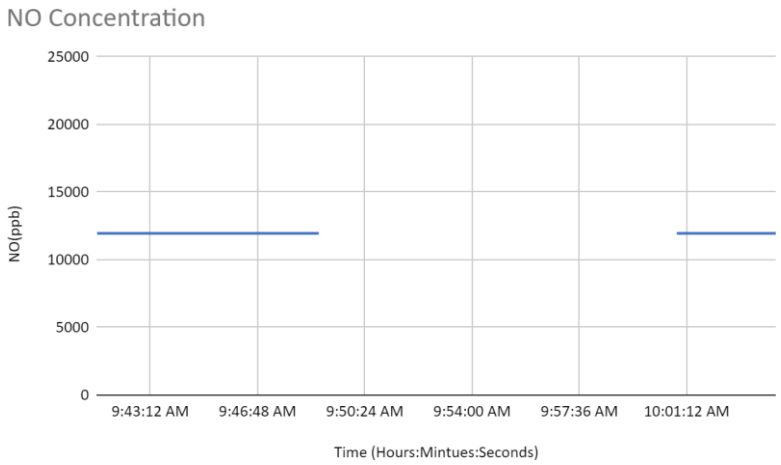

Figure 20: Transmitted NO Data

The UV sensor gave very inconsistent and unreadable values. On the ground the UV sensor tests were consistent but weren't as sensitive as hoped. The UV index was intended to go from 1-10 indicating the UV levels. During the actual flight the only values we got from the sensor were 0 and 1 although this may be due to the small data window.

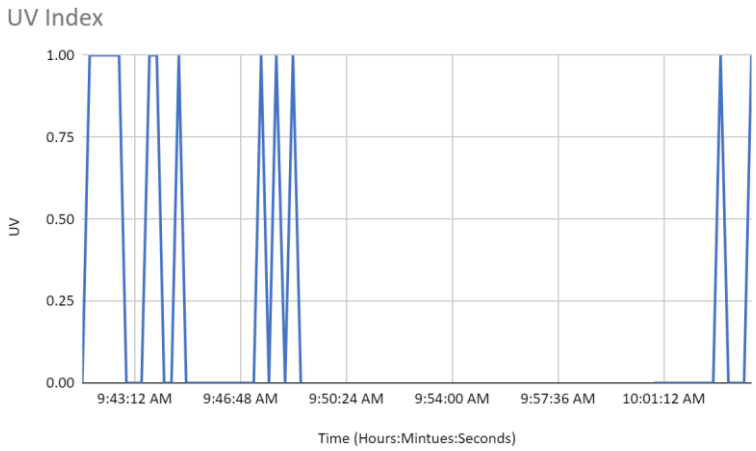

Figure 21: Transmitted UV Index Value

For our spring launch we added an additional GPS sensor to give us another fail safe in case we do not get a signal from our SPOT tracker onboard the payload as happened in this launch. We also edited and rewrote the payload to include error conditions in case of sensor failure.

4.3 March Ground Test

For the March launch, our team performed a ground test on Mount Wachusett. This decision was made due to a helium shortage that delayed us and inclement weather on the day of the planned launch, March 19th, 2022. For this ground test, we decided to simulate the conditions of a launch and flight as much as possible. To do this, we set up a base station at the bottom of Mount Wachusett And we had one team hike to the top of the mountain and back down again with the transmitting payload. We designed this test to verify our payload's ability to collect sensor data, the accuracy of those sensors, and our radio transmission system.

The results of this test were sufficient to prove that our system works as we intended it to. We were able to collect data from all our sensors with no errors in their operation. While hiking our payload up a mountain does not subject it to the same conditions it would experience in the upper atmosphere, this test did prove that our payload would likely work in those conditions because the only potential failure point remaining for the system would be a physical disconnection of the payload electronics caused by a shock from turbulent winds, which we also have protected against. Looking at Figures 22-27, we can see the data that we collected at both the base station and aboard the payload. In this data, we can see how our base station did not perfectly recover the same data that was recorded on the payload. This is most apparent in the time scale of all our base station data because each packet sent by the payload included a time stamp and we did not recover all these timestamps, nor did we have enough data to associate each piece of data to a timestamp in a 1:1 ratio. That means that parts of a packet or even whole packets were dropped between the payload and base station. This makes the base station data somewhat unreliable since we cannot be sure of the timing accuracy of the data. To plot our data, we had to fill in these missing values using previously recorded ones so that we had a 1:1 ratio with our time indices. We did this by checking the length of each data list against the time list

each time a new piece of data was read in and then repeating the previous data value until the difference in the time and data list was one, and then we add in the new piece data, making the lists equal. However, since we were sending a packet every second from the payload, we are still able to see the trends and overall changes in the graphs, so we are assuming that this timing error is small but not insignificant. This set of data is not actual climate data that we would analyze for GHGs, so we do not care about the timing error in this test.

From the ground test, we got CO2 data, but it was not a complete data set which can be seen in Figure 22. Our CO2 sensor does not read below 400 ppm accurately and we only got a few readings above this mark. This was not an issue with our December launch data and is most likely an issue with our ground test.

