University of Notre Dame

2017 - 2018

Notre Dame Rocket Team

Flight Readiness Review

NASA Student Launch 2018

Deployable Rover and Air Braking System Payloads

Submitted March 5, 2018

365 Fitzpatrick Hall of Engineering

Notre Dame, IN 46556

Table of Contents

List of Figures

List of Tables

1 Summary of CDR Report

1.1 Team Summary

Team Name: Notre Dame Rocketry Team

365 Fitzpatrick Hall of Engineering

Notre Dame, IN 46556

NAR Mentor: Dave Brunsting, NAR/TAR Level 2

dacsmema@gmail.com or (269) 838 - 4275

NAR/TRA Section: TRA #12340, Michiana Rocketry

1.2 Launch Vehicle Summary

1.2.1 Size and Mass

Table 1 below, shows the sizes of various components of the Launch Vehicle. These parameters now reflect the physical build of our rocket, and have been verified and updated in OpenRocket. The launch vehicle begins with a larger outer diameter, and narrows with a transition section to the smaller diameter that is constant for the rest of the body. This transition section is required to house the rover bay in the vehicle. The overall length of the rocket is 136 in, which is necessary to house both payloads, the recovery system, as well as add stability to the vehicle. Due to the material change, length has been added to the rocket, as well as additional ballast to lower the apogee. This added ballast is less than 10% of the vehicle's total weight.

Table 1. Length of Various Rocket Components.

Component	Dimension
Nose Cone [in]	22* (+5in shoulder)
Rover Bay [in]	20*
Transition Section [in]	4* (+6in shoulder)
Body Tube [in]	42*
Roll Control Payload Coupler [in]	19
Roll Control Mount [in]	3*

Fin Can [in]	32.5*
Engine Retention [in]	0.74*
Total Length [in]	136

*Indicates that dimension adds to total length

Table 2. Overall Rocket Properties.

Size	Length: 136 in Outer Diameter (Fore): 7.74 in Inner Diameter (Fore): 7.5 in Outer Diameter (Aft): 5.54 in Inner Diameter (Aft): 5:38 in Number of Fins: 4 Fin Span: 6.0 in
Mass	Loaded Weight: 45.25 lbs. (724 oz.) Weight without Motor: 35.56 lbs. (569 oz.)

1.2.2 Final Motor Choice

The Cesaroni L1395-BS (blue streak) was selected and has been acquired for competition. Important parameters for this motor are shown below in Table 3. The final OpenRocket simulations show an apogee of 5405 ft. with 10 mph winds and standard atmospheric conditions, which is both under the 5600 ft flight ceiling and within the altitude range for the Air Braking System to lower the vehicle to the desired 5280 ft. This altitude was initially greatly increased due to the change in materials and resulting decrease in mass. However, it was brought back within the desired range through the addition of ballast.

Table 3. Cesaroni L1395-BS Motor Characteristics.

Property	Dimension
Peak Thrust (lbf)	400
Average Thrust (lbf)	314
Total Impulse (lbf*s)	1101

Due to motor availability, the Cesaroni L1115 was used in the team's full scale test launch. This motor has similar characteristics to the L1395 and has the same diameter and length. For comparison, the L1115 has an average thrust of 252 lbf, a peak thrust of 385 lbf, and a total impulse of 1127 lbf*s.

1.2.3 Rail Size

As was reported in CDR, the team will be using a 12 feet long and 1.5 inch wide launch rail, which gives a simulated off rail velocity of 72 ft/s. This is safely above the minimum required velocity of 52 ft/s. The main body of the rocket will be attached to this rail using two airfoil shaped rail buttons mounted 1.3 inches from the fin can on 3D printed airfoil shaped blocks to keep the larger diameter forward section of the rocket from interfering with the rail. This setup is shown below in Figure 1. As the drag tabs of the Air Braking System will not be deployed until after motor burnout has occurred, they will not interact with the rail.

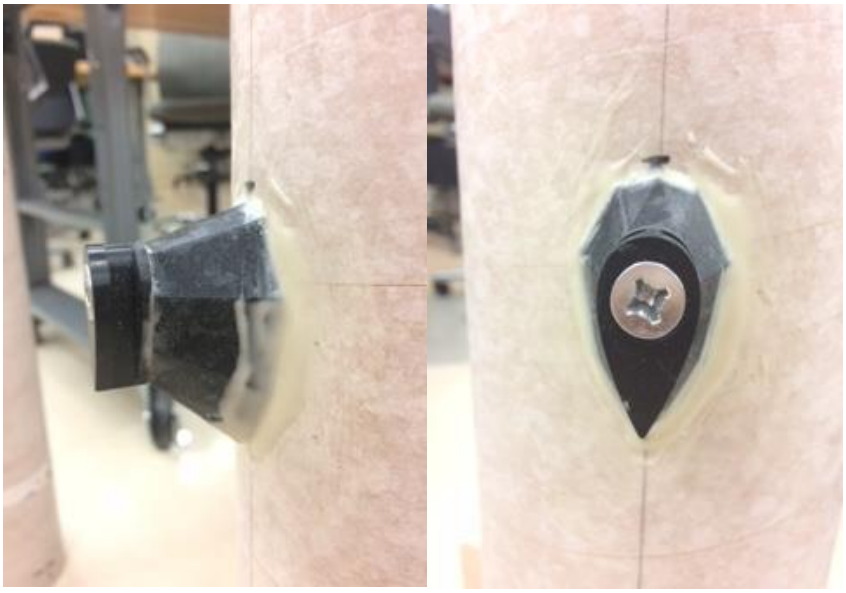


Figure 1. Rail Button mounted on 3D printed attachment block.

1.3 Recovery System Summary

The recovery subsystem is a dual deployment system consisting of a drogue parachute to be released at apogee and a main parachute to be released at 650 feet above ground level. When the charges are detonated, the rocket will separate at designated points secured with shear pins. The separated sections will remain tethered together by means of shock cords attached to integration eyebolts. Deployment of the black powder charges will be controlled by three independent subsystems of redundant altimeters, power sources, and e-matches to ensure parachute deployment in case of unanticipated complications.

1.4 Deployable Rover Payload Summary

Since the CDR three small changes have been implemented into the Deployable Rover Payload. The first was the result of a team wide supplier change. Due to the original supplier backing out on the order a new manufacturer was used for the fiberglass body tube. The body tube purchased from this supplier had a smaller inner diameter than initially chosen. The difference was 0.05 inches and mainly affected the tolerances of the 3D printed parts. The new tolerances were updated and printed to reflect this change. The change in manufacturer also led to a change in bulkhead material. The bulkheads are now cut from plywood instead of fiberglass. The second change involved the motors used to drive the rover. In order to prevent stalling, a brushed motor with an internal gearbox was chosen. The rover is now driven by four XL Lego motors. The third change involves the deployment electronics. Rather than mounting them on the back bulkhead, they are now mounted on the rover. This provides a secondary form of disarming a faulty charge after the rover is deployed. These changes are detailed further in Section 4.1.

1.5 Air Braking System Summary

The purpose of the air braking system is to assist the vehicle with reaching an apogee of exactly 5280 ft. This has been done using three different subsystems working together. The aerodynamic subsystem is used to induce an additional, controllable drag force on the rocket using four drag tabs, thus allowing the overall drag on the rocket to be manipulated as necessary. The aerodynamic subsystem is connected to the mechanical subsystem, which uses a central shaft driven by servo motors to manipulate the drag tabs. The servo motors are controlled by the electronic subsystem, which runs a control code to adjust the induced drag as needed.

2 Changes Made Since CDR

2.1 Changes Made to Vehicles Design

2.1.1 Materials

The vendor that the team had planned to use to get its carbon fiber and fiberglass materials was behind schedule, and was not able to ship materials by the Flight Readiness Report due date. Therefore, the team looked into the materials previously researched for the rocket during Proposal and Preliminary Design Report. These choices were phenolic and fiber glass. Table 4 shows a material properties summary for all of these choices.

Table 4. Material Properties for Carbon Fiber, Fiberglass and Phenolic.

Material	Component Use	Density (lb/in^3)	Tensile Strength (ksi)	Tensile Modulus (msi)	Shear Modulus (msi)	Com-pressive Strength (ksi)	Com-pressive Modulus (msi)	Specific Weight (lb/in^3)
Carbon Fiber	--	0.0578	300-350	15-30	0.6-.0725	82-120	18.5	0.065
Fiberglass	Body Tube, Transition Section	0.055	250-300	0.8-1.4	4.351	140-350	--	0.063
Phenolic Paper	Body Tubes, Couplers	--	12-15	--	--	32	--	0.049
Birch Plywood	Fins, Bulkheads, Centering Rings	0.024	13	--	2.2	10	--	--

2.1.1.1 Body Tubes

Upon realization of the new material constraints, multiple different options were taken into account. Rockwest and Apogee Components were both considered as viable options to purchase body tubes. Rockwest is a reliable vendor that the team has used in years past. They are a supplier of carbon fiber and fiberglass, and have stock tubes ready to be cut and sent, making the time between ordering and delivery minimal. Apogee Components is also a reliable vendor that the team has used for many of its parts. They sell a variety of fiberglass and phenolic body tubes from stock and also have a quick delivery time.

The team decided due to buy its new materials from Apogee Components. This decision was driven by both budgetary, time, and stock constraints. Phenolic is much cheaper than carbon fiber, and while it is weaker than carbon fiber by a factor of 10, it has a strength that is satisfactory for the mission it will be performing. Phenolic is widely used in high powered rocketry, and the team has used these materials in previous years with success. Additionally, phenolic was chosen due to carbon fiber in the sizes needed being unavailable, as well as the shipping times being much quicker.

2.1.1.2 Fins and Bulkheads

The fins and bulkheads were to be made of carbon fiber and fiberglass, respectively. The new material constraints forced the team to consider different options. Again, the team looked to the availability of materials as well as materials that have been used in the past with success. Birch plywood was chosen due to its availability at local home improvement stores and its use in the past. The material properties of birch plywood can be found in Table 4 above at the beginning of Section 2.1.1. Both the fins and bulkheads were CNC milled in Notre Dame's Stinson-Remick Hall.

Additionally, due to the change in weight because of the lighter phenolic, the stability needed to be changed from the new value of 3.4. Therefore, ballast was first simulated to change the Center of Gravity of the rocket, but this had a minimal effect due to the amount of ballast the team was able to use. Therefore, it was decided that the best was to change the stability of the rocket was to change the dimensions of the fins. It was because of this that the team has changed the height of the fins above the outside of the fin can from 7.2 inches to 6.0 inches.

2.1.2 Transition Section

The four inch long transition section that couples the larger 7.5" diameter section to the aft 5.5" diameter section was among the materials that would not be able to be delivered by the launch date. Therefore, other options have been considered to couple the two body tube sections. Primarily, Summit Aviation constructed the originally proposed fiberglass transition section to the dimensions that were previously planned in the Critical Design Report. This manufactured part also includes the correct coupler lengths on each end (0.75 calipers on a non-separating section and 1.0 caliper on a separating section).

2.1.3 Weight

With the materials changes to the rocket, and slight structural change to the transition section, the rocket weights have also changed. Without a motor, the newly designed rocket weighs 569 oz (35.56 lbs), and is 136 inches long. The rocket will also utilize ballast in the form of sand in order to increase the weight of the rocket and to decrease predicted apogee. Approximately 25 oz of sand will be used to do this, and this is under the maximum amount of ballast by the competition requirements (10% of weight without motor). This ballast will be placed in the air braking system coupler with the same design as the previous ballast retention system. The overall weight of the rocket with ballast and motor will now be 746 oz (46.63 lbs).

2.2 Changes Made to Recovery System

Since CDR, there have been virtually no changes to the recovery system design criteria. The only notable change is the directionality of the altimeters in the way they are secured within

the core of the CRAM. Originally, the design called for the altimeters to be secured to the central column structure within the core as pictured below in Figure 2. However, prototypes of the system revealed an oversight in construction planning because it was actually impossible to secure and remove the altimeters due to physical clashing with the battery boxes. To remedy the situation, the core piece was partially redesigned so that (1) the altimeters face the opposite direction, (2) the altimeters are secured to a narrow intermediate wall, and (3) the core is divided into two halves. This is considered a minor change because the altimeters have not actually moved from the originally intended location, they have merely been rotated 180 degrees within the same space. This new design can be seen below in Figure 3. This design change is not expected to impact the functionality of the recovery system in any way.

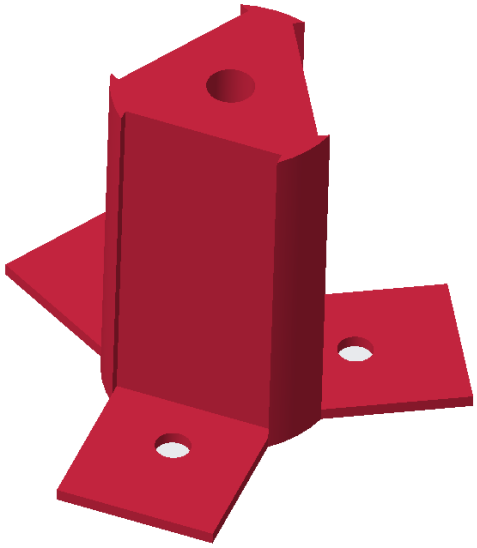


Figure 2. CDR core iteration.

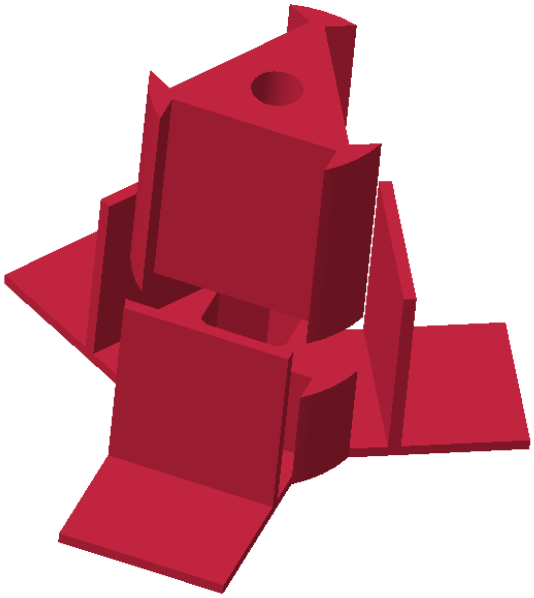


Figure 3. FRR core iteration showing intermediate walls and split column design.

2.3 Changes Made to Deployable Rover Payload

There are no deployable rover payload changes since CDR.

2.4 Changes Made to Air Braking System

No changes to the system requirements have been made since the critical design review. The goal of the system is to assist the vehicle with reaching an apogee of exactly 5280 ft. This has remained the purpose of the system not only during competition but also during test flights. However, an additional goal has been added to test flights in particular. Their purpose has also been to collect data on the system performance so that improvements can be made to increase the effectiveness and consistency of the system.

2.5 Changes Made to Project Plan

3 Vehicle Criteria

3.1 Design and Construction of Vehicle

3.1.1 Mission Statement

The mission is to successfully design, construct and launch a rocket carrying a deployable rover to an altitude of exactly 5,280 ft above ground level. An air braking system consisting of

four drag tabs will be used to induce varying amounts of drag to ensure success in meeting the exact altitude goal. The amount of drag induced will be dictated by the flight conditions experienced by the rocket after motor burnout and before apogee. A rover contained in a section directly below the nose cone will deploy upon landing and be able to drive at least 5 feet away from any part of the launch vehicle. The recovery systems will allow the rocket to separate into 3 sections and deploy both a drogue and a main parachute to secure a safe descent and landing while preserving recoverability by keeping drift to acceptable levels. The vehicle and its payloads must be reusable on the same day without need for repairs or modifications. The team also will make an impact on the local community of South Bend and Notre Dame through educational events with area youth and media presence.

3.1.2 Mission Success Criteria

Several conditions must be met for the mission to be considered a success. The following criteria have remained constant, and are the team's main design drivers throughout this process and have been considered in all design choices and verification models.

The dominant criteria for a successful mission are:

1. *Altitude*: The vehicle must reach an apogee of as close to 5280 ft as possible. Success of this criterion will be determined based on readings from an altimeter onboard the rocket. A desirable range is 5280 ± 100 ft, or 5180-5380 ft.
2. *Stability*: The rocket must maintain an acceptable degree of stability for the duration of its flight. While the vehicle is required to have a stability margin of at least 2.0, excessively large static margins can cause undesirable flight performance. Stability is determined theoretically with OpenRocket and RockSim models.
3. *Structural Integrity*: The vehicle must remain intact for the duration of its flight. Each component of the rocket from the motor retention and the internal bulkheads to the drag tabs on the air braking system and the onboard rover must survive the flight without compromise and be in acceptable condition to be reused without modification or repair.
4. *Recovery*: The vehicle must be reusable upon recovery without requiring repairs. Success in recoverability is predicted by the kinetic energy of each section upon landing based on simulation data. Recoverability of the rocket will be determined based on the condition of each component after the rocket lands.
5. *Rover Payload*: The rover payload must safely deploy from the internal structure of the launch vehicle when remotely triggered after landing, move 5 ft away from all rocket components, and deploy a set of foldable solar cell panels. Success of the rover payload is determined by GPS coordinates before and after movement and by the level of solar charge on the panels.
6. *Air Braking System*: The air braking system must successfully deploy its four drag tabs based on conditions of flight in order to slow the rocket to reach the goal apogee. Success of the air braking system will be determined based on the difference between the actual and predicted apogee of the rocket as well as the onboard computers logging the actions of the air braking payload servo motors.

3.1.3 System Level Review

3.1.3.1 Overview of Vehicle Design

The launch vehicle will have a variable diameter with a 7.74 inch fore diameter and a 5.54 inch aft diameter. The transition section will be 4 in long and will transition linearly from one diameter to another. The total weight of the rocket is approximately 746 ounces, and the total length is 136 in. The four fins, placed 90 degrees from one another, have a root and tip chord of 7 in, a height of 6.0 in, and a sweep angle of 30 degrees. These dimensions are shown in Tables 5 and 6 below. The center of gravity is 79.99in aft of the nose cone and the center of pressure is 102 in aft of the nose cone.

The rocket consists of 5 subsections: the nose cone, the Deployable Rover payload Bay, the Transition Section, the secondary body tube and the parachute bay, the Air Braking System (ABS), and the fin can. The nose cone, made of polypropylene, is 22 in long with a 4 in shoulder. The Deployable Rover Payload Bay contains all of the components necessary for rover deployment, including altimeters and accelerometers. The transition section allows for the connection of the Deployable Rover Payload bay and secondary body tube. This body tube is then connected with a coupler to the parachute bay, which contains the Compact Removable Avionics Module (CRAM) and both the main and drogue parachutes. The ABP contains 4 tabs and a gear system used to extend them. The fin can contains the motor and the 4 main fins.

The 5 subsections of the rocket fall into 3 main sections based on how they separate during the descent of the rocket. At apogee, Section I separates from Section II and the drogue is ejected. At 650 feet AGL, Section II separates from Section III and the main parachute is deployed. Section I contains the nose cone, the Deployable Rover Payload Bay, and the transition section. Section II is the parachute bay, and Section II is made up of the ABP and fin can. Figure 4 shows the sections and subsections. Figures 5 and 6 contain more detailed CAD drawings of the rocket.

Table 5. Launch Vehicle Dimensions.

Property	Dimension
Total Length [in]	136
Outer Fore Diameter [in]	7.74
Inner Fore Diameter [in]	7.5
Outer Aft Diameter [in]	5.54
Inner Aft Diameter [in]	5.38

Number of Fins	4
Fin Cord Length [in]	7
Fin Height [in]	6
Fin Thickness [in]	0.25
Weight with Motor [lbs / oz]	46.625 / 746
Weight without Motor [lbs / oz]	37.125 / 594
Stability with Motor	4.3
Stability without Motor	2.86

Table 6. Length of Rocket Components.

Component	Length (in)
Nose Cone	22 (+ 5inch shoulder)
Rover Bay	20
Transition Section	4 (6 inch shoulder on each side)
Secondary Recovery Tube	12
Tube Coupler	11
Primary Recovery Tube	42
Air Braking Coupler	12
Air Braking System	4
Fin Can	32.5
Engine Mount	26.5

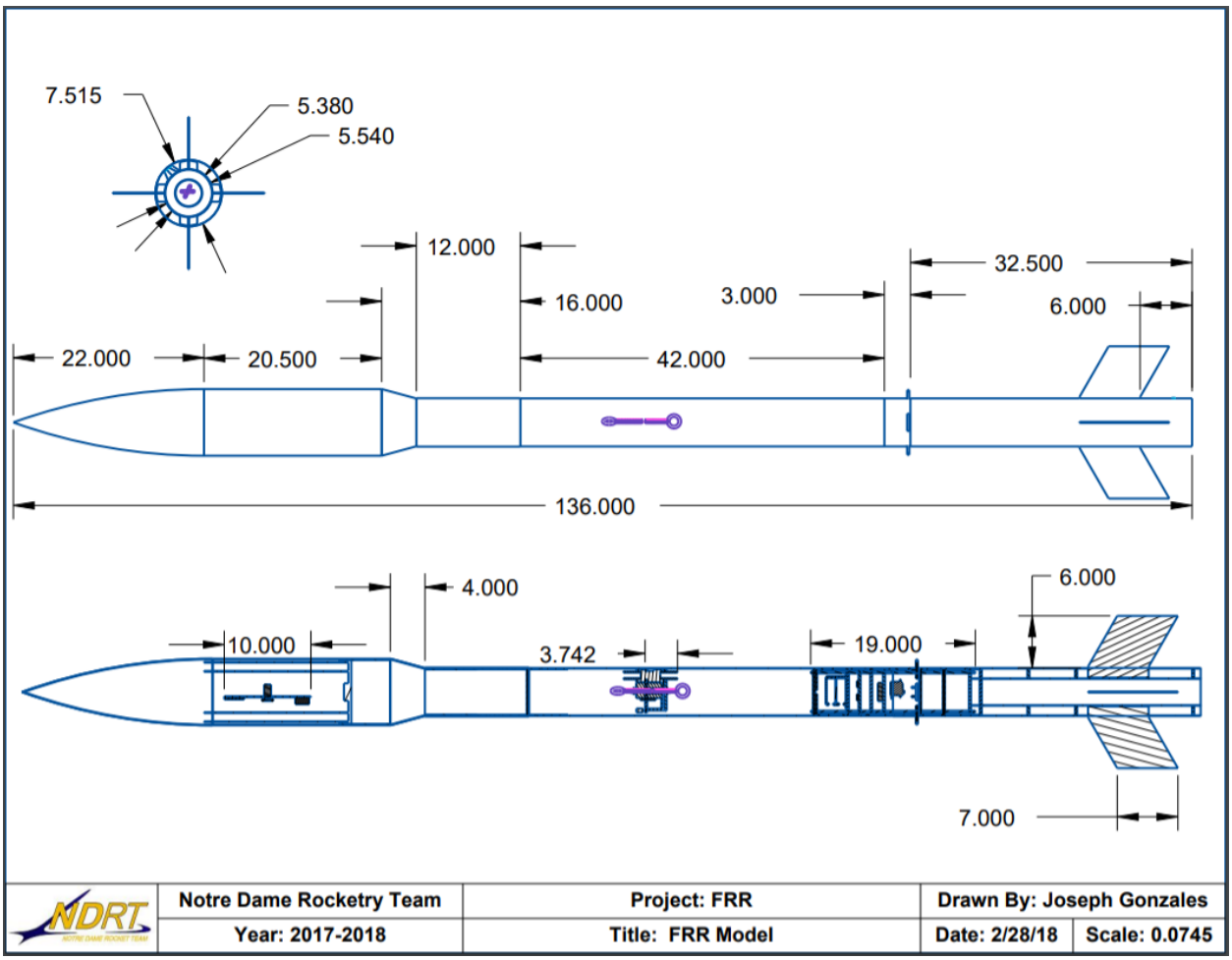


Figure 4. Dimensioned drawing of the overall Launch Vehicle. Units in inches.

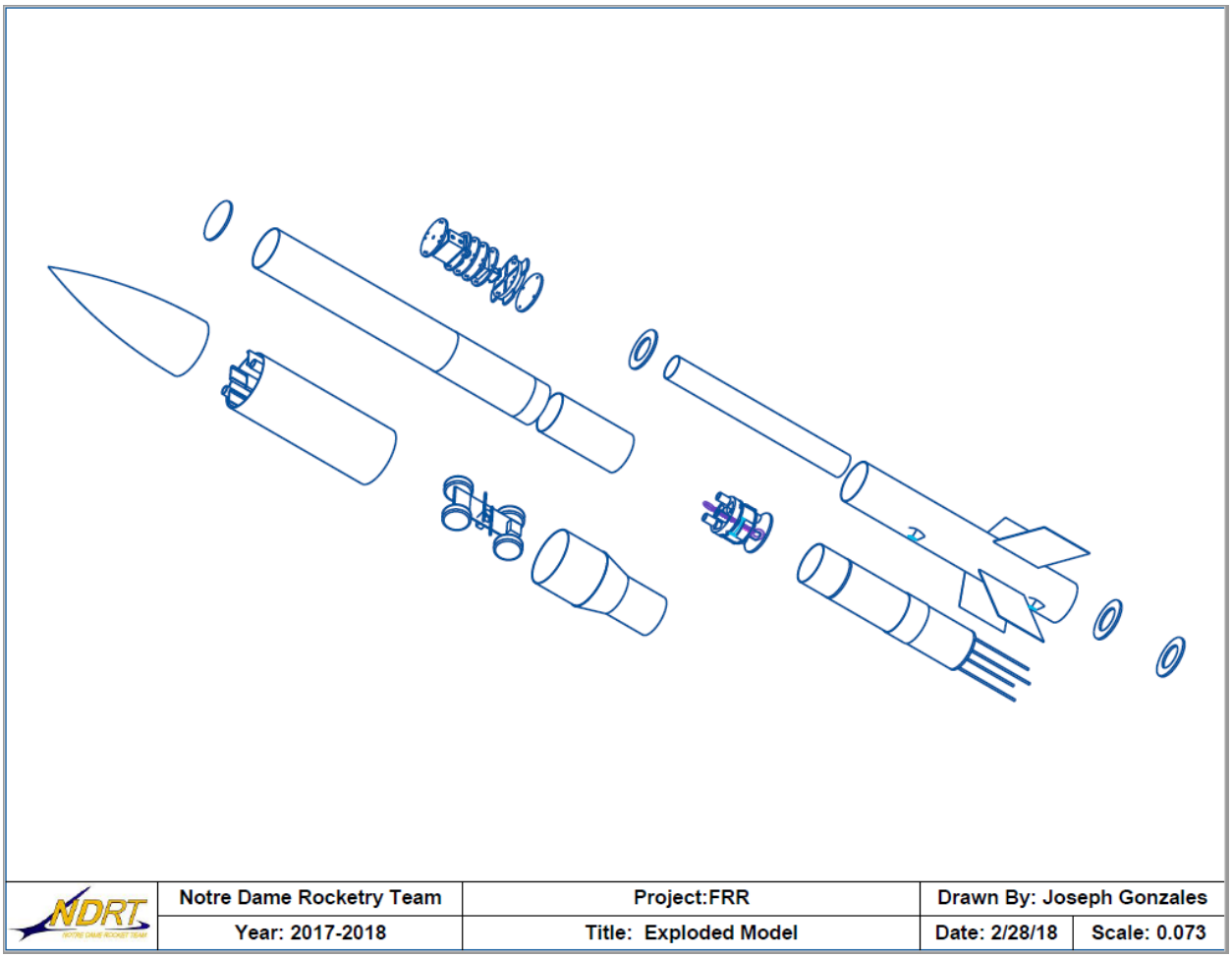


Figure 5. Exploded drawing of the overall Launch Vehicle. Units in inches.

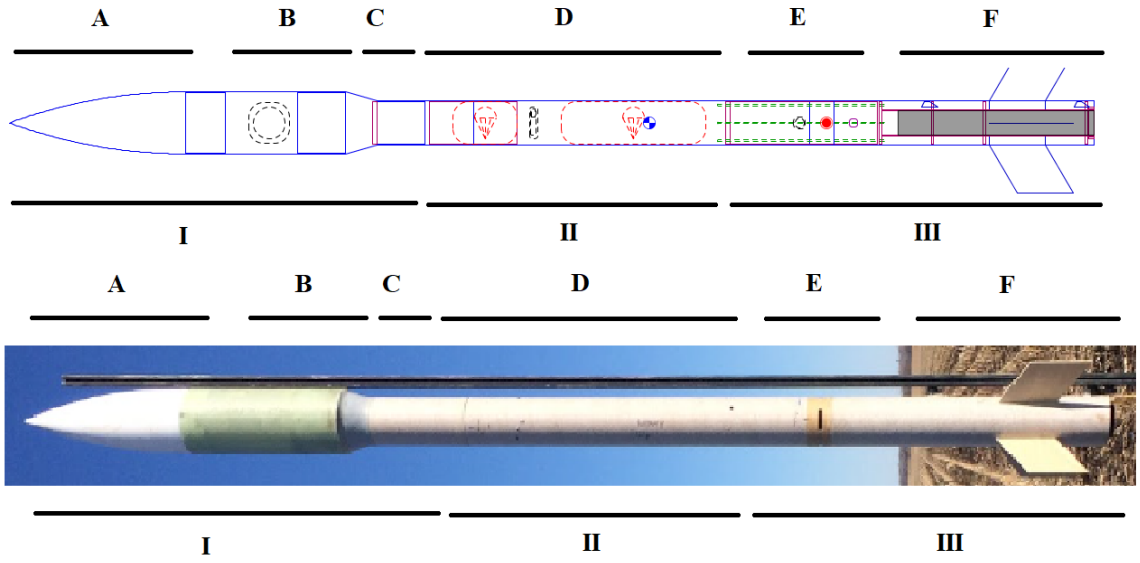


Figure 6. Diagram of Launch Vehicle Sections and Sub-Sections.

Table 7. Description of Launch Vehicle Sections and Sub-Sections.

Section	Sub-Section	Label	Composed of	Description
I	Nose Cone	A	Hollow nose cone, 22" in height and 7.675" in diameter, made of polypropylene	Connected to the deployable rover payload bay below
	Deployable Rover Payload Bay	B	20" of fiberglass body tube	Holds Deployable Rover Payload
	Transition Section	C	4" of fiberglass	Connects Deployable Rover Payload and Parachute Bay below
II	Parachute Bay	D	54" of Phenolic body tube in two pieces	Holds CRAM (Compact Removable Avionics Module), as well as main and drogue parachute
III	Air Braking Payload Bay	E	15" Phenolic coupler	Holds tabs used for changing apogee of rocket during flight
	Fin Can and	F	32.5" Phenolic tube	Holds motor and motor mount and

	Motor Mount		and Plywood fins	carbon fiber fins

3.1.4 Component Design Review

3.1.4.1 Nose Cone

The full scale launch vehicle will use a polypropylene nose cone purchased from Apogee Rockets. The team has used polypropylene nose cones in the past to great success. Polypropylene is strong enough to withstand any forces during landing or the rover deployment, and is both lightweight and inexpensive. Other materials that were considered were carbon fiber and fiberglass. Both are stronger than polypropylene, but given that the material properties of the nose cone are not critical to the launch vehicle design, the cheaper option was chosen and carbon fiber and fiberglass were neglected. The option of the team fabricating our own nose cone was considered, but given the increased chance of manufacturing error for little to no benefit, this option was quickly disregarded. Given that the nose cone is being purchased rather than fabricated by the team, the selection of a nose cone was also limited by the inventory off commercial vendors. Fiberglass nose cones were available for the specified dimensions, but were disregarded due to the reasons stated above.

The final option chosen for the launch vehicle and purchased from Apogee Rockets is the PNC-7.51". This nose cone has an outer diameter at the shoulder of 7.51 inches, which matches the inner diameter of the body tube section that houses the Rover payload. The overall length of the nose cone is 22 in, with a shoulder length of 5 in. The weight of the nose cone is 30.66 oz. This nose cone features an ogive shape, which is consistent with nose cones used in years previous. The main benefits of the ogive shape come with ease of construction.

Mounted in the nose cone will be two fiberglass bulkheads. These bulkheads will be used to eject the nose cone for deployment of the Rover payload. One will be placed roughly halfway up the length of the nose cone, while the other will be placed at the top of the shoulder. As discussed later in Section 4.1.2.1.3, the bulkhead at the top of the shoulder will integrate the Rover payload with the nose cone to ensure both systems remain in place during the launch and deploy properly after landing. These bulkheads will be attached using epoxy. Black powder charges will be used to eject the nose cone once the launch vehicle has landed. More information about the ejection of the nose cone can be found in Section 4.1.3.1.5.

Figures 7 and 8 below show isometric views and CAD drawings of the nose cone as purchased from Apogee Rockets.

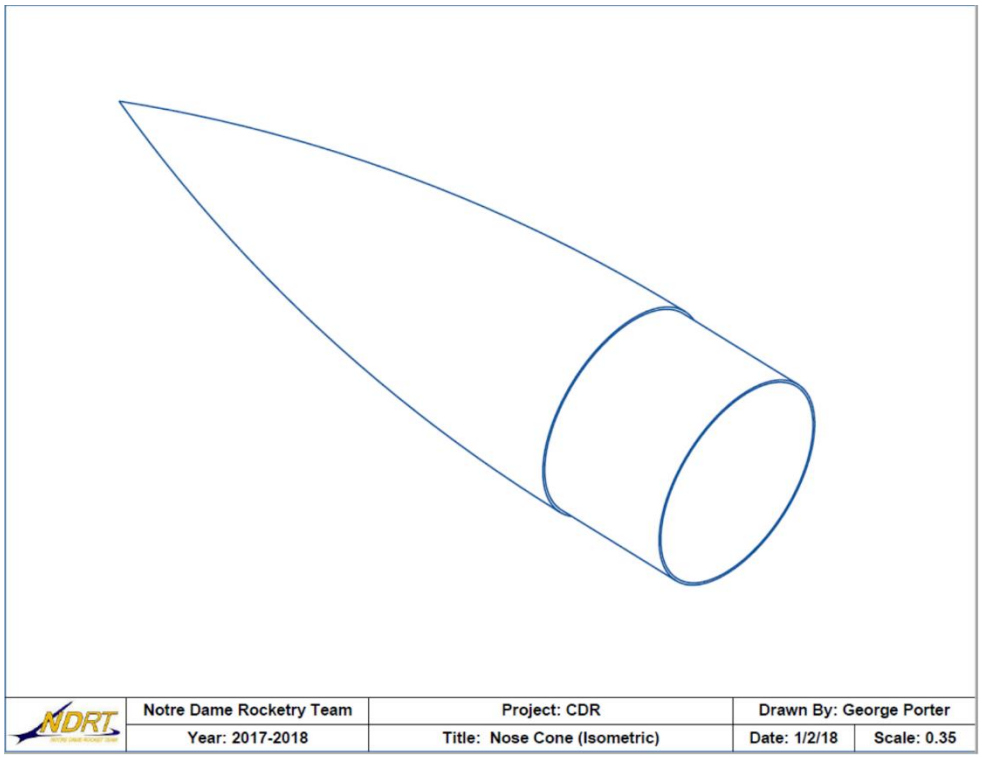


Figure 7. Isometric view of nose cone as purchased from Apogee Rockets. Units in inches.

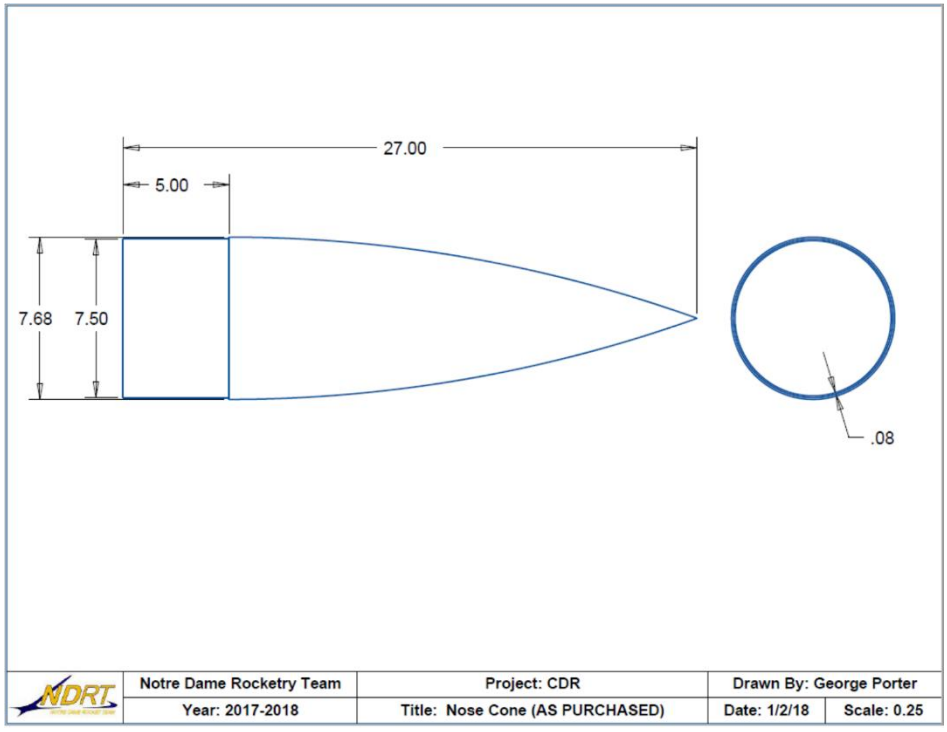


Figure 8. Dimensioned drawings of side and rear view of nose cone as purchased from Apogee Rockets. Units in inches.

The dimensions of the nose cone are shown below in Table 8.

Table 8. Dimensions of Nose Cone.

Property	Dimension
Length (in)	22
Shoulder Length (in)	5
Weight (oz)	30.66
Outer Diameter (in)	7.675
Inner Diameter (in)	7.51

3.1.4.2 Airframe

The airframe of the launch vehicle will consist of both kraft phenolic and fiberglass body tubes and couplers. Phenolic tubing was used in the previous year and was shown to be a versatile material. It provides structural support compared to materials while still keeping costs low. Phenolic body tubes were purchased commercially through Apogee Rockets.
Section I of the launch vehicle is constructed out of fiberglass tubing to allow transmission of signals to the rover, as well as added strength to the Rover. This section consists of the Rover Payload Bay with an outer diameter of 7.74 in. to allow more space to fully develop the rover, and the Transition Section to taper the main body down to 5.54 in. to conserve weight.

This transition section consists of a 4 in. tapered section for a smooth reduction of 2.2 in. of body tube diameter, as well as two 6 in. shoulder couplers. This component was made of fiberglass, and purchased commercially. This reduced the risk of improperly manufacturing the part in house and further ensures that no signals are inhibited from reaching the Rover Payload. A picture of this section is shown below in Figure 9.

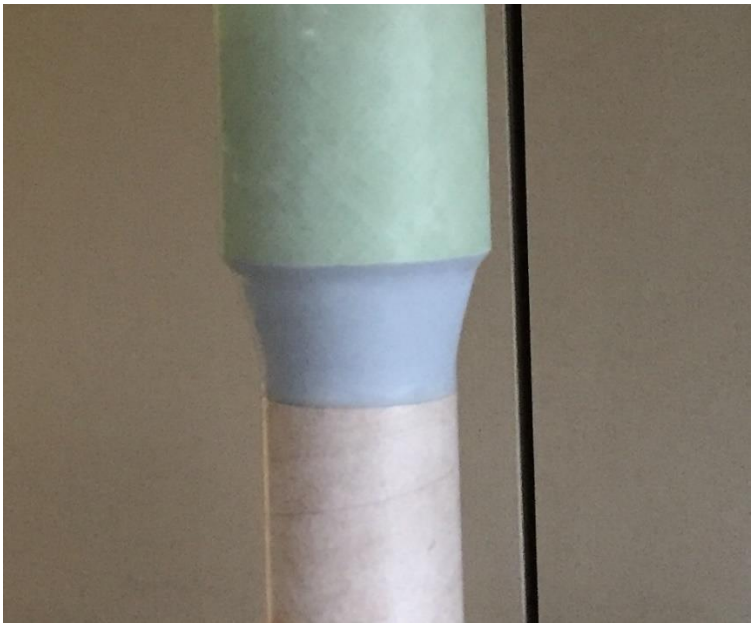


Figure 9. Transition Section.

In order for the Air Braking System to function properly and be able to reduce apogee by 250 ft, the transition in the airframe diameter cannot induce major turbulence or flow separation at the location of the air braking tabs. Preliminary analysis of the flow field was conducted during PDR using ANSYS Fluent for the maximum simulated velocity of the rocket (200 m/s) using a body tube transition from 8.5 in. down to 5.5 in. It was determined that this large transition would not cause significant boundary layer growth or flow separation at the location of the air braking tabs. For this reason, further and more developed CFD analysis of the flow field was deemed unnecessary. The final design of the Deployable Rover payload only needs a 7.675 in diameter body tube and therefore would reduce the boundary layer growth seen in the preliminary analysis. It was determined that developing a more in depth model for a more comprehensive flow field analysis would be an inefficient use of resources and that the work done for PDR was sufficient to validate the use of the transition.

The overall integrity of the airframe was ultimately verified through full scale testing. The materials chosen are historically reliable and will be further validated throughout construction of the rocket that will be flown at competition. The design presented in this report has been simulated using OpenRocket and RockSim software for a variety of flight conditions. The simulations tested the effects of mass distribution on the center of gravity, calculated the center or pressure, and predicted an apogee for the launch vehicle. Additionally, they allowed for different airframe finishes and ballast locations.

3.1.4.3 Fins

The shape of the fins was chosen to be a parallelogram for multiple reasons: this shape produces favorable stability at high Reynolds numbers and this shape is easy to construct and replicate multiple times. Additionally, with this shape, the velocity that would cause flutter in the fins exceeds the maximum velocity of the rocket. This is confirmed by the fact that the same fin design was used last year, and the fins did not experience flutter during the Full Scale Launch. A MATLAB program built on the following equations also proves that the flutter speed and the divergence speed won't be reached during the flight.

$$\frac{1}{2}\rho U^2|_{\text{flutter}} = \frac{[-E \pm (E^2 - 4DF)^{1/2}]}{2D}, \qquad (4\text{-}66)$$

$$U_{\text{diverge}} = \left[\frac{k_\theta}{\pi \rho c a}\right]^{1/2}. \qquad (4\text{-}53)$$

The effectiveness of the fins at high Reynolds numbers was important because of the speed the rocket achieves during flight. The parallelogram shape does a better job of reducing drag while maintaining stability, which was more advantageous for the given launch conditions and criteria than, for example, an elliptical shape, which is more effective at maintaining stability for lower Reynolds numbers. In order to maintain flight in the vertical direction, fins were chosen that maximize stability and minimize drag and flutter, thereby also maximizing apogee. Furthermore, with this fin configuration, the team can quickly adjust the apogee of the rocket, if necessary, by increasing or decreasing the height of the fins. The ease of construction was important for our originally intended material, carbon fiber. The ease of construction was also an important factor, however, because we could not attain our desired material for the fins at the last minute, and when the team decided to use plywood as a replacement material for the fin it was quick and easy to construct. It is also easy to shape the edges of the fin into an airfoil when it is a parallelogram shape, which save us time during last minute construction. Carbon fiber was originally chosen for its high strength to weight ratio, but plywood was chosen as a replacement because it was used successfully in past team rockets. Additionally, plywood could be attained quickly and cheaply, and the team has previous knowledge in laser cutting plywood. The only fin dimension changed for the switch in material was the thickness, which was changed from an eighth of an inch to a quarter of an inch to account for the loss of strength in plywood compared to carbon fiber.

The fins were designed in standard parallelogram shape with a height that would extend to the motor mount at the sweep angle. For the sake of easy integration into the fin can, it was the bottom end of the fin would be extruded to attach to the fin can at a right angle. The height, sweep angle, and shape from the outside of the fin can was retained. Figure 10 shows the finalized design of the fins, including the integration feature at the bottom of the fin. Figure 11 shows a picture of the actual fins pre-sanding.

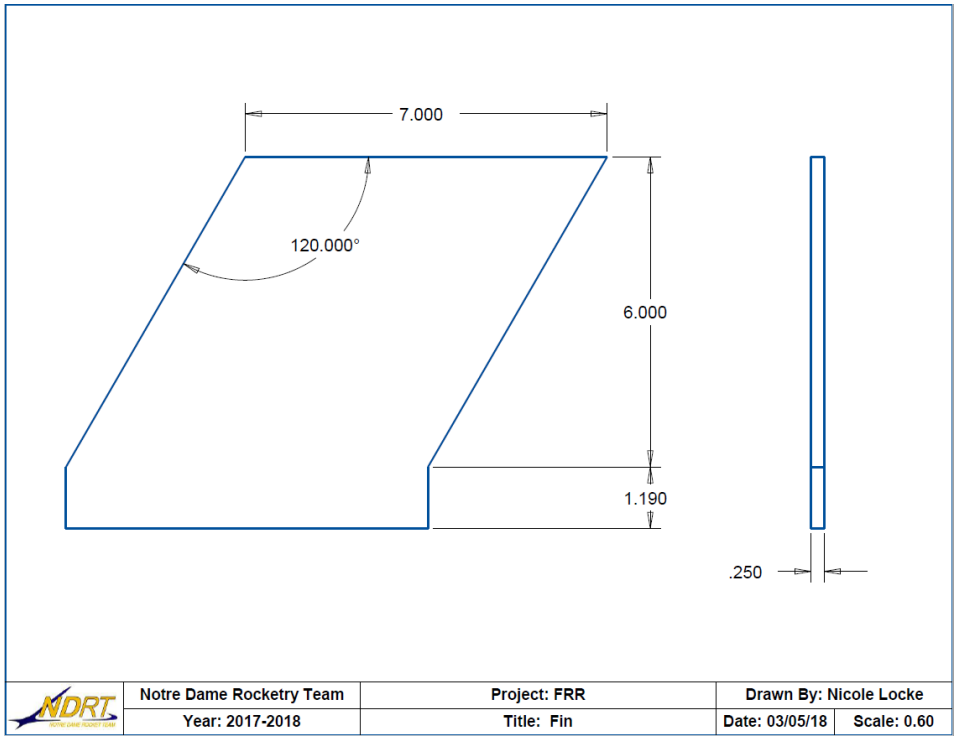


Figure 10. Final Fin Design. Units in inches.

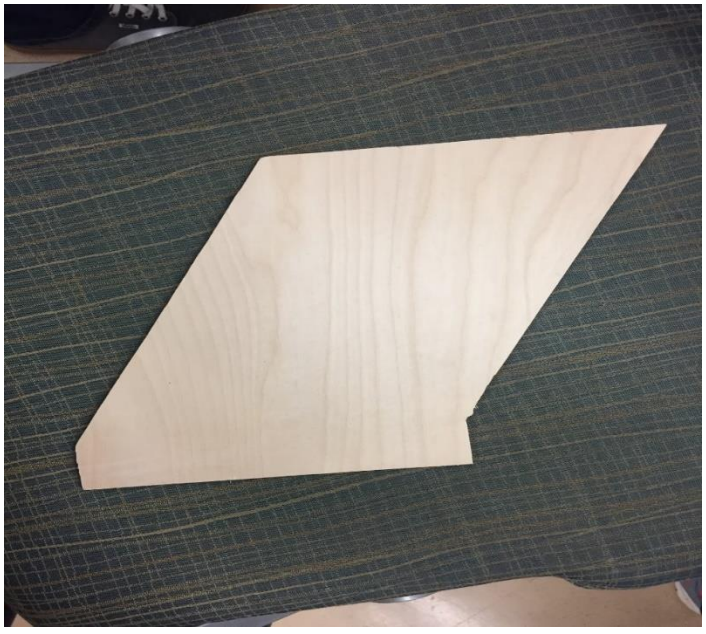


Figure 11. Picture of Pre-Sanded Fin.

Table 9 gives the finalized dimensions of the fins, which were measured after construction of the fins was complete. These were designed to provide ample stability margins as calculated by OpenRocket and RockSim. It was also insured that the fins could withstand the stress put upon them by Finite Element Analysis.

Table 9. Finalized Fin Dimensions.

Detail	Dimension
Number of Fins	4
Tip Chord [in]	7.0
Root Chord [in]	7.0
Height above Fin Can [in]	6.0
Sweep Angle [°]	30
Attachment Height [in]	1.19

The fins were built from plywood ordered from a local hardware store. All four were cut from the same quarter-inch plate. Each shape was cut out using a CNC Techno Router found in Stinson Remick Hall on the University of Notre Dame's Campus. After each fin was cut, the leading and trailing edges were sanded into an airfoil shape. The trailing edges were sanded using the mold on the left and the leading edges were made using the mold on the right, which can be seen in Figure 12. All safety protocols were followed during the construction and sanding of the fins.

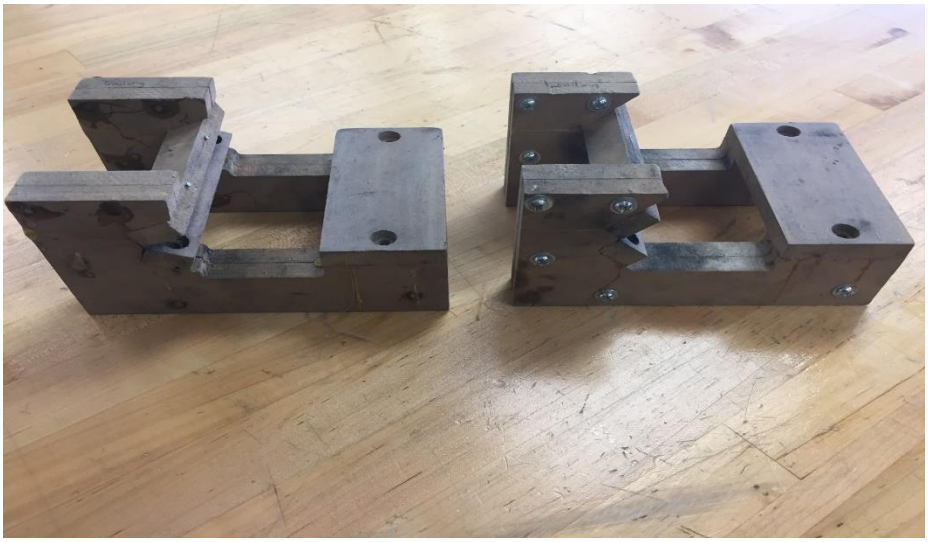


Figure 12. Fin molds.

For the leading edge, the angle of the sanding mold is ninety degrees, and for the trailing edge, the angle of the sanding mold is forty-five degrees. This changed the fins from a flat plate cross section to a symmetric airfoil cross section, which helped reduce drag on the fins while maintaining equal pressure from the flow on both sides of the fin. The equal pressure ensures stability during flight. Documentation of the process is shown in Figure 13.

Figure 13. Sanding process.

3.1.4.4 Couplers

Couplers were selected in order to allow the rocket to easily be separated into various payloads both for construction purposes and during parachute deployment. While other options such as access doors were considered, couplers were decided upon because of their convenience and lightweight, simple design.

Two of the couplers used were constructed out of 0.08 inch thick Kraft Phenolic tubing. The additional coupler and the switch to phenolic from CDR were due to material availability. Due to its light weight, strength, and its ability to easily interface with the phenolic of the rest of the rocket, phenolic couplers will be able to easily sustain the stresses of flight. The outer diameter of these coupler was sanded down to be as close as possible to the inner diameter of the tubes they are connecting allowing for a tight fit to hold the rocket together, as well as provide extra stability for the body tubes during flight. There are additional shoulders built into the transition section which were constructed out of fiberglass with the transition section in order to minimize error in construction.

The first phenolic coupler connects the primary recovery tube, which contains the CRAM and both parachutes, to the secondary recovery tube. The second coupler is attached to the Air Braking System and connects the primary recovery tube to the fun can. The recovery coupler is

11 in in length, and the Air Braking System coupler is 19 in in length. These couplers can be seen in Figures 14 through 17.

The two couplers built into the transition section of the rocket connect the rover tube with the secondary recovery tube. The couplers are 6 in in length and 0.08 in thick. The outer diameters were designed to be as close as possible to the inner diameter of the Rover Payload Bay and the secondary recovery tube, respectively. The outer diameters of the fore and aft shoulders are 7.675 in and 5.38in respectively. The transition section and shoulders are shown in Figures 18 and 19.

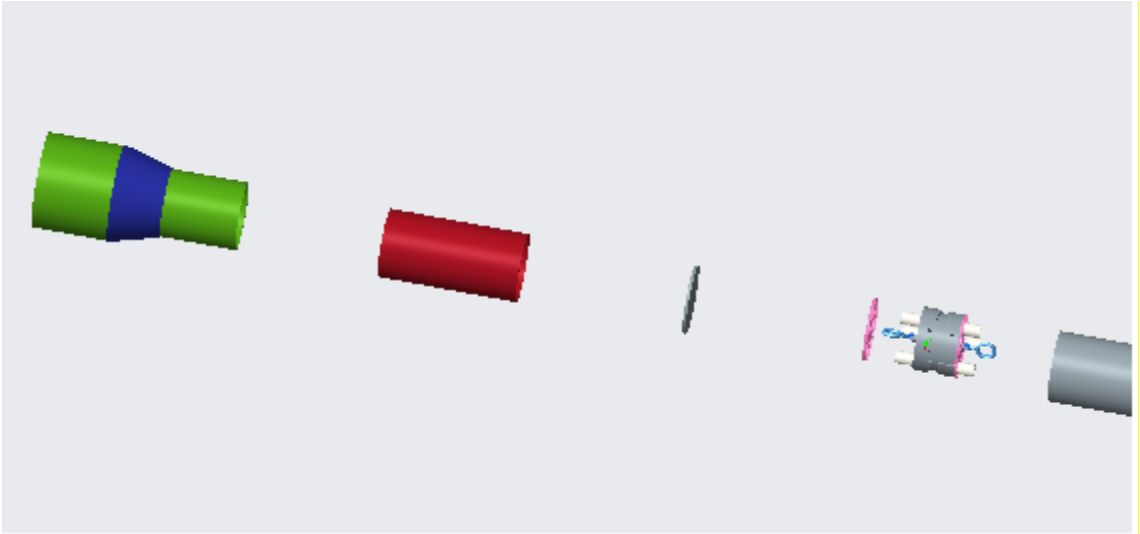


Figure 14. Recovery Coupler Shown in Red.

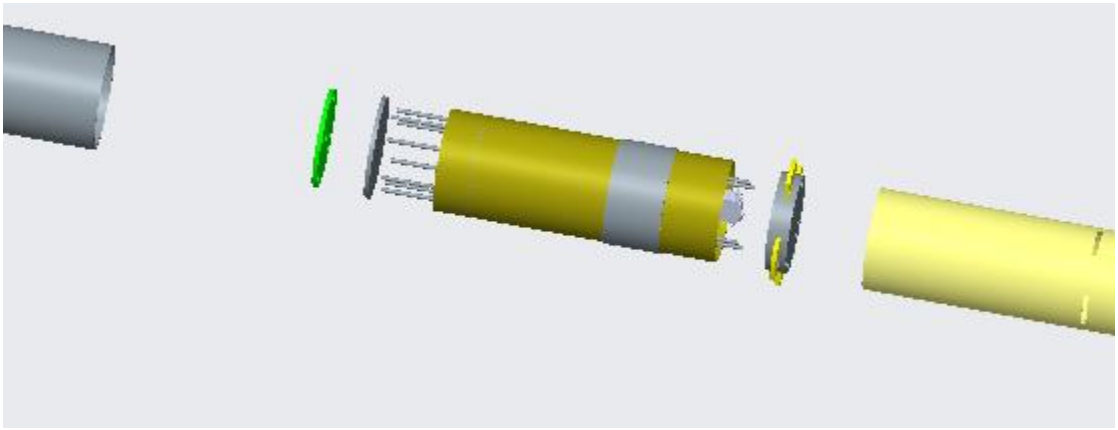


Figure 15. ABP Coupler Shown in Gold.

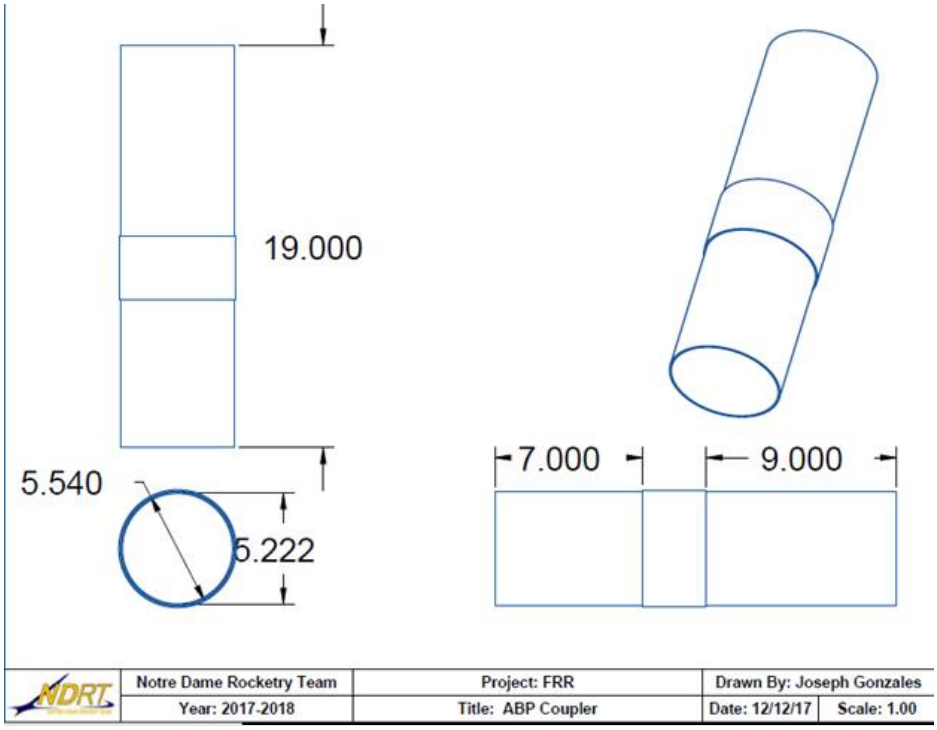


Figure 16. ABP Coupler Dimensions. Units shown in inches..

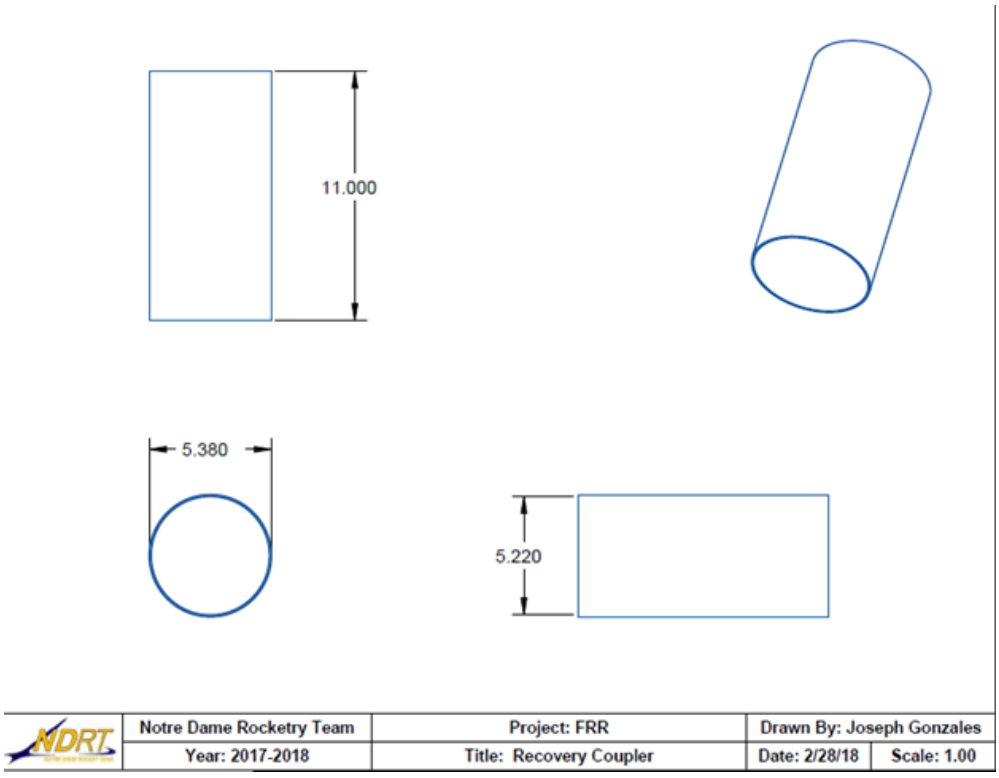


Figure 17. Recovery Coupler Dimensions. Units shown in inches.

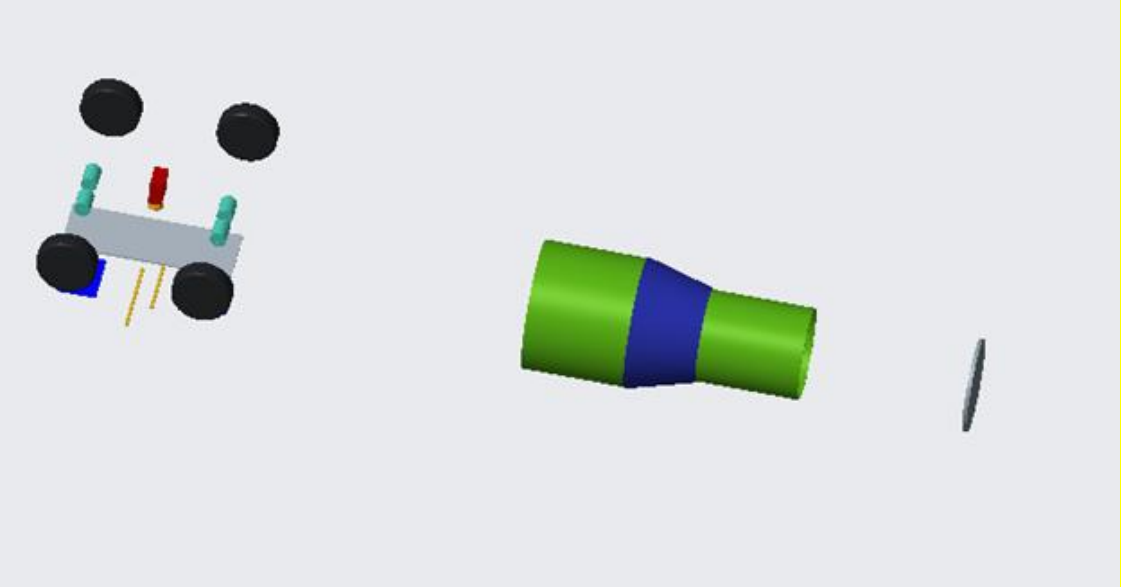


Figure 18. The Transition Section with Shoulders Shown in green.

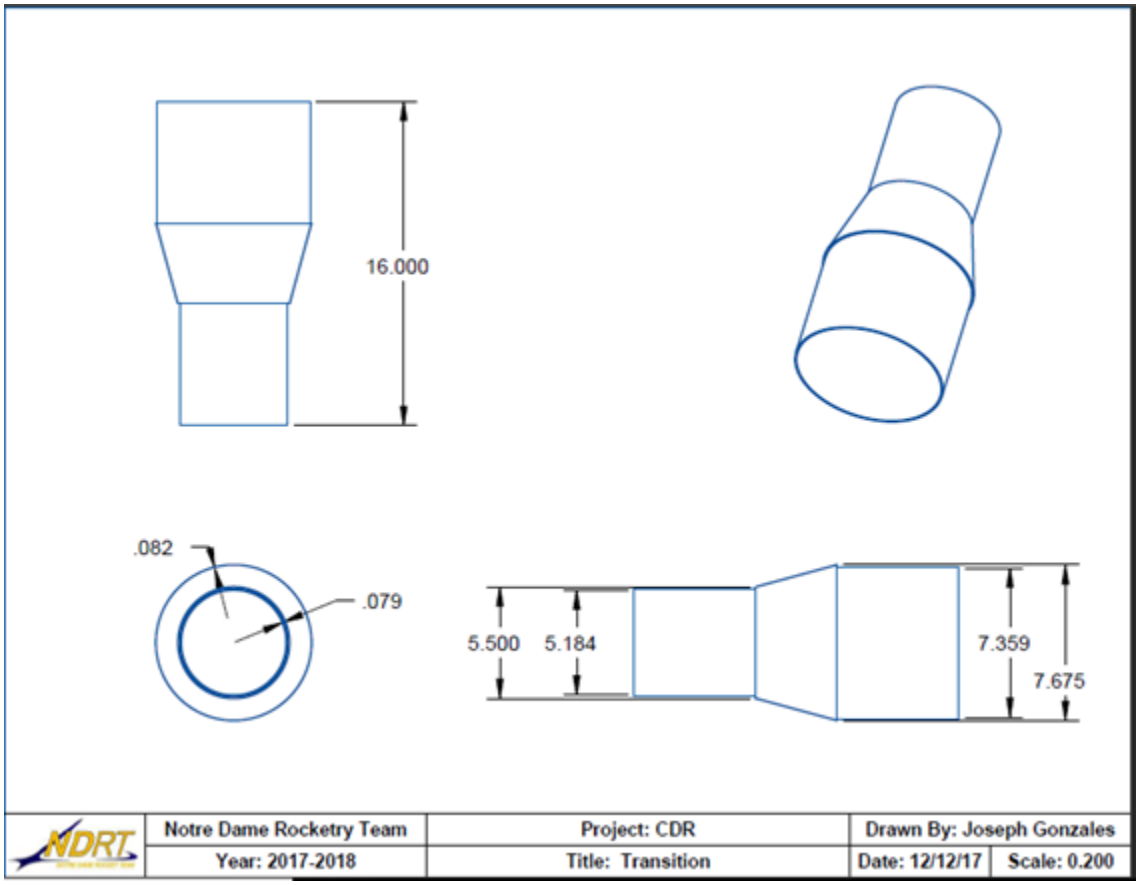


Figure 19. Dimensions of the Transition Section and Shoulders (Shown in Green). Units in inches.

3.1.4.5 Bulkheads

Bulkheads play an important role in the structure of the rocket. They allow the integration of eye bolts and other attachment hardware in addition to providing support to their respective placements on the airframe and holding components of the rocket in place. All of the vehicle's bulkheads are all made out of either plywood or acrylic. While the vehicle's original design called for some fiberglass and some plywood bulkheads, fiberglass supplier issues led the team to an all-plywood and acrylic design. Each material will be suitable for each bulkhead's intended purpose.

The plywood bulkheads were made by laser cutting identical circles from a sheet of plywood and acrylic and sanding the outer surface until each bulkhead fit snugly in its intended location. The vehicle has 7 total bulkheads: CRAM (2 acrylic), Recovery connection to front and rear of the rocket (2 double layered plywood), motor retention (1 plywood), and two in the nosecone/rover bay (2 plywood).

The maximum stress in the bulkhead can be found using the equation:

$$\sigma_m = \frac{P}{t^2}\left(0.631 \ln\left(\frac{\gamma}{t}\right)\right) + 0.676),$$

Where σ_m is the maximum stress, P is the load applied in the center, and t is the thickness

The maximum load can be calculated using Parachute shock equations from a NASA report (LINK):

$$F_{max} = C_k C_d A Q,$$

Where C_k is the parachute opening shock factor, C_d is the parachute drag coefficient, A is reference area, and Q is dynamic pressure.

$$C_k = \frac{\rho (C_d S)^{\frac{1}{5}}}{M},$$

Where C_k is the parachute opening factor, is density, $C_d S$ is parachute drag area, and M is system mass.

For the plywood used in the bulkheads, σ_y (yield stress) is 13.8MPa. Based on parachute shock calculations, the maximum load to be experienced by any bulkhead is 2kN. Therefore, σ_m= 117kPa for plywood which is below the threshold of 13.8MPa, making the bulkheads well suited for the maximum possible loading and therefore any loading below that value. For the acrylic σ_y = 64MPA. The maximum load results in a stress in the acrylic bulkheads of 117kPa, well below σ_y.

