University of Minnesota – Twin Cities and MN Space Grant Consortium

AEM 1301 Freshman Seminar: Fall 2021
Introduction to Spaceflight
(with Stratospheric Ballooning Project)

Team Project Documentation

Team C (Space Raccoons)

Written by:
Rory Conway, Ray Lyon, Ryan Levendusky, Tina Li, and Mel Pham

Report Submission Date: 12/03/2021
Revision C

Revision Log

Revision	Contents	Due Date
A	Conceptual Design	Friday, Oct. 15, 5 p.m.
B	Build/Testing Report	Friday, Nov. 5, 5 p.m.
C	Flight/Analysis Final Report	Friday, Dec. 3, 5 p.m.

Table of Contents

0.0 Team Project Documentation Writing/Build/Flight Day Assignments

Team Name Team C (Space Raccoons)

Introduction	Mel P
Mission Overview	Rory C
Payload Design	Mel P
Project Management	Tina L
Project Budgets	Ray L
Payload Photographs	Tina L
Test Plan and Results	Rory C + Ryan L
Expected Science Results	Tina L
Flight Day Narrative	Mel P
Results and Analysis	Ryan L + Ray L
Conclusions and Lessons Learned	Ryan L
References	Ray L
Program Listings	Ryan L

Oral Presentation Assignments

Conceptual Design Review (CDR)	Tina L, Mel P, Rory C
Flight Readiness Review (FRR)	Ryan L, Ray L

Payload Build Assignments

Overall team lead and ground-testing lead	Tina L
Payload box build	Mel P
PTERODACTYL basic sensor suite	Ryan L
Programmer lead for the PTERODACTYL	Tina L + Ray L
Camera experiment	Rory C
Neulog modules (including set-up)	Rory C
“Other” science experiment(s)	Ray L

Flight Day Assignments

Documentation/photographer	Tina L
Flight prediction specialist	Rory C
Payload handling/start-up specialist	Mel P
Comms telemetry data specialist	Ray L
Payload telemetry data specialist	Ryan L
Aprs tracking specialist	Ryan L

SPOT tracking specialist	Tina L

1.0 Introduction

Despite humanity's best efforts, the “Final Frontier" is still vastly unexplored. This may be accredited to a lack of suitable technology, but according to L. Paul Verhage[1], the main obstacle preventing the public from casual space exploration or travel is cost. Regardless of what one may believe, the fact remains that the infinite expanse of Outer Space has much to be discovered.

While “space” is defined as the area above the Kármán Line (the elevation at which conventional aircraft cannot effectively fly, which is an altitude of 62 miles), “near-space” is generally defined as the stratosphere[2]. Located 7 to 31 miles above the earth’s surface, the stratosphere contains about 20% of Earth’s atmosphere[3]. Because of its warmer temperature and fewer clouds, it makes for a smooth and calm elevation for many large commercial flights and private aircrafts[4].

In AEM 1301, our primary focus will be stratospheric ballooning, which is the use of stratospheric balloons (more commonly known as weather balloons) to deliver data-collecting payloads to the stratosphere. These modules can serve different purposes depending on what the scientist wants to explore. They may have temperature probes, light sensors, cameras, or even less common sensors like Geiger counters and magnetic field sensors. Once the payloads reach the stratopause at 100,000 feet above the ground, the balloon will burst. At this point, a parachute will deploy and slow the “stack’s” descent to earth. We can then retrieve the payloads and the data collected on the modules. Using this data, we can understand more about Earth’s atmosphere and near space as a whole.

2.0 Mission Overview

We will be sending a payload to the stratosphere, or near-space, by means of a helium-filled latex balloon. The payload will have to be less than 12 lbs in order to adhere to federal guidelines. These 12 lbs account for the entire stack, which includes our group’s payload, group D’s payload, rigging, parachute, etc. In order to account for these other materials, the total mass of our payload and group D’s payload must be under 5 lbs (thus our payload weight will be at most approximately 2.5 lbs).

Our group’s objectives are intentionally complementary to group D’s objectives, as we will both be studying similar environmental qualities in near-space, more specifically light and heat. Our payload will contain a PTERODACTYL control board, six Neulog modules, one camera, and various batteries for power. The six Neulog modules will consist of three temperature sensors (one being a wide-range sensor and the other two being more precise surface temperature sensors), a pressure sensor, a magnetic field sensor, and a battery pack. Overall, we hope to observe how conditions are different in the stratosphere than at the Earth’s surface. Our core experiment will be focused on the two surface temperature sensors. They will be in contact with a white and black piece of cardstock, respectively, to observe the extent

to which each absorbs/reflects visible light in the form of heat. They will be placed in direct view of the sun in order to do this. Although pressure steadily drops as altitude increases, temperature initially decreases then begins to increase in the stratosphere. Using this information, we can compare the temperatures measured by the two surface temperature modules with the values from the thermometers on the PTERODACTYL, wide-range temperature module, and pressure modules which will be acting as reference points. We expect that the black cardstock will have a higher temperature overall than the white cardstock, given that black should absorb all wavelengths of light and thus more heat (this is discussed in more detail in the following sections).

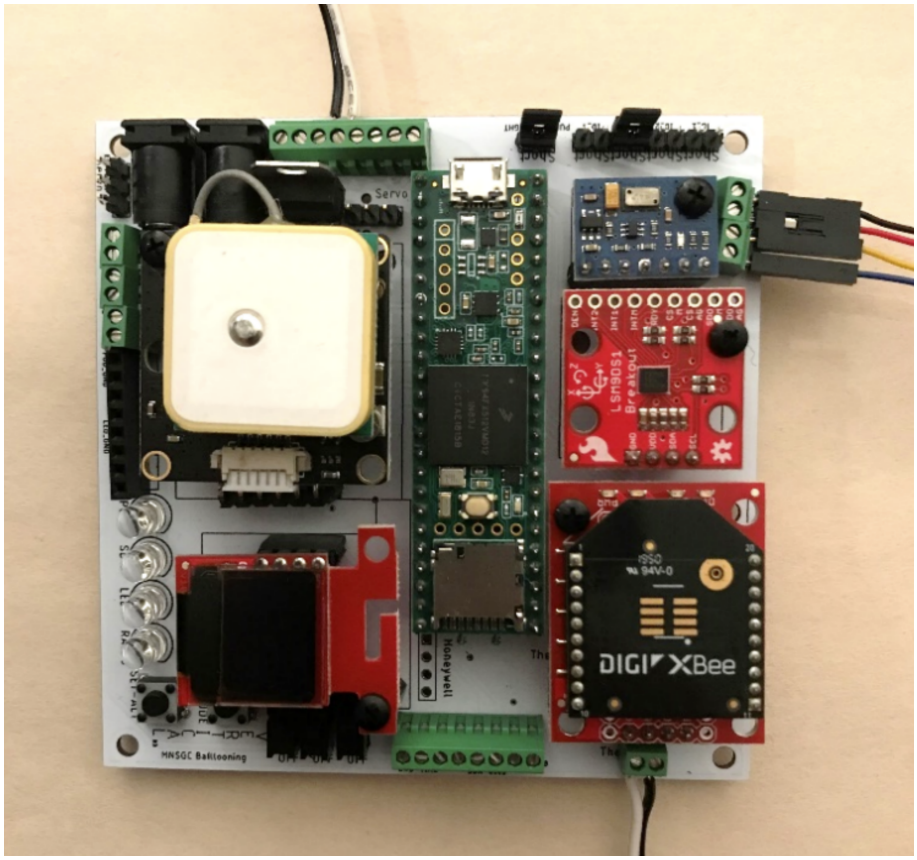
The magnetic field sensor will observe how the effects of Earth's magnetic field change with respect to altitude. The data from this module will not be compared to a control subject but will instead be used to observe how the reading changes during flight, since we are relatively unfamiliar with how the magnetic field behaves in near space as opposed to at ground level.

The payload will have to handle and survive both stratospheric conditions and re-entry conditions. In the lower stratosphere, the temperature will drop to below -40 degrees Fahrenheit, and as the payload rises the ambient pressure will be less than 1% of the pressure at sea level. We will have to ensure that all of the modules and systems continue to function and produce recoverable data under these conditions. Additionally, an important component of our objective is returning the payload safely. A parachute will be deployed at apogee, and will slow the payload's descent through the atmosphere. This, along with the payload's low mass, should ease the difficulties of re-entry typically experienced by space vehicles. The high deployment of the parachute should limit the "drifting" during descent that the vehicle will experience, allowing for easier recovery.

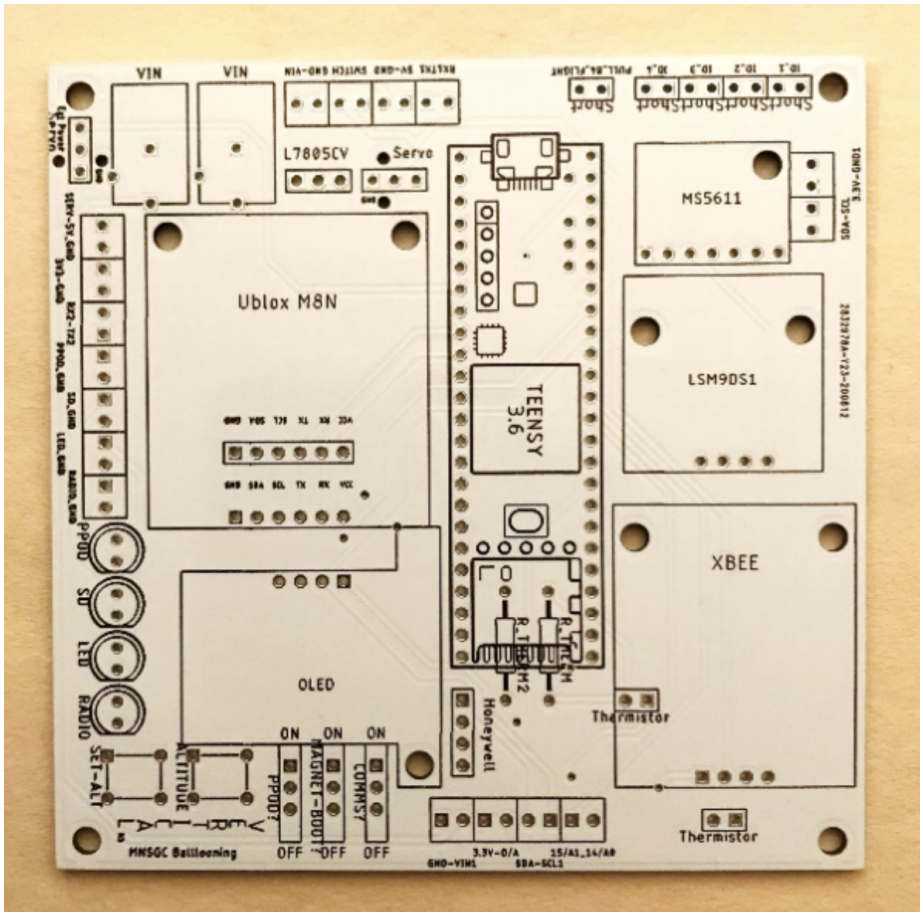
3.0 Payload Design

Wiring Diagram for PTERODACTYL

A populated, and wired, PTERODACTYL is shown below. The pair of wires at the top go to an external switch to start the PTERODACTYL The pair of wires at the bottom go to the external thermistor. The four wires on the right (Ground/black, 3.3V/red, SCL/yellow, SDA/blue) go to the QWIIC GridEye Sensor. Four external LED lights were also added to the PTERODACTYL for an external control panel. They were soldered into the circuit board in the bottom left corner, right next to the internal LEDs.



An Unpopulated PTERODACTYL Custom Printed Circuit Board (pcb)



Our payload is going to be constructed from ½ inch styrofoam insulation. We will be using this material because of its durability and its insulation properties. The styrofoam box will then be covered in copious amounts of black duct tape in order to secure the payload together as well as cover the surface in black material. Darker materials are known to absorb more light, and therefore, more heat. The extra heat will help maintain the temperature inside the payload, allowing the modules to stay warm enough to function properly.

As mentioned above, our payload is also engineered to record the ambient pressure, magnetic field, and wide range temperature. For this, we will need the three Neulog modules respective to each task. For our group-specific temperature experiment, we will utilize two surface temperature Neulog modules. Their temperature sensors will be attached to black and white cardstock in an effort to record a difference in temperature between the differently colored cardstock.

Design limitations

The most prominent limitation to our design is the weight requirement. In accordance with Federal Aviation Administration guidelines limiting the entire stack to 12 lbs and

to leave allowance for the second payload, parachute, balloon, and radio comms, our payload must weigh less than 2.5 lbs.
Additionally, our team has a self-imposed limitation of money. In this classroom setting, all our materials are provided, however, we would still like to be cautious of our payload cost. We wish to minimize the total price of our payload while maximizing functionality and experiment capabilities.

Parts and Equipment List

- Payload construction
  - ½ inch insulation
  - Black Duct Tape
  - Plexiglass
  - Rigging tubes
  - Strapping tape
  - Braided mason twine
  - 4 - Keyrings
  - Black cardstock
  - White Cardstock
  - LEDs
  - Button switch
- PTERODACTYL
  - PTERODACTYL board
    - Custom PCB
    - Teensy 3.5 without headers
    - 9 volt battery snap
    - 8G microSD with SD adapter
    - L7805CV5V voltage regulator
    - uBlox M8N gps
    - LSM9051 9DOF IMU
    - Thermistor
    - Resistor for thermistor divider(1%)
    - MS5611 pressure sensor
    - OLED (miniature screen)
    - XBee3 radio
    - XBee breakout board
    - LED (with built-in resistors)(4colors)
    - male/male jumper wires
    - male/female extender wires
    - Battery jack (solder in)
    - Male headers strips(first option for Teensy)
    - Female header strips
    - 2-position terminal blocks
    - 8-position terminal blocks
    - Shorting plugs
  - 2 - 9 volt batteries

- Neulog
    - Modules
        - Magnetometer
        - Pressure
        - Wide-range Temperature
        - 2 - Surface Temperature
    - Neulog chain wire
- Camera
    - Lightdow Camera
    - Anker battery
    - USB to microUSB cord
    - 32G microSD card
- Tools and Programs Used
    - Arduino
    - Neulog Software
    - Neulog USB module
    - Teensyduino
    - Breadboard
    - Soldering
        - Soldering Iron
        - Flux
        - Solder
        - Helping Hands
        - Solder sucker
    - Box cutter
    - Trident USB cord for programming Teensy and charging batteries
    - Tape measure
    - Pen
    - Flathead/Phillps screwdriver
    - L Square Ruler
    - Standard ruler
    - Chemical hand warmer

Internal Interactions

The Neulogs, PTERODACTYL board, and Lightdow camera will function as independent modules in our payload.

The Neulogs will be connected in two groups of three. These two groups will be connected with a USB extender cable, which will supply power to all the units.

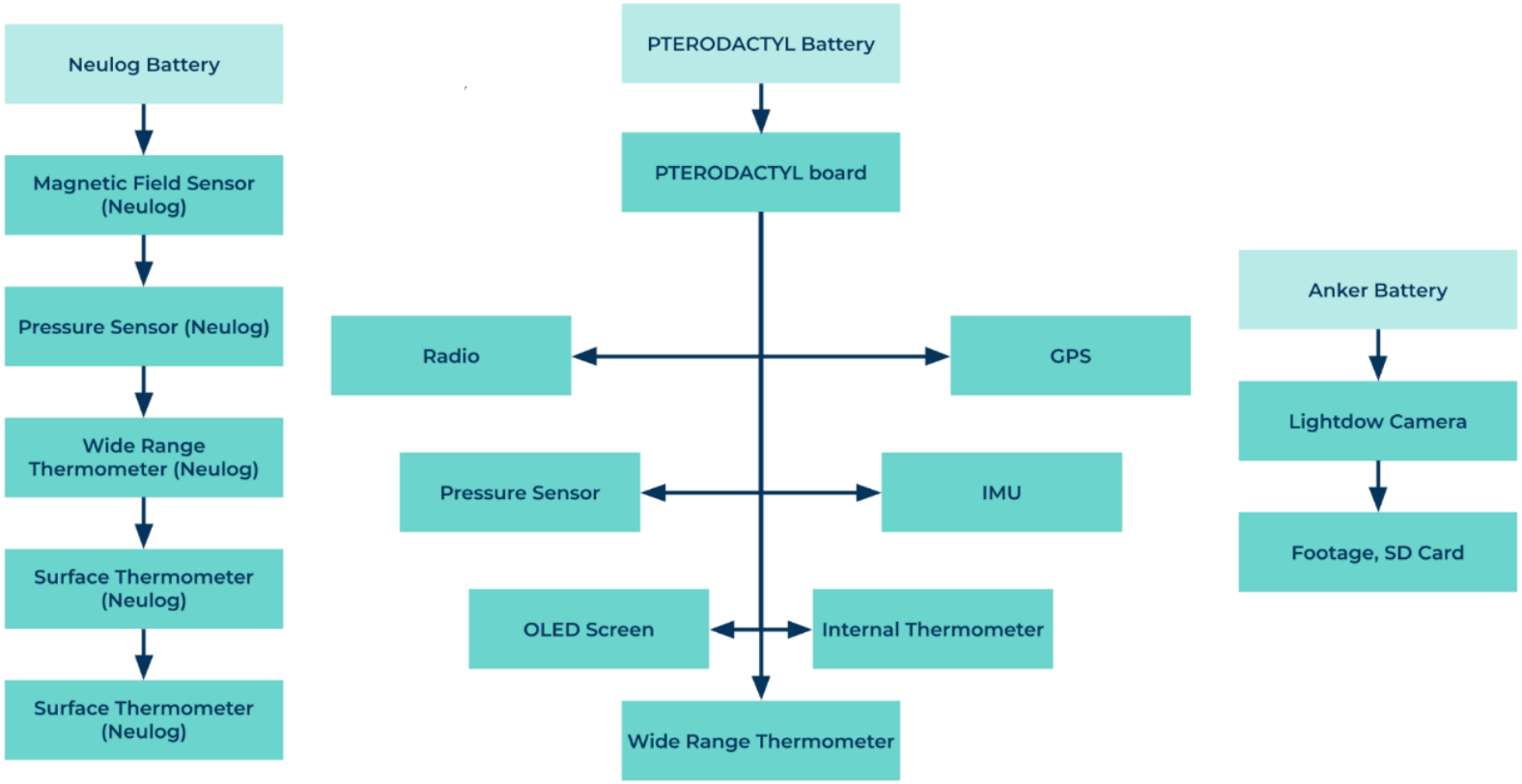
The PTERODACTYL board will have most of its components soldered directly on the board. The parts that will be connected externally are the IR sensors and the control panel, consisting of four LEDs and a switch, that will allow us to monitor the board's activities after the board itself is out of view. The LEDs will blink in accordance with the proper activity of the sensors and functions onboard the PTERODACTYL, specifically, PPOD, SD, LED, and Radio.

Our camera will be mounted 45 degrees below the horizontal axis to capture the Earth's image The camera is set to record for the duration of the flight and descent of the payload. The Lightdow itself is able to run while taking in energy but once it reaches full charge, it will no longer accept any more. Therefore, the camera battery will be drained prior to flight, then connected to the portable battery just before flight to maximize its life.

