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

The Best Team

Written by:
Ethan Cederberg, Mustafa Moneer, Ethan Gramowski, Caleb Boll, Mohamed Farah

Report Submission Date: 12/3/2021
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.

User Notes (document adapted from Colorado Space Grant):

This template describes the topics which should be discussed during the evolution of your documentation. The following sections have a Rev (Revision) letter following the section description. This indicates when this section is expected to be a part of this document. **If a section is required in Rev A, then that section should be written for Rev A then updated if necessary in all subsequent revisions.** As your project becomes more defined, return to previous sections and update them accordingly.

Each time when you submit your Team Project Documentation, remove any unnecessary template notes (but leave in place all to-be-written sections). For example, if you are submitting Rev A, leave alone all instructions about sections not requested until Rev B and Rev C.

This report is due in electronic form (Microsoft Word, not pdf) at the times listed in the table above. Please follow this template format exactly. Submit only one copy for the whole team.

Write your text sections just like this page – single spaced, 1 inch margins, 12 point font of your choice, leaving single blank lines (no indentation) between paragraphs.

Table of Contents

0.0 Team Project Documentation Writing/Build/Flight Day Assignments

Team Name The Best Team

Introduction	Ethan Cederberg
Mission Overview	Ethan Cederberg
Payload Design	Mohamed Farah and Mustafa Moneer
Project Management	Caleb Boll
Project Budgets	Caleb Boll
Payload Photographs	Mohamed Farah
Test Plan and Results	Ethan Gramowski
Expected Science Results	Ethan Gramowski
Flight Day Narrative	Ethan Cederberg
Results and Analysis	Ethan Gramowski
Conclusions and Lessons Learned	Mustafa Moneer
References	Mustafa Moneer
Program Listings	Ethan Cederberg

Oral Presentation Assignments

Conceptual Design Review (CDR)	Mustafa Moneer	Mohamed Farah
Flight Readiness Review (FRR)	Ethan Cederberg	Ethan Gramowski

Payload Build Assignments

Overall team lead and ground-testing lead	Ethan Cederberg
Payload box build	Mohamed Farah
PTERODACTYL basic sensor suite	Ethan Gramowski
Programmer lead for the PTERODACTYL	Ethan Cederberg
Camera experiment	Ethan Gramowski
Neulog modules (including set-up)	Caleb Boll
“Other” science experiment(s)	Mustafa Moneer

Flight Day Assignments

Documentation/photographer	Mustafa Moneer
Flight prediction specialist	Ethan Cederberg
Payload handling/start-up specialist	Mohamed Farah
Comms telemetry data specialist	Ethan Gramowski
Payload telemetry data specialist	Ethan Gramowski
Aprs tracking specialist	Ethan Cederberg
SPOT tracking specialist	Caleb Boll

1.0 Introduction

Since ancient times, humanity has looked up at the stars and wished to walk amongst them. We have had dreams and written stories about what may lie beyond the reaches of the highest mountains. As the dust cleared from the carnage of the first half of the 20th century, we began to work towards making those dreams a reality. In 1961, the first person went into space and just eight years later, the first person walked on the surface of our moon. These events as well as many of those that followed still continue to inspire us and, for many of us, make us wish we could visit space ourselves.

This reality is still a fixture of the future unfortunately. However, we can experience and understand many of the same, space-like conditions in the upper atmosphere of our planet; the grey boundary between our world and the unknown. It is this domain where we can explore and test our limits. The craft which explore the upper atmosphere are known as near-spacecraft as they come very close to the limits of outer space. Most often, near-spacecraft are carried aloft by high-altitude, helium or hydrogen-filled weather balloons. While near-spaceflight may not seem as glamorous as sending something into outer space itself, many of the conditions in the upper atmosphere are extremely similar to those in the vacuum of space. Importantly, near-spaceflight is substantially cheaper and more accessible than spaceflight with people from around the world taking part in it. While, with the resources available to the average person, you can't send people up to near-space, we can send sensors and cameras to experience the journey for us. This document is a record of our journey to near space, complete with our triumphs as well as our errors and we are excited to become a part of the near-spaceflight tradition.

2.0 Mission Overview

Our mission is to record the environment in near-space conditions and to better understand the effect this environment has on life. In order to accomplish this, our goal is to design a simple durable payload. The smaller the payload, the higher altitudes we'd reach. Durability is key to ensuring our box survives in the harsh environment of near-space. A crucial part of our payload will be a hand warmer heat pack which will keep the electronics warm and functional in near-space conditions. Simplicity will also play a role in this, we're trying to not over complicate our payload's components so there is minimal error percentage. By focusing on size, durability, and simplicity, we hope to design an effective, no-frills payload that does everything it's designed for. This brings us to our goal. We will have a small sensor suite consisting of a wide-range temperature sensor, a pressure sensor, light sensor, and a Geiger counter. We will also have a 55-degree IR array. Compounding this, we will also send two seed packets up. One will be on the inside, next to the heat pack while the other will be on the outside of the payload in the atmosphere. As for expected data, we would anticipate for temperature to decrease as we ascend, before increasing again as we move up through the stratosphere. We expect to read a

decrease in pressure as we ascend and the Geiger counter will likely read an increase in radiation. We expect the light sensor to read an increase in light intensity as altitude increases. As for the seed packets, we would expect our control packets on the ground to grow faster and with a higher yield than those sent up to the upper-atmosphere, with the one on the outside of the payload having the slowest growth and lowest yield.

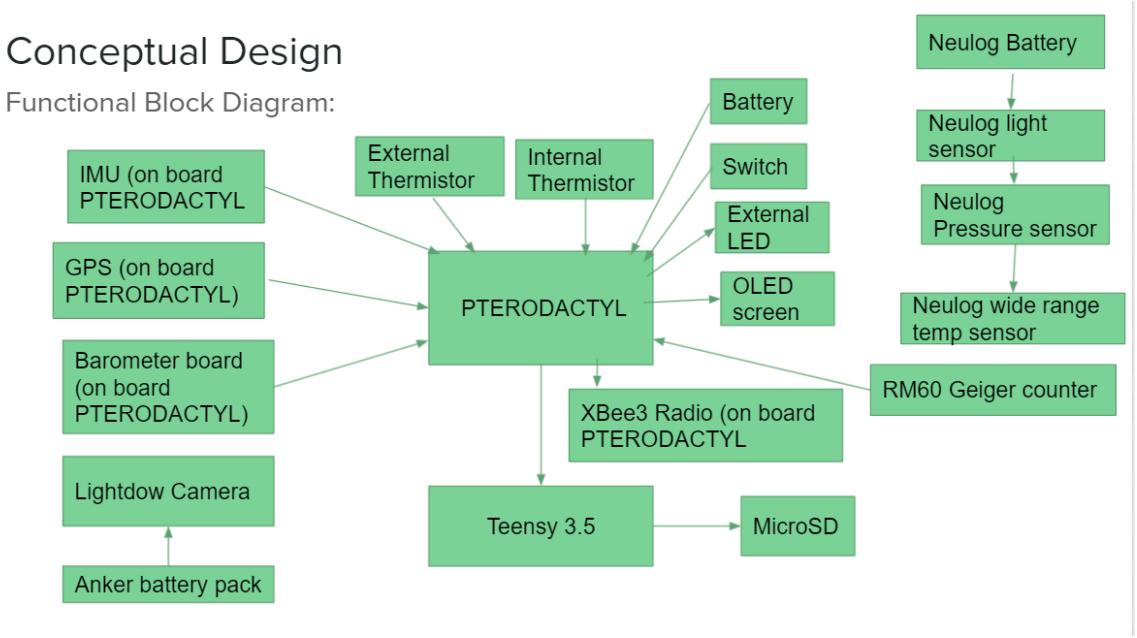
3.0 Payload Design

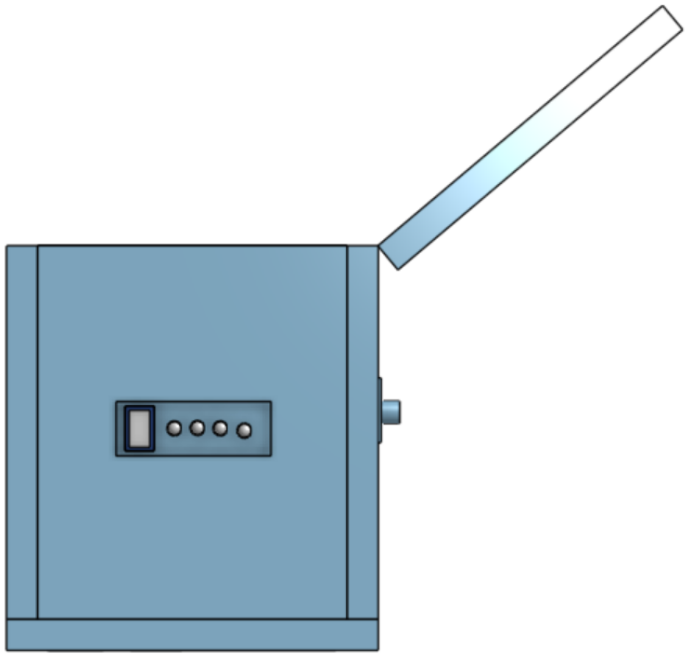
The design of the box is pretty basic, as a team, we decided it is best to keep the box simple and do a rectangular prism. The dimensions of the box are 6 inches(width), 12 inches(length), and 7 inches(height). The top, bottom and long side pieces are all the same dimensions, allowing us to simply make multiple cutouts of the same piece. The box is big enough to fit everything we need in the box. The dimensions I chose allow us to maneuver around the box and are not super complicated. The Lightdow video camera will be facing downwards and its battery pack will be right next to it. Along with the camera, the Pterodactyl will also be on the bottom floor. Neulog sensors will be attached to one of the long sides along with the neulog battery. There will be a small recessed panel for the LED lights and a whole for the PTERODACTYL switch, both holes will most likely be on one of the short sides, if not one one each. About how our systems interface with one another, it all starts with turning on the switch from outside and then battery powering the Pterodactyl. On the side of the box, we have our Neulog sensors that will be collecting information such as pressure and temperature, and then sending that information straight to Teensy where we will be collecting all the data. The OLED screen and external LED will get data from the Teensy through the Pterodactyl. Our payload will interact with group A's payload, because their payload will be below us, our camera will be watching their payload.

Box	styrofoam Box (1/2" thick)	Custom PCB
	rigging tubes (need 4, one for each corner) (cost and weight estimate for 4 for one box)	Teensy 3.5 without headers
	strapping tape (cost and weight estimate for one box)	USB 3-way cable for programming Teensy, charging batteries
	duct tape (cost and weight estimate for one box)	9V battery snap
	braided mason twine (cost and weight estimate for one box)	8 Gig microSD card class 10 with SD adapter
	key rings (need 4) (cost and weight estimate for 4 for one box)	L7805CV 5V voltage regulator
		uBlox M8N gps
Other Stuff	Lightdow LD4000 video/still camera	LSM9051 9DOF IMU
	Anker battery pack (for Lightdow camera)	thermister
	32 Gig microSD card	resistor for thermistor divider (1%)
	Seeds For Experiment	MS5611 pressure sensor
		OLED (minature screen)
Eletrioninc s	PTERODACTYL (Teensy 3.5) microcontroller board with basic breakout boards plugged in *	XBee3 radio
	9-volt Energizer Ultimate Lithium battery (for PTERODACTYL) - might use two	XBee breakout board
	Neulog battery module	LED (with built-in resistors) (4 colors)
	Neulog pressure module	male/male jumper wires
	Neulog wide-range temperature module	male/female extender wires
	Neulog light module	battery jack (solder in)
	SparkFun IR Array Breakout - 55 Degree FOV, MLX90640	male headers strips (first option for Teensy)
	RM-60 Geiger counter	female header strips
		2-position terminal blocks
		8-position terminal blocks
		shorting plugs

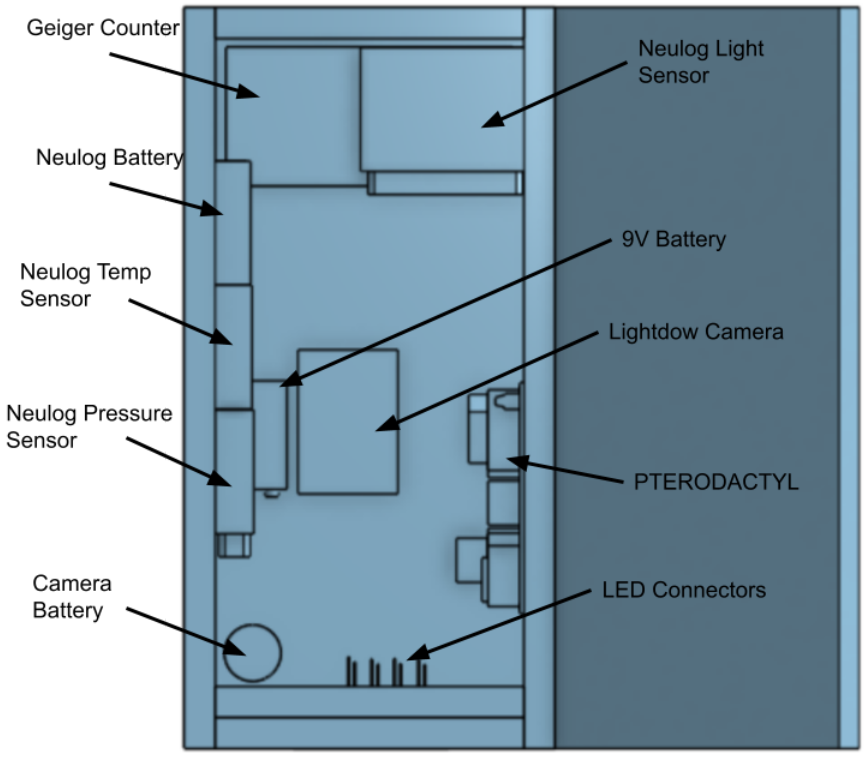
Conceptual Design

Functional Block Diagram:

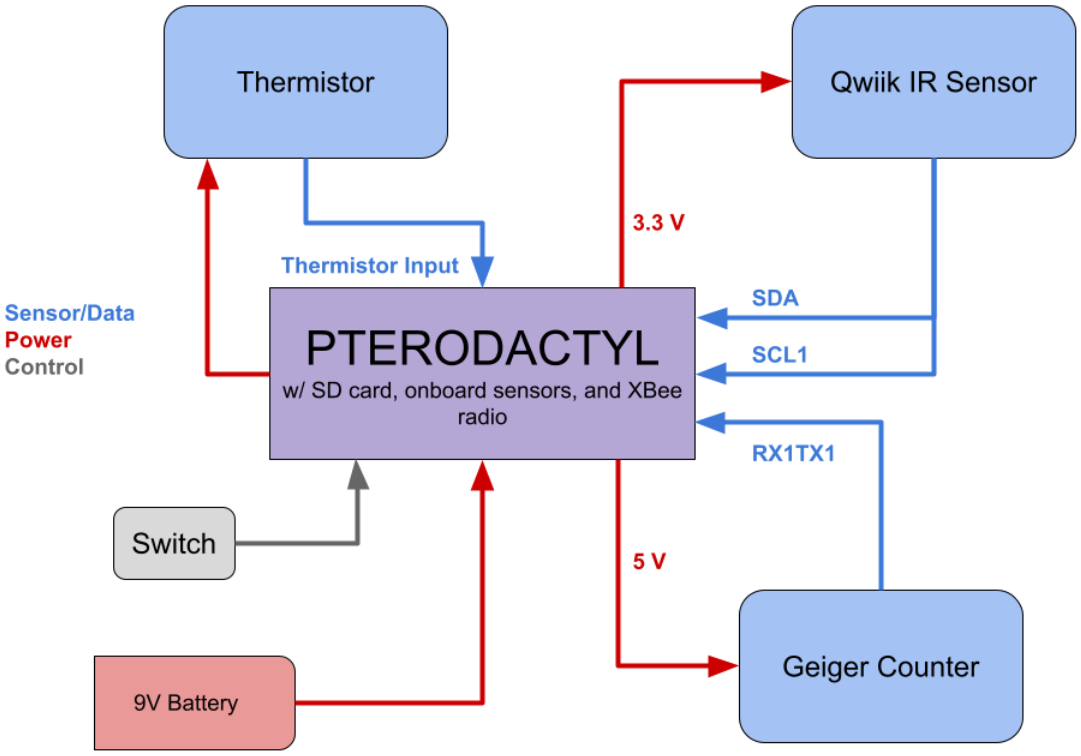




The front view of the payload in onshape. Here the "control panel" with the four indicator LEDs and power switch is viewable. The IR Qwiik sensor is also apparent on the right side.

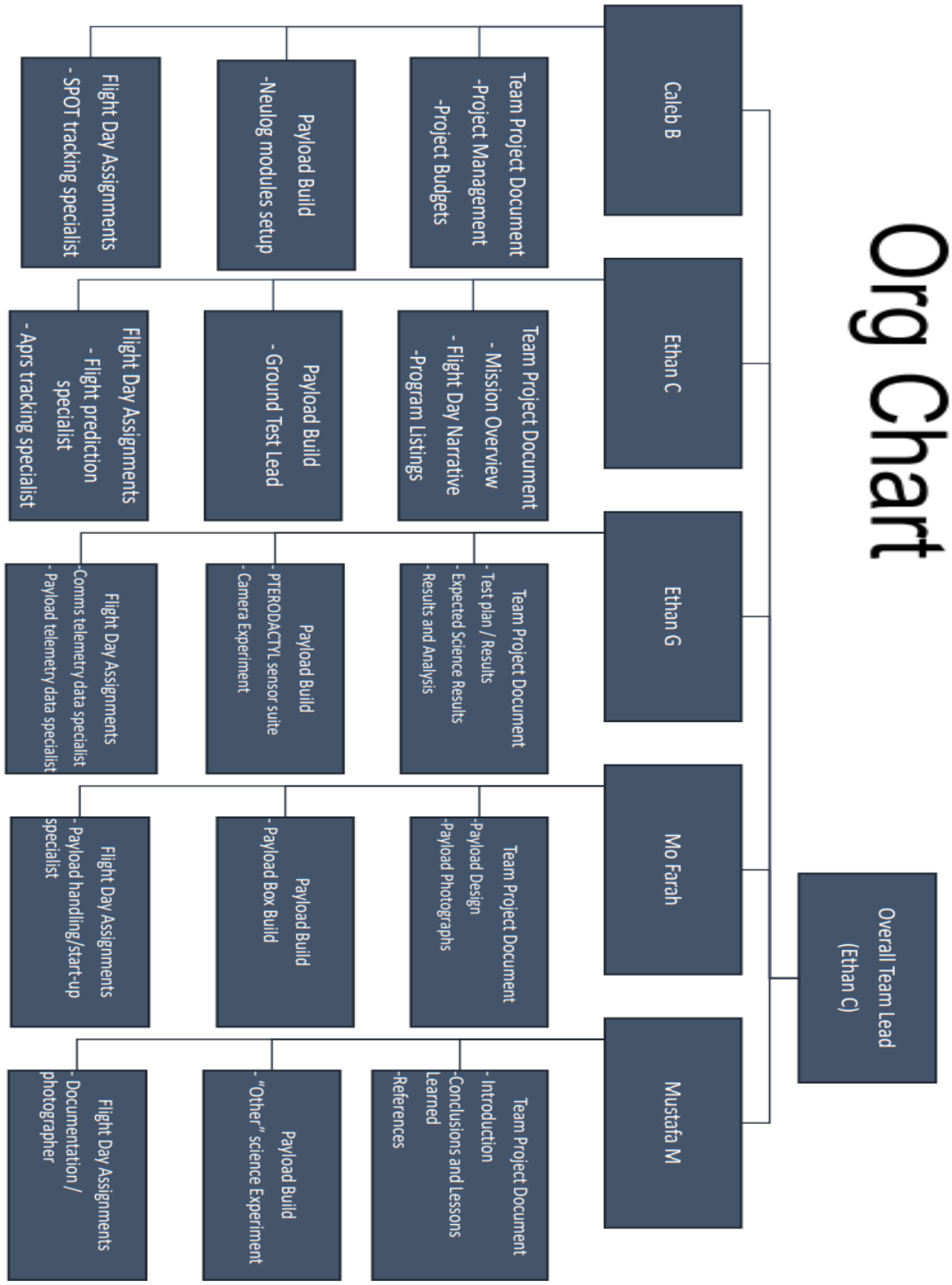


The second screenshot is from the top of the payload and clearly shows all of the main components. It should be noted that the Neulog temperature sensor has a probe extending off of it, as does the PTERODACTYL's thermistor.



The wiring diagram for the PTERODACTYL. Connections which transport power are in red while the connections that are in blue indicate the transfer or generation of data. The grey connection is to denote user input.

4.0 Project Management and Schedule



Management - Calendar	
Tuesday, Oct. 5	Begin work on payload body construction and component integration and testing
Thursday, Oct. 14	Weekly team Meeting to discuss Project.
Friday, Oct. 15	Team Project Documentation – Rev A: Design sections, due by 5pm
Tuesday, Oct. 19	Begin work on Flight prediction software, launch/tracking/recovery logistics, remote monitoring logistics
Wednesday, Oct. 20	Deadline for the payload box build and having all of the electronics and sensors installed.
Thursday, Oct. 21	Weekly team Meeting to discuss Project.
Tuesday, Oct. 26	Flight Readiness Reviews Due
Wednesday, Oct. 27	Deadline for testing the payload and writing the code. Payload ready @8:00 PM
Thursday, Oct. 28	Final deadline for payload weigh-in/turn-in is noon
Thursday, Oct. 28	Weekly team Meeting to discuss Launch.
Saturday, Oct. 30	Launch day : gather at Akerman
Thursday, Nov. 4	Weekly team Meeting to discuss Rev B.
Friday, Nov. 5	Team Project Documentation – Rev B: Design/Build sections, due by 5pm
Thursday, Dec, 2	Weekly team Meeting to discuss Rev C.
Friday, Dec. 3	Team Project Documentation – Rev C: All sections, due 5 p.m.
Monday, Dec. 13	End-of-semester - Video
Tuesday, Dec. 14	End-of-semester public - exhibit

5.0 Project Budgets

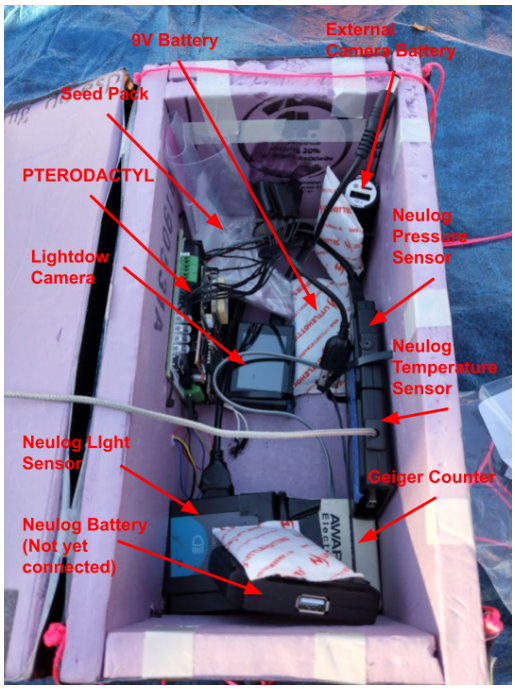
Parts list for AEM 1301 Ballooning Payloads - Fall 2021					
			Approx	Approx	
	Item	Dimensions (in)	Weight (oz)	Cost ($)	Vendor
Box	styrofoam Box (1/2" thick)	each box needs multiple sides	0.5	$1.0	Home Depot
	rigging tubes (need 4, one for each corner) (cost and weight estimate for 4 for one box)	8 length ea	0.5	$1.0	Home Depot
	strapping tape (cost and weight estimate for one box)	multiple encircle	2	$1.5	Ax-Man
	duct tape (cost and weight estimate for one box)	enough to cover	8	$3.5	Home Depot
	braided mason twine (cost and weight estimate for one box)	6 feet length ea	0.4	$1.0	Home Depot
	key rings (need 4) (cost and weight estimate for 4 for one box)	1-1/4 (diameter circle)	0.7	$1.0	Michaels
Other Stuff	Lightdow LD4000 video/still camera	1-3/4 x 1-1/4 x 2-3/8	1.8	$40.0	Amazon
	Anker battery pack (for Lightdow camera)	1 (dia) x 3-3/4 plus cable	3.1	$20.0	Amazon
	32 Gig microSD card	very small	negligible	$8.7	Amazon
	Seeds For Experiment	4-1/2 x 3-1/4	negligible	$9	Online
Eletrionincs	PTERODACTYL (Teensy 3.5) microcontroller board with basic breakout boards plugged in *	3-3/4 x 3-3/4 x 7/8	4	$178.61	custom pcb; total mass and price listed
	9-volt Energizer Ultimate Lithium battery (for PTERODACTYL) - might use two	1 x 2 x 11/16	1.2	$8.00	Home Depot
	Neulog battery module	2-7/8 x 2-1/4 x 5/8	2.3	$45.00	Neulog (misc vendors)
	Neulog pressure module	2-7/8 x 2-1/4 x 5/8	1.7	$83.00	Neulog (misc vendors)
	Neulog wide-range temperature module	2-7/8 x 2-1/4 x 5/8	1.7	$64.00	Neulog (misc vendors)
	Neulog light module	2-7/8 x 2-1/4 x 5/8	1.7	$55.00	Neulog (misc vendors)
	SparkFun IR Array Breakout - 55 Degree FOV, MLX90640	1 x 1 x 1/2	0.64	$69.95	sparkfun
	RM-60 Geiger counter	4-1/2 x 2-1/2 x 1-1/4	4.1	$180	Aware Electronics
			Weight (oz)	Cost ($)	
		Total	34.34	$770.30	