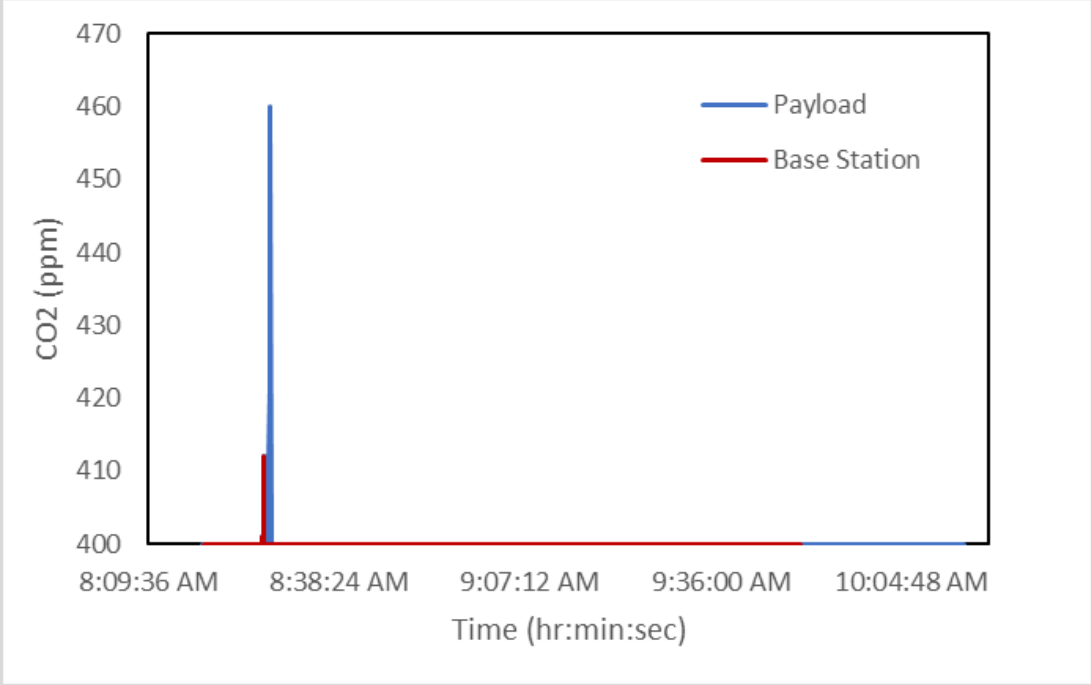

Figure 22: CO2 Data from the Base Station and Payload

The payload also recorded the Ozone and UV index values which can be seen in Figure 23 and Figure 24 below which show values consistent with what we were expecting. These graphs display a clear picture of when we were receiving data on the base station versus when signal was lost.