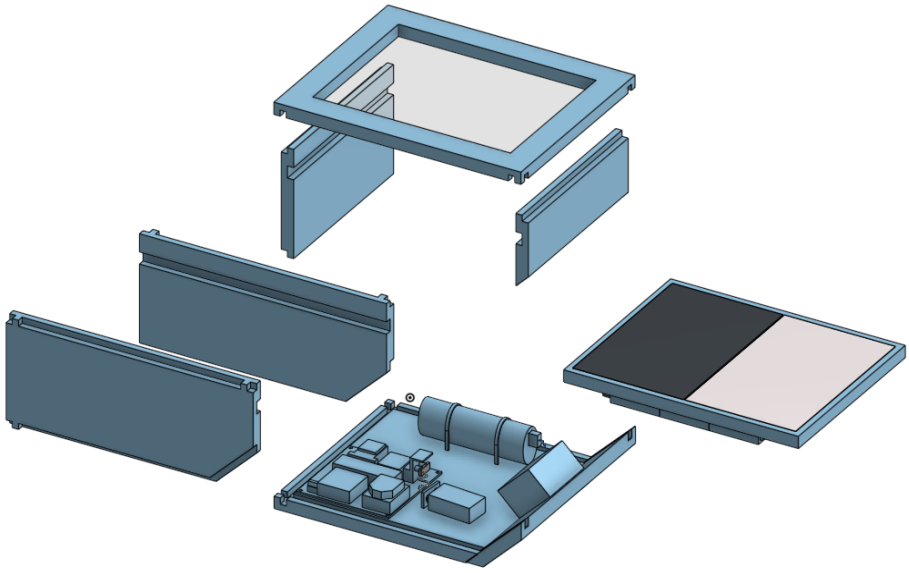
External Interactions

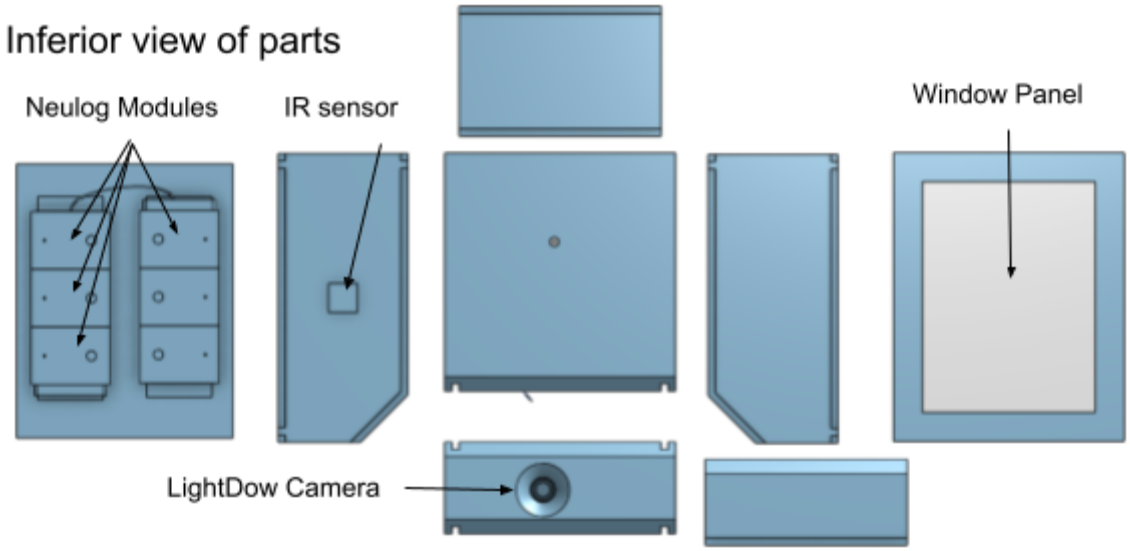
Our stack will be shared with Team D: Pyramid of Light. As their name suggests, their module is constructed in the shape of a pyramid, however, they plan on inversely mounting their payload so the point is facing down and the square base will be facing up. Due to this, we have decided to assemble our payload above theirs in the "stack". We have also coordinated with Team D to have our downward-facing cameras on opposite sides of the stack. Having two cameras mounted in opposite directions should allow us to retrieve footage even if one side of the stack becomes obstructed.

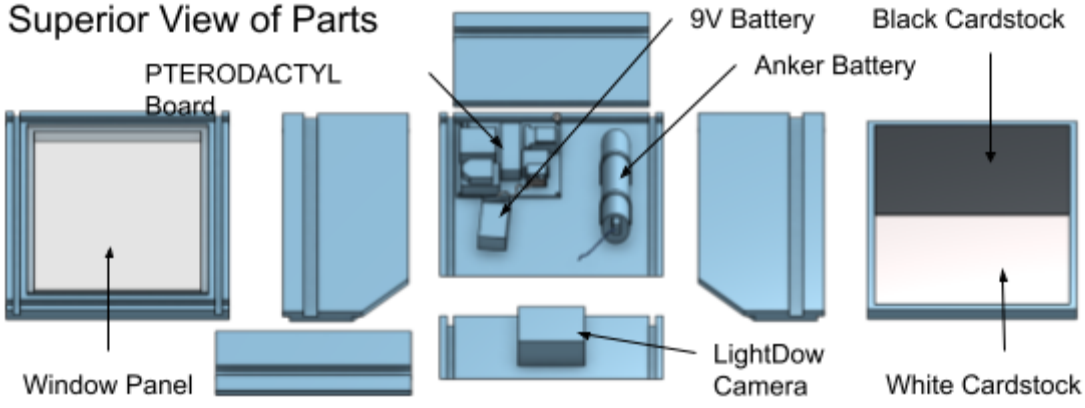
Functional Block Diagram

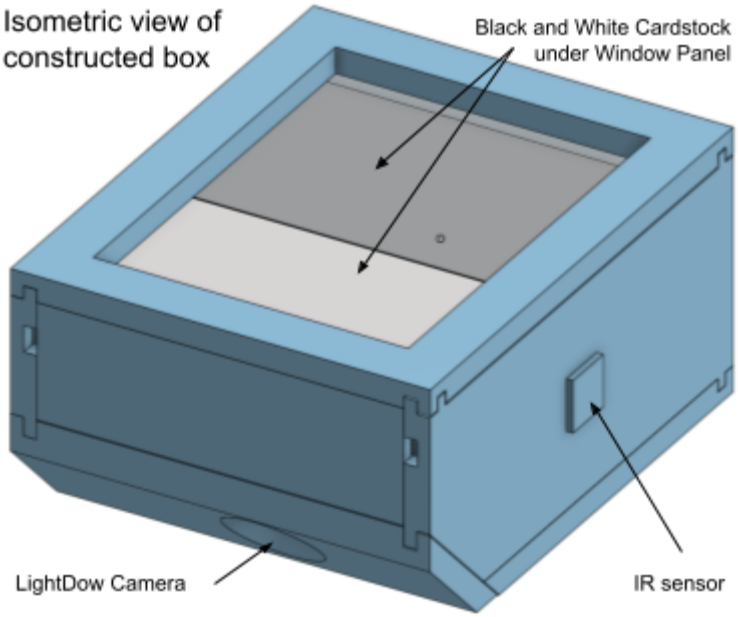


CAD Drawing of Payload Layout

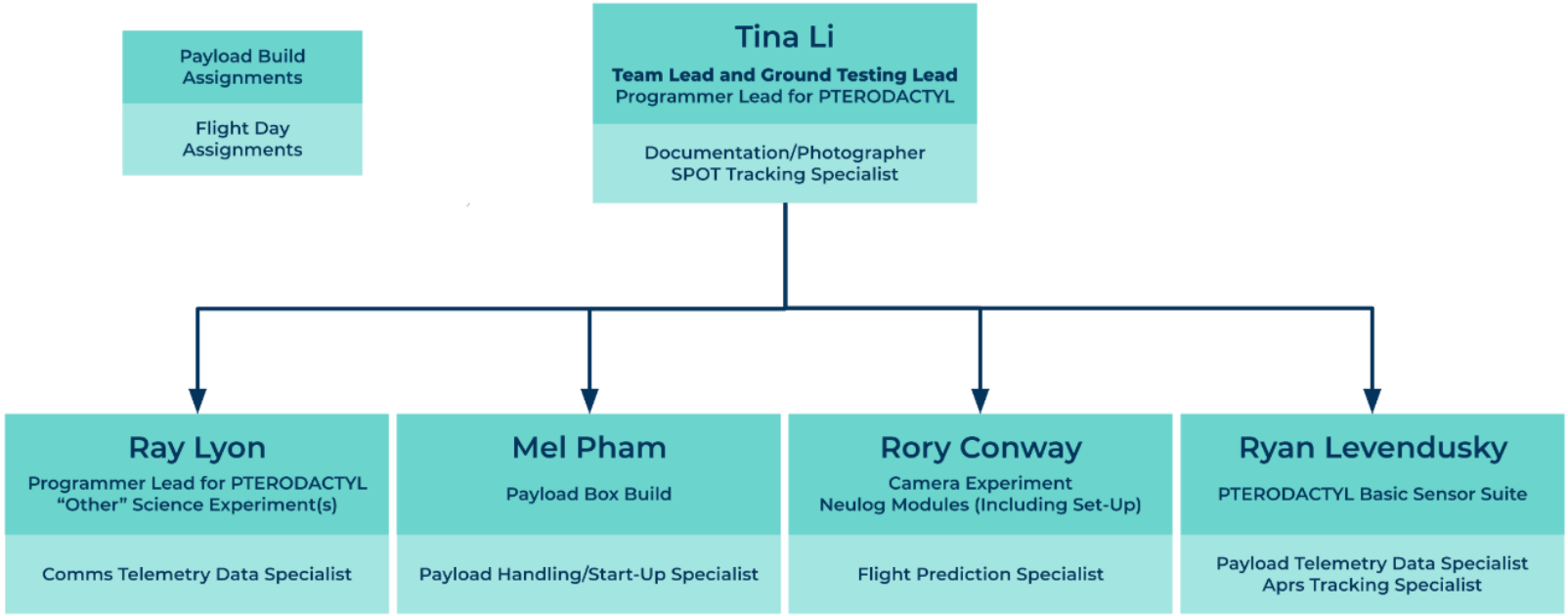







Isometric view of constructed box
Black and White Cardstock under Window Panel
LightDow Camera
IR sensor

4.0 Project Management and Schedule



Schedule

- 10/10: Team Meeting
  - Payload body construction
  - Component integration and testing
- 10/15: Rev A is due
- 10/17: Team Meeting
  - Review Teensy assignments
  - Confirm new payload layout and design
  - Start work on FFR
- 10/19: Class
  - Check status on FFR
  - Finalize payload layout
  - Box build finished
- 10/24 : Team Meeting
  - Finalize FFR
  - Finalize payload components (Neulog modules, PTERODACTYL, Camera, etc.)
- 10/26: Class
  - FFR presentation
  - Start work on Rev B
- 10/30 : Flight Day
- 11/2: Class
  - Check status on Rev B
- 11/5: Rev B is due
- 11/7 - 11:21: **TBD**
- 11/23: Class
  - Start work on Rev C
- 11/30: Class
  - Check Status on Rev C
- 12/3: Rev C is due

5.0 Project Budgets

The payload our group is constructing is projected to weigh around 34.5 ounces. This weight is divided amongst six Neulog modules which collect data from their surrounding environment. Combined, the six Neulog modules weigh approximately 10.8 ounces or near one-third of the total weight. The payload weight has significantly decreased since the initial design due to a new design that uses fewer materials and fewer electronics. The goal for this project is to keep our individual payload under 40 ounces. The estimate allows for design changes or structural changes if needed.

The final weight of our payload at the weigh-in was 38 oz even. This increase from the projected weight was due to an increase in tape used to reinforce the payload.

Although our payload had increased in weight, the total was still less than the max possible weight of 40 oz. This allowed our group to fly the payload without any changes.

Our payload is projected to cost $833. The bulk of this cost originates from the PTERODACTYL, costing just under $400, and the Neulog modules, costing around $200. The payload cost has been decreased by nearly $400 from our original design, due to changing to a less expensive camera. The sheets of costs and weights are listed below.

The final cost of our group's payload was $876.29. There was also an increase in price from our expected cost. This increase was caused by the addition of a QUIIC sensor. Our QUICC sensor was the Grid-EYE which came in at just under $43. Although this addition caused a price increase, our payload was relatively inexpensive for the data that was collected.

Building Materials	Weight(oz)	Number of Item	Total Weight(oz)	Total Cost($)
rigging tubes	0.50	1.00	0.50	$1.00
Strapping Tape	2.00	1.00	2.00	$1.50
Duct Tape	8.00	1.00	8.00	$3.50
Braided Mason Twine	0.40	1.00	0.40	$1.00
Key Rings	0.70	1.00	0.70	$1.00
Mobius	1.50	1.00	1.50	$69.00
Light Dow	1.80	1.00	1.80	$40.00
Anker Battery Pack	3.10	1.00	3.10	$20.00
MicroSD	0.00	1.00	0.00	$8.74
9V energizer battery	1.20	1.00	1.20	$8.00
Styrofoam	0.50	8.00	4.00	$1.00

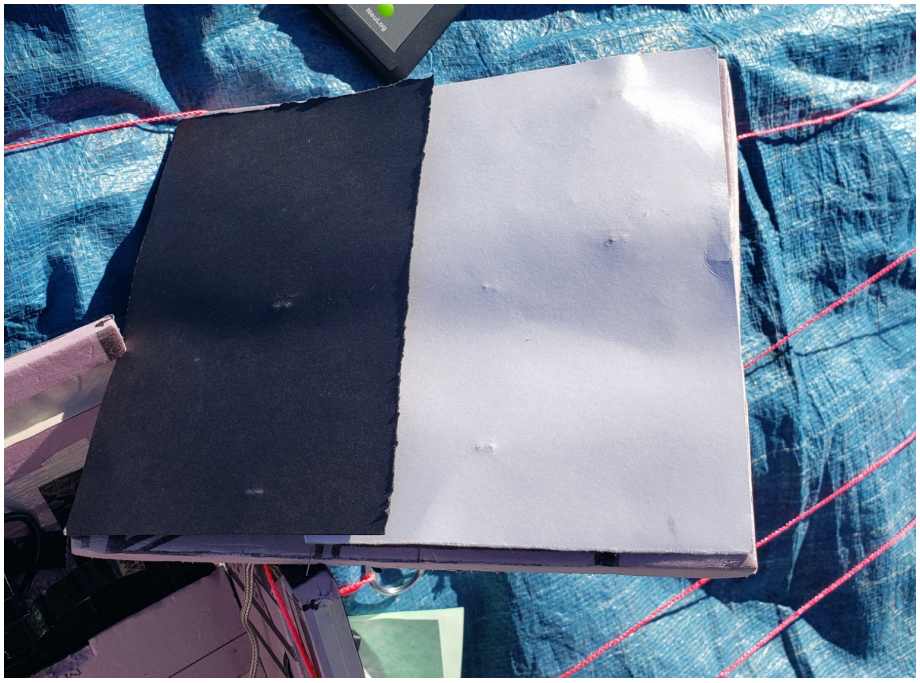
Neulogs	Weight(oz)	Cost($)
surface temp module	1.70	$25.00
surface temp module	1.70	$25.00
Battery module	2.30	$45.00
pressure module	1.70	$83.00
1-axis magnetometer	1.70	$58.00
wide-range temp module	1.70	$64.00

Payload Total Cost ($):	$876.29
Payload Total Weight (oz):	34.50

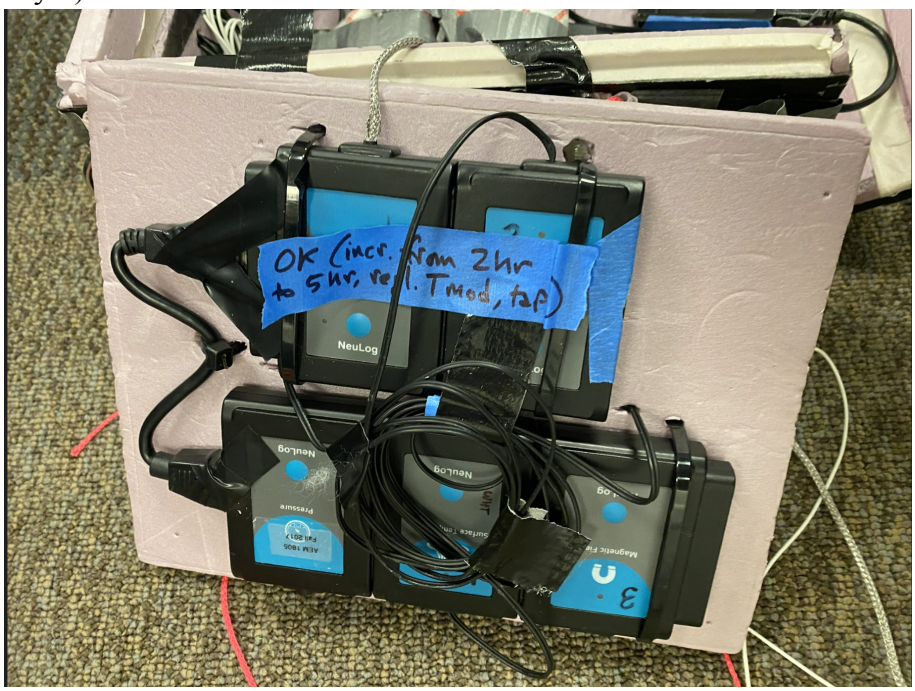
Pterodactyl Components	Weight(oz)	Total Cost($
Pterodactyl	4.00	$200.00
Custom PCB	INCLUDED	$5.00
Teensy 3.5 without headers	INCLUDED	$26.25
3 way cable	INCLUDED	$6.97
9v battery strap	INCLUDED	$1.25
8gb micro sd card	INCLUDED	$11.99
5v voltage regulator	INCLUDED	$0.43
uBlox M8N GPS	INCLUDED	$26.99
IMU	INCLUDED	$15.95
Thermister	INCLUDED	$4.10
Resistor for Thermister	INCLUDED	$0.05
Pressure sensor	INCLUDED	$8.53
OLED screen	INCLUDED	$16.95
Xbee3 Radio	INCLUDED	$20.06
Xbee3 breakout board	INCLUDED	$10.95
LED	INCLUDED	$1.79
Jumper Wires	INCLUDED	$0.45
Extender Wires	INCLUDED	$0.79
Battery Jack	INCLUDED	$1.27
Male header strips	INCLUDED	$2.54
Female header strips	INCLUDED	$3.27
2-Position terminal block	INCLUDED	$3.71
8-position terminal block	INCLUDED	$8.99
Shorting Plugs	INCLUDED	$0.32
GRIDEYE	n/a	$42.95

6.0 Payload Photographs

Cardstock Layer

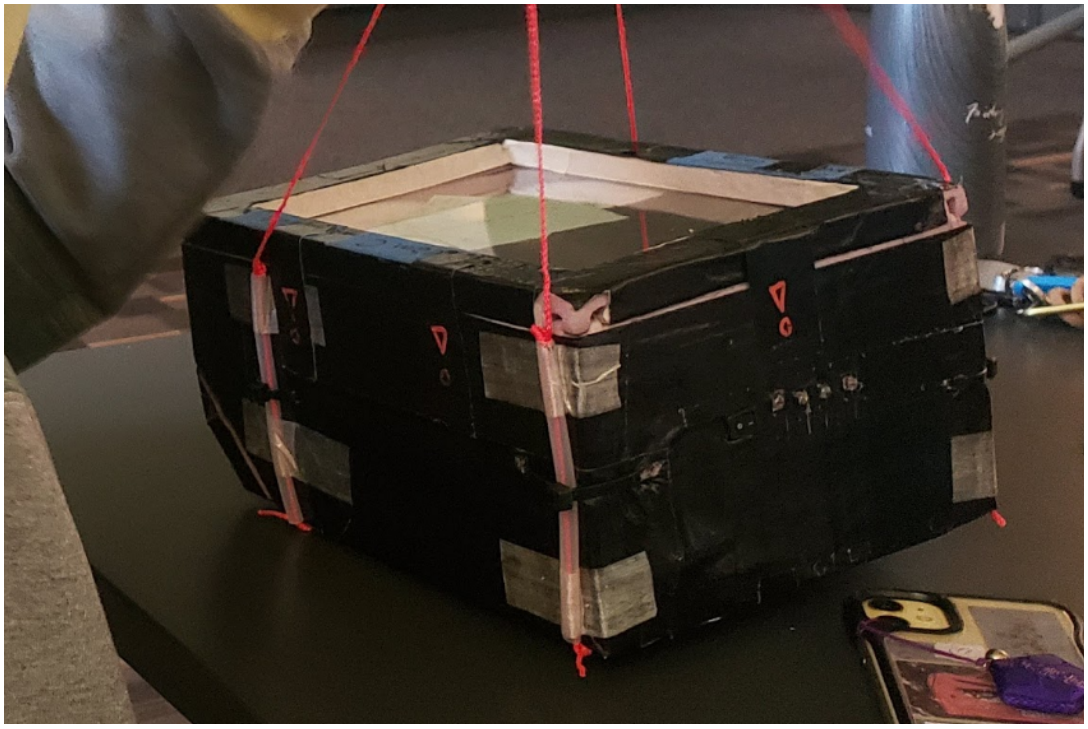


Neulog Chain (Located on the "Top" of the Box, the Underside of the Cardstock Layer)"



PTERODACTYL, LightDow Camera, and Anker Battery (for the Camera)

Full Payload Assembled



7.0 Pre-flight Ground Testing Plan and Results

Testing Plan

Neulog Surface Temperature Modules: We will be utilizing two Neulog surface temperatures modules to study the differences in temperature change of white and black materials. Since black materials absorb all wavelengths of light, the black piece of cardstock should have a higher temperature overall, compared to the white cardstock and the wide range temperature module. Because our Neulog sensors are located on the top of our payload (attached to the 'lid'), we will test the surface temperature sensors while they are connected to the top of the payload box. We will test the difference in heat by touching one sensor to white cardstock and the other to black cardstock, both of which are exposed to sunlight through a plexiglass window. Specifically, we are interested in the differences between the rates of change of temperature as well as the temperature change overall between the different colors of cardstock when exposed to white light. We plan to run the sensors for an hour outdoors (in order to mimic the lower temperatures of spaceflight), along with the rest of the Neulog chain in order to test for functionality prior to flight.