Actual Total Weight: 28.9 oz

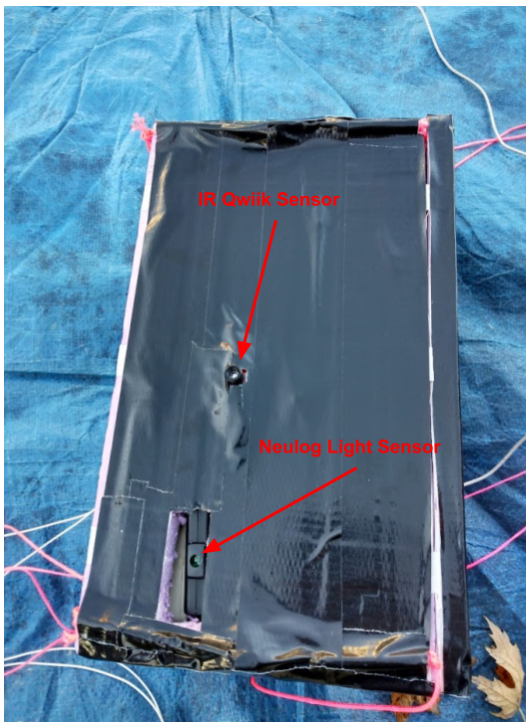
* Parts list for AEM 1301 Ballooning PETRODACTY - Fall 2021

		Weight (oz)	Cost ($)	
Custom PCB	included above	incl abv	$ 5.00	MnSGC
Teensy 3.5 without headers	included above	incl abv	$ 26.25	Digi-Key
USB 3-way cable for programming Teensy, charging batteries	not flown	not flwn	$ 6.97	Digi-Key
9V battery snap	included above	incl abv	$ 1.25	Digi-Key
8 Gig microSD card class 10 with SD adapter	included above	incl abv	$ 11.99	Amazon
L7805CV 5V voltage regulator	included above	incl abv	$ 0.43	Digi-Key
uBlox M8N gps	included above	incl abv	$ 26.99	Newegg
LSM9051 9DOF IMU	included above	incl abv	$ 15.95	Digi-Key
thermister	included above	incl abv	$ 4.10	Digi-Key
resistor for thermistor divider (1%)	included above	incl abv	$ 0.05	Digi-Key
MS5611 pressure sensor	included above	incl abv	$ 8.53	Banggood
OLED (minature screen)	included above	incl abv	$ 16.95	Digi-Key
XBee3 radio	included above	incl abv	$ 20.06	Semiconductor Store
XBee breakout board	included above	incl abv	$ 10.95	Digi-Key
LED (with built-in resistors) (4 colors)	included above	incl abv	$ 1.79	Sparkfun
male/male jumper wires	included above	incl abv	$ 0.45	Digi-Key
male/female extender wires	included above	incl abv	$ 0.79	Digi-Key
battery jack (solder in)	included above	incl abv	$ 1.27	Digi-Key
male headers strips (first option for Teensy)	included above	incl abv	$ 2.54	Digi-Key
female header strips	included above	incl abv	$ 3.27	Digi-Key
2-position terminal blocks	included above	incl abv	$ 3.71	Digi-Key
8-position terminal blocks	included above	incl abv	$ 8.99	Digi-Key
shorting plugs	included above	incl abv	$ 0.32	Digi-Key
		Total Cost Of PETRODACTYL	$ 178.61	

6.0 Payload Photographs



Internal payload components. These components were exposed to the outside environment of the upper-atmosphere as little as possible. Hand warmers were put on the batteries to keep them warm and operational.



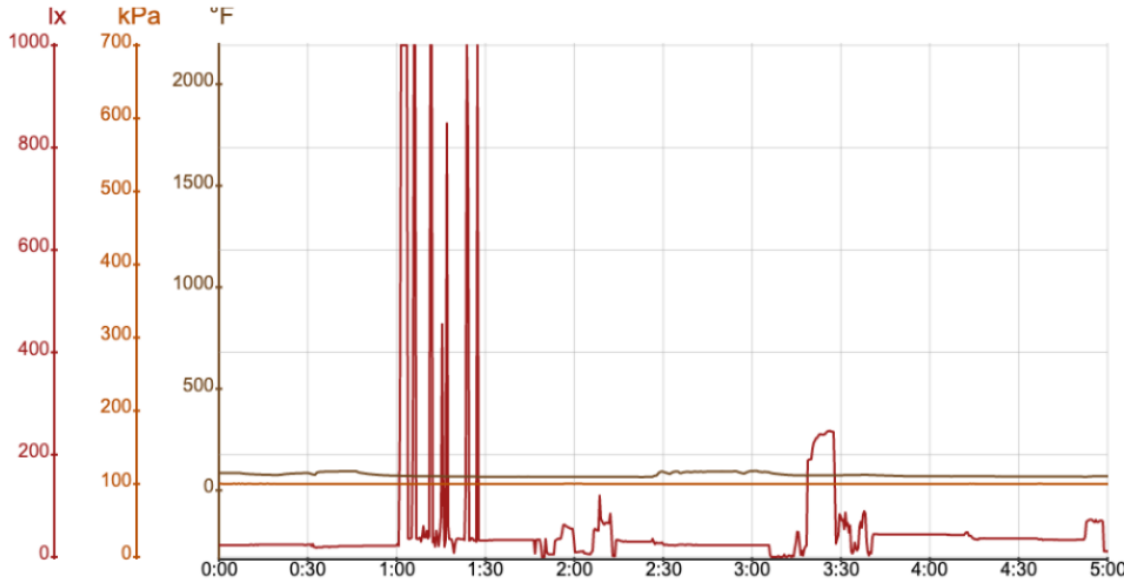
Side payload components. These sensors needed to be exposed to the outside in order to properly collect data.

A view of the payload from the front. The cutout held indicator LEDs and a switch to turn on and off the PTERODACTYL. It was recessed with the hope that these components would be safe from potential harm and so that the switch would not be flipped accidentally.

Another internal view of the payload prior to the application of heat packs on the batteries.

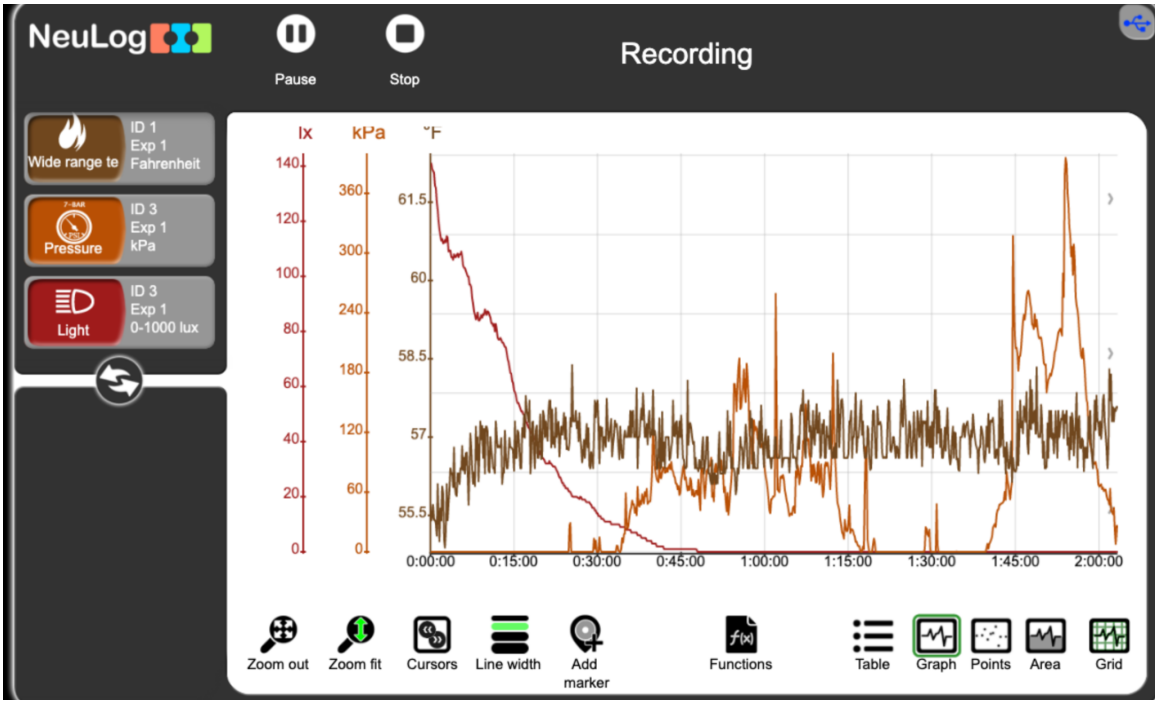
7.0 Pre-flight Ground Testing Plan and Results

On Monday October 25, several tests with the Neulog sensors and Geiger counter were conducted. The first test was to plug the Neulog chain into a laptop and record data for a short period of time (roughly 5 minutes). During this test, each sensor was periodically subjected to some sort of change (temperature, light exposure, pressure, etc…) and the time at which this was done was monitored externally (with a stopwatch) and compared to the data received from the sensors themselves. Data from this test came directly from the desktop Neulog software.



This test was a complete success, with resealable data for all time values on the 5 minute interval.

The second test itself was performed once the Neulog chain had been fitted inside the capsule. The data was written onto the Neulog sensors onboard storage system and exported to an Excel spreadsheet. The test will be repeated for a longer period of time equalling approximately 5 hours.

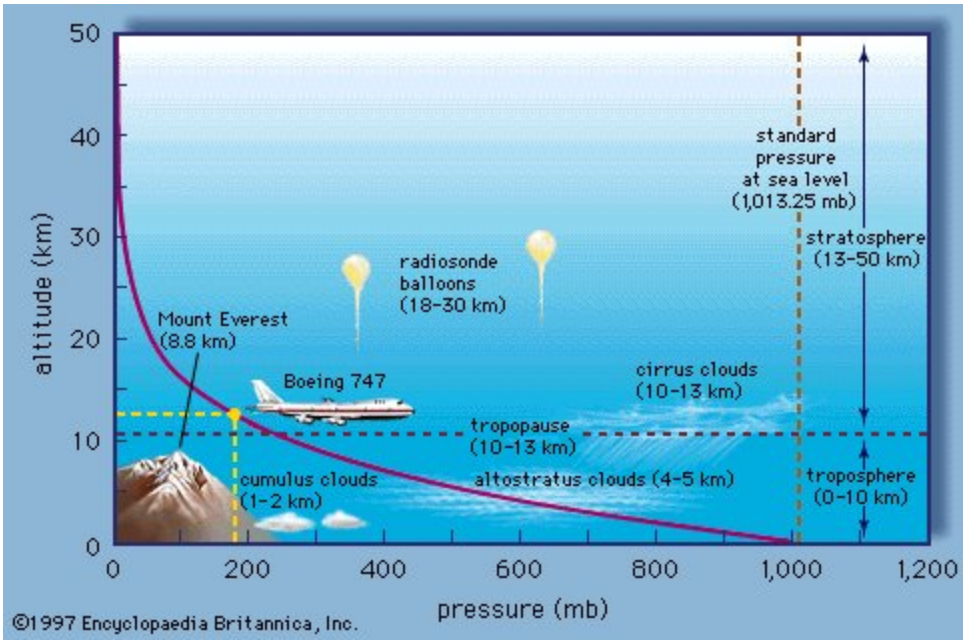


This second test's purpose was multifaceted; it tested the Nuelog chain's ability to capture data and store it and it ensured that our sensors would work for a prolonged

period of time. The second test was a partial success. all of the sensors recorded the expected data values except for the pressure sensor, which appeared to be malfunctioning as can be seen in the graph of the data. The pressure sensor was swapped with another sensor that worked on the day of the flight. The third and final test would have been completed with the Geiger counter, however, the code for the geiger counter was not completed until the night before the flight, making it impossible to run any test before the flight. Professor Flaten assured us that the sensor was fine and it would work with the code of the day of the flight.