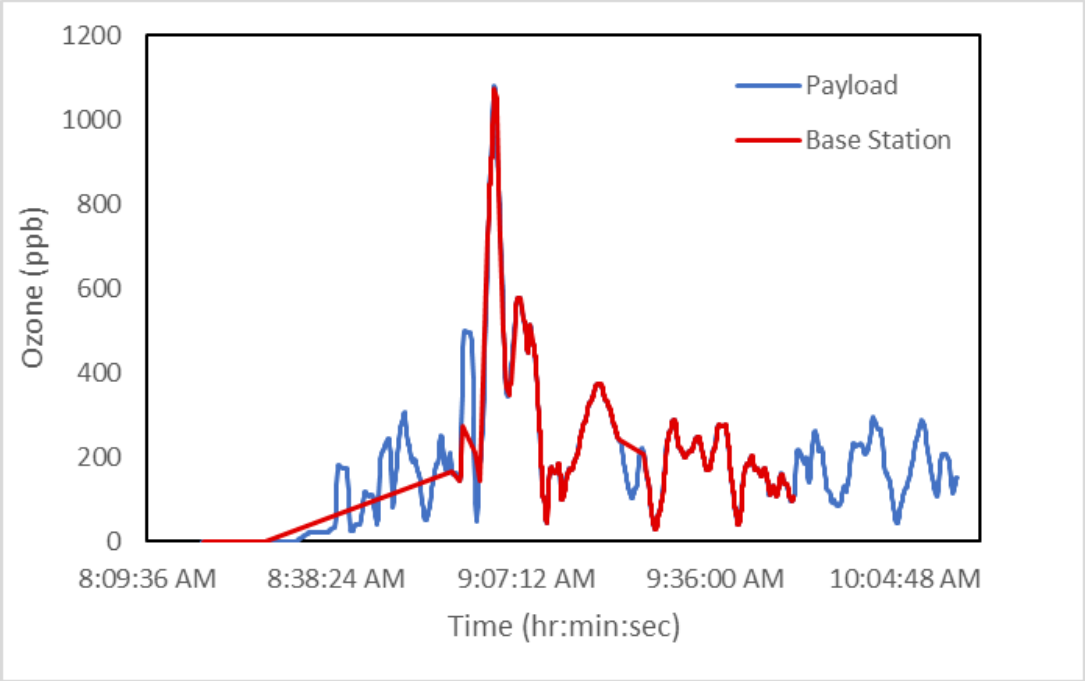

Figure 23: Ozone Data from the Base Station and Payload

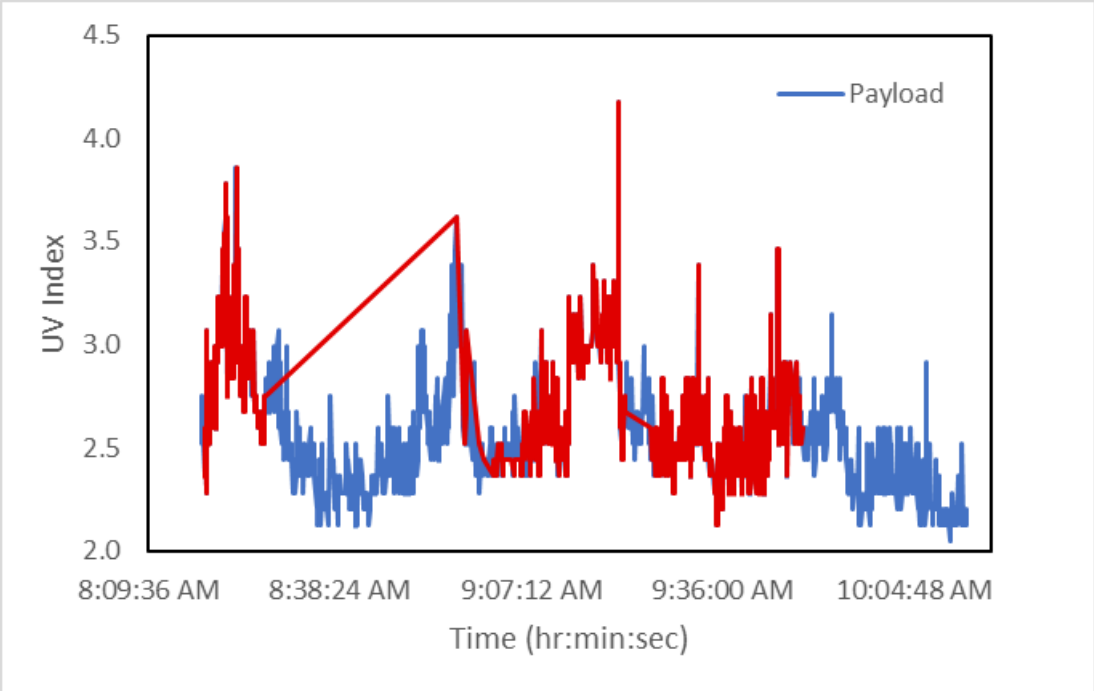

Figure 24: UV Data the Base Station and Payload

The pressure data from the payload is shown in Figure 25 below. On-board the payload there were two pressure sensors, a barometric pressure sensor and a low pressure sensor. This data is as expected because we climbed up the mountain, which causes the pressure values to drop.

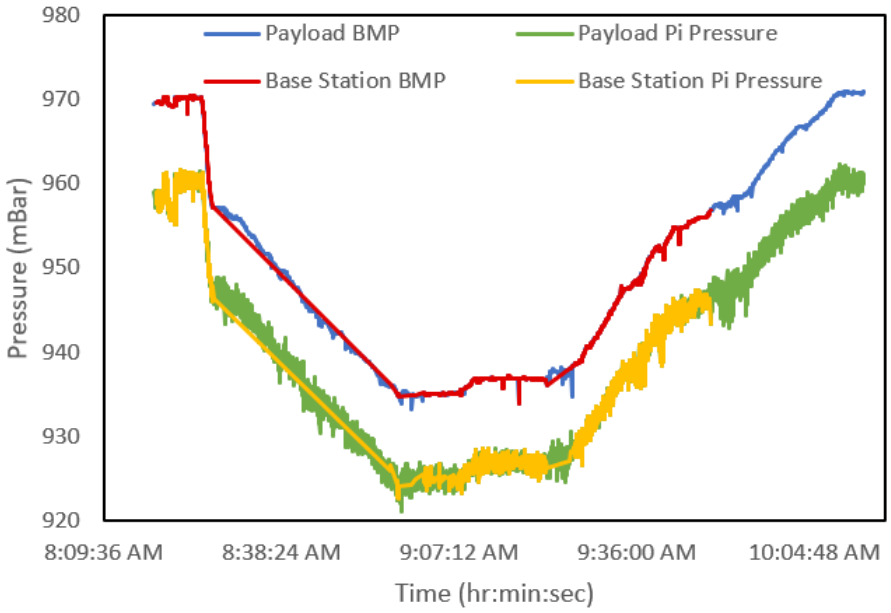

Figure 25: Barometric Pressure and Pressure Readings from the Base Station and Payload

Similar to the pressure sensor the payload also had two temperature sensors. The BMP temperature was external to the payload and the one-wire sensor was internal. Figure 26 below shows this difference with the one-wire temperature signal having much less fluctuation.