Neulog Magnetic Field Module: We will test the Neulog magnetometer similar to the two surface temperature modules - outside, for one hour with the rest of the Neulog chain. We will find the known values of the magnetic field at the location we test at and act as a baseline for our measured data to ensure there are no errors in the Neulog functioning.

Neulog Pressure Module: Similar to the rest of the Neulog modules, we will test the pressure sensor for one hour, outside, with the other Neulogs, in the chain

order that we will use them during flight. We will compare our measurements from this pre-flight test with known values at our testing location in order to confirm there are no issues with our Pressure Neulog or connections.

PTERODACTYL: During our flight, we will be using the wide range thermometer (internal and external), GPS, IMU, and radio. We will test all of these units at once, outside, for one hour. By testing all units together, we are able to test two things: if all units can run together with no problems, and if each unit works individually. Testing them all together simultaneously will help us to identify whether there are any problems with the PTERODACTYL as a whole. If something goes wrong, we will be able to see which unit had the problem and isolate it. For the two thermometers, we will be able to compare the known temperatures with the data we receive from the sensors. If they are not the same as the known values, we will know that there is an issue, and we will start troubleshooting. In order to test the GPS, we will use our known position (location of testing) and compare it to data we receive from the GPS. For the IMU, we will use the following procedure to test its functionality.

IMU: With the IMU turned on…

Testing the Gyroscope: Pitch the unit forward for one second, then back to center, then right, back to center, then backward, center, left, center. Using this four-directional test for the gyroscope, the IMU should be able to accurately measure when it experiences changes on the three-axis.

Testing the Accelerometer: The same test for the gyroscope will serve as a test for the accelerometer except we will be examining the data from the changes in motion.

Testing the Magnetometer: We will test this similar to how we tested the Neulog Magnetic Field Module. We will use the known values at the location of testing and compare it to the data recorded by the PTERODACTYL and Neulogs

Radio: This will just be a simple radio test to confirm that the PTERODACTYL is communicating with the receiver.

Camera: With the camera turned on, we will start to record and every 10 minutes a team member will signal the time interval. This is to ensure that there are no cuts or missing sections of recordings. We will be filming a clock to ensure there are no cuts in the footage.

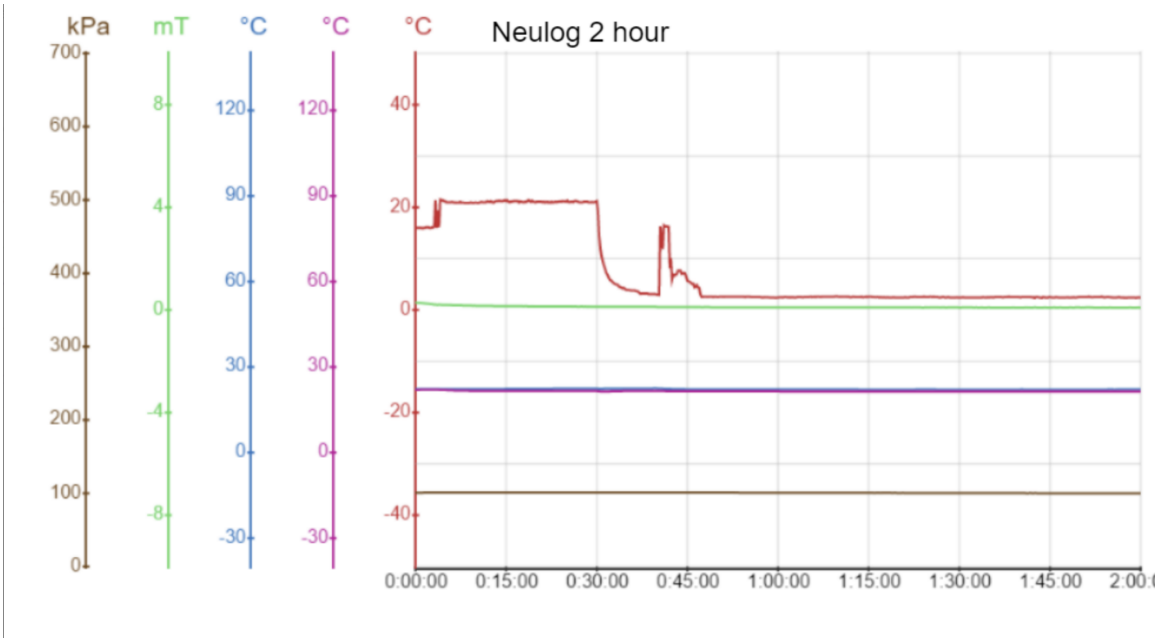
Payload Test: Assuming all of the previously listed tests revealed no problems, we will assemble the payload with all of its components, just as it would be on the day of the flight, and test all components together. After recharging the batteries and clearing all previous data from previous tests, we will run all of the tests previously described for each component simultaneously, for one hour.

Results:

- **Neulogs:** Neulogs were tested for approximately 2 hours, indoors concurrently with the rest of the components of the payload. Within the 2 hours, all modules

provided accurate data. These tests were sufficient enough to have confidence in the modules and their flight integrity.

- **Neulog Surface Temperature Modules:** Modules were run for two consecutive hours along with the rest of the Neulog chain. The two sensors were attached to the black and white cardstock paper respectively, and the test showed an approximate 0.6 °C temperature difference. The surface temperature modules are shown in pink and blue below.
- **Neulog Wide Range Temperature Module:** The module was run with the rest of the Neulog Chain for two hours. There were no issues observed during this period and the values detected are displayed below in red.
- **Neulog Magnetic Field Module:** Module was run with all other modules for two consecutive hours. No issues were observed during this period, and the magnetic field module results are shown in green below.
- **Neulog Pressure Module:** This module was run with all other modules for two hours. There were no issues seen during this period, and the values measured are shown below in brown.



- **PTERODACTYL:** The PTERODACTYL was tested for around two hours, concurrently with the Neulog Chain. This produced 5390 lines of data, which was recorded in an SD card as a CSV file. The first 15 lines of the first 38 columns are shown below as an example (not all lines of code are shown for conciseness). The test was conducted inside, and thus the GPS was unable to get a lock on the location. The PTERODACTYL radio was not tested, due to time constraints. This test demonstrated the functionality of all the sensors on the PTERODACTYL, which reinforced confidence in the ability of the PTERODACTYL on flight day.

Date	Hour	Minute	Second	Lat	Lon	Alt(ft)	AltEst(ft)	intT(F)	extT(F)	msTemp(F)	msPressure(PSI)
0/0/2000	-5	0	0	0	0	0		71.71	72.46	72.09	14.18
0/0/2000	-5	0	0	0	0	0	1021.33	71.55	72.29	72.1	14.17
0/0/2000	-5	0	0	0	0	0	1022.18	71.49	72.33	72.12	14.18
0/0/2000	-5	0	0	0	0	0	1021.61	71.61	72.35	72.14	14.18
0/0/2000	-5	0	0	0	0	0	1021.05	71.63	72.31	72.14	14.18
0/0/2000	-5	0	0	0	0	0	1021.9	71.64	72.28	72.14	14.17
0/0/2000	-5	0	0	0	0	0	1022.46	71.64	72.28	72.16	14.17
0/0/2000	-5	0	0	0	0	0	1022.74	71.67	72.24	72.18	14.17
0/0/2000	-5	0	0	0	0	0	1022.74	71.54	72.24	72.19	14.17
0/0/2000	-5	0	0	0	0	0	1022.74	71.7	72.23	72.19	14.17
0/0/2000	-5	0	0	0	0	0	1022.74	71.55	72.15	72.21	14.17
0/0/2000	-5	0	0	0	0	0	1022.46	71.7	72.19	72.23	14.17
0/0/2000	-5	0	0	0	0	0	1022.18	71.75	72.22	72.23	14.17
0/0/2000	-5	0	0	0	0	0	1023.02	71.71	72.07	72.25	14.17
0/0/2000	-5	0	0	0	0	0	1023.02	71.76	72.16	72.25	14.17

time since bootup (s	magnetometer x	magnetometer y	magnetometer z	accelerometer x	accelerometer y	accelerometer z	gyroscope x	gyroscope y	gyroscope z	gridEye Index	gridEye Value
72.25	0.41	-0.28	1.12	-0.24	0.06	1.13	3.33	1.39	5.25	DATA STRING TRAI	0
73.25	0.41	-0.28	1.12	-0.33	0.13	1.13	8.84	5.45	9.8		15
74.25	0.41	-0.28	1.11	-0.22	0.12	1.15	3.04	0.97	8.88		1
75.25	0.41	-0.28	1.12	-0.13	0.12	1.31	2.82	2.09	8.72		55
76.25	0.41	-0.28	1.11	0.06	0.11	1.11	2.15	4.38	-11.16		15
77.3	0.41	-0.28	1.12	0.03	0.07	1.17	5.34	5.23	5.87	DATA STRING TRAI	15
78.3	0.41	-0.28	1.11	0.04	0.08	1.18	4.1	1.44	5.27		1
79.3	0.41	-0.28	1.12	0.03	0.09	1.19	5.05	1.89	5.35		22
80.3	0.41	-0.28	1.12	-0.03	0.08	1.18	3.99	3.82	3.75		15
81.3	0.41	-0.28	1.11	-0.24	0.14	1.15	-5.64	-15.72	6.83		6
82.3	0.41	-0.28	1.12	-0.4	0.15	1.09	2.43	-3.37	6.62	DATA STRING TRAI	15
83.3	0.41	-0.28	1.11	-0.37	0.14	1.12	7.26	8.11	3.46		1
84.31	0.41	-0.28	1.12	-0.22	0.11	1.15	8.51	4.22	5.26		22
85.31	0.41	-0.28	1.12	-0.13	0.05	1.19	6.63	7.87	4.38		39
86.31	0.41	-0.28	1.12	0.18	0.08	1.14	8.4	10.16	-0.56		7

gridEyeArray[0]	gridEyeArray[1]	gridEyeArray[2]	gridEyeArray[3]	gridEyeArray[4]	gridEyeArray[5]	gridEyeArray[6]	gridEyeArray[7]	gridEyeArray[8]	gridEyeArray[9]	gridEyeArray[10]	gridEyeArray[11]
0	0	0	0	0.000.00	0	0	0.000.00	0	0	0.000.00	0
20.75	19.5	19.75	19.75	20.0020.00	20	20.25	20.5019.50	19.75	19.5	20.2520.00	20.25
20.75	20	19.75	20	19.5020.00	20	20	20.0019.75	19.5	19.5	20.0020.25	20.25
20.75	20	20.25	20	20.2520.00	20.25	20.5	20.5019.25	20	20	20.0020.25	20.25
20.75	20	21.25	20	20.2520.00	20.25	20	20.5019.75	20	19.75	20.2520.00	20
20.75	19.75	20.5	20.25	20.2520.50	20	20.25	20.5019.75	20	20	20.2520.25	20.5
20.5	20	20.5	20.25	20.0019.75	19.75	20.25	20.5019.75	19.75	19.75	20.0020.00	20
21	20	20	20	20.2520.00	19.75	20.5	20.5019.50	19.25	19.5	20.2520.00	20
21	20	20	20	20.0019.75	20.25	20.75	20.2519.75	19.5	19.5	19.7520.00	20.25
20.5	20	19.5	19.75	20.0020.00	20.25	20.5	20.5019.75	19.25	19.75	19.7520.00	20
20.75	19.75	20.25	19.5	20.0019.75	20	20	20.2519.25	20	19.75	19.5020.00	19.75
20.75	20	20.75	20	20.5020.00	20.25	20.25	20.5020.00	20	19.75	19.7520.25	20
20.5	20.25	20.25	20.5	20.5020.00	20.75	20	20.2519.50	20	19.75	20.2520.00	20.5
20.75	19.75	19.75	20.25	20.2519.50	20.5	20.25	20.2519.75	19	19.5	20.2519.75	20.25
20.75	20	20.25	20.25	20.0020.25	20	21	20.5019.75	19.5	19.25	20.2519.75	20.5

gridEyeArray[12]	gridEyeArray[13]	gridEyeArray[14]	gridEyeArray[15]	gridEyeArray[16]	gridEyeArray[17]	gridEyeArray[18]	gridEyeArray[19]	gridEyeArray[20]	gridEyeArray[21]	gridEyeArray[22]	gridEyeArray[23]
0	0.000.00	0	0	0.000.00	0	0	0.000.00	0	0	0.000.00	0
19.75	20.7519.75	20	19.75	20.0019.75	19.75	20.25	20.0019.50	19	19.25	19.2520.00	20.25
20.25	20.2519.75	19.75	20	20.2520.75	20.25	20.25	20.0020.00	20	19.75	19.5020.00	20.25
20.25	20.2519.25	19.5	20	19.5020.50	19.75	20.5	19.7519.75	19	19.75	19.0020.25	20
20	20.7519.75	19.75	20	20.0020.00	20.5	20	20.0019.50	19	19.5	19.2520.25	19.75
20	21.0019.75	20.25	19.5	20.0020.50	20.25	20.25	20.2519.00	19	19.25	19.5020.00	20.25
20	20.2519.00	19.75	20	20.0020.25	20	20.5	20.0019.75	19	19.75	19.5020.00	20.25
20	20.2519.75	20.25	19.75	19.7520.25	19.75	21	20.0019.50	19.25	19.5	19.7520.25	20.25
19.75	20.2519.50	20	20	20.0020.25	20	20.5	20.2519.50	19.25	20	19.7520.75	20.25
20	20.5019.25	19.5	19.75	19.5020.00	20	19.75	20.0018.75	18.75	19.75	19.5020.00	20.25
19.75	20.2519.75	20.25	20	20.0020.50	20.25	20.25	19.7520.00	18.75	20	19.5020.00	20
20	20.5019.50	20	19.75	20.2520.50	20	20.5	20.5019.75	19.25	19.5	19.7520.25	20.25
19.75	20.5019.50	19.5	20	20.0020.25	20	20.25	20.0019.50	19.5	19.75	20.0020.25	20.5
20.25	20.0019.25	20.25	19.75	19.5020.00	20	20.5	20.2519.50	19	19.5	19.5020.25	20

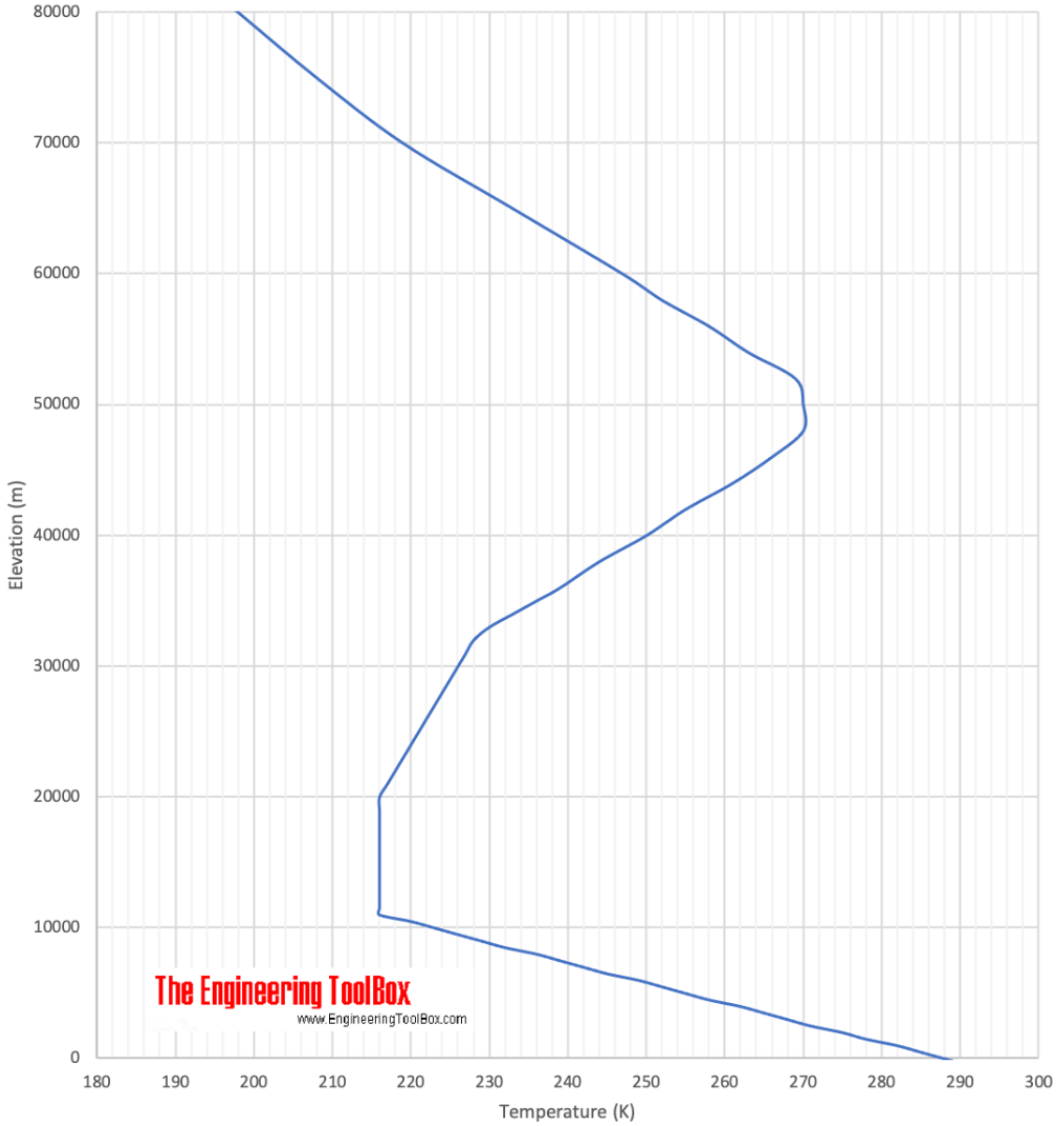
- **Camera:** The camera test consisted of filming a wall clock for two continuous hours. The footage recorded from this test showed no cuts or issues. The result of this test was sufficient for us to have confidence in its performance on flight day.
- **Payload Test**: Payload was tested with all components together as described above. There were no serious issues observed during this test and no changes were made to the payload due to the tests before flight day. Of note, the payload was wrapped in black tape following the tests before check-in, but this was unrelated to the test results and purely in order to retain heat during the flight.