8.0 Expected Science Results

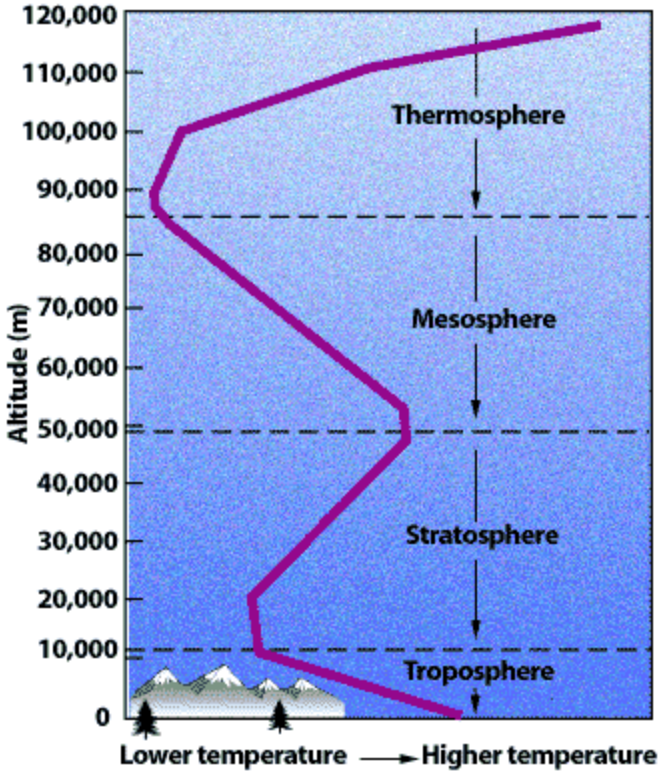
There are several different physical phenomena that where monitored during the course of the flight. The first of these is atmospheric pressure. It is expected that the higher the payload climbs, the lower the atmospheric pressure will become. At low altitude close to sea level, the standard atmospheric pressure is roughly 1.0 atm or approximately 1000 millibars (these numbers are subject to variation caused by change in base elevation and weather conditions). This is a measurement of the force per unit area exerted on by all of the air above a specific area. When the payload is at its maximum height of 100,000 feet above sea level, the expected atmospheric pressure (as measured by the Neulog sensor) is 20mb-60mb (2.0kPa-6.0kPa), as indicated by the following chart.[1]



Graph of altitude as a function of pressure.[1]

This chart shows the relationship between altitude and atmospheric pressure. By the time the near-spacecraft reaches 10,000 meters in elevation relatively early on in its journey, the atmospheric pressure will already be roughly $\frac{1}{5}$ the starting atmospheric pressure. As indicated by the table, nearly 80% of all of the earth's atmosphere by mass is contained with the first 2.5% of the atmosphere by volume. This means that early on in the flight, changes in pressure will be drastic, but as the payload gains altitude and surpasses 10,000 meters, the rate at which pressure declines will decline significantly.

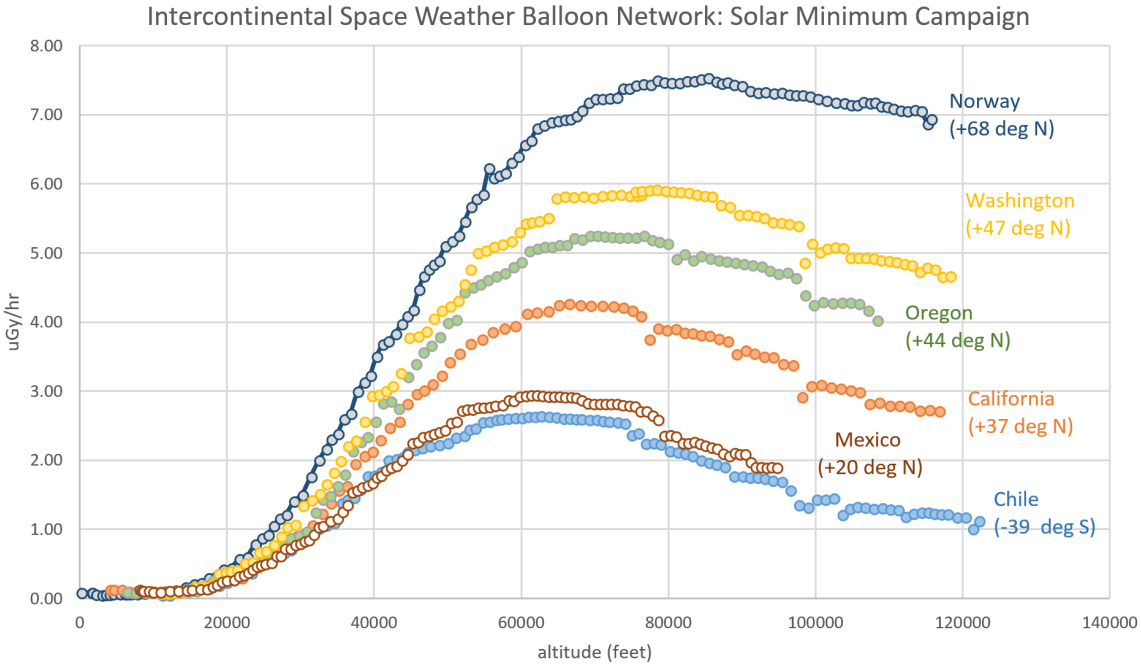
The second phenomenon being measured is the change in temperature as elevation increases. This is not as straightforward as the change in pressure because temperature is dependent on heat transfer. Heat transfer can occur as either conduction, convection, or radiation. In the lower levels of the atmosphere, radiation from the sun heats up the air and convection (the cyclical vertical flow of air molecules) circulates warmer air upwards and cooler air downwards. The specifics of convection are a whole other topic on their own and are not relevant for the purpose of this experiment, however, it is important to understand that air is essential for convection to exist. The temperature at various stages of the atmosphere are represented by this figure.[2]



Graph of Altitude as a function of temperature.[2]

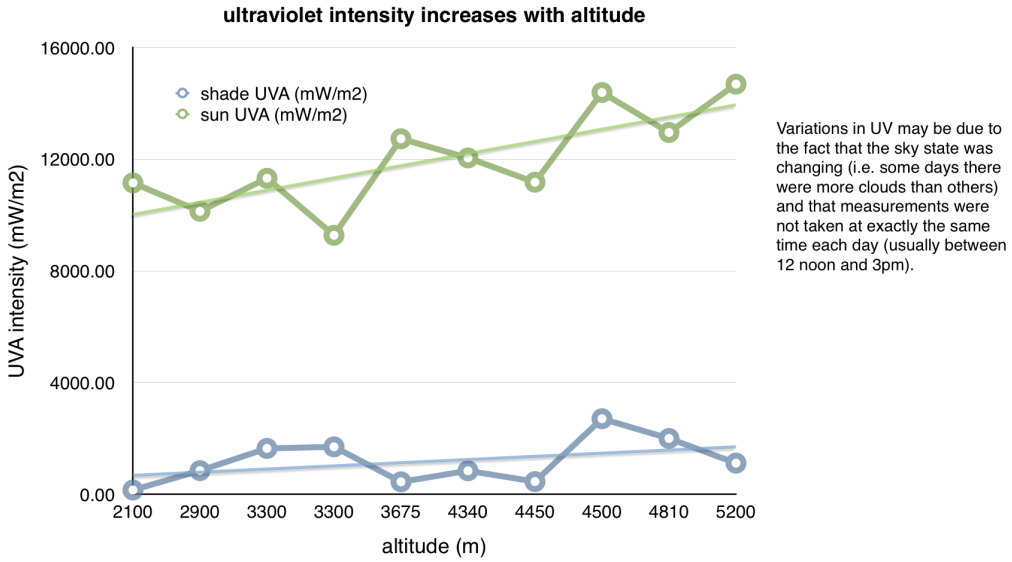
As altitude increases, the amount of matter in the atmosphere decreases. This means that as altitude increases, the space between air molecules increases, causing less and less heat transfer due to convection, as well as less energy stored per cubic unit of area. However,

this also means there is less matter shielding the payload from radiation. It is also known that the change in pressure in the atmosphere is not constant, which can help to explain the figure above. Looking at the graph, below 10,000 meters, temperature is very much dependent on the relationship between the energy stored in air particles and their density. Between 10,000 and 50,000 meters, the range of our flight, the density of the atmosphere decreases significantly, using more heating due to radiation but less heating due to convection. From all of this, it is expected that the temperature at the max height of the payload will be less than the temperature on the ground. This is also very telling about the third value being tested, radiation. The expected amount of radiation at the maximum height of the payload is well beyond the amount of radiation measured on the ground. As seen in the figure below, increase in radiation is proportional to increase in elevation.[3]



graph of radiation per hour as a function of altitude for different latitudes.[3]

Cosmic radiation at ground level is not zero but is very close. Radiation is also dependent on the angle at which an object is facing the sun, meaning that its value will be dependent on both the latitude and the time of year. However, the overall growth pattern should remain the same. This means that the expected amount of cosmic radiation at the max height of the payload should be roughly 5 uGy/hr given the latitude of Minneapolis is 45 deg north.



UVA intensity as a function of altitude.[4]

The last value being measured is light. It is expected that levels will increase as the height of the payload increases. This is because the light from the sun in the upper atmosphere is even more intense than on the ground because there is less stuff in the way.
The graph above shows UV radiation from the sun as a function of altitude.[4] There is a clear positive trend as altitude increases, and the density of the atmosphere decreases. The trend data from our light sensor will most likely be exponential, very similar in fact to the data from our pressure sensor, as the main thing that affects both of these things is the amount of matter between the sensor and outer space.

Two seed packs are also being sent up with the payload. The growth of these seeds will then be compared to the growth of a control seed pack that has been left on the ground. The hypothesis is that the seeds sent into near space will produce a much weaker plant or possibly even no plant at all. This is due to a number of factors including possible blunt force trauma from the landing as well as exposure to both extremely high and extremely low temperatures damaging the seed physically, as well as exposure to high levels of solar radiation interfering with the DNA of the seed, affecting its ability to grow and produce a flower.

9.0 Flight Day Narrative

Flight Day began in the Akerman Hall Atrium at around 7:30 AM to receive some last-minute instructions from Professor Flaten and to double-check everything on the payload. Ensuring it is all completely functional and adding or changing any last-minute items. For our group, this included taping seeds to the outside of the payload, reconfiguring the placement of our light sensor in hopes of getting better readings. After this, we made sure that we had all of the supplies necessary for the

flight. Then we went to our assigned minivans and began to make our way down to the launch site in Janesville, MN. We were one of the first groups to make it down to the launch area. So we waited for the rest of the groups and Professor Flaten to show up. Once everyone had arrived, we moved to an ideal launch location. Which was a field behind a school; we began to unload the vehicles with all the Equipment, Payloads, and air tanks we needed for the launch. After laying out the tarp we started to prep the stacks. First, by untangling and attaching the rigging strings to the payloads, create a chain. There were two payloads, a SAT COM unit, a camera box, the main pterodactyl box, and a parachute on each stack. After we completed the stacks, we began to turn on and test electronics on the payload, including Neulogs, the Pterodactyl, And the camera. After testing everything, we began to fill up the balloons. To do so, we had one person hold the air hose to the balloon with two to three other people supporting the balloon, making sure that it wouldn't. Touch the ground or accidentally get popped. While Professor Flaten filled each balloon with helium once filled. We walked them out into the middle of the field and slowly began to let the payload go. Once we were at the end of the stack, we turned on the Beeper to make it easier to find the payload when it landed. We ended up releasing the balloons at around noon. After that, we helped the Research Group release their balloon and then pack up the launch site. Once that was complete, we drove towards the estimated landing site. While we were on the road, we continuously checked SAT COM and APRS to ensure the balloon was still on course for the estimated landing site. As we were driving through Owatonna, we stopped at the subway for lunch and met up with the other group in our stack to talk about the recovery of the balloon. They continued towards the estimated landing area, and we decided to stay and watch the balloons pop. We were able to see the balloons in the sky and notice the first two balloons were popping. But left before the research balloon had popped. Our balloon popped About 100 minutes into the flight or at about 1:40 PM. After seeing the two balloons pop we made our way to the estimated landing site A mile or two West of Roscoe, MN.

After checking SAT COM and APRS, we believe the payload will land southeast of the initial landing prediction. The other team ended up getting to the payload landing site first. It landed in the back of someone's farm and because it may have landed in or near a cattle field. The other group went with the landowner to retrieve the payload. At the same time, we were asked to wait out by the side of the road. After they recovered the payload, we met up with the other group to turn off all sensors and electronics on the payload. At this point, we had our payload. And all of the necessary items that went along with it. We were about to head back to the U of M, but the other balloon landed in a tree, and Professor Flaten, asked us to retrieve the research team's payload. So we headed a couple of miles north to retrieve the payload, which landed A couple of yards. We recovered that payload a couple of yards from someone's driveway and began to turn off all of the electronics.
We were now ready to head back to the University of Minnesota. After we made it back to the university, we checked in with the other group on our stack who had the payload, and they said they got stuck in traffic and it would be a couple of minutes.