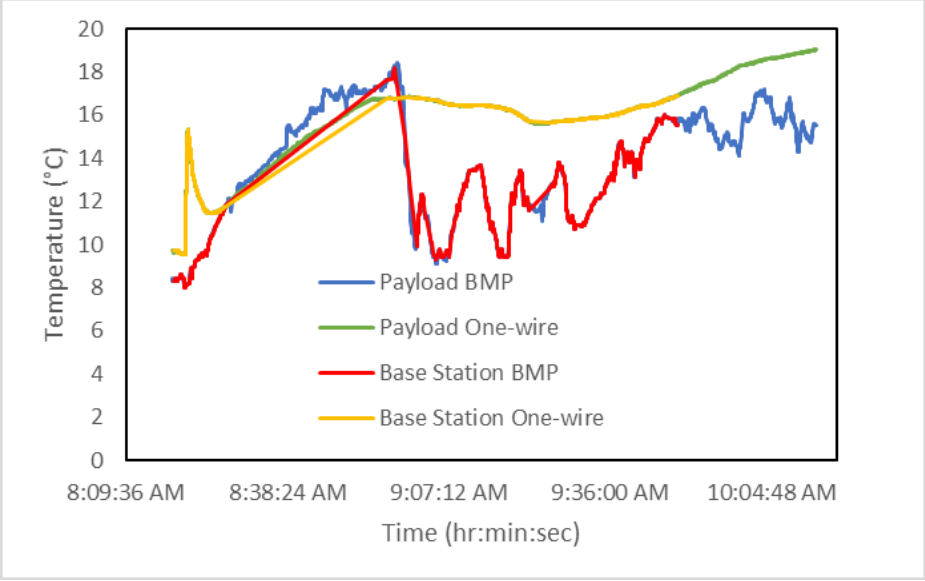

Figure 26: Temperature Values from the Base Station and Payload

The payload had two main ways of determining its altitude. The first method is calculated using the barometric pressure and temperature. The next method is using the GPS satellites. The calculated and GPS altitudes are very similar but the calculated one was slightly less accurate.

Although they both go up as the team hiked the mountain. These two altitude measurements can be seen in Figure 27.

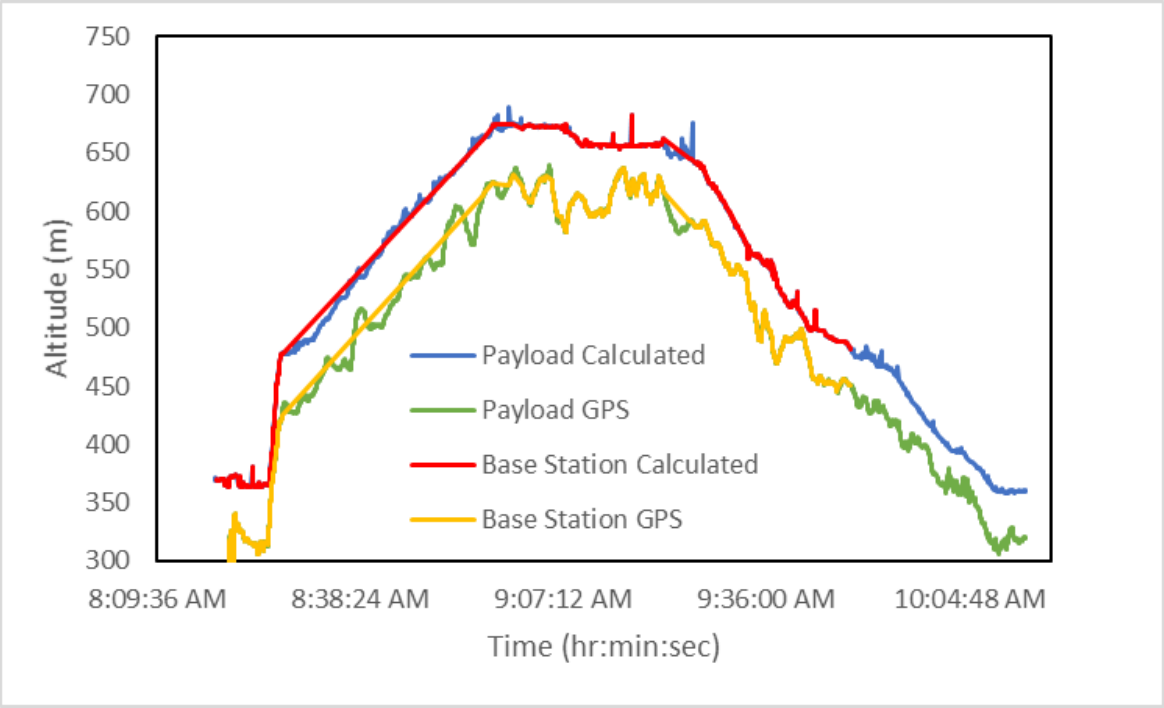

Figure 27: Calculated Altitude Values Versus GPS Altitude

Throughout the March test, the Payload GPS sensor collected data as we hiked the mountain, while also transmitting to the base station. This data shows where we lost and regained signal and gives indication as to why. Figure 28 shows the GPS data received at the base station. It has gaps in data when the payload was on the other side of the mountain or being obscured by trees. Which simulates what we would have received inflight. Figure 29 shows all the GPS data the payload actually recorded and gives a better indication of the path it took. Figure 30 and Figure 31 compare our GPS data and the path we actually hiked. We can see how accurate our GPS data we received and recorded on-board was to the actual position of the payload. Considering the relatively small scale of the ground test, the GPS sensor was extremely accurate.