8.0 Expected Science Results

Figure 1: Temperature, Gravity, Pressure, Density, and Dynamic Viscosity against Altitude[5]

Geo-potential Altitude above Sea Level - h - (ft)	Temperature - t - (^{o}F)	Acceleration of Gravity - g - (ft/s^2)	Absolute Pressure - p - (lb/in^2)	Density - ρ - (10^{-4} slugs/ft^3) (lbs/ft3)	Dynamic Viscosity - μ - (10^{-7} lb s/ft^2) (10^{-7} slug /(ft s))
-5000	76.84	32.189	17.554	27.45	3.836
0	59	32.174	14.696	23.77	3.737
5000	41.17	32.159	12.228	20.48	3.637
10000	23.36	32.143	10.108	17.56	3.534
15000	5.55	32.128	8.297	14.96	3.430
20000	-12.26	32.112	6.759	12.67	3.324
25000	-30.05	32.097	5.461	10.66	3.217
30000	-47.83	32.082	4.373	8.91	3.107
35000	-65.61	32.066	3.468	7.38	2.995
40000	-69.70	32.051	2.730	5.87	2.969
45000	-69.70	32.036	2.149	4.62	2.969
50000	-69.70	32.020	1.692	3.64	2.969
60000	-69.70	31.990	1.049	2.26	2.969
70000	-67.42	31.959	0.651	1.39	2.984
80000	-61.98	31.929	0.406	0.86	3.018
90000	-56.54	31.897	0.255	0.56	3.052
100000	-51.10	31.868	0.162	0.33	3.087
150000	19.40	31.717	0.020	0.037	3.511
200000	-19.78	31.566	0.003	0.0053	3.279
250000	-88.77	31.415	0.000	0.00065	2.846

Figure 2: Temperature against Elevation[5]



As shown in Figure 1, pressure values consistently decrease as altitude increases, and thus we expect the same to be seen in our data from both the Neulog pressure module and the pressure sensors on the PTERODACTYL. Given that we expect to fly to around 100,000 ft above sea level, we expect pressure to reach a low of 0.162 lb/in^2.

Using the same table, as well as figure 2, we can determine the expected values for external temperature, which is detected by the wide range temperature Neulog module as well as the external thermistor from the PTERODACTYL. As altitude increases, temperature initially decreases, then at around 60,000 ft, begins increasing due to the ozone present in the stratosphere which absorbs UV light. Using the same expected altitude (100,000 ft), we expect that the temperature detected at the peak of our flight will be around -51.10 °F or -46.16667 °C. The lowest temperature we expect to detect is around -69.70 °F.

Similarly, we expect that the temperature values from the two Neulog surface temperatures modules that are under the black and white cardstock respectively will vary and change with time. Since the entire experiment is housed inside the payload, there should be additional insulation from the external temperature, meaning that both values should be higher than the temperatures detected from the external thermistors. In accordance with the table above, we'd expect that the minimum temperature for either of them will be at or above the listed atmospheric temperature for a given altitude. However, because of the heat from light absorption, it is expected that there will be a significant difference in the values between the white and black cardstock, with the white cardstock falling closer to the temperatures on the table. Pre-flight tests were not performed outdoors in which more precise numbers could be obtained, but from indoors preflight tests under a dorm room light, the black cardstock was recorded at roughly 0.6^{o} C higher than the white cardstock. We hypothesize that this difference is due to the increased light absorption of the black material, and that a similar difference will be observed in flight.

There is little to expect from the magnetic field module, as the magnetic field decreases with the square of the distance from the source of the field, according to the International Geomagnetic Reference Field model of the Earth's magnetic field[6]. The center of the field in this case is the Earth's core, and relative to the radius of the Earth, the altitude gain of our payload is fairly insignificant. We expect the values from the magnetic field module to decrease slightly, if at all.

As the atmosphere thins (by increasing altitude), wind effects should decrease. Therefore, we would expect the change in gyroscope values, representing the orientation of the payload, to decrease.

Since the QWIIC GridEye sensor is relatively novel we do not have many expectations as to the data we will receive. We are recording the temperature of all 64 points captured by the sensor and should be able to "see" the earth through increased temperature points since the earth is hotter than the general atmosphere.

The LightDow Camera is pointed down and out, and thus we should be able to capture the curvature of the earth in our footage.

9.0 Flight Day Narrative

Our chosen flight day was October 30th, 2021. Team members gathered at Akerman hall at 7 a.m. and gathered flight day materials (tape, hand warmers, batteries) and discussed the plan of flight day.

At 8 a.m. teams drove to the Janesville-Waldorf-Pemberton, a school in Janesville, MN. Although this was the predetermined launch site, we were not given permission

to launch on the premises so we moved to a nearby clearing behind Trinity Lutheran Church.

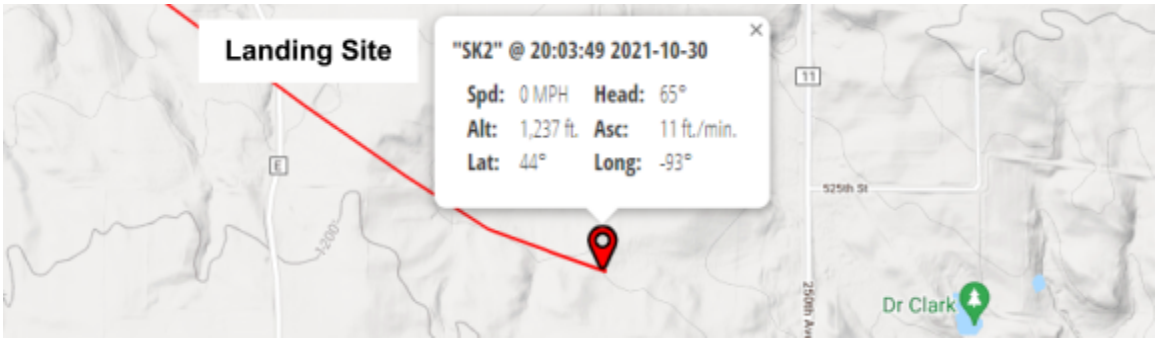
There, teams started to assemble the stacks. First, a tarp was laid out. This was to prevent the parts from getting wet, which could damage the electronics, and to minimize the loss of parts in the grass. Since Teams C and D were flying their payload on the same balloon, teams worked together to assemble the stack. Team members laid out the components on the tarp in the order that they were to fly. Next, the payloads were connected using keyrings and 550 cords. The payloads had previously been rigged with key rings on the bottom, and long cords with loops jutting out from the top. The keyrings allowed for the payload to be connected easily while ensuring a strong connection that would not come apart during flight.

Tying 550 cords to the keyrings in order to assemble all of the payloads in the stack.

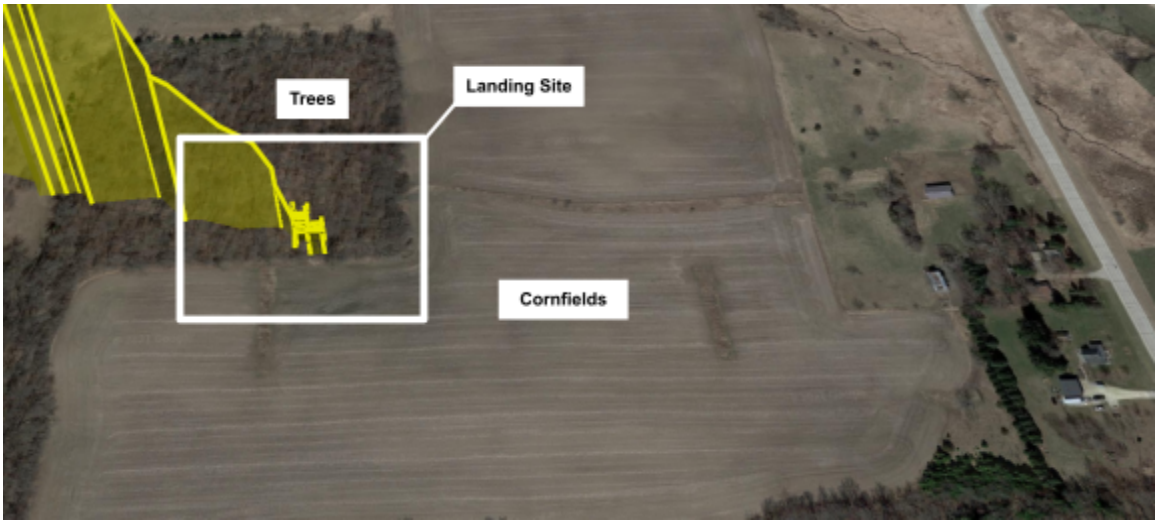
Sayeon, our TA, checked the PTERODACTYL batteries and code, most notably, the relay code. It was crucial that the payloads could transmit information from the Xbees to the SatCom, if they did not it would be impossible to tell whether any of the sensors were functioning correctly and it would be impossible to track the balloon in flight. Sayoen also took the time to double check that the rest of the PTERODACTYL code was functioning properly. Teams were then instructed to open chemical hand warmers and shake them to activate them. These hand warmers were taped to the battery components of the payloads to keep them within functioning temperature while exposed to cold stratospheric temperatures. The PTERODACTYL, NeuLogs, and camera were turned on and the box was taped shut. Students began to fill the weather balloons. These balloons had to be held while inflation. To inflate these balloons, Professor Flaten fashioned a special adapter. One end of the tube was connected to the helium tank and the other side had a ¼" A-style female coupler. A PVC pipe with an associate male plug was fitted into the neck of the balloon. The Style coupler and plug would allow for tanks to be switched when empty without losing much helium from the balloon. During inflation, one student sat on the tarp and tied a string to the balloon and themselves. This string acted as a life line so as to not lose the balloon and too much helium if the balloon was prematurely let go. Several other students wore latex gloves and held the body of the balloon, stabilizing it as it inflated. The amount of helium was measured using a luggage scale hooked onto the neck of the balloon. As the balloon inflated, the payloads were checked once again and final adjustments were made. 15lbs of helium was used to fill each balloon. After this weight was reached, the PVC adapter was removed and the neck was sealed using zip ties and tape. The life line was replaced with a rigging string which was connected to a keyring at the top of the stack. Soon after, the stacks were walked over to a clearing to get ready for launch. Siren payloads were activated to assist with recovery and the stacks were let go. The balloons were taken by the wind and headed northeast. At this point, the research team started to assemble their stack. After launch, teams decided to get lunch before pursuing the balloons.

Launching of Team C and Team D's Balloon

After lunch, the balloon GPS was checked and the calculated flight path was compared to the actual. The balloon was expected to land near West Concord, about 45 miles east of the launch site, however, the actual flight path showed the balloon tracking farther west than expected. Teammates hurried into the van and decided to head to Roscoe to intercept the balloon's course. We were hopeful to try and find it before it landed. On the way, Rory was in charge of navigation while the other teammates monitored the GPS, elevation, and speed in an effort to anticipate where it would land. About an hour into the flight, we passed through Owatonna. At this point, we contacted Team D to see where they were and where they planned to drive to. We coordinated with them to meet somewhere between Roscoe and Berne. After an hour and a half of flight, the balloon's GPS became stagnant.



The balloon had landed in what seemed to be a corn field. Unfortunately, we did not get to see the balloon popping or during its descent. Regardless, our team made it to the property that it had landed in. After walking into the middle of the field, we followed the sound of the siren and found the stack draped over a tree at the edge of the corn field.



Unable to get it down, we called Professor Flaten and asked for help. Soon after, Team D made it to the property and called asking if we had found the stack. Rory, Ray, and Ryan ran out to tell them we had and that they should stay at the front of the property to help Professor Flaten carry equipment to the landing site. Once Professor Flaten arrived, he began to instruct students on how to get the payload down. He has a long telescoping rod. At the end of this rod was a hook. The goal was to take a string and, using the rod, place it over the tree and payloads. The string would be pulled out of the tree and, hopefully, the payload would come with it. The string was placed over the stack successfully, but when pulled, the strings between the SatComm and the first payload snap. The SatCom and balloon fall down but the payloads are still stuck. The string is put back over the tree for a second attempt. When pulled, the payloads are pulled closer to the ground but they become stuck in the branches. At this point, there are too many branches in the way and the sting cannot be placed again.

Professor Flaten moves to the ladder and requests Ty Flanagan to climb up and pull the payloads down. He does so successfully and the payloads are retrieved. On the ground, they are cut apart and all the modules are turned off. The equipment is collapsed for ease of transportation. The area is cleaned of any waste and the Teams head back to the vans. After a long day, Teams C and D drive back to campus and flight day comes to an end.

10.0 Results and Analysis

Overall, our flight was relatively successful. Most of our sensors were functional throughout the entire flight, but some sensors recorded unreasonable values, thus suggesting recording or measuring error. Notably, our camera only began recording partway through the balloon flight, as discussed below in detail.

SatComm Data

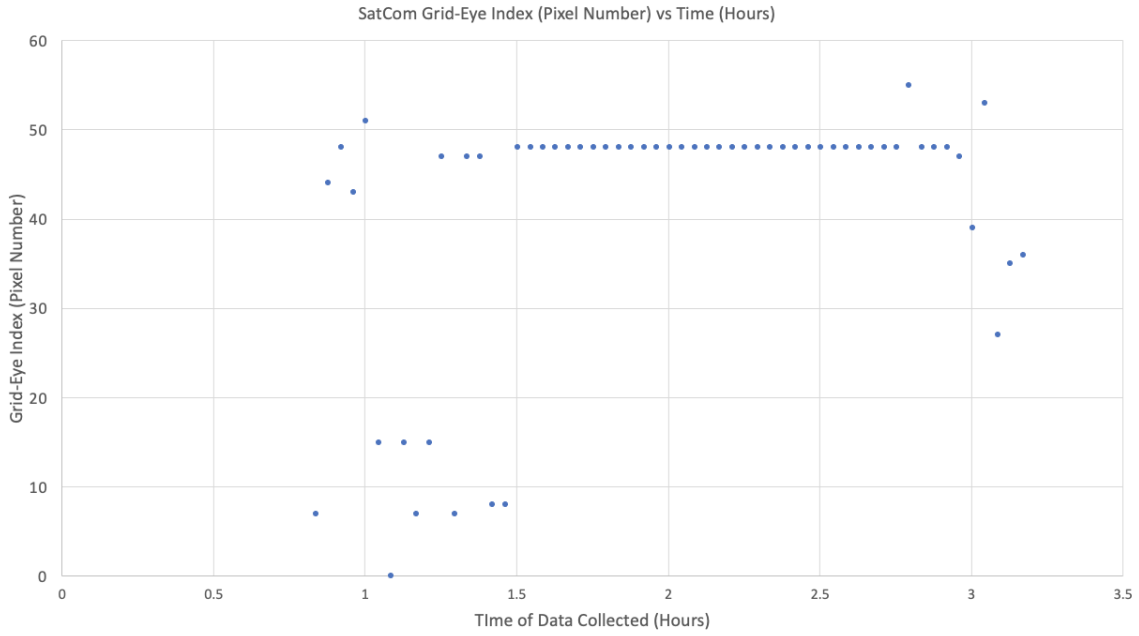


Figure 1: SatComm GridEye Index vs Time

Figure 1 depicts which pixel in an 8 by 8 grid was the hottest value over the span of the flight. Time is measured in hours since the beginning of the flight and the GridEye index is measured in the pixel number with which the data was recorded. Although this data seems plausible, after receiving the full recorded data from the GridEye it is clear that all of the values measured were implausible and there was an issue with either measurement or recording.

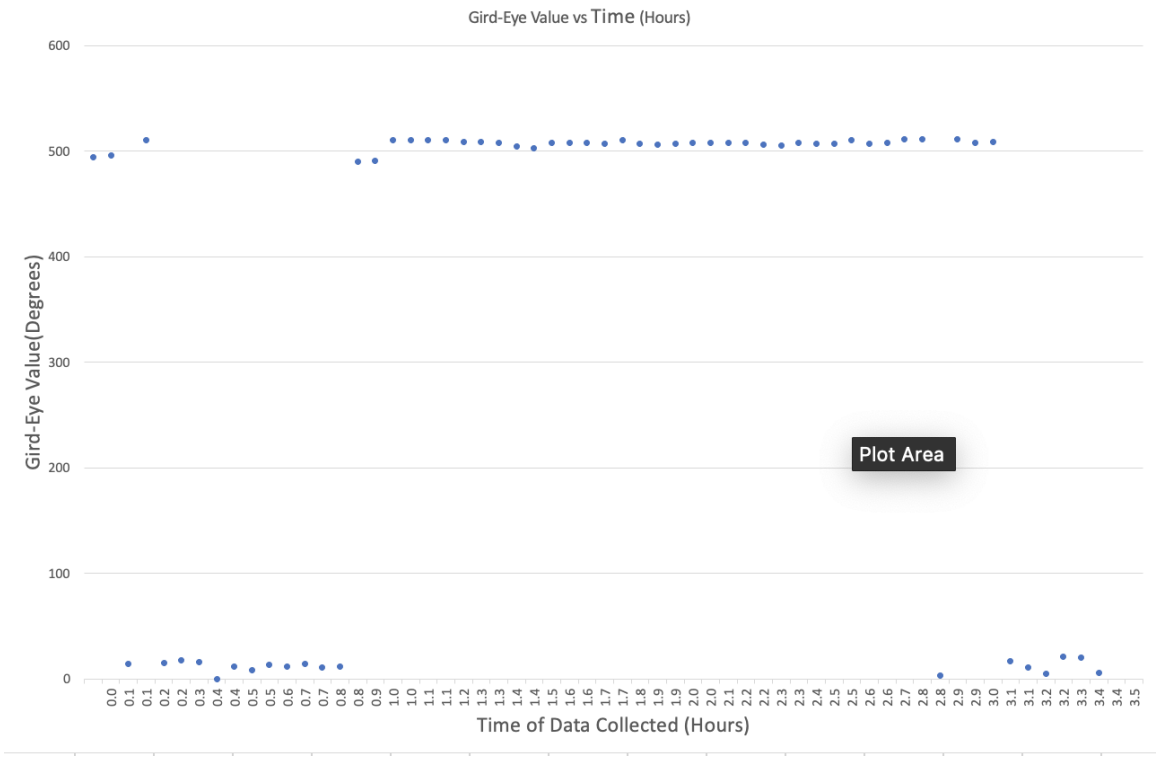


Figure 2: SatComm GridEye Value vs Time

Figure 2 represents the data collected from the Grid-Eye Quicc sensor. The data that was recorded for this figure was the value of the pixel that was displayed on figure number 1. Similar to Figure 1, all of the data here is unusable, since the QUIIC GridEye sensor was corrupted during flight.