Figure 20. A plywood bulkhead used for recovery integration.

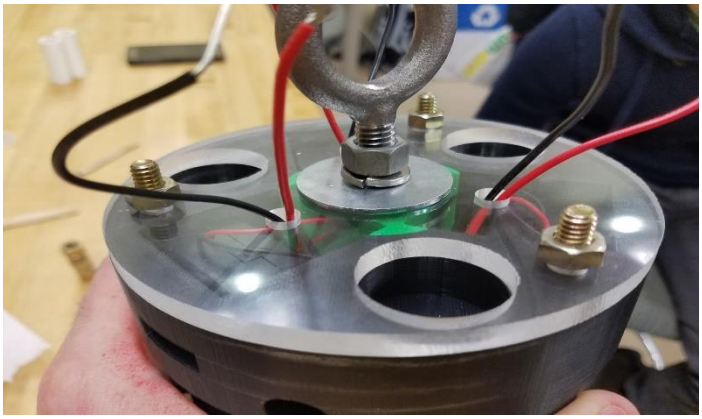


Figure 21. An acrylic bulkhead attached to the CRAM.

3.1.4.6 Motor Mount

The motor is located at the aft end of the launch vehicle, inside the fin can. The motor is mounted inside of the launch vehicle using a system consisting of three plywood centering rings, one plywood bulkhead, two phenolic body tubes of different diameters, and a motor retention system. One of the body tubes has an inner diameter equal to the outer diameter of the motor casing and is called the motor mount tube. The other body tube (the fin can) has diameter dimensions equivalent to the main body tube. A bulkhead with a diameter equal to that of the fin can inner diameter was placed at the most forward edge of the motor mount tube to prevent the motor from moving axially forward during burnout. The three centering rings have inner diameters equal to the outer diameter of the motor mount tube and outer diameters equal to the inner diameter of the fin can. These dimensions are summarized in Table 10 below. These were positioned along the motor mount tube to rings ensure that the motor casing and the internal motor remain centered throughout all stages of flight. Their positioning was selected to avoid interference with the fins. This system prevents any thrust gimbaling that would lead to catastrophic failure of the launch vehicle. For the entirety of the motor mounting system, components were integrated using JB Weld, a heat resistant epoxy that is capable of handling the high temperatures the components experience during motor burn. The bulkhead and centering

rings were all cut using a CNC machine to ensure proper dimensions. A more detailed explanation of the construction process is described in Section 3.1.7. A schematic of this system is shown below in Figure 22.

Table 10. Motor Mount Dimensions.

Component	Outer Diameter (in)	Inner Diameter (in)	Length (in)	Distance from aft end of fin can (in)
Motor Mount Tube	3.162	3.012	26.5	0
Fin Can	5.54	5.38	32.5	n/a
Bulkhead	5.38	n/s	0.25	26.5
Centering Rings	5.38	3.162	0.25	0.75, 13.5, 20

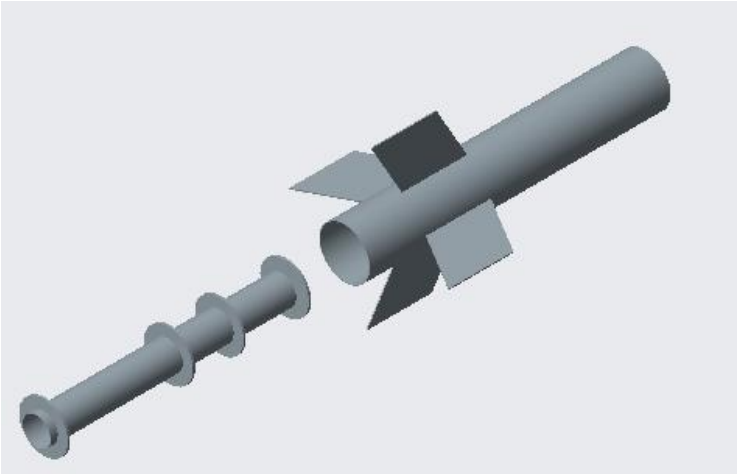


Figure 22. Schematic of Motor Mount System.

The motor is retained using a 75 mm Aero Pack Incorporated Quick-Change motor retention system. This system consists of two pieces manufactured out of precision machined 6061-T6 aluminum. A threaded cylinder adaptor was attached to the outside of the motor mount tube using JB Weld. Once the motor is inserted into the motor mount tube, a retainer cap is screwed onto the adaptor to hold the motor and its casing in place. This system is compatible with Cesaroni Motors and allows for the motor to be removed without any screws, making it a simple and robust choice. A photo of the preinstalled Quick-Change system is shown in Figure 23 below.

Figure 23. Pre-installed API Quick-Change 75 mm motor retention system.

3.1.5 Subsystem Design Review

3.1.5.1 Propulsion

The propulsion system consists of the motor and its corresponding support systems, including a retention system and a centering/mounting system. The final launch vehicle will be powered by a Cesaroni L1395-BS, which has been procured from AMW Pro-X. The specifications and the commercially published thrust curve for the L1395-BS are shown below in Table 11 and Figure 24, respectively.

Table 11. Cesaroni L1395-BS Motor Characteristics.

Property	Value
Diameter (in/mm)	2.95 (75 mm)
Length (in)	24.45
Peak Thrust (lbf)	400.5
Average Thrust (lbf)	314.0
Total Impulse (lbf*s)	1101.5
Total Weight (oz)	151.31
Propellant Weight (oz)	82.77
Burn Time (s)	3.51

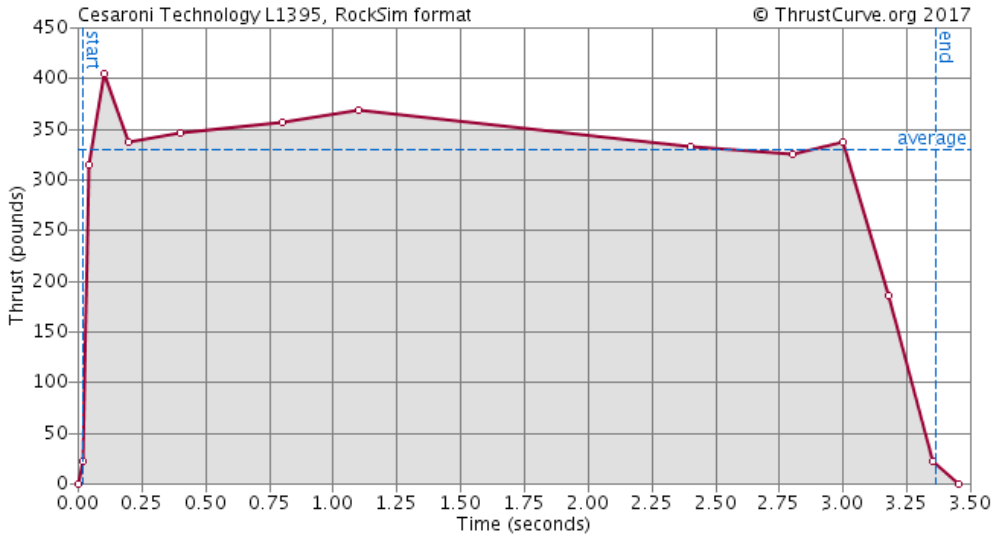


Figure 24. Thrust Curve for Cesaroni L1395-BS.

This motor uses a Cesaroni 75mm aluminum casing of length 23.95in, which can be seen being inserted into the rocket below in Figure 25. The propellant is packed into 4 grains to ensure a smooth and nearly constant burn throughout the 3.5s as can be noted in Figure 24 above. The grains are kept secured with retaining rings, while the graphite nozzle was retained with a stainless steel washer. A detailed description of the motor support and retention system can be found in Section 3.1.4.6 and Section 3.1.6.3.1.

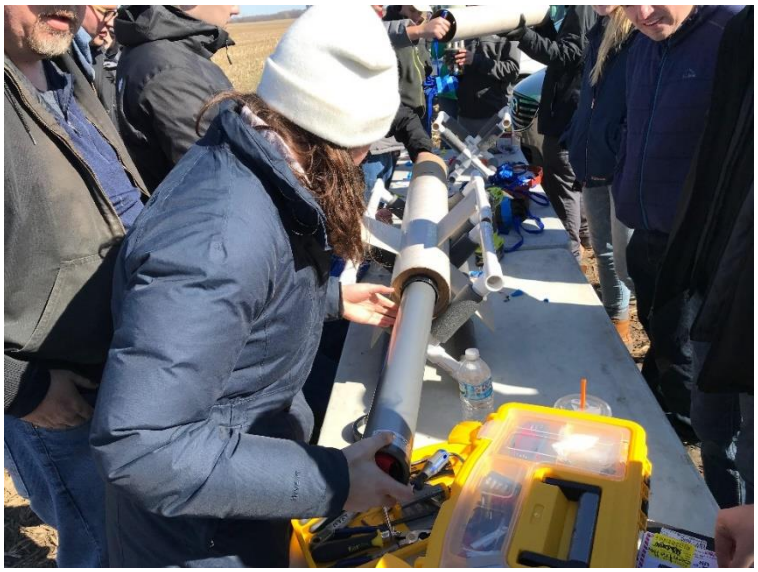


Figure 25. Cesaroni 75 mm Motor Casing being inserted into rocket.

As no members of the team are qualified to handle an L-class motor, the team mentors assembled the motor as team members watched. All components were first cleaned and the casing was inspected. At this point, the O-rings and retaining snap rings were greased. The bulkhead and thumb screw were assembled first. The O-rings were inserted, followed by the 4 grains. This was followed by the nozzle. At this point, the stainless washer nozzle was inserted, followed by a retaining ring. The assembled bulkhead was installed into the top of the motor. The second retaining ring was used here. The motor was inspected for any flaws, and when that was complete, was eventually inserted into the rocket. The motor was kept clean from the time it was fully assembled to when it was inserted into the launch vehicle. On the launch pad, an electrical igniter was inserted. It was taped to a thin wooden dowel and slid through the nozzle until it reach the bulkhead and was secured with tape. This final action is shown below in Figure 26.

Figure 26. Michiana Rocketry Mentor inserting electrical igniter into motor on launch pad.

The main goal of the propulsion system is to safely bring the launch vehicle to 5,380±100 ft so that the Air Braking System may add the necessary amount of drag to bring the rocket to exact 5,280 ft goal. The current launch vehicle with the motor totals to 764 oz. With this design, OpenRocket gives an expected apogee without drag tabs of 5405 ft when in 10 mph winds at standard atmospheric conditions. For robustness, this simulation was repeated with varying wind speeds. The results can be seen below in Table 12. In order to keep these values within the range set for the Air Braking System to be effective (5,280-5,480 ft) with the new materials ballast was added.

Table 12. OpenRocket Simulation Apogee Results.

Wind Speed	Predicted Apogee [ft]
0 mph	5415
5 mph	5405
10 mph	5367
15 mph	5325
20 mph	5271

Due to motor availability, the Cesaroni L1115 was used for the team's full scale test launch. This motor was selected because it has similar specifications to the L1395 in size, mass, and performance. A comparison these specifications for the two motors is shown below in Table 13.

Table 13. Comparison of Cesaroni L1395 and L1115 Specifications.

Parameter	Cesaroni L1395 Value	Cesaroni L1115 Value
Diameter (in)	2.95	2.95
Length (in)	24.45	24.45
Peak Thrust (lbf)	400.5	385.2
Average Thrust (lbf)	314.0	251.6
Total Impulse (lbf*s)	1101.5	1127.4
Total Weight (oz)	151.3	155.3
Burn Time (s)	3.5	4.5

OpenRocket simulations with this new motor predicted an apogee of 5754. When the vehicle was launched without the assistance of the Air Braking System, it reached an apogee of 5765 ft. A more detailed analysis of these flight results will be shown in Section 3.3.2.2. However, this initial apogee gives the team confidence that the L1395 will also be capable of bringing the rocket above the 5280 ft desired apogee. Additionally, the team is also confident in the accuracy of the flight simulations. If the team is unable to fly again with an active Air Braking System, a new motor will be considered that will lead to a lower apogee.

3.1.5.2 Ballast

Due to several material changes for this year's rocket, specifically the weight change due to using phenolic instead of carbon fiber, a small section of ballast was added to the rocket. The total weight of the ballast is 25 oz, thus fulfilling the requirement that the ballast weight must be less than 10% of the weight of the fully constructed, unloaded rocket. The ballast is secured between two plywood bulkheads attached to the air braking system via four steel rods running the full length of the coupler. A picture of this setup can be seen in Figure 27. This placement in the fin can directly aft of the air braking system of the rocket was chosen in order to bring the center of gravity closer to the center of pressure (at the air braking tabs). Placing ballast near the nose cone would have further increased stability. Overall, the placement of the small section of ballast will simultaneously add needed weight to a rocket of lighter material in order for the rocket to reach an apogee of 5280 ft and to ensure that the stability of the rocket is not compromised.

Figure 27. Ballast between couplers of the Air Braking System.

3.1.6 Flight Reliability and Confidence

3.1.6.1 Materials

3.1.6.1.1 Materials Summary

Launch success and mission performance is heavily reliant on the materials used to create the launch vehicle. These materials are also balanced based on weight, sturdiness, and cost. The

main materials for this year's vehicles have been used in the past, notably two years ago when the team had a successful launch with them in Huntsville. While last year the team used a carbon fiber body tube, this year Kraft phenolic body tubes and fin can will be used due to a manufacturer error.

The nose cone is made of polypropylene, a rugged, sturdy plastic. Polypropylene plastic has a high compressive strength, which makes it ideal for its position on the rocket. Being at the very tip breaking the flow on the flight up, the nose cone must be able to withstand compressive forces and still maintain its structural integrity. The material is also easily shaped to fit the needs of the rocket. Having a structurally defined nose cone is an important aspect for the aerodynamic stability of the vehicle design.

This nose cone fits into the Rover bay, which is made of fiberglass for added robustness while also allowing for wireless communications to the device. This fiberglass section makes up the larger 7.74" diameter section, which leads into the transition section. The fiberglass will also give the launch vehicle enough strength to support the added forces stemming from a diameter change.

The transition section is made of molded fiberglass and was made by Dave Powell from Summit Aviation. It has a 4 inch variable diameter tube that attaches to two 6 inch couplers. These couplers extend and fit into the forward fiberglass body tube and the aft phenolic body tube.

The bottom of the transition section connects to the phenolic body tube and fin can, which is strong, but light. The hardened board has proven to be more than strong enough to support any aerodynamic forces acting on the body, as well as a hard landing. The phenolic is very versatile, as it can easily be cut into different lengths, and is relatively cheap for the vehicle.

The fins are constructed out of plywood, which is stiffer than the phenolic used elsewhere, and also able to be sanded down into an airfoil shape. The extra stiffness of the fins is necessary because of the aerodynamics forces acting on the outstretched fins. This material has also been used in the past for fins, and has proven to be strong enough to support a successful flight.

The couplers are all made of phenolic due to the aforementioned low cost, and the fact that they will not support any load. The bulkheads were made of plywood. Due to failed flights in the past, any load bearing bulkheads have since been made with fiberglass or plywood for added strength and support. The centering rings will also be made with this plywood because of the importance they play in maintaining a safe flight.

All of the phenolic was ordered from Apogee rockets, so there is confidence in the manufacturing tolerance. The plywood was cut with a CNC, which allowed us to customize the pieces as well as be confident in their quality.

Depending on the need and area of the rocket, differing epoxies were used - a mixture of JB Weld and Rocketpoxy. While both have similar adhesive properties, JB Weld has a higher temperature tolerance and therefore was used around the motor due to the high heat it will

experience during takeoff. For the other sections around the vehicle - bulkheads, fins, etc. Rocketpoxy was used.

A summary of the materials used in the rocket can be found below in Table 14, and the material properties can be found in Table 15.

Table 14. Final materials used.

Part	Material
Nosecone	Polypropylene Plastic
Body Tube	Kraft Phenolic
Motor Mount	Kraft Phenolic
Fins	Plywood
Bulkheads	G10 Fiberglass / Plywood
Couplers	Kraft Phenolic
Centering Rings	G10 Fiberglass

Table 15. Material Properties for Phenolic and Plywood.

Material	Component Use	Density (lb/in^3)	Tensile Strength (ksi)	Tensile Modulus (msi)	Shear Modulus (msi)	Com-pressive Strength (ksi)	Com-pressive Modulus (msi)	Specific Weight (lb/in^3)
Phenolic Paper	Body Tubes, Couplers	--	12-15	--	--	32	--	0.049
Birch Plywood	Fins, Bulkheads, Centering Rings	0.024	13	--	2.2	10	--	--

3.1.6.1.2 Airframe Compression due to Motor Forces

The motor used for the competition is the Cesaroni L1395. The specifications for this motor can be found in Table 16 below. The motor will reside in the fin can, with the motor

mount being fixed in the fin can by three centering rings. These centering rings will allow the load from the motor to be transferred to the body of the rocket.

Table 16. Summary of Motor Specifications for Airframe Analysis.

Property	Dimension
Propellant Weight (lb)	5.2
Total Weight (lb)	9.5
Peak Thrust (lbf)	400.1
Average Thrust (lbf)	313.8
Impulse (lbf-sec)	1100.5
Thrust Duration (sec)	3.5s
Motor Diameter (in)	2.95

The flight reliability analysis was performed according to principles of Deformable Body Mechanics. The maximum force experienced from the motor was taken to be the peak thrust of the motor as shown in Table 16 above.

Using the dimensions of the fin can, the cross-sectional area was calculated using the inner and outer area of the fin can, and subtracting the values.

$$\mathrm{A_{cross}} = \mathrm{A_{outer}} - \mathrm{A_{inner}}$$

Then the compressive pressure imparted to the body by the force of the motor was calculated by using the maximum force divided by the cross-sectional area.

$$\sigma_{motor} = \frac{F_{motor}}{A_{cross}}$$

This compressive pressure calculated was then compared to the compressive strength of the body tube. This strength was found from the material properties provided by the manufacturer of the phenolic paper, which was used as the material choice for the fin can.

The factor of safety can be calculated from the compressive pressure that the fin can experiences and the compressive strength of the material.

$$FS = \frac{\sigma_{material}}{\sigma_{motor}}$$

The analysis can be followed in Table 17 below.

Table 17. Results from Analysis of Maximum Force by Cesaroni L1395 on Body Tube.

Property	Dimension
Force Motor (lbf)	400.1
Inner Area (in^2)	24.2
Outer Area (in^2)	22.9
Cross-sectional Area (in^2)	1.3
Compressive Stress due to Motor (lbf/in^2)	307.8
Phenolic Paper Compressive Strength ($kips/in^2$)	32
Factor of Safety	103.96

According to the table above, the fin can will perform will under the forces experienced by the fin can. The factor of safety is 103.96, which is more than sufficient for confidence in the airframe's structural integrity during the launch.

Due to complications in obtaining proper materials in time for the test launch, a similar motor, the Cesaroni L1115 will be used for the test flight. Using the equations above and the peak thrust for this motor, performance predictions can also be made for the test launch. The analysis for the rocket performance with this alternate motor can be followed in Table 18.

Table 18. Results from Analysis of Maximum Force by Cesaroni L1115 on Body Tube.

Property	Dimension
Force Motor (lbf)	385.2
Inner Area (in^2)	24.2
Outer Area (in^2)	22.9
Cross-sectional Area (in^2)	1.3
Compressive Stress due to Motor (lbf/in^2)	296.3
Phenolic Paper Compressive Strength ($kips/in^2$)	32
Factor of Safety	108.00

The factor of safety is 108 as seen in the table above. This indicates that the body tube around the motor will do a sufficient job at absorbing the forces experienced from the motor. This is similar to the factor of safety when using the competition motor, the Cesaroni L1395. Due to the similarity of these motors, the team is confident that the results of the test launch will be indicative of the performance of the competition launch.

The analysis performed in this section was conservative, as it calculated the stresses caused by the motor on the main airframe body, and assumed that the centering rings and bulkhead used to integrate the motor into the fin can ideally transferred the maximum thrust to the body. For a more detailed look at the integration of the motor into the body, and a structural analysis of the integration system, see Section 3.1.7.3.1.

3.1.6.1.3 Airframe Compression due to Atmospheric Forces

During the course of flight, the airframe will be exposed to high pressures against its body due to drag forces. This pressure will be experienced by the entire body of the rocket, which has three main geometries due its variable diameter. The drag experienced by the rocket during its flight, is described in Table 19.

Table 19.Summary of drag experienced during rocket ascent.

Property	Dimension
Peak Drag (lbf)	47.97
Maximum Velocity (ft/s)	637.2

The flight reliability analysis was performed according to principles of Deformable Body Mechanics. Drag was assumed to exert a constant downward axial load perpendicular to the cross section of the rocket. The maximum force exerted on the body was taken to be that of the peak drag due to the atmosphere, as shown in Table 19 above.

Using the dimensions of the fin can, the cross-sectional area was calculated using the inner and outer area of the rocket geometry being considered, and subtracting the values.

$$\mathrm{A_{cross}} = \mathrm{A_{outer}} - \mathrm{A_{inner}}$$

Then the compressive pressure imparted to the body by the force of drag was calculated by using the maximum drag force divided by the cross-sectional area.

$$\sigma_{drag} = \frac{F_{drag}}{A_{cross}}$$

This compressive pressure calculated was then compared to the compressive strength of the three sections of the airframe. This strength was found from the material properties provided by the manufacturers. The three geometries of the rocket's body that are considered are the fiberglass rover payload tube, the fiberglass transition section which has a diameter that varies as a function of length, and the phenolic paper main body tube.

The factor of safety can be calculated from the compressive pressure that the fin can experiences and the provided compressive strength of the material.

$$FS = \frac{\sigma_{material}}{\sigma_{drag}}$$

The analysis of the fiber glass rover payload bay can be followed in Table 20 below.

Table 20. Results from analysis of atmospheric compressive force experienced by rover tube during flight driven by Cesaroni L1395.

Property	Dimension
Atmospheric Drag (lbf)	47.97
Inner Area (in^2)	44.46
Outer Area (in^2)	47.05
Cross-Sectional Area (in^2)	2.59
Compressive Stress by Atmosphere (lbf/in^2)	18.53
Fiberglass Compressive Strength (kip/in^2)	140
Factor of Safety	7.56×10^3

According to the table above, the rover tube will perform well even under the maximum drag forces it will experience during flight. The factor of safety is 7.56×10^3, which is more than sufficient for confidence in the structural integrity of the rocket during its flight.

In order to calculate the stresses felt by the transition section of the rocket, an expression of the transition section's cross-sectional area as a function of its length was developed such that

$$A(x) = \frac{(A_{min} - A_{max})}{L}(x) + A_{max}$$

Where L is the total length of the transition, A_{min} is the cross-sectional area of the smaller portion of the transition section, A_{max} is the cross-sectional area of the larger portion of the transition section, and x is the position from the fore of transition section. The analysis of the fiberglass transition section can be followed in Table 21 below.

Table 21. Results from analysis of atmospheric compressive force experienced by transition section during flight driven by Cesaroni L1395.

Property	Dimension
Atmospheric Drag (lbf)	47.97
Transition Section Length (in)	4.00
Maximum Inner Area (in^2)	43.148
Maximum Outer Area (in^2)	47.051
Maximum Cross-Sectional Area (in^2)	3.9033
Minimum Inner Area (in^2)	21.335
Minimum Outer Area (in^2)	24.105
Minimum Cross-Sectional Area (in^2)	2.7698
Maximum Compressive Stress by Atmosphere (lbf/in^2)	17.32
Minimum Compressive Stress by Atmosphere (lbf/in^2)	12.29
Average Compressive Stress by Atmosphere (lbf/in^2)	14.52
Fiberglass Compressive Strength (kip/in^2)	140
Maximum Factor of Safety	1.139×10^4
Minimum Factor of Safety	8.084×10^3

Average Factor of Safety	9.643×10^3

The minimum factor of safety is 8.084×10^3, and the average factor of safety is 9.643×10^3 as seen in the table above. This indicates that the transition section of the rocket will more than sufficiently withstand any expected compressive stresses due to atmospheric drag during flight.

The analysis of the phenolic paper main body tube can be followed in Table 22 below.

Table 22. Results from analysis of atmospheric compressive force experienced by rover tube during flight driven by Cesaroni L1395.

Property	Dimension
Atmospheric Drag (lbf)	47.97
Inner Area (in^2)	22.733
Outer Area (in^2)	24.105
Cross-Sectional Area (in^2)	1.3722
Compressive Stress by Atmosphere (lbf/in^2)	34.96
Phenolic paper Compressive Strength (kip/in^2)	32
Factor of Safety	915.4

According to the table above, the main body tube will perform well even under the maximum drag forces it will experience during flight. The factor of safety is 915.4, which is sufficient for confidence in the structural integrity of the main body tube during the flight.

The factor of each portion of the rocket indicates that the structure will remain sound even when subjected to maximum compressive drag forces. However, it should be noted that the stress calculations are conservative, as they assume that the load experience by each section is

ideal, and is equivalent to the maximum drag force as an axial compressive load directly downward onto the cross-section of the rocket.

3.1.6.1.4 Adhesive Stresses due to Vehicle Separation

During the recovery of the rocket, the separation of the rocket body, and subsequent force exerted on the body by the deployment of the parachute to decelerate the body will cause an immense amount of force. The rocket will split into three sections, each attached to the parachute by an eye-bolt placed through a bulkhead held to the section by adhesive. This will exert a fair amount of stress on these sections. The maximum forces felt by the rocket during recovery are described in Table 23.

Table 23. Summary of forced experienced during rocket recovery.

Property	Dimension
Peak Drag during Drogue Parachute Deployment (lbf)	42.242
Peak Drag during Main Parachute Deployment (lbf)	1.5632×10^3

The flight reliability analysis was performed according to principles of Deformable Body Mechanics. The force felt by the bulkhead was assumed to be perfectly distributed to it by the shear force exerted on it by the eye-bolt going through its center. The distribution of this force to the body attached to the bulkhead was modelled as shear forces between the bulkhead and the body, distributed over the surface area in contact with said body.

The maximum force distributed to the bulkhead was taken to be that of the peak drag experienced during main parachute deployment, as shown in Table 23 above.

Using the dimensions of the bulkhead, the shear surface was calculated using the diameter and thickness of the bulkhead such that

$$A_{shear} = d \cdot t \cdot \pi$$

The shear stress between the rocket body and bulkhead is described by dividing the maximum force felt during the main parachute deployment by the shear surface area

$$\tau_{separation} = \frac{F_{main}}{A_{shear}}$$

This shear stress was then compared to the tensile strength of the epoxy used to adhere the bulkhead to the rocket body. This tensile strength was provided in the material properties provided by the manufacturer.

The factor of safety can be calculated from the shear stress exerted on bulkhead, and the provided tensile strength of the material.

$$FS = \frac{\sigma_{tensile}}{\tau_{separation}}$$

The analysis of the adhesive stress rover payload bay can be followed in Table 24.

Table 24. Results from analysis of adhesive stress experienced by bulkhead during recovery separation following ascent driven by Cesaroni L1395.

Property	Dimension
Parachute Force (lbf)	1.5632×10^3
Bulkhead Diameter (in)	5.380
Bulkhead Thickness (in)	0.500
Bulkhead Shear Surface Area (in^2)	8.451
Shear Stress due to Main Parachute (lbf/in^2)	184.98
Epoxy Tensile Strength (lbf/in^2)	7600
Factor of Safety	41.09

The factor of safety is 41.09 as seen in the table above. This indicates that the epoxy between the bulkhead and rocket body will do a sufficient job in standing up to the maximum

forces it will experience during recovery. This indicates that the adhesive will sufficiently allow for recovery of the rocket. However, this calculation is conservative, as it assumes that the bulkhead has sufficient strength to stand up to the stresses caused by recovery, and as a result will transfer these forces to the epoxy and to the main body.

3.1.6.2 Mass of Launch Vehicle

The final weight of the vehicle changed significantly due to a change of materials that was out of our hands. Due to the material changes there was a significant decrease in the individual weights of our components. To compensate for this, ballast was added in the Air Braking System coupler. This ballast weight is less than the competition requirement of less than 10% of the overall empty weight of the rocket.

Each final component of the rocket was weighted independently, the weights are displayed in Table 25. The weight of added epoxy and other added materials was determined by also weighing the assembled sections of the rocket and subtracting the sum of the individual components.

Table 25. Breakdown of Vehicle Component Weights.

Component	Mass (oz)
Nose Cone	31.2
Rover Payload Bay	
Rover Body Tube	67
Rover Payload	60.6
Transition Section	35
Recovery Tube Extension	
Body Tube	7.4
Bulkhead	11.2
Coupler	3.66
Parachute Bay	
Recovery Tube	28.2
Main Parachute	64.3
Drogue Parachute	35.7
CRAM	57.5
Air Braking Payload Bay	
ABP Body Tube	1.84
Coupler	11.7
Bulkhead	11.2

Tie Rods (4)	2.25
ABP	73
Fin Can	
Fin Can Body Tube	18.7
Motor Mount Tube	10.4
Bulkhead	5.1
Centering Rings (3)	0.7
Motor	117
Fins (4)	6.325
Motor Retainer	59.2
Total Weight	746.3 (46.64 lbs)

3.1.6.3 Integration

3.1.6.3.1 Motor Mount and Retention

The motor mount and retention systems were previously introduced in Section 3.1.4.6. As previously stated, the motor mount consists of a motor mount tube centering inside of the fin can using three centering rings and a capping bulkhead, and the motor retention system consists of a Quick-Change 75mm system from AeroPack Incorporated. The capping bulkhead will act as motor retention during burnout, preventing the motor from moving axially forward, while the API system will fulfill this role after burnout, when the concern becomes whether the motor will fall out the aft end of the launch vehicle. As an uncentered or unsecured motor could be catastrophic during flight, special precaution was taken during the motor mount and retention integration design and construction.

In general, the motor mount tube is integrated into the fin can by the centering rings and bulkhead. The aft bulkhead described in the Air Braking System section is epoxied to the top bulkhead to connect the motor mount to the next portion of the launch vehicle. While a more detailed description of the construction process can be found in Section 3.1.7, the critical steps will be laid out here.

First, the components were properly sized and created using a CNC machine (for the bulkheads and centering rings), and a hand saw (for the body tubes), their relative positions could be marked. The positions of the centering rings were marked with a pencil on the motor mount (0.75 in, 13.5 in, and 20 in from the end). Multiple measurements were taken to ensure the accuracy of these markings. The 13.5 in centering ring was then secured using JB Weld epoxy to the motor mount tube, before the tube was inserted into the fin can and this same centering ring was attached to the inner fin can using RocketPoxy. While JB Weld was needed to handle the high heat generated by the motor during burnout, the less heat resistant and cheaper RocketPoxy

could be used for the outer edge of the centering rings, because the heat would be less intense out near the fin can. The motor mount needed to be inserted into the fin can before any other centering rings were attached so that the middle centering ring could be accessed for attachment. These first two steps are shown below in Figure 28. In order to ensure that the motor mount was centered during the drying of the middle ring RocketPoxy, the other two centering rings were pushed into their respective positions from either end of the fin can. To allow for these rings to be easily taken in and out of the fin can, two dowel rods were attached to each of the two rings, with lengths of 0.75 in for the aft ring and 6.5 in for the forward ring. Once the middle ring was set, the forward ring (20 in from the rear of the launch vehicle) was taken out and the dowels were removed using a saw and sanding. This centering ring was then inserted around the motor mount tube from the forward end without the dowel rods and epoxied to the motor mount and fin can using the same materials as the middle ring. After this had set, the dowels were removed from the final centering ring (0.75 in from the rear of the launch vehicle), the fins were inserted, and both the final centering ring and the capping bulkhead were epoxied in the same manner as described for the previous ring.

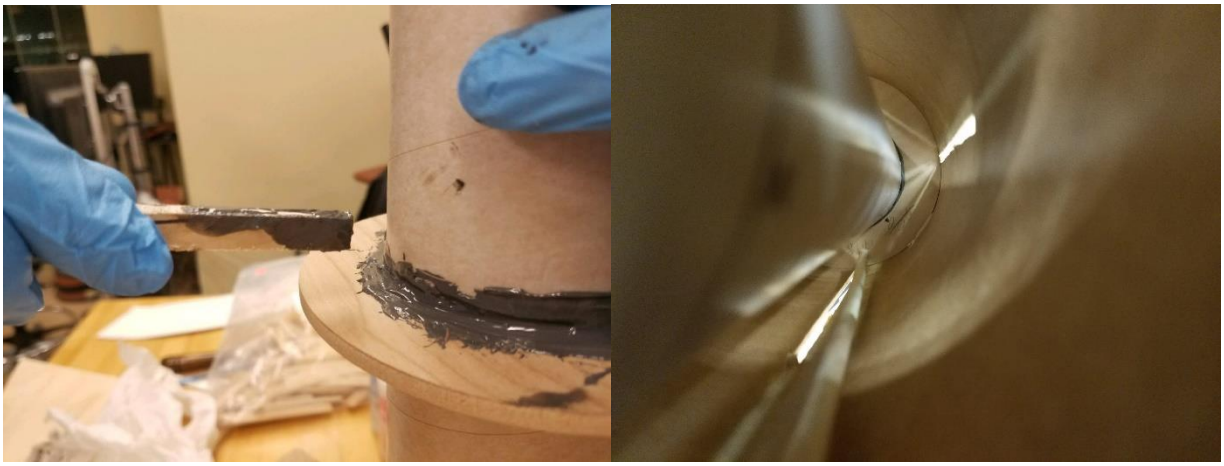


Figure 28. Initial centering ring being attached to motor mount tube (left), before being epoxied inside the fin can (right) with only one ring attached.

The motor retention system assembly was integrated into the launch vehicle by first sanding down the 0.75 in of motor mount tube that protruded from the end of the aft centering ring. JB Weld was then applied in a thin layer to the motor mount tube and a grooved portion of the retainer body. The body was then pressed over the motor mount tube until the tube was set against a lip in the retainer body. The retainer was rotated to spread the epoxy and any excess was cleaned off. After the epoxy set, the motor casing could be inserted into motor mount tube until it was flush with the bulkhead and the retainer cap was then threaded over the motor casing to secure it in the launch vehicle. Figure 29 below shows the installation of the retainer assembly and the final pre-launch assembled motor inside the retention system.

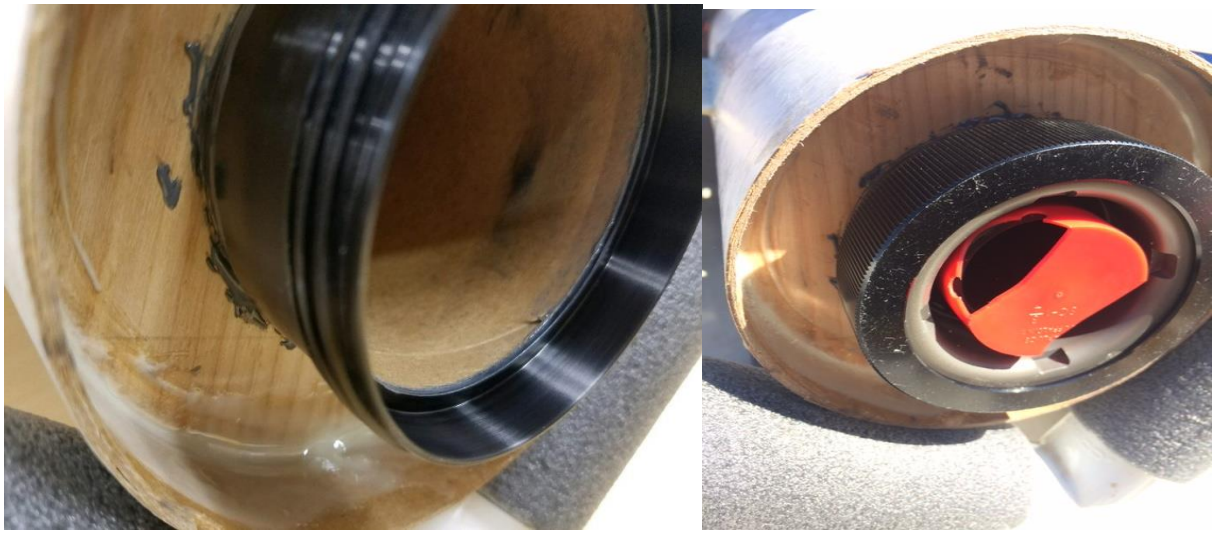


Figure 29. Installed Motor Retention System without cap (left) and Fully Installed Motor inside Retention System (right).

Although this motor mounting system was effective in the sub-scale launch and in the full-scale test launch, multiple load analyses were still completed to ensure structural stability with the final launch vehicle design. First, the maximum thrust of the motor, 400.1 lbf, was divided by the surface area of the cap bulkhead that will interact with the motor during its burn stage, 27.34 in^2 to give a maximum normal stress on the bulkhead of 14.63 psi. Plywood has a yield strength of ~10,000 psi, which makes it more than capable to hold the motor in place. Using the same maximum thrust of the motor and the surface area of contact between the bulkhead and the motor mount tube, the force felt by the JB Weld attaching the motor mount tube to the cap bulkhead will be only 166.3 psi during flight, which is also well under the 3,960 psi yield strength of JB Weld. The only force felt on the phenolic motor mount tube will be due to the shear stress between the motor and the phenolic itself. This was calculated by dividing once again the maximum thrust of the Cesaroni L1395 motor, 400.1 lbf, by the interior surface area of the motor mount tube, 230.44 inches, to give a stress of 1.74 psi. Phenolic has a longitudinal tensile strength of 12,000 psi, which makes it more than capable of handling the load of this design.

To further validate the motor retention system, another load analysis was conducted on the adaptor piece of the retainer to ensure that the JB Weld epoxy would hold throughout the flight. The retainer comes into contact with 0.75 in of motor mount tubing, thus the epoxy was applied over a surface area of 7.363 in^2. The maximum stress on the cured epoxy was found by dividing the weight of the motor, 9.457 lbs, by the surface area covered by epoxy. This yielded a stress of 1.284 psi, which again is much less than the yield strength of JB Weld epoxy, 3,960 psi. This analysis briefly shows that this system can be relied upon during flight

3.1.6.3.2 Recovery System

The main function of the recovery subsystem is to ensure a safe and controlled descent. The recovery is a dual-deployment system with a drogue parachute deploying at apogee and the main parachute deploying at 650 ft. The drogue parachute is in place to allow the rocket begin slowing its descent, without applying too much force on the rocket. This system has triple redundancy, including three altimeters and three black powder charges in order to guarantee safe and controlled descent. The low velocity upon landing assured by the recovery system will allow for reusability and protection of the rover inside the nose cone.

The recovery system is located in the center of the rocket as seen in Figure 30. It is offset towards the front within the recovery tube. The CRAM is located seven inches from the top of the center body tube. The central location of the CRAM allows for the parachutes to be compacted on either side of the CRAM so that they can easily deploy individually. The drogue parachute is located on the fore side of the CRAM while the main parachute is located on the aft.

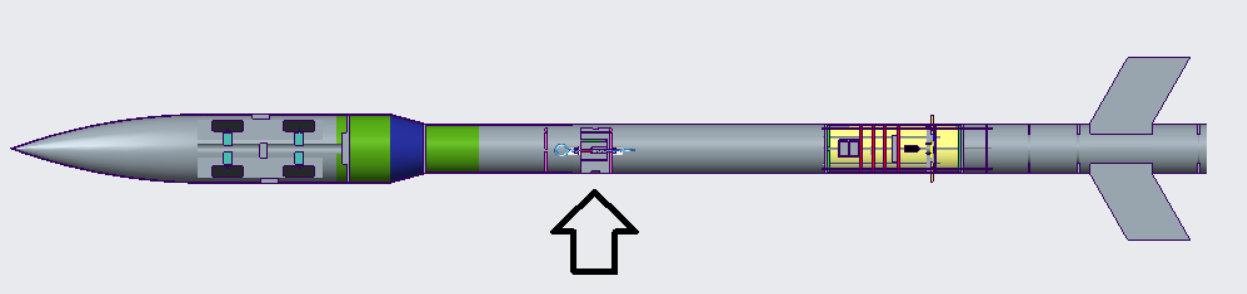


Figure 30. Location of the CRAM in the Vehicle.

The CRAM's core and body is made of 3D printed plastic. The CRAM is inserted into the body of the rocket and secured using a screw-to-lock mechanism. This mount is 3D printed and epoxied on the inner diameter of the body tube. This allows for efficiency and access to the CRAM. To ensure the CRAM is securely fastened during flight, holes will be drilled through the body tube, mount, and CRAM and three screws will be inserted perpendicularly to prevent the CRAM body from unlocking. Acrylic bulkheads will be fastened on both sides of the CRAM to protect the CRAM itself from the black powder charges. One bulkhead will be glued on while the other will be screwed on to allow access to the core. PVC pipe will be glued into the holes within the CRAM body to contain the black powder charges. The bulkhead and PVC pipe can be seen being glued to the CRAM in Figure 31.

Figure 31. Attaching the PVC Pipe to the CRAM.

The upper and lower parts of the rocket are attached to the center by shear pins that allow for separation upon the detonation of the black powder charges. The location of these pins and the separation points can be observed in Figure 32. The front section of the rocket will be held together by four shear pins and the bottom section of the rocket will be held by four shear pins as well.

Figure 32. Location of Shear Pins.

Important hardware associated with the recovery system includes the ⅜ inch eyebolts and the ⅜ inch quicklinks. The shock cords tether the rocket together at bulkheads in each section of the rocket via the quicklinks that are then attached to the eyebolts. The eyebolts are screwed into the bulkheads in each section of the rocket. The eyebolts are made of forged steel and are rated for 1,400 lbs. The quicklinks are made of zinc-plated steel and are rated for 2,200 lbs.

Three holes were drilled into the center section of the rocket in order for the separate redundant subsystems to have more accurate pressure readings. This also allows access to the

CRAM's power switch externally. Using a guide, the holes were cut out in even spacing around the body tube. This process can be seen in Figure 33.

Figure 33. Cutting Slots for the Recovery System.

3.1.6.3.3 Fins

The four fins have been placed at the rear of the rocket, 6.0 inches forward from the bottom of the fin can. This position was chosen to move the center of pressure far enough aft of the center of gravity so to provide robust stability and minimize wobble, while being far enough from the launch pad so as not to disturb launch operations. The restoring force this provides will compensate for any wobble in the rocket due to wind gusts: it will realign the orientation of the rocket with the desired vertical flight path. Since four fins were used, they have been spaced at 90.0 degree intervals about the same axis of the body to distribute the forces on them equally around the rocket.

The fins were constructed by using prefabricated birch plywood plates, as opposed to our previous material of carbon fiber. The plates were first cut using CNC milling into the desired parallelogram shape seen in Section 3.1.4.3, Figure 10, with a root and tip chord of 7.0 inches, a height of 6.0 inches, sweep angle of 30.0 degrees, and a rectangular extension on the bottom of each fin of 1.19 inches. Each fin is 0.25 inches thick, an increase from the previously planned 0.125 inch thickness. This change is due to the decrease in strength in plywood compared to carbon fiber, and the thickness will account for this. After the fins were cut into a proper shape, they were sanded to create a symmetrical airfoil. The leading edge of the fins was sanded at 90

degrees, and the trailing end was sanded at 45 degrees to achieve the desired qualities. To attach the fins to the rocket, four 0.25 inch slots were created in the rocket tube through which the fins can be inserted. These slots were originally part of the fin can when it was ordered, but due to the change in material the slots were cut by hand using a Dremel. The fin can with fins can be seen in Figure 34.

i 

Figure 34. Fin Insertion Process.

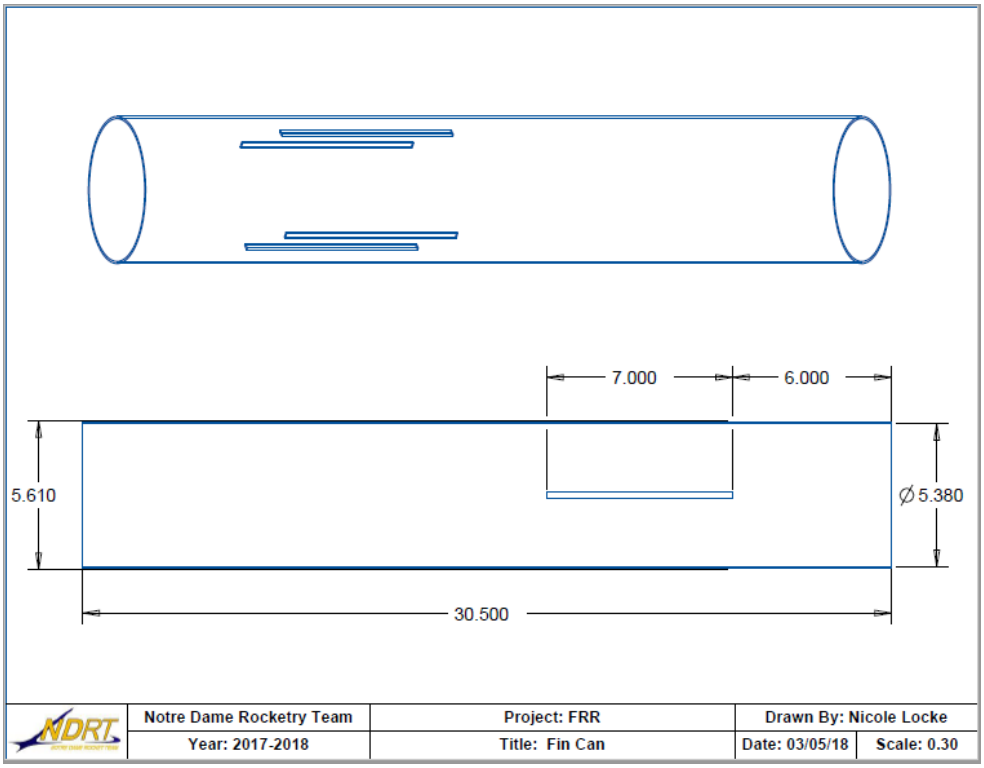


Figure 35. Final Fin Can Design. Units in inches.

After the fins were inserted, Rocketpoxy was used as an adhesive to completely attach the fins to the body of the rocket. The fins were inserted such that their rectangular bases face towards the body tube. To ensure that the fins were aligned at exactly 90 degree intervals, a fin alignment mechanism was developed. The fin alignment mechanism, which can be seen in Figure 36, consists of two circular plywood plates that are laser cut so that the fin holes are exactly 90 degrees from one another. These plates are then placed at each end of the fins for stabilization during the epoxying process. The same mechanism was used in last year's rocket design with satisfactory results. The fin can was placed in the hole in the center and then was suspended horizontally using stands made of PVC pipe. The first fin was placed into the top pre-cut slot cut into the fin can at a right angle relative to the fin can and motor mount. It was then epoxied both to the outside of the motor mount and to the inside and outside of the fin can. While the epoxy dried enough to support the fin, the fin was held at the right angle by the alignment rings. After the epoxy had dried enough to support the fin on its own, the fin can was rotated 90° so that the fin already placed was level with the ground. This was measured using a level tool. Then, the second fin was placed into the new top pre-cut slot in the fin can at a right angle relative not only to the fin can and motor mount but also relative to the fin already in place. This was determined by using a level tool as well. The fin on top was then epoxied to the outside of the motor mount, the inside of the fin can, and the outside of the fin can and was held in place by the alignment rings while the epoxy dried enough to support the fin. The process used

to integrate the second fin of the rocket was then used to place the third and fourth fins as well. All safety protocols were followed during the epoxying process. This entire process is documented in Figures 37 and 38.

Figure 36. Laser cut Fin Alignment Mechanism.

Two different epoxies were used to secure the fins to the motor mount and the fin can. To attach the fins to the motor mount, JB Weld was used due to its strength and its effective performance under the high temperature conditions that the motor mount experiences due to the motor. Therefore, the epoxy will retain its strength during the launch so that the fins will remain attached to the motor mount. Because the fin can does not experience these extreme temperatures, Rocketpoxy was used to attach the fins to the fin can. Rocketpoxy was applied to both the inside and the outside of the fin can to assure that the fins would remain attached and not shift or change orientation during the flight. When the Rocketpoxy was applied to the outside of the fin can, it was filleted using a wooden tongue depressor to ensure that the coat was even across the fin, therefore evenly distributing the force throughout the whole coat of Rocketpoxy and minimizing the effects of drag from the epoxy.

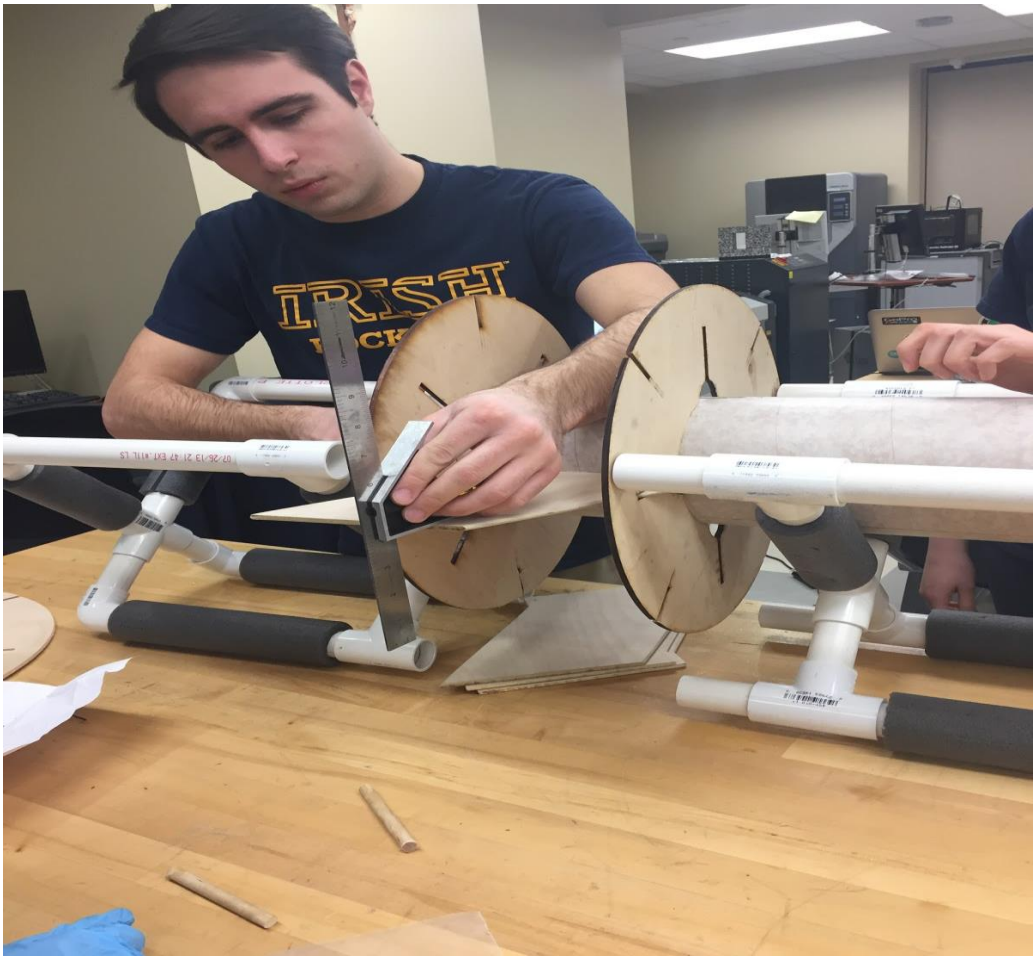


Figure 37. Fin Leveling Process.

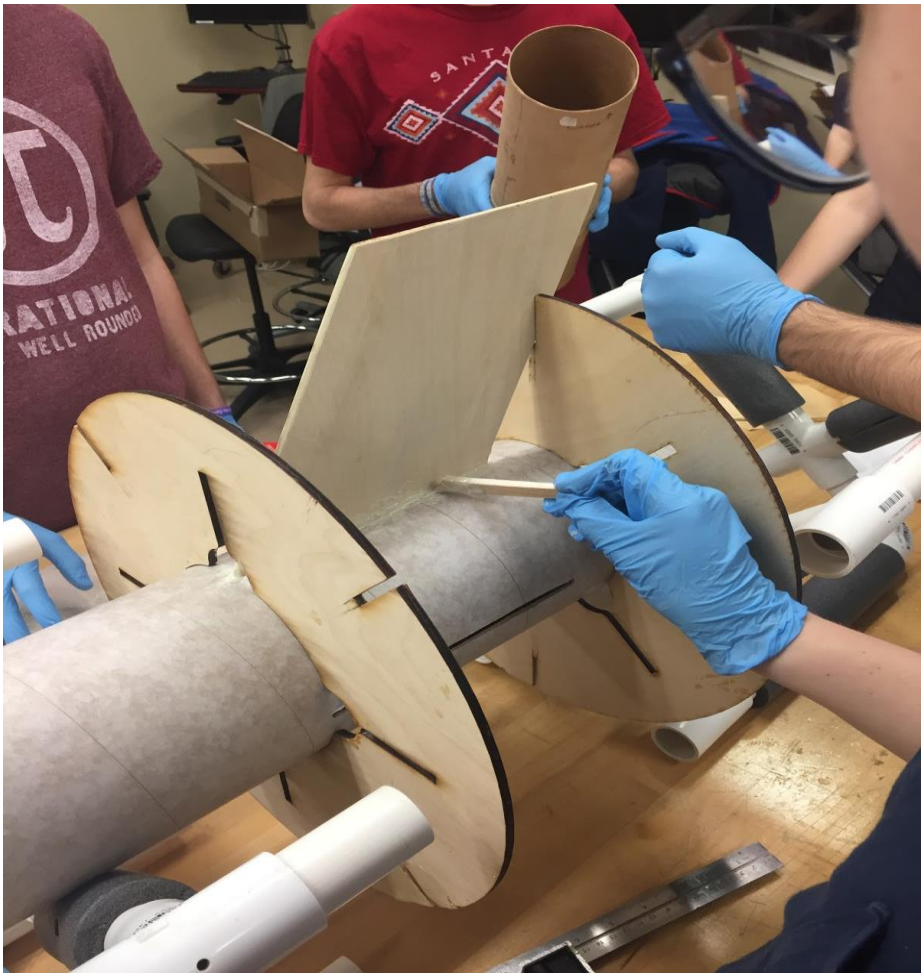


Figure 38. Fin Epoxying Process.

3.1.7 Construction and Assembly

Construction of the full scale launch vehicle began on February 18th. Since the CDR, a major change in materials used for the launch vehicle has occurred. Our supplier of carbon fiber components was unable to provide the team with the necessary body tube sections, couplers, and fins in time to allow for a proper test launch prior to the FRR. For this reason, the team has opted instead to use phenolic and plywood to replace those components. This was elaborated upon in the Addendum to the CDR submitted on February 21st, 2018. It is also elaborated upon in Section 3.1.6.1. A general construction and assembly plan was laid out for the full scale vehicle in the CDR. This plan was followed as a general guide, but some changes had to be made due to unforeseen issues during construction and the change in materials for certain sections. Proper safety precautions were followed at all points during construction and assembly. All materials used in this design had been used previously, so many team members had experience with the proper techniques required to safely construct and assemble each section. Fiberglass components were assembled by professional services (transition section and Rover Payload Bay). Phenolic

tubing was purchased in place of the carbon fiber body tubes and couplers. These tubes were purchased at a set diameter and cut to proper size by the team. The construction and assembly will be laid out in 3 main sections. These are the same sections discussed earlier that correspond to the separate tethered sections during recovery. The first is the Rover Payload Section, which consists of the nosecone, the rover payload, and the transition section. The second section, the Air Braking Payload Section, consists of the main body tube, the recovery components/parachute, and the air braking payload. The third section, the Fin Can, consists of the Motor mount and fins. Each section will be further divided into their specific components. For certain components, no construction was required as the component is purchased in its entirety. Therefore, only assembly will be covered for these components.

Many aspects of the launch vehicle were assembled/constructed concurrently. The Air Braking Payload, CRAM, and Rover Payload were constructed separately from the rest of the launch vehicle in their respective teams.

All components that were attached with epoxy were sanded down before hand to increase the surface area that is touching, and therefore the strength of the connection. This was done using a relatively fine grade sandpaper. Sanding was also used for any pieces that did not fit properly during assembly. This primarily applied to laser cut components, as the laser cutter accessible to the team is not extremely precise. The team accounts for this by cutting out pieces slightly larger than necessary, then sanding it down either manually or with a belt sander.

3.1.7.1 Rover Payload Section

The Rover Payload Section consists of the nosecone, the rover payload, and the transition section.

1. The nosecone will be purchased from Apogee Rockets in its entirety. Therefore no construction is required. More information on the nosecone specifications can be found in Section 3.1.4.1. Prior to assembly, the nosecone was sanded with very fine sandpaper to reduce drag. As discussed in Section 4.1.3.1.5, the nosecone must be capable of separating from the rover body tube and a bulkhead was placed in the nosecone. Another bulkhead, which will be used to integrate the nosecone with the rover payload, was placed in the shoulder of the nosecone. These bulkheads were placed in the nosecone using epoxy. When the charges in the Rover Payload are deployed, the nosecone will be ejected. These bulkheads were made using two laser cut pieces of ¼ in plywood.
2. The details of the Rover Payloads construction can be found in Section 4.1.4. As discussed earlier, a plywood bulkhead connects the nosecone to the Rover Payload
3. The transition section was originally planned to be purchased from Carolina Rocketry in its final configuration. Unfortunately, this component was among those that could not be delivered on time. The team was able to purchase another transition section made of fiberglass with an inner diameter one hundredth of an inch less than the original design. This component was purchased from Summit Aviation. The team does not have the proper tools or expertise to

properly machine a piece of this complexity out of fiberglass. In the event that the transition section purchased from Summit Aviation did not arrive in time, a phenolic transition section was constructed by the team using techniques recommend by vendors. The two sections were fused together with a 30 minute cure epoxy. Due to the drastically increased chance of manufacturing error when constructed with phenolic and epoxy and the overall superior material properties of fiberglass, the fiberglass transition section will be used. The transition section features a shoulder/coupler section on both ends that allows it to integrate with the Rover Payload at the wider section, and the main body tube at the smaller section. The top of the transition section is integrated to the rover body tube by epoxy along the inside of the body tube and outside of the coupler. A plywood bulkhead was placed at the top of the coupler that attaches the transition section to the main body tube. The bulkhead was then epoxied to the coupler. This bulkhead was cut from the purchased plywood with the CNC router. This bulkhead was cut with a hole through the center to allow an eyebolt to thread through. This eyebolt will be used for recovery purposes and is elaborated upon in Section 3.2.1. As this entire section must separate during recovery, shear pins will be used to for integration purposes between the main body tube and the bottom of the transition section. Holes were drilled into the transition section and the body tube using a drill press. 4 shear pin holes were drilled at 90° apart to ensure that the sections do not separate prior to the black powder charges being activated. 4 shear pins were chosen based on recommendation from our mentors. The team has used this configuration for many years.

3.1.7.2 Recovery and Air Braking Systems Section

The Recovery and Air Braking Systems Section consists of the main body tube, the recovery system and the air braking coupler.

1. The main body tube was originally going to be purchased from Carolina Rocketry, already cut to specifications. As discussed earlier, the shipment of the carbon fiber body tubes was delayed, so this section was instead made using phenolic. This body tube houses both the drogue and main parachutes, the CRAM (recovery module), and the air braking payload. Construction of the CRAM is covered in Section 3.2.1. Construction of the air braking payload is covered in Section 4.2. The only construction required for the main body tube was cutting down the phenolic tube to size. In order to integrate the main body tube with the transition section coupler, shear pins will be used as discussed earlier. This allows the vehicle sections to be properly integrated during launch but to be capable of separating during the recovery stage. Similarly, the main body tube and air braking payload coupler will be integrated using shear pins to facilitate separation during recovery. The fore bulkhead, located at the top of the air braking payload coupler, has a hole through the center for a 1,500 lb rated eyebolt as discussed in Section 3.2.1. The shock cords will be attached to this eyebolt with a 2,000 lb rated quicklink, as discussed in the section mentioned above. The bulkheads were attached to their respective components using epoxy. All bulkheads are made of plywood.

2. The air braking payload construction is covered in Section 4.2. The air braking payload coupler and the fin can will be attached using threaded rods and two bulkheads, one at the top of the motor mount and another at the bottom of the air braking payload coupler (Aft bulkhead). 4 rods, each 0.25 inches in diameter, extend through the air braking system through both bulkheads. Locknuts are used to secure the rods to the bulkheads. These bulkheads are also made of plywood. 4 holes were machined into the bulkhead for the rods to pass through.

3.1.7.3 Fin Can

The Fin Can consists of the motor mount and the fins.

1. The motor mount system is composed of two body tubes, 1 bulkhead, 1 motor retention system, and 3 centering rings. The bulkhead discussed in the Air Braking Payload Section that is attached to the motor mount serves to integrate the ABP section with the fin can. The centering rings ensure that the body tube the motor casing will be placed in is properly centered. The two body tube sections were purchased from professional vendors and cut to specifications by the team. The plywood bulkheads and centering rings were cut using a CNC machine by the team. Epoxy was used to assemble the centering rings and bulkhead within the fin can tube at the proper locations. A Dremel was used to make the fin slots. Care was taken to ensure these slots were parallel to the axis of the body tube. These slots were placed 90 degrees apart. To form the entire motor mount, the motor mount tube was inserted into the 3 centering rings that were epoxied to the fin can body tube. The motor retention system is described in Section 3.1.6.3.1. To attach the centering rings, epoxy was applied to the inside of the fin can body tube slightly before the specified location. The ring will be inserted into the tube, which will push the epoxy slightly forward into the proper location, along with the ring. More epoxy will be applied at contact points between the tube and the ring. The bulkhead was attached in a similar manner. In order to ensure the entering rings are straight, the motor mount body tube was inserted (without adhesive) through the centering rings in the same configuration it would be in during assembly.
2. The fins were originally going to be made of carbon fiber. Along with the other carbon fiber components, the necessary materials would not have arrived on time. Therefore, the team is making fins out of plywood. The team has done this is the past and the plywood provided adequate performance. The fins were cut by the team with a CNC router. Once the fins were cut to size, the edges were sanded down using a set of sanding blocks that the team has used in the past. The leading and trailing edges of the airfoil were sanded down to different profiles. The thicker radius at the leading edge helps keep flow attached over the fins. The leading edge has a thinner radius than the trailing edge. The tip chord of the fins was also sanded to reduce drag. To integrate the fins into the fin can, the fins were inserted into the slots that were cut in the fin can with a Dremel. At this point, they will be inserted until the root chord is level with the motor mount. The fins were then epoxied and filleted to ensure that the fins remain attached during flight without a dramatic increase in drag. Epoxy was applied on the underside of the root chord of the fin prior to inserting it into the fin can. More epoxy was applied at the intersection

between the fin can and the fins themselves. A fin alignment guide was used to ensure that the fins remain perpendicular to the fin can while the epoxy is curing. From past experience, it was determined that epoxying one fin at a time is best to ensure that the epoxy does not shift due to rotation while the epoxy is curing.

3. When assembling the fin can, it is crucial that the centering rings were epoxied in such a way that rail buttons can be properly placed. To ensure the rail button remains attached properly, a small block of wood was placed on the inside of the fin can. A hole was drilled through the fin can into the block of wood. The rail button was then drilled into the wood block through the phenolic tubing and 3D printed section. If the centering rings were applied improperly, it can lead to the rail button locations being inaccessible, which means that the wooden block cannot be used to support the rail button. This will lead to safety problems during launch as the rail button is more likely to fail.

4. Since the rocket is variable diameter, a standoff was needed for the rail buttons. These were printed on Notre Dame's campus to allow the buttons to be mounted clear of the larger 7.5" diameter. These standoffs were epoxied radially in the same location. One was placed one inch from the aft of the fin can and the other one inch from the fore of the fin can. They were aligned using a straightedge and verified with a piece of 1.5" rail to ensure that they do not interfere with the main fins. Inside the body tube at these locations, a nut, washer, and wooden blocks were epoxied so that bolts can secure them. The buttons were secured with 2.5" bolts that attach to these interior nuts.

3.1.8 Vehicle Risks and Mitigations

The Vehicles Sub-team presented in its Critical Design Review a list of risks that it anticipates could delay the project. They will be reproduced here with information on how the risks *were* mitigated to ensure completion of the project. The risks were split into two categories: *project risks* and *launch risks*. Projected risks can be further split into four distinct categories: resources, time, functionality, and safety. These different types of projected risks will be discussed in the next section (3.1.10.1). A summary of the projected discussed below can be seen in Table 26 and the launch risks can be seen in Table 27.

3.1.8.1 Project Risks

The Vehicles Sub-team is organized as such that members own certain sections and work on these throughout the design and construction process. For example, the owner of Rover Payload integration worked with the Vehicles Lead and the Rover Sub-team in order to design the structural aspect of the payload. When the owner of parts became unavailable, the Vehicles Lead could easily select take over. Most members have a general knowledge of other parts of the rocket beyond what they own; as such, they were able to fill in for missing members when appropriate.

The Sub-team declared in its CDR that it was unlikely that the team would run out of physical resources, as plans were made for any needed resources and they would be ordered before they were required; however, critical oversights did slow the project. The supplier of our carbon fiber body tube would not have been able to send us the body tube until after the FRR report was supposed to be submitted, for example. This forced us to switch to phenolic tubing for the body of the rocket and the fins.

All members are aware of launch dates and deadlines and work with an internal deadline of 1 week before the NASA SL or launch deadline. In cases of testing, scheduling was done in the month of November for December test dates, results of which were included in the CDR. Testing prior to FRR was scheduled in January for February and March.

Functionality of the rocket is a top priority for the team because it directly affects the mission success criteria in Section 3.1.2. Testing, computer models, and subscale models helped the Sub-team determine what steps need to be taken to ensure the final product met project goals. The Sub-team emphasized the need for robust verification methods in order to ensure that what has been designed met the requirements. Functionality went hand in hand with time because whatever did not work as intended went through a redesign process. Resources also played a role because resources were moved around so that certain functionalities can be perfected. Functionalities that directly affected flight were prioritized.

Dangerous materials (rocket motors and fiberglass) and tools were used to construct the rocket. Ensuring safety through proper protective equipment and communication with the team's safety officer mitigated risk to the Sub-team's members. The motors were handled by the mentor; therefore, this was not a concern.

Table 26. Project Risks and Mitigations.

Risk	Likelihood/Impact	Mitigation Technique
Budget	Low likelihood/Low Impact	The Vehicles Sub-team has developed a budget within whose bounds it always tried to stay. There is material left over from previous years. This material is used to perform tests or to test out ideas, particularly in form of payload integration. The budget for the Vehicles Sub-team is shown in Appendix L. The team estimates that the budget can only go down because it shot high to start choosing expensive material (such as fiberglass) that may not end up being needed. This covers the oversights. The only foreseeable budget problems lie in integration material, such as screws and nuts, but these items are not overly costly, and the University workshops keep them.
Time	High Likelihood/Medium Impact	The Vehicles Sub-team is organized in such a way as to help members stay on top of their work while not being affected by those who may be behind. Members own certain sections and work on these throughout the design and construction process. All members are aware of launch dates and deadlines and work with an internal deadline of one week before the NASA SL or launch deadline. In cases of testing, scheduling is done in the month of November for December test dates, results of which are included in the CDR.

Resources	Low Likelihood/High Impact	It is unlikely that the team runs out of physical resources, as plans will be made for any needed resources and they will be ordered before they are required but using up all available resources and not planning will slow the project. Material for construction of full scale is ordered in December for January launches so that any missed material can be ordered in time. In terms of human resources, the Vehicles Sub-team has a member who "owns" a sub-system, but there is usually a secondary person who is somewhat familiar with the sub-system and who can take over should the primary owner not be available.
Functionality	Medium Likelihood/High Impact	Functionality of the rocket is a top priority for the team. Testing, computer models, and subscale models will help the team determine what steps need to be taken to ensure the final product meets project goals. The Sub-team emphasizes the need for robust verification methods to ensure that what has been designed meets the requirements. Functionality goes hand in hand with time because whatever doesn't work as intended must go through a redesign process. Resources also play a role because resources must be moved around so that certain functionalities can be perfected. Functionalities that directly affect flight are prioritized.
Safety	Low Likelihood/High Impact	Dangerous materials (rocket motors and fiberglass) and tools will be used to construct the rocket. Ensuring safety through proper protective equipment and communication with the team's safety officer will mitigate risk to team members. The mentor handles the motors; therefore, this is not a concern.