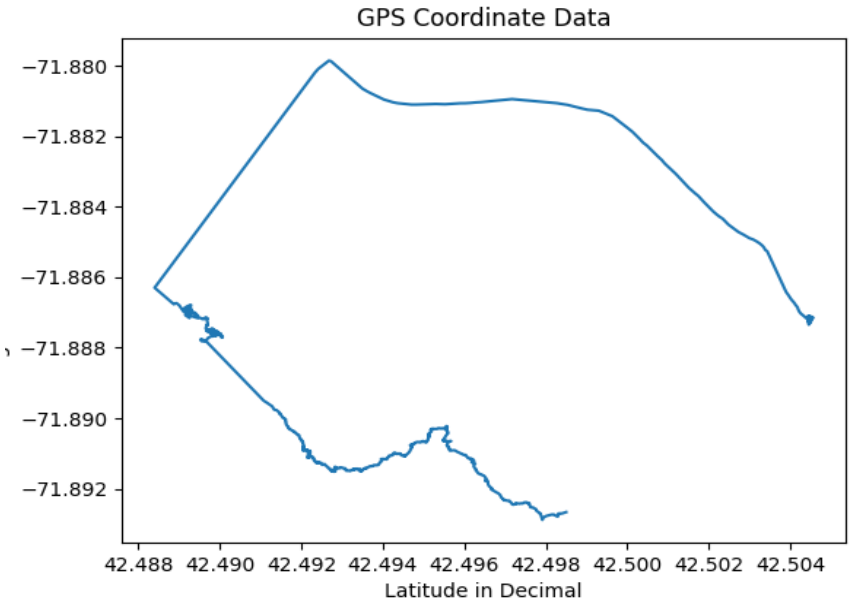

Figure 28: Base Station GPS data

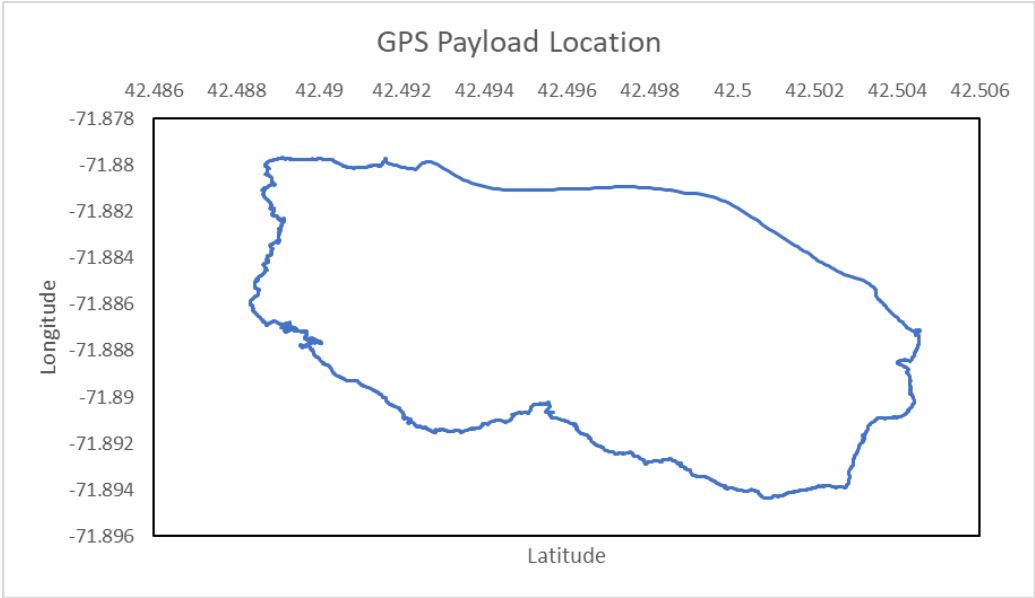

Figure 29: Payload GPS data

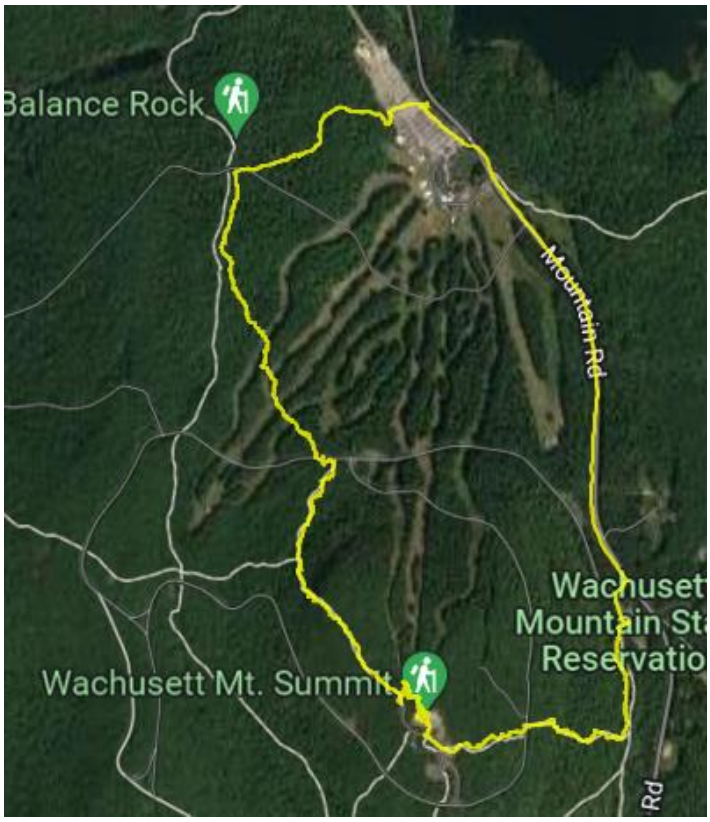

Figure 30: Payload GPS Data Overlaid on a Trail Map

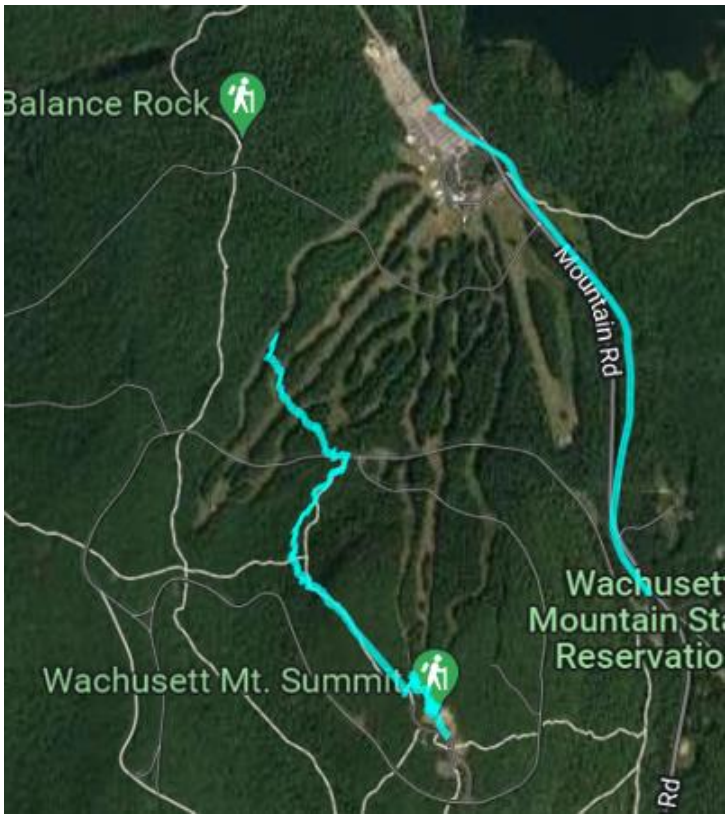

Figure 31: Base Station Received GPS Data Overlaid on a Trail Map

5. Conclusion & Recommendations

5.1 Achievement of Initial Scope

At the conclusion of this project, we realized our goals of real-time data transmission and increased reliability in the collection of data. Our Dec. 5th launch proved our real-time data transmission system is functional, and even works through some tree cover as seen in the Mar. 20th ground test. While we did not gather direct evidence that our data would be reliably collected in flight on Dec. 5th due to our lost payload, the Mar. 20th ground test did prove that our sensors could collect data for about two hours without issue; the data was all saved and recovered at the end of the test. We also did drop the payload a few times during the ground test to evaluate its resistance to shock and found that those shock tests did not affect the payload's operation at all.

In the future, the project can be further developed to make the wireless communications system more robust and add additional features to the payload. At the end of the project, we were unable to get the ESP32 and the motor operational. We ran out of time to complete this task due to the large amount of logistical work that is necessary for each launch. Before we had to cancel our last launch, we were unable to find helium easily due to a helium shortage in the area in addition to our previous supplier of helium, Michael's Party Rentals, deciding to suddenly stop selling helium. Luckily, we were able to find helium in time for our launch from another supplier, Liberty Supply Inc. This caused us to lose the time in which we would have worked on the motor to be spent instead on securing the logistics for our last launch. Moving on, we did physically implement the motor system in the base station so future teams only have to program a microcontroller in order to make the system operational. Another note on this issue is that the next team will not have the ESP32 because it belongs to one of the current team members and they will be taking it back at the end of the project. The base station can be improved by finding

a larger antenna that has more signal gain in addition to operating at lower frequencies. We discussed the possibility of getting an amateur HAM TV license as it would allow us to send information at around 440 MHz which would give our wireless system additional range but, potentially at the cost of a lower data rate. Another way that the base station could be improved is by adding in an additional motor to move the antenna in the vertical direction. This new motor would allow the antenna to rotate vertically which would result in its directional pointing being more accurate, which will in turn increase the received signal strength from the payload. This will have the ultimate effect of increasing the transmission range of our system in addition to increasing the amount of time we can receive data at the base station.