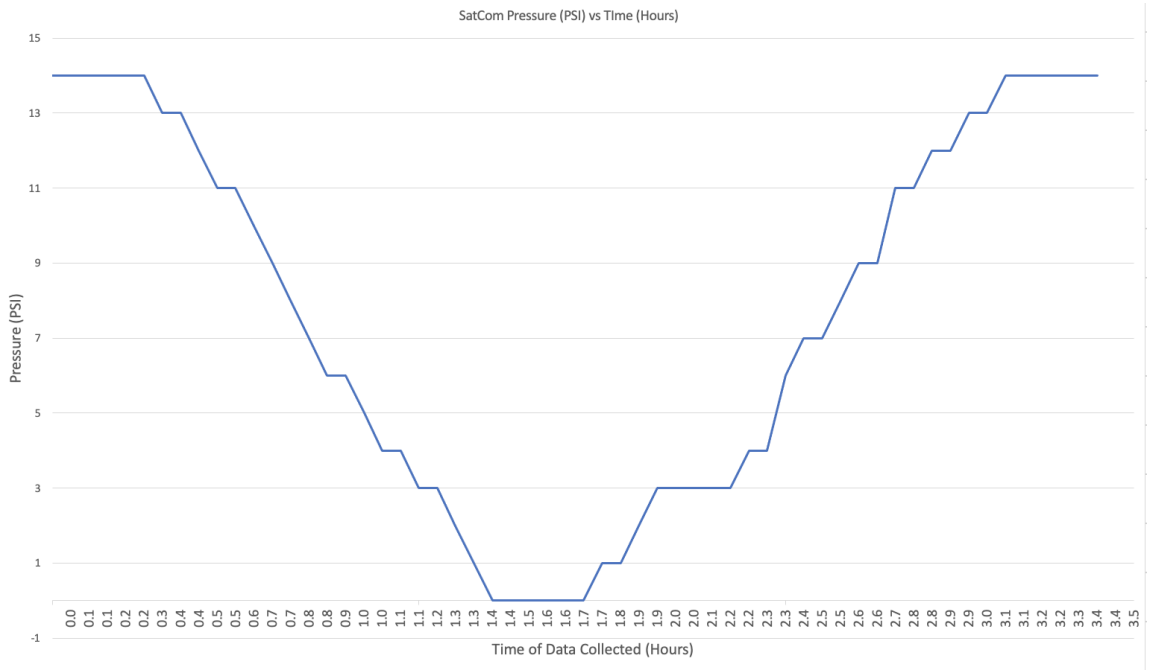


Figure 3: SatComm Pressure vs Time

Figure 3 represents the pressure relative to time from the SatCom throughout our flight. Pressure is measured in PSI and time is measured in hours. The SatCom pressure data is identical to the data collected directly from the PTERODACTYL the only difference is that the SatCom recorded on five minute intervals vs the 1 second interval on the PTERODACTYL.

Overall, the data received from the telemetry was near unusable. The only data that was readable from the received portion was the pressure data. It clearly showed the pressure decreasing as the balloon went towards the atmosphere and then a sharp increase in pressure as the balloon fell from its peak height. The other two groups of data collected were GridEye value and GridEye index. When it comes to understanding the data collected from the GridEye sensor it is practically unreadable since our GridEye values were converted incorrectly and there is no way to calculate it to a version that is understandable. The GridEye index partially makes sense since it should just display the warmest temperature of an eight by eight grid.

PTERODACTYL Data

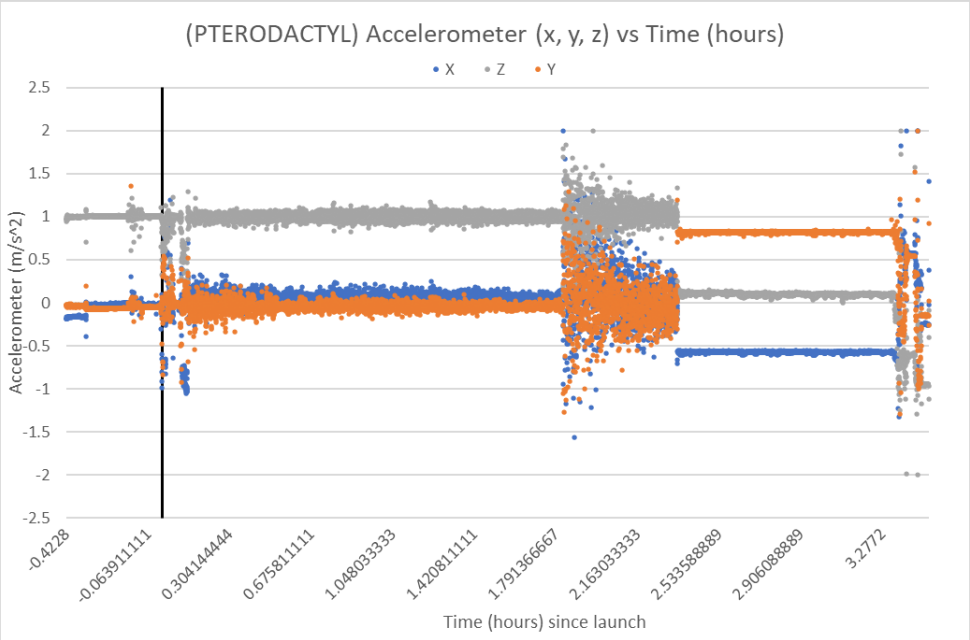


Figure 4: PTERODACTYL Accelerometer vs Time

Figure 4 depicts the values captured by the accelerometer plotted vs the time of flight. An easy way to comprehend this graph is that where there is more "noise," meaning the data points are spaced out more, such as at about 1.9 hours, there is a lot of change in motion payload is experiencing. This change could be many things, the payload swinging, spinning, falling, or an abrupt change in position. During the flight the payload experienced all of these, at T=0 (at launch) the payload was at rest then was suddenly let go, this caused an increase in noise in the data. This also happened at about 1.9 hours, this is the approximate time when the balloon popped, which caused a very abrupt change in the payload's motion. There is next to no noise roughly from 2.3 hours to 3.3 hours. This is the time that the payload was in a tree, almost completely still. The noise here is predicted to be from the sway of the tree in the wind.

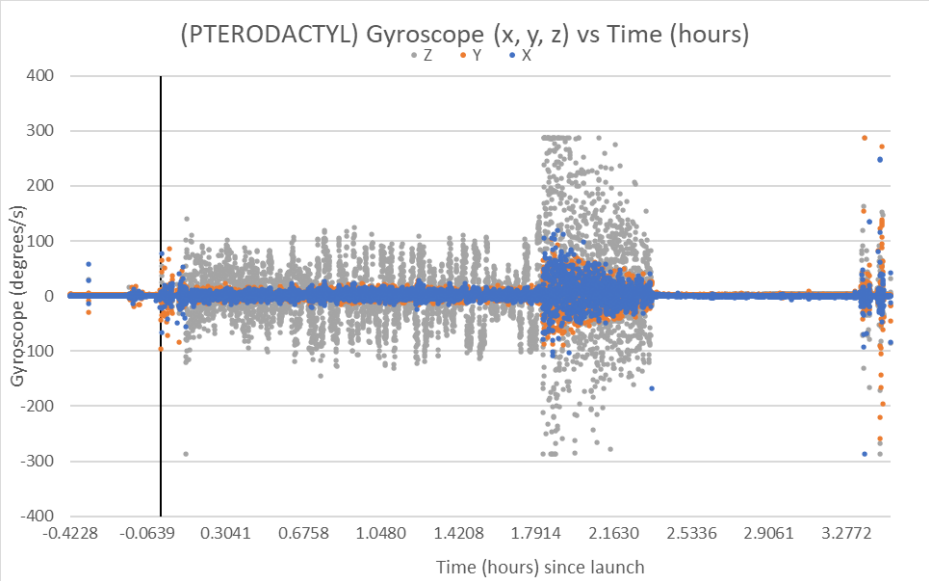


Figure 5: PTERODACTYL Gyroscope vs Time

Figure 5 depicts the change in angle relative to the time during the flight. Any noise seen can be attributed to the payload rotating on any of its axes. The reason the z-axis has the most noise is because it is the axis that is pointing straight up, which is the axis that the payload and balloon spins about. During the entire flight, it can be seen that the entire structure is spinning on its z-axis. There is a spike in noise at around 1.9 hours because that is concluded to be approximately when the balloon popped.

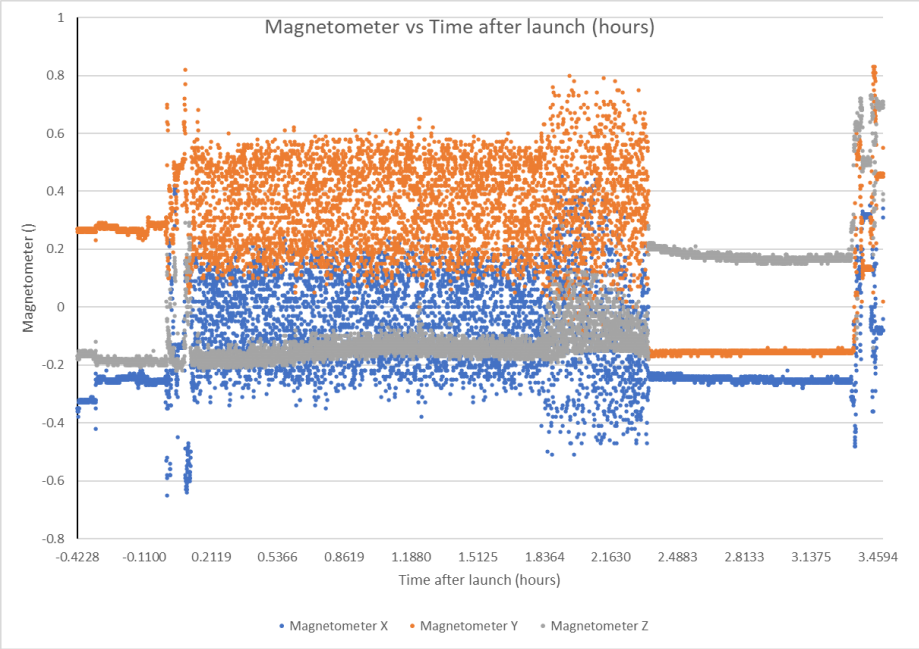
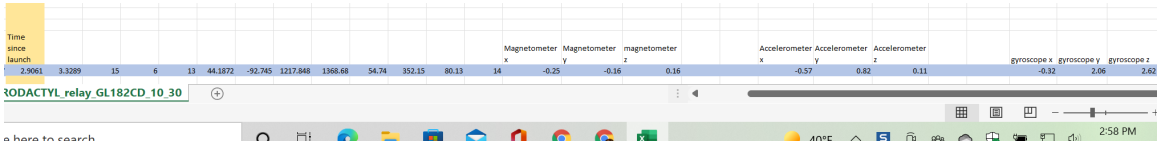


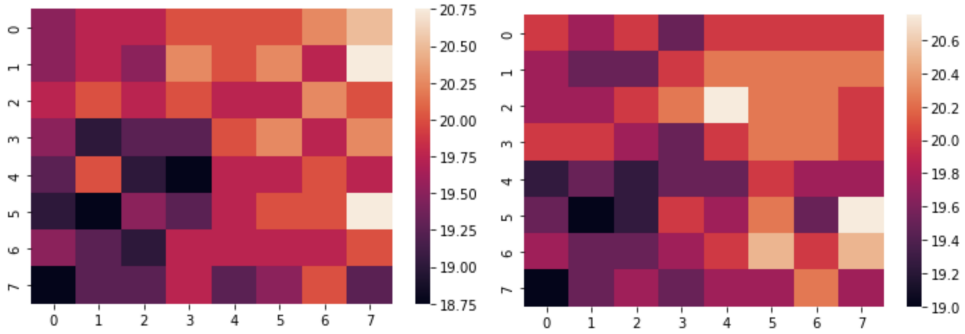
Figure 6: PTERODACTYL Magnetometer vs Time

Figure 6 is showing the data collected from the magnetometer relative to the time in the flight. Each axis collects its own data, the strength of the magnetic field in the direction along that axis. It is predicted that the x and y axis have a lot of noise due to the swinging of the payload, and because the x-y plane is generally parallel to the surface of the Earth, if the direction of the x or y axis swing from pointing to the sky to the ground constantly, there will be a constant change in the readings for the magnetic field. The z axis points straight up, and the sensor pointing along the z axis does not veer off that axis that much, at least until the balloon pops, which is when the payload flipped over which can be seen in the camera footage. Other than the time the payload is falling, there remains next to no change in the z axis magnetic field.

Payload Final Position



Given the data from all the sensors, we can determine the final position of the payload was on its side.



(a) **(b)**

Figure 7: QUIIC GridEye Sensor
(a) Temperature (°C) at 73.25 sec after Bootup
(b) Temperature (°C) at 74.25 sec after Bootup

The actual data received during the flight is unusable given that for each point on the 8 by 8 grid, the temperature recorded fluctuates frequently between extremely high and low temperatures which could not be feasibly recorded in flight. Figures 7a and 7b are from data collected during a pre-flight test. They are examples of what we originally expected to receive if the data was correctly measured and recorded. Only the first 2 plots are shown for brevity.

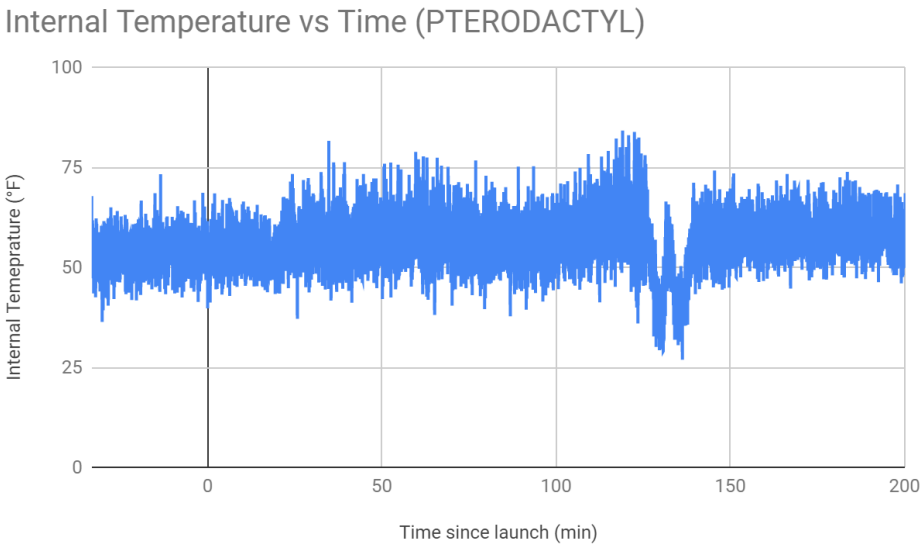


Figure 8: PTERODACTYL Internal Temperature vs Time

The internal temperature was not recorded on our team's payload, and so figure 8 shows data detected and stored by the relay payload on the same stack. The range in values seems unreasonable (the noise is very high), and so this data is relatively questionable.

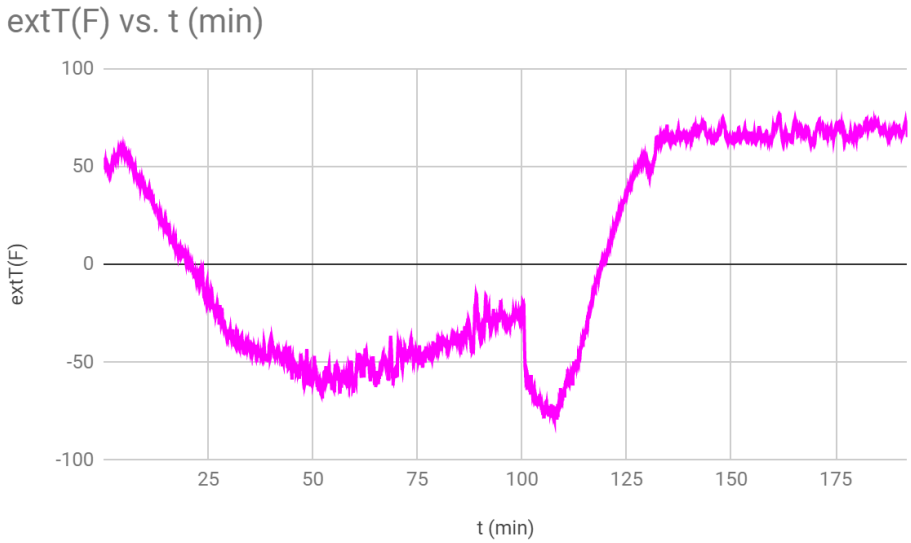


Figure 10: PTERODACTYL External Temperature vs Time

As shown in Figure 10, temperature decreased then increased during ascension of the balloon, then steadily increased upon descent. This follows the temperature initially expected, and thus is relatively reliable.

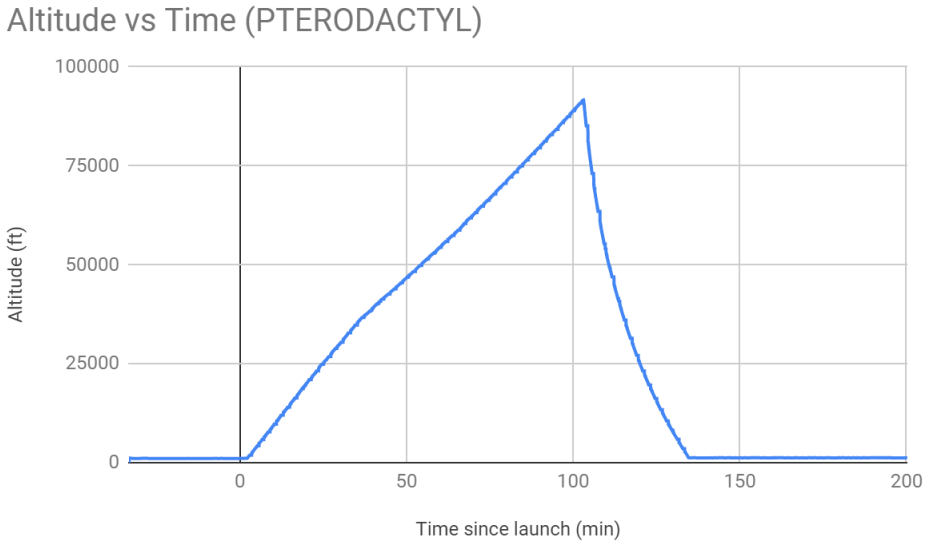


Figure 11: PTERODACTYL Altitude vs Time

The altitude was not recorded on our team's payload, and so figure 11 shows data detected and stored by the relay payload on the same stack.

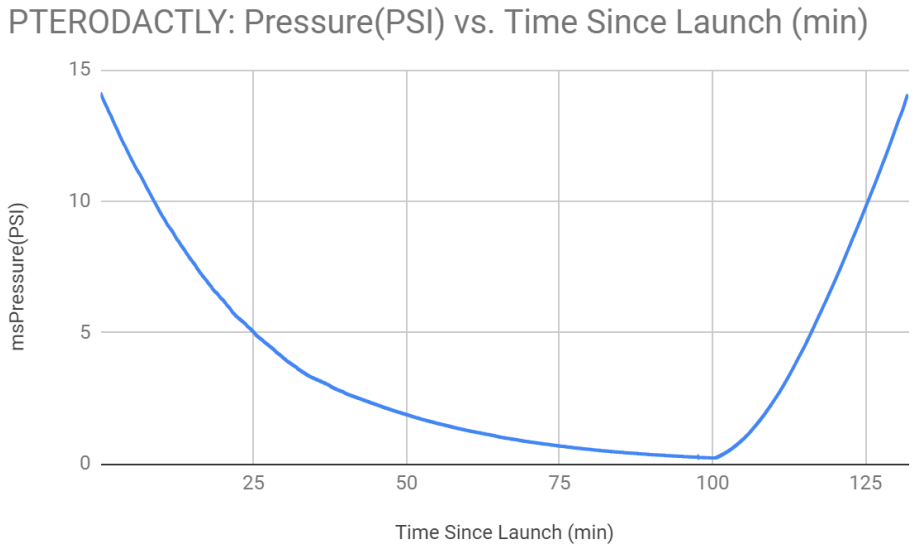


Figure 12: PTERODACTYL Pressure vs Time

Figure 12 displays the relationship between pressure and time since launch measured by the PTERODACTYL. This data is reliable given that it was correctly measured and corresponds with the pressure values detected in other stacks.

Neulog Data

Note: All Neulog graphs were graphed in Excel, and there are no lines (the only exception being the magnetic field graph). It only appears as a line because of the multitude of data collected every second. This can be seen in this graph, where the predicted altitude jumps as the equation describing it changes at its maximum.