3.1.8.2 Launch Risks

Concerning the risks involving the launch, three main categories of risks were considered to be of the utmost importance. Any of these three risks could and can have detrimental effects if not properly analyzed and prevented given the necessary criteria. These three risks are structural failures, propulsion, and stability. Proper mitigation of these potential hazards is instrumental in ensuring the safety and performance of the launch vehicle. If any of these forms of failure are not adequately analyzed and prevented, the rocket and personnel can suffer. A major proponent of each risk is presented in Table 27.

Table 27. Launch Risks and Mitigations.

Type	Risks	Cause	Effects	Controls/Mitigations
Structural Failure	Bod Tube (airframe) Failure	Load Capacity of Body Tube exceeded during Launch and or deployment	Airframe Cracks or Breaks, compromising structural integrity and rendering the vehicle non-reusable	Critically Analyzed design to identify weak points. Carried out appropriate calculations to determine expected stresses and use a reasonable safety margin. These have been done and are shown in Section 3.1.6.1. Full Scale launches verified these.
Propulsion	Motor Casing Explosion	Nozzle is clogged by a detached chunk of propellant	Motor casing explodes under pressure and partially or totally destroys the fin can	Made sure that the certified personnel properly checked the motor casing and correctly packs the motor. Analysis has been done on motor casing material.
	Motor Retention Failure	Retention system is unable to hold the motor.	Motor drops out of Launch Vehicle, a serious safety hazard.	Motor Retention has been analyzed to be sufficient according to Section 3.1.6.3.1. Full Scale flight has verified this analysis.

Stability	Vehicle is unstable	The Center of gravity is behind the center of pressure	Flight path will be unpredictable and erratic.	Use OpenRocket and our Performance Prediction Program to ensure proper placement of CG and CP. Physically verify the location of CG prior to all launches.

3.2 Recovery System

3.2.1 Structural Elements

CRAM - The CRAM is the module that contains all of the recovery electronics and black powder charges needed for proper parachute deployment, as well as all of the linkages that connect the parachutes to the rest of the rocket. The CRAM is created from two major 3-D printed parts, the body and the core, the first of which can be seen below in Figure 39.

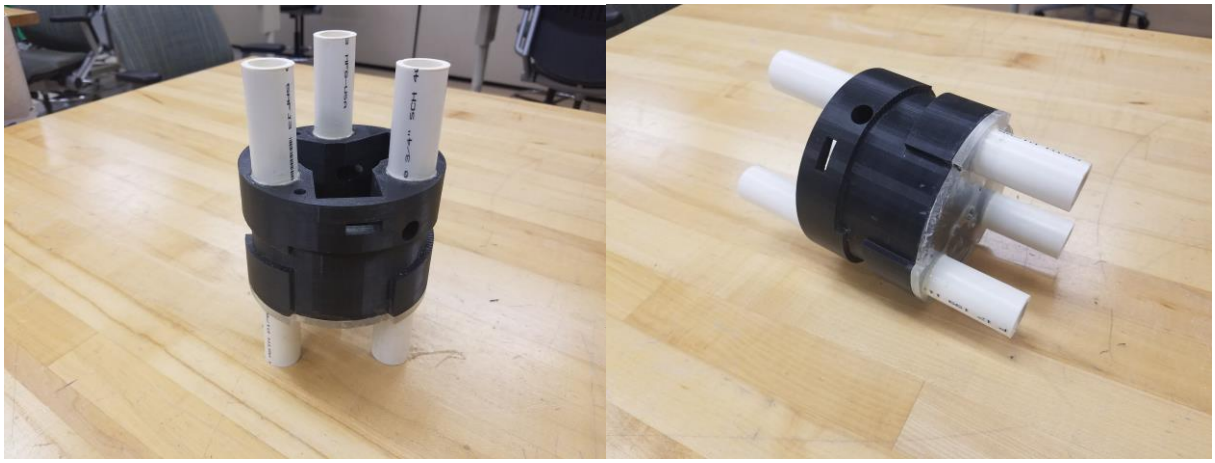


Figure 39. The CRAM body (black) with PVC pipes epoxied into place

Core - The recovery altimeters, as well as the battery boxes that power the altimeters, are mounted to the core of the CRAM, which slides in and out of the body of the CRAM for easy access and rapid turnaround after a successful rocket launch. The core along with the main avionics are shown below in Figure 40.

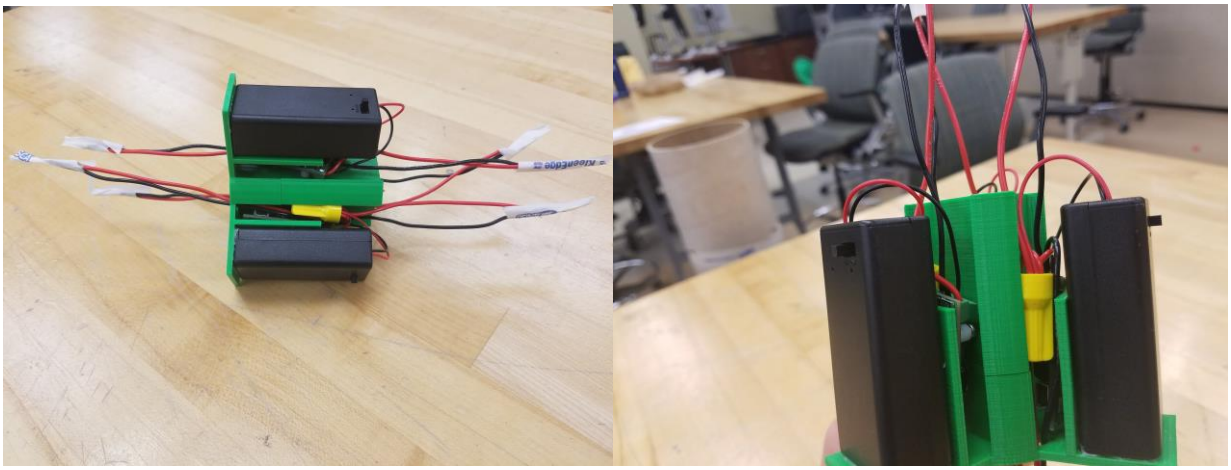


Figure 40. The core (green) with battery boxes (black) and related wiring/avionics.

Mount - The main body of the CRAM was printed with a screw-to-lock mechanism inscribed on the side, which interfaces with a 3-D printed mount epoxied to the recovery body tube of the rocket. This simplifies the process of CRAM instillation, as the entire module can be inserted into the body tube, twisted to lock in place, and bolted to ensure that the CRAM does not twist out of the tube. Access to the CRAM after a launch is as simple as unscrewing the bolts and twisting the entire module out of its body tube. The holes in the side of the CRAM are either for bolts to prevent twisting once mounted in the body tube, or to provide airflow to the altimeters inside the CRAM. The mount can be seen epoxied inside the recovery body tube in Figure 41 below.

Figure 41. The mount (black) shown securely epoxied within the phenolic recovery body tube.

Bulkheads - The bulkheads of the CRAM are made from quarter-inch acrylic and machined from a CNC router to match the shape of the body of the CRAM, as well as ensure that all of the holes in the bulkhead and CRAM match. The bottom bulkhead is permanent, epoxied to the bottom of the CRAM, while the top bulkhead is secured to the rest of the CRAM with 3, 1/4-inch diameter, 4-inch long bolts that run through the entire body of the CRAM, secured to nuts on the bottom of the CRAM. This top bulkhead is what prevents the core of the CRAM from unintentionally sliding out of place. On both sides of the CRAM, there are three large holes and seven small holes cut through the bulkheads and, in the case of the small holes, through the body of the CRAM. In the large holes, two-inch lengths of three-quarter inch PVC pipe is epoxied to the body of the CRAM. These PVC pipes will be where the separation charges will be packed. Three of the small holes in the CRAM are for the bolts that secure the top bulkhead, while the other three small holes are for wire to run from the altimeters to the e-matches that will light the separation charges. The center small hole is for the eyebolts that connect to the parachutes and shock cords. The top bulkhead which can be removable from the CRAM can be seen below in Figure 42.

Figure 42. Top recovery bulkhead with adhered copper shielding.

Shock cords - Both shock cords, for the main and drogue parachute, are 9/16” flat nylon cords measuring 42 ft in length which is long enough such that the parachutes can be attached and the sections of the rocket should not collide with each other on descent. The shock cords are tied to screw-locking stainless steel quick links, which are in turn connected to 3/8-inch, forged construction steel eyebolts. These eyebolts are screwed into an "extreme strength" steel coupling nut, which is housed inside the core of the CRAM. This configuration places the vast majority of the force of parachute deployment on the eyebolts and the coupling nut, instead of the body or core of the CRAM. A shock cord used in the system can be seen below in Figure 43.

Figure 43. Shock cord used in the recovery system.

Table 28 below lists capacities for the various load-bearing components of the recovery system. As detailed earlier, the maximum force exerted on the components occurs at main parachute deployment and the force of this event was utilized when selecting all components. To ensure the robustness of the recovery system, all components have been selected to significantly overcompensate for this anticipated load, as evidenced by the table values.

Table 28. Load capacity of force-bearing recovery components.

Component	Load Capacity
Eyebolts	1400 lbs
Coupling Nut	45,600 lbs
Quick Links	3900 lbs
Shock Cords	2400 lbs

Images of the assembled recovery system are shown below. Figure 44 below shows the fully assembled CRAM independent of other components and Figure 45 shows it inserted into the recovery body tube. Figure 46 shows the full system linked together as it appears before folding and removed from body tubes.

Figure 44. CRAM v4 (fully assembled), as it appears before insertion.

Figure 45. CRAM v4, inserted into the recovery body tube.

Figure 46. Assembled recovery system.

3.2.2 Electrical Elements

There are several electronic components of the CRAM that are crucial to the success of NDRT's recovery system. The CRAM is triple redundant, so every electronic component is present in three separate places in the system. The electronics allow NDRT to deploy black powder charges at a given altitude in order to break the shear pins holding the rocket body together and release the parachutes. There are three Raven3 altimeters within the CRAM that release a current when the rocket reaches a certain altitude. Each altimeter is responsible for releasing two different charges, one for the drogue parachute and one for the main parachute. The altimeters are set to release these currents at a delay to prevent all three black powder charges from going off simultaneously. The three drogue charges are set to be released at apogee, 1 second after apogee, and 2 seconds after apogee, while the three main charges are set to be released at 650 feet, 600 feet, and 550 feet. When the charge is released, an e-match will be ignited which will set off the black powder charges themselves. Furthermore, each successive altimeter will ignite a larger amount of black powder to insure that the shear pins are broken and the parachute will be deployed. The altimeters are connected to the e-matches using red solid-core wire, and the e-matches are connected back to ground using black solid-core wire.

Altimeters - The altimeters used in the recovery system are *Raven3* from the *Featherweight* company. They are considered robust and appropriate for the system due to their impressive

specifications and reliable integration method. Table 29 below describes the key features of the *Raven3* and Figure 47 provides images of the altimeters.

Table 29. Altimeter specifications.

Feature	Value/Specification
Barometric pressure readings	200 Hz, +/- 3% accuracy
Output voltage	8 V, 20 Hz
Output current	30 Amp, 40 Hz
Functional orientation	Vertical, either side up
Output terminal	Screw-lock (x4)
Battery voltage readout	Number of beeps after power-on
Connectivity verification	High beeps = output number connected Low beeps = output number disconnected
Post-flight apogee readout	Beeps corresponding to digits after vertical rotation

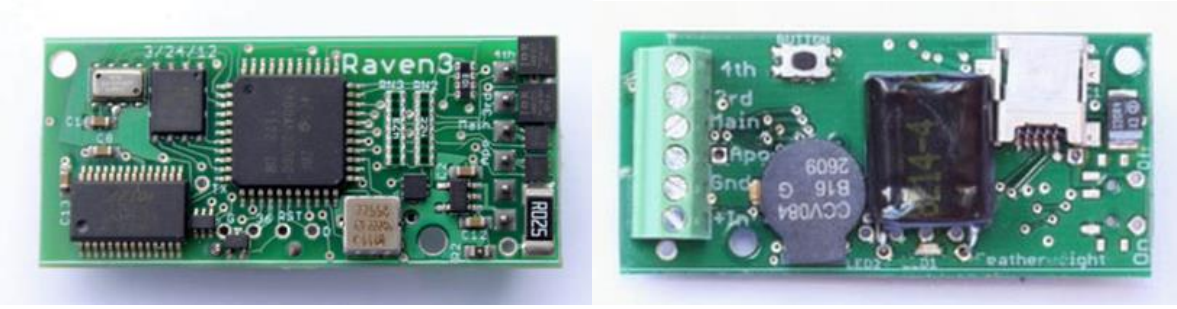


Figure 47. Raven3 altimeters, top and bottom views respectively.

Switches - The arming switches for the recovery system are connected directly to the 9V power source as part of the battery boxes which contain them. These switches are accessible from the exterior of the rocket so they can be armed from the launch pad by sliding them to the "On" position with a screwdriver or other narrow rod. These promote the robustness of the system because they are a commercially available item and they compactly and reliably combine the power source with the arming mechanism. This will ensure that the system is not armed prematurely at any time and the system connections remain intact during assembly and flight. The battery boxes and switches can be seen below in Figure 48.

Figure 48. Battery boxes with switches used to power and arm recovery system.

Connectors - The connectors used to secure avionics wires together are Twist-Lock Wire Splice connectors from *McMaster Carr*. They have been selected for their robustness, compactness, and accessibility. Their commercial availability from a reputable manufacturers provides overall security. Specifically, they are rated for the wire gauge used in the recovery system (18 gauge) and their description confirms their intended use for "quick, secure wire connectors without crimping." These features are important because the recovery system will need to be assembled and disassembled several times over the course of its life and will need to be reliable for every launch. The original intended wire connectors have been replaced with these varieties because of the reduced footprint within and outside the CRAM where they are used. The wire connectors can be seen below in Figure 49.

Figure 49. Space-Saver Twist-on Wire Splicing Connectors from McMaster Carr.

Wires - The wires themselves used in the recovery system are also critical to overall robustness and functionality. To this effect, the type, gauge, and color of wiring has been selected specifically for the needs of the system. In all cases, 18 gauge wiring is used because it is thick

enough to avoid fatiguing under repeated uses and makes tight connections with the altimeter and connectors. Solid core wire is used to connect the altimeter ports to e-matches because the altimeter outputs are tightened most securely with this variety. Stranded core wire is used to connect to the common voltage source of the altimeter and e-matches because it most reliably intertwines with other wires inside the twist-on splice connectors. Lastly, the color choices play into the overall reliability because they prevent confusion when connecting a inspecting a group of wires. The wiring was purchased from reputable hardware manufacturer *McMaster Carr.*

E-matches - The e-matches used in the recovery system have undergone serious testing to verify their reliability. As the likely source of recovery issues in the past, extra care was taken in selecting and testing these items. These tests along with all other recovery tests, are shown in the following redundancy and reliability section. Figure 50 below shows images of the e-matches that will be used in the recovery system.

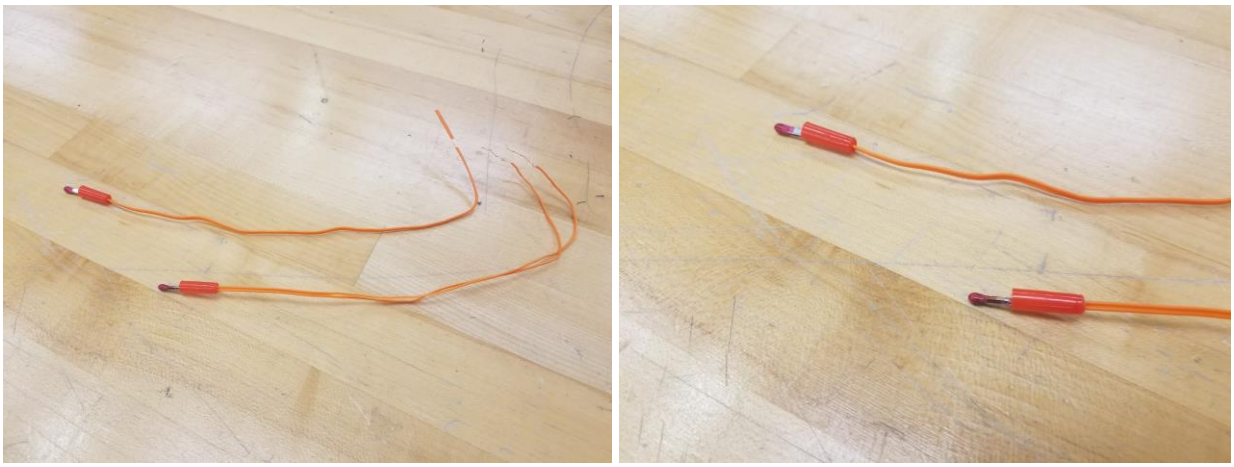


Figure 50. E-matches (exposed from protective sheath) for use in recovery system.

3.3.4 Redundancy Features

Every electrical component in the recovery system is triple redundant. The three altimeters in the CRAM are independently powered by their own battery boxes. The altimeters are all individually programed and ground tested. Each of the three altimeter controls two ejection charges, one for the drogue parachute and one for the main parachute. In the case that two of the three altimeters fail, each ejection charge carries enough black powder such that a single altimeter is fully capable of deploying both the drogue and main parachutes.

3.3.5 As-built Parachute Sizes and Descent Rates

The parachute sizes have remained constant throughout the duration of the design process, but will be restated here for the sake of thoroughness. The drogue parachute is a 24” helical parachute with a bleed hole. It is made of nylon and will be deployed at apogee. Based on

final mass of the rocket at 47.8 lbs, this will lead to an approximate descent rate of 78 ft/sec. This calculation is derived from the team-developed Runge-Kutta Matlab program and verified by descent calculator software available from the parachute manufacturer. The graph from the manufacturer can be seen in Figure 51 below.

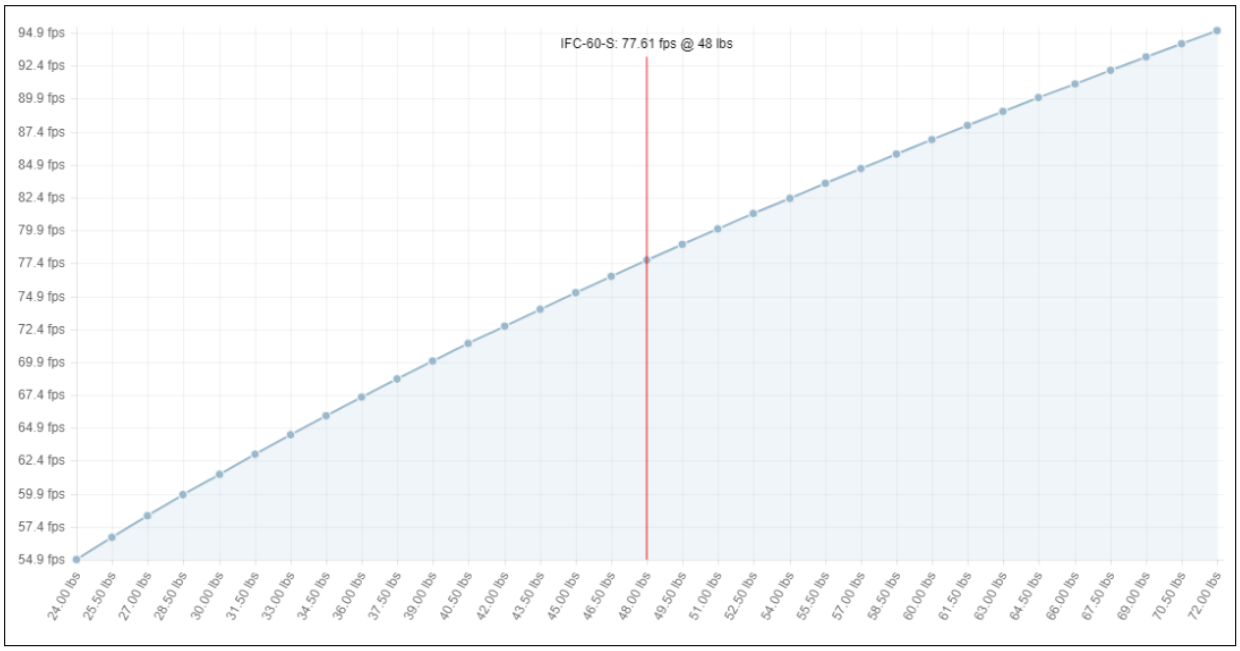


Figure 51. Descent speed under drogue.

The main parachute is a 144” helical parachute with a bleed hole. It is made of nylon and will be deployed at 650 ft AGL. Based on the final mass of the rocket at 47.8 lbs, this will lead to an approximate descent rate of 13 ft/sec. This calculation is derived from the team-developed Runge-Kutta Matlab program and verified by descent calculator software available from the parachute manufacturer. The graph from the manufacturer can be seen in Figure 52 below.

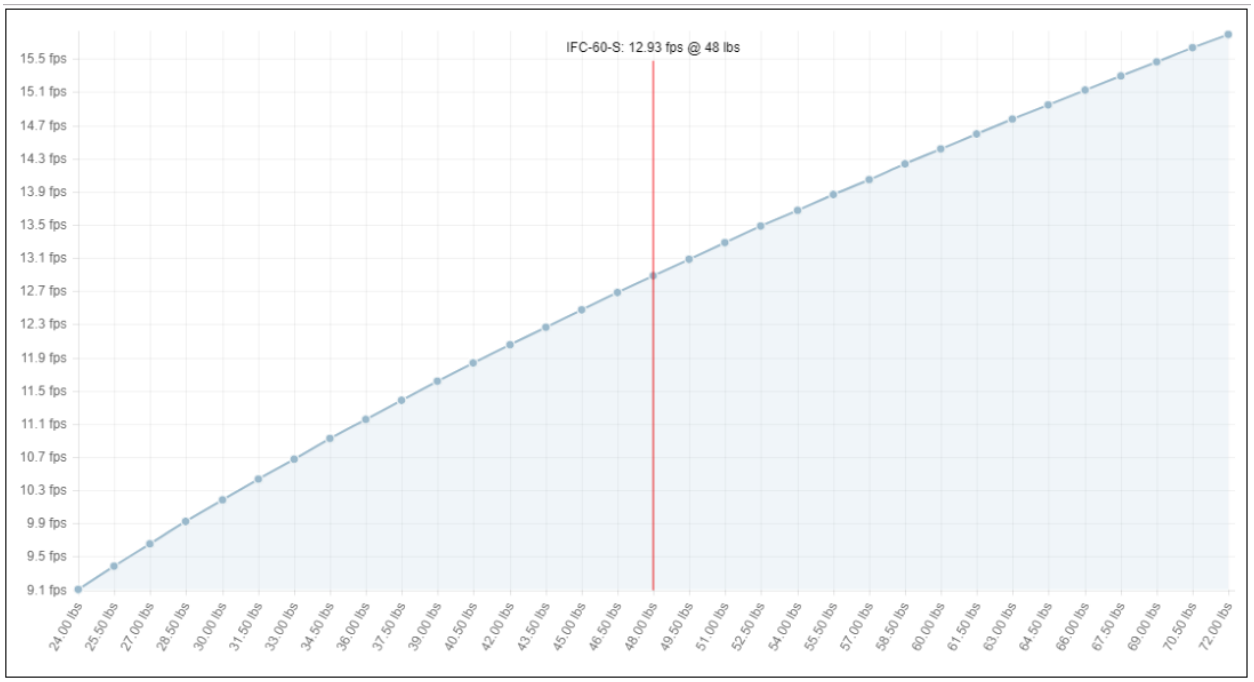


Figure 52. Descent speed under main.

At these rates, the following important flight values are achieved.

Table 30. Descent times.

Flight portion	Descent Time (sec)
Drogue deployment → main deployment	59
Main deployment → landing	50
Total	109

Furthermore, at the predicted main descent velocity and utilizing the conversion chart seen below in Table 31, the kinetic energy of the *entire rocket* at landing is limited to 118.8 ft-lbf and 52.8 ft-lbf for the fin can, which is the heaviest section. These values are below the NASA-designated maximum kinetic energy even before the individual sections are considered, which reduces the value even more.

Table 31. Useful units conversions for KE calculation.

SI Unit	Imperial Equivalent
1 Joule (J)	0.7376 ft-lbf

1 kilogram (kg)	35.274 oz
1 meter per second (m/s)	3.28 ft/s

3.3.6 Drawings and Schematics of As-built Assemblies

The recovery system relied heavily on CAD modelling to ensure the tight dimensions of the components fit properly without excessive number of physical prototypes and corrections. To this end, exact models were rendered for all relevant components and were assembled as intended to visualize the design throughout the process. Figure 53 below shows the final structural assembly drawings of the CRAM v4 created in CREO 4.0.

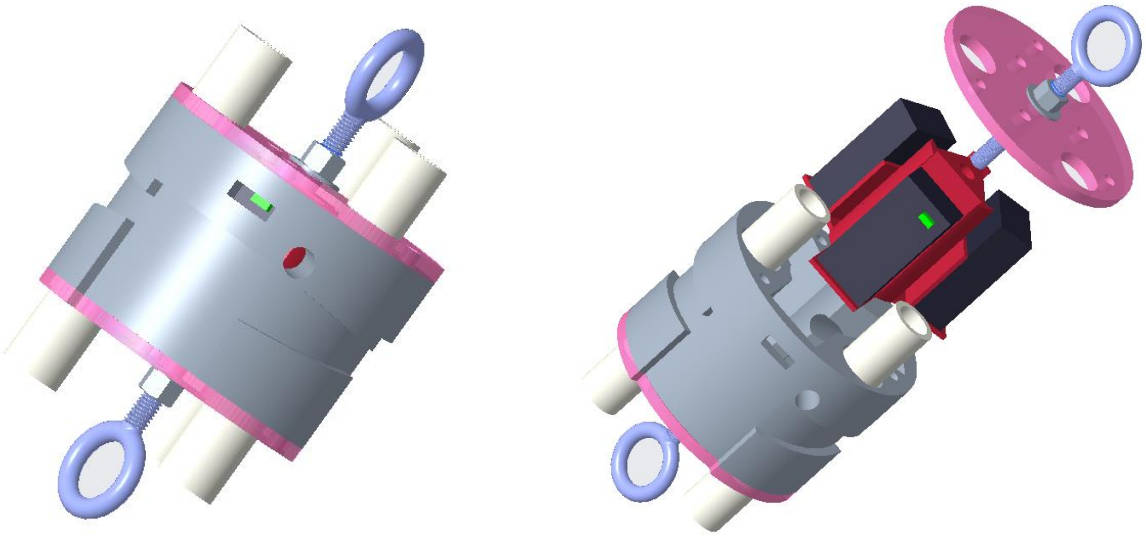


Figure 53. CRAM v4 assembled view (left) and exploded view (right).

The electrical layout for each redundant subsystem is identical. The following image, Figure 54, shows the wiring logic connecting the power source, altimeter, and e-matches. Ground wires are shown in black, voltage wires are shown in red. The battery box and altimeter are labelled as such. The twist-lock wire connector is represented as the yellow triangle, and the e-matches are shown as orange star shapes. As evidenced in the image, the three powered devices (altimeter and two e-matches) all share a common voltage from the battery. However, the altimeter alone receives the ground wire from the battery and then uses its output ports as effective ground leads to complete the circuit with the e-matches when necessary. The annotations on the altimeter indicate the relative positions of the output ports. "+" represents voltage, "G" represents ground, "A" represents apogee, and "M" represents main.

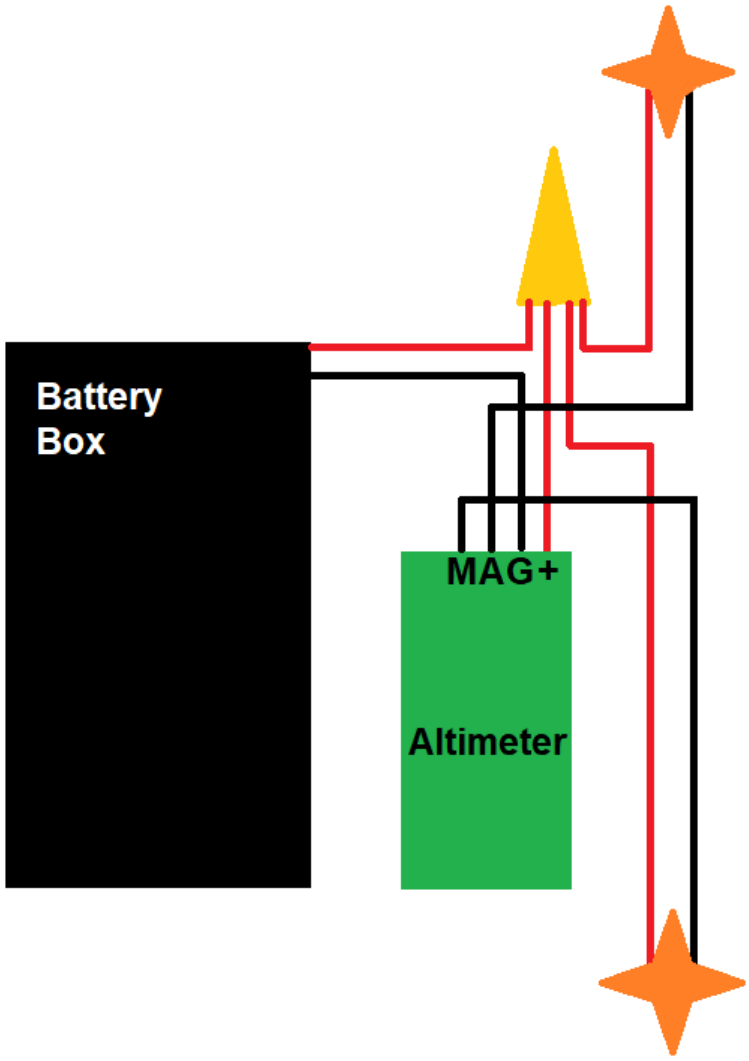


Figure 54. Recovery subsystem electrical schematic.

3.3.7 Rocket-locating Transmitters

The onboard GPS is located in the rover payload. For more information regarding the transmitters utilized, consult Section 4.1.3.2.3.

3.3.8 Recovery System Sensitivity

There are a variety of ways to ensure the recovery system does not receive interference from other onboard devices. The first of which is to limit the transmitting power of any potential interference sources to appropriate levels, which has been taken into account in selection of devices in the rover and air-braking payloads. Second, active shielding can be implemented to create a protective Faraday cage around the sensitive recovery avionics. This step has been accomplished and can be seen in the images of the CRAM and top bulkhead shown below in

Figure 55. The bottom surface of the CRAM has been coated with conductive copper tape to shield the avionics from interference from the air-braking payload below. Similarly, the top cover of the system has also been coating in the tape to prevent interference from the rover payload GPS situated above the recovery system.

Figure 55. Copper shielding above and beneath avionics.

3.3 Vehicle Performance

3.3.1 Performance Predictions

3.3.1.1 Flight Simulations

3.3.1.1.1 Derived Performance Prediction Program

The Notre Dame Rocketry Team has created a performance prediction program in Python to calculate the apogee, the maximum velocity, the center of pressure, and the stability of the rocket. The code used to do these calculations can be seen in Appendix A. The apogee is calculated using the rocket mass, the engine mass, the propellant mass, the air density, the drag coefficient during the thrusting phase, the drag coefficient during the first coasting phase, the drag coefficient during the air braking phase, the drag coefficient during the second coasting phase, the cross-sectional area, the thrust from the engine, the acceleration due to gravity, and the motor burnout motor time. The apogee is calculated from these variables by finding various coefficients, such as the burnout velocity coefficient and burnout velocity decay coefficient, at four distinct phases of flight: the thrusting phase, the first coasting phase, the air braking phase, and the second coasting phase. A relative height can be calculating for each of these phases based on their coefficients and finally these relative heights can be summed to find an approximate apogee. Variables such as wind speed and a constantly varying drag coefficient were not considered in this apogee approximation. Because of this, our performance prediction program served more as a verification of apogee approximations from OpenRocket rather than our actual expected apogee.

The coefficients of drag are especially important in this calculation. This is because there are four distinct average drag coefficients throughout our rocket's flight due to its changing speed and air brakes: the thrusting drag coefficient, the first coasting drag coefficient, the air braking drag coefficient, and the second coasting drag coefficient which happens after the air brakes retract. The drag coefficients were calculated using OpenRocket, subscale testing, and wind tunnel testing. Because each of these phases took an amount of time that could be understood as a fraction of the time of flight, the apogees from these phases were summed in a way that took into account each of their respective durations. In other words, to calculate the apogee, the apogee was calculating for each phase, dependent on the previous phase, and then finally summed. The apogee was calculated to be 5335 feet. This is only a rough estimation because the duration of the air braking phase was estimated using subscale flights.

The maximum velocity of the rocket was calculated simply by finding the burnout velocity which was conveniently a necessary variable in calculating the apogee of our rocket. Our calculation of the max velocity of our rocket (Mach 0.62) fairly accurately matched OpenRocket's estimation at Mach 0.58.

The center of pressure and subsequently, the stability, were also calculated in Python. This was done using the length of nose, the diameter at base of nose, the diameter at front of transition, the diameter at rear of transition, the length of transition, the length from the tip of nose to front of transition, the fin root chord, the fin tip chord length, the fin semi-span, the length of fin mid-chord line, the radius of body at aft end, the distance between fin root leading edge and fin tip, the leading edge parallel to body, the length of rocket minus length of fins and the number of fins. The stability calculated in this Python program was 3.4 which is approximately the same stability that OpenRocket calculated.

3.3.1.1.2 OpenRocket

In order to simulate the flight of the rocket, the rocket flight simulation program OpenRocket was used. After construction was completed on the rocket, masses and dimensions of its various parts were measured for their final, most precise values, and were used to create a final simulation model for the rocket. The resulting model's flight was then simulated for varying wind speeds and the approximate launch conditions that the rocket might experience. The results of the simulations are shown in Table 32 below.

Table 32. OpenRocket Predictions using the Cesaroni L1395.

Wind Speed	0 mph	5 mph	10 mph	15 mph	20 mph
Apogee (ft)	5415	5404	5372	5333	5268

Off Rail Velocity (ft/s)	63.5	63.5	63.5	63.5	63.5
Maximum Velocity (ft/s)	647	647	647	646	644
Maximum Liftoff Acceleration (ft/s^2)	245	245	245	245	245
Time to Apogee (s)	18.6	18.6	18.6	18.5	18.4
Flight Time (s)	196	196	195	194	195
CG Location from Nose (in)	79.99	79.99	79.99	79.99	79.99
CP Location from Nose (in)	102	102	102	102	102
Stability Margin with Motor	2.86	2.86	2.86	2.86	2.86

The center of pressure changed due to changes in the length of the rocket fins, as discussed in Section 2.1.1.2. Additionally, the finish of the rocket (e.g. smooth, rough, etc.) was changed from the previous simulation model used. Previously, the finish had been approximated as smooth, however, this was not accurate for the phenolic paper, unpolished plastic of the cone, the unfinished plywood of the fins, and the molded surface of the fiberglass rover payload bay and transition section. As a result, the finishes of each section were changed to most accurately model the conditions of each portion of the rocket. The flight model used for the simulations that yielded the values displayed above had the rocket using a Cesaroni L1395, as this is the motor that will be used during competition launch. However, the motor used for the test launch was a Cesaroni L1115, which, though similar to the Cesaroni L1395, has some differences. OpenRocket simulations were run for this motor, and compared to the test flight apogee in order to verify the prediction described above in Section 3.3.2.2.

3.3.1.1.3 RockSim

In order to simulate the flight of the rocket, while acting as verification of the predictions made by OpenRocket, and described in Section 3.3.1.1.2, the rocket flight simulation program RockSim was used. Like with the OpenRocket simulation model, the RockSim model was updated following construction so that masses and dimensions were correct and as accurate to their final values as possible. Simulations for various wind speeds were then run in order to get a range of predictions for possible launch conditions. The results of the simulations are shown in Table 33 below.

Table 33. RockSim Predictions using the Cesaroni L1395.

Wind Speed	0 mph	5 mph	10 mph	15 mph	20 mph
Apogee (ft)	5508	5494	5454	5385	5289
Off Rail Velocity (ft/s)	62.8	62.8	62.8	62.8	62.8
Maximum Velocity (ft/s)	650.4	650.3	649.8	649.1	648.1
Maximum Liftoff Acceleration (ft/s^2)	279.8	279.8	280.5	280.5	280.5
Time to Apogee (s)	18.76	18.73	18.6	18.54	18.37
Flight Time (s)	171.2	189.7	272.1	239.3	224.3
CG Location from Nose (in)	79.87	79.87	79.87	79.87	79.87
CP Location from Nose (in)	102.1	102.1	102.1	102.1	102.1
Stability Margin with Motor	3.01	3.01	3.01	3.01	3.01

The results in the table above are close enough to those made in OpenRocket that they indicate that it is safe to have confidence in the relative predictions made by both simulation programs. Differences in simulation results can be attributed to differing definitions of each program for flight events such as flight time, rail clearance, finish qualities, and motor attributes to name a few. Additionally, the way that the rocket was modelled in each program differs due to the modelling features available. The RockSim flight model also used the Cesaroni L1395 for the simulations that yielded the values displayed above because this is the motor that will be used during competition launch.

3.3.1.2 Stability

In order to ensure a safe flight, the launch vehicle must have a positive stability margin of at least 2 calipers upon clearing the launch rail and maintain this margin until reaching apogee. A positive stability margin of 2 means that the center of gravity (CG) must be located fore of the center of pressure (CP) by at least 2 body tube diameters. According to calculations done in OpenRocket, the CG is to be located 80 inches from the nose cone of the launch vehicle. Upon final assembly prior to the test launch, the center of gravity was calculated by means of balancing to be 79.5 inches from the nose cone. The CP of the launch vehicle was simulated in

both OpenRocket and RockSim to be 102 inches from the nose cone, giving the launch vehicle a static stability margin of 2.86 calipers with the motor.

Using a 12 foot rail, rail exit is determined when the foremost rail button leaves the rail guide at an altitude of 9.5 feet. Simulations in both OpenRocket and RockSim both verify that a margin of at least 2 calipers is achieved in 0 to 20 mph wind conditions, which are all conditions that may be experienced at the time of launch. Overall stability was further verified during the test launch where the rocket maintained a strictly vertical trajectory and did not display instability after clearing the rail, nor did it experience weather cocking to suggest over stability.

3.3.1.4 Validity of Predictions

3.3.1.4.1 Wind Tunnel Testing

On November 9th, 2017, both the Vehicle and Air Braking System sub-teams were able to perform wind tunnel testing in the Hessert Laboratory for Aerospace Research. By this time, construction of the subscale rocket was completed, and its design was intended not only for launch, but for testing in the wind tunnel. The characteristics of the subscale rocket can be found in Section 3.3.1.4.2.

The purpose of wind tunnel testing was to verify the drag predictions from OpenRocket and Rocksim. Additionally, the team wished to measure the impact of fully extended Air Braking Tabs on the drag coefficient of the rocket.

To perform the test, a force balance was set up in a wind tunnel big enough to fit the rocket. Data could be collected with a drag transducer. The setup for the test can be found in Figure 56.

Figure 56. Wind tunnel test setup.

Once testing was completed, the data from the transducers was inputted into Matlab in order to yield useful information. However, upon analysis of the data using code created by the team, it was revealed that the pressure transducer to gage wind speed in the tunnel had been faulty, and all of the data for this tool was corrupted.

However, using the calibration found in Figure 57, force data for the model was able to be found. The results of the wind tunnel testing can be found in Figures 58 and 59.

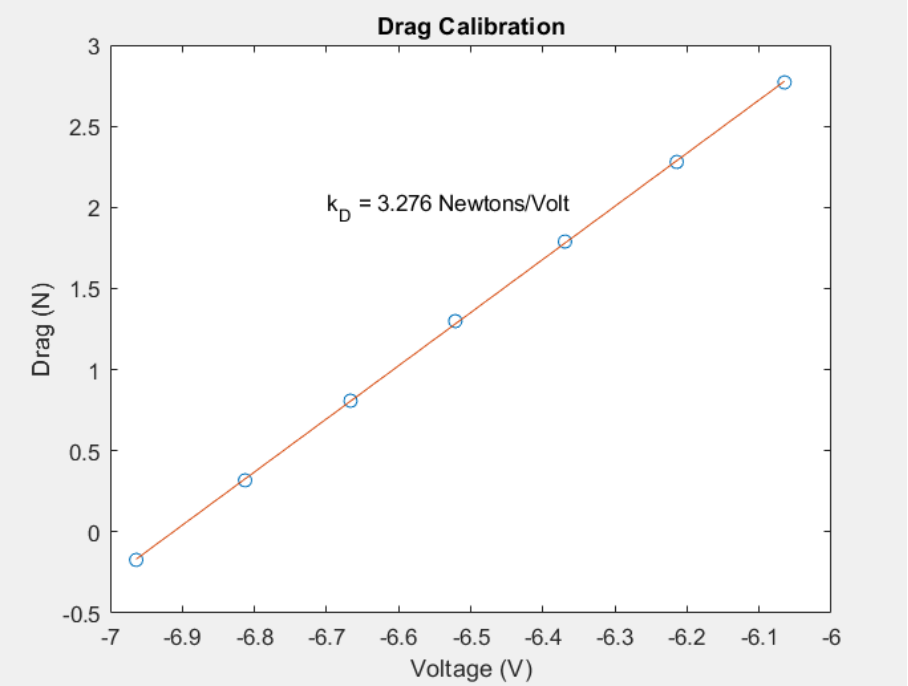


Figure 57. Drag calibration for the force balance.

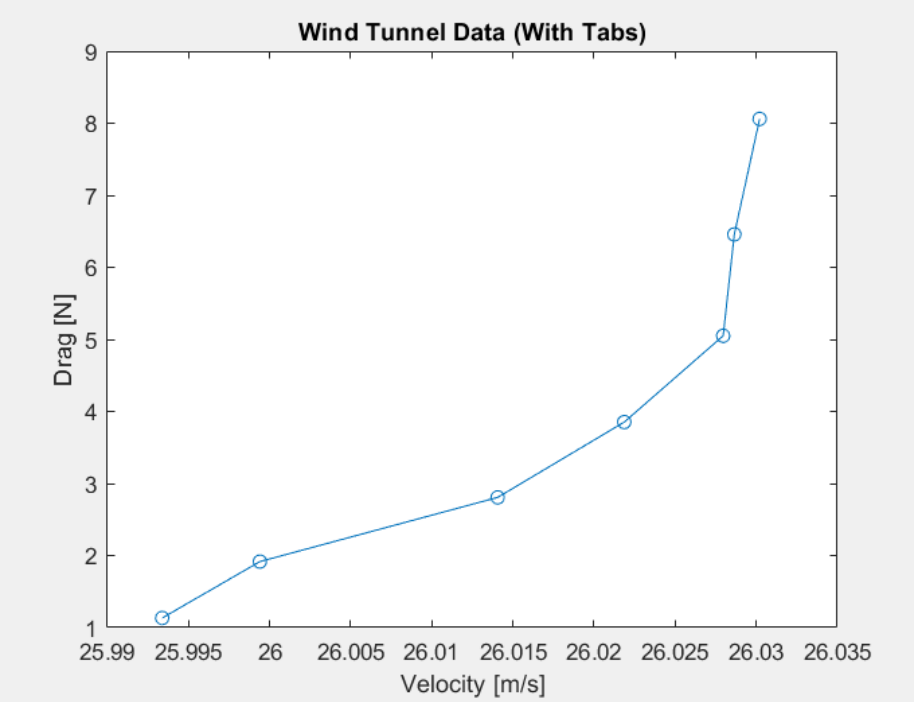


Figure 58. Drag forces on rocket with tabs deployed.

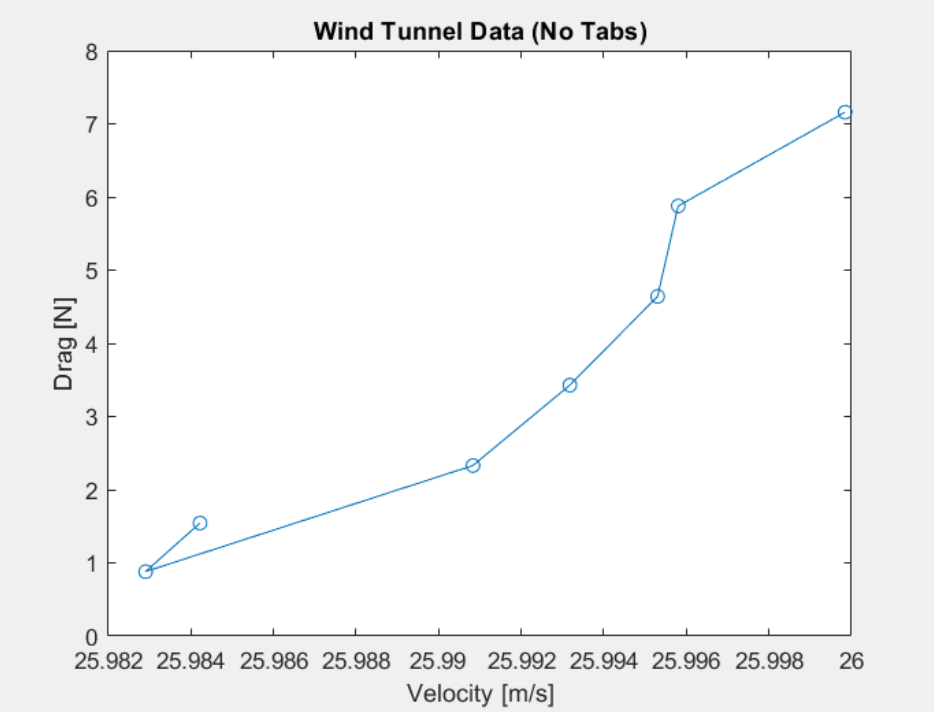


Figure 59. Drag forces on rocket with no tabs.

While the curves found do not indicate a true relationship between drag and fluid velocity, it does indicate the forces that the subscale rocket would experience during flight, as the tunnel velocity was incremented from 0 m/s to approximately 50 m/s. Assuming the largest drag value can be attributed to the largest wind speed, it can be seen in the Figures above that the rocket experiences no more than 9 Newtons of drag at the maximum fluid velocity. However, without velocity data, it is unreasonable to assume anything about the aerodynamics of the rocket based on these wind tunnel tests.

3.3.1.4.2 Subscale Launch

In order to verify that the full scale rocket will be stable as well as test the altimeters being used for the full scale, a subscale rocket was build and launched. The main goal of this rocket was to not only verify stability and the altimeters, but to compare altimeter data to that of OpenRocket and Rocksim. Doing this allows the team to gage how much it can rely on the simulation software for the prediction of flight data for the full scale rocket.

Sub Scale Rocket Scaling and Materials

The subscale rocket was built to a 40% scale of the final design by axial length and body tube diameters. This scaling was due to the existing supply of body tubes available to the team, and because of this certain components were not scaled to exactly 40%.

The subscale's aerodynamic structure is very similar to that of full scale, with the exception of the nose cone. The axial length and diameter of both the forward, aft, and transition

sections of the rocket all scale correctly to 40%, however, the nose cone was commercially bought, and does not scale to exactly 40%. This complication was taken into account in the simulations and was determined not to be a critical issue for the rocket.

The subscale's internal structure, however, was much different than the full scale. This was mainly due to the fact that the Deployable Rover Payload was not simulated in the vehicle. Therefore, to compensate for the lost weight of the payload, the avionics were placed in the forward section of the rocket where the Deployable Rover Payload would be. Additionally, the CRAM recovery system was not needed, since the motor used in the launch had an ejection charge built in. Finally, the Air Braking System was simulated by an interchangeable coupler, one with a length of phenolic body tube and another with a 3-D printed section with air braking tabs scaled to 40%. There was no internal structure to the Air Braking System for two reasons. It was not needed to control the extension of the tabs, and it would interfere with the deployment of the parachutes upon ejection charge ignition.

Another major difference between the subscale and the full scale was the choice of materials. For the subscale, due to limited time, funds, and workshop access, phenolic was used for the main body tube structure and couplers. Birch plywood was used for the bulkheads, centering rings, and fins. And finally, Bristol paper was used to craft the outer structure of the transition section. A summary of the material properties can be found in Section 3.1.6.1.

Subscale Characteristics

A summary of the subscale rocket's dimensions can be found in Table 34. As stated above, some of these dimensions are not exactly 40% of the full scale size due to constraints such as material availability and mission requirements.

Table 34. Subscale rocket dimensions.

Property	Dimension
Length of Rocket (in)	52.25
Fore Outer Diameter of Rocket (in)	3.14
Aft Outer Diameter of Rocket (in)	2.27
Transition Length (in)	1.6
Number of Fins	4
Fin Root Chord (in)	2.8
Fin Tip Chord (in)	2.8

Fin Sweep Angle (°)	31.6
Fin Height (in)	1.77
CG Position from Nose Cone (with motor) (in)	32.48
Weight without Motor (lbs)	2.23
Weight with Motor (lbs)	2.51
Estimated Stability Margin without Motor	3.44
Estimated Stability Margin with Motor	2.74

The motor used for the subscale rocket was chosen due to the fact that it has a relatively similar thrust curve to the full scale's motor, the L1395. The Aerotech G78-7G was chosen as the propulsion for the subscale rocket, and its thrust curve can be found in Figure 60. This motor has a 7 second delay deployment charge built in, which eliminates the need for the CRAM recovery system. The characteristics of this motor can be found in Table 35.

Table 35. Characteristics of the G78-7G.

Motor Classification	**AeroTech G78-7G**
Diameter (in)	1.14
Length (in)	4.88
Average Thrust (lbf)	17.96
Maximum Thrust (lbf)	22.91
Total Impulse (lbf*s)	24.70
Burn Time (s)	1.4
Total Weight (lb)	0.28

Propellant Weight (lb)	0.13

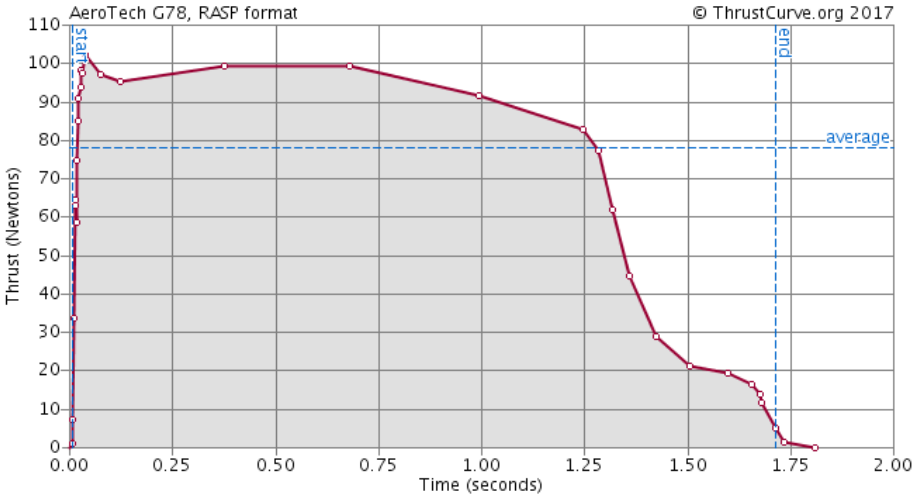


Figure 60. Thrust Curve of the Aerotech G78-7G.

Finally, an image of the fully assembled rocket can be found in Figure 61.

Figure 61. Subscale rocket fully assembled.

Subscale Results

The subscale launch took place on Saturday, December 2nd, 2017 in Three Oaks, Michigan. Two flights were carried out, one with the control body tube coupler and another with the simulated drag tabs. The launch conditions for this day can be found in Table 36.

Table 36. Launch Day Conditions in Three Oaks, MI (12/02/17).

Condition	Dimension
Temperature (°F)	42
Wind Speed (mph)	10
Pressure (inHg)	30.13

Latitude (°N)	41.799
Longitude (°W)	86.611
Altitude (ft)	692

Using these conditions, the subscale rocket was modeled in OpenRocket and Rocksim. The model was made for only the control flight, since neither of these software packages are able to physically model the drag tabs. Additionally, they are also able to accurately model the rail buttons made for the rocket, and therefore these were not included either. These results for the simulated flights, as well as the actual subscale launch, can be found in Table 37.

Table 37. Subscale predictions and results.

Source	OpenRocket	Rocksim	Subscale Control	Subscale With Tabs
Apogee (ft)	976	989	919	858
Off-rail Velocity (ft/s)	63	58	60	--
Flight Time (s)	211	218	54	--
Time to Apogee (s)	7.74	8.01	8.71	--

As seen in the table above, the predictive software overestimated the apogee by approximately 60 ft in OpenRocket, and 70 ft in Rocksim. This overestimation suggests that the software packages do not correctly estimate the drag on the rocket, which can be attributed to the absence of rail buttons on both models. The off-rail velocity and time to apogee were estimated fairly accurately, and the flight time is irrelevant for these purposes.

The subscale launch was a success from the standpoint that the team knows that the rocket design is aerodynamically stable. This verifies the previous CFD analysis on the variable diameter design, and allows the team to move forward with this for the full scale vehicle. Additionally, the test flight with tabs deployed verifies that the Air Braking System does indeed have a significant impact on the aerodynamic drag of the rocket. The altimeter data from the launch will be analyzed further in order to create an accurate altitude control model for the full scale system.

However, the subscale launch also reveals to the team that the predictions in OpenRocket and Rocksim are not to be taken as exact. The overestimation by both programs is taken into account

for the full scale models, and as of now both programs will continue to be used as sources of simulation data.

3.3.2 Full Scale Flight Results

The full scale test launch occurred on March 3rd, 2018 in Three Oaks, Michigan. This is the same location at which the subscale test occurred, and is the location where the team has launched in the past for tests. The launch day conditions can be found in Table 38. Before the launch, multiple tests were carried out, including shake tests of the assembled rocket and black powder tests for the recovery and deployable rover systems.

For the test launch, the rocket was flown on the Cesaroni L1115 motor, the properties of which can be found in Section 3.1.5.1. This motor was chosen due to constraints forced upon the team by suppliers. Two L1115 motors and two L1395 motors were purchased from Animal Motor Works for the tests and the competition. Since two motors are needed for testing (a control and an experimental for the Air Braking System), and since the future availability of motors was uncertain, it was decided that the tests would be run on the L1115 and the completion would be on the L1395. The decision for the motor used at competition was determined using the simulations run in OpenRocket and RockSim.

The rocket was assembled using the launch procedures found in Section 6.1and launched from a pad supplied by the Michiana Rocketry Club. The flight was stable and there were no visible signs of structural failure upon launch. Upon reaching apogee, the black powder charges for the drogue parachute ignited and deployed the smaller of the parachutes. However, this charge was powerful enough to also deploy the main parachute, as well as break the shear pins attaching the nose cone to the fore fiberglass body tube. The nose cone proceeded to fall from the achieved apogee of 5765 feet, and the connected sections of the rocket descended, drifting approximately 0.9 miles in the process. The nose cone was damaged from the fall, but the remaining sections of the rocket were recovered intact.

Because of this damage to the nose cone, the rocket was not able to fly twice. Two flights were planned, one control flight and one flight with the Air Braking System and rover activated, however, only the control flight was carried out.

Table 38. Conditions in Three Oaks, Michigan on March 3rd, 2018.

Condition	Dimension
Temperature (°F)	41
Wind Speed (mph)	2

Pressure (inHg)	30.51
Latitude (°N)	41.799
Longitude (°W)	86.611
Altitude (ft)	692

3.3.2.1 Stability Verification

In order to verify that the rocket was stable before being put on the pad, the locations of the CG and CP were drawn on the rocket with a pen. These markings can be seen in Figure 62. The CP was calculated using the software packages available to the team, OpenRocket and RockSim. This was determined to be 102 inches from the nose of the rocket, which is 34 inches from the rear. The CG was calculated in the same software to be 79.28 inches from the tip of the nose cone or 56.72 inches from the rear of the rocket.

To verify the CG, the fully loaded rocket was balanced on a stand made to hold the rocket from a singular point along its structure. When the rocket was able to balance on this whit no interference from the team, the CG was marked. This measured value was 57.7 inches from the rear of the rocket. This difference of 1 inch was attributed to the non-uniform packing of the parachutes and shock cords in the recovery tube, and was determined not to be a critical issue for the flight.

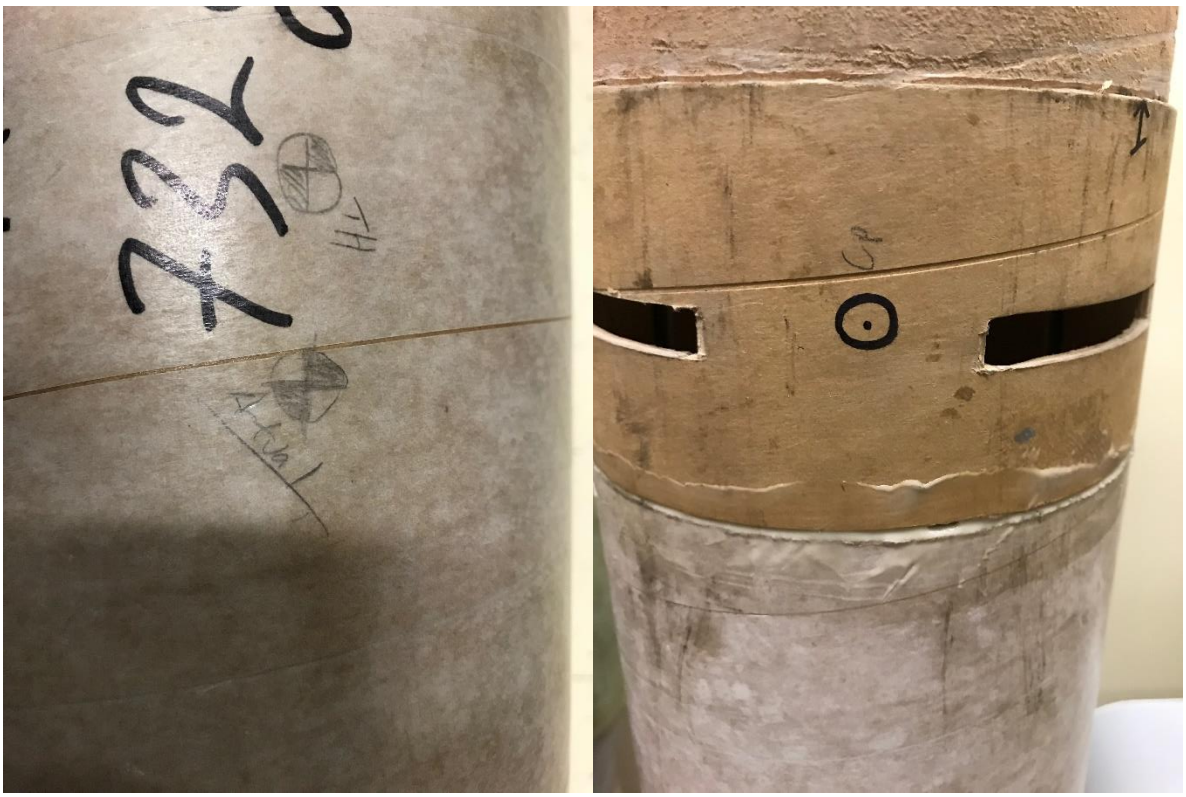


Figure 62. Location of CP, as well as the relative position of the theoretical and actual CG on the rocket.

3.3.2.2 Full Scale Test Flight Results

As stated above, the rocket launched one time during this test day, due to a malfunction in the recovery system. However, altimeter data was recorded so that it could be compared to the data gathered from simulations. Below in Figure 63 are the altitude, velocity, and acceleration plots from the altimeters for the test flight. Table 39 shows the recorded altimeter flight events of interest versus those in the simulation data. As seen in the table, both simulations were able to predict the altitude within 20 feet, as well as the time to apogee, maximum acceleration, and maximum velocity within 0.5 seconds, 50 ft/s^2, and 15 ft/s, respectively. This difference was attributed to the slight difference in actual and theoretical CG, as this impacted stability slightly and therefore could have changed the flight profile. However, these results give the team confidence that the simulation programs are able to accurately predict the flight of the rocket.

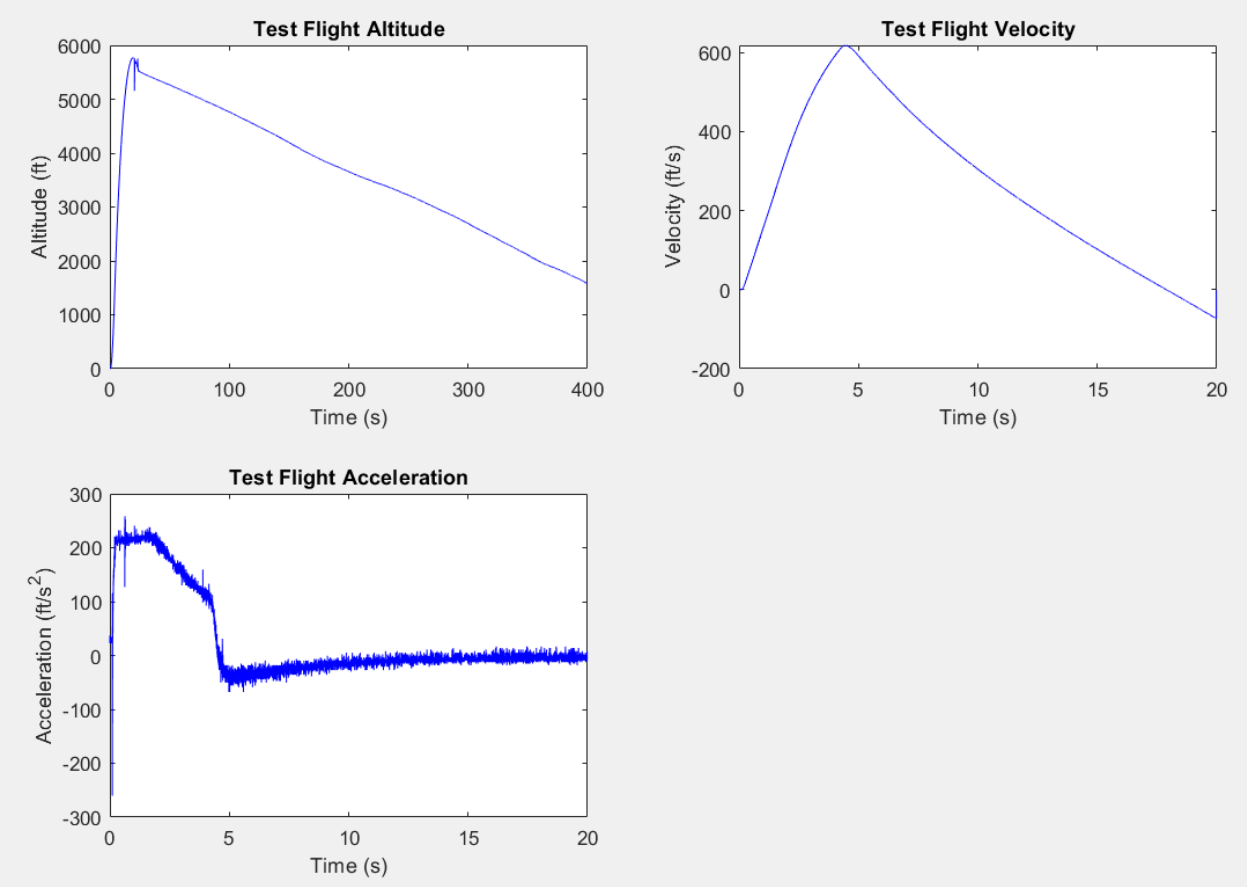


Figure 63. Test Flight Altimeter Data.

Table 39. Test Flight Data against Simulation Data.

Characteristic	OpenRocket	RockSim	Actual
Apogee (ft)	5754	5747.3	5766
Off Rail Velocity (ft/s)	71.8	70.34	78.89
Maximum Velocity (ft/s)	631	626.35	617.83
Maximum Liftoff Acceleration (ft/s^2)	241	207.1	260.8
Time to Apogee (s)	19.4	19.48	19.95
Flight Time (s)	206	211.5	540

CG Location from Nose (in)	79.281	78.90	78.3
CP Location from Nose (in)	102	102.12	102
Stability Margin with Motor	2.95	3.00	3.06

When these results are compared to those gathered in December with the launch of the subscale rocket, it can be shown that the team is able to accurately predict the flight of the rocket with OpenRocket and RockSim within ~70 feet. The larger inaccuracies of the subscale flight predictions can be attributed to inaccuracies in the weighing and placement of vehicle components in the models during construction. The slight miscalculations from the subscale were corrected, and greater care was taken during construction of the full scale rocket to ensure that every part was weighed and its final position in the rocket measured accurately.

4 Payload Criteria

4.1 Deployable Rover Payload

4.1.1 Objectives

The Deployable Rover Payload is required to deploy a rover contained within the rocket for the duration of the flight. Upon deploying the rover, it will autonomously move five feet away from the rocket and unfold two sets of solar panels. Radio frequency will be used to activate the rover after blowing off the nose cone. Ejection charges of black powder will remove the nose cone after landing to allow the rover to exit the rocket cleanly. Tracks have been placed on both top and bottom of the rover to allow it to drive out whether the rocket lands "upside-down", or "right side up". A LiDAR (Light Detection and Ranging) sensor will be used for object avoidance so the rover can safely move 5 feet away from the rocket in order to deploy the solar cells. The solar cells will be attached to a fireproof cloth and a metal frame to allow for ease of deployment. The rover will be cut from HDPE to allow for customization.

4.1.2 Success Criteria

The payload will be considered successful if all of the following criteria are met:

1. The rover autonomously drives five feet away from the rocket
2. The solar panels unfold and provide power to the rover
3. The rover will be ready to be used again on the same day

4.1.3 System Design

4.1.3.1 Structural Elements

4.1.3.1.1 Body

The body is made out of a machined block of High Density Polyethylene (HDPE). It was machined using a techno router to fit the specific requirements and dimensions of the rover. Once milled, all components fit onto the body in their correct positions; securely held into place on the body with nylon screws. Upon examination and testing of the rover it was found that the body is strong enough to withstand multiple launches without any critical failures.

4.1.3.1.2 Wheels and Motors

In order for the rover to be mobile it needed an electric motor powertrain. For simplicity, each wheel is powered individually using a brushed hobby motor. Further analysis of the initial motor selection showed that with a gear ratio of 1:1 they would stall out too easily, so a gearbox would be required. To account for this requirement, four LEGO Power Functions XL motors were purchased instead. External length dimensions were very similar to the initial design but included an integrated gearbox. The external dimensions are 2.36 inches in diameter by 1.89 inches in length. The previous motors were .944 inches in diameter and 1.38 inches in length. While not as fast, 220 rpm unloaded, the new motors provide significantly more torque, 90.4mNm at 600mA, which is ultimately more important to the rover's performance. This extra performance comes at increase in weight of 0.123lbf (56g) since each motor weighs 0.15lbf (69g) instead of 0.12lbf (55g). In addition, the change simplified the electronics of the motor driver since Lego sells the appropriate modules.

The original choice of 1.77 inches in diameter hobby truck wheels were able to be interfaced with the Lego motors through custom hubs 3D printed from PLA material. These allow the Lego axle to securely connect the motor to the wheels, and the connection was reinforced with epoxy to ensure the press-fit would not come loose. These custom hubs can be seen in Figure 65.

Figure 64. The LEGO Power Functions XL motor contains an internal gearbox that provides greater torque than the initial brushless motors.

Figure 65. Custom 3D printed hubs provide a secure connection between the wheels and motors.