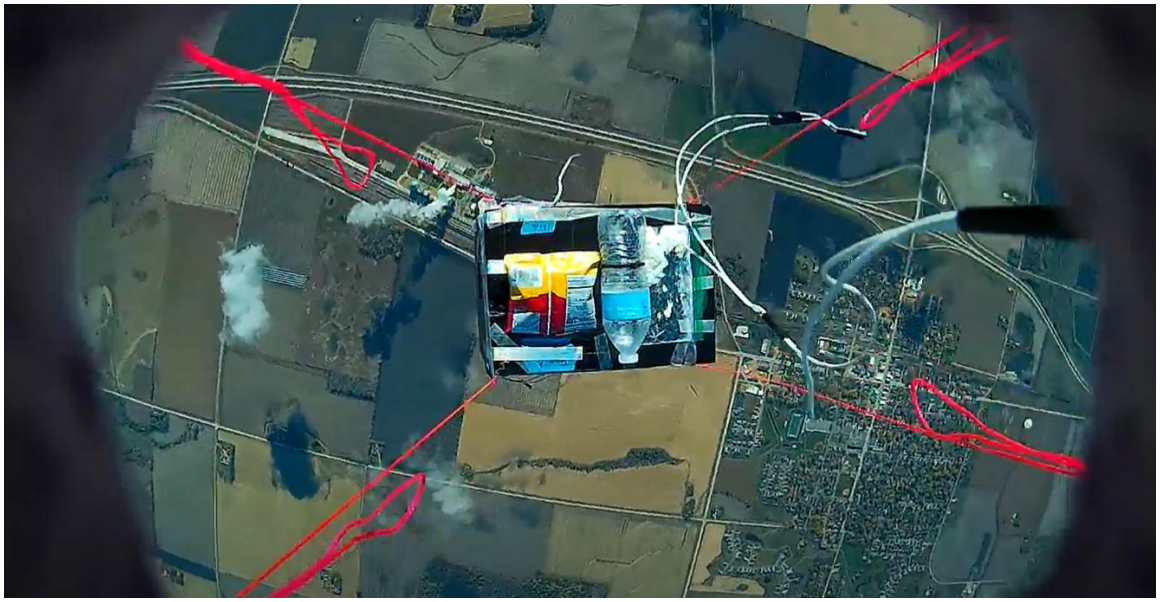
So we decided that there wasn't much more for us to do today and left. Deciding to retrieve our data the next time we had a class. Along with our seeds to conduct our experiment. Overall, our Flight Day was a success, and we did not have many issues we couldn't handle. The biggest problem during the flight was that one of the APRS trackers on our payload was not working correctly, but luckily there was a second one, so it did not affect us that much.

(Photo of Inflating the Balloon)

(Photo of The Balloon)

(Photo of Launching the Balloon)

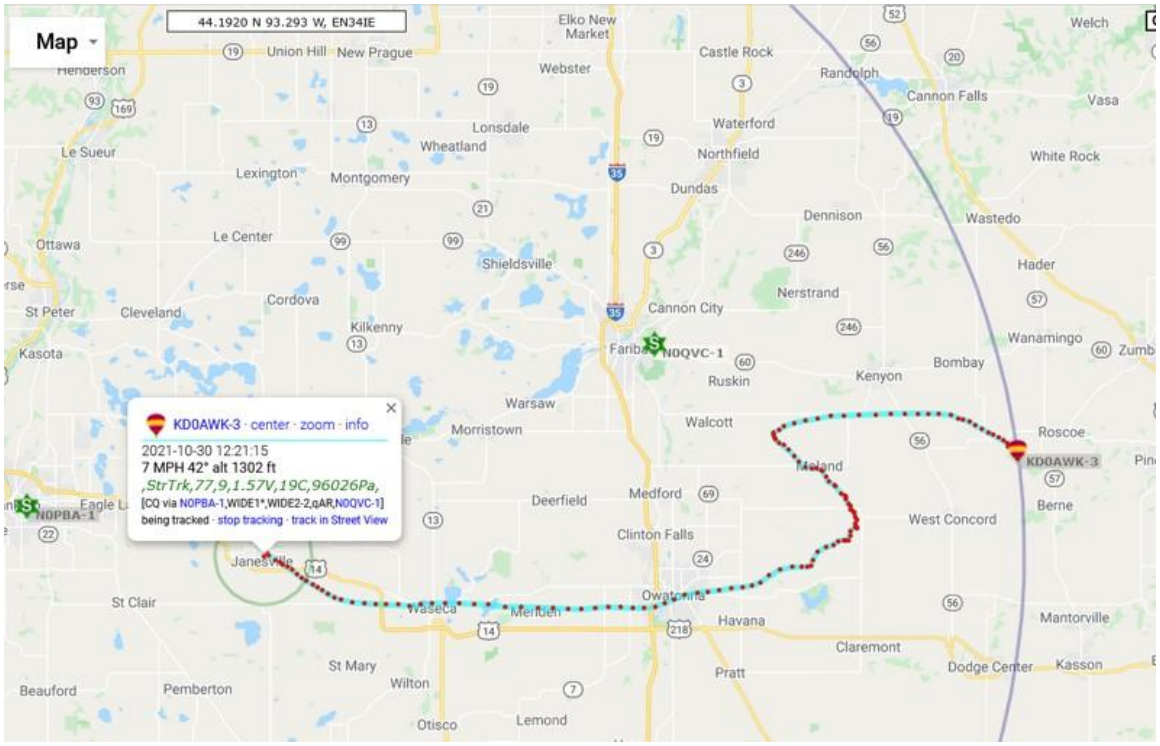
(Photo of released Balloons)



(Photo from Balloon Minutes after launch)

(Photo from Balloon after Burst)

(Photo of Where the Payload Landed)

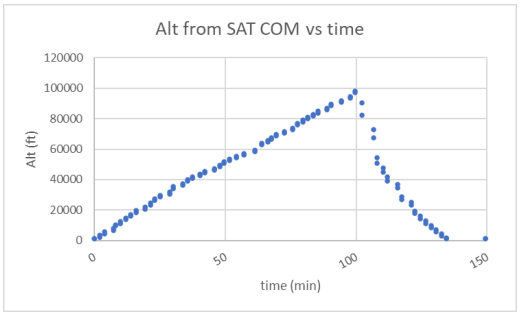


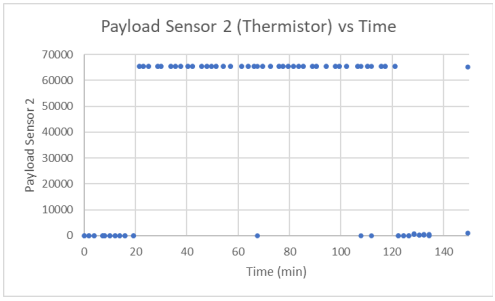
(Screenshot of Balloon Flight Path)

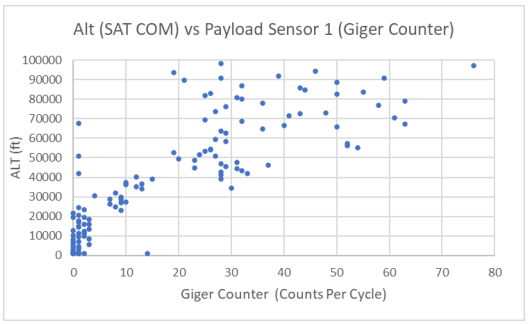
10.0 Results and Analysis

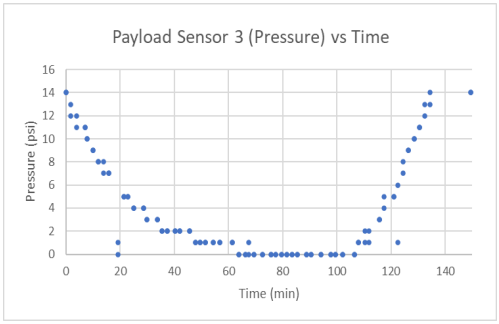
SAT COM Results

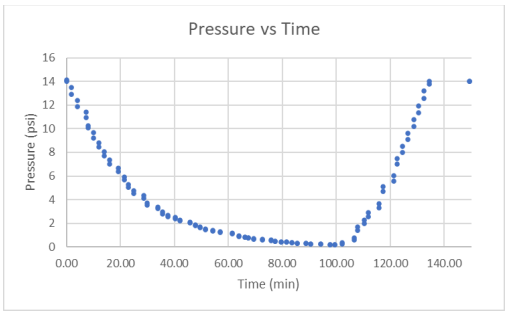
For the most part we used the SAT COM data to compare and check the data from other sources to see if they are accurate. Because the satcom does not have any unique sensor or as many data points compared to the neulogs or the PTERODACTYL. You can see that most of the SAT COM graph matches up with another graph from the neulogs or the PTERODACTYL. The one exception to this is that of Payload Sensor 2 with was meant to measure Thermistor data but something went wrong and our data got corrupted because 20 min in to the flight it jumps up to 65,000 and then at round 120 min suddenly falls back down and we have no idea why it does that.