Figure 13: Neulog Altitude vs Time

The altitude vs time demonstrates the steady ascent of the payload and the sharp, quicker descent. The slope becomes smaller during the ascent, highlighting the effects of the atmosphere as pressure increases and the parachute can produce more drag. There is little else to take away from this graph except that it suggests a fairly nominal flight path.

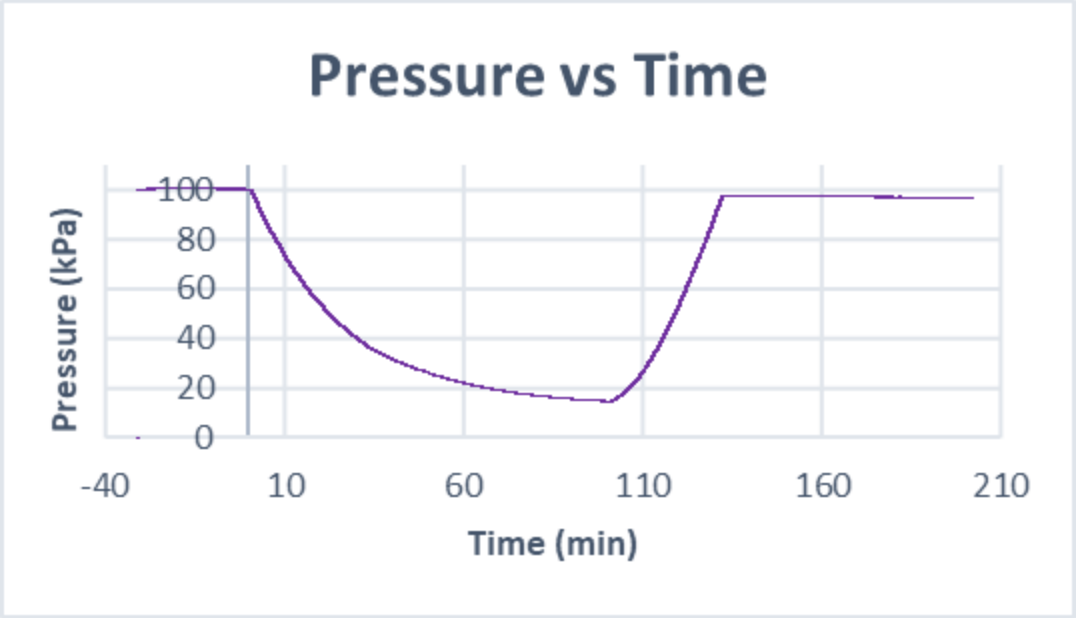


Figure 14: Neulog Pressure vs Time

Figure 14 represents pressure during the flight, and the values are what would be expected, with steadily decreasing pressure until burst where it rapidly approaches the surface level. This graph does suggest that at 90000 ft, the peak of our flight, the pressure was more than 15 kPa. This is far greater than expected values. However, the PTERODACTYL pressure sensor gave values of 1.5 kPa when converted, suggesting that the Neulog pressure sensor is simply inaccurate at near-vacuum pressures.

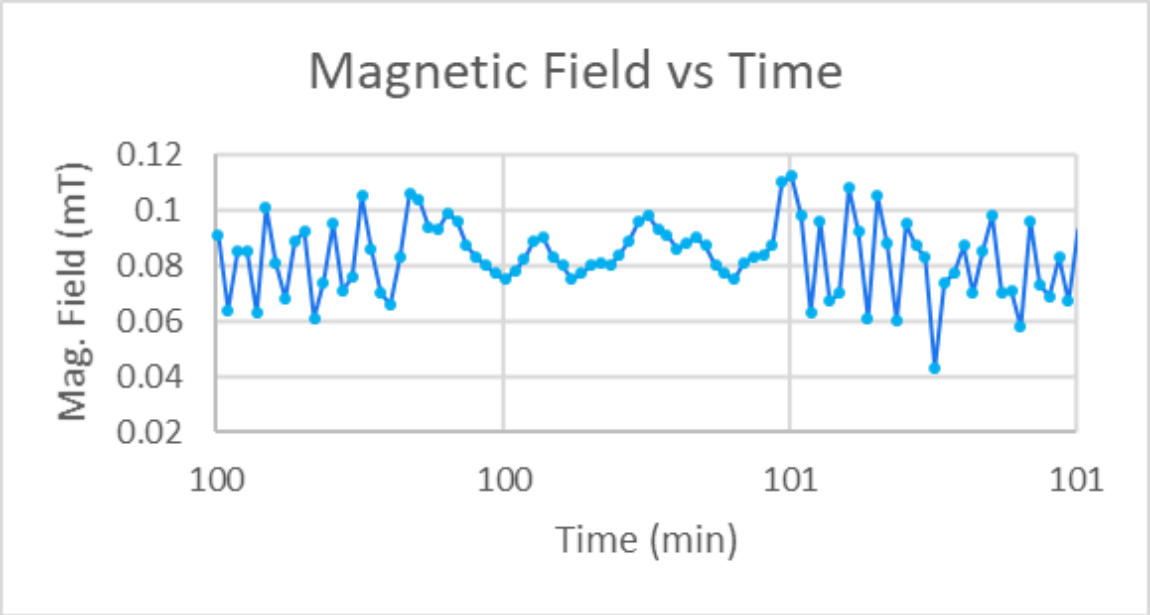


Figure 15: Neulog Magnetic Field vs Time

This graph represents the magnetic field at burst, at T plus 100.24 minutes into flight. For the most part, the magnetic field readings appear to jump up and down with each data point, regardless of the rotation or ascent rate of the payload. There is a line connecting the data to better represent this oscillatory behavior. However, the readings became far more consistent and observable in the minute after the burst. This may be because the parachute was not fully deployed yet, so the whole payload was in an almost “free fall” state, like when a car crests over a hill and you feel weightless in your seat. When the parachute fully deployed, the normal force between the payload and the Neulog module caused the readings to become more varied as before.

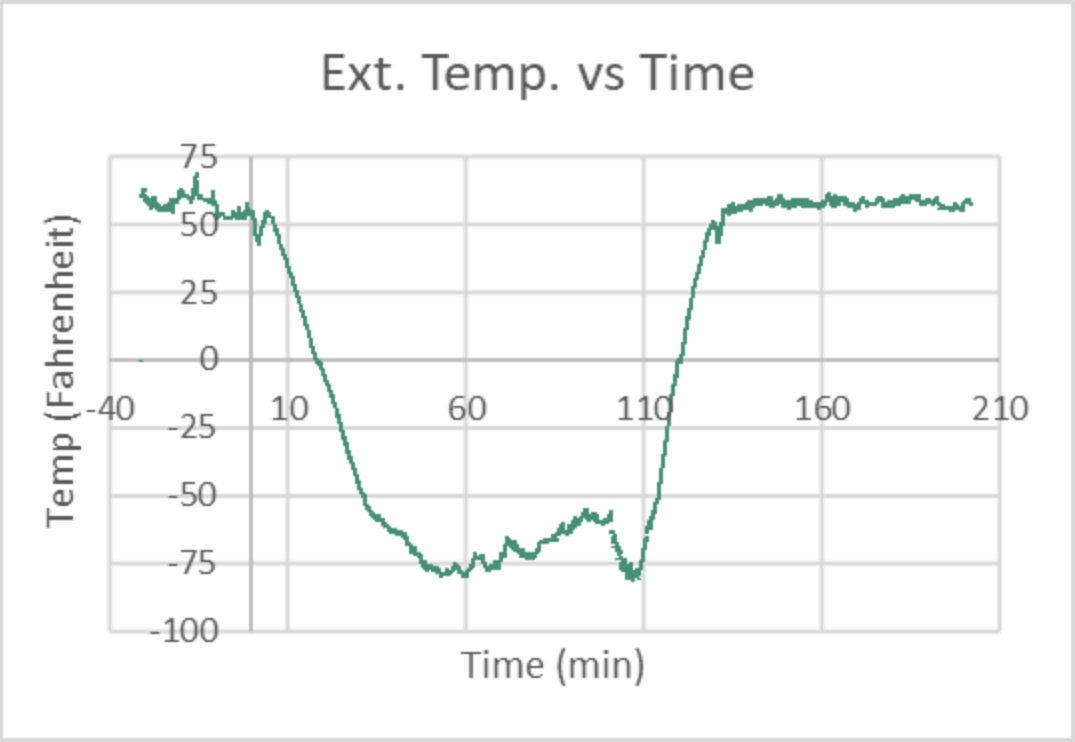


Figure 16: External Temperature vs Time

Figure 16 illustrates how the ambient, external temperature changes with time. This graph should be used in conjunction with Figure 17 to help understand the effects the different levels of the atmosphere have on temperature. This graph demonstrates that 50-60 minutes into flight, the temperature leveled out and began to increase. This change marks the entrance of the stratosphere, and the ozone layer. This change is seen with greater clarity in Figure 17. Comparing Figure 16 to Figure 18 also demonstrates the importance of sunlight with respect to temperature, with far higher readings even in the white cardstock.

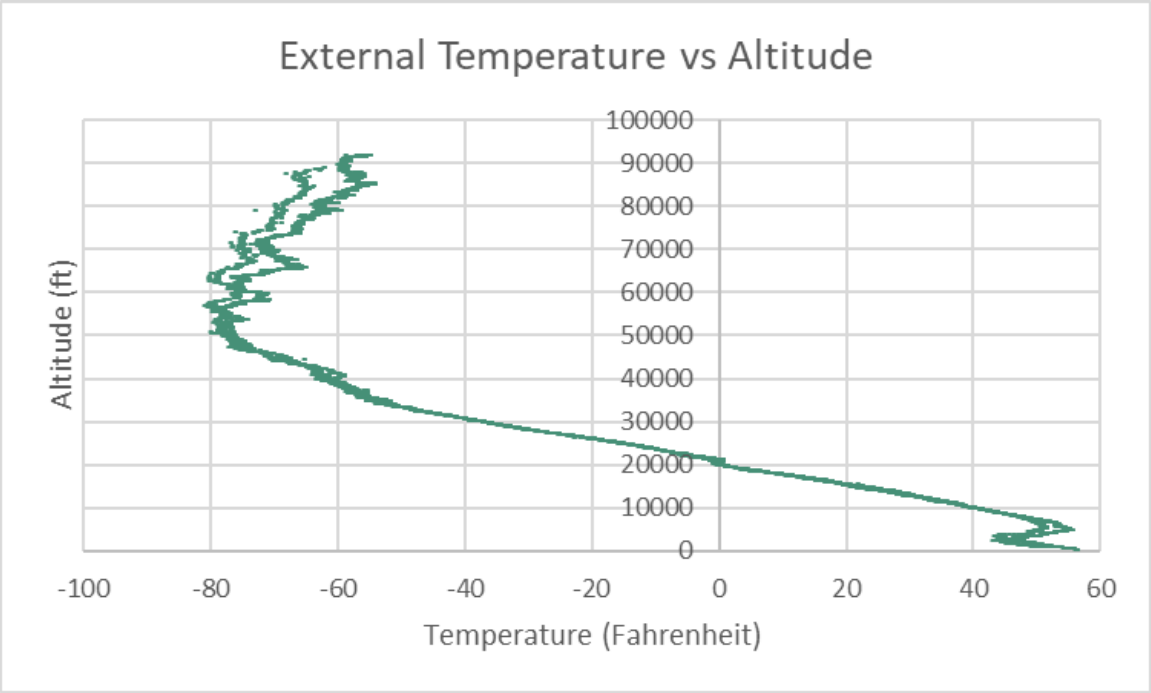


Figure 17: Neulog: Altitude vs External Temperature

The Altitude vs Ext. Temperature graph highlights how consistent temperature is with respect to altitude. The temperature on ascent and descent is almost identical for a given altitude, despite the higher velocities of re-entry. It also shows the change in temperature at approximately 50000 ft that mirrors the change in temperature at T + 50 minutes in Figure 16. This is likely the beginning of the stratosphere and the ozone layer, where ozone is formed because of the absorption of UV light, releasing heat.

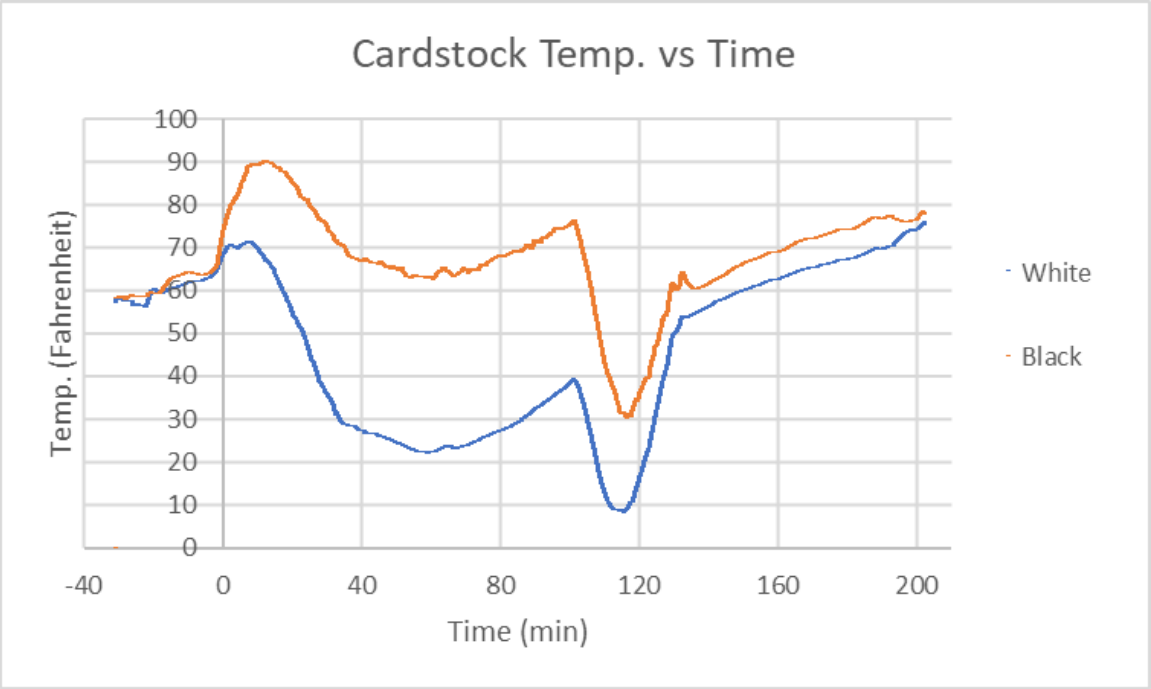


Figure 18: Neulog Cardstock Temperature vs Time

Figure 18 represents the emphasis of our mission: comparing temperature with respect to light absorption in the upper atmosphere. It is very encouraging to see such a significant difference between the two thermometers throughout the flight, supporting our hypothesis. These values also appear to accurately reflect some of the milestones during the flight, such as entering the stratosphere, burst, subsequently exiting the stratosphere, and landing. It's interesting that the values immediately start increasing on take off, particularly the black cardstock. This is likely because the top of the payload was in more direct sunlight after launch.

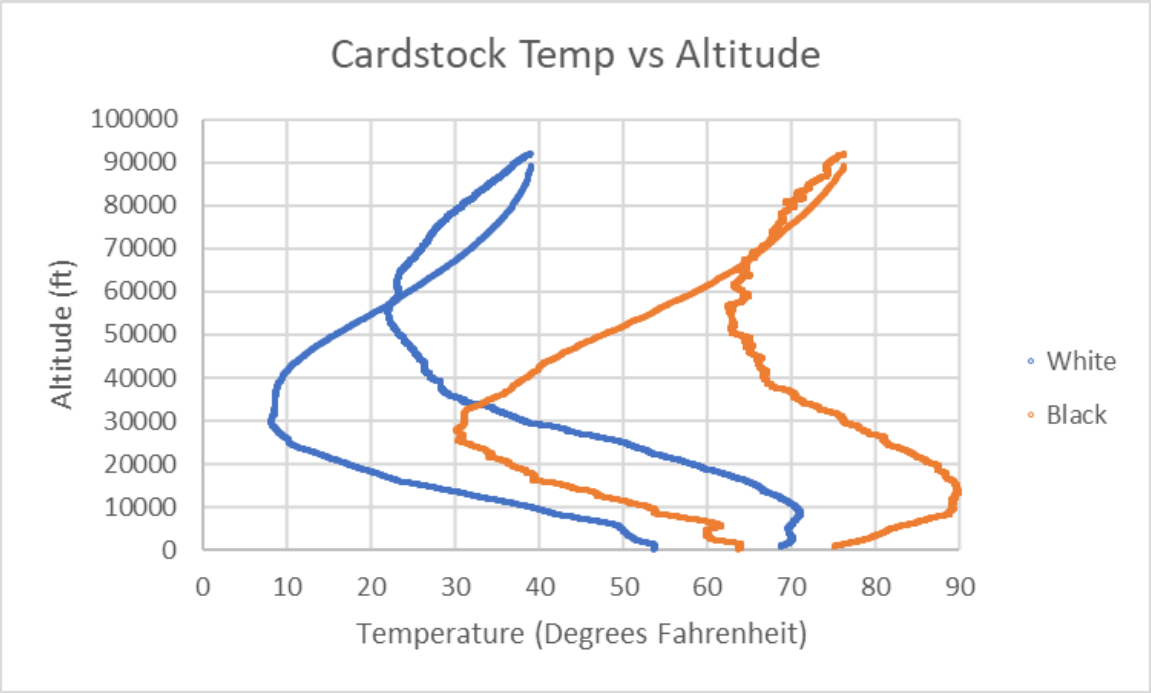


Figure 19: Neulog Altitude vs Cardstock Temperature

Figure 19 provides a different lens to analyze this experiment, this time in comparison to altitude. The higher temperature readings at the surface for the white and black cardstock, 68º and 75º respectively, are the launch temperatures, and the lower temperatures are from when the payload touches down. The behavior of the temperature in each reflects the pattern in Figure 17, with temperatures rising as the payload enters the atmosphere. The most curious part of this graph is that temperature on descent is higher than the ascent temp in the upper atmosphere, and lower than ascent in the lower atmosphere. This is a difference from the external temperature. Whether this is an effect of the cardstock or simply that they were housed within the payload is unknown, though it is presumably the latter. Overall, Figures 18 and 19 graphically represent the strong relationship between light absorption and temperature throughout the atmosphere and stratosphere.

GPS Tracking Data

Figure 20: 2D Map of Balloon Flight Path

Figure 20 is the 2D map of the balloon flight path. Labeled are the launch site, point where the balloon bursts, and landing site. This map was generated on StratoCom, which uses Google Maps as it's main carrier. Google Maps has a user-friendly interface which makes it easy to chart the flight path. However, Google Maps is not suited for this kind of plotting. Google Maps can only satisfy the longitude-latitude aspects of the flight path, but altitude is lost. With a 2D map, it is difficult to visualize the altitude.

Figure 21: 3D Map of Flight Path from Launch Site

Figure 21 is a 3D map of the flight path focused on the launch site. The top edge of the "wall" is the actual path of the balloon. The base of the map is a satellite image, which captures more detail than default Google Maps. Here you can see the grassy clearing used for launch, and the tree we had to avoid. The 3D map also allows for easy visualization of all three components of location: longitude, latitude, and altitude. The start of the flight path is on ground level. Then the stack is walked over to the clearing and let go, the wind then carries the stack northeast as the balloon rises.