4.1.3.1.3 Servo and Solar Panels

One of the primary tasks the rover must complete is unfolding solar panels once the rover exits the body tube. To accomplish this, the rover is equipped with a rack-and-pinion system driven by a Hitech HS-7245MH servomotor, modified for continuous motion. The racks are short enough to allow the rover to drive out of the body tube, but are then autonomously extended once it exits. Mounted on the racks is a folded solar array which, when the racks are extended, unfolds to triple its surface area. The array is made from polycrystalline silicon solar panels, cut to size, and connected with fire retardant fabric to allow folding. This solar array is modeled by Figure 66. The panels are fixed to both the top and reverse of the array so that the rover can still receive measurable power in either orientation. The body of the rover also has cutouts to allow for maximum solar exposure in the inverted position.

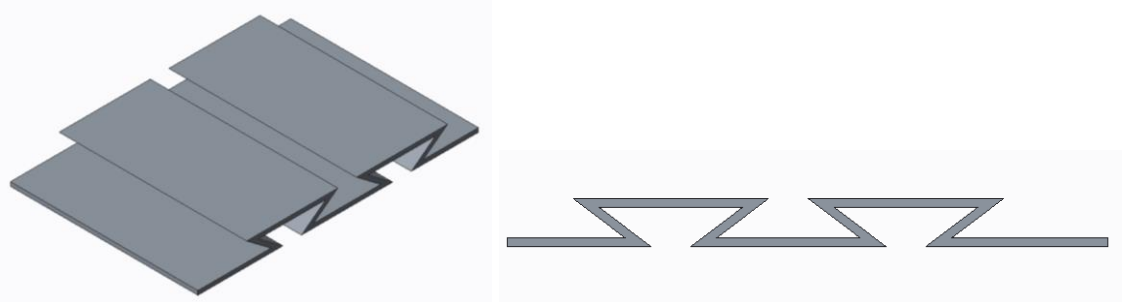


Figure 66. CAD Diagram of the folded solar array.

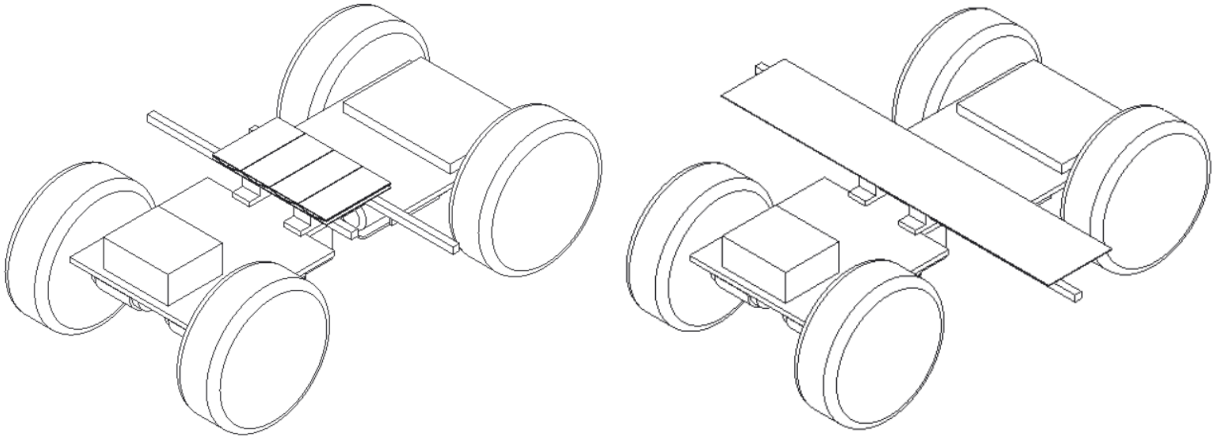


Figure 67. CAD model of the rover in both the folded and extended solar panel states.

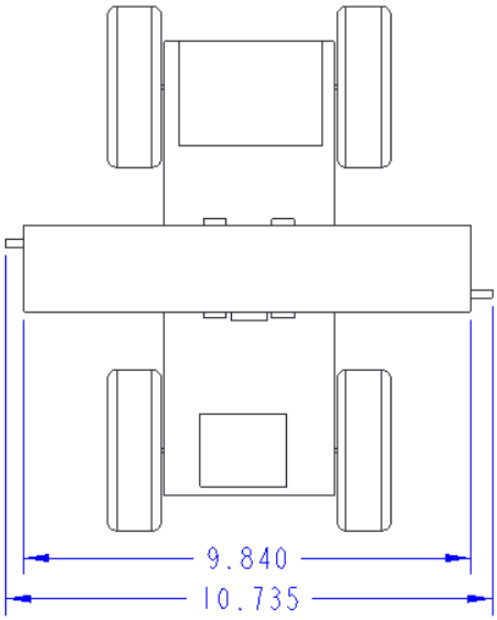


Figure 68. Dimensions of the solar panel array once fully extended. Units are in inches.

4.1.3.1.4 Securing System

Per NASA requirements, the rover must be secured within the internal structure of the rocket. This must be done in such a way as to prevent damage to both the rover and the rocket itself during flight. To achieve limited motion of the rover within the moving frame of the rocket, the solar panel deployment system is used. The brass racks upon which the solar panels rest must be retracted when the rover is within the rocket for the rover to fit inside. When the servomotor runs counterclockwise, these racks can be retracted slightly further to extend the non-panel-bearing ends of the racks into mounting blocks affixed to the interior wall of the fuselage. These two mounting blocks are 3D-printed panel-like structures that form to the inner radius of the fuselage wall. They are affixed to the interior wall of the fuselage with high-strength epoxy. This configuration can be seen in Figure 69 and the constructed system is seen in Figure 70.

Slots for the ends of the racks to fit into are cut into the sides of each mounting block. The insertion of the racks' ends into the blocks' slots fully secure the rover from any lateral or longitudinal motion along the x- and z- axes. The rover is further secured within the internal structure of the rocket by the two sets of tracks that it rests on, which prevent translation in the y-direction and rotation about the z-axis. While the rocket is positioned upright, either on the launch pad or during the powered portion of flight, the wooden bulkhead epoxied into the back of the fuselage helps reduce the strain on the ends of the brass racks. The wooden ramps positioned between the front of the rover and the first bulkhead of the nose cone section serve the same purpose for any nose-down orientation of the rocket.

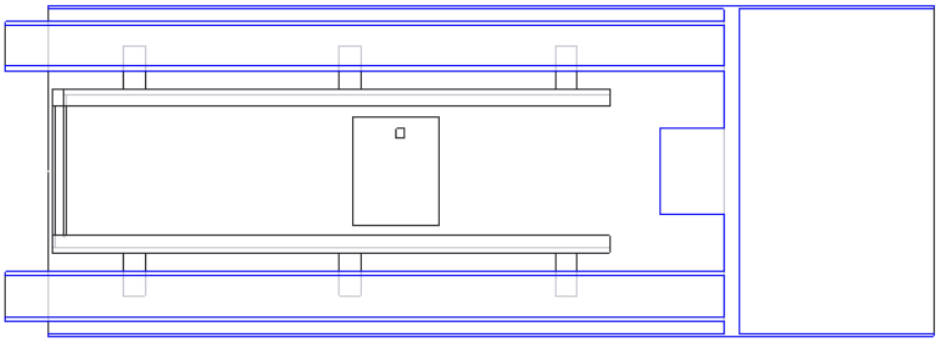


Figure 69. Engineering drawing of the internal locking system.

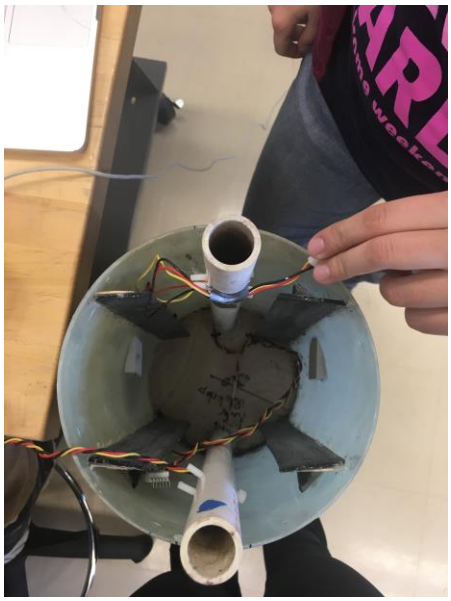


Figure 70. Completed construction of the securing system.

4.1.3.1.5 Deployment System

After the rocket safely lands, the ground station sends a signal to catalyze a black powder reaction designed to eject the nose cone from the body tube. These black powder charges are housed in two long PVC pipes that run above and below the rover and are mounted on a bulkhead at the back of the payload bay. The electronics receiving this signal to ignite the matches are mounted on the rover. This is a slight change to ensure that once the rover is out of the body tube there is no chance of a misfire if the second charge fails to go off. Initially these were mounted on the bulkhead at the base of the payload bay. Each PVC pipe contains one gram of black powder, the second pipe being redundant to the first. The previous estimate for the required mass of black powder was 3-5 grams, but it was determined that only one gram would be needed. Both charges are designed to detonate even if the nose cone is successfully removed with only the first charge. The black powder is lit via electronic matches that run along the PVC

pipes, which ignites at the front (top) of the black powder section when they receive the signal from the ground station. The PVC pipes intersect a bulkhead at the base of the nose cone and end before reaching a second bulkhead located further into the nose cone. Designing the PVC pipes to end in a void in between two bulkheads within the nose cone ensured that the contents of the body tube were not damaged by the explosive reaction. As soon as the ejection of the nose cone is complete, the rover drives out of the payload bay by first retracting the racks from the securing mount. In the payload bay the rover is positioned in between two sets of tracks parallel to the roll axis to account for either landing orientation. It drives out from these tracks onto spring loaded ramps, which unfold as the nose cone is removed, extending from the rails to the ground.

4.1.3.2 Electrical Elements

4.1.3.2.1 Electrical Schematic and Board Diagram

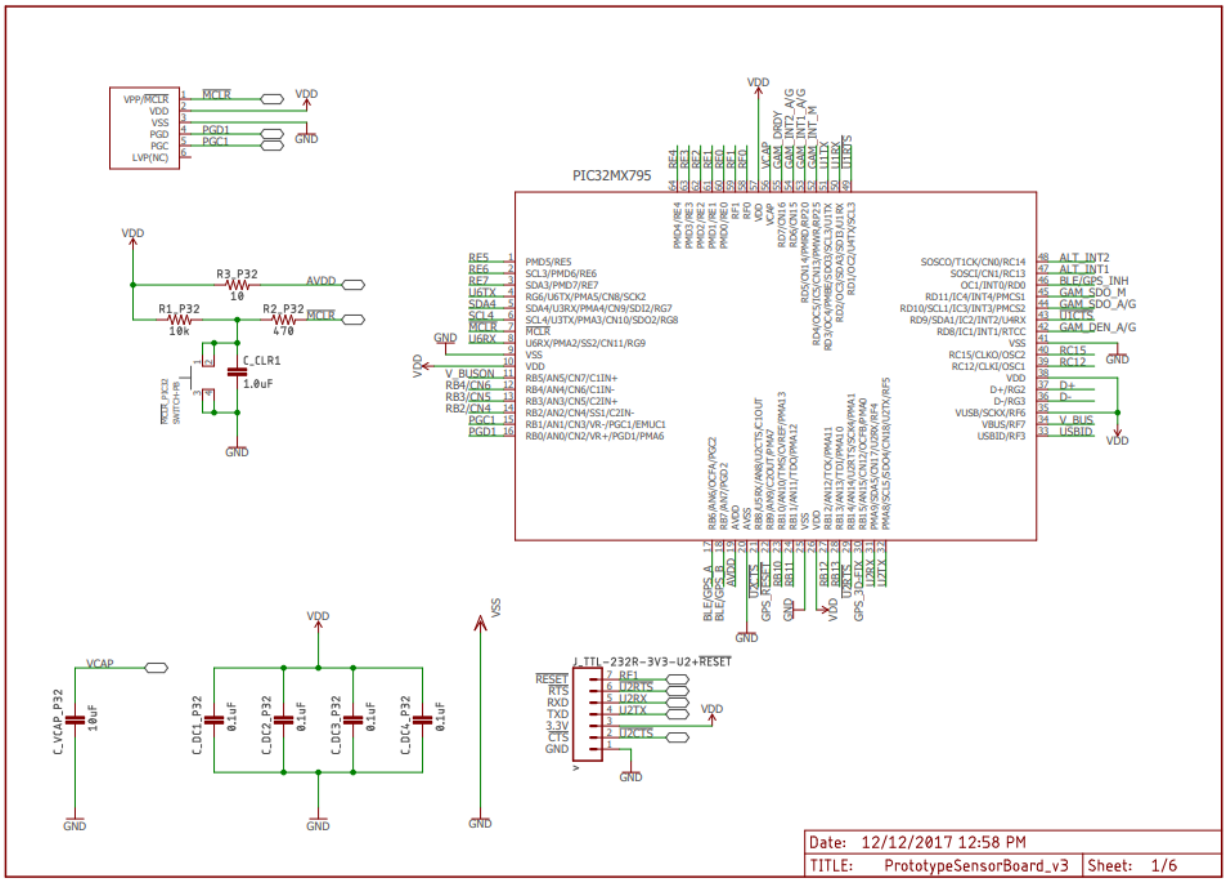


Figure 71. PIC32 Board.

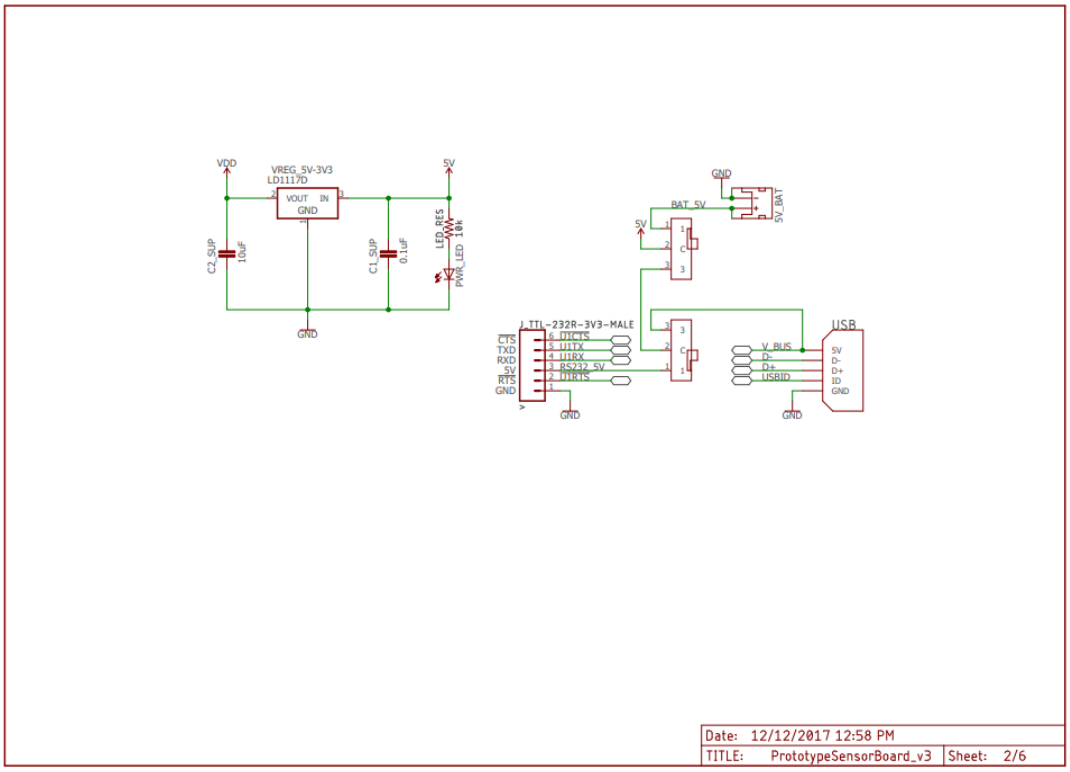


Figure 72. Power Supply System.

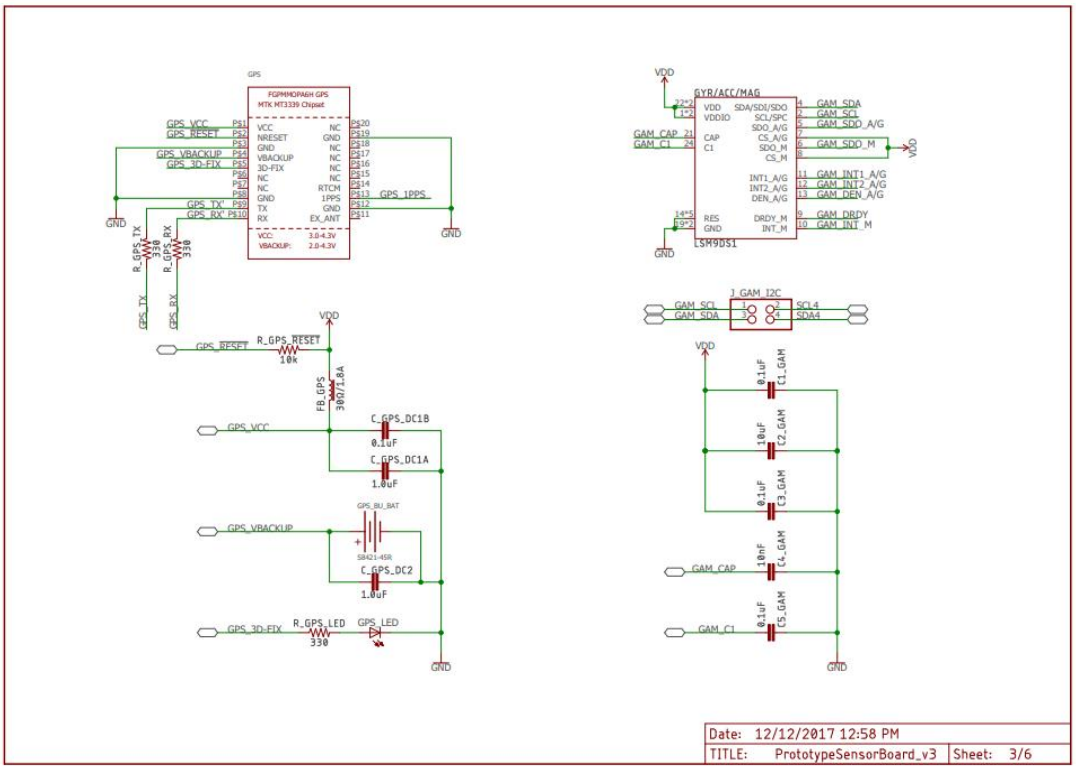


Figure 73. Gyroscope/Accelerometer/Magnetometer and GPS.

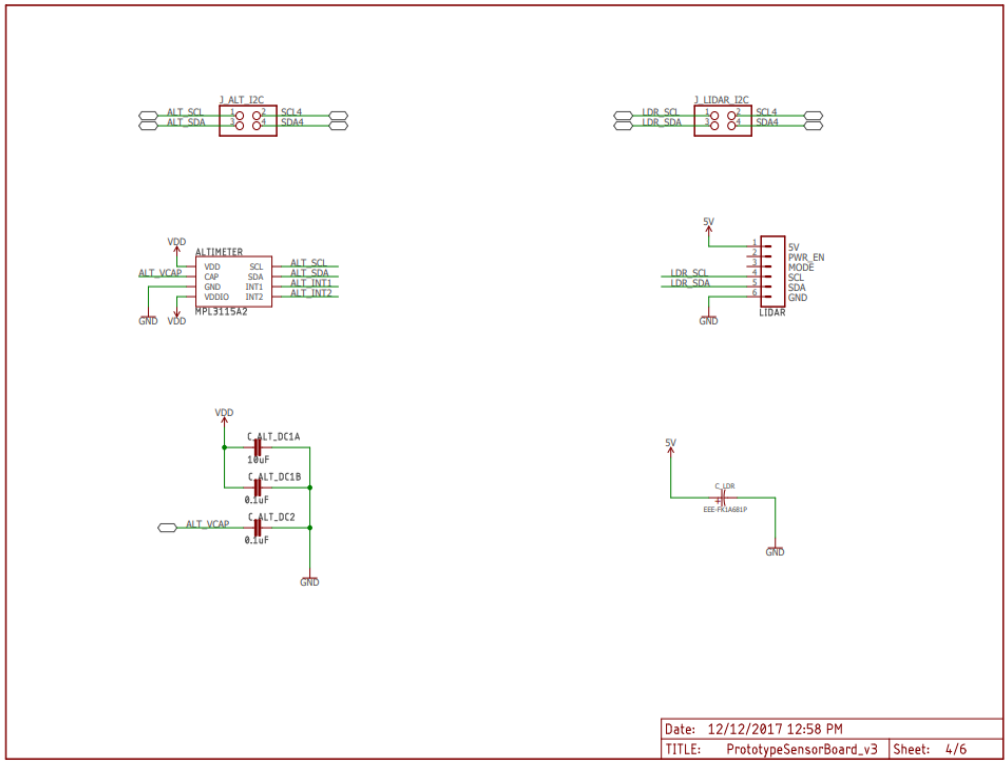


Figure 74. Altimeter and LIDAR modules.

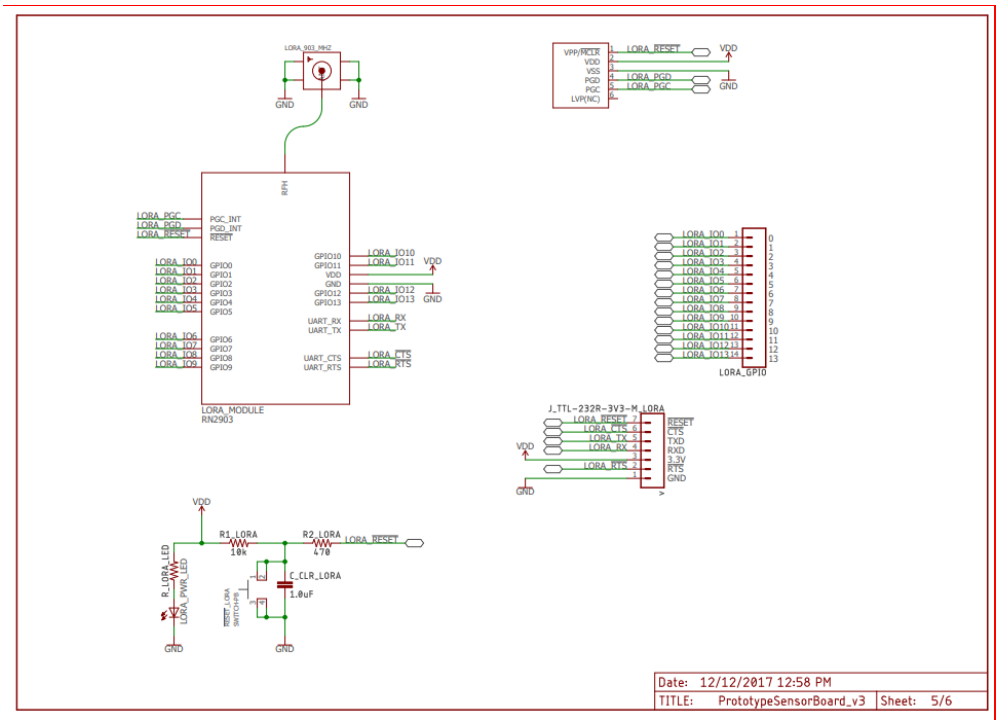


Figure 75. LoRa Module.

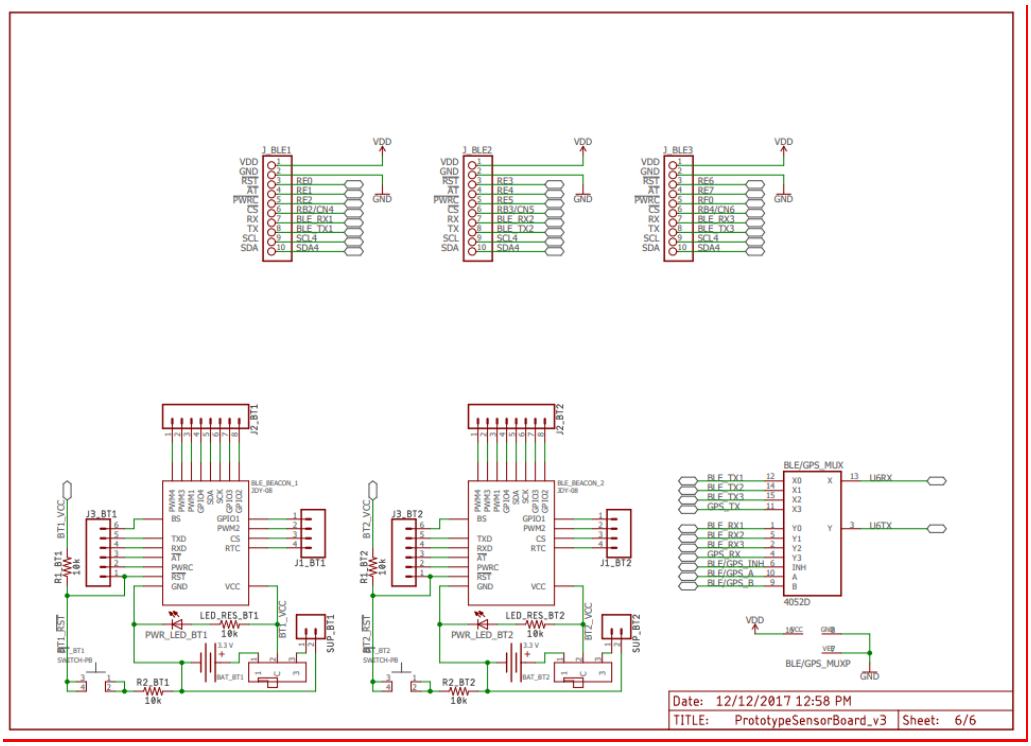


Figure 76. Bluetooth Modules.

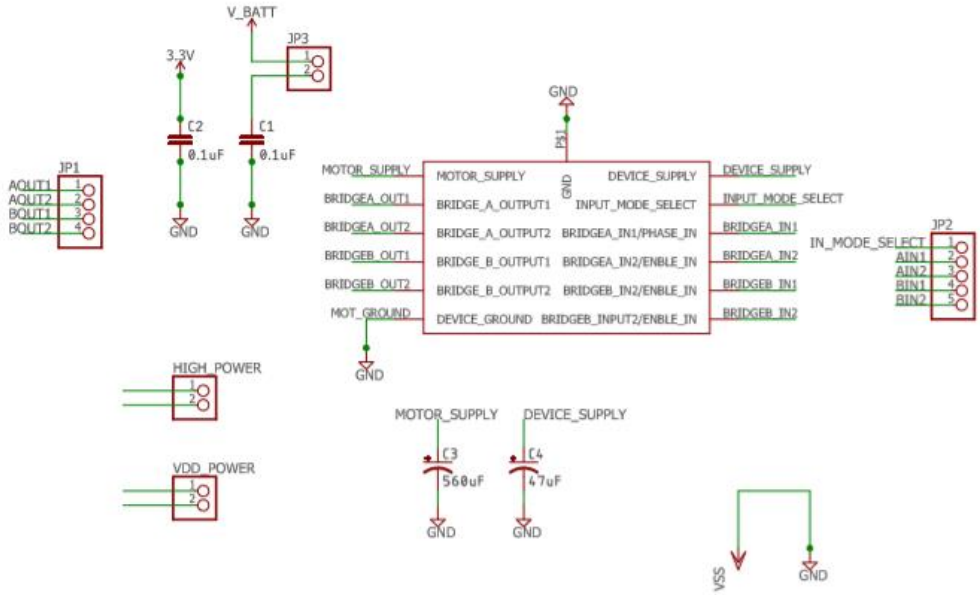


Figure 77. Motor Driver Board.

4.1.3.2.2 Electronic Control System

The electrical control system consists of a PIC32MX395 microcontroller which talks to a LoRa module to receive commands and transmit the current system information such as altitude, orientation, and GPS coordinates of the rover. Additionally, the PIC receives information from the bluetooth modules to determine whether it has moved the prescribed five feet from the rocket. The Electronic Control System is comprised of 4 subsystems, a microcontroller, and a base station.

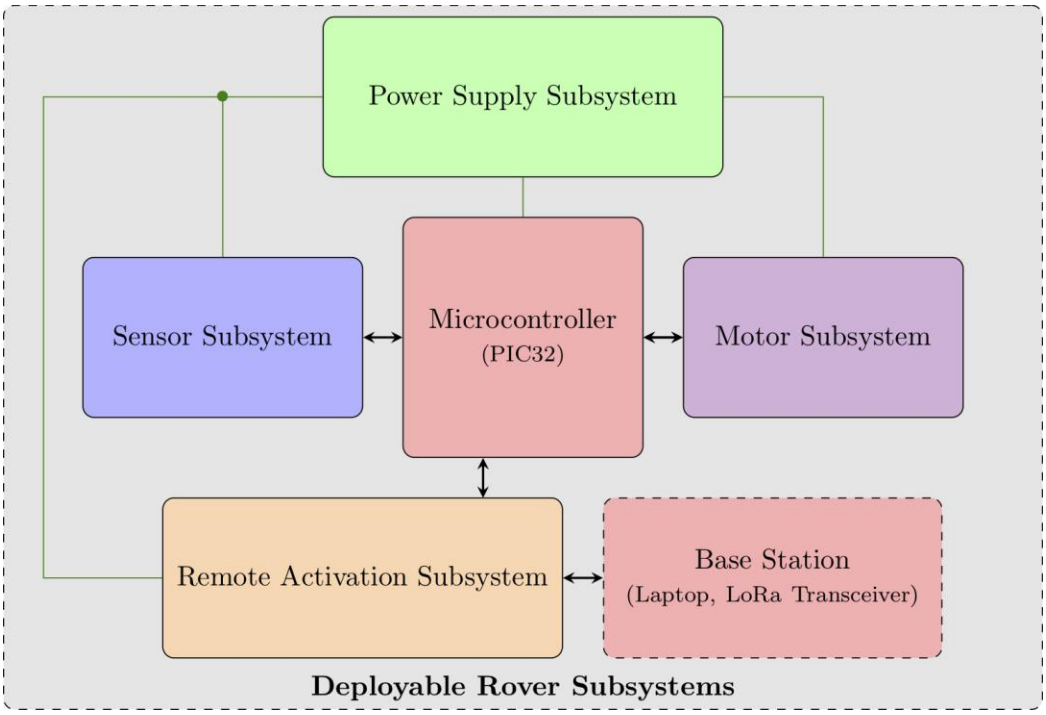


Figure 78. Electronic control system schematic.

4.1.3.2.2.1 Microcontroller

A PIC32MX795 microcontroller acts as the control center for all of the various rover subsystems. This particular microcontroller was chosen for its ability to work with 8 byte data streams, which are required by some of the sensors, and the large number of re-mappable pins that are available for controlling various inputs and outputs. In addition, utilizing a single microcontroller rather than various breakout boards to control the rover sensors allowed all of the different sensors to be connected to a single printed circuit board, reducing the footprint of the electronics necessary to operate the rover. The microcontroller receives information from the sensor subsystem and interprets this information in order to control the remote activation and motor subsystems. The microcontroller is powered by 3.3 V, which is drawn from the power

supply subsystem by passing the 8.4 V provided by the batteries through an LD1117D voltage regulator that outputs 3.3 V.

4.1.3.2.2.2 Remote Activation Subsystem

The deployable rover uses the remote activation subsystem to receive commands from the base station, transmit data to the base station, and to detonate the black powder charges. The remote activation subsystem also provides a means of safely priming these charges once the rocket is on the launch pad.

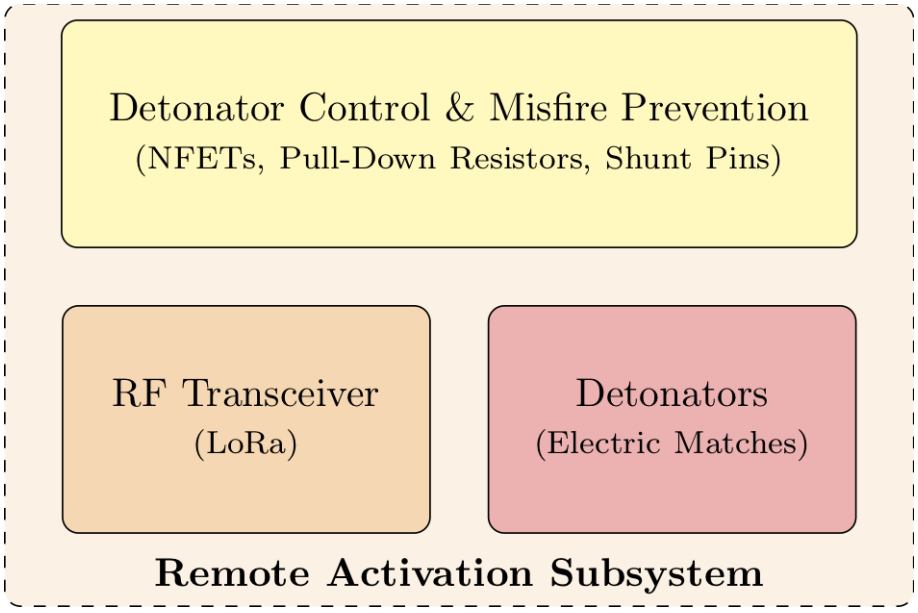


Figure 79. Remote activation for black powder schematic.

4.1.3.2.2.3 LoRa RF Transceiver

A RN2903 LoRa module is used for wireless communications. This talks to the ground station, sending in flight data such as altitude, GPS coordinates and orientation. After the Rover lands, the LoRa is responsible for sending the signal to deploy the rover. This involves lighting the electronic matches to ignite the black powder, unlock teh rover and exit the body tube. At this point the LoRa continuously sends its location and distance from the rocket back the ground station.

4.1.3.2.2.4 Detonators

Electric matches are used as detonators for setting off the black powder charges that blow the nose cone off the rocket and allow the rover to exit the rocket tube. It was determined that it takes a current of at least 500 mA in order ignite these particular electric matches. The circuit that controls the current flowing through the electric matches is described below.

4.1.3.2.2.5 Detonator Control Circuit

The electric matches that are used to detonate the black powder charges for removing the nose cone of the rocket are controlled by a simple N-Channel Logic Level Enhancement Mode Field Effect Transistor (NFET) circuit. The particular NFET that is used is an NDS355AN NFET, whose gate voltage is controlled by the output from one of the output pins on the LoRa module. When the output pin from the LoRa module is set to a logical high voltage (3.3V), it raises the gate voltage on the NFET to 3.3 V. The 3.3 V gate voltage is achieved by passing the output voltage from the batteries through an LD1117 voltage regulator that maintains a 3.3 V output voltage. This allows current to flow from the source of the NFET to the drain of the NFET. The source of the NFET receives 8.4 V directly from the batteries used to power the rover, while the drain of the NFET leads through the electric matches to ground. Therefore, when the gate voltage is set to 3.3 V, the current from the batteries flows through the electric matches to ignite the matches and detonate the black powder charges, thus removing the nose cone from the rocket. In addition, each detonator control circuit contains a 5.1 Ω resistor on the drain side of the NFET to prevent the current through the NFET from exceeding the 1.7 A for which they are rated. Finally, each detonator control circuit utilizes 4.7 kΩ pull-down resistors at the gate of the NFET to ensure that the gate is not inadvertently set to a high before the output signal from the LoRa module is received. This prevents current from accidentally flowing through the electric matches, thus setting off the black powder charges, before the output signal is received.

4.1.3.2.2.6 Misfire Prevention

The detonator control circuit also includes shunt pins across the leads of each detonator as a means of mechanically priming the nose cone's primary and secondary black powder charges. The shunt pins prevent the detonators from igniting the charges, as any incidental current that would go to the detonator instead goes through the shunt pin. Once the rocket is safely placed on the launch pad, the shunt pins are removed from the detonator control circuit through a hole in the side of the rocket. Additionally, connectors for the NFET control lines were placed on the solar panel racks so that they are only connected to the black powder charges after the rover is locked into place within the nose cone. This will help ensure that the electric matches are not ignited prematurely. After the rocket touches down and the black powder charges are blown, the solar panel rack will retract, thus removing the control lines from the detonator control circuits.

4.1.3.2.3 Sensor Subsystem

The deployable rover uses the sensor subsystem to determine its location, orientation, and velocity, relative to the Earth, the rocket, and objects in front of it.

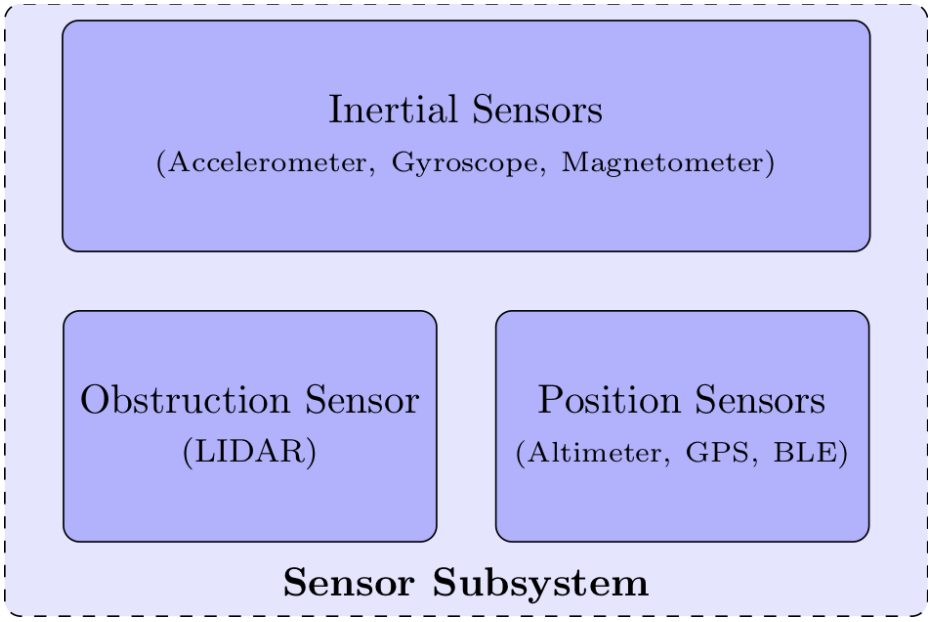


Figure 80. Sensor subsystem schematic.

4.1.3.2.3.1 Inertial Sensors

Gyroscope/Accelerometer/Magnetometer

An LSM9DS1 three-in-one gyroscope/accelerometer/magnetometer module is used to determine the orientation of the rover after the rocket has touched down, whether it is right-side-up or upside-down. This information is important for correctly interpreting the information from the LiDAR sensor on the position of potential obstacles in the rover's path and then controlling the motors so that the rover avoids the potential obstacle. This particular module was chosen for its durability and low cost. The module is powered by 3.3 V, which is drawn from the power supply subsystem by passing the output voltage from the batteries through an LD1117 voltage regulator that maintains a 3.3 V output voltage.

LiDAR Obstruction Sensor

The sensor chosen to sense obstacles in the rover's path is a Garmin LiDAR Lite v3, part number SEN-14032. A LiDAR sensor was determined to be the most reliable in the event of excess noise, which would affect an ultrasonic sensor, or poor weather conditions, which would affect both ultrasonic sensors or different visible light sensors, on the launch day. This specific LiDAR sensor was chosen because of its simple Inter-Integrated Circuit (I^2C) interface for data transfer. In addition, this particular sensor was able to transfer data at speeds up to 400 kHz, which ensured that potential obstacles were detected quickly, as well as provide a history of recorded data in case failure analysis was required. The LiDAR sensor is powered on the rover using 5 V, which is achieved by running the voltage provided by the batteries through an LD1117 voltage regulator that maintains a 5 V output voltage. The LiDAR sensor is mounted

onto the “front” of the rover, which is the side that faced toward the top of the nose cone when the rover is placed into the rocket. This side is the first to exit the rocket upon deployment, even if the orientation of the rover is flipped. Therefore, the LiDAR is able to begin sensing obstacles as soon as the rover exits the rocket.

4.1.3.2.3.2 Position Sensors

GPS

The GPS used is a FGPMMOPA6H, which was chosen for its low power consumption and easy UART interface. Additionally, this GPS module is configured to send data as soon as it finds an uplink. This is convenient due to its lack of significant configuration. The GPS has been checked to ensure that it is sending accurate data. No further testing is required.

Bluetooth

A JDY-08 bluetooth module is used to determine the location of the rocket. Each section of the rocket has a module acting as a beacon, in addition to three modules on the rover itself to triangulate position. This results in a total of six bluetooth modules. The module's signal strength was tested at constant power for various distances. During testing it was observed that the change in signal strength was negligible after distances at or greater than 10 ft. This is greater than the 5 ft requirement and so the JDY-08 bluetooth module can be used successfully to meet this criteria. The module's signal strength was also measured as a function of the orientation of the module. This test demonstrated the expected range in signal strength given a constant distance. Since the module could be in any orientation when the rocket lands, this expected range is useful for determining if the rover is within 5ft of the rocket or not. These modules are located in each section of the rocket to provide a complete “image” of the rocket: one module is located in the Air Braking Payload, one is located near the CRAM inside the middle body tube section, one is mounted on the back bulkhead of the Deployable Rover Payload and one is located inside the nose cone. Two modules are also located on the rover itself.

Altimeter

An MPL311A2 altimeter was added to the sensor board for measuring altitude throughout the flight of the rocket, which allows for confirmation that the rocket achieved the target altitude. In addition, the altimeter communicates when the rocket has landed in order to begin the process of rover deployment. This particular altimeter was chosen because it utilizes a simple Inter-Integrated Circuit (I^2C) interface for data transfer. This particular altimeter also contains a built-in altitude calculator, so additional calibration or calculation is not required. The altimeter is powered on the sensor board using 3.3 V, which is achieved by running the voltage provided by the batteries through a LD1117D voltage regulator whose output is a constant 3.3 V.

4.1.3.2.4 Motor Subsystem

The deployable rover uses the motor subsystem to control its movement, to deploy the foldable solar panels, to secure itself within the rocket, and as a means of physically disconnecting from the nose cone black powder charges, thereby disarming them.

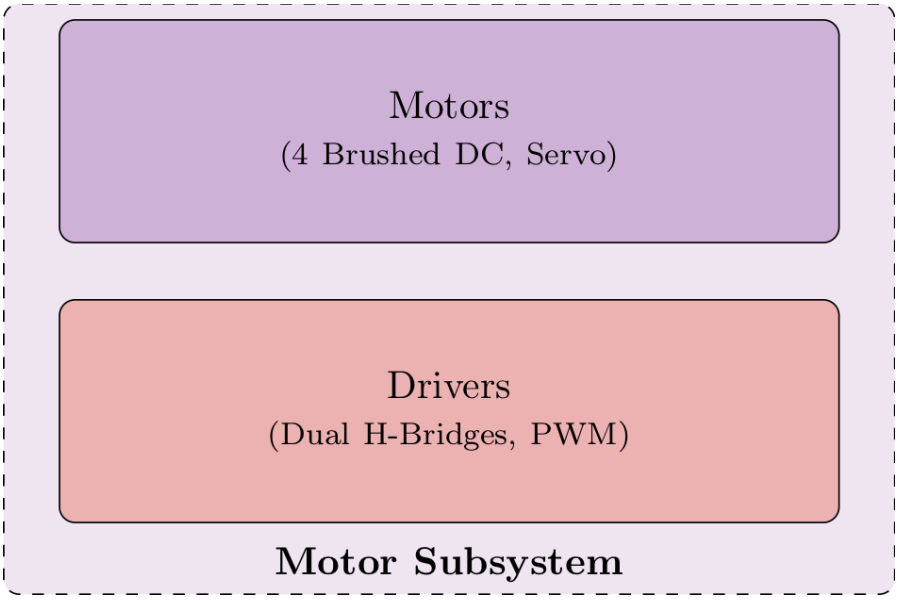


Figure 81. Motor subsystem schematic.

4.1.3.2.4.1 Motors

Brushed DC

As detailed in Section 4.1.3.1.2, the LEGO Power Functions XL motor is a brushed direct-current (DC) motor with an internal two-stage planetary gearbox. The motor subsystem uses a single XL motor for each wheel, for a total of four XL motors. The LEGO motor can be seen in Figure 64. Since the LEGO connectors would not be adequate for our application, the wires were modified to use a molex connector. This was accomplished by cutting the wires and crimping molex connectors that could plug into the power and driver boards.

Servo

The Hitec HS-7245MH is a servo motor with a metal gear train and dual ball-bearing output shaft support and a stall torque of 88.88 ounces per inch at 7.4 volts. The motor subsystem uses a single HS-7245MH modified for continuous rotation for securing itself inside the rocket, disarming the nose cone black powder charges, and deploying the foldable solar panels.

4.1.3.2.4.2 Drivers

Dual H-Bridges

The Texas Instruments DRV8835 is a Dual Low-Voltage H-Bridge. The motor subsystem uses two DRV8835s to control the speed and direction of its four brushed LEGO XL motors.

PWM

The PIC32 provides pulse-width modulation (PWM) functionality on several of its output pins. The motor subsystem uses PWM on the servo's control line to control the speed and direction of the continuous servo motor.

4.1.3.2.5 Power Supply Subsystem

The deployable rover uses the power supply subsystem to provide power for its electronics via rechargeable batteries and photovoltaics.

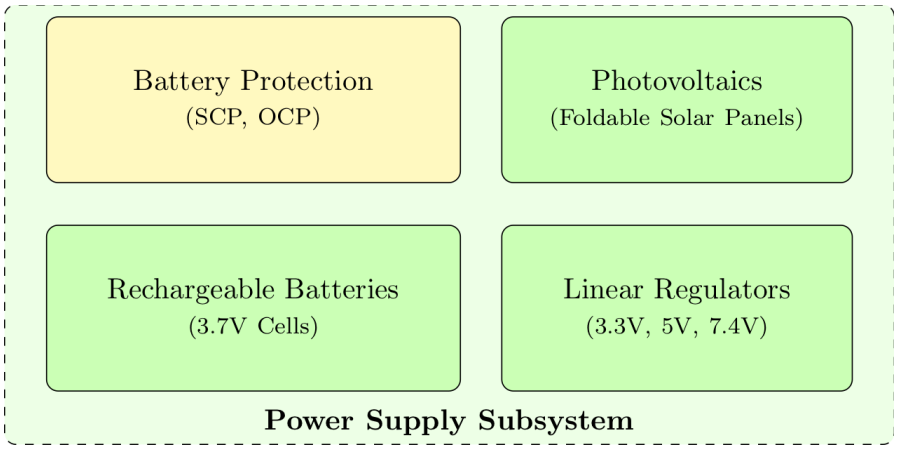


Figure 82. Power supply subsystem schematic.

4.1.3.2.5.1 Rechargeable Batteries

3.7V Cells

The power supply subsystem uses four IMREN 18650 3.7V 3000mAh lithium manganese oxide batteries. Two groups of batteries are connected in parallel, each group with two cells connected in series. This configuration yields a nominal voltage of 7.4 volts, with a capacity of 6000 milliampere-hours.

4.1.3.2.5.2 Battery Protection

Short-Circuit Protection (SCP)

Short-circuit protection for the batteries is implemented via custom 3D printed housings made out of PLA.

Over-Current Protection (OCP)

Over-current protection for the batteries is implemented through built-in overcurrent protection circuitry contained within the rover's motors, motor drivers, and the rover's linear regulators.

Linear Regulators (3.3V, 5V, 7.4V)

The STMicroelectronics LD1117 family is a family of low drop fixed and adjustable voltage regulators capable of outputting 800mA of current. The power supply subsystem uses three different types of LD1117, a 3.3V version (LD1117AV33), a 5V version (LD1117DT50TR), and an adjustable version (LD1117DTTR). The LD1117AV33 is used to provide power to the microprocessor, the inertial sensors, the position sensors, and the motor drivers. The LD1117DT50TR is used to provide power to the LIDAR. The LD1117DTTR is used to provide power from the photovoltaics to charge the batteries.

4.1.3.2.5.3 Photovoltaics

Foldable Solar Panels

The solar panels are wired in series as recommended in Figure 83. The electrical functionality is verified by a resistor placed on the board powered only by the solar cells.

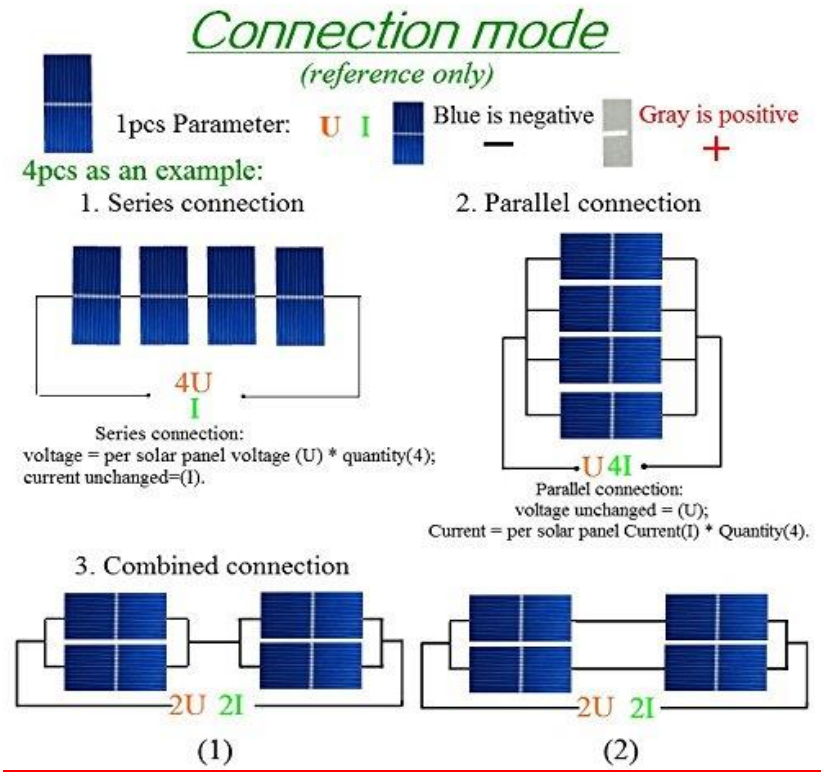


Figure 83. The wiring diagram used for the photovoltaic cells in the folding array.

4.1.3.3 Ground Station

The ground station consists of a 2017 15 inch MacBook Pro connected to a LORA module. The Macbook has many custom shell commands written for a Unix Terminal that allow messages to be written quickly and conveniently to the LORA module. All code used for the payload can be found in Appendix R.

4.1.3.4 Algorithm

The algorithm employs a state machine controlled by inputs from the LORA module. It waits indefinitely until it receives an interrupt from the LORA module, executes the corresponding subroutine. There are seven commands that the module may receive from the LORA, these are: prepare for flight, arm rover, light charges 1, light charges 2, deploy rover, move rover, and deploy solar panels. Prepare for flight extends the rails into the mounting blocks, securing the rover in the body tube for the duration of flight. Arm rover is the in flight routine for the rover, and it uses pulse width modulation (PWM) in order to provide resistance to the servo turning and falling out of position. This is necessary due to the removal of the potentiometer from the servo to modify it for continuous rotation. Light charges 1 lights the primary black powder charges. Light charges 2 lights the secondary black powder charges. Deploy rover retracts the rails from the mounting blocks, and has the rover exit the body tube. Move rover is the movement algorithm, and ensures that the rover has moved the required five feet from the rocket. Finally, deploy solar panels extends the rails and unfurls the solar panels.

4.1.4 Payload Construction

4.1.4.1 Structural Construction

4.1.4.1.1 3D Prints

Several parts on the rover were custom-designed in CREO Parametric software and manufactured with University 3-D printers. The low cost and quick turnaround time allowed several iterations of parts to be made and altered. Parts that require more strength and precision to function correctly were printed in a Stratasys Fortus 250mc printer with Acrylonitrile butadiene styrene (ABS) material. These pats included the solar panel rail mount, as well as the body tube tracks and supports. Other sections that did not demand as high a degree of precision and were required to be manufactured quickly were printed with Makerbot Replicator + and 5th Generation printers with PLA filament. These parts included the motor mounts, servo mount, wheel hubs, as well as the rover mounts on the body tube that the securing racks lock into and the cover for the electronics. Infill, shell number, and extruder temperatures were also modified for the specific demands of each part.

4.1.4.1.2 Rover Assembly

The first step was to mill the body of the rover as described previously. While the body was being milled, the racks to hold the rover in place inside the rocket and extend the solar

panels were cut. They were then combined with the pinion gear and servo motor to create the extension system. Once the body was complete, the mounts for the wheel motors, servo motor, and extension system, were attached using nylon screws. All holes were tapped to provide a secure connection without the necessity of hex nuts. The wheels were attached to the motors with Lego connector axles and the custom 3D printed wheel hubs, and the connections were reinforced with epoxy to ensure the press-fits would not come loose. The battery cases and control boards were then mounted to the rover and jumper wires were connected to allow for electronics testing. Once the testing was complete, the wires were replaced with shorter connections and secured with zip-ties. A 3D printed case was then affixed to the rover to protect the electronics and wire connections during locomotion. The prototype rover that was built for the full scale test launch can be seen in Figure 84. After the full scale launch improvements will be made for the final version of the rover.

The rover was then integrated with the rocket. This involved epoxying the tracks and the mounting blocks to hold the rover in place during flight onto the inside of the body tube. In order to facilitate accurate alignment of these components, a plywood alignment tool was milled using a CNC Techno Router. This alignment tool can be seen in Figure 85. During assembly it was noted that the tracks needed to be cut down and sanded to provide for better clearance inside the rocket. The final track configuration can be seen in Figure 86.

The black powder deployment system was constructed using bulkheads and PVC pipes. Three custom bulkheads were milled: one for the back of the payload and two for the nose cone. The two PVC tubes were epoxied to the base bulkhead. The smallest bulkhead was epoxied five inches into the nose cone. The final bulkhead was epoxied at the opening of the nose cone with slots for the PVC pipes. This configuration can be seen in Figure 87. In order to slide the nose cone into the body tube four slots were cut from the nosecone.

During construction personal safety was of utmost importance. All sanding of carbon fiber and plastic was performed while wearing a mask and safety glasses, and all epoxying was performed while wearing gloves.

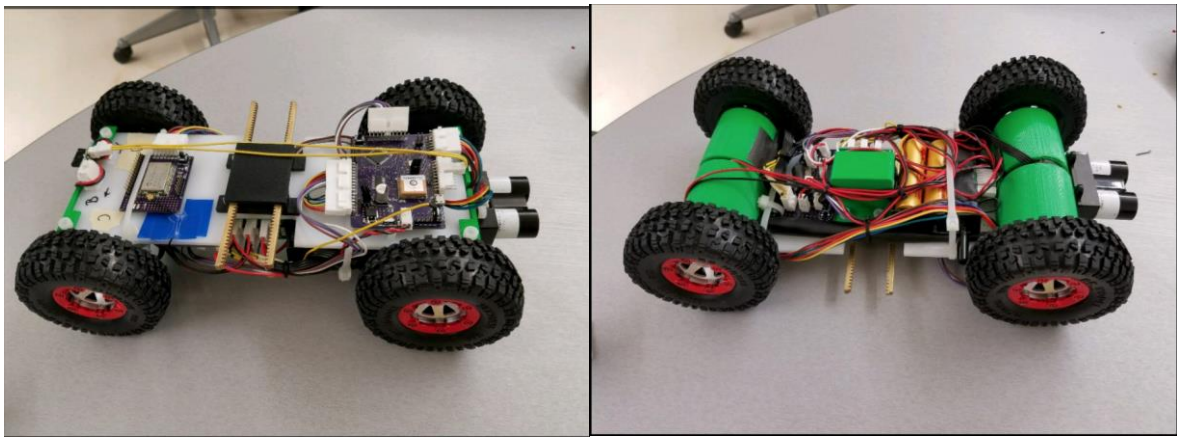


Figure 84. Constructed prototype of the rover for the full scale test launch. Left shows the rover from a "right-side-up" orientation. Right shows the rover in an inverted orientation.

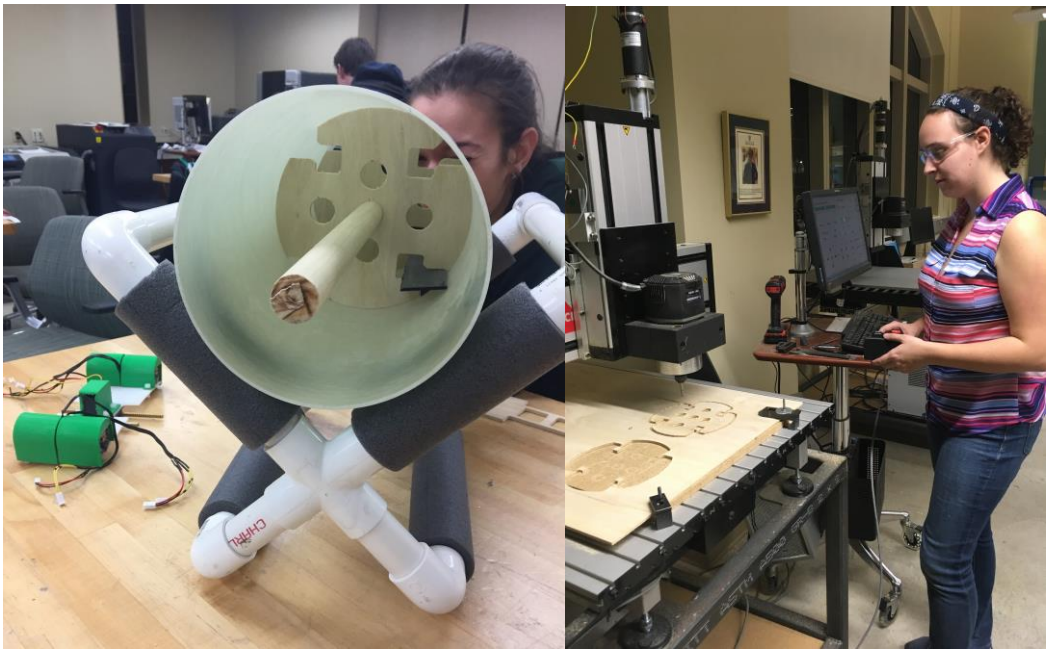


Figure 85. An alignment tool was created for ease of construction and placement of the tracks.

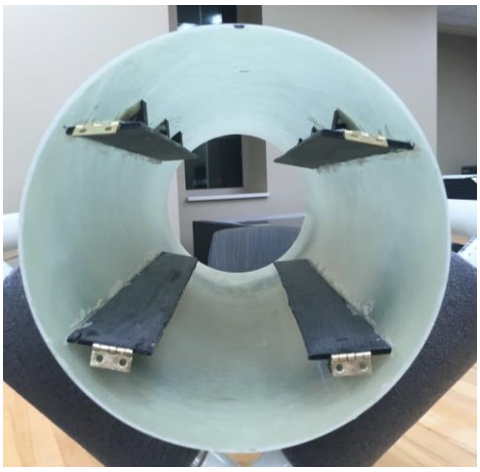


Figure 86. Construction and epoxying the tracks that the rover drives on.

Figure 87. Mounting the PVC pipes to the back bulkhead.

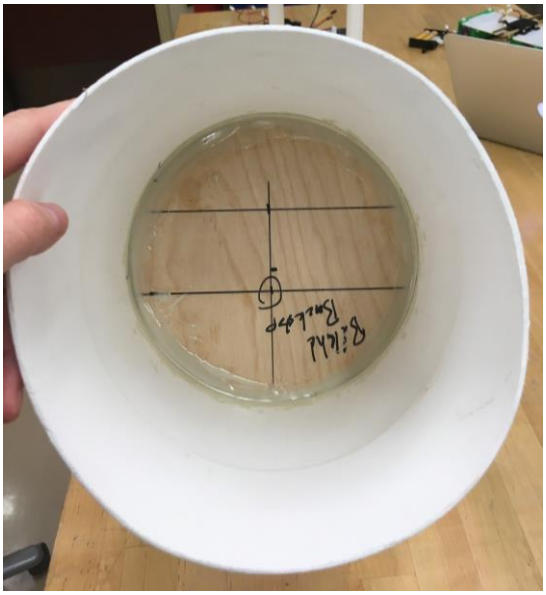


Figure 88. Epoxying the front bulkhead of the nose cone.

4.1.2.1.3 Payload Integration

The Deployable Rover Payload is integrated into the launch vehicle through bulkheads, tracks and nose cone alteration. The payload section contains three plywood bulkheads. One at the back of the payload that is mounted with two PVC pipes. This bulkhead is secured using RocketPoxy. The tracks are 3D printed components that allow for the rover to drive out of the vehicle. They interface with the body tube and are secured with RocketPoxy. For extra support, 3D printed triangular supports are also epoxied to the tracks and the wall of the body tube. In between the tracks two 3D printed mounting blocks are secured with RocketPoxy. These blocks prevent motion of the rover during the flight path. The tracks run to the opening of the body tube. Due to this, the shoulder of the nose cone had to be altered to slide over the tracks and supports. Four slots were cut into the shoulder to achieve this clearance. Two more slots were cut into the nose cone to allow clearance for the shunt pins that arm the ejection charges. The nose cone interfaces with the fiberglass body tube through eight shear pins. This number was optimized via black powder testing. The nose cone has two bulkheads fixed with RocketPoxy. The larger bulkhead has two circles cut out for the PVC pipes. This bulkhead is fixed at the base of the shoulder of the nose cone. The smaller bulkhead is epoxied six inches back from the shoulder. This configuration provides a closed area for the ejection charges to detonate.

The entire payload section interfaces with the transition section of the launch vehicle. The back mounting bulkhead is flush against the bulkhead of the transition section. The six inch shoulder of the transition section was secured to the fiberglass body tube of the payload using RocketPoxy.

4.1.2.2 Electrical Construction

The OSH Park printed circuit board was received in mid-December 2017. The board was constructed in the days that immediately followed. This was completed using solder paste and solder. There were several issues adequately soldering the pic32 microcontroller, specifically with solder bridges between the pins from a stencil that was slightly too large. This resulted in the PIC being soldered by hand rather than with solder paste. The LORA board and the bluetooth modules were constructed in the same manner. However, the footprint was slightly off for the bluetooth modules; therefore, it was necessary to bridge each of the connections rather than use solder paste.

In addition to each of the boards, it quickly became apparent that other boards for power control, remote deployment and various other functions would be necessary. To that end, several other boards were constructed using PCBs that were on hand. The first of these was a voltage board. This board converted a molex to several molex, allowing each of the motors to be plugged in as well as the servo and anything else that required battery voltage.

The final board that was constructed was used to set off the black powder charges. This used an nFET to connect battery to ground, with a control line on the gate to switch it on. The control line was taken off of the pic32.

4.1.2.3 Construction Differences

During construction of the payload several differences occurred. The placement of all components remained the same, except for the Lidar sensor. In order to provide greater clearance, the sensor placement was changed to be mounted on the front of the Lego motors rather than flat on the HDPE. This led to a change in the overall length of the rover from 11.893 inches to 12.5625 inches. The width of the rover, tire to tire, also ended up being slightly wider than designed. This can be attributed to tolerances and epoxying of the axles. The original width was 6.535 inches and the constructed width is 6.75 inches.

To provide a safe way of arming the rocket, a small shunt pin was placed at the opening of the body tube near the PVC pipes. These pins were epoxied to the side of the body tube with the pin sticking outside the launch vehicle. The original design did not include this external arming system.

Due to the track system and the shunt pins the nose cone had to be cut to provide clearance when inserting into the body tube. This involved cutting slots into the shoulder of the nose cone. The front bulkhead inside the nose cone also had slots cut out for the shunt pin.

The original design for the ramps that provided an extension of the tracks of the payload were four individual ramps. During construction a cross beam was added to provide more support and prevent the rover from bottoming out on the PVC pipes.

During construction it was also realized that the PVC pipes could be used to hold the wires that run from the back bulkhead to the front of the payload. Two holes were drilled allowing the wire to feed to the front and connect to the shunt pins. This eliminates the possibility of the rover getting caught on a stray wire.

4.2 Air Braking System

4.2.1 System Design

4.2.1.1 System Overview

The system is composed of three physical subsystems and a control code. The first subsystem is the aerodynamic subsystem, composed of the drag tabs. These tabs are flat plates with a curved outer edge designed to sit flush with the rocket body when retracted. After motor burnout, the tabs extend into the flow to induce an additional drag force. Because the tabs have a variable extension, the drag force is controllable. The tabs are connected to the second subsystem, the mechanical subsystem. This is composed of four crank slider mechanisms, one connected to each tab. The mechanism is driven by a central shaft, which interfaces with the third subsystem, the electronic subsystem. Two servo motors drive the central shaft of the mechanism. These are controlled by a microcontroller running the control code. This code uses data from a barometer and accelerometer to calculate the current flight path of the rocket, then uses PID control to match to a pre-calculated ideal flight path. All of these components have placed on decks and slid onto threaded rods to secure everything in place. The system is designed to fit inside of a 5.27 in. coupler. The fully assembled height of the payload is 12.5 in. and the fully assembled weight of the payload is 73 oz. Full images of the assembly are shown in Figures 89 and 90 below.

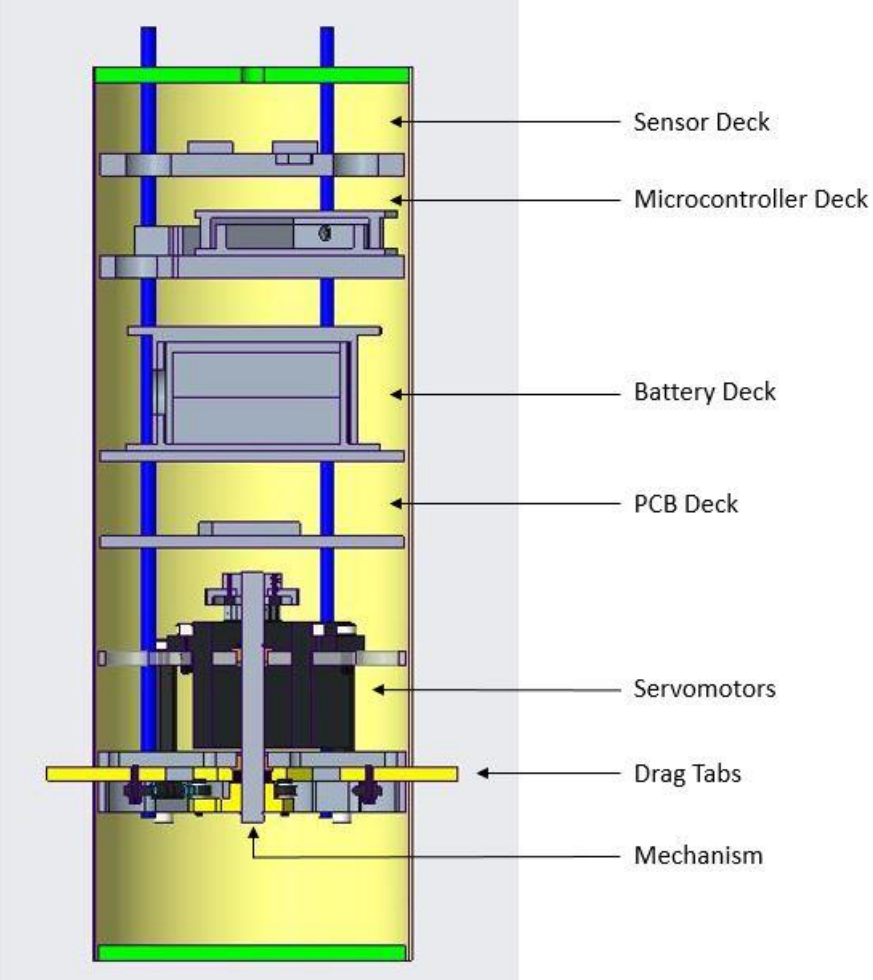


Figure 89. Cross-sectional CAD model of the full air braking system.