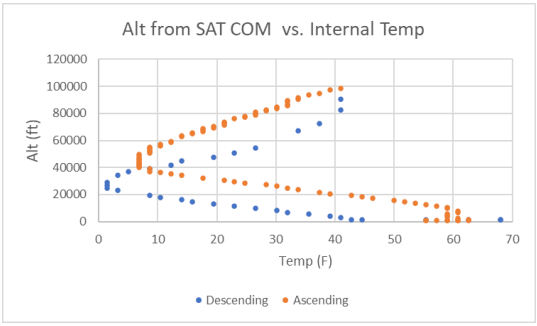






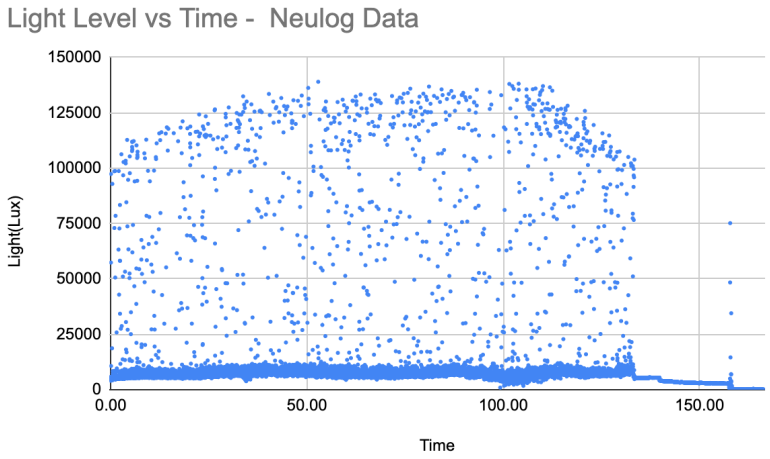


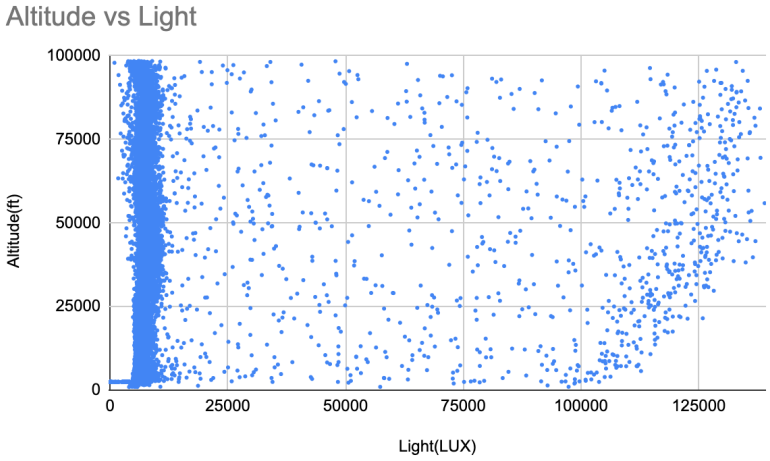




Neulog data results

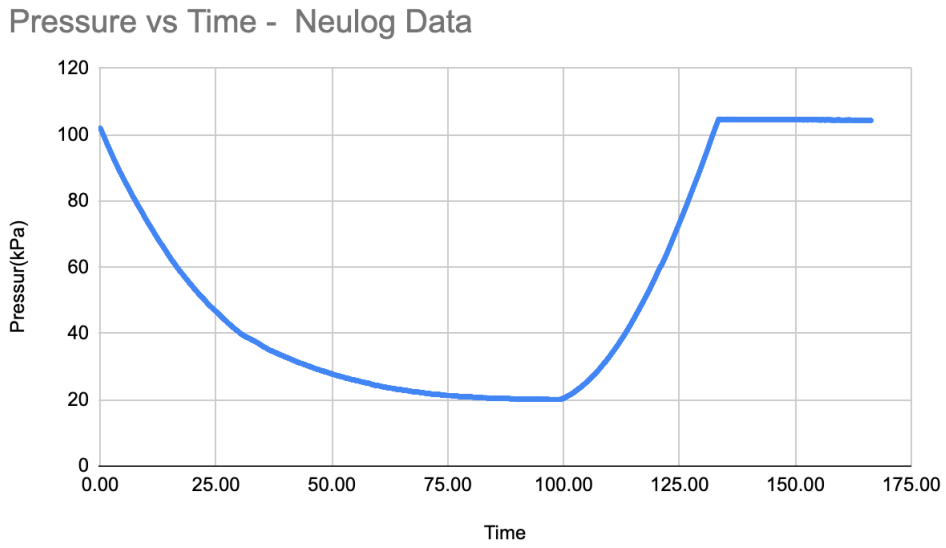
Light Level

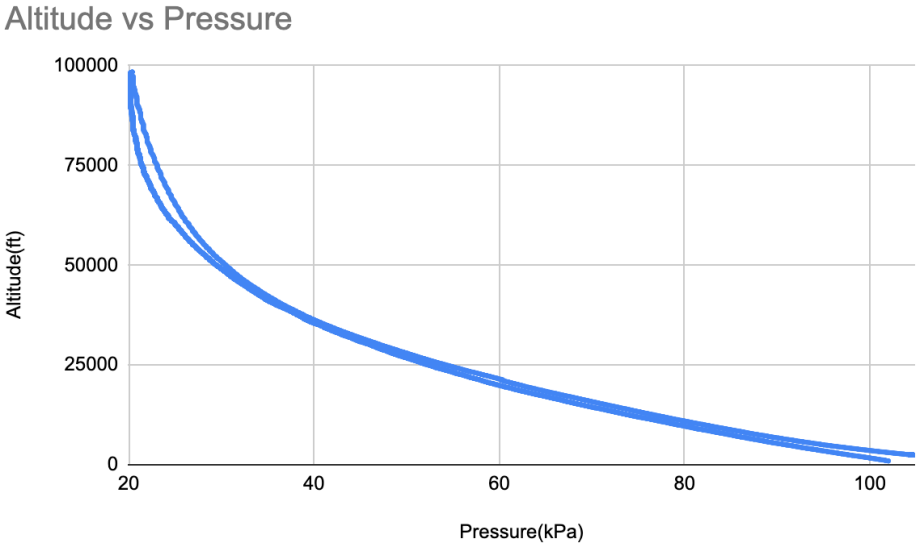




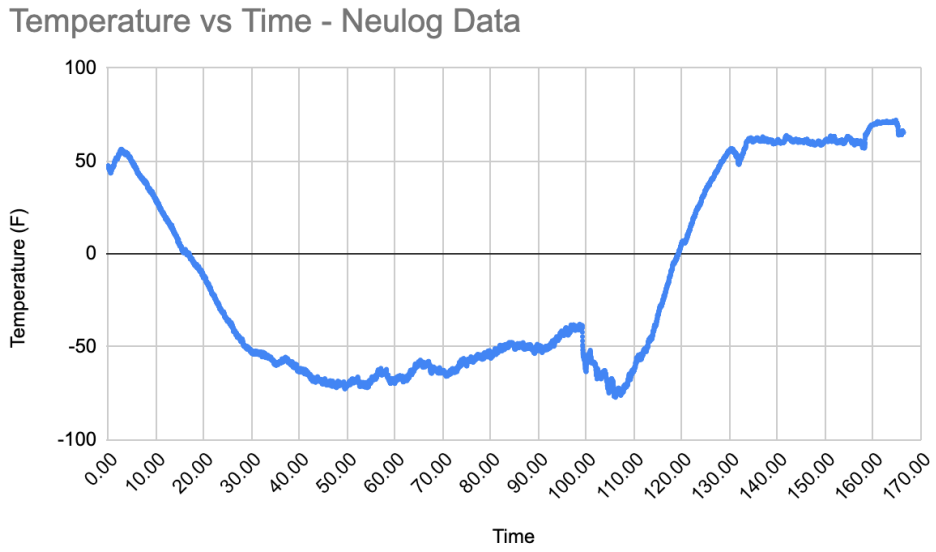
The top graph is a graph of light level as a function of time and the other is of altitude as a function of light level. Each point indicates a one light level reading. The shape made by the top of the graph for the time graph and the right side of the graph for the altitude graph shows the maximum light level as the time/altitude increases. a positive trend can be seen all the way up to 100 minutes or 100,000 feet, where the average peaks and begins to decrease. This indicates that the higher the altitude, the higher the light level on average. the density of points at the bottom of the graph shows that the sensor was pointed away from the sun for most of the flights duration, which was to be expected, direct light from the sun only comes from a small percentage of the whole sky

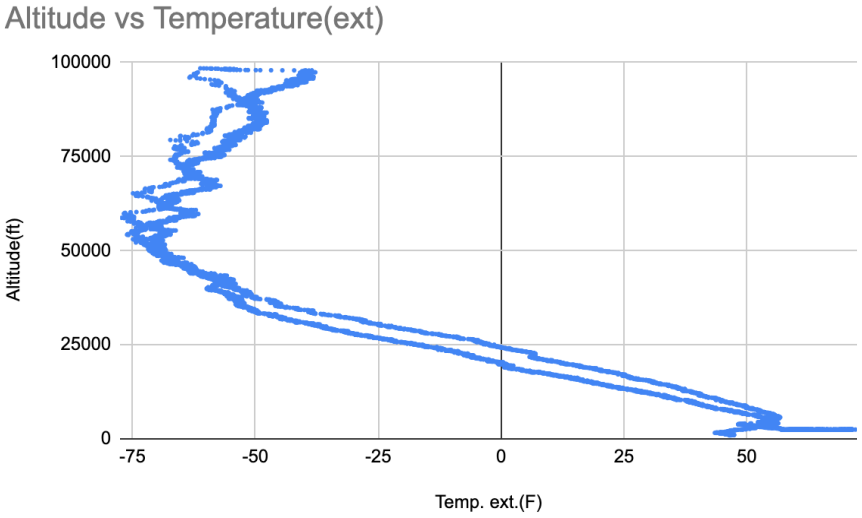
Pressure





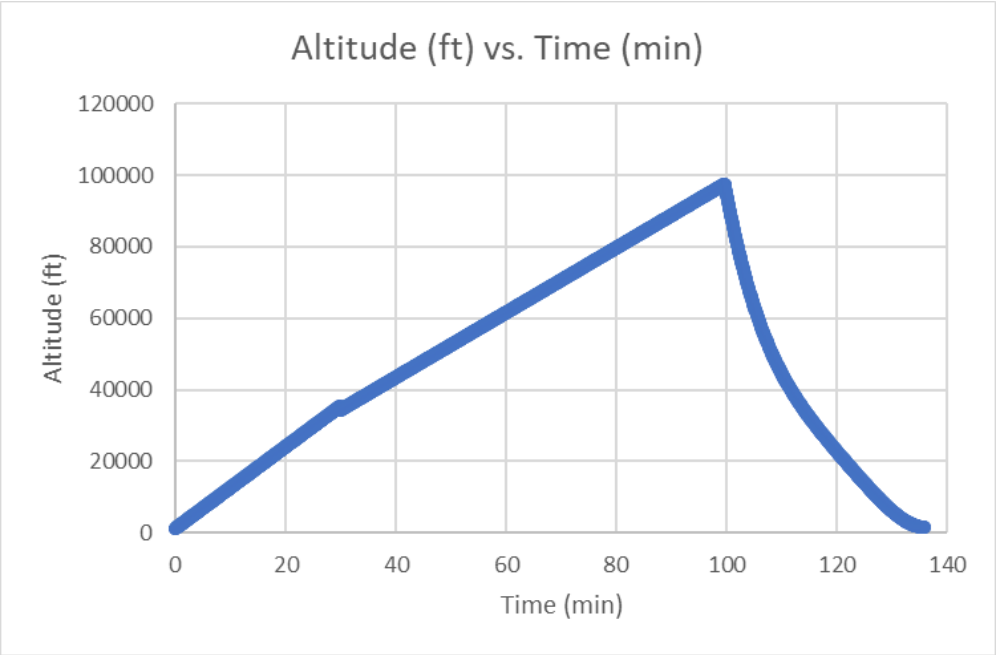
The top figure is a graph of pressure as a function of time and the bottom chart is altitude as a function of pressure. The decrease and subsequent increase in pressure as altitude and time increased was exponential as expected. since the composition of the atmosphere is not linear, we would not expect the pressure graph to be linear. minimum pressure was 20 kPa, which was a little bit larger than expected, but well within the margin of error that can be explained by changes in weather conditions and uneven particle dispersion in the atmosphere itself



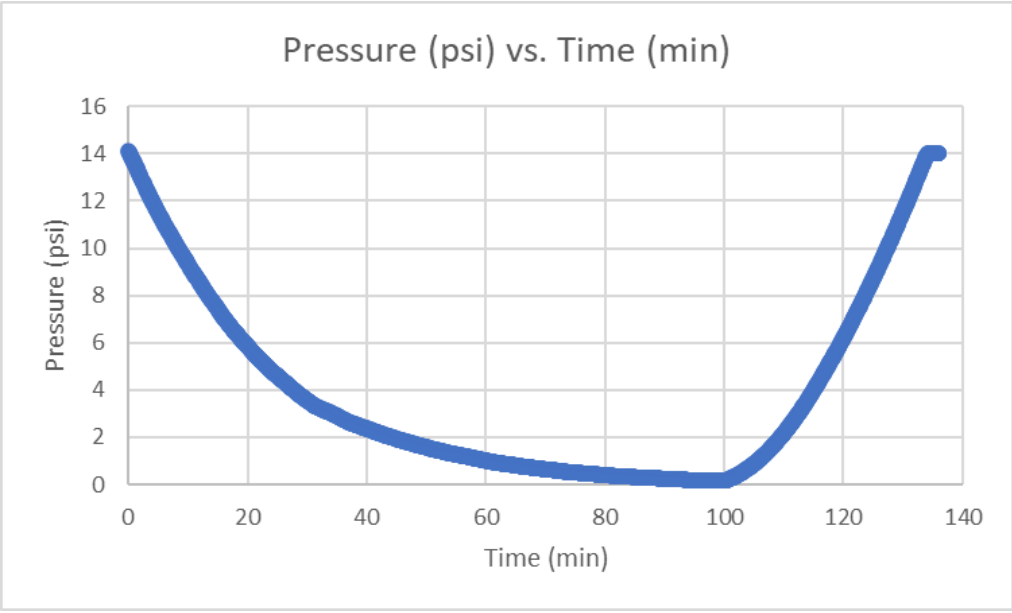


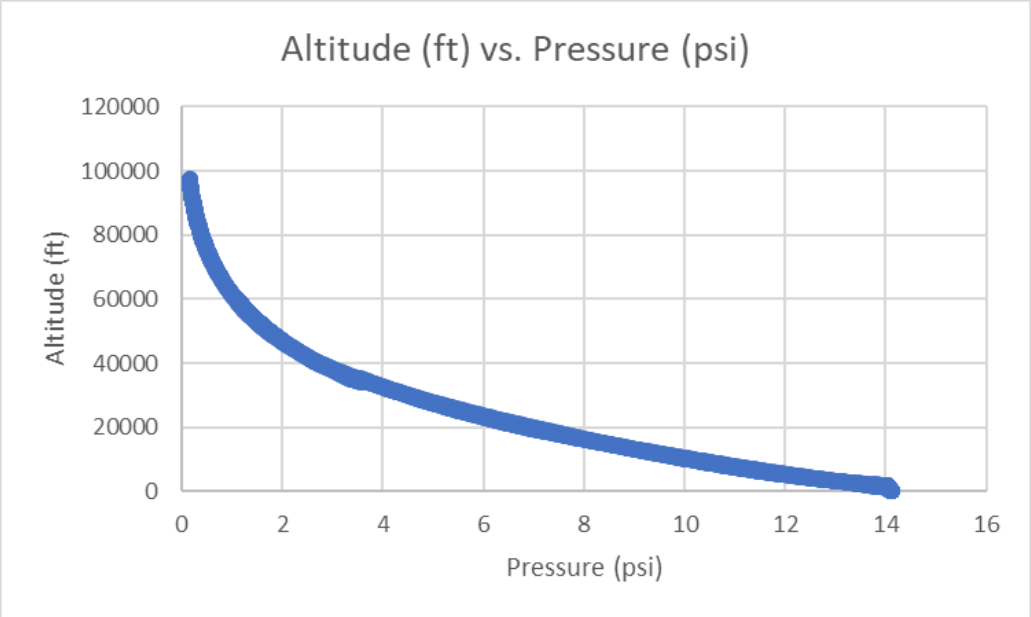
The upper graph is a graph of temperature as a function of time and the lower graph is a graph of altitude as a function of temperature. The progression of the temperature change looks as expected. The air cooled off to -75 degrees fahrenheit, then got warmer around 50,000 feet, reaching a peak of about -40 degrees fahrenheit, before dropping to its lowest point of -80 degrees fahrenheit at our maximum altitude of 100,000 feet.