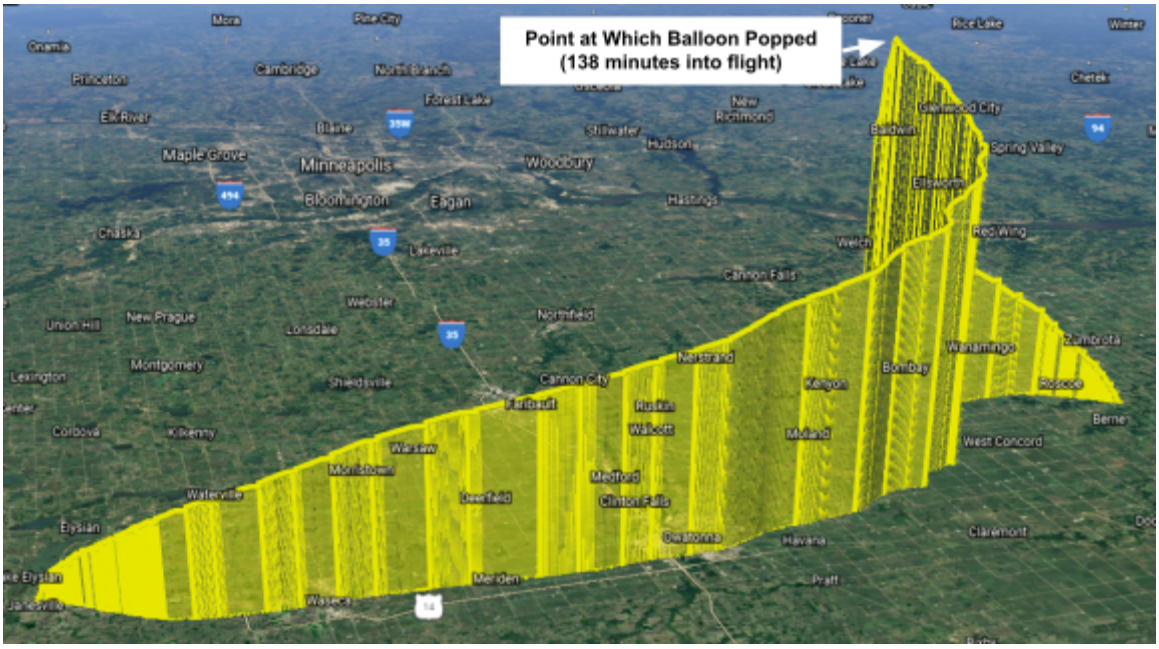


Figure 22: 3D Map of Flight Path

Figure 22 is a 3D rendering of the balloon flight path. In comparison to the 2D map, the balloon burst point is easily visualized. The stack is rising at a constant rate, but when the balloon bursts, it begins its descent, thus maximum altitude is reached when the balloon pops. Due to the flight path shape, it is difficult to capture longitude, latitude and altitude without obstructing the overall shape. This is one of the downfalls of 3D mapping. There isn't an angle where the ascent or descent will not be obstructed while preserving the longitude, latitude, and altitude visuals.

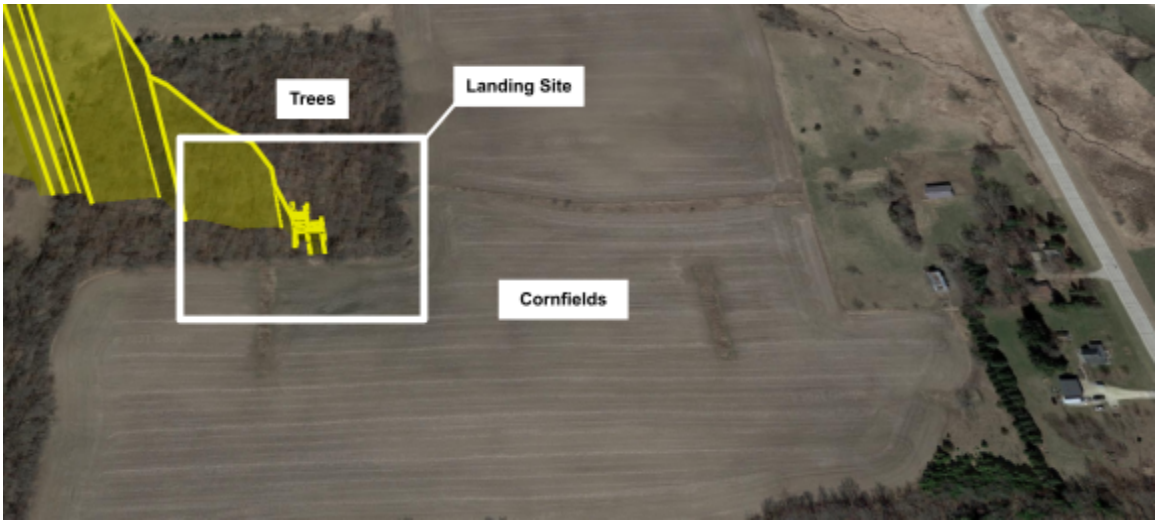


Figure 23: 3D Map of Flight Path at Landing Site

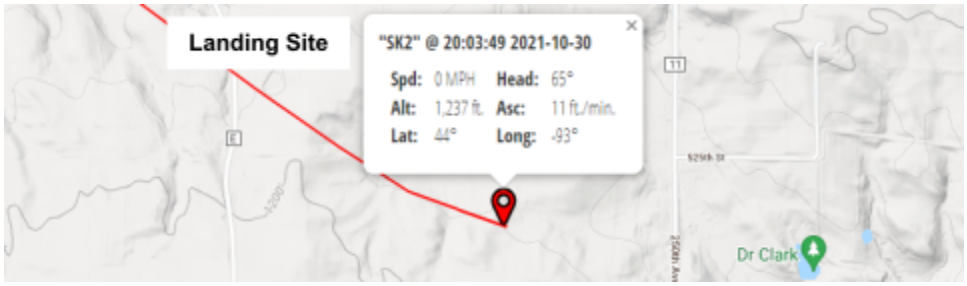


Figure 24: 2D Map of Flight Path at Landing Site

Figure 23 and 24 are a 3D and 2D rendering of the flight path centered on the landing site. In Figure 24, the GPS values are listed, but the topography is not easily interpreted. In Figure 23, the landscape is clear, though the location of the site is obscure. Additionally, the descent of the balloon as it falls into the treeline is clear. Contrastingly, in Figure 24, the altitude is not pictured and there is no visual cue to signal presence of trees. In regards to the recovery of the payloads, both maps are useful. The 2D map is useful when traveling to the general area, but the 3D map is more useful in locating the payload and viewing what conditions it has landed in. Our team went onto the property blind, not knowing where the payloads were or what landmarks were near it. Had we used the 3D map on site, we could have known beforehand that the payload had fallen into trees. This information would have allowed us to call for assistance sooner and make the overall retrieval process more efficient.

Camera Recordings

When wrapping the payload with black tape for flight, the hole carved out for the camera was partially covered in tape, causing our footage to be blocked on the side. A screenshot of the footage is shown below. A screenshot from the footage of Group D's camera is also shown below for reference of what our camera should have captured. Also, footage from our camera only began recording approximately 66 minutes into the flight, which is strange given that the camera was on the entire time and did not have any issues recording in our preflight tests.

Figure 25: Footage from Our Camera during Ascension

Figure 26: Footage from Group D's Camera during Ascension

Based on the camera footage along with the information from figure 5, the payload seems to be rotating at a rate of around 1 rotation every 4 seconds, although this is very much variable and the payload switches direction of rotation very frequently. All of our camera footage is stored online in a Google Drive folder.

11.0 Conclusions and Lessons Learned

Summarize what your team has learned and what you would have done differently if you had a chance to do this again (about 0.5 to 1 page of text). Write some "Words of Wisdom" (2 or 3 sentences or bullet points) as advice to a future ballooning class about what worked well and what could have worked better. (Rev C)

Communication amongst teammates and planning are essential to accomplish the goal in mind. When assigning responsibilities to team members it is much more efficient to play to everyone's strengths, although it is always valuable to take time if possible to learn new skills and concepts outside of everyone's comfort zones. Planning is important because of the deadlines incorporated with projects like this. If we would have planned a little more in depth and abided by the plan we most likely could have had more time to run tests and other "housekeeping" tasks. Another important idea is to keep records of everything that the team did, whether that was changing the design of the payload or what time the payload was launched. Detailed records will be a helpful reference tool when either reflecting on the overall performance of the payload, or considering performing more tests. A record will encourage a more streamlined evolution of the project.

If there was going to be a second launch, we would have tested the GridEye Sensor more thoroughly and figured out a more efficient way to store and display the data received from it. We would also do more testing ont he camera, given that it only begen recording footage partway into the flight.

Advice to a future team would be to try to get everyone involved and working on the payload and share as much information with your team as possible. Knowing as much as you can about your payload and the process in which you built it will be very beneficial in the future.

12.0 References

1. Verhage, Paul. "Near Space: How Some Hobbyists are Getting Around the Difficulties Associated with Amateur Space Exploration, Part I." Near Space February (2004): 64-69.
2. “Where Is Space?” NESDIS, www.nesdis.noaa.gov/news/where-space.
3. “Stratosphere.” Stratosphere - an Overview | ScienceDirect Topics, www.sciencedirect.com/topics/earth-and-planetary-sciences/stratosphere.
4. “Do Airplanes Fly in the Stratosphere? (Commercial, Private Jets, Light Aircraft).” EXECUTIVE FLYERS, 27 Mar. 2021, executiveflyers.com/do-airplanes-fly-in-the-stratosphere/.
5. “U.S. Standard Atmosphere.” Engineering ToolBox, www.engineeringtoolbox.com/standard-atmosphere-d_604.html.
6. Federico GentileFederico Gentile 19522 silver badges55 bronze badges, and user2821user2821 5. “Does Magnetic Deviation Depend on Altitude?” Earth Science Stack Exchange, 1 Apr. 1965, earthscience.stackexchange.com/questions/9610/does-magnetic-deviation-depend-on-altitude.

13.0 Appendix: Program Listings

The PTERODACTYL portion of the code defines all the pins, ports, and breakout boards on the custom board. This portion of the code also creates strings of data that are going to be transmitted later to comms and the SD card. The PTERODACTYL portion of the code also sets up all of the teensy functions and defines all the variables to be later used and referenced.

The code for comms was provided for the group to use while our group only had to make slight changes. The changes our group made added a string of data that was transmitted to the ground using the SatCom. Our group chose to send data regarding pressure, gridEYE value, and GridEYE index so that we could confirm that the onboard electronics were working properly and that our payload was ascending or descending based on the change in pressure. The comms code also communicates with the Xbee system to tell the ground systems that all onboard systems are working or not working using a string of ones and zeros.

The SD portion of the code was used to write all the sensor data to an SD card so that it could be viewed later. Our group chose to record a vast amount of data to our SD

card such as pressure, longitude, latitude, altitude, temperature, time, and values from our GridEYE breakout board. Our data was set to record data every one second so that we could plot major changes and have enough data if something were to happen.

The final portion of the code is the sensor code. This section defines all of the sensor functions so that the sensors work properly. The code also uses formulas to change those values so that the data is readable and can be understood. The sensor code also defines the ground data and comms data strings which compile all of the data our group wanted to record into a single string that can be called and printed later.

PTERODACTYL CODE:

```
// A heavily modified version of the PTERODACTYL flight code for use by the Fall 2020 AEM 1301 class as a platform engine.
// PTERODACTYL designed by and code written by Andrew Van Gerpen.
// Modifications made by Paul Wehling.
// Last updated 10/25/2020
//
// Rory C, Ray L, Ryan L, Tina L, Mel P


// Main code block, contains setup(), loop(), and code to check the initial states of the slide switches and shorts
//  as well as variable declarations and logical code.
// Students will want to modify many of the functions here to add their own data logging and calls to new sensor functions.

#define fixLED 26              // LED to indicate GPS fix
#define ppodLED 24
#define xbeeLED 27              // LED to indicate xbee communication
#define sdLED 25
#define serialBAUD 9600         // when using arduino serial monitor, make sure baud rate is set to this same value

#define ppodSwitchPin 30
#define satSwitchPin 31
#define commsSwitchPin 32
#define setAltSwitch 28
#define altSwitch 29
#define baroSwitchPin 9
#define id1SwitchPin 20
#define id2SwitchPin 21
#define id3SwitchPin 22
```

```
#define id4SwitchPin 23
#define pullBeforeFlightPin 16

int rfd900 = 1; // set true if you want this thing to operate with a rfd900 on serialX (declare above)
int ppod = 1; // set true if you want this thing to operate as ppod flight computer
int satCom = 1; // set true if you want to activate flight by waving a magnet over the IMU
int setAltVal = 1;
int altVal = 1;
int baroOn = 1;
int id1On = 1;
int id2On = 1;
int id3On = 1;
int id4On = 1;
int pullOn = 1;

// Students will want to modify header to include the new data they're logging.
String header = "Date, Hour, Minute, Second, Lat, Lon, Alt(ft), AltEst(ft), intT(F), extT(F), msTemp(F), msPressure(PSI), time since bootup (sec), magnetometer x, magnetometer y, magnetometer z, accelerometer x, accelerometer y, accelerometer z, gyroscope x, gyroscope y, gyroscope z, gridEye Index, gridEye Value, gridEyeArray[0], gridEyeArray[1], gridEyeArray[2], gridEyeArray[3], gridEyeArray[4], gridEyeArray[5], gridEyeArray[6], gridEyeArray[7],gridEyeArray[8], gridEyeArray[9], gridEyeArray[10], gridEyeArray[11], gridEyeArray[12], gridEyeArray[13], gridEyeArray[14], gridEyeArray[15], gridEyeArray[16], gridEyeArray[17], gridEyeArray[18], gridEyeArray[19], gridEyeArray[20], gridEyeArray[21], gridEyeArray[22], gridEyeArray[23], gridEyeArray[24], gridEyeArray[25], gridEyeArray[26],gridEyeArray[27], gridEyeArray[28], gridEyeArray[29], gridEyeArray[30], gridEyeArray[31], gridEyeArray[32], gridEyeArray[33], gridEyeArray[34], gridEyeArray[35], gridEyeArray[36], gridEyeArray[37], gridEyeArray[38], gridEyeArray[39], gridEyeArray[40], gridEyeArray[41], gridEyeArray[42], gridEyeArray[43], gridEyeArray[44], gridEyeArray[45], gridEyeArray[46], gridEyeArray[47], gridEyeArray[48], gridEyeArray[49], gridEyeArray[50], gridEyeArray[51], gridEyeArray[52], gridEyeArray[53], gridEyeArray[54], gridEyeArray[55], gridEyeArray[56], gridEyeArray[57], gridEyeArray[58], gridEyeArray[59], gridEyeArray[60], gridEyeArray[61], gridEyeArray[62], gridEyeArray[63]";
unsigned long int dataTimer = 0;
unsigned long int dataTimerIMU = 0;
unsigned long int ppodOffset = 0;
int dataRate = 1000; // 1000 millis = 1 second
int dataRateIMU = 250; // 250 millis = .25 seconds
int analogResolutionBits = 14;
int analogResolutionVals = pow(2,analogResolutionBits);
```

```
float pressureBoundary1;
float pressureBoundary2;
float pressureBoundary3;
float pressureOnePSI;
float msPressure = -1.0;
float msTemperature = -1.0;
float altitudeFt = -1.0;
unsigned long int fixTimer = 0;
bool fix = false; // determines if the GPS has a lock

//////////////////// Sensor Global Variables ////////////////////////////

float thermistorInt;
float thermistorExt;
float magnetometer[3]; // {x, y, z}
float accelerometer[3]; // {x, y, z}
float gyroscope[3]; // {x, y, z}

float altitudeFtGPS;
float latitudeGPS;
float longitudeGPS;
String data;
String groundData;
String IMUdata;
String satData;


String exclamation = "!"; // this needs to be at the end of every XBee message
String groundCommand;
String xbeeID = "NULL";
String xbeeProID = "AAA";
unsigned long int xbeeTimer = 0;
unsigned long int xbeeRate = 5000; // 10000 millis = 10 seconds
String xbeeMessage; // This saves all xbee transmissions and appends them to the data string

void setup() {

  Serial.begin(serialBAUD); //define baud rate in variable decleration above
  Serial.println("Serial online");
  pinMode(fixLED,OUTPUT);
  pinMode(xbeeLED,OUTPUT);
  pinMode(sdLED,OUTPUT);
  pinMode(ppodLED,OUTPUT);
```

```
pinMode(13,OUTPUT);
pinMode(setAltSwitch, INPUT_PULLUP);
pinMode(altSwitch, INPUT_PULLUP);
checkSwitches(); // slide and button switch statuss

if(id1On==0)xbeeID="UMN1";
if(id2On==0)xbeeID="UMN2";
if(id3On==0)xbeeID="UMN3";
if(id4On==0)xbeeID="UMN4";

Serial.print("starting OLED setup... ");
oledSetup();
Serial.println("OLED setup complete");

Serial.print("starting IMU setup... ");
imuSetup();
updateIMU();
Serial.println("IMU setup complete");

Serial.print("starting Altimeter setup... ");
if(baroOn==1)msSetup();
else{ updateOled("MS5611\nOffline.");
  delay(2000);
}
Serial.println("Altimeter setup complete");

Serial.print("starting SD setup... ");
sdSetup();
Serial.println("SD setup complete");

Serial.print("starting xbee setup... ");
xbeeSetup();
Serial.println("xbee setup complete");

Serial.print("starting ublox setup... ");
ubloxSetup();
Serial.println("ublox setup complete");

Serial.print("starting gridEye setup... ");
gridEyeSetup();
Serial.print("gridEye setup complete");
```

```
  pressureToAltitudeSetup();
  logData(header);
  if(pullOn==0) pullPin();
}

void loop() {
  updateData();
}

////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////// Functions  ////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////

void pressureToAltitudeSetup()
{
  float h1 = 36152.0;
  float h2 = 82345.0;
  float T1 = 59-.00356*h1;
  float T3 = -205.05 + .00164*h2;
  pressureBoundary1 = (2116 * pow(((T1+459.7)/518.6),5.256));
  pressureBoundary2 = (473.1*exp(1.73-.000048*h2)); // does exp function work??
  pressureBoundary3 = (51.97*pow(((T3 + 459.7)/389.98),-11.388));
}

void pressureToAltitude(){
  //float pressurePSF = (pressureOnePSI*144);
  float pressurePSF = (msPressure*144);

  float altFt = -100.0;
  //UNCOMMENT WHEN RELIABLE PRESSURE SENSORS ARE ON BOARD
  if (pressurePSF > pressureBoundary1)// altitude is less than 36,152 ft ASL
  {
    altFt = (459.7+59-518.6*pow((pressurePSF/2116),(1/5.256)))/.00356;
  }
  else if (pressurePSF <= pressureBoundary1 && pressurePSF > pressureBoundary2) //
altitude is between 36,152 and 82,345 ft ASL
  {
    altFt = (1.73-log(pressurePSF/473.1))/.000048;
  }
  else if (pressurePSF <= pressureBoundary2)// altitude is greater than 82,345 ft ASL
  {
    altFt = (459.7-205.5-389.98*pow((pressurePSF/51.97),(1/-11.388)))/-.00164;
```

```
  }
  else{altFt = -1.0;}

  altitudeFt = altFt;
  if(baroOn==0)altitudeFt = -1.0;
}

void updateData(){
  updateUblox();
  updateXbee();

  if(millis() - dataTimerIMU > dataRateIMU){
    dataTimerIMU = millis();
    updateIMU();
  }
  if(millis() - dataTimer > dataRate){
    dataTimer = millis();
    if(fix == true){
      digitalWrite(fixLED,HIGH);
    }
    digitalWrite(sdLED,HIGH);
    delay(30);
    digitalWrite(sdLED,LOW);
    digitalWrite(fixLED,LOW);
    pressureToAltitude();
    updateThermistor();
    if(baroOn==1) updateMS(); //Not every payload has one
    updateIMU();
    updateDataStrings();
    xbeeMessage="";
    updateGridEye();
    updateAllGridEye();
  }
}

void checkSwitches(){
  pinMode(ppodSwitchPin, INPUT_PULLUP);
  pinMode(satSwitchPin, INPUT_PULLUP);
  pinMode(commsSwitchPin, INPUT_PULLUP);
  pinMode(baroSwitchPin, INPUT_PULLUP);
  pinMode(id1SwitchPin, INPUT_PULLUP);
  pinMode(id2SwitchPin, INPUT_PULLUP);
  pinMode(id3SwitchPin, INPUT_PULLUP);
```

```
  pinMode(id4SwitchPin, INPUT_PULLUP);
  pinMode(pullBeforeFlightPin, INPUT_PULLUP);

  ppod = digitalRead(ppodSwitchPin);
  rfd900 = digitalRead(commsSwitchPin);
  satCom = digitalRead(satSwitchPin);
  baroOn = digitalRead(baroSwitchPin);
  id1On = digitalRead(id1SwitchPin);
  id2On = digitalRead(id2SwitchPin);
  id3On = digitalRead(id3SwitchPin);
  id4On = digitalRead(id4SwitchPin);
  pullOn = digitalRead(pullBeforeFlightPin);
}
```