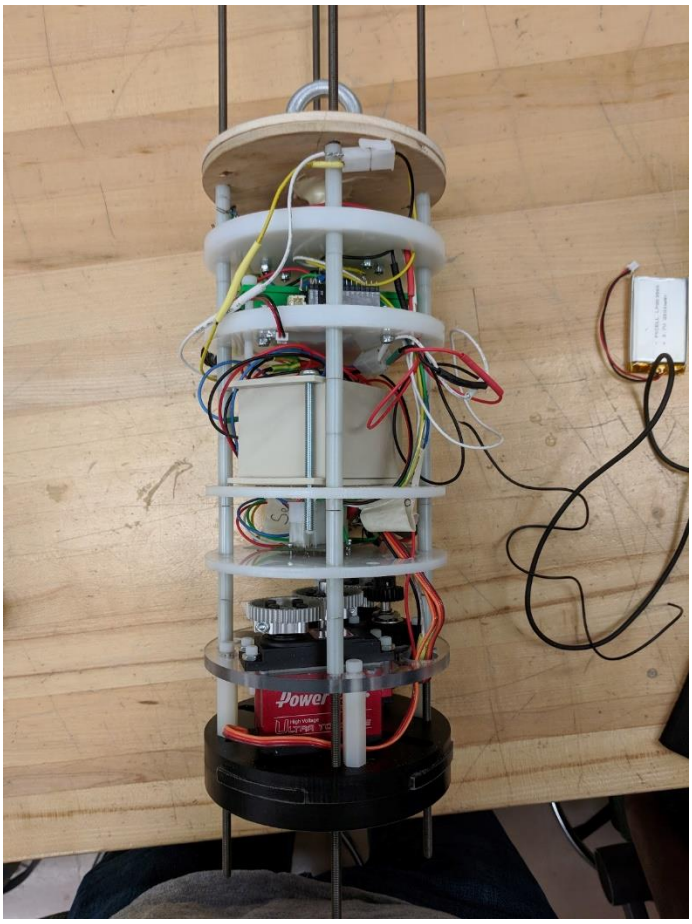


Figure 90. Image of the full air braking system assembly.

4.2.2 Success Criteria

The purpose of the system is to assist the vehicle with reaching an apogee of exactly 5280 ft. The progress toward this goal is measured in four different stages of success. Stage 1 success is to affect the rocket in a measurable way. This was tested during the subscale test flight, when two flight were launches, one without the air braking system, and one with a 3D printed scale model of the fully extended drag tabs. By comparing the apogees of the two flights, it was determined that at full extension during the whole flight the tabs brought down the apogee by 61 ft, which was significantly larger than the 48 ft reduction it was predicted to cause based on scaling a full scale flight. For this reason stage 1 success has been reached. Stage 2 success is to reach an apogee within the margin of error of previous methods used, namely using ballast. Stage 3 success is to reach an apogee closest to 5280 ft within the competition. Whether or not this stage has been achieved can only be determined at competition. Stage 4 success is to hit apogee at exactly 5280 ft. If stage 4 success is reach the system has achieved its purpose and the only improvements that can be made are to increase redundancy and consistency. The success stages are used to determine what type of improvements should be made to the system.

4.2.3 Aerodynamic Subsystem

4.2.3.1 Drag Tab Design

The area of each drag tab was determined to be 2 in^2. This tab area was calculated by assuming that the tabs would be fully extended right after motor burnout and for the duration of the flight. The tab area was calculated using a summation of forces for the rocket seen in the equation below.

$$m_{rocket}a = F_{drag,rocket} + F_{gravity} + F_{drag,tabs}$$

In the above equation, m_{rocket} is the mass of the rocket, a is the deceleration of the rocket required to reach an apogee of 5280 ft, $F_{drag,rocket}$ is the skin-friction drag of the rocket, $F_{gravity}$ is the total weight of the rocket, and $F_{drag,tabs}$ is the total drag of the tabs at full extension. The drag forces of the rocket and the tabs were calculated using the equation below.

$$F_{drag} = \frac{1}{2}\rho v^2 A C_D$$

In the above equation, ρ is the density of the fluid, which is air in this situation, A is the cross-sectional area, v is the velocity of the rocket, and C_D is the coefficient of drag. The drag tabs were approximated as a flat plate and the rocket was approximated as a bullet. According to NASA, these approximations provided drag coefficients of 1.28 and 0.295 for the drag tab and the rocket, respectively. Using the approximate drag coefficients and the assumption that the tabs are fully extended for the duration of the flight, the tab area was calculated to be 2 in^2. However, because of variations in air density and rocket velocity, the full extension of the tabs throughout the duration of the flight is not necessary. As such, a control system will be implemented to continually calculate the drag needed to reach apogee and will vary the extension of the tabs into the flow as necessary. The varying extension of the tabs allows for the manipulation of the cross-sectional area exposed to the flow and, in turn, the manipulation of the drag force created by the tabs.

The four tabs were cut on a Techno CNC router from a 0.25 in. sheet of Ultra High Molecular Weight polyethylene (UHMW), which is a high strength plastic. After a shallow chamfer was made on each tab and all tabs were cut from the sheet, M3 holes were drilled and tapped at the chamfer locations to allow the tabs to connect with the tie rods. In manufacturing, one face of the original UHMW sheet was left unmachined to provide a smoother sliding surface, reducing friction. The computer-aided design (CAD) drawing used for volume removal and center drill pecking is shown in Figure 91, while images of the manufacturing process are shown in Figure 92.

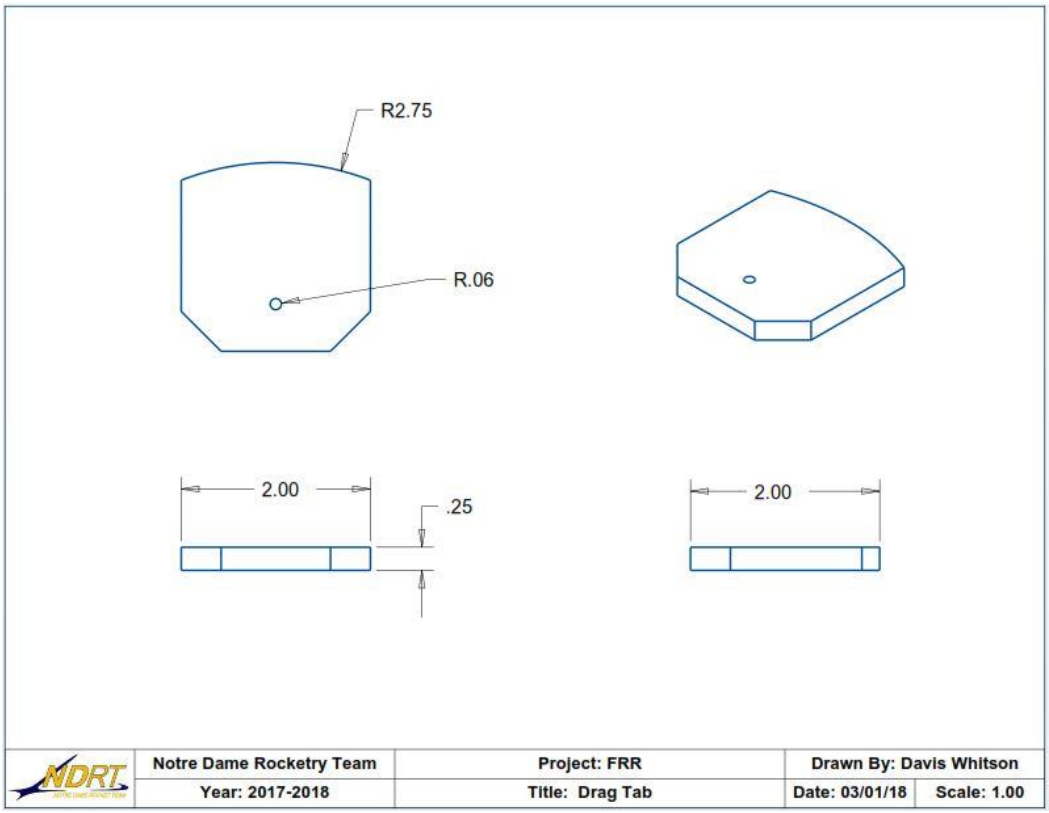


Figure 91. A detailed drawing of one drag tab.

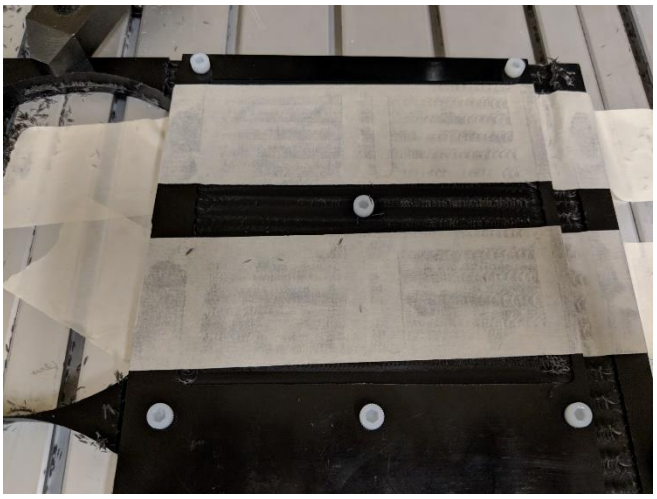


Figure 92. Cutting a set of four drag tabs from a sheet, which is secured in multiple places with nylon bolts to prevent warping. The tabs are also taped down to provide extra security before the final profile cut.

4.2.3.2 Finite Element Method Simulation

Before the final decision to use UHMW, finite element method (FEM) analyses were conducted on the tabs to ensure their durability under expected forces. A load of 34 psi was applied to the area the wind flow would see at maximum tab extension. This load gives a predicted factor of safety of 4. In the FEM analyses, three constraints were applied to simulate the boundary conditions during operation of the air braking system. One displacement constraint was placed on the hole inside of the tab and two planar constraints were applied to the sides of the tab, as the tabs would be able to slide in and out. The maximum shear stress and maximum displacement obtained by the FEM analysis for UHMW tabs were 25.7 psi, as seen in Figure 93 and 4.9 10-4in., as seen in Figure 94, respectively. As the yield stress of UHMW is 5000 psi, the maximum shear stress is three orders of magnitude smaller. Thus, the durability of the tabs under expected wind forces was verified. In addition, as the maximum displacement is four orders of magnitude smaller than the width of the tab, this was also deemed acceptable.

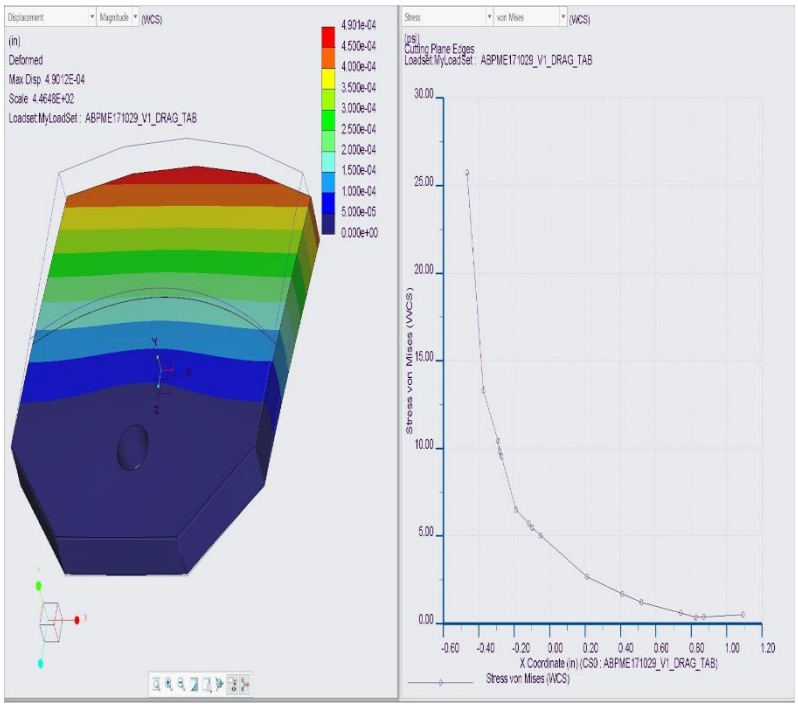


Figure 93. Plot of maximum shear stress as a function of distance for the UHMW tab. The maximum shear stress observed was 25.7 psi.

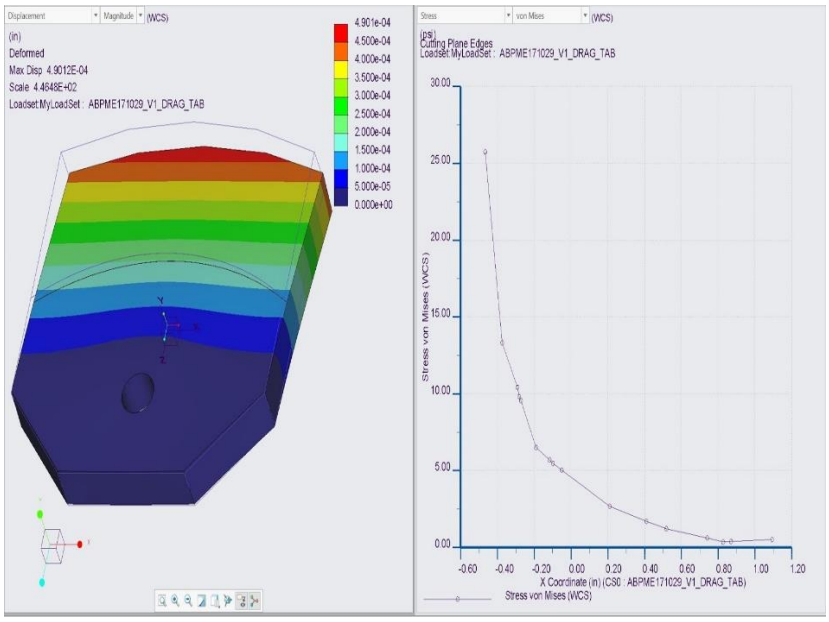


Figure 94. Schematic of UHMW tab displacement. The maximum tab displacement was 4.9 x 10^4 in.

4.2.4 Mechanical Subsystem

4.2.4.1 Mechanism Design

The mechanical subsystem deploys the drag tabs by way of a crank-slider mechanism, as seen in Figure 95. In this mechanism, the central cross-arm is turned by the servo motors. This in turn moves a set of four tie rods to extend the four drag tabs at the same time. The mechanism is designed such that the tabs can be retracted all the way into the slotted plate to allow for disassembly. Similarly, the tabs are mechanically limited to the maximum desired extension position, so it is impossible for them to overextend. The constructed version of the payload is given in Figure 96.

The system is designed to reduce weight wherever possible, so several considerations were made to accomplish this. First, all the main parts of the system are plastic. The UHMW components, in addition to reducing friction, are very light. Where UHMW would not work and more rigidity was needed, in the servo mount plate, polycarbonate was used instead. The payload decks are made of HDPE, and all spacers are nylon. Heavy clamping shaft collars were replaced with light retaining rings. Finally, there are almost no nuts used in the entire system's construction. Instead, most of the fasteners thread into tapped holes, and in applications where the fasteners were not being sheared, steel components were replaced by nylon components instead.

The required torque was verified using a dynamic model of the system according to the vector loop method. In this model, the system was considered in the worst case scenario, which is immediately after burnout with the tabs almost fully extended (at full extension the required

torque is zero, since the motors have infinite mechanical advantage). Since the main load on the mechanical system is friction, and friction models are imperfect, the selected gearmotors chosen provide more than three times the required worst-case stall torque. The full dynamic model of the mechanism is in Appendix J.

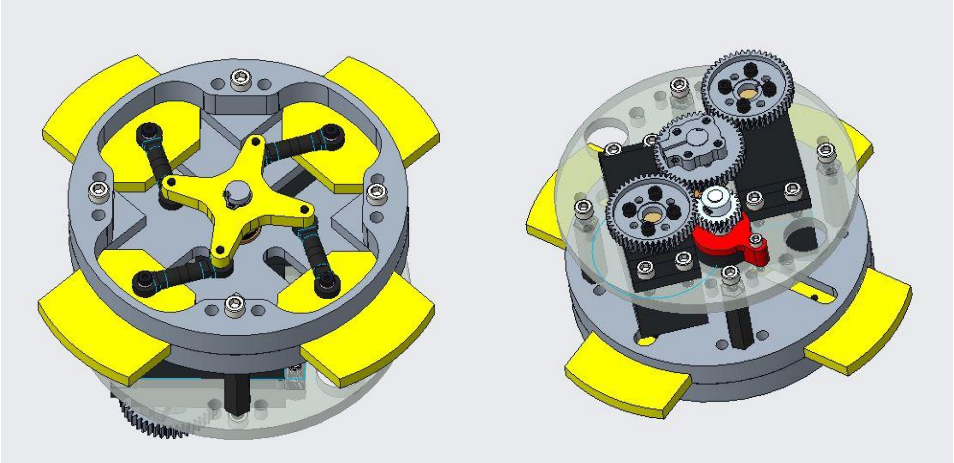


Figure 95. CAD model of the mechanical system.

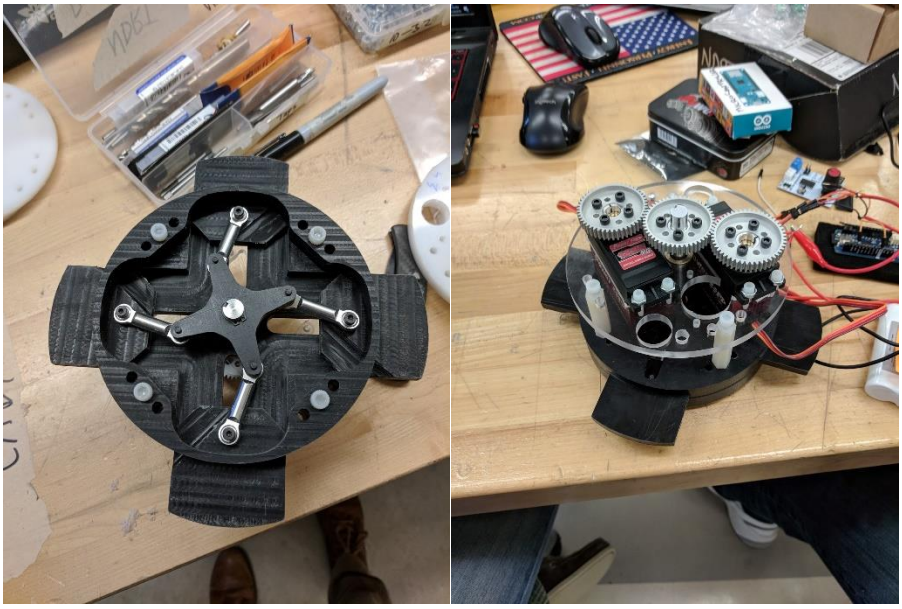
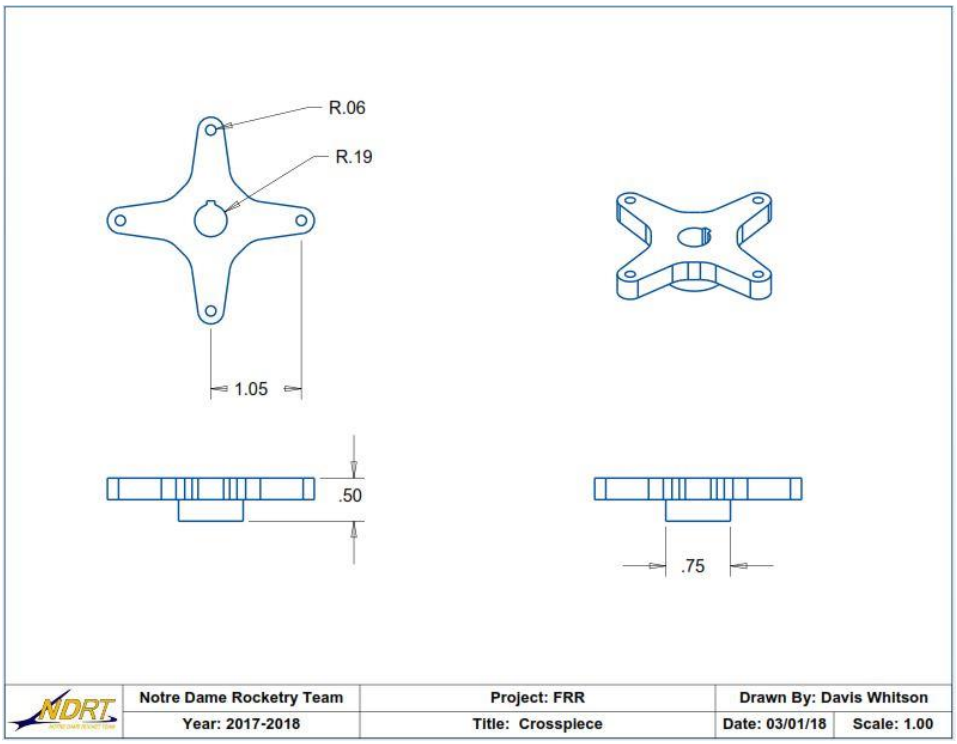


Figure 96. Real image of the mechanism.

4.2.4.2 Mechanism Components

The mechanism is composed of 5 part designations: the servo crosspiece, tie rods, drag tabs, and a top and bottom sliding plate. These images are shown in respective order in Figures 97 through 101 below with a corresponding image of the manufactured part also included.



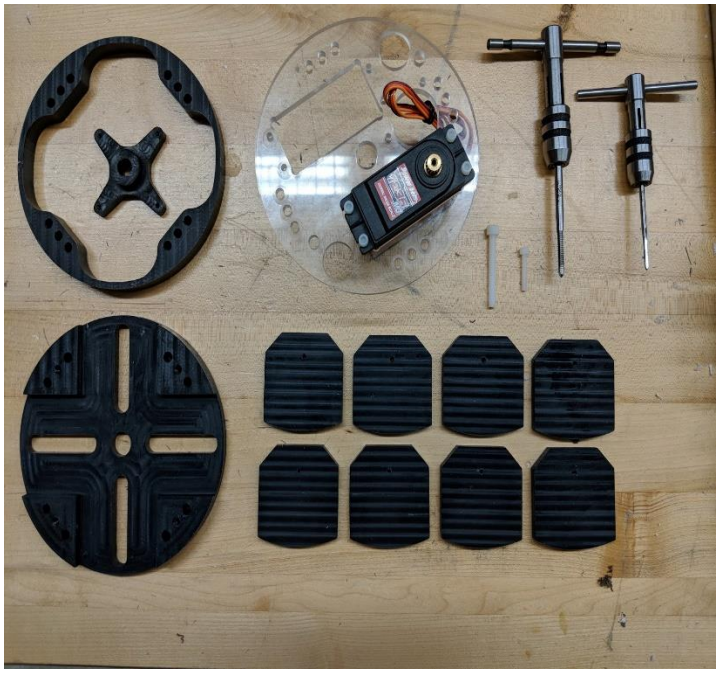


Figure 97. A detailed drawing and picture of the manufactured servo crosspiece. Units in inches.

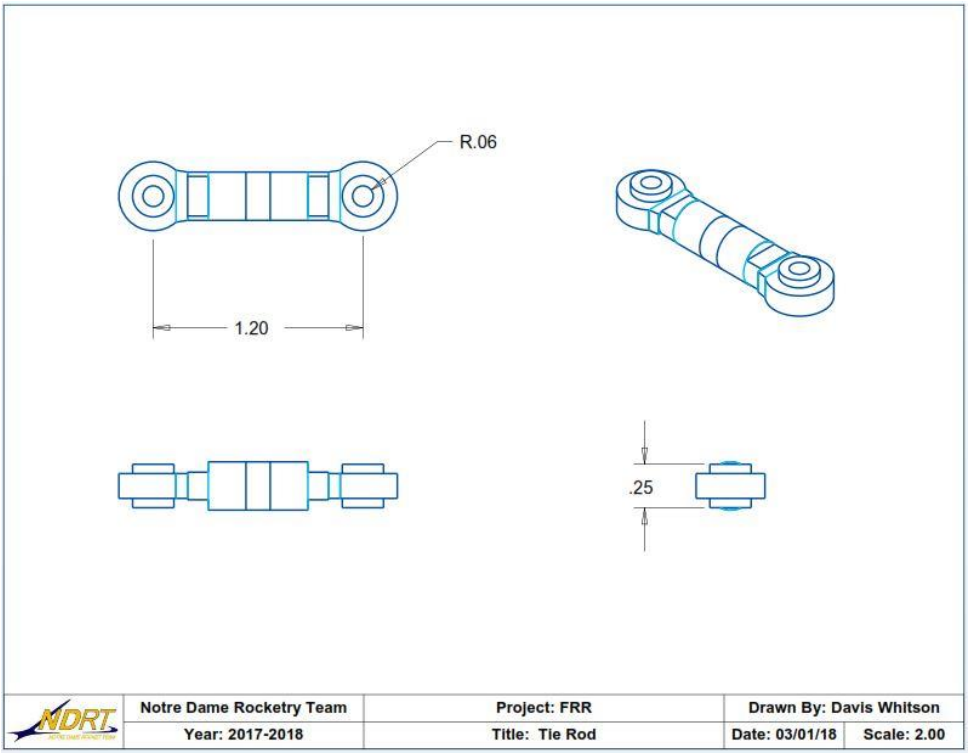
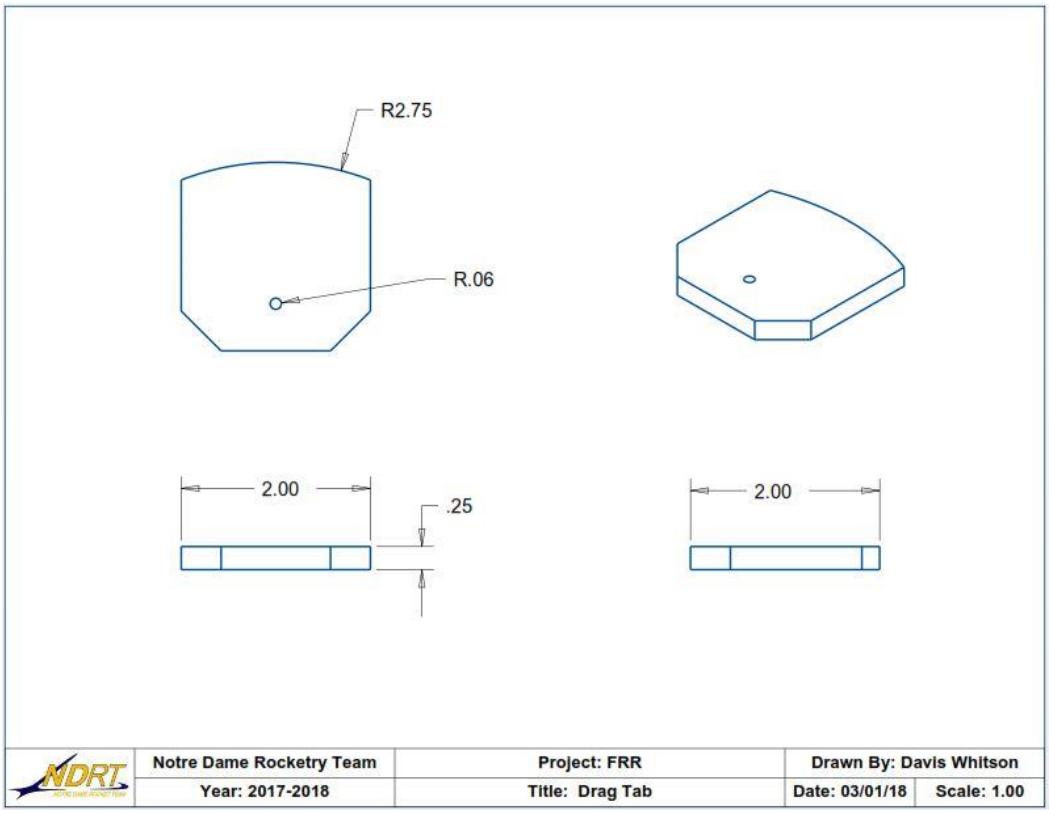


Figure 98. A detailed drawing and picture of one tie rod, though there are four on the real mechanism. Units in inches.



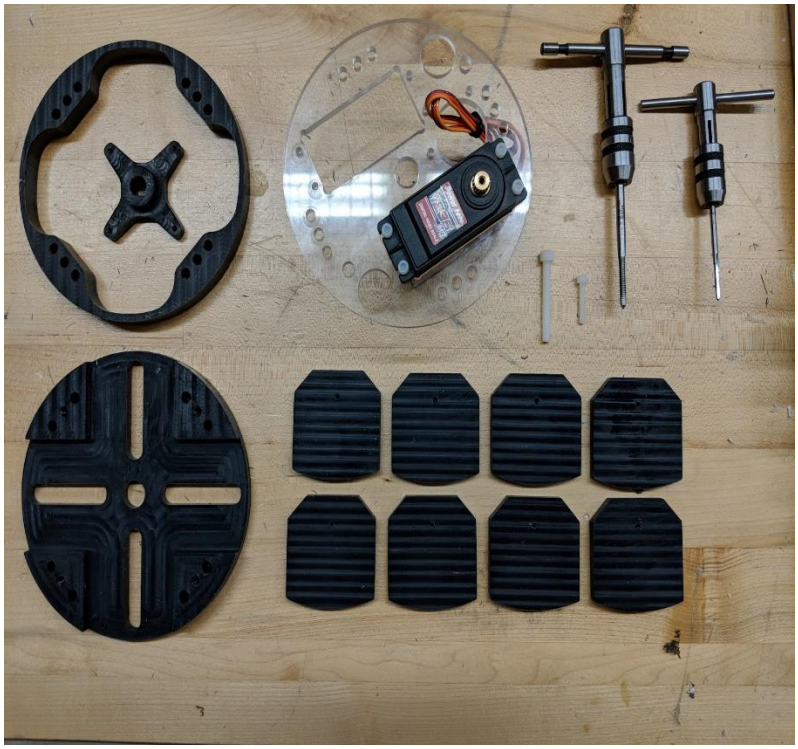


Figure 99. A detailed drawing and picture of one drag tab, though there are four on the real mechanism. Units in inches.

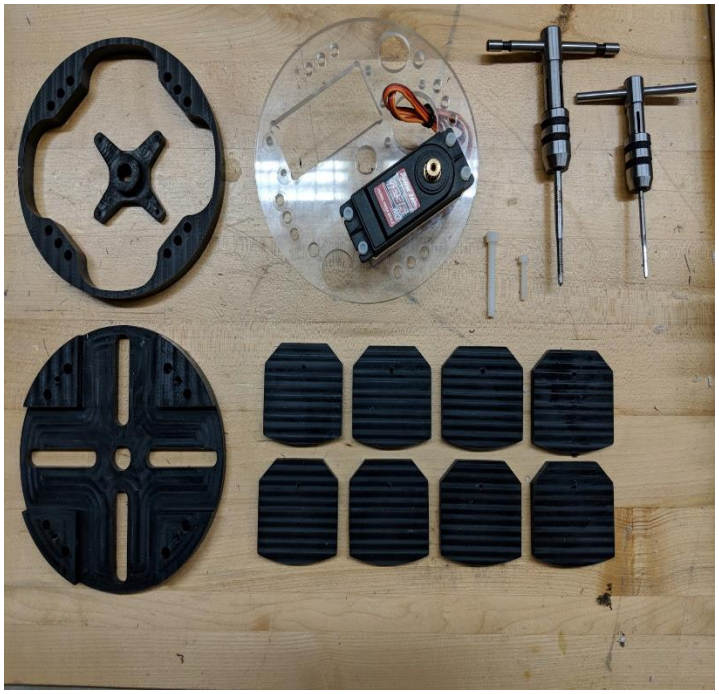


Figure 100. An image of the manufactured bottom slide plate (with cross piece in center).

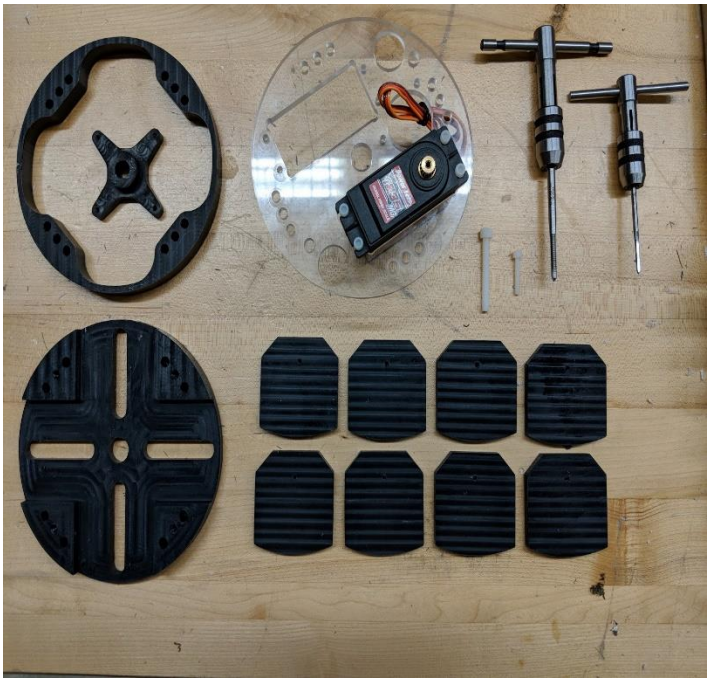


Figure 101. An image of the manufactured top slide plate.

The servo crosspiece, top slide plate, and bottom slide plate were all machined on a CNC router from a 0.5 in. UHMW sheet. All necessary holes were touched off in the router using a center drill for precision, and then drilled and tapped as needed. These holes included tapped

holes for screws and bolts, as well as thru-holes for multiple sets of integration rods. In addition, a keyway was broached in the servo crosspiece to aid in aligning it with and securing it to the servo driven shaft. As noted previously, the drag tab manufacturing method followed this same path, but 0.25 in. UHMW stock was used instead. One necessary alteration of the tabs was the removal of material on a back corner of each tab, which allows the mechanism to retract completely without the tie rods interfering with the tops of the tabs when fully closed. Because the tabs require higher precision to slide smoothly but without slip, a fixture plate was made to machine the tabs on the router. The fixture had tapped holes to accommodate nylon hold-down screws (in case they were hit by the endmill), and was faced in position before machining the tabs, to ensure the tab stock was mounted on a surface true to the x-y plane of the machine. This allowed the tabs to be machined within a 0.001 in. tolerance in every dimension. Pictures of the manufacturing fixtures used by the team are shown in Figure 102 below.

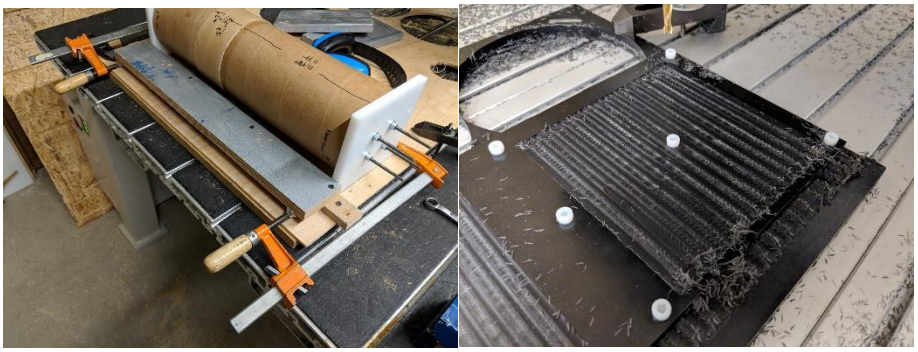


Figure 102. Router securing fixtures for the tab sheet and payload coupler.

Alternatively, the tie rods are made of two components: commercially sourced aluminum hobby rod ends and M3 threaded rod. To fabricate the tie rods, both the rod ends and threaded rod were milled so that the center to center distance of the rod end holes would equal 1.20 in. while both attached rods have their holes in the same orientation, such as in Figure 102 above. After this, all tie ends were secured to the threaded rod using thread locker.

In the mechanism, all connections are as follows. The servo crosspiece is connected to one end of the tie rods by an M3 bolt to a tapped hole in the crosspiece. The other end of the tie rods are connected to the tabs via another M3 bolt to a tapped hole in the tabs. Each tab is secured and surrounded by the top and bottom slide plate, which have a channel for each of the four tabs. Finally, the top and bottom slide plates are held together by nylon bolts.

4.2.4.3 Mechanism Integration

Some minor changes were made to the air braking mechanism during construction. The primary alteration was the removal of material on each tab to allow complete retraction, as mentioned previously, due to interference with the tie rods. This change did not affect the aerodynamic surface or shape of the tab, and will not affect the strength of the part. Additionally,

the anticipated spacers in the manufactured tie rods were removed, as milling down the end of each rod end provided the desired tie rod length of 1.20 in. without the need for spacers.

4.2.5 Electronic Subsystem

4.2.5.1 Servo Motors

Two Power HD 1235 servo motors were chosen to actuate the mechanism. To achieve maximum performance, the motors are operated at their maximum operating voltage of 7.4 V. At this voltage, each motor is capable of producing a maximum of 560 oz-in of stall torque with a maximum current draw of 9 Amps and a speed of 0.18 sec/60°. Operating in tandem, they are capable of producing a maximum torque of 1120 oz-in. The full specification of the 1235 MG servo are listed in Table 40 below.

Table 40. Servo-motor specifications.

Motor	Stall Torque (oz-in)	Operating Voltage (V)	Current at Stall Torque (A)	Speed (sec/60°)	Size (in)	Weight (g)	Cost ($)
Power HD 1235MG (use 2)	560	7.4	9	0.18	2.34 x 1.16 x 2.14	170	$60.00

Figure 103 shows one of the servos outside of the mechanism.

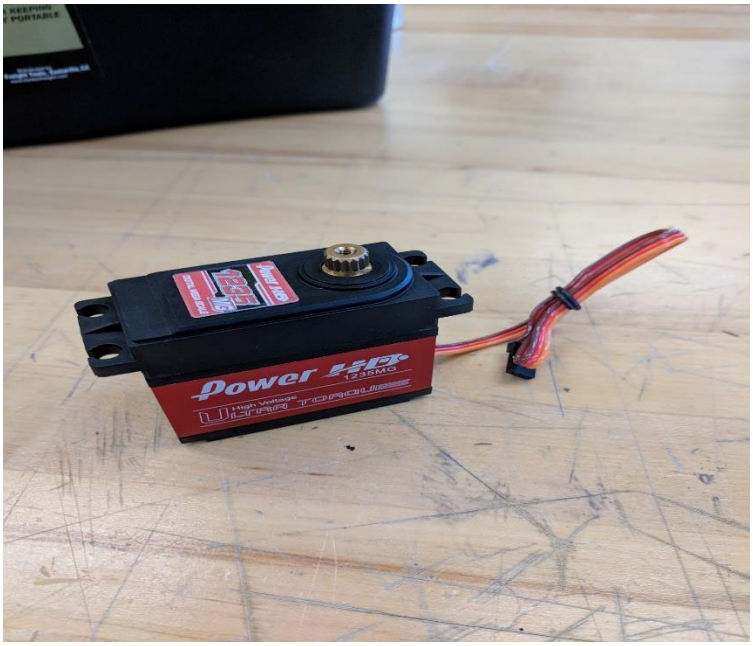


Figure 103. Image of one Power HD 1235 servo motor used to drive the air braking system mechanism.

4.2.5.2 Microcontroller

An Arduino MKR Zero is used to control the air braking system. The Arduino MKR Zero was chosen over other controller options due to better specifications and the advantage of a built in SD card reader that will be utilized to read in an ideal flight path for the PID controller to use and to store data on the performance of the system. The MKR Zero also runs on 3.3 V logic, which allows for less power used by the controller. For full specifications, see Table 41 below.

Table 41. Full technical specifications of the Arduino MKR Zero.

Technical Specifications for Arduino MKR Zero	
Microcontroller	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Operating Voltage	3.3 V
Digital I/O Pins	22
PWM Pins	12
Analog Input Pins	7
Analog Output Pins	1
Flash Memory	256 KB

SRAM	32 KB
Clock Speed	48 MHz
Dimensions	65 mm x 25 mm
Cost	$21.90

Figure 104. Image of the Arduino MKR Zero before integration into the system.

4.2.5.3 Printed Circuit Board

The final printed circuit board consisted of three removable molex connectors and a switch. The switch controls the master power going to the servo motors. In the interest of

preserving battery life, the switch will be flipped to “on” immediately prior to the system being sealed inside of the rocket body. The switch stretches from PCB to the top of the payload using extended wires. An image of the PCB with the molex connectors is shown in Figure 105.

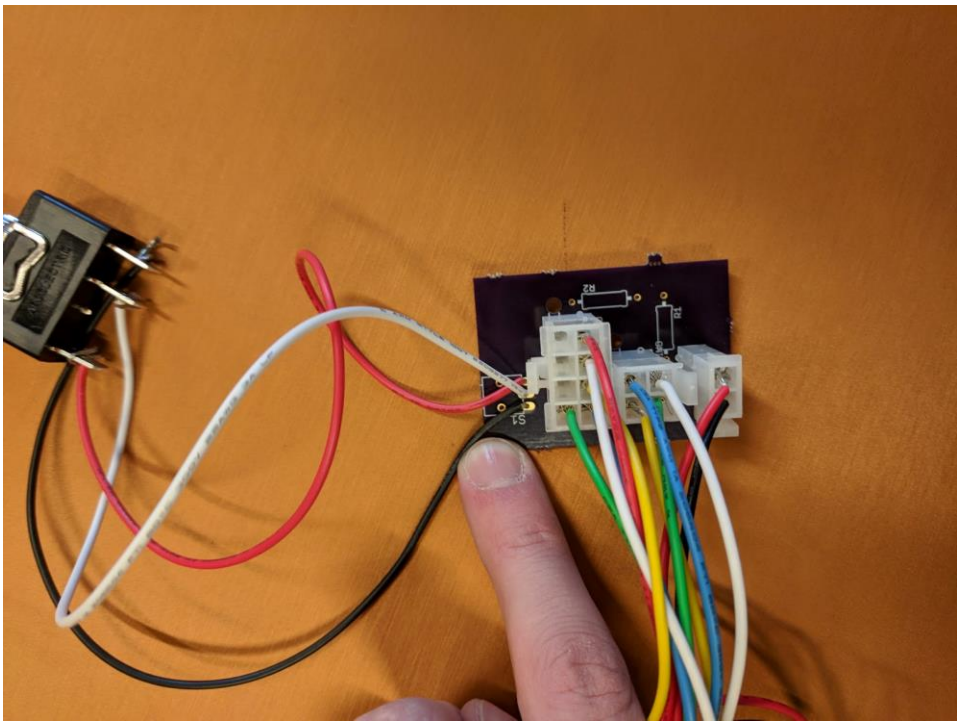


Figure 105. Image of the printed circuit board with molex connectors and the switch.

The first of the molex connectors was an eight port, 4x2 molex. Only five of the connections were utilized in this design. The connector routed power and ground to the servo and potentiometer as well as three signal lines that were relayed to the microcontroller through the PCB.

The second molex connector has four ports in a 2x2 configuration. Each of the four terminals were wired onto specific header pins that interface with the microcontroller. The lines consist of three signals from the servo and potentiometer and a common ground.

The final molex connector on the PCB has a 2x1 port configuration. These ports correspond to the positive and negative terminals of the servo batteries. The two batteries used to power the servo motors were wired together in parallel and then connected to a single molex header to plug into the PCB. When disconnected from the circuit, the batteries can also be charged in parallel using a molex receptacle fit to the charger.

Board design was further simplified by choosing to use separate power sources for the servos and microcontroller. The servo batteries were routed through the PCB while the microcontroller power remained independent. Due to this simplification, no voltage division was necessary to step down power between components on the PCB. The wide traces on the board allow for the relatively high power to go to the the servo and potentiometer, while the only things

traveling to the microcontroller are relatively low powered signals. Without a voltage divider, small strips of wire were used to short across the resistor traces on the PCB. The resulting value of resistance is approximately 0 ohms, not accounting for finite resistance in the wires and traces. Therefore almost all of the voltage from the batteries and signal wires is able to pass through the PCB undivided. A schematic of the PCB layout is shown in Figure 106.

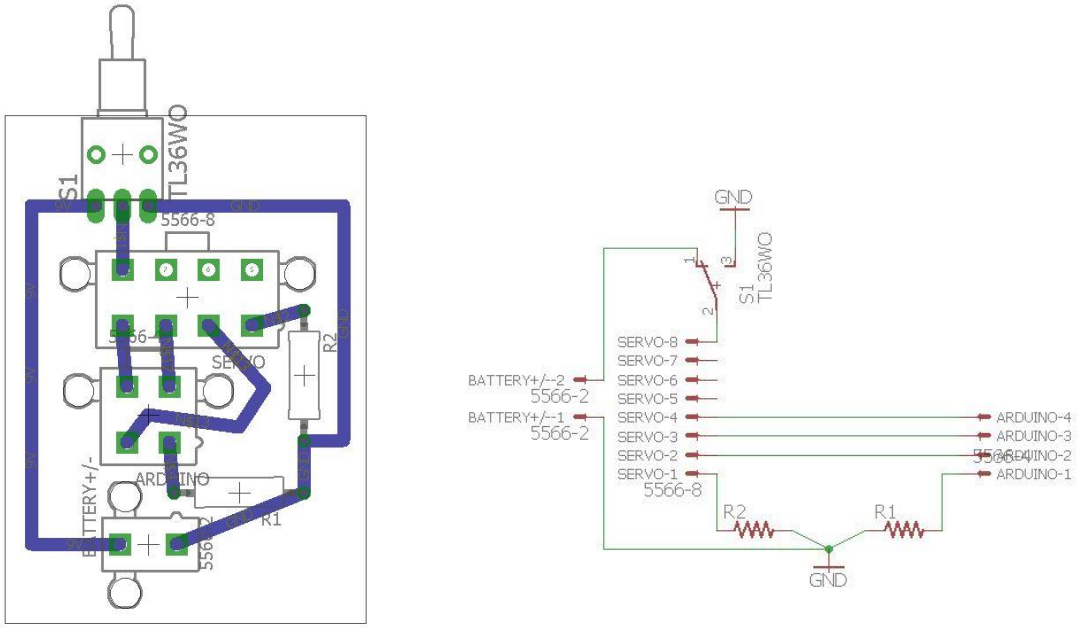


Figure 106. Schematic of printed circuit board used in the air braking system.

Installation of the PCB within the system was eased by use of the molex connectors because the wires and molex headers were run through the various holes and slots cut into the component decks and then plugged into the receptacles on the PCB. To prevent shorts and increase the strength of solder joints, all exposed wires were covered with either shrink wrap or electrical tape.

3.2.5.4 Power System

The batteries used to power the servo motors in the air braking system are two 7.4 V Tenergy lithium ion batteries. The Tenergy battery has a capacity of 2600 mAh and a voltage rating of 7.4 V, which is the operating voltage of the servo motors. At full charge, the voltage across the Tenergy batteries was measured with a voltmeter to be 8.4 V, which is the cutoff voltage at which the charger stops providing charge. The difference in voltage was deemed not to be an issue as servo motor testing found the power supplies are sufficient for the servo motors and excess voltage is dissipated in other circuit components.

A 3.7 V Adafruit lithium ion battery is used to power the Arduino MKR Zero. The battery has a capacity of 2000 mAh which is listed to power the Arduino MKR Zero for roughly

four days, ensuring the control system maintains power. See Table 42 below for full specifications of both the Tenergy and Adafruit Lithium Ion batteries used in operating the Air Braking Payload.

Custom 3D-printed cases were made to: secure the batteries' location during flight, protect the battery in the event of a system malfunction, and contain the battery in the event of a crash, which could result in the damaged battery posing a threat to team members.

Table 42. Technical specifications for both the servo batteries and the microcontroller battery.

Technical Specifications for Air Braking System Power Supplies		
Battery Brand	Tenergy	Adafruit
Intended Use in System	Servomotor	Arduino
Chemistry	Lithium Ion	Lithium Ion
Size (mm)	72 x 38 x 19.5	60 x 36 x 7
Capacity (mAh)	2600	2000
Max Discharge Current (A)	5	0.5
Nominal Voltage (V)	7.4	3.7
Weight (g)	99	34
Cost per Battery ($)	19.99	12.50

Figure 107. Tenergy battery used to power the servo motors.

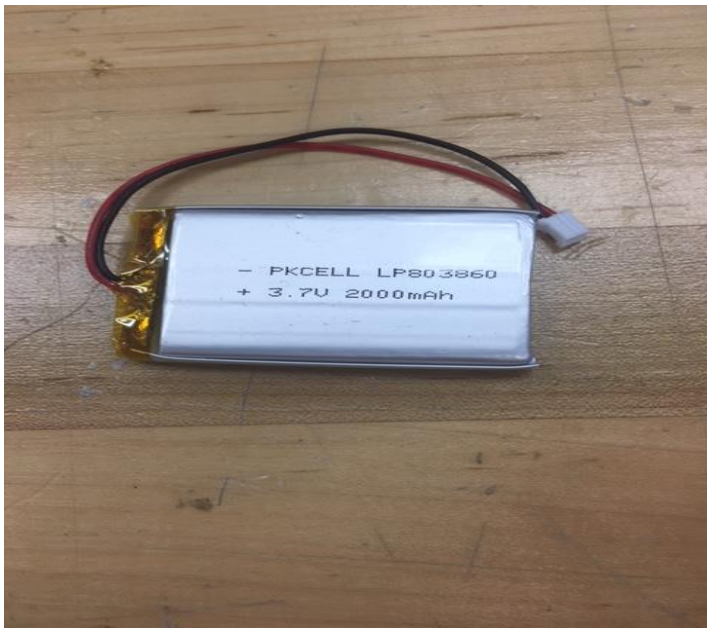


Figure 108. Adafruit battery used to power the microcontroller.

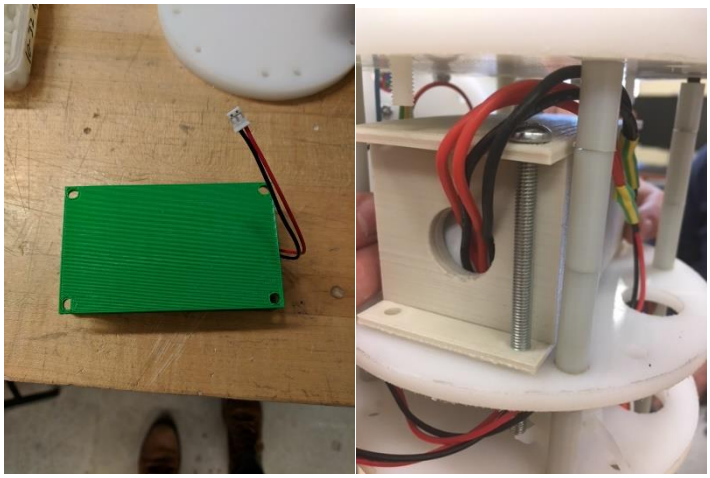


Figure 109. 3D printed battery cases used to secure the batteries in the system.

3.2.5.5 Sensors

The primary sensors utilized in the air braking system are a barometer, the BMP280, and an accelerometer, the ADXL345. The BMP280 was chosen for its high resolution, low noise level, and onboard altitude calculation. The ADXL345 was chosen for its high precision and output rate. While its measurement range is limited to ±16g, previous years' flight data and this year's simulation results indicate that the ADXL345's range is more than sufficient to accurately capture the rocket's behavior throughout flight. The relevant technical specifications for both sensors can be found in Table 43 below, and images of the barometer and accelerometer are located in Figures 110 and 111, respectively.

Table 43. Technical specifications for the sensors used in the air braking system.

Sensor	Resolution	Noise Level	Output Rate	Weight	Size	Cost
BMP280	1.3 cm	11 cm	157 Hz	1.3 g	19.2x18 mm	$9.95
ADXL345	0.004g	0.015g	3200 Hz	1.27 g	25x19 mm	$17.50

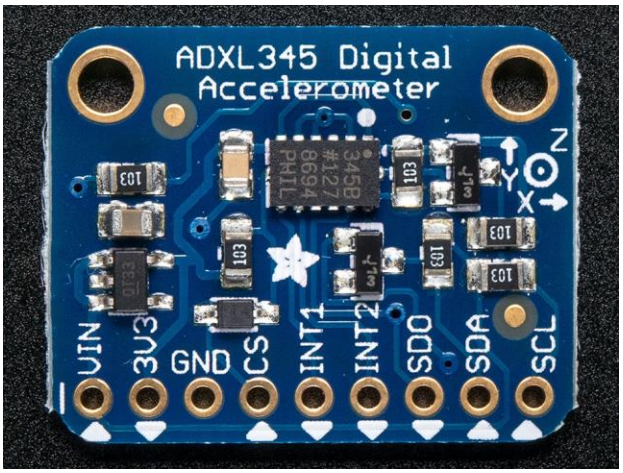


Figure 110. Image of the ADXL345 accelerometer.

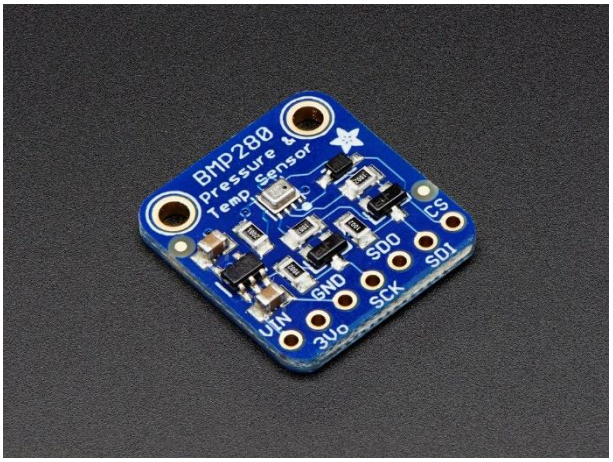


Figure 111. Image of the BMP280 barometer.

Both sensors receive power and communicate with the Arduino MKRZero through an I2C communication bus. Both sensors are necessary to track the rocket's flight progress: the accelerometer can more accurately and rapidly detect liftoff and burnout since both events are marked by rapid changes in acceleration, while the barometer can more precisely detect apogee because that event is marked by a decreasing altitude. The barometer is also utilized to track the rocket's altitude during the coast phase, since velocity at a given altitude is used by the control algorithm as a prediction of apogee. Furthermore, both sensors are used to calculate the rocket's velocity at different points during flight, which is explained in more detail later.

In addition to the accelerometer and barometer, the control code will receive data from a R25W R10K L1% potentiometer attached to the servo gearbox, pictured below in Figure 112. It is wired to the Arduino as a variable voltage divider between its output power and ground read in through an analog port. This sensor will allow the control code to compare the servos' intended position to their actual position, indicating whether or not the mechanism has jammed.

Figure 112. Image of the R25W R10K L1% Potentiometer.

4.2.6 Control Code

4.2.6.1 Code Architecture

The general code architecture begins with an initialization sequence, which occurs immediately after the payload is powered on. Once all of the sensors are initialized and the ideal flight path data is loaded from the onboard SD card, the payload enters a "waiting" state, and the drag tabs are retracted as far as possible. This state allows easy removal of the system from the rocket after a flight is completed. At this point, the code waits for a button accessible from the rocket exterior to be pressed, a task which will be accomplished by the setup team when the rocket is placed on the launch pad. Pressing this button will set the payload to an "armed" state, sending a signal to the servos to fully extend and fully retract the drag tabs, providing a visual confirmation for the setup team that both the control code and the tab mechanism are functioning. If the system is armed unintentionally, the button can be pressed again to return it to the "waiting" state.

Once armed, the code begins to check whether liftoff has occurred, based on an accelerometer threshold of more than 8 g. Once liftoff is detected, the code begins to calculate the rocket's velocity using a linear regression of a 10-point running buffer of barometer data. Additionally, it begins to monitor for burnout based on an accelerometer threshold of less than -1 g.

Once burnout is detected, the drag tab system activates. The control algorithm begins calculating velocity by performing a running Riemann sum of accelerometer data, and compares that velocity to a pre-calculated ideal velocity at the given altitude. This error information is then fed to a PID controller which continuously modifies the servos' position to change the extension of the drag tabs and achieve the desired change in velocity. This process is continued until apogee, detected via a decrease in barometer readings, at which point the drag tab system deactivates and the tabs are retracted until flush with the rocket body. Throughout this process, data from a potentiometer mounted to the gears is monitored to check whether the mechanism

has jammed, and data from all sensors is saved to the onboard SD card for post-flight review. A flowchart summarizing the code architecture can be found in Figure 113 below, and the current control code can be found in its entirety under Appendix K.

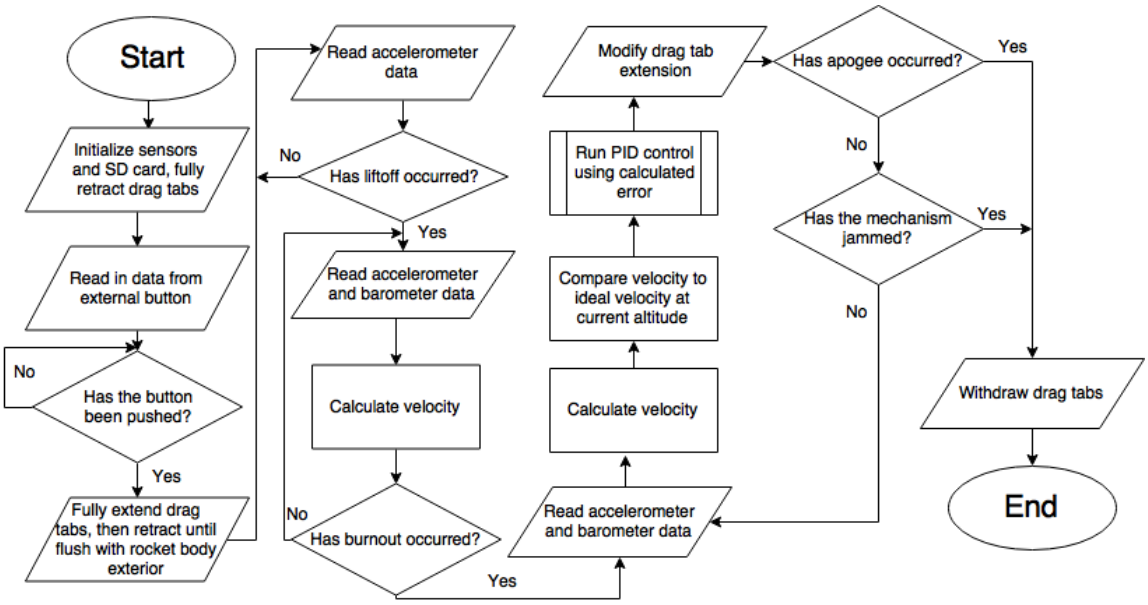


Figure 113. Flowchart of the primary control algorithm.

4.2.6.2 Flight Path Monitoring System

The control algorithm monitors the rocket's flight path using a combination of barometer and accelerometer data, all of which takes place in the main switch-case statement under void *loop()*. During engine burn, a running linear regression will be performed on samples of barometer data to establish a fairly reliable baseline velocity. New data points are added in the *UpdateBaroBuffers()* function, and linear regression is performed in the *CalcBaroVel()* function. Once burnout is detected, accelerometer data will be integrated using a Riemann sum to calculate the rocket's velocity for the rest of flight, which occurs in the *CalcAccelVel()* function.

This "hand-off" method has been demonstrated by the control code simulations to be the most reliable method of obtaining accurate velocity information. Given reasonable sensor noise distributions and a variable launch angle, integrating accelerometer data more closely corresponds to the rocket's actual velocity than differentiating barometer data, and has the added benefit of being less computationally intensive. However, accelerometer-calculated velocity is prone to error during the initial phase of launch; if the integration does not begin precisely at liftoff, or if there is too much vibration-induced noise in the vertical axis during engine burn, then the resulting velocity calculations become highly unreliable because each data point is dependent on the accuracy of the previous data point.

Therefore, the barometer is used to calculate data during engine burn in order to provide a fairly accurate baseline for the accelerometer data to be built upon during coast phase. In order to

track the rocket's flight path, the control algorithm employs this calculated velocity along with altitude data from the barometer. The rocket's vertical velocity at a given altitude is compared to an ideal vertical velocity at that altitude, wherein the ideal velocity is loaded from a pre-calculated dataset with an apogee of precisely 5280 ft. This dataset is loaded in the *ReadBestFlight()* function, and the comparison is performed in the *CalcError()* function. This error, the difference between the rocket's current velocity and the ideal velocity serves as an indirect prediction of whether, and to what degree, the rocket will overshoot or undershoot its target apogee.

4.2.6.3 PID Control

The error value mentioned in the prior section is fed into a PID (Proportional, Integral, and Derivative) controller, which multiplies the error, the derivative of the error, and the integral of the error by separate constants to generate a value for the servo's position. After testing the controller on the physical system and scaling the constants to map to an appropriate range of angles, a C_p of 80, a C_i of 0, and a C_d of 4 have proven the most effective.

However, once an ideal output angle is determined, it is processed through several transformations to account for the limitations of the physical system. Firstly, the servos are configured so that their minimum angle (0 degrees) corresponds to full extension of the drag tabs, while the maximum angle (70 degrees greater than the minimum angle) corresponds to full retraction of the tabs. An additional angle, 60 degrees greater than the minimum, corresponds to a tab extension flush with the body tube of the rocket. Since the PID controller is configured such that larger angles correspond to more extended tabs, the output must first be subtracted from the flush angle to reverse the mapping.

From there, the output is limited by the minimum angle and flush angle in order to prevent the servos from rotating beyond the limits of the tab mechanism. Then, in order to limit the "jitteriness" of the controller, each output is limited to a deviation of 10 degrees from the previous output. All of these transformations occur within the *PID()* function, and once they are complete the angle is passed to void *loop()*. In this section of the code, when the servos are set to the given angle, a final transformation is applied: since the physical system sometimes slips in jam testing so that an angle other than 0 degrees corresponds to full extension, a constant shift value is added to the angle which is sent to the servo.

4.2.6.4 Code Redundancy

Through extensive ground testing and two full-scale launches, the core PID control algorithm has been determined to have robust functionality. Steps have been taken throughout the rest of the control code, however, to provide redundancy and minimize the risk of mechanical or human error interfering with the algorithm's functionality.

To mitigate the effects of mechanical failure, the control code reads data from a potentiometer mounted to the gears of the mechanism to determine if the servos have jammed. The resistance of the potentiometer has been experimentally determined to map linearly to the rotation of the servos, so subtracting and dividing the potentiometer data by two constants allows the actual displacement angle of the mechanism to be determined. This angle is compared to the last commanded angle the servos were set to, and if the difference is outside a threshold of 12 degrees (a value experimentally determined to be sufficient to account for propagation delay given the step size limiter of 10 degrees), a flag is set which communicates that the mechanism has jammed. This process is performed within the *Check_Jam()* function.

Additionally, the control code relies on an exterior-accessible button to mitigate the effects of human error. This button toggles the system between a "waiting" state and an "armed" state before flight. In the "waiting" state, liftoff cannot be detected, and the tabs are fully retracted in case the system needs to be removed from the body tube. In the "armed" state, the control code first fully extends the drag tabs, retracts them until flush with the rocket body, and then begins to check if liftoff has occurred. These visible changes in tab extension allow the setup crew to be aware of the system's state at all times before flight, and assist in indirect error assessment; if the button is pushed and the tabs do not move, then the system is known to be malfunctioning and can be reset or debugged before flight.

4.2.7 Subsystem Integration

4.2.7.1 Electronics Decks

To secure the electrical components, four HDPE decks with a diameter of 5.255 in. were constructed using a Techno CNC router. From top to bottom, the decks house the following components: the sensors, the microcontroller and its battery, the main batteries, and the PCB. The first two decks were cut to a thickness of ⅜ in., and the second two to a thickness of 3/16 in. Figures 114 through 117 below are pictures of said decks without their components.

The sensors and main batteries were placed within 3D-printed cases. These, along with most of the other components, were secured to the decks using 10-32 nylon fasteners. Electronics, such as the PCB, the microcontroller, and the sensors, were secured using #2 screws and locknuts. The decks were integrated into the rest of the payload by sliding them down four 10-32 steel rods that form the skeleton of the payload.

Figure 114. Sensor deck without components attached.

Figure 115. Microcontroller deck without components attached.

Figure 116. Battery deck without components attached.

Figure 117. PCB deck without components attached.

4.2.7.2 Threaded Rods and Spacers

There are a total of four 20 in. long, 10-32 steel threaded rods which extend all the way through the system. These four rods serve the purpose of holding the system together by connecting all of the integration decks in sequence to the tab mechanism. All four rods run completely through each of the four integration decks as well as the mechanism, and the mechanism rests on top of four 10-32 lock nuts which are secured to the rods. The decks and the mechanism are each separated from one another by 1 in. plastic spacers on the rods, shown in Figures 118 and 119, which are cut down to specific sizes to create the necessary amount of room needed for the components on each deck. The mechanism is separated from the PCB deck by 1.57 in. of spacers. The PCB deck is separated from the battery deck by 1.27 in. of spacers. The battery deck is separated from the microcontroller deck by 2.65 in. of spacers. The microcontroller deck is separated from the sensor deck by 1.23 in. of spacers. Finally, the sensor deck is separated from the forward bulkhead by 1.37 in. of spacers.

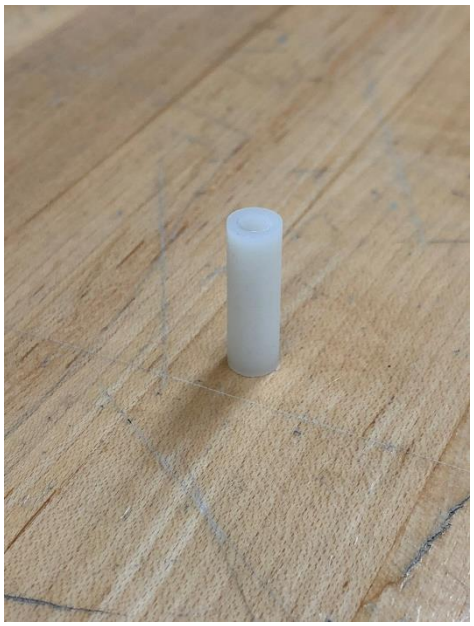


Figure 118. Image of a stock 1 in. plastic spacer.

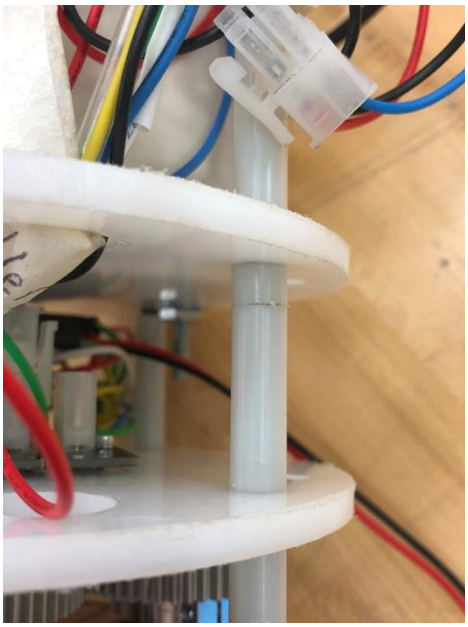


Figure 119. Image of the spacers in use on the integration rods.

4.2.7.3 Vehicle Integration Accommodations

The four vehicle integration rods run completely through the air braking system, and are secured to the bottom of the mechanism and the top of the forward bulkhead. The mechanism rests on top of lock nuts which are secured to the vehicle integration rods, and the forward bulkhead is held in place by lock nuts secured to the integration rods, which prevent the bulkhead from sliding up on the rods.

4.2.8 Testing and Verification

All testing and verification for this system is detailed in Section 7.1.4.

5 Safety

As with every design review, safety will constantly and heavily be enforced throughout the academic year. This is to provide awareness and alertness to the hazards and precautionary measures of sensitive materials, whether it is physical or hypothetical. The team will be enforcing the overall safety of the project by creating a team which derives all levels of safety requirements. This will identify serious and crucial criteria necessary for the team to maintain an acceptable level of safety.

The team will be creating safety requirements that will ensure all aspects of safety are met for NASA's final rocket launch. There will be two categories addressing the necessary safety requirements for the team. Those two categories consist of Human Safety and Environmental Safety.