PTERODACTYL Results

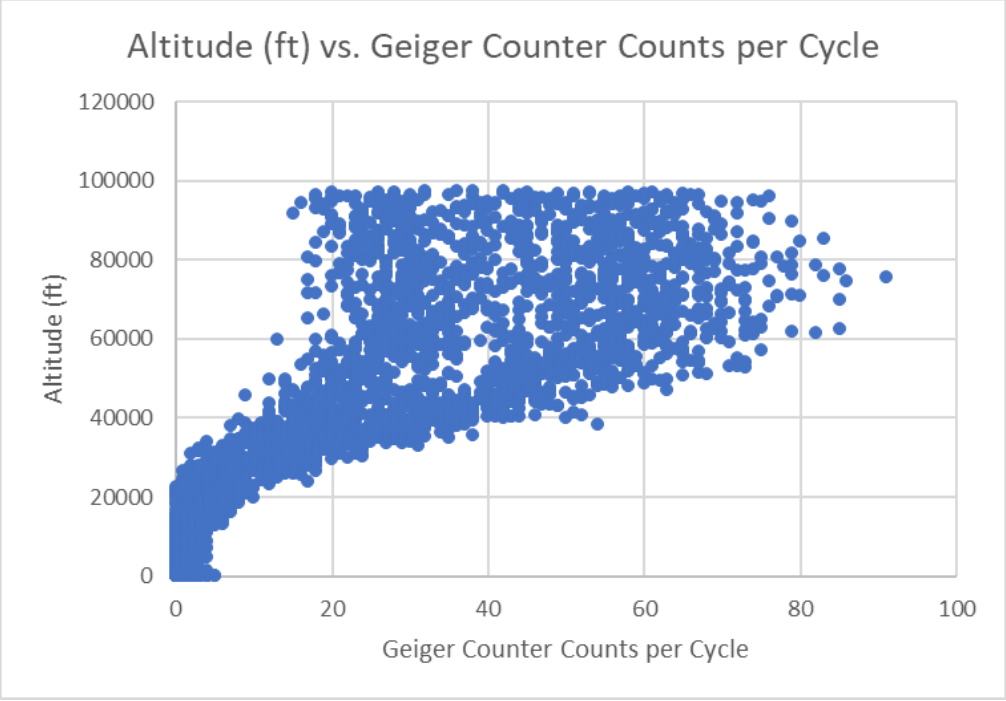


Beginning with the Altitude vs. Time graph, this data needed to be generated from the PTERODACTYL at the bottom of the stack as ours did not collect useful altitude data. However, this data seems to be an effective estimate as it is similar in overall shape and slope to the pressure-based altitude estimate generated by the PTERODACTYL.



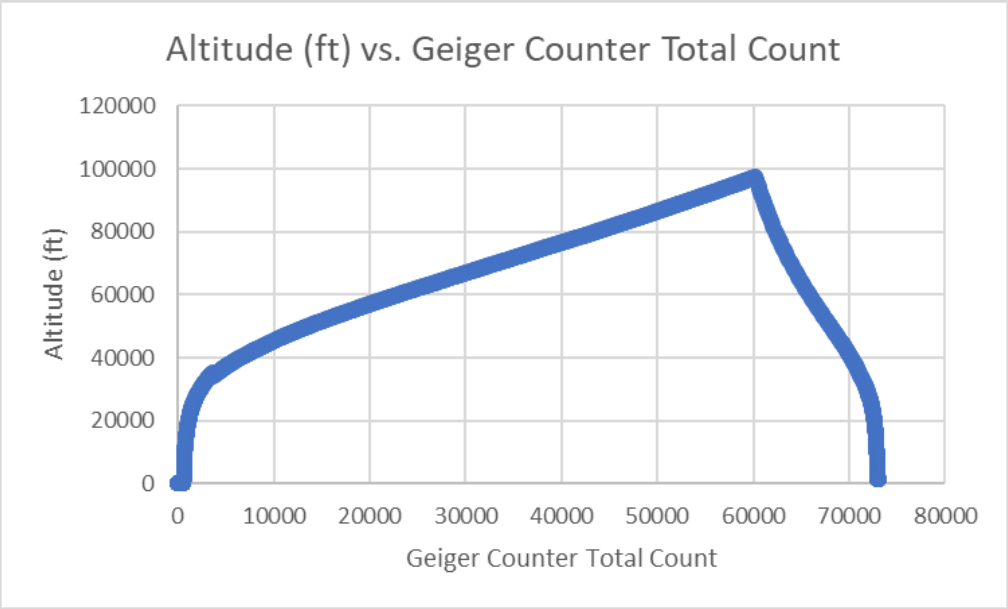


Moving on to the pressure data, the initial Pressure vs. Time graph supports the altitude graph as the pressure is at the lowest point when the altitude is at the highest. This is to be expected and follows our expected results quite nicely. This is further corroborated by the Altitude vs. Pressure graph which further supports our hypothesis that pressure decreases at higher altitudes.

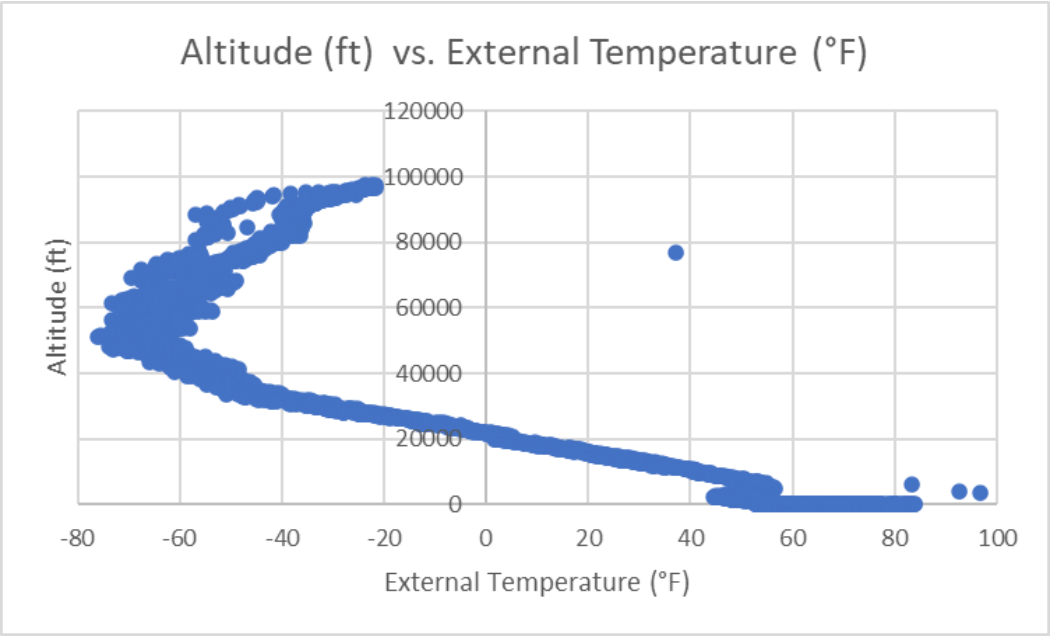


The Geiger counter data is among the most interesting. This graph shows how altitude influenced the number of hits the Geiger counter got per cycle. As expected, it does generally increase as altitude increases, suggesting that there is radiation being stopped by the atmosphere that doesn't make it to the ground. However, this data is

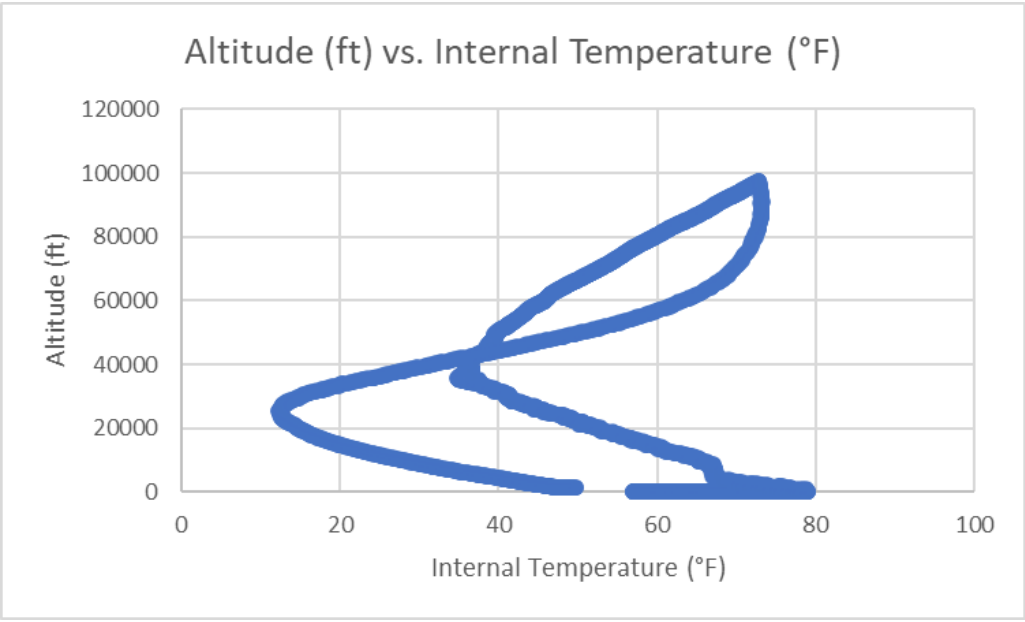
also slightly misleading as it suggests that the highest amount of radiation is at 80,000 ft. This doesn't make very much sense if the atmosphere is what is truly blocking radiation.



Fortunately, the Altitude vs. Geiger Counter Total Count graph sheds some light on this. If there really was more radiation around 80,000 ft, then the slope of this graph should be much steeper around 80,000 ft, but it is not. In fact, the slope generally stays constant between about 40,000 ft and 100,000 ft. This would suggest that the radiation detected by the Geiger counter at 80,000 ft is much the same as the radiation detected at 60,000 ft or 100,000 ft. The most likely reason for the "bulge" at 80,000 ft is that the payload was there twice whereas it was only at 100,000 ft once. Essentially, this means that it was twice as likely to encounter one or two high cycle counts at 80,000 ft than it was at 100,000 ft. 80,000 ft seems to be in a "Goldilocks zone" where radiation is high enough to create those high cycle counts, but also a region where they payload spends enough time to actually register them. The payload spends most of its time on the ground, but there is very little radiation to measure. Conversely, there is probably the most radiation at 100,000 ft, but the payload spends very little time up there. The fact that the counts per cycle tapers off after 70,000 ft could be an indication that there is less radiation to measure at that altitude or that the payload is falling faster and so has less time to measure it. Given that the slope of the Altitude vs. Total Geiger Count graph stays the same, it seems that the latter is more likely. However, a more controlled balloon and payload could better measure the radiation in each region as it could spend roughly the same amount of time in each zone and get cycle counts more representative of the actual radiation present.

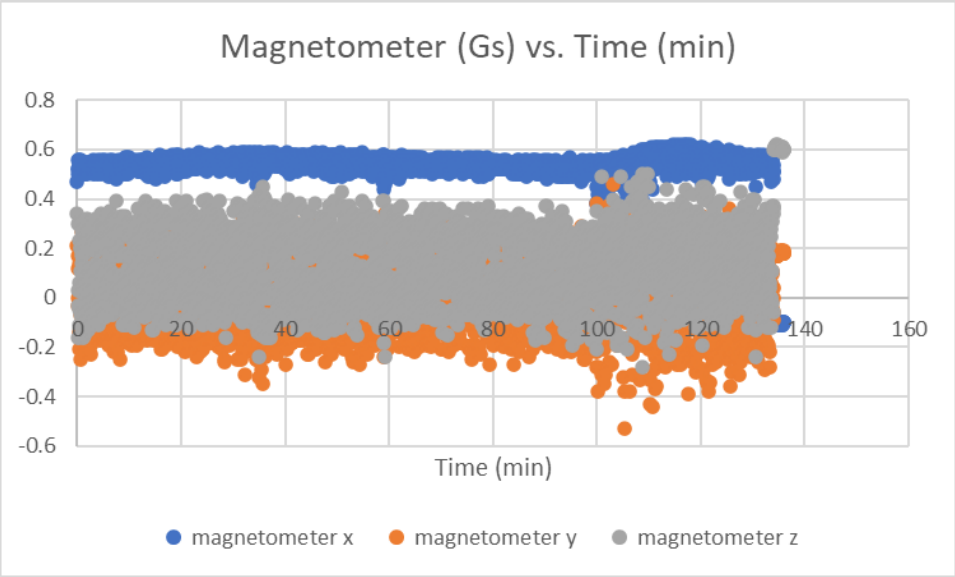


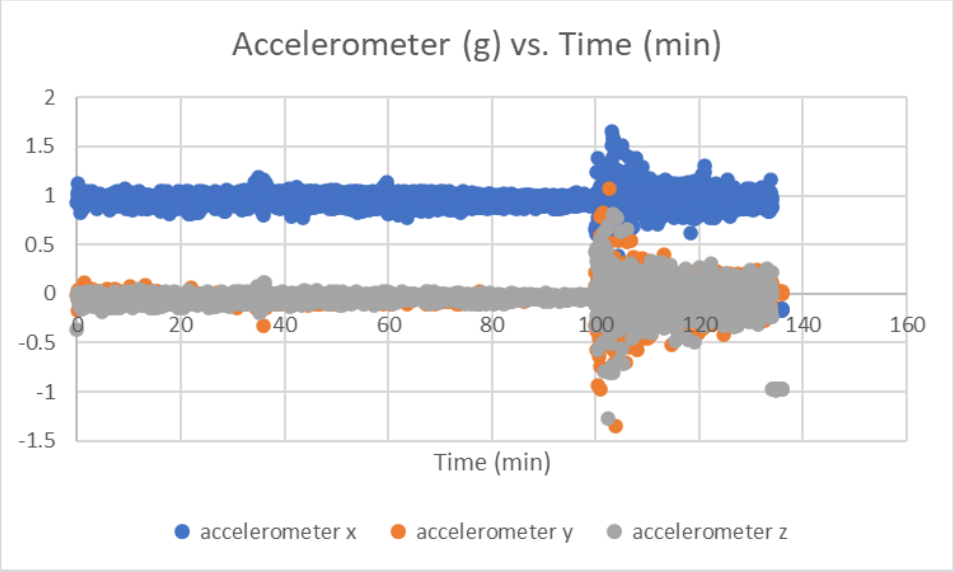
The external temperature data was a little disappointing as much of the data read in excess of 500 °F on the ground and as low as -750 °F in the upper atmosphere. However, disregarding the outliers and clearly bad data points, the Altitude vs. External Temperature graph tells a surprisingly clear story regarding temperature changes in the upper atmosphere, with the tropopause being surprisingly apparent at around 50,000 ft, which seems reasonable for this latitude. This data also follows our expected results for the temperature changes between the troposphere and the stratosphere.