COMMS CODE:

```
// Contains all of the setup information and functions related to XBee communication.
// Students should not have to use or add any code here unless adding new radio commands
// Some functions for converting numbers to the proper binary elements have been kept in
case they're needed fro the SatCom system

#include <RelayXBee.h> //Library can be found at
https://github.com/MNSGC-Ballooning/XBee

#define xbeeSerial Serial5        // Serial communication lines for the xbee radio -- PCB pins:
Serial3

String xbeeSendRequest = "Readyfordata";
String a0;

RelayXBee xbee = RelayXBee(&xbeeSerial, xbeeID);

float testFloat = 128.0;
int testInt = 1280;
int sentBytes = 0;

void xbeeSetup(){
  updateOled("Xbee Radio\nInit...");
  char xbeeChannel = '1';
  xbeeSerial.begin(XBEE_BAUD);
  xbee.init(xbeeChannel); // Need to make sure xbees on both ends have the same identifier.
"AAAA"
```

```
  xbee.enterATmode();
  xbee.atCommand("ATDL0");
  xbee.atCommand("ATMY1");
  xbee.exitATmode();
  Serial.println("Xbee initialized on channel: " + String(xbeeChannel) + "; ID: " + xbeeID);
  updateOled("Xbee\nChannel: " + String(xbeeChannel) + "\nID: " + xbeeID);
  delay(2000);
}

// Function required to convert floats into 4 byte hex vals
typedef union
{
  float number;
  uint8_t bytes[4];
}FLOATUNION_t;

// Function required to convert ints into 2 byte hex vals
typedef union
{
  int number;
  uint8_t bytes[2];
}INTUNION_t;

void sendFloat(float sendMe){
  FLOATUNION_t myFloat;
  myFloat.number = sendMe;
  for (int i=0; i<4; i++)
  {
    // Send converted floats here
  }
  sentBytes += 4;
}

// function that takes in an int value and prints it to the xbeePro serial as 2 bytes.
void sendInt(int sendMe){
  INTUNION_t myInt;
  myInt.number = sendMe;
  for (int i=0; i<2; i++)
  {
    // Send converted ints here
  }
  sentBytes += 2;
}
```

```
// rounds out the 21 byte xbee pro transmission with "0" bytes to fill packet
void finishSend(){
  for (int i=0; i<(21-sentBytes); i++){
    // Send completed packet here
  }
}

void updateXbee(){ // This is disgusting

//    For new satCom relay code
//    a0 = xbeeSerial.readString();
//    xbeeSerial.flush();
//    //Serial.println(a0);
//    if (a0 == xbeeSendRequest) {
//   delay(500);
//   xbeeSerial.print(xbeeID + "," + satData + "!" );
//   Serial.println(xbeeID + "," + satData + "!");

  if((millis() - xbeeTimer) > xbeeRate){

    xbeeTimer = millis();

    xbeeSerial.print(xbeeID + "," + groundData + "!");

    digitalWrite(xbeeLED,HIGH);
    delay(80);
    digitalWrite(xbeeLED,LOW);
    xbeeMessage = "DATA STRING TRANSMITTED";
  }

 if (xbeeSerial.available() > 10){
  groundCommand = xbeeSerial.readString();
  if(groundCommand.startsWith(xbeeID))
  {
    groundCommand.remove(0,xbeeID.length()+1);
    xbeeSerial.println(xbeeID + ", " + interpretMessage(groundCommand) + "!");
    xbeeMessage = xbeeID + " RECEIVED: " + groundCommand + "; SENT: " +
interpretMessage(groundCommand);
  }
 }
}
```

```
String interpretMessage( String myCommand ){

  if(myCommand.startsWith("ALT"))
  {
   xbeeMessage = "Altitude calculated from pressure: " + String(altitudeFt);
  }
  else if(myCommand.startsWith("DATA"))
  {
   xbeeMessage = data;
  }
  else if(myCommand.startsWith("MARCO"))
  {
   xbeeMessage = "POLO";
  }
  else if(myCommand.startsWith("FREQ="))
  {
   myCommand.remove(0,5);
   xbeeMessage = "New Send Rate: " + myCommand;
   xbeeRate = myCommand.toInt();
  }
  else{
   xbeeMessage = "Error - command not recognized: " + groundCommand;
  }
  if(xbeeMessage!="") return xbeeMessage;
 }
```

SD CODE:

```
// Contains setup and code for SD logging.
// Students should not have to add code here

#include <SD.h> //Should come automatically with Arduino, although you may have to disable the native library.

#define chipSelect BUILTIN_SDCARD //Should highlight if you have teensy 3.5/3.6/4.0 selected

File datalog;
File datalogIMU;
char filename[] = "SDCARD00.csv";
```

```
bool sdActive = false;

void sdSetup(){
 pinMode(chipSelect,OUTPUT);
 if(!SD.begin(chipSelect)){
  Serial.println("Card failed, or not present");
  updateOled("Turn off\nand Insert SD card");
  for(int i=1; i<20; i++){
   digitalWrite(13,HIGH);
   digitalWrite(sdLED,LOW);
   delay(100);
   digitalWrite(13,LOW);
   digitalWrite(sdLED,HIGH);
   delay(100);
  }
 }
 else {
  Serial.println("Card initialized.\nCreating File...");
  for (byte i = 0; i < 100; i++) {
   filename[6] = '0' + i/10;
   filename[7] = '0' + i%10;
   if (!SD.exists(filename)) {
    datalog = SD.open(filename, FILE_WRITE);
    sdActive = true;
    Serial.println("Logging to: " + String(filename));
    updateOled("Logging:\n\n" + String(filename));
    delay(1000);
    break;}
  }
  if (!sdActive) {
   Serial.println("No available file names; clear SD card to enable logging");
   updateOled("Clear SD!");
   delay(5000);
  }
 }
}

void logData(String Data){
 datalog = SD.open(filename, FILE_WRITE);
 datalog.println(Data);
 datalog.close();
 Serial.println(Data);
```

}

SENSORS CODE:

```
// Setup and code for all sensors mounted on PTERODACTYL board.
// Students should add their own setup and update functions here and reference them from the main setup() and loop() functions in PTERODACTYL.ino
// Students should not have to modify and of the existing code.

#include <UbloxGPS.h> //Library can be found at https://github.com/MNSGC-Ballooning/FlightGPS
#include <TinyGPS++.h>  //Library comes in the same download as UbloxGPS.h
#include <Wire.h> //This should come automatically with Arduino
#include <SPI.h>  //This should come automatically with Arduino
#include <SparkFunLSM9DS1.h> //Library can be found at https://github.com/sparkfun/SparkFun_LSM9DS1_Arduino_Library
#include <OneWire.h>  //This should come automatically with Arduino
#include <MS5611.h> //Library can be found at https://github.com/Patrikpcm/Arduino-MS5611-Library
#include <SFE_MicroOLED.h>  //Library can be found at https://github.com/sparkfun/SparkFun_Micro_OLED_Arduino_Library

#include <SparkFun_GridEYE_Arduino_Library.h>
#include <Wire.h>


//The library assumes a reset pin is necessary. The Qwiic OLED has RST hard-wired, so pick an arbitrarty IO pin that is not being used
#define PIN_RESET 9
//The DC_JUMPER is the I2C Address Select jumper. Set to 1 if the jumper is open (Default), or set to 0 if it's closed.
#define DC_JUMPER 1

#define thermIntPin A16
#define thermExtPin A17
#define ubloxSerial Serial3       // Serial communication lines for the ublox GPS -- PCB pins: Serial5

MS5611 baro;
MicroOLED oled(PIN_RESET, DC_JUMPER);    // I2C declaration
LSM9DS1 imu;
```

```
UbloxGPS ublox(&ubloxSerial);
GridEYE grideye;

/////////////// Thermistor constants //////////////////////

float adcMax = pow(2,analogResolutionBits)-1.0; // The maximum adc value given to the thermistor
float A = 0.001125308852122;
float B = 0.000234711863267;
float C = 0.000000085663516; // A, B, and C are constants used for a 10k resistor and 10k thermistor for the steinhart-hart equation
float R1 = 10000; // 10k Ω resistor
float Tinv;
float adcVal;
float logR;
float T; // these three variables are used for the calculation from adc value to temperature
float currentTempC; // The current temperature in Celcius
float currentTempF; // The current temperature in Fahrenheit
float gridEyeIndex;
float gridEyeValue;
float gridEyeArray[64];
bool gridEyeActive = false;

byte statusCheck;
int lineNumber = 0;


void ubloxSetup(){
 ubloxSerial.begin(UBLOX_BAUD);
 ublox.init();

 byte i = 0;
 while (i<50) {
  i++;
  if (ublox.setAirborne()) {
   Serial.println("Air mode successfully set.");
   break;}
  if (i==50){
   Serial.println("Failed to set to air mode.");
   updateOled("Failed to set GPS Air Mode");
   delay(5000);
  }
 }
```

```
  updateOled("GPS init\ncomplete!");
  delay(1000);
}

void imuSetup(){
  Wire.begin();
  if(!imu.begin()){
    Serial.println("Failed to communicate with LSM9DS1.");
    updateOled("IMU\nOffline.");
    delay(5000);
  }
  else{
    updateOled("IMU init\ncomplete!");
    delay(1000);
  }
}

void updateIMU(){
  if( imu.gyroAvailable() ) imu.readGyro();
  if( imu.accelAvailable() ) imu.readAccel();
  if( imu.magAvailable() ) imu.readMag();

  magnetometer[0] = imu.calcMag(imu.mx);
  magnetometer[1] = imu.calcMag(imu.my);
  magnetometer[2] = imu.calcMag(imu.mz);
  accelerometer[0] = imu.calcAccel(imu.ax);
  accelerometer[1] = imu.calcAccel(imu.ay);
  accelerometer[2] = imu.calcAccel(imu.az);
  gyroscope[0] = imu.calcGyro(imu.gx);
  gyroscope[1] = imu.calcGyro(imu.gy);
  gyroscope[2] = imu.calcGyro(imu.gz);
}

void oledSetup(){
  Wire.begin();
  oled.begin();    // Initialize the OLED
  oled.clear(ALL); // Clear the display's internal memory
  oled.display();  // Display what's in the buffer (splashscreen)
  //delay(1000);     // Delay 1000 ms
  oled.clear(PAGE); // Clear the buffer.

  randomSeed(analogRead(A0) + analogRead(A1));
```

```
  updateOled("Initializing...");
}

void updateOled(String disp){
  oled.clear(PAGE);
  oled.setFontType(0);
  oled.setCursor(0, 0);
  oled.println(disp);
  oled.display();
}

void msSetup() {
  //updateOled("initializing\nbaro...");
  while(!baro.begin()){
    updateOled("baro init failed!");
  }
  /*if(!baro.begin()){
    Serial.println("MS5611 Altimeter not active");
    updateOled("digital baro not active");
  }*/
  updateOled("baro init\ncomplete!");
  delay(1000);
}

void updateMS() {
  msTemperature = baro.readTemperature();
  msTemperature = msTemperature*(9.0/5.0) + 32.0;
  msPressure = baro.readPressure();
  msPressure = msPressure * 0.000145038;
}

void updateThermistor(){
  analogReadResolution(analogResolutionBits);
  adcVal = analogRead(thermIntPin);
  logR = log(((adcMax/adcVal)-1)*R1);
  Tinv = A+B*logR+C*logR*logR*logR;
  T = 1/Tinv;
  currentTempC = T-273.15; // converting to celcius
  currentTempF = currentTempC*9/5+32;
  thermistorInt = currentTempF;

  adcVal = analogRead(thermExtPin);
  logR = log(((adcMax/adcVal)-1)*R1);
```

```
  Tinv = A+B*logR+C*logR*logR*logR;
  T = 1/Tinv;
  currentTempC = T-273.15; // converting to celcius
  currentTempF = currentTempC*9/5+32;
  thermistorExt = currentTempF;
}

void gridEyeSetup(){
   // Start your preferred I2C object
  Wire.begin();
  // Library assumes "Wire" for I2C but you can pass something else with begin() if you like
  grideye.begin();
}

void updateGridEye(){
  float testPixelValue = 0;
  float hotPixelValue = 0;
  int hotPixelIndex = 0;
  for(unsigned char i = 0; i < 64; i++){
    testPixelValue = grideye.getPixelTemperature(i);
     if(testPixelValue > hotPixelValue){
       hotPixelValue = testPixelValue;
       hotPixelIndex = i;
     }
  if(-100<hotPixelValue> 100){
     gridEyeActive = true;
  }else{
    gridEyeActive = false;
  }
  }

  // Print the result in human readable format
  gridEyeIndex = hotPixelIndex;
  gridEyeValue = hotPixelValue;
}

void updateAllGridEye(){
  for(unsigned char i = 0; i < 64; i++){
    gridEyeArray[i] = grideye.getPixelTemperature(i);
  }
}
```

```
void updateUblox(){
  ublox.update();
}

void updateDataStrings(){
  altitudeFtGPS = ublox.getAlt_feet();
  latitudeGPS = ublox.getLat();
  longitudeGPS = ublox.getLon();

 groundData = String(ublox.getMonth()) + "/" + String(ublox.getDay()) + "/" +
String(ublox.getYear()) + "," +
          String(ublox.getHour()-5) + "," + String(ublox.getMinute()) + "," +
String(ublox.getSecond()) + ","
         + String(ublox.getLat(), 4) + ", " + String(ublox.getLon(), 4) + ", " +
String(altitudeFtGPS, 4)
         +  ", " + String(altitudeFt) + ", " + String(thermistorInt) + ", " + String(thermistorExt)
+ "," +
         String(msTemperature) + ", " + String(msPressure) + ", " + String(millis()/1000.0) + ",
";

 data = groundData + String(magnetometer[0]) + ", " + String(magnetometer[1]) + ", " +
String(magnetometer[2]) + ", " +
          String(accelerometer[0]) + ", " + String(accelerometer[1]) + ", " +
String(accelerometer[2]) + ", " +
          String(gyroscope[0]) + ", " + String(gyroscope[1]) + ", " + String(gyroscope[2]) + ","
+ xbeeMessage + "," + String(gridEyeIndex) + "," + String(gridEyeValue)+ "," +
String(gridEyeArray[0])+ "," + String(gridEyeArray[1])+ "," + String(gridEyeArray[2]) + ","
+ String(gridEyeArray[3])+ String(gridEyeArray[4])+ "," + String(gridEyeArray[5])+ "," +
String(gridEyeArray[6]) + "," + String(gridEyeArray[7])+ String(gridEyeArray[8])+ "," +
String(gridEyeArray[9])+ "," + String(gridEyeArray[10]) + "," + String(gridEyeArray[11])
          + String(gridEyeArray[12])+ "," + String(gridEyeArray[13])+ "," +
String(gridEyeArray[14]) + "," + String(gridEyeArray[15]) + String(gridEyeArray[16])+ ","
+ String(gridEyeArray[17])+ "," + String(gridEyeArray[18]) + "," +
String(gridEyeArray[19])
          + String(gridEyeArray[20])+ "," + String(gridEyeArray[21])+ "," +
String(gridEyeArray[22]) + "," + String(gridEyeArray[23])+ String(gridEyeArray[24])+ "," +
String(gridEyeArray[25])+ "," + String(gridEyeArray[26]) + "," + String(gridEyeArray[27])+
String(gridEyeArray[28])+ "," + String(gridEyeArray[29])+ "," + String(gridEyeArray[30]) +
"," + String(gridEyeArray[31])
          + String(gridEyeArray[32])+ "," + String(gridEyeArray[33])+ "," +
String(gridEyeArray[34]) + "," + String(gridEyeArray[35])+ String(gridEyeArray[36])+ "," +
String(gridEyeArray[37])+ "," + String(gridEyeArray[38]) + "," + String(gridEyeArray[39])+
```

```
String(gridEyeArray[40])+ "," + String(gridEyeArray[41])+ "," + String(gridEyeArray[42]) +
"," + String(gridEyeArray[43])
        + String(gridEyeArray[44])+ "," + String(gridEyeArray[45])+ "," +
String(gridEyeArray[46]) + "," + String(gridEyeArray[47])+ String(gridEyeArray[48])+ "," +
String(gridEyeArray[49])+ "," + String(gridEyeArray[50]) + "," + String(gridEyeArray[51])+
String(gridEyeArray[52])+ "," + String(gridEyeArray[53])+ "," + String(gridEyeArray[54]) +
"," + String(gridEyeArray[55])
        + String(gridEyeArray[56])+ "," + String(gridEyeArray[57])+ "," +
String(gridEyeArray[58]) + "," + String(gridEyeArray[59])+ String(gridEyeArray[60])+ "," +
String(gridEyeArray[61])+ "," + String(gridEyeArray[62]) + "," + String(gridEyeArray[63]);
```

```
 satData = "|" + String(statusCheck) + ", " + String(lineNumber) + ", " + groundData + String
(gridEyeIndex) + ", " + String(gridEyeValue);
 lineNumber +=1;

  updateOled(String(latitudeGPS,4) + "\n" + String(longitudeGPS,4) + "\n" +
String(altitudeFtGPS,1) + "ft\nInt:" + String(int(thermistorInt)) + " F\nExt:"
  + String(thermistorExt) + " F\n" + String(msPressure,2) + " PSI");
  if(ublox.getFixAge() > 2000) fix = false;
  else fix = true;
  logData(data);
}

void pullPin(){
  updateOled("Pull Flight Pin to start timer");

  while(pullOn==0)
  {
    digitalWrite(fixLED,HIGH);
    digitalWrite(ppodLED,HIGH);
    digitalWrite(xbeeLED,HIGH);
    digitalWrite(sdLED,HIGH);
    pullOn = digitalRead(pullBeforeFlightPin);
  }
  ppodOffset = millis();
  updateOled("Timer Starting");
  digitalWrite(fixLED,LOW);
  digitalWrite(ppodLED,LOW);
  digitalWrite(xbeeLED,LOW);
```

```
  digitalWrite(sdLED,LOW);
  delay(2000);
}

void ledGlissando() {
  digitalWrite(fixLED,HIGH);
  delay(50);
  digitalWrite(ppodLED,HIGH);
  delay(50);
  digitalWrite(fixLED,LOW);
  digitalWrite(xbeeLED,HIGH);
  delay(50);
  digitalWrite(ppodLED,LOW);
  digitalWrite(sdLED,HIGH);
  delay(50);
  digitalWrite(xbeeLED,LOW);
  delay(50);
  digitalWrite(sdLED,LOW);
  delay(150);
}

void updateStatus(){
  bitWrite(statusCheck, 7, 1);
  bitWrite(statusCheck, 6, fix);
  bitWrite(statusCheck, 5, sdActive);
  bitWrite(statusCheck, 4, thermistorInt <100 );
  bitWrite(statusCheck, 3, thermistorInt > -130);
  bitWrite(statusCheck, 2, 0 < msPressure < 15);
  bitWrite(statusCheck, 1, -100 < thermistorExt< 80);
  bitWrite(statusCheck, 0, gridEyeActive);
}
```