- The Human Safety criteria includes any and all material, physical or psychological, that will injure the individual or the team. This category is broken up into five (5) separate sections; Vehicle, Payload, Recovery, Air Braking System, and Launch Day.
- The Environmental Safety criteria includes any and all material that affects the relationship between humans and nature. This category consists of two (2) separate sections; Team Effecting the Environment and Environmental Concerns

5.1 Risks and Concerns During Pre-Launch

Knowledge regarding to the launch of the rocket is essential in order to facilitate the team and the overall general public of its safety procedure. For example, based on the FMEA chart, cause and failure mode are both utilized by the team to help reduce all possible hazards that might hinder before and during the launch of the rocket. The team also used both the launch procedures and assembly of the rocket in order to address any and all possible types of hazards before the rover safely deploy.

Most of the information prior to launch stems directly to technical blueprints of each component of the rocket which includes the assembly of the rover vehicle. By exploring each component within the rover, the team addressed a Failure Mode Effect Analysis (FMEA) table to identify the cause and failure mode criteria. Each individual section is outlined to best represent the safety hazards that are probable or not.

5.1.1 Rover Vehicle

The rover vehicle incorporates the deployment aspect of the cause and failure mode analysis. In this case, most of the hazards begin to occur as soon as the rover safety ascends and descends upon launch and as the rover deploys itself from the rocket at ground level. A greater breakdown of what hazards are incorporated into this section are located in the Rover Vehicle FMEA table.

5.1.2 Payload System

The payload system incorporates hazards during assembly, especially as the solar panels and the sub-component electronics are being assembled within the body of the rover. With the complexity of the payload system, hazards associated with launching were broken up into individual FMEA sections within the same table. Also, after the main FMEA table are two focused FMEA tables for the recovery system and the ABP system as those are two of the most hazardous aspects of the rocket. These additional FMEA tables can be seen in Appendices C through E, respectively.

5.2 Safety Officer

The Notre Dame Rocketry Team has chosen Robert Stiller as the Safety Officer for the 2017-2018 year. This is Robert's second year on the team, and he has rocketry construction experience. He is a senior physics major at Notre Dame. Robert will oversee ensuring the team carries out the proper safety procedures and will perform risk and mitigation analysis along with contingency planning for all safety aspects of the project.

The safety officer will ensure that MSDS and other safety documents are up to date and readily available to all team members. MSDS will be heavily emphasized during the 2017-2018 year. The safety officer will be present throughout the construction process and whenever a potential hazard exists for any personnel if the safety officer cannot be present, then he will appoint a capable representative to take his place. All members of the Notre Dame Rocketry Team have signed a safety agreement to ensure safe practices throughout the year and this agreement can be seen in Appendix B.

5.3 Checklist of Final Assembly and Launch Procedures

A detailed pre-launch checklist will guide the final assembly process for the rocket with step-by-step instructions. A repeatable launch procedure will also be developed to mitigate risk of failure at the launch site, and a post-launch procedure will ensure the all personnel retrieve the rocket in a manner that is safe for both the personnel and the rocket. These steps must be followed precisely to ensure successful execution of the project. These procedures can be found in Section 6.

5.4 Preliminary Personnel Hazard Analysis

The Notre Dame Rocketry Team understands that the construction, testing, and launch of the rocket pose several potential hazards to team members. The table below explores the personnel hazards that may occur during different phases of constructing or testing the launch vehicle and its subsystems. Similar to the FMEA table, a severity, likelihood, and overall risk level was assigned to each hazard to better understand what mitigations are necessary. The risks and likelihoods were assessed assuming that all team members have been properly trained, are following the correct procedures, and are wearing the proper personal protective equipment (PPE). By recognizing these hazards now, the team can be better prepared to mitigate them and to take the proper actions in the event that an accident occurs. This table can be found in Appendix F.

5.5 Preliminary Failure Modes and Effects Analysis (FMEA)

A Failure Modes and Effects Analysis (FMEA) table was developed to identify the potential technical failures of the vehicle. For each failure, the effects and causes were identified, as well as their likelihood of happening, and the severity of their occurrence. The last two

parameters were used to assess the risk of each failure through the Risk Assessment Codes (RACs) suggested in the handbook for the competition. The risk matrix used, based on the one shown in the handbook's appendix, is shown in Figure 120.

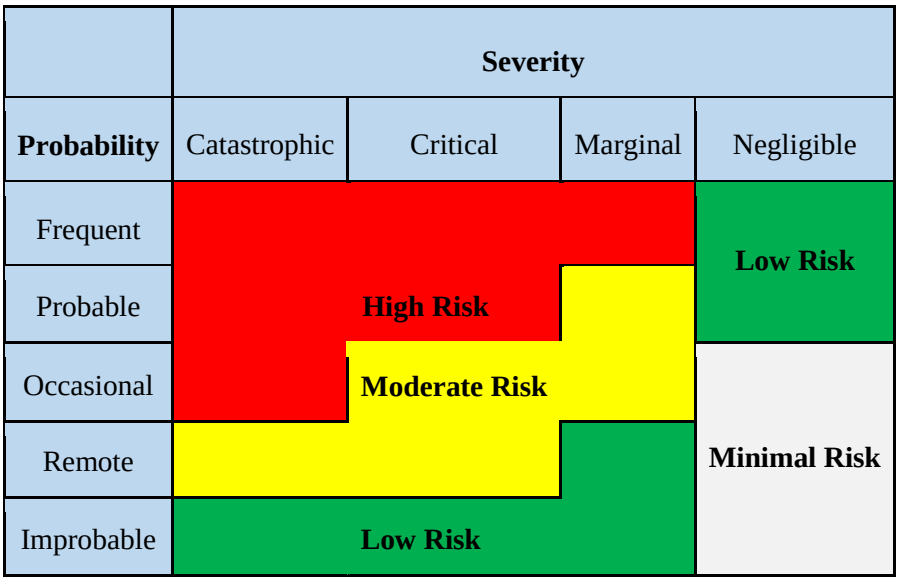


Figure 120. Risk Assessment Matrix

After classifying the risk of each failure mode, mitigations and controls to prevent said failures were developed. It is important to note the importance of first determining the level of risk of each failure as to implement appropriate mitigation levels. Figure 121 depicts how the failure modes were divided into six categories: Structural, Recovers, Propulsion, Stability, and relating to the specific payloads. The FMEA tables for all possible failure modes the launch vehicle and its subsystems may experience can be found in Appendices C through E.

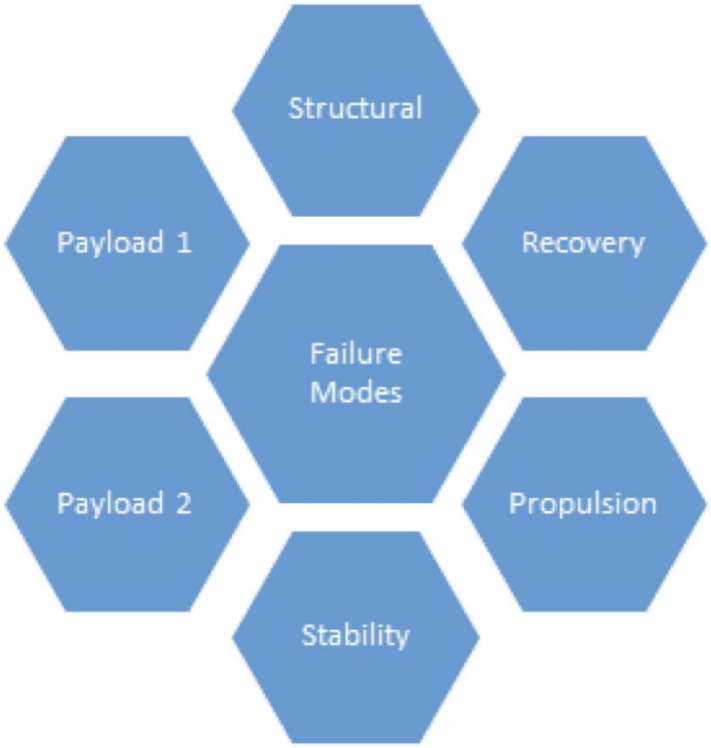


Figure 121. Failure Mode Classification.

5.6 Environmental Concerns

The environment in which the rocket will be operated also poses a certain amount of risk. Specific problems related to inclement weather at the launch and landing sites have been identified and solutions have been devised to decrease the negative effects of the environment on the rocket. Additionally, many of the materials used in the construction of the rocket pose a significant hazard to the environment if they are improperly handled. By considering these things, one can ensure that the rocket is able to adequately perform and not be negatively affected by the environment. Failure modes tables have been constructed for both the environmental effects on the rocket and the rocket's effect on the environment. These tables can be found in Appendices G and H, respectively.

5.7 Project Risks

There is the possibility of encountering a number of roadblocks throughout the rocket design and launch process. Each of these risks has been identified and categorized in terms of their potential impact on the project and the likelihood of that specific problem occurring. Risk is minimized with specific mitigation plans for each scenario. Failure to mitigate these risks will result in significant time delays for the project, which in turn lowers the chance of success on launch day. A table has been constructed outlining potential risks associated with the project,

their likelihood, their impact on the project, and how they will be mitigated. This table can be found in Appendix I.

6 Launch Operations Procedures

The team has developed procedures to follow for every launch, including the test launches and the launch in Huntsville. The launch procedures are to be followed in order to ensure a flawless flight and to help meet the mission success criteria.

Each sub-team has its launch procedures to follow for assembling its payload/sub-systems. Designated members of the sub-team sign off on the launch procedures followed by the leaders of the sub-team to ensure that the procedures were followed correctly.

6.1 Vehicles Design Sub-team

6.1.1 Prior to Departure for Launch Site

Personnel Safety Checks

- Items to bring
    - Safety goggles
    - Gloves
    - Masks
- Ensure lids to epoxy bottles are appropriately sealed

Vehicle components

- Items to bring
    - Nose cone
    - Rover body tube
    - Recovery body tube
    - Fin can
    - Shear pins
    - Tie rod washers
    - Tie rod nuts
    - Tie rod lock nuts
    - Extra epoxy
    - Safety Goggles
    - Electronics
    - Any Arduino connections must be soldered
    - Any batteries must be unplugged to save power.
    - Items to bring (as applicable):
    - Soldering iron, with extra solder
    - Spare batteries

    - Electric tape
    - Extra wire
    - Wire crimpers
    - Wireless Data Receiver
    - GPS Receiver
    - Ground Station
    - Voltage Dividers
    - Microcontroller
    - Sensor Bay
    - Transmitter
    - Ensure the items are stored in safe boxes at a reasonable temperature.
    - Ensure all applicable electronics are turned off.

- Inspect the body tubes and couplers to ensure they have not been damaged during storage.
- Ensure that extra nuts, lock nuts, and washers are packed
- Ensure that wrenches, screw drivers, hammers, wire cutters, and drill are packed in toolbox

Structural Integrity

- Ensure the items are stored in such manner as to not cause physical damage.
- Ensure the fin can is stored on the rocket holder so as not to damage the fins during transportation.
- Perform visual inspection to make sure outer surface has not been damaged during storage.
- Shake the fin can to ensure the payload components do not wiggle when shaken

Subteam Member: _____ Date: _____
Signature: _____
Team Lead: _____ Date: _____
Signature: _____

6.1.2 Prior to Launch

Personnel Safety Concerns

- Ensure everyone operating on the vehicle has proper safety equipment
    - Safety goggles

Prepare the vehicle for launch

- Insert Rover Payload into the top-most body tube, and nose cone on top of that (See rover procedure)
- Ensure CRAM core is inside the CRAM body
    - Process performed by the Recovery Sub-team
- Ensure the CRAM can be armed directly from the rocket's rail position. **Do not arm CRAM before on rail.**

- Attach rocket sections
    - Check that all interfaces are aligned correctly
    - Ensure that body tubes are not tightly fitted
    - Insert shear pins to secure each section
- Perform Cg test to ensure the center of gravity matches the simulated center of gravity. Do this by balancing rocket on rocket stand
    - Add ballast as necessary to keep the stability margin.
- Prepare and insert the motor (Process performed by Team Mentor Dave Brunsting) **Do not attempt any of these procedures without Team Mentor.**
    - Remove motor from packaging
    - Check that motor is properly assembled according to manufacturer's instructions
    - Remove pre-installed ejection charge
    - Properly dispose of black powder
    - Insert motor into casing
    - Ensure two spacers precede motor
    - Screw on rear closure
    - Insert motor into rocket
    - Attach motor retainer
    - Check for secure fit
    - Check rocket stability (at least 1-2 calibers) and final weight
    - Register with LCO and RSO at launch site.
- Ignite motor right before launch (Process performed by Team Mentor Dave Brunsting) **Do not attempt any of these procedures without Team Mentor.**
    - Remove igniter clips from igniter
    - Remove igniter from rocket
    - Ensure igniter has properly exposed ends which are split apart
    - Insert igniter into motor
    - Attach clips to igniter, ensuring good contact

Payload Member: _____ Date: _____
Signature: _____
Payload Member: _____ Date: _____
Signature: _____
Team Lead: _____ Date: _____
Signature: _____

6.1.3 Post Launch

Personnel Safety Concerns

- Instruct all personnel to get clearance before starting recovery process
- Assess there is no harmful physical damage before removal. **If there are discrepancies, ask the Range Safety Officer to confirm it is safe to proceed.**

- Ensure nothing is on fire. **If anything is on fire, immediately contact the Range Safety Officer.**
- Verify that all black powder charges have been ignited or have been disarmed
- Wait for at least 5 minutes before removing due to lingering motor heat. **Confirm motor is a safe temperature by placing a hand near the motor.**
- Instruct all personnel not let the fin can safely land before approaching.
- Instruct all personnel not begin recovering the payload until given clearance by ground personnel.
- Ensure the fin can has adequately cooled before handling. **Confirm the find can is a safe temperature by placing a hand near the fin can.**
- Document the state in which the system is before any tampering
- Check that all electronics survived the flight intact.

Structural Integrity

- Document state of rocket before removing by taking photos
- Check the physical state of the overall body tube
- Check the physical state of each payload
- Recovery: _____
- Deployable Rover: _____
- Air Braking system: _____
- Nose Cone: _____
- Fin Can: _____
- Perform visual inspection to make sure outer surface has not been damaged during flight. **Make sure all rocket sections are a safe temperature to handle and that there is no battery acid on any materials before handling them.**
- Assess any damages that may have occurred during operations. **Pay particular attention to areas of the rocket near the cram and any places where there are batteries.**
- Determine if the damages are severe enough to prevent additional launches. Repair any minor damages, where possible.
- After recovery, re-perform component tests to ensure that operation has been uninhibited.

Recovery

- Gather all rocket sections and return to assembly area. **Make sure all rocket sections are a safe temperature to handle and that there is no battery acid on any materials before handling them.**

Subteam Member: _____ Date: _____
Signature: _____
Team Lead: _____ Date: _____
Signature: _____

6.2 Recovery Sub-team

6.2.1 Prior to Departure for Launch Site

Personnel Safety Concerns

- Items to bring
- Safety goggles
- Electronics
- Items to bring
- CRAM
- Main parachute
- Drogue Parachute
- Shock cords
- Shear pins
- Extra batteries
- Talcum powder
- Ensure the items are stored in safe boxes at a reasonable temperature.
- Ensure all applicable electronics are turned off.

Structural Integrity

- Ensure the recovery body tube has not been damaged during storage.
- Ensure the holes in the recovery body tube are the appropriate size
- Ensure the recovery body tube is clear of electronics before storage.

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.2.2 Prior to Launch

Personnel Safety Concerns

- Ensure everyone operating on the payload has proper safety equipment
    - Safety goggles
- Prepare CRAM
- Insert fresh batteries into CRAM core
    - Ensure batteries are connected to altimeters by listening for beeps from altimeters

- Insert CRAM core into CRAM body
- Put CRAM core cover on
- Tighten nuts down onto cover
- Insert long eyebolt through center of CRAM
- Place washer against both the bottom bulkhead and the CRAM cover
- Tighten nut against CRAM cover to hold bolt in place
- Connect the wires from CRAM core to screw terminals
- Attach short eyebolt to the long eyebolt with coupling nut
- Tighten nut on either side of coupling nut

Electronics

- Prepare Avionics
- Mark the Primary Raven as official contest altimeter
- Ensure arming switch is "safe"
- Properly secure altimeters and batteries
- Install the CRAM until it locks
- Ensure the CRAM can be armed directly from the rocket's rail position

Structural Integrity

- Prepare ejection charges
- Ensure personnel are wearing safety glasses
- Move all non-essential personnel away from rocket
- Connect electric matches/ejection charges to altimeter
- Properly load and prepare parachutes
- Check that shroud lines are not tangled
- Apply talcum powder to each parachute
- Ensure that shock cord is not tangled
- Insert parachutes, chute protector, and shock cord into rocket
- Attach rocket sections
- Check that all interfaces are aligned correctly
- Insert shear pins to secure each section
- Ensure tight fit of all components
- Leave hatched door open
- Check shock cord for brittleness
- Replace shock cord that appears brittle

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.2.3 Post Launch

Personnel Safety Concerns

- Instruct all personnel to get clearance before starting recovery process
- Assess there is no harmful physical damage before removal
- Ensure nothing is on fire **If anything is on fire, immediately contact the Range Safety Officer.**
- Check that ejection charges have ignited **If any ejection charges did not go off, immediately get Team Mentor and Range Safety Officer.**
- Document state of rocket before removing

Electronics

- Disarm altimeters
- Disconnect batteries **Check for battery acid before handling batteries.**

Structural Integrity

- Check the physical state of the recovery body tube **Make sure all rocket sections are a safe temperature to handle and that there is no battery acid on any materials before handling them.**
- Is it re-usable?
- Check that components are safely inside the payload

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.3 Air Braking System

6.3.1 Prior to Departure for Launch Site

- Confirm full assembly of air braking mechanism. This includes:
    - Drag tabs, tie rods, crosspiece are mounted in the sliding plates
    - Drive shaft is mounted. Both snap rings are in place
    - Servos are mounted
    - Gears are mounted and secured to drive shaft and servos

    - Potentiometer and potentiometer gear are mounted
- Confirm full assembly of electronic subsystem. This includes:
    - PCB is mounted to PCB deck
    - Servo batteries are fully charged the night before (connect to charger to confirm)
    - Servo batteries are placed in their battery case and the case is secured to the battery deck
    - Microcontroller is secured to the microcontroller deck
    - Microcontroller battery is placed in its case and the case is secured to the microcontroller deck
    - Accelerometer, barometer, switch, and bluetooth receiver are secured the the sensor deck
- Confirm full assembly
    - Threaded rods are run through the mechanism and all decks
    - Spacers are placed on threaded rods between decks
    - Locknuts are secured on threaded rods directly below mechanism
    - Ballast is secured below the mechanism
    - Forward and aft bulkheads are secured with locknuts

Equipment Needed for Launch Site

- Allen key set
- #2 locknuts
- #10 locknuts
- 10-32 nylon screws, 1.5" and ⅝"
- 6-32 steel screws, ⅜"
- 6-32 nylon screws, ¾"
- #2 screws
- M3 screws, 14 mm
- Wrench set
- Solder iron
- Wire
- Heat shrink/electrical tape
- Battery charger
- Extra Arduino, barometer, accelerometer, PCB, servo battery, and microcontroller battery

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.3.2 Prior to Launch

- Confirm security of screws in mechanism
- Confirm security of wiring connections
- Place air braking system in coupler
- Activate system. Confirm operation of the mechanism and arming of the system
- Monitor payload for post burnout deployment - if this occurs press exterior button

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.3.3 Post Launch

- Check for any visual signs of hazards, particularly smoke or fire coming from the system
- Before touching the system, place hand near the system to determine if there is any extra heat in the system
- After the motor casing has been taken out of the fin can, remove the lock nuts from the forward bulkhead of the system
- Press the arming button to retract the tabs inside the rocket body
- Pull the system out of the fin can. Turn off the servo battery and unplug the microcontroller battery. Check that all components remained secure during flight
- Check for structural damage to the system
- Remove the SD card from the microcontroller and download the flight data

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.4 Deployable Rover Payload

6.4.1 Prior to Departure for Launch Site

- Check that all electronics are powered off.

- Pack the rover vehicle.
- Pack the ground station.
- Confirm the nose cone and rover body tube is packed with the vehicles.

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.4.2 Prior to Launch

- Ensure that the payload system is intact post travel.
- The deployment electronics inside the launch vehicle will be turned on prior to loading the rover.
- The rover will be loaded inside the launch vehicle and secured inside the securing blocks.
- Perform black powder and deployment testing.
- The ejection charges will be loaded into the payload a safe distance away from the rest of the rocket (50 feet) with a shunt pin placed to prevent arming until needed. The black powder will be handled by the certified team mentor.
    - When the ejection charges are loaded, the body tube will be pointed away from all people at the launch site and remain that way until taken to the launch pad.
- The nose cone will be placed on the rover body tube.
- The shear pins will be inserted into the nose cone and body tube, closing off the payload.
- The rover portion of the launch vehicle will be connected to the rest of the launch vehicle.

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.4.3 During Launch

- Once the vehicles team loads the rocket on the launch rail, the deployment system will be armed by removing a shunt pin.
- First person signs off on the arming.
- Secondary person signs off on the arming.

- Upon safe landing and once the field has been cleared for retrieval, the rover deployment team will approach the launch vehicle.
    - **The rover body tube is armed with live ejection charges, this section must be approached from behind, no team member will walk in front of the nose cone.**
    - The launch vehicle will not be touched or moved until the rover sequence is complete.
- The designated videographer will begin recording the deployment prior to any action.
- The ground station manager will ensure that no one is in the firing range (500 feet) of the nose cone.
- The ground station manager will announce loudly that the deployment sequence is about to begin.
- The ground station manager will count down from five and on the count of one will initiate the deployment sequence.
- Once the deployment sequence has begun, no team member will approach or touch the rover or launch vehicle.
    - If both black powder charges do not go off begin the disarming sequence.
- After the rover has stopped moving and the solar panels are deployed the rover recovery team will collect the nose cone, rover and rover body tube.
- Any debris from the rover deployment will be collected.
- Once the rover sequence is completed entirely, the rest of the launch vehicle and recovery system will be collected.

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

6.4.4 Post Launch

- The rover will be inspected for any damage
- Any minor damage will be fixed on site.
- The securing system will be inspected for any damage.
- If any damage, the rover will not be eligible for relaunch on the same day.
- The deployment system will be inspected for any damage.
- Any minor damage will be fixed on site.
- If any major damage, the rover will not be eligible for relaunch on the same day.
- The deployment video will be watched and analyzed for future improvements.
- The electronic data will be analyzed for future improvements.
- If the rover is launching a second time, begin pre-flight launch procedures.
- If final launch of the day: pack the rover, nose cone and rover body tube.

Squad Member: _____ Date: _____

Signature: _____

Team Lead: _____ Date: _____

Signature: _____

7 Project Plan

7.1 Testing

7.1.1 Vehicles Design

In order to verify that the design of the launch vehicle is sound, multiple tests were performed on sections of the rocket. These test will be run on the materials being used for the full scale rocket, as well as on the full scale rocket itself.

7.1.1.1 Subscale testing

Since it is unreasonable to do all testing on the full scale rocket, a subscale rocket was built and tested. Testing only the full scale is unreasonable due to factors such as time, budget, and testing apparatus constraints, and for this reason, a smaller rocket made of cheaper materials was made. Details of its design and launch can be found in Section 3.3.1.4.2.

Subscale Flight

As discussed in Section 3.3.1.4.2, the rocket underwent two flights on December 2nd, 2017 in Three Oaks, Michigan with the Michiana Rocketry Club. The objective of this test was to gather altimeter data to compare with simulation packages as well as test the effectiveness of the Air Braking Tabs when fully deployed. The results of this test are discussed in Section 3.3.1.4.2.3, where it is mentioned that the simulation packages overestimated the apogee by approximately 60 ft, and the Air Braking Tabs did indeed have a significant impact on the flight.

Wind Tunnel Testing

The subscale rocket was also subjected to wind tunnel testing in the Hessert Laboratory on Notre Dame's campus on November 9th, 2017. The purpose of these tests were to derive drag coefficients for the subscale rocket with and without drag tabs deployed, as well as verify initial CFD findings that there is no separation of the boundary layer aft of the transition section. The results of the testing can be found in Section 3.3.1.4.1, where it was revealed that an instrumentation error caused no velocity data to be gathered, and as such no coefficients of drag

could be found. It was, however, verified that there was no separation due to the magnitude of the drag forces on the aerodynamic structure of the rocket.

7.1.1.2 Software

Another way of testing the rocket while being cost and time effective is to use software packages to predict the performance of the rocket. An abundance of these programs are used to predict the flight, aerodynamics, and structural integrity of the design.

Flight Simulations

The rockets initial design occurs in flight simulation software. The Notre Dame Rocketry Team primarily uses the packages OpenRocket and Rocksim to predict the flight of the rocket, and the rocket is entirely modeled in these. The objectives of the tests run in these packages are to predict the apogee, velocities, accelerations, and flight times of the final rocket, as well as gain data on the physical nature, such as center of gravity and center of pressure. The program results are compared to one another to verify that the team is getting reliable results.

Testing in these packages has been ongoing since September 2017, and new simulations are run with each change in the rocket. During construction, each component of the rocket was weighed and its position was accurately measured so that the models in OpenRocket and RockSim are as accurate as possible. As discussed in Section 3.3.2.2, the simulations were able to predict the apogee of the full scale rocket within 20 feet. It was in this section that the differences were attributed to slight differences in the CG due to packing densities, which allow the team to put faith in the simulations that they will correctly predict the performance of the rocket.

Computational Fluid Dynamics

Initial CFD analysis began in September 2017, when it was decided that a variable diameter rocket was going to be used for the full scale vehicle. ANSYS AIM Student was used to simulate the environment around the rocket during a 200 m/s flight. This test was run with forward diameters ranging from 5.5” to 8.5”, and the boundary layer of each rocket was observed. None of these tests revealed any separation of the boundary layer or any excessive turbulence, and therefore the design was able to move forward.

Further CFD with the Notre Dame Center for Research Computing was begun in November 2017 to further investigate the validity of these findings, and are currently underway. However, this additional testing is being carried out for the purpose of teaching team members how to utilize it for future years.

FEM Analysis

Analysis of the structure of the rocket is paramount to a successful flight. In order to do this, the Finite Element Method software ADINA was used in order to verify that the structure of the rocket will hold up to the forces experienced during flight. Additionally, the connections between sections during separation and descent are being looked at to make sure that they are able to handle the stresses of the controlled black powder explosions, as well as the stresses from the parachute and shock cords upon parachute deployment.

These tests show that the minimum factor of safety during the mission is 41, and this was located at the joints between the stress bearing eyebolt bulkheads and the phenolic body tubes. This analysis led the team to the conclusion that the rocket was structurally stable under the loads experienced during flight and recovery. These values from ADINA were double checked with hand calculations which can be found in Sections 3.1.6.1.2 through 3.1.3.1.4.

7.1.1.3 Physical Testing

Once the materials for the full scale were in the team's hands, physical testing began on them. This is to ensure that manufacturer data is accurate, as well as ensure that the designed rocket is able to withstand some of the events that may happen during flight.

Laboratory Stress Testing

Stress testing was scheduled to be done in February 2018 in Notre Dame's Fitzpatrick hall. However, due to the change in material and quick turnaround from reception of materials to launch, stress testing was not able to be scheduled with the operators of the laboratory. However, testing on phenolic body tubes and couplers has been performed by the team in years past, and the team is confident in the quality of these tests and in the quality of the material purchased from Apogee Components.

Component Testing

Component testing was performed on multiple parts of the rocket to ensure that integration and retention were designed to specifications that would ensure a successful flight. These tests were performed the day of the test launch, March 3rd, 2018 at the launch site in Three Oaks, Michigan.

Test for different components and their results can be found in Table 44.

Table 44. Component Test Results.

Component(s)	Purpose of Testing	Testing Method	Results
Motor Mount/Retention	Ensure that the motor mount was properly epoxied to the fin can, and that the motor retention does not unscrew with the vibrations experienced during flight	-Motor casing was placed in the rocked and the retention was screwed on. The rocket was then held and vigorously shaken by a team member in 3 dimensions to ensure that vibrations were tested in every direction. -Full Scale Test Launch on March 3rd, 2018	The motor mount and retention were both deemed suitable for flight, as they were not damaged, did not shift, and did not screw off during or after the vibrations.
Fins	To ensure that the epoxy fillets on the joints of the fins and body tube/motor mount are able to retain the fins. Also to ensure the strength of the fins.	-The fin can was held by two fins and shaken vigorously by a team member in 3 dimensions. This was done with both sets of fins. -Full Scale Test Launch on March 3rd, 2018	None of the fins in the fin can shifted or were warped in any way due to the testing. This verifies that the fillets connecting the fins are suitable, as well as verifies that the fins are of suitable strength for flight
Transition Section Connection	To ensure that the transition section is firmly attached to the fiberglass body tube, and will not separate during flight.	-The full forward section of the rocket (fiberglass body tube, transition section, phenolic body tube and coupler) was held vertically by a team member and shaken vigorously in 3 dimensions.	The transition section did not shift in any way relative to the body tubes, and there were no cracks in the epoxy at the joints between the transition and body tubes, verifying that the connection between these components is

		-Additionally, the connection points were impact tested with a hammer to simulate the impulse from separation and landing. -Full Scale Test Launch on March 3rd, 2018	strong and able to withstand the forces of flight, separation, and landing.
Ballast Integration	To ensure that the ballast placed in the Air Braking System does not shift in flight and create instabilities.	-The bag of ballast was placed between two bulkheads in the Air Braking System and tightened (see Section 3.1.5.2). The payload was then shaken vigorously by a team member to simulate vibrations during flight.	The ballast in the payload did not shift axially or radially during the test, verifying that the mass remains in the same place during flight and does not cause any instabilities.
Air Braking System	To ensure that the Air Braking System is firmly secured to the fin can, and that it does not come loose during flight, causing a misalignment of the fins.	-The payload was placed on the fin can tie rods and tightened using nuts and lock nuts. The payload was then shaken vigorously in 3 dimensions by a team member to simulate vibrations during flight. -Additionally, the payload was impact tested using a hammer to ensure that the payload's drag tabs do not become misaligned. -Full Scale Test Launch on March 3rd, 2018	The payload did not shift during testing and remained firmly secured to the fin can's tie rods. Additionally, the drag tabs remained in perfect alignment with the tab slots, verifying that they will jam during flight due to a misalignment with the fin can.

Recovery System	To ensure that the Recovery System does not come lose during flight or separation, which can cause critical failures in the structure of the rocket.	-The recovery payload was loaded into the recovery body tube with shock cords attached. The payload was then shaken vigorously in 3 dimensions by a team member. Additionally, the shock cords were pulled by team members to simulate the impulse during separation. -Full Scale Test Launch on March 3rd, 2018	The recovery payload did not shift during the shake test, and all components remained in their initial positions during the impulse test. This verified that the system will not cause any stability or structural issues for the rocket during flight and separation, respectively.
Deployable Rover Payload	To ensure that the rover and nose cone remain in their initial position during flight, during separation, and upon landing.	-The rover was placed in its position in the rover body tube, and the nose cone was attached with shear pins to the tube. The section was then shaken vigorously in 3 dimensions to simulate flight. -The section was impact tested axially with a hammer to simulate separation and landing. -Full Scale Test Launch on March 3rd, 2018	Neither the rover nor the nose cone broke free or shifted during the shake testing. This verified that the components could withstand the forces of flight. However, while impact testing with a hammer was successful, the nose cone separated from the payload during drogue deployment in the Full Scale Test Launch. This is leading the team to further black powder testing to ensure that the nose cone will remain attached until

			deployed upon safe landing.
Fully Assembled Rocket	To ensure that the rocket will not prematurely separate during flight due to vibrations.	The rocket was fully assembled with motor and shear pins, then rotated upright and shaken vigorously by multiple team members to simulate vibrations during flight.	The rocket did not separate and no shear pins were broken during the shake tests. However, the rocket completely separated at apogee, causing the drogue and main parachutes, as well as the nose cone, to deploy at apogee. This is leading the team to further black powder testing to prevent any premature separations.

= Successful Test

= Mixed Results

= Unsuccessful Test

These component tests led the team to multiple conclusions. First, the payload integration into the body of the rocket is formidable, and can withstand the forces experienced during flight. Secondly, it was revealed during the full scale test flight that additional black powder testing is required to ensure that the vehicle's sections separate when designed to. It was determined after the test launch that the amount of black powder in the recovery section was increased after black powder testing at the recommendation of the team's mentor. This increase was untested on the ground, and resulted in the full separation of the rocket upon drogue deployment at apogee. Further black powder testing will be carried out on Thursday, March 8th, 2018, where the number of shear pins and amount of black powder will be determined.

7.1.2 Recovery System

Some of the first recovery system tests performed were those dealing with e-matches and altimeter output. These were developed incrementally, beginning with simple direct e-match contact to 9V batteries. Next the altimeter functionality and circuitry was verified with a prototype board/LED simulation. Once this was shown to be successful, the tests graduated to e-

match testing combined with altimeter simulation. Table 45 below shows the results of the various tests conducted. Figure 122 shows an LED simulation, and Figure 123 shows an altimeter simulation/e-match test being performed.

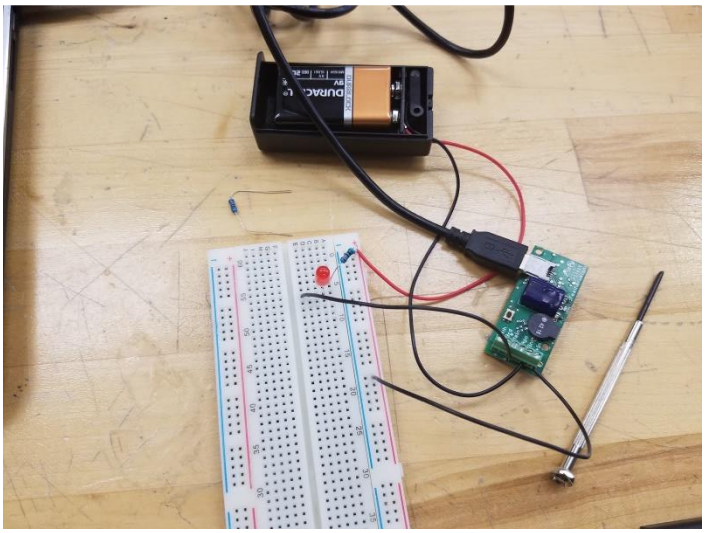


Figure 122. Altimeter simulation testing, wired to LED for circuit verification.

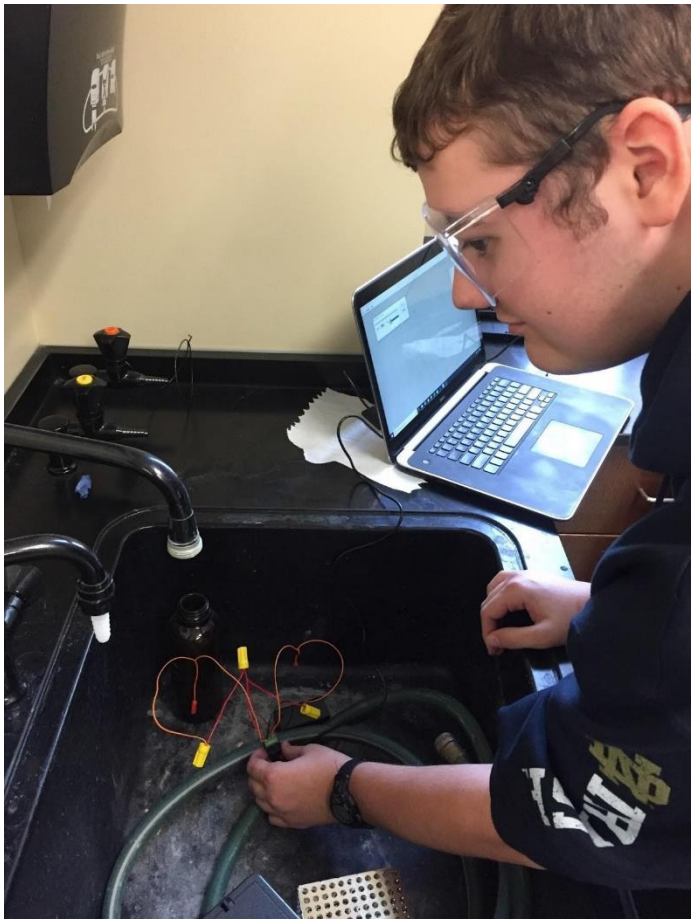


Figure 123. E-match testing, connected to altimeters simulation software.

Table 45. E-match verification testing.

Trial Type	Number of Trial Attempts	Number of Trial Successes
Battery directly to e-match	10	10
Altimeter simulation to LED	10	10
Altimeter simulation to e-match	9	10

As evidenced in the table, the e-matches and altimeters performed almost perfectly in all tests. Because the e-matches have been purchased new this year and tested rigorously, the team is confident that they will not be an issue in launches moving forward.

A new test to the recovery system this year was an altimeter arming test. The purpose of this was to calibrate the timing of a procedure which could ensure the audial feedback from the altimeters did not interfere with each other before launch. This was done by assuming an equal

boot-up time for each altimeter and then measuring the time between cyclical arming "beeps." The period was measured to be six seconds, so the three altimeters were power-on at two second intervals. When this was complete, three distinct arming sounds could be perceived with no interference and the test was ruled a success.

Another new test to the recovery system this year was a connectivity/shake test performed after assembling the CRAM in its entirety for the first time. Following the cadence of the previously described test, the altimeters were powered on and the bare wire leads from the altimeters were crossed to safely simulate the connection of an e-match. When the arming sounds commenced, the CRAM was shaken vigorously by a team member to simulate a very rough flight pattern and the effects of a black powder ignition. No amount of shaking was able to jar loose the internal connections, so the test was ruled a success.

Another important test performed by the recovery system was the black powder/shear pin combination. These were performed in the field prior to the first test launch. The CRAM was rigged with the desired amount of black powder and then the e-match leads were fed outside the rocket through the battery switch holes in the body tube. The assembled rocket was then perched slightly above the ground to simulate the conditions of free fall as closely as possible. Then the long wire leads attached to the e-match were touched to a power source to separate the sections. The black powder testing setup is shown below in Figure 124.

Figure 124. Black powder/shear pin ground testing.

The first test was for the main parachute section and included 4g of black powder. Upon ignition, the sections separated, but not with significant force such that the main chute was

deployed from the body tube. The second test was for the drogue and also involved 4g of black powder. Upon ignition, the sections separated a satisfactory amount. The third test was for the main parachute and involved 5g of black powder. Upon ignition, the sections separated a satisfactory amount.

The ultimate test of the recovery system robustness was the actual set-up and launch of the full-scale rocket. While no major failures occurred, there were a few hiccups which were observed and will be improved upon moving forward. The first of these was the unanticipated difficulty in arming the altimeters as planned. The standing height of the rocket combined with the added height of the launch pad made the arming switches accessible only by ladder, and made the process much more cumbersome. As a result, the arming cadence required to achieve ideal audio feedback was disrupted and the final altimeter connection "beeps" were nearly indistinguishable from the others. To correct this issue, a new arming procedure will be developed that is more adaptive to the difficulties of real launches.

The second complication occurred in flight when both the main parachute and drogue deployed at apogee. The theorized cause of this error was the amount of black powder used for drogue deployment. Since the ground testing results were not exceptional, additional black powder was added to each charge to ensure deployment. Specifically, 6g per drogue charge and 7g per main charge were used. It is likely that the impulse of the blast caused a reaction force on the fin can side of the rocket which pulled hard enough to sever the main section shear pins and deploy the chute prematurely. To correct this issue, further ground testing will be performed until the delicate balance between shear pins and black powder is met perfectly.

7.1.3 Deployable Rover Payload

7.1.3.1 Component Testing

All motors tested together, with and without wheels

The motors were tested with the driver boards shown previously. Applying a PWM signal of various duty cycles, the speed of the motors was able to be controlled using the PIC and the driver board. Code was then written to move all of the motors at the same time. This was implemented and it was verified that the rover could be driven at various speeds.

Servo motor

Using the OC2 register, an extremely precise PWM signal was used so that the servo motor would not get an undefined signal. However, the servo, as ordered, could only rotate 120 degrees. Therefore, it was necessary to modify the servo for continuous rotation. This was accomplished by opening the servo and removing the mechanical stops. After testing this configuration, the servo was still behaving erratically, and it was determined that the built in

potentiometer used to find the center position was also preventing continuous rotation. Therefore the potentiometer was removed, allowing the servo to move as far as needed.

Each sensor tested individually before placed on the rover

After the board was constructed, each sensor was methodically tested. Firstly, after the PIC was constructed it was verified that the device ID could be found using the PICkit3. Next, it was ensured that the device could be programmed. RE5 was chosen, and a simple program was written to toggle the output of RE5. This output was verified using a SALEAE logic analyzer.

The second test was to ensure that the GPS module was working as expected. A logic analyzer was connected to both the RX and the TX pins of the GPS, and it was ensured that the GPS was transmitting correctly. Later in testing, after the UART to print to a computer was working correctly, the GPS was read continuously and its output was printed to the terminal. This ensured that both the GPS and the MUX worked as expected.

After the MUX and the GPS were tested, test code for each of the other sensors was written. First, it was ensured that the LiDAR worked properly. After test code was written, the accuracy of the sensor was established. While the datasheet suggests that the sensor should get accuracy within 1 cm, the accuracy of the sensor was only able to be established within approximately 2 inches. Nevertheless, for our purposes this should be acceptable error.

Next, the altimeter was tested. Using code repurposed from the LiDAR testing, the altimeter was established as receiving accurate data. While it was not able to be compared to the exact altitude above sea level, it was close enough to the values that one would expect for South Bend, and it was verified for differential altitude. Therefore there is reason to believe that it is working as expected.

Finally, the GAM (gyroscope/accelerometer/magnetometer) was tested. Using repurposed I2C code, it was established that the GAM was able to adequately establish orientation data.

7.1.3.2 System Testing

Securing System→ shake test

Upon final construction of the payload the securing system was tested via a shake test. This test was critical, as any movement of the rover could affect the flight path of the launch vehicle. Furthermore, if the rover does not stay in place throughout the flight, there is risk of the nose cone falling from the launch vehicle without recovery. The first step of the shake test involved placing the rover in the rover body tube and locking it in place with the solar panel racks. This can be seen in Figure 125. The body tube was then inverted and the position of the rover was noted. Since no movement or shifting was recorded the body tube was then agitated to

simulate vibrations during flight. Again there was no movement or shifting of the rover. Figure 126 shows the set up and testing of the securing system.

Figure 125. The actuation of the racks into the securing mount was tested to ensure the alignment of the mounts match the alignment of the solar panel racks. This also tested the servomotor's dual functionality as the securing system.

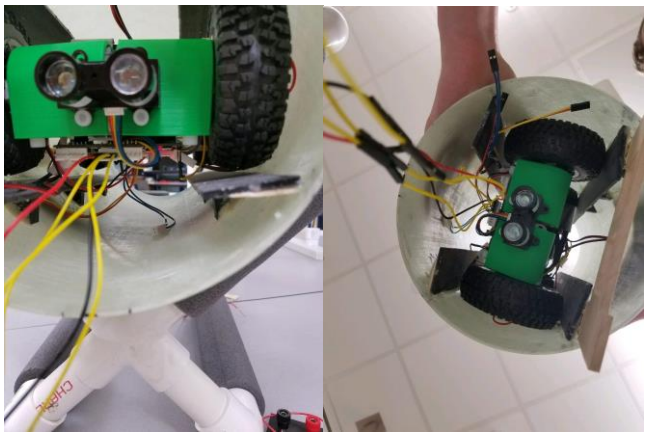


Figure 126. (Right) View of the rover secured in the mounting blocks, the clearance between the tracks is visible. (Left) View from below the body tube showing the rover remained secured during the shake test.

Once this initial test was complete, the nose cone was placed on the body tube and secured with eight shear pins. The entire payload was inverted and again shook. The nose cone had no movement. Each of the eight shear pins was examined for fractures. All pins remained intact.

These tests were deemed a success as the rover did not move. This will be further tested with further full scale launches.

7.1.3.3 Object Avoidance Test

After it was established that each of the subsystems worked as expected, the entire system was tested. First, each of the motor was tested, and after code was created that allowed for movement, a test was conducted to see how well the rover was able to avoid obstacles. The results of the test were promising; however there were problems polling the LiDAR system every quarter of a second, as for thin, tall objects, the rover would turn out of the way, then hit the object. This was resolved by, whenever the rover went into a turning subroutine, it would wait to poll the LiDAR for a half second, allowing the LiDAR to turn enough to get out of the way of the object that was in front of it.

Rocket detection test -- Bluetooth

After construction of the bluetooth boards, several tests were conducted. First, using an iPhone app called NRF Connect, made by Nordic Semiconductor, a quick test to check the feasibility of using the bluetooth to determine distance from the rocket. This test demonstrated that the pic would be able to sense the distance from the rocket based off of the bluetooth signal strength, but at short distances, it was difficult to see the enough of a difference in power levels to adequately calculate distance. When the bluetooth power level was changed, the drop off was much more predictable at short distances, allowing for adequate distance measuring.

7.1.3.4 Deployment System and Black Powder Testing

The rover deployment system was tested prior to the full scale launch and with black powder at the full scale launch. The initial deployment test was conducted in the lab and in a controlled setting. The rover was loaded into the body tube and secured within the mounting blocks. The nose cone remained off of the body tube for these initial tests. To simulate the deployment, two E-matches were placed in the PVC pipes and connected to the deployment electronics. Once the set up was complete, the shunt pins were removed from the E-matches and the testing personnel moved away from the charges. The base station initiated the command to light the matches and begin the deployment sequence. The initial testing resulted in inconsistent ignition of the E-matches. After analysis of the deployment circuits, a flaw in the wiring was discovered and corrected. Further controlled testing was conducted and resulted in both E-matches igniting each time. Once the E-matches ignited, simulating the nose cone removal, the rover exited the body tube. The rover was able to drive out of the body tube at various orientations and angles. For example: the body tube was purposely placed with the rover at a 45 degree angle to the ground and was successful in driving out on the tracks and onto the ground.

The system was further tested during the full scale launch with ground testing of the black powder. The main goal of this test was to optimize the amount of black powder needed to remove the nose cone that is secured with eight shear pins. Prior to the test, the team planned to use three to five grams in each PVC pipe to remove the nose cone. However, the test was successful with only one gram of black powder. This discrepancy in our prediction lies in the small area that is pressurized by the ejection charges. The area in between the nose cone bulkheads is relatively small, six inches, and in turn does not require the amounts initially designed for.

Figure 127. Screen grabs from the video of the black powder and deployment test of the rover payload.

7.1.3.5 Full scale Launch Testing

Due to a recovery system error in the initial control flight, the rover payload was unable to be tested prior to the FRR. During the control flight the rover was simulated with ballast located at the CG of the Deployable Rover Payload. The rover was planned for the second launch, but damage to the nose cone prevented a second launch. The rover will be tested in flight March 17th.

7.1.3.6 Lessons Learned

During the construction and testing of the payload several lessons learned were identified. During the assembly of the rover several components had fit issues that could be traced back to tolerance issues. In the future, tolerancing will be more deliberate in the design process. In terms of the electronics, different color wires should have been used for the different power levels. While no mistake was recorded with the wires being the same color, having easily distinguished wires would lead to easier construction and securing of the rover.

7.1.4 Air Braking System

7.1.4.1 System Ground Testing

7.1.4.1.1 Mechanical Subsystem Testing Procedures and Results

The confirmation of the mechanism's operation was performed via 2 tests.

First, after all tabs were cut and sanded to size for smooth movement in the channel, the mechanism was set to different positions, and the extension of each tab was compared. This test allowed the team to make sure that the mechanism operated symmetrically.

Next, full range of motion tests were performed in order to check for hitches in movement or dead spots during rotation. The lack of dead spots and sudden movement allowed the team to conclude that the mechanism operated smoothly.

7.1.4.2 Electronic Subsystem Testing Procedures and Results

7.1.4.2.1 Confirmation of Servo Motor Operation

To confirm servo operation, a test control code was uploaded to a microcontroller with the microcontroller pwm output connected to the servo control input. The servo was then powered by a 7.4 V supply. Operation was verified by comparing servo arm angles to the angles specified in the control code. The test was repeated for each servo, with all servos performing as expected. These results have confirmed the viability of using two servos operating in tandem to actuate the mechanism.

7.1.4.2.2 Confirmation of Printed Circuit Board Operation

Testing was performed on the PCB to ensure connectivity between all wires and traces. First the ohmmeter functionality of a multimeter was used to measure the resistance between ports that were supposed to be connected by traces on the PCB. Each connection showed a very low resistance value, indicating no broken traces or solder joints. The ability of the switch to turn on and off the connections on the board was also tested. The ohmmeter was placed on the output line of the switch as well as the high input, and later the low input. Based on switch orientation, the resistance between a given input and the output was very low when selected and very high when not selected. The positive results of this test indicated the switch was working as expected.

Additional tests were then run with the batteries connected through the molex port. The voltmeter functionality of the multimeter was used to measure the voltage through the circuit and ensure that the PCB voltage matched the voltage across the battery. The test of the switch was also replicated to demonstrate that with the switch turned off it would no longer allow the voltage into the PCB and the entire voltage drop would occur at the switch.

7.1.4.2.3 Power System Loading Test

The power system loading test had two goals:

1. To verify that the batteries were capable of operating the servos
2. To ensure current draw from the motors was not significantly greater than the recommended maximum battery supply current of 5.9 A. (as specified by the battery manufacturer.)

The power system was only tested outside of the mechanism due to inaccessibility once installed. To measure current, the circuit was configured as follows: the negative lead of the battery was inserted into a breadboard column and a jumper wire was used to connect the breadboard column to the negative terminal of servo. A second jumper wire was used to connect the positive servo terminal to a jumper-to-banana adapter. A second jumper-banana adapter was connected to the negative column on the breadboard and a banana-to-banana cable was connected between the two adapters. An oscilloscope current probe was connected to the banana-to-banana cable to measure current. Once configured, the servo was operated with various load torques. For the majority of the tests, the current was well below the 5.9 Amp recommended maximum. At high torque, the current reached 6.75 Amps, as shown in Figure 128. Because this is above the recommended maximum battery supply current, the servo was allowed to operate at this torque for over 120 seconds. During operation, the battery was observed for signs of degradation (such as warmth or smoke). Because no short term degradation was observed and because the expected high-torque operating time during flight is well below 120 seconds, the high current draw was determined to be acceptable given the limitations on cost and size. Furthermore, because this max load current was only 14% above the recommended max load, we consider this difference to be a concern for the longevity of the battery, but not an in-flight concern or danger to the system.

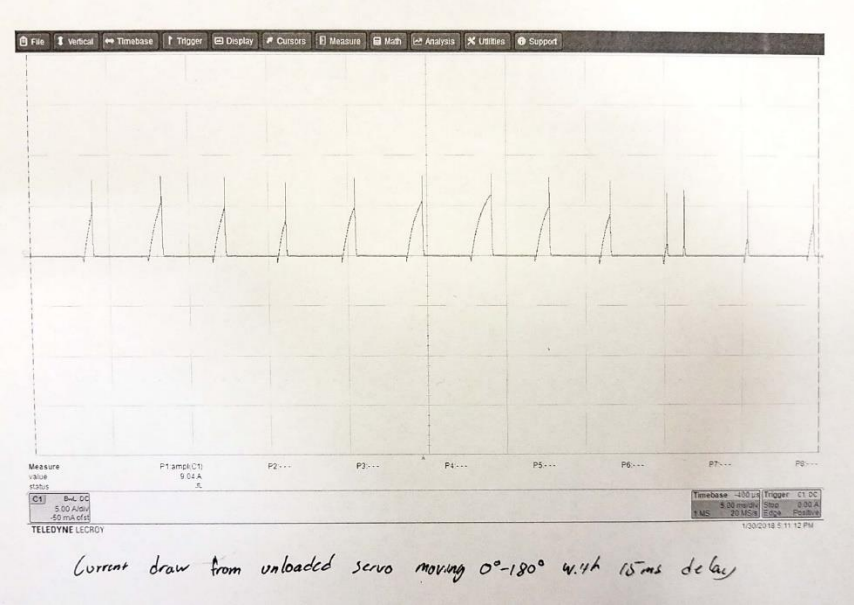


Figure 128. Graph obtained from testing of the battery showing that the peak voltage during operation is 6.75 V.

The first goal of testing, to verify that the batteries were capable of operating the servos, was successful. The second goal, to verify that the current draw is not significantly higher than recommended maximum, was not met. However, the batteries were determined to be safe for flight as per the previous justifications.

7.1.4.2.4 Sensor Noise Test

The chosen accelerometer and barometer models have both undergone performance testing to ensure they can fulfill the requirements of this system. By wiring both sensors to an Arduino MKR Zero and running sample code, a large number of data points were obtained for both sensors. Since the surrounding environment was held constant and did not undergo any change in acceleration, pressure, or temperature during the testing period, an analysis of this data allowed the sensors' precision and sampling rate to be experimentally measured. Through the creation of histograms such as Figure 129 below, a standard deviation was obtained for each sensor.

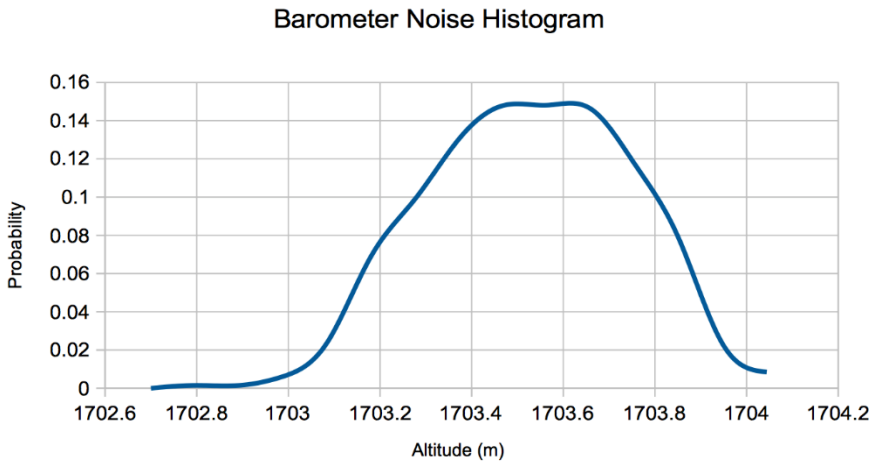


Figure 129. Histogram of barometric altitude data generated from 5000 samples.

The BMP280 has a standard deviation of 0.22 m (0.72 ft), and the ADXL345 has a standard deviation of 0.12 m/s (0.4 ft/s). Additionally, the barometer has an average update rate of 36 Hz, while the accelerometer has an average update rate of 122 Hz (running in 100 Hz mode instead of its maximum 3200 Hz). These specifications are in line with the manufacturer-provided information used in control code simulations, so the algorithm's simulated performance should closely correspond to its real-world performance.

7.1.4.3 Control Code Testing Procedures and Results

7.1.4.3.1 Simulated Successful Flight

In order to simulate the entire control algorithm on the ground, a modified version of the final flight code was produced which instead of reading in sensor data took an initial position, velocity, and acceleration and simulated the rocket's vertical motion using kinematics, with drag taken into account. Print statements were added throughout the code to report the rocket's current position, velocity, and acceleration, along with the intended output angle for the servos as calculated by the PID controller.

At first, this code was simply run and debugged on the mechanically and electronically complete payload until an entire flight could be executed without the processor hanging or displaying otherwise aberrant behavior. This process allowed errors and failure points in the code to be detected and debugged, improving the functionality of multiple subroutines, primarily *CalcError()* and *SaveSensorData()*. Once the control code was debugged in its entirety, an entire flight could be simulated on the payload with variable initial conditions.

While the rocket's velocity was simulated, the drag tabs did actually extend and retract appropriately, providing visual confirmation of how the payload operates mid-flight. Additionally, areas where the tabs' motion could be improved were identified, leading to the implementation of an anti-jittering limiter in the *PID()* function.

All previous simulations of the control code manually approximated the effects of propagation delay. This test, on the other hand, relied on data from the potentiometer to determine the tabs' actual extension at any given time, which was then used to calculate the resultant drag force. Since these tests consistently simulated an apogee of 5279 ft., this indicates that the payload's performance in a full-scale test flight should be similar.

7.1.4.3.2 Simulated Jammed Flight

In order to test the mechanism jam detector, the *Check_Jam()* function was isolated in a separate piece of code that repeatedly cycled the tabs between full extension and full retraction. The current angle was outputted to the Arduino's Serial monitor, along with whether *Check_Jam()* returned true. By running this code, first without interfering with the tabs and then while applying force counter to the tabs' extension, the accuracy and effectiveness of the *Check_Jam()* function was tested. This allowed for tuning of the linear transformation constants and acceptable error value to a range which would not detect a jam for unimpeded motion but did for impeded motion.

7.1.4.4 System Testing Procedures and Results

7.1.4.4.1 Shake Test

This test consisted of vigorously shaking the payload for an extended period of time to confirm that the system was structurally sound and all components were secure during a vibrational state similar to the one the system will experience in flight. No component of the payload was loosened, fell out, or structurally failed, meaning the test was a success. These results showed that the components of the payload are securely fastened and can resist jostling during launch.

7.2 Requirements Verification

7.2.1 Vehicles Design

NASA Requirements

Requirement	Requirement will be met by	Results of Verification
The launch vehicle will hit an apogee of 5280 feet.	-Making a motor choice that provides the launch vehicle with enough thrust to overcome its mass.	-The Subscale Test on Dec. 2nd provided initial comparisons for predictive programs and actual results

	-Constructing a launch vehicle that minimizes drag by utilizing a smooth surface	-Full Scale Tests on March 3rd, 2018 verified software apogee predictions -Secondary Full scale launch will verify effect of the Air Braking System on the vehicle's apogee
The launch vehicle shall carry one commercially available, barometric altimeter.	-Attaching the altimeter to the launch vehicle within the recovery system.	-The recovery sub-team lead has ensured that the recovery system of the launch vehicle includes a barometric altimeter.
The launch vehicle shall be recoverable and reusable.	-Ensuring with extensive testing that each sub-system (the recovery system especially) performs both individually and integrated with every other system. -Designing the vehicle to launch safely more than once.	-The tests outlined the Vehicle Test plan verified the performance of each sub-system individually. -Secon Full Scale Test scheduled to verify the performance of the vehicle as a whole; evaluation of the vehicle after the launch verified its reusability.
All recovery electronics shall be powered by commercially available batteries.	-Ensuring that the recovery system is designed to include only commercially available batteries.	-Vehicle has been designed to include the appropriate batteries.
The launch shall be limited to a single stage and four (4) independent sections	-Ensuring that the launch vehicle only requires one launch stage and that it includes the necessary amount of sections.	-The motor has been chosen so that the vehicle reaches apogee with only one stage. -The launch vehicle has been designed to have three (3) independent sections. -The launch vehicle used during the Full Scale Tests was confirmed as having

		three (3) independent sections.
The launch vehicle shall be capable of being prepared for launch in 4 hours.	-Integrating all of the vehicle's subsystems in such a way that allows for assembly within the required time.	-Procedural checklists have been created to streamline the vehicle assembly process with minimal error -Full Scale launch on March 3rd, 2018 confirmed that the launch vehicle was assembled within 4 hours
The launch vehicle shall be launched using a 12-volt direct firing system.	-Ensuring that the vehicle design includes the required 12-volt direct firing system.	-During the Full Scale Test March 3rd, 2018, the team used the required firing system to launch the vehicle.
The launch vehicle shall require no special ground equipment to initiate launch.	-Designing the launch vehicle to successfully launch without the aid of any special equipment.	-The vehicle has been designed to launch using only the required igniter and supplied firing system -Full Scale Test on March 3rd, 2018 confirmed that the launch vehicle requires no special ground equipment to initiate launch
The launch vehicle shall use a commercially available motor.	-Choosing a motor in alignment with the NAR and TRA regulations. -Staying in contact with the team's mentor regarding any updates to motor choices as the vehicle design changes.	-A commercially available motor has been chosen that aligns with NAR and TAR regulations. -The teams mentor has been contacted about motor and design choices. -During the Full Scale Test on March 3rd, 2018 the team mentor handled all motors

The minimum velocity off the rail shall not be below 52 ft/s.	-Selecting a motor with the right impulse to achieve the required velocity. -Meticulously calculating interruption angles for rail buttons to ensure that the launch vehicle's interaction with the launch pad is smooth and uninterrupted.	-Simulations using OpenRocket and RocketSim have verified that the rocket will have an off-rail velocity greater than 52ft/s. -Full scale test on March 3rd, 2018 confirmed the off rail velocity is 78.89 ft/s, which is above the required 52 ft/s
The team shall launch and recover a subscale.	-The team will construct and launch a subscale, the specifications and materials of which are outlined in Section 3.3.1.4.2.1 and Section 3.3.1.4.2.2, respectively.	-Subscale successfully launched and recovered on Dec. 2nd.
The launch vehicle shall have a static stability of at least 2.0 at rail exit.	-Placing the Air Braking System slightly above the center of pressure so that the deployment of tabs in flight does not cause overstability when deployed -Ensuring that necessary ballasts are spread across the launch vehicle so as not to affect the stability margin	-The pre and post burnout positions of the Center of Gravity and Center of Pressure have been predicted using OpenRocket and Rocksim. -CG measurement during Full Scale Test on March 3rd, 2018 confirmed the stability above the required margin of 2.0
The launch vehicle shall have a sufficient thrust-to-weight ratio to achieve required apogee.	-Choosing a motor that provides the proper amount of thrust to overcome the weight of the launch vehicle.	-OpenRocket and RockSim simulations have verified the thrust-to-weight ratio provided by the motor choice. -Full Scale Test on March 3rd, 2018 confirmed the accuracy of the software predictions of the thrust-to-weight ratio
The launch vehicle shall contain a remotely activated	-Designing, constructing, and deploying the Rover Payload	-Physical testing confirmed that that rover can

rover which will autonomously travel five feet and deploy a set of foldable solar panels.	system, the specifications of which are included in Section 4.1.4.	autonomously deploy from the body tube, move five feet, and deploy solar panels
Payload affecting flight shall be verified before launch at competitions	- Confirming the effect of Air Braking System's tabs on flight path compared to lack thereof - Confirming the structural strength of Payload Integration - Confirming the efficacy of Payload in a practical sense	- Subscale Flight successfully completed on Dec. 2nd, significant decrease in apogee linked to Air Braking Tabs. - Load analysis using principles from solid mechanics verified that the structure is strong with factors of safety - Full Scale Flights in March 2018 will verify the impact of the Air Braking System
Launch Vehicle's subsystems shall have finished the design phase; Subscale will have been launched by CDR	- Ensuring each subsystem fits in with the overall system and can be edited at short notice. - Designing and launching a subscale that will verify the accuracy of our software prediction for confidence.	-Subscale successfully launched on Dec. 2nd, verifying prediction programs. -Launch Vehicle Subsystems finished design phase, finished construction in February 2018

= Complete Verification

= Incomplete Verification

Team Requirements

Requirement	Requirement met by	Result of Verification
The launch vehicle will not be over stable; it will not have a stability margin exceeding 3.10.	-Design a launch vehicle with the CG forward of the CP by a measure of at least two calibers of the largest body tube diameter. -Model the rocket in OpenRocket and RockSim to simulate stability. -Verify the position of the CG and therefore stability with a physical CG test.	-Vehicle designed with a stability margin of 2.95. -Vehicle modeled in both OpenRocket and Rocksim, with both programs reporting similar caliber values. -CG test performed before test launch on March 3rd, 2018. CG accurate within an inch, stability margin measured at 3.06.
The launch vehicle will reach an un-altered altitude of 5450 ± 100 ft so that the Air Braking System can be utilized to reach the target apogee of 5280 ft.	-Design a launch vehicle so that the predicted apogee is 5450 ± 100 ft. -Verify that the launch vehicle reaches this apogee with a full scale test. -Verify that the Air Braking System decreases the apogee to 5280 ft.	-Vehicle designed so that the predicted apogee is 5405 ft. -Vehicle test flight on March 3rd, 2018 verifies that the simulations are accurate within 100 ft. -Air Braking System to be tested in second test flight late March 2018
Transition section does not alter flow along aft section of the rocket.	-ANSYS Fluent computer analysis of the boundary layer aft of this geometric change -Subscale and full scale test flights to verity that flow does not impact apogee or stability by separating the boundary layer or creating too much turbulence	-Preliminary ANSYS Fluent analysis completed in October 2017 indicated that transition section does not impact flow in a drastic manner. -Subscale flight on December 2nd, 2018 verified that the transition does not have a major impact on flight profile as predicted by computer simulations. Full scale flight on March 3rd, 2018 also confirms this.
The drag tabs of the Air Braking System will not interfere with the flow reaching the main fins.	-Design the rocket such that the drag tabs and fins are radially offset. -ANSYS Fluent computer analysis to simulate the flow around both the drag tabs and fins.	-Rocket designed so that the drag tabs and fins are offset by 45 degrees radially. -Preliminary ANSYS Fluent analysis indicated that the flow around the fins was not disturbed in a drastic manner.

	-Subscale and full scale flights to confirm the stability of the vehicle.	- Subscale flight on December 2nd, 2017 and full scale on March 3rd, 2018 confirmed that the rocket remained stable during flight. -Second test flight in late March 2018 to verify that extended tabs do not impact the stability of the rocket.
Payloads will be easily accessible and removable in the event of a needed change or a safety emergency involving the electronics.	-Design each system so that payloads are accessible with minimal effort while at the same time being firmly secured throughout flight profile.	-Deployable Rover Payload designed for easy access through the nose cone and easy removability from its body tube. Removability verified during pre-test launch checks. -Air Braking System designed on a system of tie rods that are easily accessible and removable from the fin can. Removability verified during pre-test launch checks and after recovery. -Recovery system designed on an easily removable screw-to-lock mechanism. Removability verified during pre-test launch checks and after recovery. -Air Braking and recovery systems verified as firmly secured with physical testing and with test launch. -Deployable Rover payload verified as secured with physical testing. -Rover payload retention to be tested in full scale launch in late March 2018

= Complete Verification

= Incomplete Verification

7.2.2 Recovery System

Requirement	Verification Plan
The launch vehicle will stage the deployment of its recovery devices, where a drogue parachute is deployed at apogee and a main parachute is deployed at a lower altitude. Tumble or streamer recovery from apogee to main parachute deployment is also permissible, provided that kinetic energy during drogue-stage descent is reasonable, as deemed by the RSO.	Demonstrate this requirement with a successful launch wherein a drogue is deployed at apogee and a main is deployed a lower altitude.
Each team must perform a successful ground ejection test for both the drogue and main parachutes. This must be done prior to the initial subscale and full-scale launches.	Test this requirement by fine-tuning the proper black powder/sheer pin ratio in a series of tests prior to first full scale launch.
At landing, each independent sections of the launch vehicle will have a maximum kinetic energy of 75 ft-lbf.	Analyze for this requirement by performing simulations and performing calculations which accurately predict vehicle parameters and select design variables which satisfy these parameters.
The recovery system electrical circuits will be completely independent of any payload electrical circuits.	Inspect for this requirement by ensuring no wires from other vehicle sections pass through or enter into the recovery section.
All recovery electronics will be powered by commercially available batteries.	Demonstrate this requirement by indicating the commercial brand name on the batteries power the recovery system and showing no other possible sources of power in the system.

The recovery system will contain redundant, commercially available altimeters. The term “altimeters” includes both simple altimeters and more sophisticated flight computers.	Demonstrate this requirement by pointing out the three axially symmetrical subsystems which independently carry out the task of recovery.
Motor ejection is not a permissible form of primary or secondary deployment.	Demonstrate this requirement by showing separate ejection charges within the recovery system for parachute deployment and indicate no possible way for motor ejection to separate the necessary section.
Removable shear pins will be used for both the main parachute compartment and the drogue parachute compartment.	Inspect for this requirement by ensuring the correct number of shear pins are present at all separation point prior to launch.
Recovery area will be limited to a 2500 foot radius from the launch pads.	Analyze and demonstrate this requirement by running simulations to predict drift radius and observing the actual distance relative to these predictions after full scale launches.
An electronic tracking device will be installed in the launch vehicle and will transmit the position of the tethered vehicle or any independent section to a ground receiver.	Inspect for this requirement in the air-braking payload section of the launch vehicle which will house this instrument in place of the recovery section.
Any rocket section, or payload component, which lands untethered to the launch vehicle, will also carry an active electronic tracking device.	Demonstrate this requirement by noting that only one connected series of rocket sections will descend together and that a tracking device is situated in the air-braking payload.