The improvements made to the payload helped to ensure that we achieved our major project goals. We added new sensors that ensured that our data was accurate, and we had the best chance to get good data from our launches. We also improved on wire connections and payload insulation. Future developments to the payload would involve finding more sensors for the NO and Methane gas measurements we were unable to do this year. We also suggest using the GPS sensor and some nichrome wire on board the Payload to terminate the flight if the balloon either goes too high or too far east. This would help to prevent landing where the payload cannot be retrieved.

During this MQP, our team was able to decide on many launch and base station sites based on the simulation data for that particular day. We learned that the launch site location requires more permissions and planning than the base station sites. The launch sites should only be changed in cases of drastic simulation changes that would make a launch site no longer viable as happened in our December launch. Since the base station sites must be roughly in the middle of the flight path, that location changes quite often. Luckily, these sites require less permissions to use and take less time to set up.

One of the major obstacles we encountered during our last launch was a helium shortage. This made it so that most of the places we tried to buy helium were not able to sell it to us. This in combination with some of the simulation sites going down emphasized the need to both plan ahead but be flexible with our execution of this project.

References

Sparkfun. Temperature Sensor - High Temperature, Waterproof (DS18B20). Retrieved March 24, 2021, from https://www.sparkfun.com/products/18367

Sparkfun. SparkFun Pressure Sensor Breakout - MS5803-14BA. Retrieved March 24, 2021, from https://www.sparkfun.com/products/12909

Digikey. YIC93030PGMFGG-U8-N. Retrieved March 24, 2022, from https://www.digikey.com/en/products/detail/yic/YIC93030PGMFGG-U8-N/13980841?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campaign=Smart%20Shopping_Product_Zombie%20SKUS&utm_term=&utm_content=General&gclid=CjwKCAjwrfCRBhAXEiwAnkmKmasBMknsYAFjc2cRlB_Lh8WAvMw8mpoKzO4IJDHcwmPrGsA1JlJQcRoCbtUQAvD_BwE

DFRobot. Gravity: I2C Ozone Sensor (0-10ppm). Retrieved March 24, 2022, from https://www.dfrobot.com/product-2005.html

Adafruit. Adafruit MiCS5524 CO, Alcohol and VOC Gas Sensor Breakout. Retrieved March 24, 2022, from https://www.adafruit.com/product/3199

Adafruit. Analog UV Light Sensor Breakout - GUVA-S12SD. Retrieved March 24, 2022, from, https://www.adafruit.com/product/1918

DFRobot. Gravity: Analog CO2 Gas Sensor For Arduino (MG-811 Sensor). Retrieved March 24, 2022, from, https://www.dfrobot.com/product-1023.html

Amazon. JBtek BMP180 Barometric Pressure, Temperature and Altitude Sensor. Retrieved March 24, 2022, from, https://www.amazon.com/JBtek-Barometric-Pressure-Temperature-Altitude/dp/B00UUS12PO

Adafruit. Raspberry Pi 4 Model B - 2 GB RAM. Retrieved March 24, 2022, from, https://www.adafruit.com/product/4292?gclid=CjwKCAiA-9uNBhBTEiwAN3IlNGw9M90vNBuXoC2hNfUXMC3PoHyPCoCJZnKTasskjoaCZ9b6ew48DBoCCeMQAvD_BwE

Amazon. Arduino A000057 Leonardo with Headers. Retrieved March 24, 2022, from https://www.amazon.com/Arduino-A000057-Leonardo-with-Headers/dp/B008FZJC74/ref=asc_df_B008FZJC74/?tag=hyprod-20&linkCode=df0&hvadid=312069228616&hvpos=&hvnetw=g&hvrand=52774086906638461

18&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9001847&hvtargid=pla-492686100233&psc=1

Saved by Spot. Spot Gen 4. Retrieved March 24, 2022, from, https://www.findmespot.com/en-us/products-services/spot-gen4

Mouser. DIGI XBP9B-DMST-002. Retrieved March 24, 2022, from, https://www.mouser.com/ProductDetail/DIGI/XBP9B-DMST-002?qs=%2Fsr0sUXBX%2Foiu%252B5b6BKNJQ%3D%3D

Digi-key. 76000956. Retrieved March 24, 2022, from, https://www.digikey.com/en/products/detail/digi/76000956/5877530

Amazon. Waterproof 4dBi LoRa Gateway Antenna 900-930MHZ Omni-Directional Pole/Wall Mount Glass Fiber LoRaWan Antenna for Helium Hotspot HNT Miner Mining Ultra Distance Transmission 915MHz, Eifagur. Retrieved March 24, 2022, from, https://www.amazon.com/dp/B093DG1B6D?psc=1&ref=ppx_yo2_dt_b_product_details

Amazon. STEPPERONLINE Low Current Nema 23 CNC Stepper Motor 1.8A 340oz.in/2.4Nm CNC Mill Lathe Router. Retrieved March 24, 2022, from, https://www.amazon.com/STEPPERONLINE-Current-Stepper-340oz-Router/dp/B00PNEPKH6?ref_=ast_sto_dp