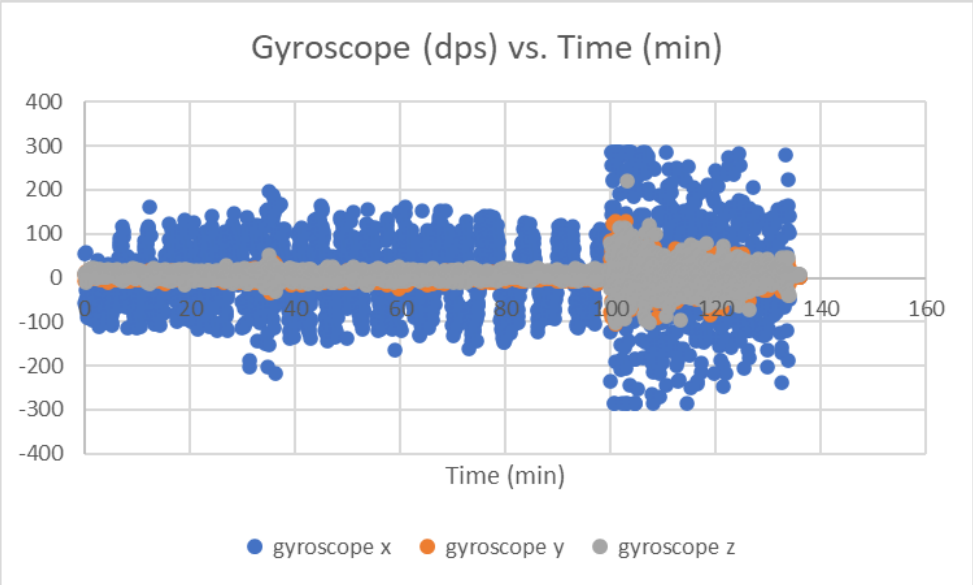


Rather opposite to the external temperature, the Altitude vs. Internal Temperature graph seems like good data on initial inspection, but seems to make less sense the

deeper into it you look. For one thing, the lowest temperatures on both the ascent and descent occur at an altitude lower than that of the tropopause. This could somewhat be explained away on the ascent by the fact that the handwarmers had yet to warm the payload sufficiently, and on the descent by the saying that the box has a resistance to change in temperature and so needs prolonged exposure to the cold before changing temperature rather than changing as the atmosphere changes. However, if there is a delayed reaction in cooling down, then shouldn't there also be a delayed reaction in warming up? The fact that there doesn't appear to be is interesting in the least. What is important to note is that the payload never seems to have gotten lower than 10 °F. It should be tested to see if having the hand warmers warmed up for a longer period of time prior to launch would push this number up even further. If so, that could help shed some light on the low point during the ascent and confirm that it was due to the hand warmers.







The IMU data gives information on the orientation and motion of the payload throughout the flight. Beginning with the easiest to interpret, the gyroscope data suggests a large degree of horizontal movement throughout the flight. The data even creates "bands" as would be expected if the payload rotated in a certain direction only to twist the string to far and then return in the other direction. This is supported by the accelerometer which shows near constant acceleration in the x-direction. All three cameras onboard the stack also suggested a large amount of horizontal rotation. All IMU sensors show a disruption at around 100 minutes. This is almost certainly when the balloon popped and sent the payload stack into disarray.

A brief note on the IR Qwiik Sensor; the way that data was collected from this sensor made it almost impossible to interpret. More will be discussed about this in the conclusion, however it is worth noting that data collection with this sensor is rather difficult on the payload as there is simply so much data to collect and interpreting the data without a visual is rather difficult as well. What can be said is that the sensor was able to register the earth as a higher temperature than outer space with many of the pixels at the top of the "image" having no data at all as the temperature was too low.

11.0 Conclusions and Lessons Learned

As a group, we learned plenty of lessons over the course of building and flying our payload. We learned about the concept of pressure and how it is affected by temperature. We slightly learned to program. We also learned how to come together and work as a team on something quite difficult. If we had the chance to do such a project again, we agreed to research light levels in higher altitudes and Radiation levels in higher atmospheres mainly. We knew little about these topics coming into this class and we should have researched more before flight day. Some of us also struggled with programming. Programming would require plenty of research and

practice to master. It would've been very helpful if we had prior knowledge about programming.

The Qwiik sensor was rather difficult to work with for two reasons: firstly, it records a large amount of data and secondly, that data is designed to be interpreted in a visual interface such as an image or video. The lack of said image or video makes it difficult to understand what the numbers mean. If we were to do this again, rather than sampling every 5th pixel, it would instead be helpful to record data such that the overall resolution is decreased. For example, if you had a square, 11x11 image, rather than counting off the pixels row-by-row and picking the 5th one, instead throw out the 2nd, 4th, 6th, 8th, and 10th row and column. That would turn an 11x11 image into a 6x6 image, decreasing the resolution and making it manageable. In taking every 5th pixel, you are essentially left with simply an assortment of pixels without any context or position. Multiple attempts were made to locate and place each pixel, however this was tedious and never reached a satisfactory image.

The second thing we would do if we were to fly the IR Sensor again would be to only fly the IR Sensor and the GPS data on the PTERODACTYL. That way, you would minimize the amount of data collected and hopefully lead to more consistent sampling rates and more usable IR data. The third (and likely most important) thing to do differently is have a plan in place for how these raw numbers will be translated into an image (and ideally video) when back on the ground. This is more of a camera than a sensor and looking at each individual pixel is rather like looking at an image, but instead of seeing the color of each pixel, the color is written out in a paragraph with commas to separate each pixel. In such a situation, it would be very difficult to understand what you were looking at and such is the case with the way the IR data was processed. This was a cool sensor and fun to program, but if I were to launch another balloon, I don't think I would send his sensor up without excessive ground testing and practice first.

- Invest your time into every part, it pays off in the end
- Don't overlook research, or you'll do the research later
- Very fun once you see the outcome of your flight

12.0 References

1. The Editors of Encyclopedia Britannica. atmospheric pressure. https://www.britannica.com/science/atmospheric-pressure (accessed 10/26/2021)

2. Center for Science Education. Change in the Atmosphere with Altitude. https://scied.ucar.edu/learning-zone/atmosphere/change-atmosphere-altitude (accessed 10/26/2021)

3. Dr. Tony Phillips. Global Cosmic Radiation Measurements. https://spaceweatherarchive.com/2018/06/10/global-cosmic-radiation-measurements/ (accessed 10/26/2021)

4. Dr. Malanie Windridge. Ultraviolet Radiation at Altitude. https://melaniewindridge.co.uk/mountain-science/mountain-science-preliminary-results-3.html (accessed 11/20/2021)

13.0 Appendix: Program Listings

13.1 Flight Code:

This is the main flight code. It is largely responsible for calling other functions from the other program tabs and controlling the higher-level PTERODACTYL functions such as running diagnostics. It is essential that this code is clean and streamlined so that the flight runs smoothly.

```
//Team B Flight Code
//Complete: Rev 28
//Based on heavily modified PTERODACTYL flight code.
//PTERODACTYL flight code written by Andrew Van Gerpen.
//Modifications made by Paul Wehling.
//Team B flight code written by Ethan E. Cederberg for use with Team B's payload.
//Target flight date: 10/30/2021

#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;

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, geiger cycle count, geiger total count, temp pixels";
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 irData;

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 Geiger setup...");
  geigerSetup();
```

```
  Serial.print("Geiger setup complete");

  Serial.print("starting IR setup...");
  irSetup();
  Serial.print("IR 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");

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

void loop() {
  updateData();
  updateStatus();
}

////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////// 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="";
   }
  }

  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);
  }
```

13.2 Sensor Code:

The second code which I modified was the sensor code. This code is called in the main flight code and is responsible for initializing all of the sensors and collecting data from them. This tab is the longest of the four tabs as it does most of the things that are required of the PTERODACTYL.

```
#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 <GeigerCounter.h> //Includes geiger counter library
#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 "MLX90640_API.h"  //Drivers for IR Sensor
#include "MLX90640_I2C_Driver.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

int lineNumber = 0;
byte Statusbyte;
String satData;

bool geigerActive = false;

bool irActive = false;

MS5611 baro;
MicroOLED oled(PIN_RESET, DC_JUMPER);    // I2C declaration
LSM9DS1 imu;
UbloxGPS ublox(&ubloxSerial);

///Geiger Counter Setup///
GeigerCounter geiger = GeigerCounter(1); //Assigning Geiger counter to pin
unsigned long totalCount = 0, cycleCount = 0; //Creating global variables for printing
///End Geiger Setup///

/////////////// 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

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
///IR Sensor Setup///
const byte MLX90640_address = 0x33; //Default 7-bit unshifted address of the MLX90640
#define TA_SHIFT 8 //Default shift for MLX90640 in open air
static float mlx90640To[768];
paramsMLX90640 mlx90640;

boolean isConnected()
{
  Wire.beginTransmission((uint8_t)MLX90640_address);
  if (Wire.endTransmission() != 0)
    return (false); //Sensor did not ACK
  return (true);
}
//End IR Sensor Setup//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////

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 geigerSetup(){
  geiger.init();
  geigerActive = true;
}

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 irSetup(){


  Serial.begin(9600);
  while (!Serial); //Wait for user to open terminal
  Serial.println("IR Array");

  if (isConnected() == false)
  {
      Serial.println("MLX90640 not detected at default I2C address. Please check
wiring. Freezing.");
    while (1);
  }
  Serial.println("MLX90640 online!");

  //Get device parameters - We only have to do this once
  int status;
  uint16_t eeMLX90640[832];
  status = MLX90640_DumpEE(MLX90640_address, eeMLX90640);
  if (status != 0)
    Serial.println("Failed to load system parameters");

  status = MLX90640_ExtractParameters(eeMLX90640, &mlx90640);
  if (status != 0)
```

```
    Serial.println("Parameter extraction failed");

  //Once params are extracted, we can release eeMLX90640 array
  irActive = true;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

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

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

  totalCount = geiger.getTotalCount();
  cycleCount = geiger.getCycleCount();

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
  //Ir Sensor Stuff//
  for (byte x = 0 ; x < 2 ; x++) //Read both subpages
  {
    uint16_t mlx90640Frame[834];
    int status = MLX90640_GetFrameData(MLX90640_address, mlx90640Frame);
    if (status < 0)
    {
      Serial.print("GetFrame Error: ");
      Serial.println(status);
    }

    float vdd = MLX90640_GetVdd(mlx90640Frame, &mlx90640);
    float Ta = MLX90640_GetTa(mlx90640Frame, &mlx90640);

      float tr = Ta - TA_SHIFT; //Reflected temperature based on the sensor ambient
temperature
    float emissivity = 0.95;
```

```
            MLX90640_CalculateTo(mlx90640Frame, &mlx90640, emissivity, tr,
mlx90640To);
  }

//  for (int x = 0 ; x < 10 ; x++)
//  {
//    Serial.print("Pixel ");
//    Serial.print(x);
//    Serial.print(": ");
//    Serial.print(mlx90640To[x], 2);
//    Serial.print("C");
//    Serial.println();
//  }

   irData = String(mlx90640To[0]) + ",";
   for (int x = 5 ; x <= 768 ; x = x + 5){
    irData = irData + String(mlx90640To[x]) + ",";
   }


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

    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]) + "," + String(cycleCount) +
          "," + String(totalCount) + "," + irData + xbeeMessage;
```

```
    satData = "|" + String(Statusbyte) + "," + String(lineNumber) + "," +
String(cycleCount) + "," + String(thermistorExt) + "," + String(msPressure);

    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(Statusbyte, 7, 0);
  bitWrite(Statusbyte, 6, fix);
  bitWrite(Statusbyte, 5, sdActive);
  bitWrite(Statusbyte, 4, thermistorInt < 120);
  bitWrite(Statusbyte, 3, thermistorInt > 40);
  bitWrite(Statusbyte, 2, geigerActive);
  bitWrite(Statusbyte, 1, cycleCount > 0);
  bitWrite(Statusbyte, 0, irActive);
}
```