The electronic tracking device will be fully functional during the official flight on launch day.	Test and inspect for this requirement by determining the capabilities and limitations of the device in test launches, and maintaining appropriate operating conditions (weather, interference from other transmitters) on launch day.
The recovery system electronics will not be adversely affected by any other on-board electronic devices during flight (from launch until landing).	Test and inspect for this requirement by attempting to interfere with the recovery system remotely in a lab setting and ensuring the copper coating is pieced together properly before each launch.
The recovery system altimeters will be physically located in a separate compartment within the vehicle from any other radio frequency transmitting device and/or magnetic wave producing device.	Demonstrate this requirement by locating the altimeters and the radio transmitters and noting they are housed in distinct sections.
The recovery system electronics will be shielded from all onboard transmitting devices, to avoid inadvertent excitation of the recovery system electronics.	Inspect for this requirement by checking for holes or tears in the protective copper tape.
The recovery system electronics will be shielded from all onboard devices which may generate magnetic waves (such as generators, solenoid valves, and Tesla coils) to avoid inadvertent excitation of the recovery system.	Demonstrate that no such devices are utilized in the launch vehicle and refer to previous verifications regarding EM radiation.
The recovery system electronics will be shielded from any other onboard devices which may adversely affect the proper operation of the recovery system electronics.	Test and demonstrate that the transmitters/receivers aboard the rocket do not interfere with the shielded recovery system.

Additional Requirement 1. The recovery system will utilize heat-resistant materials on all surfaces exposed to black powder charges.	Demonstrate that high quality, heat-resistant epoxies, acrylics, PVCs, and electronic components are used on the exterior of the CRAM.
Additional Requirement 2. The recovery system will use new, *Duracell* brand batteries because of their exceptionally robust fused composition.	Demonstrate that only the proper brand of batteries is purchased and installed - new - before each flight.
Additional Requirement 3. The recovery system will use duct seal to protect any wire access holes in the bulkhead from black powder damage during flight.	Demonstrate that the sealant has been applied and minimal access to the core is possible for the black powder explosives

7.2.3 Deployable Rover Payload

NASA Requirements

Requirement	Requirement will be met by	Verification
Teams will design a custom rover that will deploy from the internal structure of the launch vehicle	The rover is built from a combination of custom milled HDPE, 3D printed components and commercially available parts. The rover is secured directly below the nose cone and is deployed using a radio controlled ground station.	- A fully constructed rover payload was present for the full scale launch - Securing system tested with shake test and full scale launch
At landing, the team will remotely activate a trigger to deploy the rover from the rocket	A LoRa module is used for wireless communication to the rover. This module will interface with a laptop base station to initiate the deployment sequence	- The LoRa module was tested without ejection charges - Ground testing was performed to optimize the amount of black powder used - Full scale launch to test the entire system will be performed in late March

	upon a safe landing	
After deployment, the rover will autonomously move at least 5ft (in any direction) from the launch vehicle.	Bluetooth chips are placed throughout the body of the rocket to provide triangulation to the rover.	- Isolated bluetooth tests were performed prior to construction - Ground tests were performed to ensure the placement of the chips allowed for accurate triangulation - Full scale flight test performed in late March
Once the rover has reached its final destination, it will deploy a set of foldable solar cell panels	When the rover registers the safe distance from the rocket a servomotor drives the solar array out. The array will triple in surface area due to the folds.	- Prior to the full scale test flight the deployment sequence was tested and confirmed using a dummy array. - The full scale flight tested the entire rover sequence including the solar panels.

Team Requirements

Requirement	Requirement will be met by	Verification
The nose cone will be deployed via black powder charges allowing the rover to drive out of the rocket	Two PVC pipes mounted at the rear of the payload will contain two grams of black powder (one gram in each pipe). The base station initiates the deployment sequence that ignites e-matches and sets of the ejection charges. Shunt pins are in place to prevent misfire before the rocket is loaded onto the pad	- Ground tests were performed to optimize the number of shear pins securing the nose cone and the amount of black powder used. - Full Scale Launch verified the deployment - to be done in late March - A redundant system is in place to ensure the ignition of the ejection charges.
Upon landing, the rover will be capable of driving in an inverted position	Oversized wheels provide clearance in both orientations. The electronics are configured	- The rover was tested in both orientations and successfully avoided objects and detected the rocket - Rover successfully drove and

	for driving in either orientation.	maneuvered on rough terrain similar to the expected competition terrain
The solar panels will provide a measurable source of power.	Electrical functionality is verified by a resistor placed on the board powered only by the solar cells.	Post launch the data will be inspected for power.

7.2.4 Air Braking System

Verification	Description	Method of Verification	Status
Drag Tab Efficacy	Confirm that the drag tabs are appropriately sized	Subscale flight test	Complete
Mechanism Operation	Confirm that the mechanism moves as expected and does not jam	Ground testing	Complete
Servo Motor Operation	Confirm that the servomotors function and are able to be controlled as necessary	Ground testing using the servo tester	Complete
PCB Operation	Confirm that the connections on the PCB are functioning	Ground testing using a multimeter	Complete
Battery Operation	Confirm that both the servo batteries and microcontroller battery are functioning, and that both are able to charge properly	Ground testing using a multimeter	Complete
Microcontroller Operation	Confirm that the Arduino MKR Zero is functioning, that all pins are reading data, and that the SD card can read and write data	Ground testing using sample code	Complete
Sensor Operation	Confirm sensor function, determine noise levels in the sensors	Ground testing	Complete

Control Code Operation	Confirm code function, debug errors in the PID controller, debug errors in the jammed mechanism redundancy, debug errors in the state switches	Ground testing using simulation code	Complete

7.3 Educational Engagement

The Notre Dame Rocketry Team has placed a large emphasis on educational engagement this year. As stated in the Student Launch Handbook, the team is required to engage a minimum of 200 participants in "educational, hands-on science, technology, engineering, and mathematics activities." However, with this in mind, the team set a goal in the fall of engaging 500 participants. A specific attempt was made to both continue community partnerships from past years and to create new relationships to reach even more students. The team also wanted to focus on more direct and personal interactions with students. While presentations can reach a larger number of participants, the team has determined that personal, mentorship can have a larger impact on students.

These initial goals and focus have helped guide NDRT to make this the team's most active year yet for educational outreach. The team worked with over 10 different schools and organizations and was able to reach over 1200 local students, which far exceeded the team's initial goal! These numbers are broken down below in Table 46. Overall, participants ranged in age from 3-18. The team led children from two local after-school programs through a 5-week Rocketry 101 series, which introduced students to Newtonian forces, rocket stability, propulsion, and recovery systems before they were able to build and launch their own personalized rockets. Children in local after-school programs, as well as several one-time interactive events with groups ranging in size from 7 participants to 90 participants, were part of the 5-week Rocketry 101 Series. As the team focused heavily on direct interactions with students, a particular effort was made to engage students during more indirect interactions, through quick activities such as rocket drawing at the Science Alive! Fair or simply allowing students to interact up close with past rockets during events like the Engineering Expo at Madison Primary. The pictures shown below in Figure 130 are just a few visual examples of these events. In addition to the reported educational activities, NDRT members also participated in several events for prospective students and first year engineering students at Notre Dame to help mentor them in their decision about whether to continue in STEM.

Figure 130. NDRT Members working with students at the Science Alive! Fair and at events with Harrison Primary Center and Notre Dame's College Mentors for Kids Program.

Overall, the team is excited about the impact it was able to make this year in the South Bend and Notre Dame communities and is motivated to continue working with these and more organizations for the remainder of the year and in future years.

Table 46. Breakdown of NDRT Educational Engagement Activities.

Event and Partnership Organization	Students Reached
College Mentors For Kids – *Physics is Phun*	70
St. Joseph County Library – *Science Alive! Fair*	500 (very conservative estimate as over 3600 children came to the event)
St. Joseph County Boys and Girls Club – *Rocketry 101 (5-week program)*	Ranging from 8-12
Robinson Community Learning Center – *Rocketry 101 (5-week program)*	Ranging from 30-45
Navarre Intermediate Center – *STEM Carnival*	50
Harrison Primary Center – *Physics is Phun*	90
SWE/Madison Primary Engineering Expo – *Rocketry 101 (1-time interactive presentation)*	400
Girl Scouts of St. Joseph County – *What is Aerospace Engineering? & Physics is Phun*	63
Lotus Preschool – *Propulsion for Everyone*	7
NASA Requirement	200
NDRT Goal	500

Total Students Reached	**1237**

7.4 Budgeting

The overall team budget can be seen below in Table 47. While all the budgets prior to this report have been much higher, the final budget has been much lower due to the materials issue the team experienced after CDR. While originally the vehicle body was going to be constructed using fiberglass and carbon fiber, it ended up being constructed of fiberglass and phenolic. Phenolic has much lower costs than custom carbon fiber and drove the total cost of the vehicle body down significantly.

Table 47. Budget allocation breakdown.

Allocation Group	Budget Spent
Vehicle Design Sub-team	$ 2598.16
Recovery Systems Sub-team	$ 975
Deployable Rover Sub-team	$ 806.73
Air Braking System Sub-team	$ 299.81
Rocket Subtotal	**$ 4, 679.70**
Educational Outreach Events	$ 300
Miscellaneous	$ 300
Competition Travel	$ 7990.91
GRAND TOTAL	**$ 13,270.61**

The costs shown in Table 47 can be accounted toward the following items:

Vehicle Construction and Propulsion: These costs account for all materials that were used to build the launch vehicle as well as for the motors used in all launches.

Recovery System: The recovery costs include all parachutes, altimeters, 3D printed materials and all items necessary for a safe and robust integration into the vehicle.

Deployable Rover Payload: The costs associated with the experimental payload include all materials, wheels, solar cells, rover motors, all electronics and items that were needed to ensure a safe and successful integration.

Air Braking System: The costs for the extra payload account for all materials, electronics, servo motors and 3D printed items needed.

Educational Outreach: These funds are set for use during educational and community engagement events, and were used to purchase Estes rockets with kids, as well as a variety of other arts and crafts materials used for the events.

Miscellaneous: In this category are costs for posters and other items associated with a professional team image and presentation.

Travel: All costs associated with traveling are included in this number including transportation, food and lodging.

The Notre Dame Rocketry Team has recently become a self-sustained team in terms of acquiring funds. While in previous years the team has been very dependent on the University's College of Engineering for funding, recently the team has partnered with two more companies for sponsorship, in addition to the continual sponsorship from Boeing. The Boeing Company continues to be the primary sponsor to the team, but Textron and Timkensteel have also recently graciously donated to the team's cause. With all three companies' recent support, the Notre Dame Rocketry Team has not needed to ask the University for funding this year. The fact that the team now has three company sponsors shows the continual growth from year to year.

For future years, the team has funding secured from the University should it be necessary. This ensures the continuation of the program for future years.

Individual line-item budget breakdowns can be seen in Appendices L through P. There are breakdowns for the vehicles design sub-team, the recovery system, the deployable rover payload, the air braking system and a final one for travel to the competition.

7.5 Timeline

A timeline for all the systems and overall team can be found in Appendix Q.

Appendix A: Performance Prediction Code Using Python

```
import math
def apogee(m_r, m_e, m_p, p, Cd_t, Cd_c, A, T, g, t):
"""
m_r   : rocket mass [M]
m_e   : engine mass [M]
m_p   : propellant mass [M]
p     : air density [M/L^3]
Cd_t  : drag coefficient during thrust phase
Cd_c1 : drag coefficient during first coasting phase
Cd_c2 : drag coefficient during second coasting phase
A     : rocket cross sectional area [L^2]
T     : thrust [F]
g     : acceleration due to gravity [L/T^2]
t     : burnout motor time [T]
"""
Cd_b = Cd_c + 1.28*math.sin(4*math.pi/180)

k_t = .5 * p * Cd_t * A #aerodynamic drag coefficient [M/L] (thrust phase)
k_c1 = .5 * p * Cd_c1 * A #aerodynamic drag coefficient [M/L] (coast phase)
k_b = .5 * p * Cd_b * A #aerodynamic drag coefficient [M/L] (air brake phase)
k_c2 = .5 * p * Cd_c2 * A #aerodynamic drag coefficient [M/L] (coast phase)
# thrust
m_a = m_r + m_e - m_p/2 #average mass [M]
q1 = math.sqrt((T - m_a*g) / k_t) #burnout velocity coefficient [L/T]
x1 = (2 * k_t * q1) / m_a #burnout velocity decay coefficient [1/T]
v1 = q1 * ((1-math.exp(-x1*t)/(1+math.exp(-x1*t)))) #burnout velocity [L/T]
max_v = v1
y1 = (-m_a/(2*k_t)) * math.log((T-m_a*g-k_t*v1**2)/(T-m_a*g)) #altitude
burnout [L]

      # coast 1
qc1 = math.sqrt((T - m_c*g) / k_c1) #burnout velocity coefficient [L/T]
xc1 = (2 * k_c1 * q1) / m_c #burnout velocity decay coefficient [1/T]
vc1 = q1 * ((1-math.exp(-x1*t)/(1+math.exp(-x1*t)))) #burnout velocity [L/T]
yc1 = (-m_c/(2*k_c1)) * math.log((T-m_c*g-k_c1*v1**2)/(T-m_c*g)) #altitude
burnout [L]
# air brake
m_c = m_r + m_e - m_p #coasting mass [M]
qb = math.sqrt((T - m_c*g) / k_b) #burnout velocity coefficient [L/T]
xb = (2 * k_b * q1) / m_c # burnout velocity decay coefficient [1/T]
vb = q1 * ((1-math.exp(-x1*t)/(1+math.exp(-x1*t)))) #burnout velocity [L/T]
yb = (-m_c/(2*k_b)) * math.log((T-m_c*g-k_b*v1**2)/(T-m_c*g)) #altitude
burnout [L]
# coast 2
qc2 = math.sqrt((T - m_c*g) / k_c2) #burnout velocity coefficient [L/T]
xc2 = (2 * k_c2 * q1) / m_c #burnout velocity decay coefficient [1/T]
vc2 = q1 * ((1-math.exp(-x1*t)/(1+math.exp(-x1*t)))) #burnout velocity [L/T]
yc2 = (-m_c/(2*k_c2)) * math.log((T-m_c*g-k_c2*v1**2)/(T-m_c*g)) #altitude
burnout [L]
peak_altitude = y1 + (yc1*.625)/2 + yb*.375 + (yc2*.625)/2
return max_v, peak_altitude*1.75*3.28084 #constant multiplier & meters to
feet
def CP(Ln, d, df, dr, Lt, Xp, Cr, Ct, S, Lf, R, Xr, Xb, N):
```

```
"""
   Ln : length of nose [13 in]
   d  : diameter at base of nose [5.5 in]
   df : diameter at front of transition [5.5 in]
   dr : diameter at rear of transition [5.5 in]
   Lt : length of transition [0 in]
   Xp : distance from tip of nose to front of transition [0 in]
   Cr : fin root chord [7 in]
   Ct : fin tip chord [7 in]
   S  : fin semispan [7.2 in]
   Lf : length of fin mid-chord line [7 in]
   R  : radius of body at aft end [5.5 in]
   Xr : distance between fin root leading edge and fin tip
        leading edge parallel to body [13 in]
   Xb : length of rocket minus length of fins [88 in]
   N  : number of fins [4]
"""
# nose cone terms
Cn_n = 2
# Xn = .666*Ln #for cone
Xn = .466*Ln #for ogive
# conical transition terms
Cn_t = 2 * ((dr/2)**2 - (df/2)**2)
Xt = Xr
# Xp + (Lt/3) * (1 + (1-(df/dr)) )#/ (1-(df/dr)**2))
# fin terms
Cn_f = (1 + (R/(S+R)) * ((4*N*(S/d)**2) / \
                         (1 + math.sqrt(1 + ((2*Lf)/(Cr+Ct))**2))))
Xf = Xb + ((Xr*(Cr+2*Ct))/(3*(Cr+Ct))) + \
        (1/6)*((Cr+Ct)-((Cr*Ct)/(Cr+Ct)))
# center of pressure calculation
Cn_r = Cn_n + Cn_t + Cn_f #sum of coefficients
CP = (Cn_n*Xn + Cn_t*Xt + Cn_f*Xf) / Cn_r #CoP d from nose tip
return CP
def stability(d, CG, CP):
"""
   d: diameter of rocket (in)
   CG: center of gravity (in from nose ogive)
   CP: center of pressure (in from nose ogive)
"""
stability = (CP - CG) / d
return stability
```

Appendix B: Safety Agreement

(The following is the safety agreement that all team members have signed)

By signing below, I agree to abide by all regulations, standards and guidelines set forth by the National Association of Rocketry. I have read and understand the High-Powered Rocketry Safety Code and will follow all rules outlined within the code. I am cognizant of all local, state, and federal laws regarding the regulation of airspace and handling of explosive or controlled materials.

I understand that the Huntsville Area Rocketry Association will oversee the contest launch, and I will abide by all club rules at the launch. I acknowledge that the Notre Dame rocket will be subject to range safety inspections before flight, and I will comply with the determination of the safety inspection. The Range Safety Officer has the final say on all rocket safety issues, and failure to comply with safety requirements will prohibit the team from launching its rocket.

I agree to abide by all procedures outlined by the Safety Officer of the Notre Dame Rocket Team, Team Leader, and Team Advisor when working on the NASA Student Launch project. I will use laboratory equipment and tools only when properly trained or under appropriate supervision. I will follow all Material Safety Data Sheets for materials used in design, construction, launch, and conclusion of the project.

I understand that failure to comply with anything in this safety agreement can result in my removal from the Notre Dame Rocketry Team.

_____

(Team Member Name Printed)

_____

(Team Member Signature) (Date)

Appendix C: Vehicles FMEA Table

Type	Failure Mode /Hazard	Cause	Effect	Probability	Severity	Risk	Verification and Mitigation
Structural Failures	Failure at Nose Cone and Rover Bay Interface	Early shear pin shearing	Separation of nosecone from rocket body; exposure of rover to atmosphere	Remote	Catastrophic	Moderate Risk	Ground testing will be performed to ensure that the shear pins will hold up under regular aerodynamic stress and will only shear when the ejection charges are detonated
	Nose Cone Deformation	Loss of Structural Stability in leading component of the rocket.	Flight becomes imbalanced altering flight path, potentially leads to further structural failure.	Improbable	Critical	Low Risk	The Nose Cone has been chosen to be made of polypropylene, with a tensile and compressive strength of 4,800 psi and 7,000 psi, respectively. These are much greater than the expected load forces during flight and has been proven to hold up to these forces with the test launch.
	Body Tube Deformation	Loss of Stability in main structure of the rocket.	Flight becomes imbalanced, or fails, leading to structural failure and possible loss of the rocket.	Improbable	Catastrophic	Low Risk	The Body of the rocket has been chosen to be made of Phenolic, with a tensile and compressive strength of 310 MPa and 450 MPa, respectively. These are much greater than the expected load forces during

							flight, and has been proven to hold up to these forces with the test launch
	Coupler Deformation	Loss of internal structural stability.	Mass shifts within the rocket, creating an imbalance, and possible alteration of the flight path or stability.	Improbable	Critical	Low Risk	Couplers have been chosen to be made of Kraft Phenolic, which has been proven to stand up to expected load forces. Couplers will not be subjected to these forces, since this load will be taken on by the Body Tubes. The couplers have been proven strong enough with the successful completion of the test launch and recovery of the rocket.
	Failure of the recovery bulkheads that are attached to the Shock Cords	The bulkheads are not properly attached. The stress of the parachute deployment is too great for eyebolt or bulkhead to withstand	Sections I and III of the rocket can become detached from the other sections and can fall to ground with no recovery system and result in more structural damage.	Remote	Catastrophic	Moderate Risk	The load-bearing bulkhead will be made of 2 layers of plywood in order to ensure it can support the force it is expecting. The strength of the bond between the bulkhead and the body will be maximized. It has been adhered with epoxy. This has proven successful

							with the completion of the test launch.
	Shear Pin Failure at Parachute Bay	After drogue is deployed there is an excessive force placed on shear pins.	The main parachute deploys at a high altitude. Drift radius is larger than what was modeled.	Remote	Marginal	Low Risk	Ground testing will be performed to ensure that the shear pins can sustain the force of the drogue ejection charges. Long shock cord will be used that are 5 times the length of the rocket.
	Destabilization of CRAM	The CRAM fails under tensile stress	The avionics housed in the CRAM can be disturbed or destroyed leading to issues with the recovery system such as proper deployment of the parachutes.	Remote	Critical	Moderate Risk	The CRAM features have been produced using additive manufacturing. The material is much stronger in compression that in tension. Great care has been put into the design in order to minimize the tensile stress. The CRAM design has proven successful with the successful launch

							and recovery of the rocket.
	Zippering of Parachute Bay Body Tube	Parachutes are ejected early or when the vehicle still has significant lateral velocity.	Shock chord "rips" through the body tube, rendering the vehicle incapable of immediate reusability.	Occasional	Marginal	Moderate Risk	Increase the shock chord area at the point of contact with the body tube. This will distribute the force over a larger area. Use simulations to predict the near vertical path.
	Rail Button Failure	The rail buttons are not properly secured onto the body tube of the rocket.	This can cause an unstable flight of the rail and affect the predicted stability margin and flight path of the rocket.	Improbable	Critical	Low Risk	The Rail buttons have been secured to the body tube with screws, nuts, and washers in order to prevent any damage to the rocket. They were successful and were not damaged during the test launch.
	Failure at Fin and Fin Can Interface	The fins were improperly bonded to the fin can.	Fins can separate from the fin can, greatly effecting the stability of the rocket and its ability to withstand flight disturbances	Improbable	Catastrophic	Low Risk	The fins extend into the body tube and are adhered to both the fin can and the motor mount with four fillets. The surfaces of the points of adhesion were sanded prior to adhesion to increase surface area and form a stronger bond between interfaces.

	Failure of Motor Mount and Centering Ring Interface	The centering rings are not properly adhered to the motor mount.	The motor mount can shift during flight and cause the motor casing and motor to be unstable.	Remote	Catastrophic	Moderate Risk	The strength of the bond between the motor mount and the centering rings has been maximized. A shake test was performed to ensure all connections are stable.
	Failure of Centering Rings and Fin Can Interface	The centering rings are not properly adhered to the Fin Can.	The centering rings and motor mount can shift during flight and can cause the motor casing and motor to be unstable.	Remote	Catastrophic	Moderate Risk	The strength of the bond between the fin can and the centering rings will be maximized. They will be adhered with epoxy and the construction will be overseen by experienced personnel. A shake test will be performed to ensure all connections are stable.
	Failure of Fins	Fins are subjected to excessive loads during flight.	Fins can fracture or break into pieces, causing destabilization, possible loss of rocket, and endangering team members nearby.	Improbable	Catastrophic	Low Risk	The fins have been chosen to be made of plywood, which has proven successful in the past. These are much greater than the expected load forces during flight, and have been proven to hold up to these forces during the test launch.
	Failure of the transition section	Transition section buckles, shears, or is otherwise compromised	Front end of the rocket becomes deformed or breaks apart, resulting in a severe failure of the rocket	Improbable	Catastrophic	Low Risk	The transition section is made in one piece of fiberglass and therefore will be able to withstand

							any of the forces it is subjected to.
	Failure of the rail button standoffs	The standoffs for the rail buttons are subjected to high loads	Rail buttons become bent upon launch, causing the rocket to not launch straight	Improbable	Critical	Low Risk	The standoffs won't bear much, if any of the takeoff force, so they will be safe from failure.
Propulsion	Failed Ignition	Malfunction in the ignition of the solid fuel motor.	Large safety concern because a primed motor is on the Launch Pad and could theoretically go off at any moment. Personnel may believe that the motor is inactive when it may still be active.	Improbable	Critical	Low Risk	If no activity then a wait of a minimum of 60 seconds before approaching the rocket. Our team will work with trained professionals to insure that there is proper and reliable ignition that will occur on launch day and they will troubleshoot causes of failed ignition.
	Motor Casing explosion	Nozzle can be clogged by a detached chunk of propellant.	Motor casing can explode on launch pad under pressure and can partially or totally destroy the fin can.	Improbable	Catastrophic	Low Risk	The motor selected will have a high safety rating. It will be inspected for any visual cues of faulty manufacturing prior to installation on launch day.
	Failure of the Motor Retention	The screws, nuts or washers could not support the load they were experiencing.	Motor casing and spent motor can fall out of the motor mount during descent after main parachute deploys.	Improbable	Critical	Low Risk	The motor retention system will be tested during ground testing of the ejection charges. The screws used for the motor retention will be adhered with epoxy and JB weld to insure structural stability of the retention.

	Inadequate Motor Thrust	Motor is too weak to provide the necessary power to keep the rocket on the necessary flight path and reach the desired apogee.	Rocket flies off the desired path and underperforms.	Remote	Critical	Moderate Risk	Simulations will be run using OpenRocket and RockSim to ensure the motor selected provides the adequate amount of thrust. The motor will be properly packed into the rocket and only motors from reliable manufacturing brands will be used.
Stability	Vehicle is Unstable	Center of Gravity is Behind the Center of Pressure.	Flight path will be unpredictable and erratic.	Improbable	Catastrophic	Low Risk	Use simulations and computer models to ensure proper placement of the CG and CP. Physically verify location of CG prior to all launches.
	Vehicle is Overstable	Center of gravity is too far ahead on the vehicle; large frontal section of the rocket with rover moves center of mass too far forward	Vehicle weathercocks into the wind and off path. Will not achieve altitude and drogue ejection may zipper airframe due to lateral speed.	Occasional	Marginal	Moderate Risk	Perform multiple center of mass calculations, such as using OpenRocket and RockSim, as well as hand written math. Compare results to ensure the center of mass predictions are accurate. Physically test prior to flights to verify CG is in expected location.
	Vehicle Unstable off the Rail	Vehicle does not achieve a high enough speed to be aerodynamically stable as it leaves the rail.	Rocket is launched at an angle and goes off path. May not achieve altitude and drogue ejection may zipper airframe due to lateral speed.	Remote	Critical	Moderate Risk	By analyzing the thrust curve of the engine to be used and taking the mass of the vehicle, calculate the speed attained by the vehicle off the rail to ensure it is fast

							enough, as determined by the appropriate literature (i.e. NAR).
	Rocket does not follow Intended Flight Path	Poor construction or significantly overstable	Rocket veers from intended flight path	Occasional	Marginal	Moderate Risk	Run simulations to ensure approximations are correct. Have experienced personnel oversee construction and utilize resources on fin construction.
	The Fins are Misaligned	The fins were not properly adhered or placed at the correct angle or distance from each other.	The rocket becomes unstable and has the potential for flying off angle or spinning.	Improbable	Critical	Low Risk	The slots in the fin can for fin attachment were carefully measured and manufactured, while the fins were attached using a custom-made guide thus greatly lowering the chance that any misalignment occurred in manufacturing
	The Rail Buttons are Misaligned	The Rail buttons were not properly attached or aligned on the same axis.	The rocket will have an unstable flight of the rail or not be able to properly sit on the launch pad.	Improbable	Critical	Low Risk	The Rail buttons were secured to the body tube with screws, nuts, and washers in order to prevent any damage to the rocket. The alignment was ensured using a right angle.

Appendix D: Recovery FMEA Table

Failure mode	Cause	Effect	Probability	Severity	Risk	Controls/ Mitigations	Verification
CRAM Shears from Mount	Excessive force during main parachute deployment and/or improper construction of CRAM	Could compromise connection of parachute to the rest of the rocket.	Improbable	Critical	Low Risk	CRAM uses robust screw-to-lock mechanism with large surface area to distribute force to the CRAM mount, the majority of the force of parachute deployment will be on the eyebolts instead of the CRAM body.	Visual inspection of CRAM before instillation, to ensure that there are no obvious cracks or failures, careful packing of the parachute to ensure proper parachute deployment.
CRAM Torques out of its Mount	Torque from main parachute deployment twists the CRAM such that it is no longer screwed into its mount.	Could compromise connection of the parachute to the rest of the rocket.	Remote	Critical	Low Risk	Once screwed in place, bolts are secured from the outside of the body tube into the body of the CRAM to prevent it from twisting in its mount.	Visual inspection of CRAM before instillation, to ensure that there are no obvious cracks or failures, ensuring that the screws are set properly.
CRAM Mount shears from Body Tube	Excessive force during main parachute deployment and/or improper instillation of the CRAM mount into the body tube.	Could compromise connection of the parachute to the rest of the rocket	Improbalbe	Critical	Low Risk	CRAM mount is designed with to have a large area of contact with the body tube so that the epoxy can form a strong connection between the two.	Visual inspection of CRAM mount before instillation of the CRAM, to ensure that there are no obvious cracks, failures, or separations between the mount and the rocket body tube.
Inadequate airflow to the Altimeters	Lack of air-flow holes in body tube.	Inaccurate altimeter readings that could cause premature or late ignition of separation charges	Remote	Critical	Low Risk	Ensure that airflow holes are present that go from the outside of the rocket all the way to the core of the CRAM.	Visual confirmation that airflow holes go all the way into the CRAM core.

Incorrect calibration of Altimeters	Improper programing and ground calibration of altimeters	Inaccurate altimeter readings that could cause premature/late ignition of separation charges	Improbable	Critical	Low Risk	Ground checks to ensure expected performance of altimeters, careful recalibration of altimeters.	Altimeters tested on the ground to ensure proper calibration.
Broken wire connection	Improper instillation of the altimeters and electronic wires.	Failure to ignite separation charges	Improbable	Catastrophic	Moderate Risk	Ground tests to ensure proper connection of all altimeters prior to flight, triple redundant altimeters and ignition electronics.	Altimeter and other wire connection have been tested multiple times to ensure strong and reliable connection.
Dead batteries during launch	Improper ground check before launch	Failure to ignite separation charges	Remote	Catastrophic	Low Risk	Use fresh batteries before every launch, check all batteries for charge before instillation.	Voltage testing of all batteries before every launch.
Electromagnetic Wave interference	Emitted electromagnetic waves, either in background or from another electronic component of rocket, interferes with altimeter circuitry.	Premature/late ignition of separation charges, possible failure to ignite separation charges.	Improbable	Critical	Low Risk	Copper shielding covers the inside of the CRAM, three completely independent altimeters.	On the ground "dry run," where all rocket electronics are activated and tested for expected performance to ensure no electromagnetic interference.
Tear or large hole in parachute	Improper inspection of parachute prior to flight, lack of proper shielding from separation charges.	Parachute fails to slow the rocket down to a safe speed upon descent.	Improbable	Critical	Low Risk	Inspection of parachutes prior to launch, Nomex blanket used to protect parachute from separation charges.	Complete inspection of both parachutes prior to parachute packing to ensure no holes or potential compromises in the parachute material.

Shock cords break	Improper inspection of shock cords prior to launch, lack of proper shielding from separation charges.	Parachute separates from the rest of the rocket.	Remote	Catastrophic	Low Risk	Inspection of the shock cords prior to launch, Nomex tubing used to protect shock cords from ejection charges.	Complete inspection of the shock cords prior to parachute packing, looking specifically for fraying or signs of excessive stress.
Shock cords/ Parachute tangles on descent	Improper folding and packing of shock cords or parachute.	Parachute fails to slow rocket to safe speed on descent, sections of rocket collide during descent	Occasional	Marginal	Low Risk	Proper packing and folding of parachutes and shock cords prior to launch	Experienced team member will pack the shock cords, using the same method of folding for every launch. Shock cord folding will be double checked by other members of the team prior to launch.
Black powder charges too powerful	Improper calculation of the necessary amount of black powder needed for separation.	Shock cords could break upon separation.	Remote	Catastrophic	Low Risk	Careful recalculation of charge quantities, separation tests to confirm proper separation.	Black powder separation testing performed on the ground confirms proper separation and charge quantities.
Black powder charges not powerful enough	Improper calculation of the necessary amount of black powder needed for separation.	Rocket fails to separate, parachutes fail to deploy.	Remote	Catastrophic	Low Risk	Ground separation tests to confirm proper separation.	Black powder separation testing performed on the ground confirms proper separation and charge quantities.

Appendix E: Payloads FMEA Tables

Rover Payload FMEA Table

Failure Mode /Hazard	Cause	Effect	Probability	Severity	Risk	Verification and Mitigation
Rover damaged during flight	Vibrations, parachute shock	Inability of rover to deploy/carry out mission	Remote	Catastrophic	Moderate risk	Rover will be vibration tested to ensure that it will withstand flight forces. It will also be built of high strength materials so that it will not likely experience a failure
Rover fails to deploy on ground	Failure of nosecone to separate, failure of rover to get enough traction	Rover won't be able to leave the vehicle	Remote	Critical	Low Risk	The rover and nosecone have both been designed to circumvent these issues and both will be confirmed through ground testing
Failure to communicate with rover from ground	Rover out of transmission range, rover electronics failure	Rover won't be confirmed to deploy	Remote	Critical	Low Risk	Rover's communications and electronics will all be ground tested prior to flight

Air Braking Payload FMEA Tables

Type	Failure Mode /Hazard	Cause	Effect	Probability	Severity	Risk	Verification and Mitigation
Payload 2: Air Braking	Controller Malfunction: fins adjusted to wrong position or adjustment takes place at wrong time	Arduino/Control code error, calibration failure in servo motor and fin adjustment mechanism	Increases/decreased braking of rocket; failure to adjust rocket speed for a mile apogee	Remote	Marginal	Low Risk	Simulate and test control algorithm code, ground test fin adjustment mechanism
	Fins breaking off rocket	Excessive force applied on fins during initial deployment or sustained pressure during flight	Uneven drag distribution on rocket, loss of structural integrity, potential airflow into body of rocket, loss of airbraking control	Remote	Critical	Moderate Risk	Fin material will be significantly stronger than expected max force, structural testing to be done before flight
	Power Failure	Battery depletion during flight	Loss of controller function, no experimental data collected, underpowered servo motor	Remote	Critical	Moderate Risk	Insure batteries have sufficient charge to provide necessary power for the motor and sensors exceeding flight duration
	Control structural failure	Design lacks necessary robustness to handle takeoff forces or sustained pressure during flight	Damage to payload, loss of data, failure of entire superstructure	Remote	Critical	Moderate Risk	Controller will be structurally contained, failure will not affect data, appropriate isolation from other payloads to be included in design
	Battery becomes damaged	Failure of another part of the rocket causes the battery to become damaged	Fire risk, damage to rocket, hazard for recovery	Remote	Catastrophic	Low Risk	Battery will be secured in payload and care will be taken in fabricating nearby components such that they don't

							affect the battery should they fail.
	Structural failure or deformity of tab	Tab impact with airborne object mid flight	Unbalanced drag forces on rocket; possible inability to retract tabs	Improbable	Catastrophic	Low	Ensure that the flight path of the rocket is clear at launch
		Higher than expected stress forces from air flow	Unbalanced drag forces on rocket; possible inability to retract tabs	Remote	Catastrophic	Moderate	Run FEA simulations on models of the tabs at higher than expected stress forces
	Loss of power to payload	Dead battery	Tabs lock in place, loss of control to the payload	Occasional	Marginal	Moderate	Mark dead batteries during competition, do not turn on battery unnecessarily, keep batteries charged.
		Defective solder joint	Tabs lock in place, loss of control to the payload	Occasional	Marginal	Moderate	Use PCBs to decrease the chance of a poor solder joint, have experienced members perform final solder, have multiple people look over finished solder joint
		Exposed wires touch, trigger short circuit	Tabs lock in place, loss of control to the payload	Occasional	Marginal	Moderate	Use heat shrink and non-frayed wires as much as possible when wiring and soldering, clip stray wires strands off
	Jammed tab system	Fluid forces angle tabs, servo cannot contract tabs	Tabs lock in place, temporary loss of control to the payload	Remote	Marginal	Low	Test and verify coefficient of friction for HDPE, hang weights on tabs to create higher than expected forces

		Tabs retract too far inward, create dead position that motor is incapable of turning	Motor stalls, tabs lock in place, loss of control to the payload	Remote	Marginal	Low	Identify dead positions, code the microcontroller to not allow those positions
	Structural Failure of Vertical Rail System	Payload experiences too much acceleration coupled with weight of subsystems	Loss of rigidity of system components, possible inability of subsystems to function	Improbable	Marginal	Low	Test rod threads for maximum force limits, ensure peak motor acceleration is not a threat
	Non-optimal flight adjustments	Unexpected physical friction /resistance	Failure to meet precise apogee	Remote	Marginal	Low	Use PID controller to account for error, design code so that some variation is expected
		Accelerometer is more than 10 degrees from level, caused by improper construction or extreme launch angle	Failure to meet precise apogee	Occasional	Marginal	Moderate	Ensure that accelerometer is flat with payload during construction, ensure that payload's orientation is as close to parallel as possible in relation to the rocket body tube, do not launch at an angle
		Algorithm cannot account enough for error caused by high pressure pockets or wind gusts	Failure to meet precise apogee	Frequent	Marginal	High	High volume of simulation and testing of code with manipulation of variables, physical ground tests to ensure code functions as expected, run code in shop to observe precise results

Appendix F: Personnel Hazard Analysis

Possible Failure	Failure Mode	Effect	Probability	Severity	Risk	Controls/ Mitigations
Construction						
Drill	Drill slips and accidentally cuts into flesh or object being drilled into	Physical wound, damage to object	Improbable	Critical	Low	Ensure any team members using tool have been trained in proper tool use, ensure object being drilled into is securely fastened down
Knife	Knife slips and accidentally cuts into flesh	Small cut to severe laceration	Occasional	Marginal	Moderate	Ensure any team members using tool have been trained in proper tool use, always cut away from the body (sharp edge of knife faces away from user)
Sandpaper	Sandpaper kicks up lots of fine particles	Depending on material, particles have potential for lung irritation	Remote	Marginal	Low	Ensure any team members using tool have been trained in proper tool use, wear filter mask when sanding materials that can cause lung irritation
Electric Sander	Electric sander kicks up lots of fine particles, can physically damage flesh	Depending on material, particles have potential for lung irritation, physical wound from sander	Remote	Marginal	Low	Ensure any team members using tool have been trained in proper tool use, wear filter mask when sanding materials that can cause lung irritation, do

						not put hands near moving pieces
Hot Glue Gun	Physical contact with either metal tip or fresh glue (that possesses napalm-like qualities)	First- or second-degree burns	Remote	Marginal	Low	Ensure any team members using tool have been trained in proper tool use, allow glue to cool and solidify before handling glue, do not touch tip while plugged in, allow device to cool once unplugged
Heat Gun	Physical contact with metal tip, prolonged exposure to heated air	First- or second-degree burns	Remote	Marginal	Low	Ensure any team members using tool have been trained in proper tool use, do not rest hot metal tip on any surfaces prone to fire or melting, do not touch metal tip, point tip away from people while gun in use, allow device to cool once unplugged
Soldering iron	Physical contact with plugged in soldering iron	First- or second-degree burns	Remote	Marginal	Low	Ensure any team members using tool have been trained in proper tool use, avoid contact with soldering iron, tie back long hair
Epoxy hardener or resin	Physical contact with material	Minor skin irritation	Remote	Marginal	Low	Rinse area immediately with soap and water

Ammonium Perchlorate	Physical contact with material	Irritation to skin or mucous membranes	Remote	Marginal	Low	Minimize handling of motor, rinse area immediately with soap and water
Lead	Ingesting solder after soldering	Lead poisoning	Improbable	Critical	Low	Wash hands after soldering, avoid eating or drinking while soldering
Electrical Shock	Touching exposed wiring	Low level shock to person handling payload	Remote	Negligible	Minimal	Cover up exposed wires, heat shrink as many visible solders as possible
Fumes	Joining components with epoxy; soldering; spray-painting	Nausea, light-headedness	Occasional	Marginal	Moderate	Ensure adequate ventilation and air flow when working with solder and epoxy through use of fans or windows
Preparation and Launch						
Rocket launch ignition	Personnel being too close to rocket motor at ignition	Risk of burns	Improbable	Critical	Low	Ensure that personnel distance when launching rocket complies with NASA minimum distance table
Moving heavy objects	Transportation of ground station, transportation of rocket	Muscle strains; toe injury	Remote	Marginal	Low	Ensure any heavy object is lifted with two hands by multiple people
Exposure to detonated black powder	Packing, handling, or cleaning black powder charges	Risk of fire, burns, or irritation of respiratory system	Probable	Marginal	Moderate	Ensure proximity of fire safety equipment; ensure that eye and skin protection is used; wash exposed area

charges or residue	before or after launch					thoroughly with water.
Ascent						
Rocket flight path	Rocket flies toward objects or people	Risk of blunt force trauma or lacerations	Improbable	Catastrophic	Low	Ensure rocket is pointing in safe direction at launch and launch stand is stable
Descent						
Rocket flight path	Parachute deployed: rocket falls predictively at moderate speeds	Risk of minor injuries	Frequent	Marginal	High	Point at rocket as it descends to ensure personnel are aware of rocket's position and out of harm's way
	Parachute not deployed: rocket falls unpredictably at high speeds	Risk of blunt force trauma or lacerations	Remote	Catastrophic	Moderate	Move out of the way of the rocket, alert others if they are in harm's way
Rocket Motor	Motor can still be hot after flight	Minor burns	Remote	Marginal	Low	Only handle rocket after several minutes on the ground to allow to cool
Sharp Pieces	In event of crash, many pieces may be broken or splintered	Minor wounds	Remote	Marginal	Low	When handling crashed rocket, only place hands on pieces that are visibly intact
Battery Acid	Battery overheating, event of crash	Potential chemical burns	Remote	Marginal	Low	Ensure batteries are properly maintained and operated, flush the affected area with either water or sodium bicarbonate

						solution, depending on specific acid

Appendix G: Environmental Effects on the Rocket

Failure Mode/ Hazard	Cause	Effects	Probability	Severity	Risk	Controls/ Mitigations/ Verifications
Bodies of Water	Launching too close to bodies of water	This can damage the electronic components of the rocket if submerged and the rocket can become irretrievable in the body	Remote	Catastrophic	Moderate Risk	Ensuring that there are no bodies of water near the drift radius of the rocket
Humidity	Launching in excessive humidity	The charges well become wet do to the humidity and be unable to properly ignite	Improbable	Critical	Low Risk	Motors and charges should be stored by certified personnel in a dry place
Lightning	Launching in a thunderstorm	Electrical shock to rocket systems by lightning and can ground the launch	Improbable	Catastrophic	Low Risk	This will ground the launch; no rocket should be launched during a thunderstorm
Low Hanging Clouds	Launching with low cloud cover	It is difficult to keep track or rocket and to properly test all the rocket systems	Occasional	Critical	Moderate Risk	Low hanging clouds should be avoided during launch days paying careful attention to monitor the forecast
Low Temperatures	Launching in extremely cold temperatures	Batteries can discharge at a faster rate and the fiberglass can shrink; the rocket will not be able to perform at its optimum level	Occasional	Critical	Moderate Risk	Battery levels will be monitored by the ground station and battery life will be conserved by turning on the systems at designated times and not leaving them on when function is not necessary
Rain	Launching with risk of rain	Can damage electrical components of rockets and ground the launch	Remote	Catastrophic	Moderate Risk	This will ground the launch; rockets should not be launched in the rain

Swampy Area	Launching in swampy area	If rocket lands in this area it can permanently damage certain components or become irretrievable in the swamp	Remote	Catastrophic	Moderate Risk	Ensuring that there are no swampy areas near the drift radius of the rocket
Trees	Flying near wooded areas	This can damage the rocket and the parachute if caught by a tree and can also cause the rocket to be irretrievable	Occasional	Critical	Moderate Risk	Ensuring that there are no trees near the drift radius of the rocket
UV Rays	Rocket exposed to sun for long periods of time	This can weaken material adhesives if exposed for long durations of time	Improbable	Critical	Low Risk	The rocket will not be exposed to the Sun for a long period of time and extensive work on the rocket will be performed indoors
Wind	Launching in over 20 mile an hour winds	This can reduce the altitude achieved by the rocket, affect the stability of the flight and increase the drift of the parachute, and will ground the launch in excess winds	Improbable	Catastrophic	Low Risk	The launch will be grounded if the winds are too severe and there will be no obstructions in the estimated drift radius

Appendix H: Safety Concerns for the Environment

Failure Mode/ Hazard	Cause	Effects	Probability	Severity	Risk	Controls/ Mitigations/ Cerifications
Battery Leakage	Improper disposal of damaged or used batteries	Contaminate groundwater and in turn contaminate any organic material that is in the water system	Remote	Critical	Moderate Risk	Using proper battery disposal techniques and ensuring all batteries are not damaged
Carbon Emissions	Using cars to travel to launch sites	Damage the ozone layer with emissions	Occasional	Marginal	Low Risk	Using carpooling as much as possible to minimize the amount of vehicles
Epoxy Leakage	Improper use and disposal of epoxy resin in an uncontrolled environment	Contaminate drinking water, be ingested by wildlife, or pollute as solid waste	Improbable	Critical	Low Risk	Using proper techniques in application to ensure the resin is properly dried and disposing of the resin in designated areas
Field Fire	Igniting rockets near dry grass and shrubs or motor CATO	Set the launch site or other nearby objects on fire	Remote	Critical	Moderate Risk	Making sure that any field in use is not near any shrubs and using the proper launching pad to ensure the ignition doesn't affect the surrounding area
Harmful Gas Emissions	Motors emitting gases upon ignition into the environment	Pollute the atmosphere with harmful substances	Remote	Critical	Moderate Risk	There will not be many launches done by the team so the emissions will not be to a concerning level
Harm to Wildlife	Launching a vehicle in a non-designated area around an animal's natural habitat	Destroy animal habitat and result in loss of food source, water source, or life	Improbable	Critical	Low Risk	Ensuring that we only launch in predesignated areas that will have minimal effect on surrounding wildlife
Plastic/Wire Waste	Stripping a wire on site and not properly disposing of the waste	If not properly disposed it can cause solid waste or be ingested by an animal	Improbable	Critical	Low Risk	Ensuring that any stripped wires have the waste properly collected and disposed

Spray Paint Fumes	Spray painting the rocket	Can contaminate the water supply or atmosphere	Remote	Critical	Moderate Risk	Painting the rocket in a painting booth that properly disposes of waste
Waste	Improper disposal or storage of rocket components	Can result in pollution of environment if improperly disposed or stored.	Improbable	Critical	Low Risk	Correctly storing any piece of the rocket that is still waste and disposing off the rest in the proper fashion
Water/Ground Pollution	Leakage of motor chemicals into the ground and water	Pollute the water system with improper disposal	Improbable	Critical	Low Risk	There will not be many launches done by the team so the pollution will not be to a concerning level

Appendix I: Project Risks

Likelihood: Rare, Unlikely, Even, Probable, Extremely Likely

Impact: Negligible, Low, Moderate, High, Critical

Risk	Likelihood	Impact	Mitigation	Verification
Time Possibility of falling behind schedule and/or missing deadlines	Probable	Low	All aspects of the project will be divided up among team members to reduce the chances of falling behind in work. Additionally, multiple team members will coordinate together to ensure that deadlines are met and to keep each other accountable.	A test-launch will commence Saturday, March 3.
Budget Failure to have enough funds to purchase rocket materials, cover transportation costs, and pay for other expenses	Rare	High	All material costs will be determined prior to construction. The team will determine how much material must be ordered in order to prevent overspending. Similarly, travel/transportation expenses will be planned out. Overall budget and spending plans will help ensure that this constraint is met.	Everything has been successfully paid for.

Equipment and Facility Physical injury associated with on- and off-campus facilities and the material/equipment used to build and operate the rocket	Unlikely	High	Dangerous materials and equipment, including power tools, machinery, and rocket engines, will be used. Every team member will have proper knowledge and training before using laboratories, workshops, materials, and/or equipment. In addition, team members will use personal protective equipment when working with the rocket. The team safety officer, and subteam safety liaisons will communicate proper safety practices.	No team members were injured in the now complete construction of our rocket.
Personnel Potential issues involving team members leaving, which may impact time and budget	Unlikely	Negligible	In the case of someone leaving the team, their responsibilities will be spread among other members.	The rocket and payloads were able to be successfully build with no major absences.
Payload Possibility of malfunctioning or inoperative payload(s)	Unlikely	High	The payload subteams will ensure that work is split among members and adequate time is spent on each step of payload design, construction, and testing. Payload functionality will be verified at the full-scale test launch.	The payloads fit and function within the rocket.
Launch Launch errors and hazards, including defective launch component(s)	Unlikely	Critical	Prior to launch, the rocket will be thoroughly inspected, and all the launch checklists and procedures will be reviewed. Additionally, the	To be verified before Saturday,

			team mentor, David Brunsting, will assist the team at every launch.	March 3 launch.
Recovery Failure of planned rocket recovery, which may result in physical injury or more likely, damage to the rocket and its components	Unlikely	High	The recovery subteam will ensure that the recovery system functions properly by thoroughly designing, constructing, and testing the system. On launch day, following the pre-launch procedures and checklists will reduce recovery system issues. Recovery system functionality will be verified at the full-scale test launch.	There is a triply redundant black powder charge in the CRAM.
Resources Risk of lacking materials, equipment, and facilities to construct and operate the rocket	Rare	High	Each subteam will outline necessary materials, equipment, and facilities prior to construction. Budget and spending plans will also help ensure that all necessary materials are purchased/obtained.	Everything was purchased and build successfully.

Appendix J: Dynamic Model of the Mechanism

The analysis of the crank-slider mechanical system is heavily based on the Vector Loop Method (VLM), as described in *Mechanisms and Machines: Kinematics, Dynamics, and Synthesis* by Michael M. Stanisic (Cengage, 2015). Using the VLM, a vector is assigned to each link in the mechanism, such that Vector 1 starts at the center of the cross piece and extends to the hole at the end of the crosspiece. Vector 2 starts at the same point but on the tie rod, and extends to the other hole in the tie rod, which is coincident with the bolt hole on the drag tab. These vectors have fixed length and a variable angle. Together, they sum to Vector 3, which points from the center of the cross piece to the bolt hole in the drag tab, such that changes in the direction of Vectors 1 and 2 (rotating the cross piece) result in a change in magnitude for Vector 3 (sliding the tab). Note that all three vectors have an angle measured counterclockwise from the positive x-axis, and that the vectors are depicted in Figure 131.

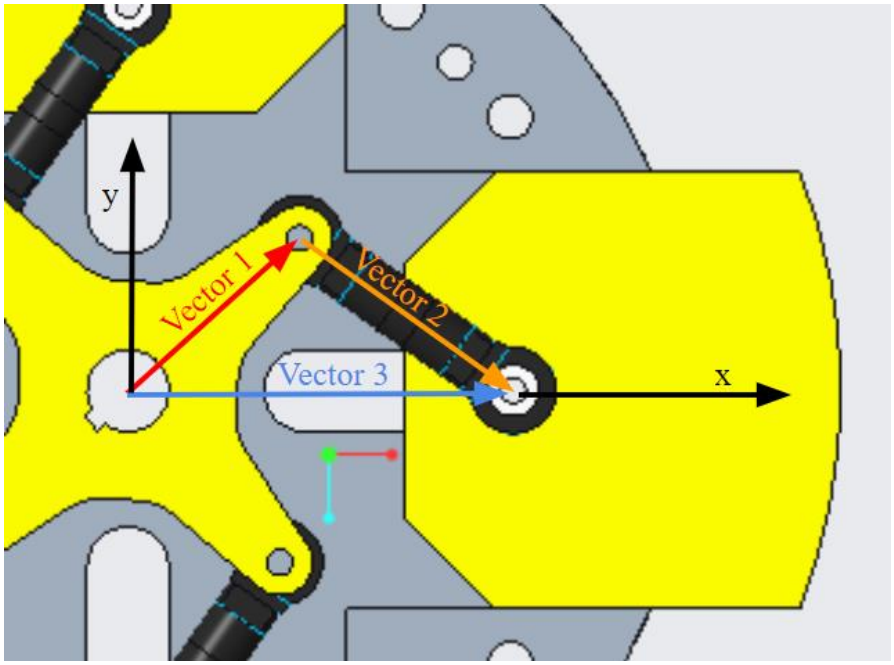


Figure 131. Assignment of vectors for application of the Vector Loop Method.

The mechanism is planar, so the summation of the vectors (note that the sum of all vectors in a loop is always zero) can be broken down into x and y components. For this document, r denotes the magnitude of each vector, while θ denotes its angle.

$$r_1 \cos(\theta_1) + r_2 \cos(\theta_2) + r_3 = 0$$
$$r_1 \sin(\theta_1) + r_2 \sin(\theta_2) = 0$$

By squaring and adding these two equations the value of the cross piece rotation can be solved for directly, as a function of only the drag tab extension.

$$\theta_1 = \arccos \frac{3_3^2 - r_2^2 - r_1^2}{2r_1 r_2}$$

This forms the beginning of the accompanying MATLAB code, provided in Appendix K, which is used to execute the model. The code is given a range of desired values for the tab extension, and then calculates the corresponding link angles. This section omits some calculations and parameters for brevity and clarity, but they are present in the code to adapt this theoretical model to the actual system.

There major force acting on the system is friction due to the drag force. A partial free-body diagram of the tab is shown in Figure 132. The tie rod force, however, does not act exclusively in the direction of sliding, and is only drawn this way for clarity. Because of this, there is a third friction component acting on the side of the tab, which is accounted for in the calculation. If Friction Forces 1 and 2 (caused by corresponding Normal Forces 1 and 2) as shown in Figure 132 are combined to *f*, and μ is the coefficient of friction, then the axial force in the tie rod (F_2) can be solved for.

$$f = \mu(N_1 + N_2)$$
$$F_2 = \frac{f}{\cos\theta_2 - \mu \sin\theta_2}$$

Then, the torque required is simply the magnitude of the cross product of Vector 1 and the tie rod axial force, which is in the opposite direction as Vector 2.

$$T = r_2 F_2(-\cos\theta_1 \sin\theta_2 + \sin\theta_1 \cos\theta_2)$$

This is the torque required for a single tab, so the whole system needs four times this torque to operate. The code does this calculation for every configuration of the system, scaling the drag force linearly with tab extension, and adjusting the dimensions as necessary to calculate the normal forces, before giving the maximum torque for a given dimension.

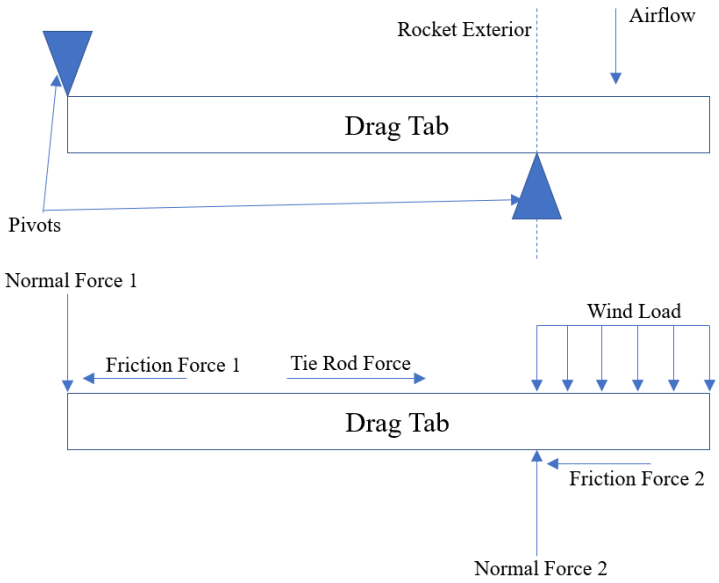


Figure 132. Drag tab free body diagram, side view.

MATLAB Code for performing Vector Loop Analysis

```
% This code simulates the ABP crank-slider mechanism using the
% vector loop method.  This code uses theta_1 as the angle of the
% servo rotation, and theta_2 as the resulting angle of the
% tie rod. Link 1 is the servo horn, link 2 is the tie rod,
% and link 3 is the sliding drag tab.
% all units in inches, pounds, degrees unless otherwise noted

clear all;
close all;
clc

%% kinematics of motion
l_out_max = 1; % max extension of fin, inches
l_out = linspace(0,l_out_max,1000); % vector of extension lengths
l_tab = 2; % tab length
l_in = l_tab - l_out; % inner length

r_rocket = 5.5/2; % outer radius of rocket
inset = 0.5; % distance pin joint inset from interior tab edge
r3 = r_rocket - l_in + inset; % length used for vector loop

r1 = 1.05; % servo horn arm length
r2 = 2.25-r1; % tie rod length
test1=(-(r2.^2) + r1.^2 + r3.^2)./(2.*r1.*r3);
theta_1 = real((acosd((-(r2.^2) + r1.^2 + r3.^2)./(2.*r1.*r3))));
% the theta_1 value is manipulated in the previous step to fit in the right
% quadrant and the correct direction of rotation
fig_num = 1;

% get a quick plot of theta_1 vs l_out for sanity check
```

```
figure(fig_num);
fig_num = fig_num + 1;
plot(theta_1, l_out);
xlabel('Servo Rotation [degrees]');
ylabel('Tab Extension [in]');
set(gca,'fontsize',12,'fontname','times new roman');
grid on

theta_2 = asind((-r1/r2)*sind(theta_1)) + 360;

% sanity check theta_2 plot
figure(fig_num);
fig_num = fig_num + 1;
plot(theta_1, theta_2);
xlabel('\Theta_1 [degrees]');
ylabel('\Theta_2 [degrees]');
set(gca,'fontsize',12,'fontname','times new roman');

%% forces time
f_drag_max = 15.5; % pounds, max force
f_drag = f_drag_max .* l_out ./ l_out_max;

figure(fig_num);
fig_num = fig_num + 1;
plot(l_out, f_drag);
xlabel('l_{out}');
ylabel('f_{drag}');
set(gca,'fontsize',12,'fontname','times new roman');

% normal forces in z-direction
N_out = ((l_in + l_out ./ 2) ./ l_in) .* f_drag;
N_in = N_out - f_drag;

coeff_of_friction = 0.35; % UHMW

f_fric = coeff_of_friction .* (N_in + N_out);

f2 = f_fric ./ (cosd(theta_2) - coeff_of_friction...
    .* sind(theta_2));

figure(fig_num);
fig_num = fig_num + 1;
plot(l_out, N_out)
hold on
plot(l_out, N_in)
plot(l_out, f_drag)
plot(l_out, f_fric)
plot(l_out, f2, 'k-')
xlabel('l_{out}')
h = legend('N_{out}','N_{in}','f_{drag}','friction',...
    'Tie Rod Force F2');

Torque = r1.*f2.*(-cosd(theta_1).*sind(theta_2)...
    + sind(theta_1).*cosd(theta_2));

figure(fig_num);
fig_num = fig_num + 1;
plot(l_out, Torque,'k-')
```

```
xlabel('Tab Extension [in]');
ylabel('Required Torque per Tab [in-lb]');
set(gca,'fontsize',12,'fontname','times new roman');
grid on

figure(fig_num);
fig_num = fig_num + 1;
hold on
plot(l_out, f_drag, 'k-.')
plot(l_out, f_fric, 'k--')
plot(l_out, f2, 'k-')
xlabel('Tab Extension [in]');
ylabel('Force [lb]');
h = legend('Drag','Friction',...
    'Tie Rod');
set(gca,'fontsize',12,'fontname','times new roman');
grid on

maxTorque = max(Torque)
```

Appendix K: Air Braking System Control Code

```
/*
* Notre Dame Rocket Team Air-Breaking Payload Flight Code Version 1.0
*
* Authors: Aidan McDonald, Tommy Flanagan
* Last Update: 02/27/2018
*
* Update description: Incorporated bugfixes discovered in the ground
testing code
*/


#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <Adafruit_BMP280.h>
#include <SPI.h>
#include <SD.h>
#include <Servo.h>


Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);
Adafruit_BMP280 bmp; //I2C barometer initialization; for other versions
check sample code
Servo tabServos;


//Flight-staging constants, for code readability
#define WAITING -1
#define ARMED 0
#define LAUNCHED 1
#define BURNOUT 2
#define APOGEE 3
#define LANDED 4


//Other useful constants
const int chipSelect = 28;
const String dataFileName = "datalog.txt";
const String inFileName = "BESTFL~1.TXT";
const int preCalcSize = 1887; //Number of data points in the pre-
calculated ideal flight layout
const int potPin = A1;
const int servoPin = 6;
const int armPin = 7;
const int baroRegSize = 10; //Number of data points to use in linear
regression
const float seaPressure = 1013.25;
const int potNoiseThreshold = 5; //Degrees
const int maxPropDelay = 250; //millis
const int sdWaitTime = 67; //millis
```

```
const float accelLiftoffThreshold = 50; //m/s^2
const float baroLiftoffThreshold = 10; //m
const float accelBurnoutThreshold = -10; //m/s^2
const float baroApogeeThreshold = 5; //m
const float baroLandedThreshold = 40; //m
const float accelFreefallThreshold = 30; //m/s^2
const float thetaMin = 0; //Degrees. Note that the mechanism is such
that thetaMin causes full extension and thetaMax causes full
retraction.
const float thetaFlush = 50;
const float thetaMax = 70;
const float thetaOffset = 20;
const float maxStep = 10; //Degrees
const float servoJamThreshold = 12; //Degrees, approx. two standard
deviations


//Flags
bool pushed = false;
bool armed = false;
bool saveData = false;
bool runPIDControl = false;
bool emergencyRescue = false;


//Control variables
int flightState = WAITING;
float accelX, accelY, accelZ;
float temperature, pressure, altitude;
float potValue;
float lastA, lastCalcT, lastPIDT, launchT, launchA;
int lastSDT=0;
float bestAlt[preCalcSize], bestVel[preCalcSize];
float theta, velocity = 0, maxA = 0, lastTheta=theta;
float integralTerm = 0, lastError = 0;
float baroRegArr[baroRegSize], timeRegArr[baroRegSize];
int buttonArray[10] = {0,0,0,0,0,0,0,0,0,0};


void setup() {
//Freeze the code if a sensor initialization failure occurs
if(!SD.begin(chipSelect) || !accel.begin() || !bmp.begin())
{
while(1);
}
pinMode(armPin, INPUT);


tabServos.attach(servoPin);
tabServos.write(thetaMax); //Fully retract drag tabs initially
accel.setRange(ADXL345_RANGE_16_G);
altitude = bmp.readAltitude(seaPressure); //Set a baseline starting
altitude
launchA = altitude;
```

```
ReadBestFlight();
PrintHeader();
}


void loop() {
GetSensorData();
switch(flightState){ //Main control section- runs commands and sets
flags based on the current state, updates state where necessary
case WAITING:
RunButtonControl();
if(armed){
flightState = ARMED;
tabServos.write(thetaMin);
delay(1000);
tabServos.write(thetaFlush);
}
break;
case ARMED:
UpdateBaroBuffers();
RunButtonControl();
if(accelZ > accelLiftoffThreshold || (altitude-launchA) >
baroLiftoffThreshold){
flightState = LAUNCHED;
launchT = millis();
saveData = true;
}
if(!armed){
flightState = WAITING;
tabServos.write(thetaMax);
}
break;
case LAUNCHED:
if(fabs(altitude-lastA) > 0.0001){
UpdateBaroBuffers();
velocity = CalcBaroVel();
}
if(accelZ < accelBurnoutThreshold){
flightState = BURNOUT;
runPIDControl = true;
}
break;
case BURNOUT:
velocity = CalcAccelVel(velocity);
if(maxA > altitude + baroApogeeThreshold){
flightState = APOGEE;
runPIDControl = false;
tabServos.write(thetaFlush);
}
break;
case APOGEE:
velocity = CalcAccelVel(velocity);
if(fabs(velocity) > accelFreefallThreshold)
```

```
emergencyRescue = true;
if(fabs(altitude - launchA) <= baroLandedThreshold){
flightState = LANDED;
tabServos.write(thetaMax);
saveData = false;
emergencyRescue = false;
}
break;
case LANDED:
break;
}
//Run other subroutines based on the states of flags
if(runPIDControl){
float error = CalcError(altitude, velocity, (int)(millis()-
launchT)/10);
theta = PID(error, lastError, integralTerm, millis()-lastPIDT);
lastError = error;
lastPIDT = millis();
tabServos.write(theta+thetaOffset); //Account for occasional servo
slippage in testing by adding a constant offset
}
if(saveData){
if(millis()-lastSDT > sdWaitTime){ //Only save data once every few
cycles for the sake of processing speed
SaveSensorData();
lastSDT = millis();
}
}
if(emergencyRescue)
tabServos.write(thetaMin); //Fully deploy tabs if free fall is
occurring


}




//Functions:
void ReadBestFlight(){ //Read in ideal velocity and altitude data
File inFile = SD.open(inFileName);
int c = 0; //Counter variable
if(inFile){
while(inFile.available()){
bestVel[c] = inFile.parseFloat();
bestAlt[c] = inFile.parseFloat();
c++;
}
inFile.close();
}
else{
while(1); //Freeze code if comparison dataset cannot be read
}
}
```

```
void UpdateBaroBuffers(){ //Cycle barometer buffer data for linear
regression purposes
for(int c = 0; c < baroRegSize-1; c++){
baroRegArr[c] = baroRegArr[c+1];
timeRegArr[c] = timeRegArr[c+1];
}
baroRegArr[baroRegSize-1] = altitude;
timeRegArr[baroRegSize-1] = millis();
}


float CalcBaroVel(){ //Calculate velocity based on stored barometric
data
float sumX = 0, sumY = 0, sumXX = 0, sumXY = 0;
for(int c = 0; c < baroRegSize; c++){ //Calculate variance and
covariance components; fill time buffer
sumX += timeRegArr[c];
sumY += baroRegArr[c];
sumXX += pow(timeRegArr[c],2);
sumXY += timeRegArr[c]*baroRegArr[c];
}
double slope = (baroRegSize*sumXY - sumX*sumY) / (baroRegSize*sumXX -
sumX*sumX); //Slope = covariance/variance
return slope;
}


float CalcAccelVel(float lastVel){ //Perform numeric integration to
calculate velocity based on acceleration data
float newVel = lastVel + accelZ*(millis()-lastCalcT)/1000;
lastCalcT = millis();
return newVel;
}


float CalcError(float realAlt, float realVel, int startT){ //First PID
control function
//Looks up closest matching altitude point in the model data, and
calculates error based on the corresponding model velocity
bool match = false;
if(startT > preCalcSize-1) //Limit starting point to the size of the
array
startT = preCalcSize-1;
int c = startT; //To optimize search time, start at the current time
point in the model dataset
int last_c = c;


while(!match){ //Find closest altitude in buffer to current altitude
float test = fabs(realAlt - bestAlt[c]); //Calculate delta-Y at, above,
and below current test value
float above = fabs(realAlt - bestAlt[c+1]);
```

```
float below = fabs(realAlt - bestAlt[c-1]);
last_c = c;
if(test <= above && test <= below) //If current point has least error,
we have a match
match = true;
else if (above < test) //Otherwise, go up or down accordingly
c++;
else if (below < test)
c -= 1;
if(last_c == c) //Backup code in case an infinite search loop is
entered
match = true;
}
return (realVel - bestVel[c]); //Return difference in velocities at the
given altitude point
}


float PID(float error, float lastE, float &iTerm, int deltaT){ //Second
PID control function; returns an output angle based on the error
static float cP = 80; //P constant
static float cI = 0; //I constant
static float cD = 4; //D constant
float dT = (float)deltaT/1000; //Variable delta-T term


float pTerm = error*cP; //Calculate each term
iTerm = iTerm + error*cI*dT; //The i-term is passed by reference and
updated throughout flight
float dTerm = (error - lastE)*cD/dT;


float thetaOut = pTerm + iTerm + dTerm; //calculate intended output
angle
thetaOut = thetaFlush-thetaOut; //Flip because min is full extension
while flush is full retraction
if(thetaOut < thetaMin) //Limit the output to the range of possible
values
thetaOut = thetaMin;
else if(thetaOut > thetaFlush)
thetaOut = thetaFlush;


//Subroutine to limit servo jitteriness by restricting how far one
output can deviate from the previous output
if(thetaOut > lastTheta + maxStep)
thetaOut = lastTheta + maxStep;
if(thetaOut < lastTheta - maxStep)
thetaOut = lastTheta - maxStep;
lastTheta=thetaOut;
return thetaOut; //With all transformations complete, return the output
angle
}
```

```
void GetSensorData(){ //Read in data from all sensors
potValue = analogRead(potPin); //Read potentiometer data
sensors_event_t event; //Read accelerometer data
accel.getEvent(&event);
accelX = event.acceleration.x;
accelY = event.acceleration.y;
accelZ = event.acceleration.z;
temperature = bmp.readTemperature();
pressure = bmp.readPressure();
lastA = altitude; //Variable to track if new altitude data has come in
altitude = bmp.readAltitude(seaPressure)-launchA; //Shift all altitude
data relative to the starting point
if(altitude > maxA)
maxA = altitude; //Also track maximum altitude
}


void SaveSensorData(){ //Save data from all sensors
File dataLog = SD.open(dataFileName, FILE_WRITE);
if(dataLog){
dataLog.print(millis()); dataLog.print(","); dataLog.flush();
dataLog.print(accelX); dataLog.print(","); dataLog.flush();
dataLog.print(accelY); dataLog.print(","); dataLog.flush();
dataLog.print(accelZ); dataLog.print(","); dataLog.flush();
dataLog.print(temperature); dataLog.print(","); dataLog.flush();
dataLog.print(pressure); dataLog.print(","); dataLog.flush();
dataLog.print(altitude); dataLog.print(","); dataLog.flush();
dataLog.print(potValue); dataLog.print(","); dataLog.flush();
dataLog.println(Check_Jam()); dataLog.flush();
dataLog.close();
}
}


void PrintHeader(){ //Print a descriptive header to the SD datalog
File dataLog = SD.open(dataFileName, FILE_WRITE);
if(dataLog){
dataLog.print("Time,"); dataLog.flush();
dataLog.print("X Accel,"); dataLog.flush();
dataLog.print("Y Accel,"); dataLog.flush();
dataLog.print("Z Accel,"); dataLog.flush();
dataLog.print("Vertical Velocity,"); dataLog.flush();
dataLog.print("Temperature,"); dataLog.flush();
dataLog.print("Pressure,"); dataLog.flush();
dataLog.print("Altitude,"); dataLog.flush();
dataLog.print("Intended Position,"); dataLog.flush();
dataLog.print("Encoder Value,"); dataLog.flush();
dataLog.println("Jammed?"); dataLog.flush();
dataLog.close();
}
}


bool Check_Jam(){ //Check if the tabs are jammed
```

```
float realTheta = (potValue-330)/10.314; //Always calibrate these
constants before flight
if(fabs(realTheta-lastTheta) > servoJamThreshold)
return true;
else
return false;
}


void RunButtonControl(){ //Button debouncing subroutine for arming /
disarming the rocket
float threshold = 0.8;
for(int c=9; c>0; c--)
buttonArray[c] = buttonArray[c-1];
buttonArray[0] = digitalRead(armPin);
float avg = 0;
for(int c=0; c<10; c++)
avg += buttonArray[c];
avg /= 10;
if(!pushed && avg > threshold){
armed = !armed;
pushed = true;
}
else if(avg <= threshold)
pushed = false;
}
```