Amazon. STEPPERONLINE CNC Stepper Motor Driver 1.0-4.2A 20-50VDC 1/128 Micro-Step Resolutions for Nema 17 and 23 Stepper Motor. Retrieved March 24, 2022, from, https://www.amazon.com/STEPPERONLINE-1-0-4-2A-20-50VDC-Micro-step-Resolutions/dp/B06Y5VPSFN?ref_=ast_sto_dp

Amazon. Bolton Technical Long Ranger Antenna | 2021 Parabolic - Over 20 Miles Range | All Cell Bands: 5G, 4G, LTE | WiFi 2.4/5 GHz WiFi 6 | High Gain Cellular/WiFi Antenna up to +28 dB | All Carriers. Retrieved March 24, 2022, from, https://www.amazon.com/Bolton-Technical-Ranger-Antenna-Parabolic/dp/B08FBN5LJ7/ref=psdc_3015438011_t4_B01FV1DD9C#customerReviews

EPA, Sources of Greenhouse Gas Emissions. Retrieved March 25, 2022, from, Sources of Greenhouse Gas Emissions | US EPA

EPA, History of Air Pollution. Retrieved March 25, 2022, from, History of Air Pollution | US EPA

Ru, Hanyang, Lee, Jonney, Jiang, Jiayi. (2020, April 6) High-Altitude Balloon. Retrieved March 25, 2022, from, https://digital.wpi.edu/concern/student_works/0g354h99d?locale=en

Falsarella Guerreiro, Lucas, Je, Tae Hyun, Gross, Leo, Langlois, Zachary. (2021, April 6) High Altitude Weather Balloon Launch for Measuring Environmental Pollution. Retrieved March 25, 2022, from, https://digital.wpi.edu/concern/student_works/js956j577?locale=en

Wang, Zhanzhao, Min Huang, Lulu Qian, Baowei Zhao, and Guangming Wang. (2020, April 7) High-Altitude Balloon-Based Sensor System Design and Implementation. Retrieved October 19, 2020, from High-Altitude Balloon-Based Sensor System Design and Implementation - PMC (nih.gov)

Appendix A: GitHub Repository

Link to the GitHub with all of the project code

https://github.com/HAB-MQP-WPI/HAB-III-Code

Appendix B: Live Video from our Dec. 5th Launch

Livestream of our December Launch

HAB3.0

Appendix C: FAA HIBAL Form

HIBAL WORKSHEET

(High Altitude Unmanned Free Balloon (HIBAL)

Answer all questions and send via email to 9-ATO-ESA-OSG-AirspaceWaiver@faa.gov at least 14 days before launch.

ORGANIZATION/CONTACT INFORMATION: Today's Date: Select a date.

Organization/Company:
POC Name:
Street Address:
City, State Zip:
Phone:
Email:

Answer the following questions regarding your balloon, payload and suspension device.

1. Does the balloon have more than one payload? Yes ☐ No ☐ If yes, how many? Select
2. Is the individual weight of any payload more than 4 pounds and has a weight/size ratio of more than three ounces per square inch on any surface of the package, determined by dividing the total weight in ounces of the payload package by the area in square inches of its smallest surface? Yes ☐ No ☐ (If your answer is no, proceed to #9 and continue)
3. Any payload greater than 6 pounds? Yes ☐ No ☐
4. Is the total weight of 2 or more payloads greater than 12 pounds? Yes ☐ No ☐
5. Does the payload suspension device require more than 50 pounds of force to separate from the balloon? Yes ☐ No ☐
6. Is the balloon equipped with at least two independent payload cut-down systems? Yes ☐ No ☐
7. Is the balloon equipped with at least two methods, systems, devices, or combinations thereof, that function independently of each other that are employed for terminating the flight of the balloon envelope? Yes ☐ No ☐
8. Is the balloon equipped with a radar reflective device(s) or material that will present an echo to surface radar operating in the 200 MHz to 2700 MHz frequency range? Yes ☐ No ☐
9. Is the balloon equipped with any of the following?: Tracking Device ☐ Transponder ☐ Camera ☐

If you answered yes to question 2, 3, 4, or 5 you may require a Certificate of Waiver or Authorization.
Please review Title 14 CFR Part 101, Subpart A and D, complete this form and the attached FAA Form 7711-2.

BALLOON/LAUNCH INFORMATION:

Proposed Launch Date: March 19, 2022 Launch Time: (24 HOUR) Select
Launch Coordinates: (*Lat/Long) N/ W
Flight Duration:
Ascent Rate: m/s
Burst Altitude: m.
Descent Rate: m/s
Estimated Landing Coordinates: (*Lat/Long) N/ W
On-Site Operator:
On-Site Contact Number:
Tracking Information:
**Coordinates may be entered as Degrees/Minutes/Seconds or Decimal Degrees*

Other Information: Balloon and payload description (size, colors etc.) alternate launch dates, etc.

Provide a flight trajectory from http://predict.habhub.org (or other suitable site) and attach additional sheets if necessary

If you are in doubt whether you require a waiver, please send this form and any questions to:
9-ATO-ESA-OSG-AirspaceWaiver@faa.gov