Appendix L: Vehicles Design Budget

Vehicle Design Budget				
	Material	Quantity	Per unit Price	Total Price
Subscale	Polypropylene Nose Cone	1	$20.74	$20.74
	Phenolic Body Tube	1	$4.99	$4.99
	Bulkheads, Centering Rings, Fins (cut from same material)	1	$12.61	$12.61
	Couplers	1	$6.39	$6.39
	Motor Mount	1	$4.99	$4.99
	Transition Section Material	1	$5.98	$5.98
	Motors	2	$27.99	$55.98
	Motor Retention	1	$24.61	$24.61
	Subtotal			$136.29
Full Scale	Motor Casing	1	$331.65	$331.65
	PNC-7.51" Nose Cone	1	$87.95	$87.95
	Motor Retention	1	$47.08	$47.08
	Plywood for Fins, Bulkhead and Centering Rings	5	$12.00	$60.00
	Body Tubes (5.38 LOC)	3	$38.50	$115.50
	Fiberglass Body Tube (7.5" Diameter)	1	$280.00	$280.00
	5.38" LOC Coupler	3	$9.08	$27.24
	Rail Button Set	1	$10.00	$10.00
	75mm LOC Motor Mount	1	$14.95	$14.95
	10-32 Threaded Rods 2' long	6	$2.15	$12.90
	10-32 Heavy Vibration Lock Nuts	1	$12.19	$12.19
	10-32 Hex Nuts	1	$3.77	$3.77
	Number 10 Screw Size Washer	1	$4.24	$4.24
	3/8"-16 Steel Eyebolt with Shoulder	3	$5.32	$15.96
	3/8"-16 Hight Strength Steel Nylon Insert Locknut	1	$3.20	$3.20
	3/8"-16 Hight Strength Steel Hex Nut	1	$6.61	$6.61
	3/8" Screw Size Mil. Spec Split Lock Washer	1	$5.94	$5.94
	7.51" LOC Coupler	1	$15.95	$15.95
	Motors	5	$246.95	$1,234.75
	Quick Links	6	$1.50	$9.00
	Subtotal:			$2,298.88
	RocketPoxy	1	$65.00	$65.00

Multi-Purpose Material	JB Weld	1	$19.99	$19.99
	15 Minute Mid Cure Epoxy	3	$13.00	$39.00
	30 Minute Mid Cure Epoxy	3	$13.00	$39.00
	Subtotal:			$162.99
	Total:			$2,598.16

Appendix M: Recovery System Budget

Recovery System Budget			
Material	**Quantity**	**Price per Unit**	**Total Cost**
Main parachute	1	$620	$620
Nomex (tubular)	2	$15	$30
Nomex (square)	4	$3.75	$15
PVC	1	$5	$5
Acrylic	1	$15	$15
Copper plating	1	$12	$12
Altimeter	1	$155	$155
Shock cords	1	$70	$70
9V battery boxes	3	$3	$9
9V batteries	3	$6	$18
Wire	1	$6	$6
Wire connectors	20	$1	$20
TOTAL			**$975**

Appendix N: Deployable Rover Payload Budget

Deployable Rover Budget	
Item	**Price**
HDPE block	$7.46
Microcontroller	$10.00
Altimeter	$5.00
LoRa	$15.00
Gyroscope	$7.00
Lidar	$150.00
Batteries (2 sets of 4)	$80.00
PCB Boards (2 sets of 3)	$80.00
Wheels (2 sets of 2)	$27.98
Solar Panels (set of 100, they are very fragile, we will probably break some while figuring out how to manufacture the array)	$38.68
Various Hardware/Tools	$100.00
Servomotor	$69.99
Lego Motors (x4)	$39.96
Ejection System	$50.00
Gear Rack	$33.73
Servo Gear	$15.00
3D printed components (tracks, securing cubes, covers)	$40.00
Hinges	$3.00

Total Shipping	$33.93
TOTAL	**$806.73**

Appendix O: Air Braking System Budget

Air-Braking System						Total Weight (oz)	Total Cost
For parts that come in larger quantities, cost per is omitted and total cost is overriden with total pack cost. Similarly, custom part costs are given as the total cost of the required stock. For these parts, per part cost is listed as zero. Total costs listed as zero come from stock also used by other parts.						13.890	$299.81
Component	**Supplier**	**Part No or CAD File**	**Qty.**	**Cost Per Part**	**Total Cost**	**Weight Per Part (oz)**	**Total Weight (oz)**
Top Slotted Plate	Custom	ABPME171227_V2_Channel_Base.prt	1	$0	$22.09	0.000	0.000
Other Sliding Plate	Custom	ABPME171227_V2_Channel_TopPlate.prt	1	$0	$0	0.000	0.000
Drag Tab	Custom	ABPME171227_V2_Drag_Tab.prt	4	$0	$13.61	0.000	0.000
Cross Piece	Custom	ABPME171227_V2_Crossarm.prt	1	$0	$0	0.000	0.000
Drive Shaft	Custom	ABPME171227_V1_Drive_Shaft.prt	1	$0	$11.94	0.000	0.000
Servo Mount Plate	Custom	ABPME180103_V1_Servo_Mount.prt	1	$0	$0	0.000	0.000
Tapped Servo Mount Standoff	Custom	ABPME180103_Servo_Mount_Spacer.prt	4	$0	$6.25	0.000	0.000
Potentiometer Mount	Custom	ABPME180103_Pot_Mount.prt	1	$0	$0	0.000	0.000
32DP 24T Potentiometer Gear	Custom	32DP_24Tooth_20PA_Gear.prt	1	$0	$0	0.000	0.000
Cross Piece Spacer	Custom	ABPME180103_V1_CrossarmSpacer.prt	1	$0	$1.71	0.000	0.000

Tie Rod	Tower Hobbies	LXGFVE	8	$0	$29.99	0.080	0.640
PowerHD 1235MG Servo	Banana Robotics	BR010234	2	$49.99	$99.98	5.820	11.640
0.3125" Clamping Hub	ServoCity	545592	2	$5.99	$11.98	0.000	0.000
0.375" Clamping Hub	ServoCity	545596	1	$5.99	$5.99	0.000	0.000
48T 32DP 20PA Gear	ServoCity	615190	3	$12.99	$38.97	0.300	0.900
P3 R25W Potentiometer	P3 America	R25W-C100-R10K	1	$6.00	$6.00	0.710	0.710
3/8" ID Bronze Bushing	McMaster-Carr	1677K4	2	$1.05	$2.10	0.000	0.000
Retaining Ring for 3/8" Shaft	McMaster-Carr	97633A170	2	$0	$8.74	0.000	0.000
6-32x3/8" Steel Socket Head Screw	McMaster-Carr	91251A146	12	$0	$8.42	0.000	0.000
6-32x3/4" Nylon Socket Head Screw	McMaster-Carr	95868A301	2	$0	$6.07	0.000	0.000
M3x14mm Steel Low Profile Socket Head Screw	McMaster-Carr	93070A071	8	$0	$6.14	0.000	0.000
10-32x1.5" Nylon Socket Head Screw	McMaster-Carr	95868A096	4	$0	$6.40	0.000	0.000
10-32x5/8" Nylon Socket Head Screw	McMaster-Carr	95868A090	12	$0	$6.71	0.000	0.000
6-32x1/4" Nylon Flat Tip Set Screw	McMaster-Carr	94564A023	1	$0	$5.48	0.000	0.000
3/32" Steel Square	McMaste	98830A050	1	$0	$1.24	0.000	0.000

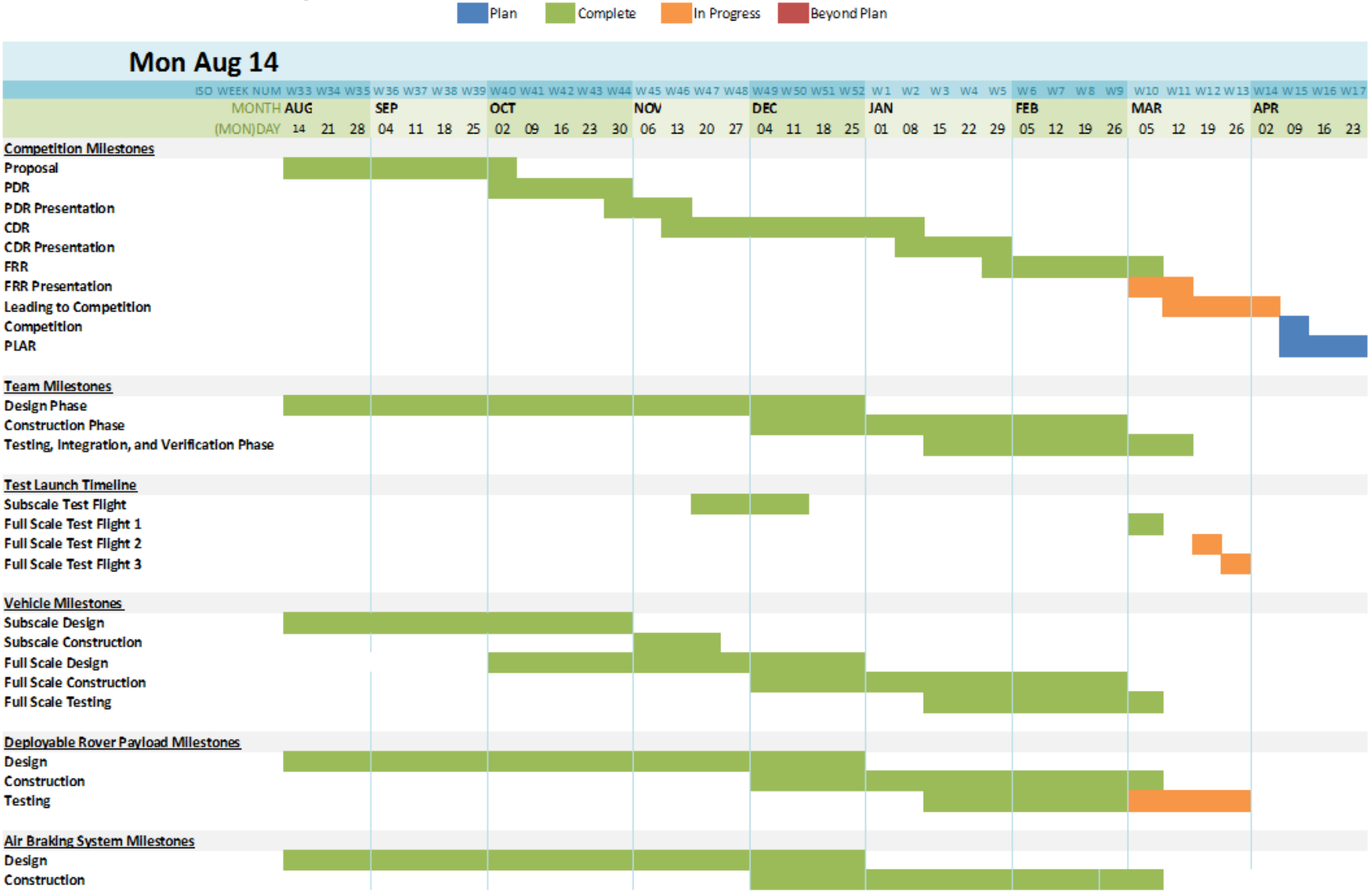
Key	r-Carr						
Arduino MKR Zero	Arduino		1	$21.90	$21.90	0.332	0.332
ADXL345	Adafruit		1	$17.50	$17.50	0.045	0.045
BMP280	Adafruit		1	$9.95	$9.95	0.046	0.046
Adafruit Li-Ion Battery 3.7 V 2000 mAh	Adafruit	2011	1	$12.50	$12.50	1.199	1.199
Tenergy Li-Ion Battery 7.4 V 2600 mAh	All-Battery	31004	2	$19.99	$39.98	3.492	6.984

Appendix P: Team Travel Budget

	Details	Number of Item	Cost Per Day	Item Cost
Housing (Team)	AirBNB House	4 nights, 5 days		$1,884.96
Housing (Mentors)	Extended Stay America Hotal, 1 room	4 nights, 5 days		$318.45
Meals	$10 / meal, 3 meals / day / person	15 meals / person x 23 people	$720.00	$3,600.00
Transportation (ND Vans)	$55 / day / van	5 vans, 5 days	$275.00	$1,375.00
Gas	1300 miles (there and back, all driving in Huntsville)	20 MPG at $2.50 / gal		$812.50
TOTAL				**$7,990.91**

Appendix Q: Timeline

NDRT 2017-2018 NSL Competition Plan



Appendix R: Deployable Rover Payload Code

CODE:

```
/*
 * File:   mainProgram.c
 * Author: John
 *
 * Created on December 13, 2017, 11:03 AM
 */

#include "config_pins.h"
#include "config_general_macros.h"
#include "config_addresses.h"
#include "uart_config.h"
#include "lora_config.h"
#include "timer_config.h"
#include "driveFunctions.h"
#include "interrupt_config.h"
#include "config.h"
#include <xc.h>
#include "I2C_function.h"
#include "lidarHeader.h"
#include "BLE_functions.h"
#include "setup_devices_file.h"
#include "GAM_header.h"
#include "altimeter_header.h"
#include "interrupt_config_UART.h"
#include "interrupt_config_UART_LoRa.h"

/*
 *
 */
void FRR_Routine(void);
void deployRoutine(void);
void moveRover(void);
void deploySolarPanels(void);
void config_outputCompare(void);
void prepareForFlight(void);
void armRover(void);
void function_that_does_something_with_data(unsigned char data[received_string_length-received_data_offset]);
int main(int argc, char** argv) {
    unsigned char GPSchar;
    setup_devices();
    configure_uart();
    configI2C();
    configure_interrupts();
    configure_timers();
```

```
    config_outputCompare();
    //initializeGAM();
    //initializeAlt();
    //timer_test();
    PWM_Timer_On = true;
    OC2CONbits.ON = true;
//    DRIVE_Timer_On = true;
    MEASURE_Timer_On = true;
    while (true)
    {
       LoRa();
    }
    return (EXIT_SUCCESS);
}
void deployRoutine(void) {
    DEPLOY_Timer_Prescale_8;
    DEPLOY_Timer_Period = 20000;
    MEASURE_Timer_Prescale_256;
    MEASURE_Timer_Period = 2500;
    MEASURE_Timer_On = true;
    DEPLOY_Timer_On = true;
}

void deploySolarPanels(void) {
    PWM_Timer_On = true;
    OC2CONbits.ON = true;
    MEASURE_Timer_On = true;
}
void config_outputCompare(void) {
    OC2CONbits.ON = 0;
    OC2RS = 1872;
    OC2R = 1872;
    OC2CONbits.OC32 = 0;
    OC2CONbits.OCTSEL = 0;
    OC2CONbits.OCM = 0b110;
}
void prepareForFlight(void){
    PWM_Timer_On = true;
    OC2CONbits.ON = true;


}
void function_that_does_something_with_data(unsigned char data[received_string_length-received_data_offset]) {
 //printf("Received data: %s\r\n",data);
 switch (data[0]) {
     case 'C':
         if(data[1] == '1') {
```

```
            DEPLOY_Timer_On = false;
            MEASURE_Timer_On = false;
            DRIVE_Timer_On = false;
            OC2RS = 1872;
            LIGHT_POWDER1 = high;
//            LIGHT_POWDER1 = low;
        }
        else if (data[1] == '2') {
            DEPLOY_Timer_On = false;
            MEASURE_Timer_On = false;
            DRIVE_Timer_On = false;
            OC2RS = 1872;
            LIGHT_POWDER2 = high;
//            LIGHT_POWDER2 = low;
        }
        break;
    case 'D':
        if (data[1] == '1') {
            DEPLOY_Timer_On = false;
            MEASURE_Timer_On = false;
            DRIVE_Timer_On = false;
            OC2RS = 1872;
            driveForward();
            MEASURE_TIMER_CONDITION = 2;
            COUNT_SECONDS = 0;
            MEASURE_Timer_Period = 60000;
            MEASURE_Timer_On = true;

        }
        else if (data[1] == '2') {
            DEPLOY_Timer_On = false;
            MEASURE_Timer_On = false;
            DRIVE_Timer_On = false;
            OC2RS = 1872;
            MEASURE_TIMER_CONDITION = 3;
            deployRoutine();
        }
        else if (data[1] == '3') {
            DEPLOY_Timer_On = false;
            MEASURE_Timer_On = false;
            DRIVE_Timer_On = false;
            OC2RS = 1872;
            coast();
        }
        break;
    case 'A':
        DEPLOY_Timer_On = false;
```

```
      MEASURE_Timer_On = false;
      DEPLOY_Timer_On = false;
      MEASURE_Timer_On = false;
      DRIVE_Timer_On = false;
      OC2RS = 1872;
      if (data[1] == '1'){
        DEPLOY_Timer_On = false;
        MEASURE_Timer_On = false;
        DRIVE_Timer_On = false;
        OC2RS = 1807;
      }
      if (data[1] == 'F') {
        DEPLOY_Timer_On = false;
        MEASURE_Timer_On = false;
        DRIVE_Timer_On = false;
        OC2RS = 1872;
        DRIVE_Timer_On = true;
      }
      if (data[1] == '5') {
        DEPLOY_Timer_On = false;
        MEASURE_Timer_On = false;
        DRIVE_Timer_On = false;
        OC2RS = 1935;
        MEASURE_TIMER_CONDITION = 4;
        MEASURE_Timer_Period = 29000;
        MEASURE_Timer_On = true;
      }
      if (data[1] == '9') {
        DEPLOY_Timer_On = false;
        MEASURE_Timer_On = false;
        DRIVE_Timer_On = false;
        OC2RS = 1872;
      }
      if (data[1] == 'D') {
        DEPLOY_Timer_On = false;
        MEASURE_Timer_On = false;
        DRIVE_Timer_On = false;
        OC2RS = 1935;
        MEASURE_TIMER_CONDITION = 4;
        MEASURE_Timer_Period = 65000;
        MEASURE_Timer_On = true;
      }
      break;

    default:
      printf("Unexpected Command");
}
```

```
 lora_transceiver_state = IDLE;
}

void initializeAlt(void);
void readAlt(void);
float convertPressure(int P_high, int P_mid, float P_low);
#define ALT_CTRL_REG1 0x26
#define ALT_STATUS 0x06

void initializeAlt(void) {
   int reg_1_data = 0x00;
   int PT_DATA = 0x00;
   int reg_1_ACTIVE = 0x00;
   while (reg_1_data != 0xB8){
      startI2C();
      putI2C(ALT_WRITE & 0xFE);
      putI2C(ALT_CTRL_REG1);
      putI2C(0xB8);
      stopI2C();
      startI2C();
      putI2C(ALT_WRITE & 0xFE);
      putI2C(ALT_CTRL_REG1);
      startI2C();
      putI2C(ALT_READ |0x01);
      reg_1_data = getI2C(1);
      stopI2C();
   }
   while (PT_DATA != 0x07) {
      startI2C();
      putI2C(ALT_WRITE & 0xFE);
      putI2C(0x13);
      putI2C(0x07);
      stopI2C();
      startI2C();
      putI2C(ALT_WRITE & 0xFE);
      putI2C(0x13);
      startI2C();
      putI2C(ALT_READ | 0x01);
      PT_DATA = getI2C(1);
      stopI2C();
   }
   while (reg_1_data != 0xB9){
      startI2C();
      putI2C(ALT_WRITE & 0xFE);
      putI2C(ALT_CTRL_REG1);
      putI2C(0xB9);
```

```
        stopI2C();
        startI2C();
        putI2C(ALT_WRITE & 0xFE);
        putI2C(ALT_CTRL_REG1);
        startI2C();
        putI2C(ALT_READ |0x01);
        reg_1_data = getI2C(1);
        stopI2C();
    }
}
void readAlt(void) {
    int check = 0x00;
    int OUT_P_MSB;
    int OUT_P_CSB;
    float OUT_P_LSB;
    float OUT_T_MSB;
    float OUT_T_LSB;
    float newPressure;
    while (check & 0b00001000 > 0){
        startI2C();
        putI2C(ALT_WRITE & 0xFE);
        putI2C(0x00);
        startI2C();
        putI2C(ALT_READ | 0x01);
        check = getI2C(1);
        stopI2C();
        printf("check = %i \n\r", check);
    }
    startI2C();
    putI2C(ALT_WRITE & 0xFE);
    putI2C(0x01);
    startI2C();
    putI2C(ALT_READ | 0x01);
    OUT_P_MSB = getI2C(0);
    OUT_P_CSB = getI2C(0);
    OUT_P_LSB = getI2C(1);
    stopI2C();
    newPressure = convertPressure(OUT_P_MSB, OUT_P_CSB,OUT_P_LSB);
    printf("Current Altitude = %f meters \n\r", newPressure);
}

float convertPressure(int P_high, int P_mid, float P_low){
    int sign;
    int count;
    float TOTAL;
    if (P_high > 0x7F) {
        sign = 1;
```

```
    }
    else {
        sign = 0;
    }
    TOTAL = (long) P_high << 8 | P_mid;
    for (count = 0; count<8; count++) { //convert LSB to decimal
        P_low = P_low/2;
    }
    TOTAL = TOTAL + P_low;
    return TOTAL;
}
-----------------------------------------------------------------------------------------------------
#ifndef CONFIG_H
#define  CONFIG_H
/*
 * File:   BLE_functions.h
 * Author: John
 *
 * Created on February 4, 2018, 2:18 PM
 */

void setGPS(void);
void setBLE1(void);
void setBLE2(void);
void setBLE3(void);

void setGPS(void) {
    LATDbits.LATD0 = 0;
    LATBbits.LATB6 = 1;
    LATBbits.LATB7 = 1;
}
void setBLE1(void) {
    LATDbits.LATD0 = 0;
    LATBbits.LATB6 = 0;
    LATBbits.LATB7 = 0;
}
void setBLE2(void) {
    LATDbits.LATD0 = 0;
    LATBbits.LATB6 = 1;
    LATBbits.LATB7 = 0;
}
void setBLE3(void) {
    LATDbits.LATD0 = 0;
    LATBbits.LATB6 = 0;
    LATBbits.LATB7 = 1;
}
```

```
#include <xc.h>// include processor files - each processor file is guarded.

// TODO Insert appropriate #include <>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>        /* Includes uint16_t definition */
#include <stdbool.h>       /* Includes true/false definition */

// TODO Insert appropriate #include ""
#include "config_bits.h"
#define FRC 8000000 /* 8MHz */
#define SCLK (FRC/2*20/1) /* 80MHz */
#define FPB (SCLK/(1<<OSCCONbits.PBDIV))
#endif    /* CONFIG_H */
-------------------------------------------------------------------------------------------------------------
/*
 * File:   config_addresses.h
 * Author: John
 *
 * Created on February 4, 2018, 2:25 PM
 */

#define ALT_WRITE 0xC0
#define ALT_READ 0xC1
#define lidar_read 0xC5
#define lidar_write 0xC4
#define GAM_WRITE 0xD6
#define GAM_READ 0xD7

#define low 0
#define high 1
#define output 0
#define input 1
#define disabled 0
#define enabled 1
#define active_low_enable 0 /*! @define active_low_enable\ The value assigned to an active low pin to enable it
(0). */
#define active_low_disable 1 /*! @define active_low_disable\ The value assigned to an active low pin to disable it
(1). */

#define MOTOR_A1 LATEbits.LATE6
#define MOTOR_A2 LATEbits.LATE7
#define MOTOR_B1 LATFbits.LATF0
#define MOTOR_B2 LATBbits.LATB4

#define MOTOR_C1 LATEbits.LATE3
```

```
#define MOTOR_C2 LATEbits.LATE4
#define MOTOR_D1 LATEbits.LATE5
#define MOTOR_D2 LATBbits.LATB3

#define LIGHT_POWDER1 LATEbits.LATE0
#define LIGHT_POWDER2 LATEbits.LATE1

#define CHECK_INITIALIZATION LATBbits.LATB0
#define DRIVE_CENTER_ON 10000
#define DRIVE_PERIOD 30000

bool SLOW_MOTOR = 0;
int SWITCH_MODE = 0;
int MEASURE_TIMER_CONDITION = 1;
int COUNT_SECONDS = 0;
#define DRIVE_FORWARD 0
#define DRIVE_BACKWARD 1
#define TURN_LEFT 2
#define TURN_RIGHT 3
#define COAST 4
int in_flight_mode = 0;
int SWITCH_COMMAND = 0;
#define PREPARE_FOR_FLIGHT 1
#define ARM_ROVER 2
#define LIGHT_CHARGES1 3
#define LIGHT_CHARGES2 4
#define DEPLOY_ROVER 5
#define MOVE_ROVER 6
#define DEPLOY_SOLAR_PANELS 7

#ifndef CONFIGURATION_BITS_H
#define  CONFIGURATION_BITS_H

/* using external osc
 peripheral clock = at 40 MHz (80 MHz/2)
*/
//#pragma config FNOSC = FRCPLL // Oscillator selection (Core 80 MHz)
//#pragma config POSCMOD = HS // Primary oscillator mode
//#pragma config FPBDIV = DIV_2 // Peripheral bus clock divider 80/2 = 40 MHz
//#pragma config FSOSCEN = OFF // Secondary oscillator enable

#pragma config FNOSC = FRCPLL // Oscillator selection
#pragma config POSCMOD = HS // Primary oscillator mode
#pragma config FPLLIDIV = DIV_2 // PLL input divider (8 -> 4)
#pragma config FPLLMUL = MUL_20 // PLL multiplier  ( 4x20 = 80)
#pragma config FPLLODIV = DIV_1 // PLL output divider
#pragma config FPBDIV = DIV_8 // Peripheral bus clock divider 80/2 = 40 MHz
```

```
#pragma config FSOSCEN = OFF // Secondary oscillator enable
/* Clock control settings
*/
#pragma config IESO = ON // Internal/external clock switchover
#pragma config FCKSM = CSECME // Clock switching (CSx)/Clock monitor (CMx)
#pragma config OSCIOFNC = OFF // Clock output on OSCO pin enable
/* USB Settings
*/
//#pragma config UPLLEN = OFF // USB PLL enable
//#pragma config UPLLIDIV = DIV_2 // USB PLL input divider
//#pragma config FVBUSONIO = OFF // VBUS pin control
//#pragma config FUSBIDIO = OFF // USBID pin control
/* Other Peripheral Device settings
*/
#pragma config FWDTEN = OFF // Watchdog timer enable
#pragma config WDTPS = PS4096 // Watchdog timer post-scaler
#pragma config FSRSSEL = PRIORITY_7 // SRS interrupt priority
//#pragma config DEBUG = ON

#pragma config    ICESEL = ICS_PGx1        // ICE pin selection


#endif    /* CONFIGURATION_BITS_H */
-----------------------------------------------------------------------------------------------------------------------
/*
 * File:   general_macro_config.h
 * Author: cstgeo
 *
 * Created on February 10, 2018, 8:01 PM
 */

#ifndef GENERAL_MACRO_CONFIG_H
#define  GENERAL_MACRO_CONFIG_H
#include "config.h"

/**
 * Basic
 */

// Pattern Matching
#define CAT(a, ...) PRIMITIVE_CAT(a, __VA_ARGS__)
#define PRIMITIVE_CAT(a, ...) a ## __VA_ARGS__

#define IIF(c) PRIMITIVE_CAT(IIF_, c)
#define IIF_0(t, ...) __VA_ARGS__
#define IIF_1(t, ...) t
```

```
#define COMPL(b) PRIMITIVE_CAT(COMPL_, b)
#define COMPL_0 1
#define COMPL_1 0

#define BITAND(x) PRIMITIVE_CAT(BITAND_, x)
#define BITAND_0(y) 0
#define BITAND_1(y) y

#define INC(x) PRIMITIVE_CAT(INC_, x)
#define INC_0 1
#define INC_1 2
#define INC_2 3
#define INC_3 4
#define INC_4 5
#define INC_5 6
#define INC_6 7
#define INC_7 8
#define INC_8 9
#define INC_9 9

#define DEC(x) PRIMITIVE_CAT(DEC_, x)
#define DEC_0 0
#define DEC_1 0
#define DEC_2 1
#define DEC_3 2
#define DEC_4 3
#define DEC_5 4
#define DEC_6 5
#define DEC_7 6
#define DEC_8 7
#define DEC_9 8


// Detection
#define CHECK_N(x, n, ...) n
#define CHECK(...) CHECK_N(__VA_ARGS__, 0,)
#define PROBE(x) x, 1,
//CHECK(PROBE(~)) // CHECK(PROBE(~)) Expands to 1
//CHECK(xxx) // CHECK(xxx) Expands to 0

#define IS_PAREN(x) CHECK(IS_PAREN_PROBE x)
#define IS_PAREN_PROBE(...) PROBE(~)
//IS_PAREN(()) // IS_PAREN(()) Expands to 1
//IS_PAREN(xx) // IS_PAREN(xx) Expands to 0

#define NOT(x) CHECK(PRIMITIVE_CAT(NOT_, x))
#define NOT_0 PROBE(~)
```

```
//NOT(1) // NOT(1) Expands to 0
//NOT(0) // NOT(0) Expands to 1

#define BOOL(x) COMPL(NOT(x))
#define IF(c) IIF(BOOL(c))
//IF(0)(HELLO) // IF(0)(HELLO) Expands to nothing
//IF(1)(HELLO) // IF(1)(HELLO) Expands to HELLO

#define EAT(...)
#define EXPAND(...) __VA_ARGS__
#define WHEN(c) IF(c)(EXPAND, EAT)

/**
 * Recursion
 */

// Deferred expression
#define EMPTY()
#define DEFER(id) id EMPTY()
#define OBSTRUCT(...) __VA_ARGS__ DEFER(EMPTY)()
#define EXPAND(...) __VA_ARGS__

#define EVAL(...)  EVAL1(EVAL1(EVAL1(__VA_ARGS__)))
#define EVAL1(...) EVAL2(EVAL2(EVAL2(__VA_ARGS__)))
#define EVAL2(...) EVAL3(EVAL3(EVAL3(__VA_ARGS__)))
#define EVAL3(...) EVAL4(EVAL4(EVAL4(__VA_ARGS__)))
#define EVAL4(...) EVAL5(EVAL5(EVAL5(__VA_ARGS__)))
#define EVAL5(...) __VA_ARGS__

//#define REPEAT_INDIRECT() REPEAT
#define REPEAT(count, macro, ...) \
    WHEN(count) \
    ( \
        DEFER(REPEAT_INDIRECT) () \
        ( \
            DEC(count), macro, __VA_ARGS__ \
        ) \
        DEFER(macro) \
        ( \
            DEC(count), __VA_ARGS__ \
        ) \
    )
#define REPEAT_INDIRECT() REPEAT

//An example of using this macro
//#define M(_,i) i
//EVAL(REPEAT(8, M, HELLO, HI)) // 0 1 2 3 4 5 6 7
```

```
#define WHILE(pred, op, ...) \
   IF(pred(__VA_ARGS__)) \
   ( \
      DEFER(WHILE_INDIRECT) () \
      ( \
         pred, op, op(__VA_ARGS__) \
      ), \
      __VA_ARGS__ \
   )
#define WHILE_INDIRECT() WHILE

#define OUTER(i, j) { REPEAT(j, INNER, i) }
#define INNER(j, i) if (j == INC(i)) printf("Match\n");
//EVAL(REPEAT(2, OUTER, 3))

#define NARGS_SEQ(_1,_2,_3,_4,_5,_6,_7,_8,N,...) N
#define NARGS(...) NARGS_SEQ(__VA_ARGS__, 8, 7, 6, 5, 4, 3, 2, 1)
#define IS_1(x) CHECK(PRIMITIVE_CAT(IS_1_, x))
#define IS_1_1 ~, 1,
#define PRED(x, ...) COMPL(IS_1(NARGS(__VA_ARGS__)))
#define OP(x, y, ...) CAT(x##HI();, y), __VA_ARGS__
#define M(...) CAT(__VA_ARGS__)

//M(EVAL(WHILE(PRED, OP, x, y, z))) //Expands to xyz

#define SETUP(...) PRIMATIVE_SETUP(__VA_ARGS__,)
#define PRIMATIVE_SETUP(...) void method() {\
 M(EVAL(WHILE(PRED,OP,__VA_ARGS__)))\
}

//NARGS(Hi, Hello, Hey) // Expands to 3

// Make a FOREACH macro
#define FE_1(WHAT, X) WHAT(X)
#define FE_2(WHAT, X, ...) WHAT(X)FE_1(WHAT, __VA_ARGS__)
#define FE_3(WHAT, X, ...) WHAT(X)FE_2(WHAT, __VA_ARGS__)
#define FE_4(WHAT, X, ...) WHAT(X)FE_3(WHAT, __VA_ARGS__)
#define FE_5(WHAT, X, ...) WHAT(X)FE_4(WHAT, __VA_ARGS__)
#define FE_6(WHAT, X, ...) WHAT(X)FE_5(WHAT, __VA_ARGS__)
#define FE_7(WHAT, X, ...) WHAT(X)FE_6(WHAT, __VA_ARGS__)
#define FE_8(WHAT, X, ...) WHAT(X)FE_7(WHAT, __VA_ARGS__)
#define FE_9(WHAT, X, ...) WHAT(X)FE_8(WHAT, __VA_ARGS__)
//... repeat as needed

#define GET_MACRO(_1,_2,_3,_4,_5,_6,_7,_8,_9,NAME,...) NAME
```

```
#define FOR_EACH(action,...) \
  GET_MACRO(__VA_ARGS__,FE_9,FE_8,FE_7,FE_6,FE_5,FE_4,FE_3,FE_2,FE_1)(action,__VA_ARGS__)


#endif    /* GENERAL_MACRO_CONFIG_H */
-------------------------------------------------------------------------------------------------------
/*
 * File:   config_pins.h
 * Author: cstgeo
 *
 * Created on December 29, 2017, 11:42 AM
 */

#ifndef CONFIG_PINS_H
#define  CONFIG_PINS_H
#include "config.h"

// Pin TRIS Mapping
/* TRISB */
 #define PicKit3_PGD_IO TRISBbits.TRISB0
 #define PicKit3_PGC_IO TRISBbits.TRISB1
 #define BLE_nCS1_IO    TRISBbits.TRISB2
 #define BLE_nCS2_IO    TRISBbits.TRISB3
 #define BLE_nCS3_IO    TRISBbits.TRISB4
 // TRISBbits.TRISB5
 #define MUX_A_IO       TRISBbits.TRISB6
 #define MUX_B_IO       TRISBbits.TRISB7
 #define LORA_nRTS_IO   TRISBbits.TRISB8 /* UART 2 !CTS */
 #define GPS_nRESET_IO  TRISBbits.TRISB9
 // TRISBbits.TRISB10
 // TRISBbits.TRISB11
 // TRISBbits.TRISB12
 // TRISBbits.TRISB13
 #define LORA_nCTS_IO   TRISBbits.TRISB14 /* UART 2 !RTS */
 #define GPS_3DFIX_IO   TRISBbits.TRISB15

/* TRISC */
 // TRISCbits.TRISC12
 #define ALT_INT1_IO    TRISCbits.TRISC13
 #define ALT_INT2_IO    TRISCbits.TRISC14
 // TRISCbits.TRISC15

/* TRISD */
 #define MUX_nE_IO      TRISDbits.TRISD0
 #define RS232_nCTS_IO  TRISDbits.TRISD1 /* UART 1 !RTS */
 #define RS232_TXD_IO   TRISDbits.TRISD2 /* UART 1 RXD */
 #define RS232_RXD_IO   TRISDbits.TRISD3 /* UART 1 TXD */
```

```
 #define GAM_INT_M_IO   TRISDbits.TRISD4
 #define GAM_INT1_AG_IO TRISDbits.TRISD5
 #define GAM_INT2_AG_IO TRISDbits.TRISD6
 #define GAM_DRDY_IO    TRISDbits.TRISD7
 #define GAM_DEN_AG_IO  TRISDbits.TRISD8
 #define RS232_nRTS_IO  TRISDbits.TRISD9 /* UART 1 !CTS */
 #define GAM_SDO_AG_IO  TRISDbits.TRISD10
 #define GAM_SDO_M_IO   TRISDbits.TRISD11

/* TRISE */
 #define BLE_nRESET1_IO          TRISEbits.TRISE0
 #define BLE_nAT1_IO    TRISEbits.TRISE1
 #define BLE_nPWRC1_IO  TRISEbits.TRISE2
 #define BLE_nRESET2_IO          TRISEbits.TRISE3
 #define BLE_nAT2_IO    TRISEbits.TRISE4
 #define BLE_nPWRC2_IO  TRISEbits.TRISE5
 #define BLE_nRESET3_IO TRISEbits.TRISE6
 #define BLE_nAT3_IO    TRISEbits.TRISE7

/* TRISF */
 #define BLE_nPWRC3_IO           TRISFbits.TRISF0
 #define LORA_nRESET_IO TRISFbits.TRISF1
 #define USBID_IO       TRISFbits.TRISF3
 #define LORA_TXD_IO  TRISFbits.TRISF4 /* UART 2 RXD */
 #define LORA_RXD_IO  TRISFbits.TRISF5 /* UART 2 TXD */

/* TRISG */
 #define D_MINUS_IO     TRISGbits.TRISG2
 #define D_PLUS_IO      TRISGbits.TRISG3
 #define MUX_RXD_IO     TRISGbits.TRISG6 /* UART 6 TXD */
 #define I2C_SDA_IO       TRISGbits.TRISG7 /* SDA4 */
 #define I2C_SCL_IO       TRISGbits.TRISG8 /* SCL4 */
 #define MUX_TXD_IO   TRISGbits.TRISG9 /* UART 6 RXD */

// Pin PORT Mapping
/* PORTB */
 #define PicKit3_PGD_R  PORTBbits.RB0
 #define PicKit3_PGC_R  PORTBbits.RB1
 #define BLE_nCS1_R     PORTBbits.RB2
 #define BLE_nCS2_R     PORTBbits.RB3
 #define BLE_nCS3_R     PORTBbits.RB4
 // PORTBbits.RB5
 #define MUX_A_R        PORTBbits.RB6
 #define MUX_B_R        PORTBbits.RB7
 #define LORA_nRTS_R    PORTBbits.RB8 /* UART 2 !CTS */
 #define GPS_nRESET_R   PORTBbits.RB9
 // PORTBbits.RB10
```

```
 // PORTBbits.RB11
 // PORTBbits.RB12
 // PORTBbits.RB13
 #define LORA_nCTS_R    PORTBbits.RB14 /* UART 2 !RTS */
 #define GPS_3DFIX_R    PORTBbits.RB15

/* PORTC */
 // PORTCbits.RC12
 #define ALT_INT1_R     PORTCbits.RC13
 #define ALT_INT2_R     PORTCbits.RC14
 // PORTCbits.RC15

/* PORTD */
 #define MUX_nE_R       PORTDbits.RD0
 #define RS232_nCTS_R   PORTDbits.RD1 /* UART 1 !RTS */
 #define RS232_TXD_R    PORTDbits.RD2 /* UART 1 RXD */
 #define RS232_RXD_R    PORTDbits.RD3 /* UART 1 TXD */
 #define GAM_INT_M_R    PORTDbits.RD4
 #define GAM_INT1_AG_R  PORTDbits.RD5
 #define GAM_INT2_AG_R  PORTDbits.RD6
 #define GAM_DRDY_R     PORTDbits.RD7
 #define GAM_DEN_AG_R   PORTDbits.RD8
 #define RS232_nRTS_R   PORTDbits.RD9 /* UART 1 !CTS */
 #define GAM_SDO_AG_R   PORTDbits.RD10
 #define GAM_SDO_M_R    PORTDbits.RD11

/* PORTE */
 #define BLE_nRESET1_R            PORTEbits.RE0
 #define BLE_nAT1_R     PORTEbits.RE1
 #define BLE_nPWRC1_R   PORTEbits.RE2
 #define BLE_nRESET2_R            PORTEbits.RE3
 #define BLE_nAT2_R     PORTEbits.RE4
 #define BLE_nPWRC2_R   PORTEbits.RE5
 #define BLE_nRESET3_R  PORTEbits.RE6
 #define BLE_nAT3_R     PORTEbits.RE7

/* PORTF */
 #define BLE_nPWRC3_R             PORTFbits.RF0
 #define LORA_nRESET_R  PORTFbits.RF1
 #define USBID_R        PORTFbits.RF3
 #define LORA_TXD_R    PORTFbits.RF4 /* UART 2 RXD */
 #define LORA_RXD_R    PORTFbits.RF5 /* UART 2 TXD */

/* PORTG */
 #define D_MINUS_R      PORTGbits.RG2
 #define D_PLUS_R       PORTGbits.RG3
 #define MUX_RXD_R      PORTGbits.RG6 /* UART 6 TXD */
```

```
 #define I2C_SDA_R        PORTGbits.RG7 /* SDA4 */
 #define I2C_SCL_R        PORTGbits.RG8 /* SCL4 */
 #define MUX_TXD_R     PORTGbits.RG9 /* UART 6 RXD */

// Pin LAT Mapping
/* LATB */
 #define PicKit3_PGD    LATBbits.LATB0
 #define PicKit3_PGC    LATBbits.LATB1
 #define BLE_nCS1       LATBbits.LATB2
 #define BLE_nCS2       LATBbits.LATB3
 #define BLE_nCS3       LATBbits.LATB4
 // LATBbits.LATB5
 #define MUX_A          LATBbits.LATB6
 #define MUX_B          LATBbits.LATB7
 #define LORA_nRTS      LATBbits.LATB8 /* UART 2 !CTS */
 #define GPS_nRESET     LATBbits.LATB9
 // LATBbits.LATB10
 // LATBbits.LATB11
 // LATBbits.LATB12
 // LATBbits.LATB13
 #define LORA_nCTS      LATBbits.LATB14 /* UART 2 !RTS */
 #define GPS_3DFIX      LATBbits.LATB15

/* LATC */
 // LATCbits.LATC12
 #define ALT_INT1       LATCbits.LATC13
 #define ALT_INT2       LATCbits.LATC14
 // LATCbits.LATC15

/* LATD */
 #define MUX_nE                 LATDbits.LATD0
 #define RS232_nCTS    LATDbits.LATD1 /* UART 1 !RTS */
 #define RS232_TXD     LATDbits.LATD2 /* UART 1 RXD */
 #define RS232_RXD     LATDbits.LATD3 /* UART 1 TXD */
 #define GAM_INT_M     LATDbits.LATD4
 #define GAM_INT1_AG   LATDbits.LATD5
 #define GAM_INT2_AG   LATDbits.LATD6
 #define GAM_DRDY      LATDbits.LATD7
 #define GAM_DEN_AG    LATDbits.LATD8
 #define RS232_nRTS    LATDbits.LATD9 /* UART 1 !CTS */
 #define GAM_SDO_AG    LATDbits.LATD10
 #define GAM_SDO_M     LATDbits.LATD11

/* LATE */
 #define BLE_nRESET1   LATEbits.LATE0
 #define BLE_nAT1      LATEbits.LATE1
 #define BLE_nPWRC1    LATEbits.LATE2
```

```
 #define BLE_nRESET2   LATEbits.LATE3
 #define BLE_nAT2      LATEbits.LATE4
 #define BLE_nPWRC2    LATEbits.LATE5
 #define BLE_nRESET3   LATEbits.LATE6
 #define BLE_nAT3      LATEbits.LATE7

/* LATF */
 #define BLE_nPWRC3    LATFbits.LATF0
 #define LORA_nRESET   LATFbits.LATF1
 #define USBID         LATFbits.LATF3
 #define LORA_TXD      LATFbits.LATF4 /* UART 2 RXD */
 #define LORA_RXD      LATFbits.LATF5 /* UART 2 TXD */

/* LATG */
 #define D_MINUS       LATGbits.LATG2
 #define D_PLUS        LATGbits.LATG3
 #define MUX_RXD       LATGbits.LATG6 /* UART 6 TXD */
 #define I2C_SDA       LATGbits.LATG7 /* SDA4 */
 #define I2C_SCL       LATGbits.LATG8 /* SCL4 */
 #define MUX_TXD       LATGbits.LATG9 /* UART 6 RXD */

// Pin CNEN Mapping
typedef union {
  struct {
    unsigned _ALT_INT2:1;  //CNEN0:1;
    unsigned _ALT_INT1:1;  //CNEN1:1;
    unsigned :1;  //CNEN2:1;
    unsigned :1;  //CNEN3:1;
    unsigned _BLE_nCS1:1;  //CNEN4:1;
    unsigned _BLE_nCS2:1;  //CNEN5:1;
    unsigned _BLE_nCS3:1;  //CNEN6:1;
    unsigned :1;  //CNEN7:1;
    unsigned :1;  //CNEN8:1;
    unsigned :1;  //CNEN9:1;
    unsigned :1; //CNEN10:1;
    unsigned :1; //CNEN11:1;
    unsigned _GPS_3DFIX:1; //CNEN12:1;
    unsigned _GAM_INT_M:1; //CNEN13:1;
    unsigned _GAM_INT1_AG:1; //CNEN14:1;
    unsigned _GAM_INT2_AG:1; //CNEN15:1;
    unsigned _GAM_DRDY:1; //CNEN16:1;
    unsigned :1; //CNEN17:1;
    unsigned :1; //CNEN18:1;
  };
  struct {
    unsigned w:32;
  };
```

```
} CNEN_t;
extern volatile CNEN_t _CNEN __asm__ ("CNEN") __attribute__((section("sfrs"), address(0xBF8861D0)));

typedef union {
  struct {
    unsigned _ALT_INT2:1;  //CNPUE0:1;
    unsigned _ALT_INT1:1;  //CNPUE1:1;
    unsigned :1;  //CNPUE2:1;
    unsigned :1;  //CNPUE3:1;
    unsigned _BLE_nCS1:1;  //CNPUE4:1;
    unsigned _BLE_nCS2:1;  //CNPUE5:1;
    unsigned _BLE_nCS3:1;  //CNPUE6:1;
    unsigned :1;  //CNPUE7:1;
    unsigned :1;  //CNPUE8:1;
    unsigned :1;  //CNPUE9:1;
    unsigned :1; //CNPUE10:1;
    unsigned :1; //CNPUE11:1;
    unsigned _GPS_3DFIX:1; //CNPUE12:1;
    unsigned _GAM_INT_M:1; //CNPUE13:1;
    unsigned _GAM_INT1_AG:1; //CNPUE14:1;
    unsigned _GAM_INT2_AG:1; //CNPUE15:1;
    unsigned _GAM_DRDY:1; //CNPUE16:1;
    unsigned :1; //CNPUE17:1;
    unsigned :1; //CNPUE18:1;
  };
  struct {
    unsigned w:32;
  };
} CNPUE_t;
extern volatile CNPUE_t _CNPUE __asm__ ("CNPUE") __attribute__((section("sfrs"), address(0xBF8861E0)));



#endif    /* CONFIG_PINS_H */
-----------------------------------------------------------------------------------------------

/*
 * File:   driveFunctions.h
 * Author: John
 *
 * Created on February 20, 2018, 11:12 AM
 */

void motorTest(void);
void motorAForward(void);
void motorABackward(void);
void motorBForward(void);
```

```
void motorBBackward(void);
void motorCFoward(void);
void motorCBackward(void);
void motorDForward(void);
void motorDBackward(void);
void driveForward(void);
void driveBackward(void);
void turnRight(void);
void turnLeft(void);
void driveSetup(void);

void coast(void);
void motorTest(void) {
   MEASURE_Timer_On = true;
   DRIVE_Timer_On = true;
   PWM_Timer_On = true;

}
void motorABackward(void){
   MOTOR_A1 = 0;
   MOTOR_A2 = 1;
}
void motorAForward(void) {
   MOTOR_A2 = 0;
   MOTOR_A1 = 1;
}
void motorBForward(void){
   MOTOR_B1 = 0;
   MOTOR_B2 = 1;
}
void motorBBackward(void) {
   MOTOR_B2 = 0;
   MOTOR_B1 = 1;
}
void motorCBackward(void) {
   MOTOR_C2 = 0;
   MOTOR_C1 = 1;
}
void motorCForward(void) {
   MOTOR_C1 = 0;
   MOTOR_C2 = 1;
}
void motorDBackward(void) {
   MOTOR_D1 = 0;
   MOTOR_D2 = 1;
}
void motorDForward(void) {
```

```
    MOTOR_D2 = 0;
    MOTOR_D1 = 1;
}
void coast(void) {
    MOTOR_A1 = 0;
    MOTOR_A2 = 0;
    MOTOR_B1 = 0;
    MOTOR_B2 = 0;
    MOTOR_C1 = 0;
    MOTOR_C2 = 0;
    MOTOR_D1 = 0;
    MOTOR_D2 = 0;
    SWITCH_MODE = 4;
}

void driveForward(void) {
    if (!SLOW_MOTOR) {
        motorAForward();
        motorBForward();
        motorCForward();
        motorDForward();
    }
    else {
        coast();
    }
    SWITCH_MODE = 0;
}
void driveBackward(void) {
    if (!SLOW_MOTOR) {
        motorABackward();
        motorBBackward();
        motorCBackward();
        motorDBackward();
    }
    else {
        coast();
    }
    SWITCH_MODE = 1;
}

void turnLeft(void) {
    motorABackward();
    motorBForward();
    motorCForward();
    motorDBackward();
    SWITCH_MODE = 2;
}
```

```
void turnRight(void) {
   motorAForward();
   motorBBackward();
   motorCBackward();
   motorDForward();
   SWITCH_MODE = 3;
}
void driveSetup(void) {
   int distance = 0xFFFF;
   distance = readLidar();
   if (distance < 100) {
      turnLeft();
   }
   else {
      driveForward();
   }

}

#define negative 1
#define positive 0

void initializeGAM(void);
void readGAM(void);

void initializeGAM(void) {
   startI2C();
   putI2C(GAM_WRITE & 0xFE);
   putI2C(0x1F);
   putI2C(0b00111000);
   stopI2C();
   startI2C();
   putI2C(GAM_WRITE & 0xFE);
   putI2C(0x20);
   putI2C(0b11000000);
}
void readGAM(void) {
   int x_high = 0x00;
   int x_low = 0x00;
   int y_high = 0x00;
   int y_low = 0x00;
   int z_high = 0x00;
   int z_low = 0x00;
   int check = 0x00;
   int xsign;
   int ysign;
```

```
int zsign;
float xacc_final;
float yacc_final;
float zacc_final;
while ((check & 0x01) == 0) {
   startI2C();
   putI2C(GAM_WRITE & 0xFE);
   putI2C(0x27);
   startI2C();
   putI2C(GAM_READ |0x01);
   check = getI2C(1);
   stopI2C();
}
startI2C();
putI2C(GAM_WRITE & 0xFE);
putI2C(0x28);
startI2C();
putI2C(GAM_READ | 0x01);
x_high = getI2C(0);
x_low = getI2C(0);
y_high = getI2C(0);
y_low = getI2C(0);
z_high = getI2C(0);
z_low = getI2C(1);
stopI2C();

int xacc =  x_high << 8 | x_low;
int yacc =  y_high << 8 | y_low;
int zacc =  z_high << 8 | z_low;
if ((xacc & 0x8000) == 0){
   xsign = positive;
}
if ((yacc & 0x8000) == 0){
   ysign = positive;
}
if ((zacc & 0x8000) == 0){
   zsign = positive;
}
xacc = xacc & 0x7FFF;
yacc = yacc & 0x7FFF;
zacc = zacc & 0x7FFF;
xacc_final = ((float) xacc*2)/((float) 0x7FFF);
yacc_final = ((float) yacc*2)/((float) 0x7FFF);
zacc_final = ((float) zacc*2)/((float) 0x7FFF);
if (xsign != positive) {
   xacc_final = -xacc_final;
}
```

```
    if (ysign != positive) {
        yacc_final = -yacc_final;
    }
    if (zsign != positive) {
        zacc_final = -zacc_final;
    }
    if (xacc_final < 0) {
    printf("the acceleration in the x direction is %f \n\r", xacc_final);
    printf("the acceleration in the y direction is %f \n\r", yacc_final);
    printf("the acceleration in the z direction is %f \n\r", zacc_final);
    }
    else if (yacc_final < 0) {
    printf("the acceleration in the x direction is %f \n\r", xacc_final);
    printf("the acceleration in the y direction is %f \n\r", yacc_final);
    printf("the acceleration in the z direction is %f \n\r", zacc_final);
    }
    else if (zacc_final < 0) {
    printf("the acceleration in the x direction is %f \n\r", xacc_final);
    printf("the acceleration in the y direction is %f \n\r", yacc_final);
    printf("the acceleration in the z direction is %f \n\r", zacc_final);
    }
}
-----------------------------------------------------------------------------------------
/*
 * File:   I2C_function.h
 * Author: John
 *
 * Created on February 4, 2018, 2:15 PM
 */
#define MST_INT IFS1bits.I2C4MIF
void configI2C();
int startI2C();
int putI2C(int data);
void stopI2C(void);
void rstartI2C(void);
unsigned char getI2C(int ack2send);

void configI2C() { // configure I2C communication
    //printf("Configuring I2C");
    I2C4BRG = 0x02F;
    I2C4CONbits.ON = 1;
}
int startI2C() {
    MST_INT = 0;
    I2C4CONbits.SEN = 1;
    while (!MST_INT); //wait for communication from I2
```

```
}
int putI2C(int data) {//send data to I2C
    MST_INT = 0;
    I2C4TRN = data;
    while(!MST_INT);
    return(I2C4STATbits.ACKSTAT);
}
void stopI2C(void) { //turn off I2C
    MST_INT = 0;
    I2C4CONbits.PEN = 1;
    while(!MST_INT); //wait for response
}
void rstartI2C(void) {
    MST_INT = 0;
    I2C4CONbits.RSEN = 1;
    while(!MST_INT);
}
unsigned char getI2C(int ack2send) { //get response from I2C
    MST_INT = 0;
    unsigned char inByte;
    I2C4CONbits.RCEN = 1;
    while(!MST_INT);
    MST_INT = 0;
    inByte = I2C4RCV;
    I2C4CONbits.ACKEN = 1;
    I2C4CONbits.ACKDT = ack2send;
    while(!MST_INT);
    return(inByte);
}
-----------------------------------------------------------------------------------------------
/*
 * File:   interrupt_config.h
 * Author: Chris St. George
 */

#ifndef INTERRUPT_CONFIG_H
#define  INTERRUPT_CONFIG_H

#include "config.h"

// Interrupt Flag, Priority, and Enable bits

#define Global_Interrupts_Enabled asm("ei")
#define Global_Interrupts_Disabled asm("di")
#define Multivector_Interrupts_Enable INTCONbits.MVEC

#define Timer_1_Vector _TIMER_1_VECTOR
```

```
#define Timer_1_Interrupt_Flag IFS0bits.T1IF
#define Timer_1_Interrupt_Priority IPC1bits.T1IP
#define Timer_1_Interrupt_Subpriority IPC1bits.T1IS
#define Timer_1_Interrupt_Enable IEC0bits.T1IE

#define Timer_2_Vector _TIMER_2_VECTOR
#define Timer_2_Interrupt_Flag IFS0bits.T2IF
#define Timer_2_Interrupt_Priority IPC2bits.T2IP
#define Timer_2_Interrupt_Subpriority IPC2bits.T2IS
#define Timer_2_Interrupt_Enable IEC0bits.T2IE

#define Timer_3_Vector _TIMER_3_VECTOR
#define Timer_3_Interrupt_Flag IFS0bits.T3IF
#define Timer_3_Interrupt_Priority IPC3bits.T3IP
#define Timer_3_Interrupt_Subpriority IPC3bits.T3IS
#define Timer_3_Interrupt_Enable IEC0bits.T3IE

#define Timer_4_Vector _TIMER_4_VECTOR
#define Timer_4_Interrupt_Flag IFS0bits.T4IF
#define Timer_4_Interrupt_Priority IPC4bits.T4IP
#define Timer_4_Interrupt_Subpriority IPC4bits.T4IS
#define Timer_4_Interrupt_Enable IEC0bits.T4IE

#define Timer_5_Vector _TIMER_5_VECTOR
#define Timer_5_Interrupt_Flag IFS0bits.T5IF
#define Timer_5_Interrupt_Priority IPC5bits.T5IP
#define Timer_5_Interrupt_Subpriority IPC5bits.T5IS
#define Timer_5_Interrupt_Enable IEC0bits.T5IE

#define PRAGMA(x) _Pragma(#x)
#define VECTOR(x) _##x##_VECTOR
#define _PRIORITY(x) ipl ## x ## auto
#define PRIORITY(x) _PRIORITY(x)

#define Interrupt_Settings(X) X##_Interrupt_Settings()
#define _Configure(X) Interrupt_Settings(X);
#define Configure(...)\
 void configure_interrupts() {\
  printf("\n\rvoid configure_interrupts()");\
  Global_Interrupts_Enabled;\
  Multivector_Interrupts_Enable = enabled;\
  FOR_EACH(_Configure,__VA_ARGS__)\
 }

#define Interrupt(object, full_name, event_type, priority, subpriority, is_enabled)\
 void Interrupt_Settings(object) {\
  /*interrupt object##_##event_type##_Event autoIPL(priority) vector VECTOR(object);*/\
```

```
  object##_Interrupt_Priority = priority; /* Sets the priority for the interrupt */\
  object##_Interrupt_Subpriority = subpriority; /* Sets the subpriority for the interrupt */\
  object##_Interrupt_Flag = false; /* Sets the initial flag state for the interrupt */\
  object##_Interrupt_Enable = is_enabled; /* Sets the enable bit for the interrupt */\
 }\
 __attribute__((vector(VECTOR(full_name)),interrupt(PRIORITY(priority)), nomips16))\
 void object##_##event_type##_Event()

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/

// LED Timer (Timer 2)

#define PWM_Timer_Interrupt_Flag Timer_2_Interrupt_Flag
#define PWM_Timer_Interrupt_Priority Timer_2_Interrupt_Priority
#define PWM_Timer_Interrupt_Subpriority Timer_2_Interrupt_Subpriority
#define PWM_Timer_Interrupt_Enable Timer_2_Interrupt_Enable
#define PWM_Timer_Vector _TIMER_2_VECTOR

#define MEASURE_Timer_Interrupt_Flag Timer_4_Interrupt_Flag
#define MEASURE_Timer_Interrupt_Priority Timer_4_Interrupt_Priority
#define MEASURE_Timer_Interrupt_Subpriority Timer_4_Interrupt_Subpriority
#define MEASURE_Timer_Interrupt_Enable Timer_4_Interrupt_Enable
#define MEASURE_Timer_Vector _TIMER_4_VECTOR

#define DRIVE_Timer_Interrupt_Flag Timer_3_Interrupt_Flag
#define DRIVE_Timer_Interrupt_Priority Timer_3_Interrupt_Priority
#define DRIVE_Timer_Interrupt_Subpriority Timer_3_Interrupt_Subpriority
#define DRIVE_Timer_Interrupt_Enable Timer_3_Interrupt_Enable
#define DRIVE_Timer_Vector _TIMER_3_VECTOR

#define DEPLOY_Timer_Interrupt_Flag Timer_5_Interrupt_Flag
#define DEPLOY_Timer_Interrupt_Priority Timer_5_Interrupt_Priority
#define DEPLOY_Timer_Interrupt_Subpriority Timer_5_Interrupt_Subpriority
#define DEPLOY_Timer_Interrupt_Enable Timer_5_Interrupt_Enable
#define DEPLOY_Timer_Vector _TIMER_5_VECTOR

Interrupt(PWM_Timer, TIMER_2, Overflow,5,1,enabled) {
// Test_Timer_Count = 0; // reset timer
   //MOTOR_B2 = !MOTOR_B2;
   PWM_Timer_Interrupt_Flag = false; // clear flag

}

Interrupt(MEASURE_Timer, TIMER_4,Overflow,6,1,enabled) {
   if (MEASURE_TIMER_CONDITION==1) {
      LORA_nRESET = high;
      MEASURE_Timer_On = false;
```

```
    }
    else if (MEASURE_TIMER_CONDITION == 2) {
          coast();
          MEASURE_Timer_On = false;
    }
    else if (MEASURE_TIMER_CONDITION == 3) {
       driveSetup();
    }
    else if (MEASURE_TIMER_CONDITION == 4) {
       OC2RS = 1872;
       MEASURE_Timer_On = false;
    }
    MEASURE_Timer_Interrupt_Flag = false;
}
Interrupt(DRIVE_Timer, TIMER_3,Overflow,6,1,enabled) {
      in_flight_mode = in_flight_mode + 1;
      if ((in_flight_mode%2) == 0){
         OC2RS = 1872;
      }
      else {
         OC2RS = 1807;
      }
      DRIVE_Timer_Interrupt_Flag = false;
}
Interrupt(DEPLOY_Timer, TIMER_5,Overflow,6,1,enabled) {
    SLOW_MOTOR = !SLOW_MOTOR;
    DEPLOY_Timer_Period = DRIVE_PERIOD - DEPLOY_Timer_Period;
    switch (SWITCH_MODE) {
       case DRIVE_FORWARD:
          driveForward();
          break;
       case DRIVE_BACKWARD:
          driveBackward();
          break;
       case TURN_LEFT:
          turnLeft();
          break;
       case TURN_RIGHT:
          turnRight();
          break;
       case COAST:
          coast();
          break;
    }
    DEPLOY_Timer_Interrupt_Flag = false;

}
```

```
//#include "interrupt_config_Timer.h"
//#include "interrupt_config_Timer_Test.h"
#include "interrupt_config_UART.h"
//#include "interrupt_config_UART_PC.h"
#include "interrupt_config_UART_LoRa.h"
//#include "interrupt_config_UART_MUX.h"
//#include "interrupt_config_Change_Notification.h"

Configure(PWM_Timer, MEASURE_Timer,DRIVE_Timer,DEPLOY_Timer, LoRa_UART)
#endif    /* INTERRUPT_CONFIG_H */
-----------------------------------------------------------------------------------------
/*
 * File:   interrupt_config_UART.h
 * Author: cstgeo
 *
 * Created on February 11, 2018, 2:22 PM
 */

#ifndef INTERRUPT_CONFIG_UART_H
#define  INTERRUPT_CONFIG_UART_H
#include "config.h"

/*
 * URXISEL<1:0>: Receive Interrupt Mode Selection bit
 * 11 = Reserved
 * 10 = Interrupt flag bit is asserted while receive buffer is 3/4 or more full (has 6 or more data characters)
 * 01 = Interrupt flag bit is asserted while receive buffer is 1/2 or more full (has 4 or more data characters)
 * 00 = Interrupt flag bit is asserted while receive buffer is not empty (has at least 1 data character)
 */
struct UART_Receive_Interrupt_Mode_tag {
   unsigned Buffer_Three_Quarters_Full:2;
   unsigned Buffer_Half_Full:2;
   unsigned Buffer_Not_Empty:2;
} UART_Receive_Interrupt_Mode = {0b10, 0b01, 0b00};

/*
 * UTXISEL<1:0>: TX Interrupt Mode Selection bits
 * 11 = Reserved, do not use
 * 10 = Interrupt is generated and asserted while the transmit buffer is empty
 * 01 = Interrupt is generated and asserted when all characters have been transmitted
 * 00 = Interrupt is generated and asserted while the transmit buffer contains at least one empty space
 */
struct UART_Transmit_Interrupt_Mode_tag {
   unsigned Buffer_Empty:2;
   unsigned Buffer_Just_Emptied:2;
   unsigned Buffer_Not_Full:2;
```

```
} UART_Transmit_Interrupt_Mode = {0b10, 0b01, 0b00};

// Interrupt Flag, Priority, and Enable bits
#define UART_1_Vector _UART_1_VECTOR
#define UART_1_Transmit_Interrupt_Flag IFS0bits.U1TXIF
#define UART_1_Transmit_Interrupt_Mode_Select U1STAbits.UTXISEL
#define UART_1_Receive_Interrupt_Flag IFS0bits.U1RXIF
#define UART_1_Receive_Interrupt_Mode_Select U1STAbits.URXISEL
#define UART_1_Error_Interrupt_Flag IFS0bits.U1EIF
#define UART_1_Parity_Error_Status U1STAbits.PERR
#define UART_1_Framing_Error_Status U1STAbits.FERR
#define UART_1_Receive_Buffer_Overflow_Status U1STAbits.OERR
#define UART_1_Interrupt_Priority IPC6bits.U1IP
#define UART_1_Interrupt_Subpriority IPC6bits.U1IS
#define UART_1_Transmit_Interrupt_Enable IEC0bits.U1TXIE
#define UART_1_Receive_Interrupt_Enable IEC0bits.U1RXIE
#define UART_1_Error_Interrupt_Enable IEC0bits.U1EIE

#define UART_2_Vector _UART_2_VECTOR
#define UART_2_Transmit_Interrupt_Flag IFS1bits.U2TXIF
#define UART_2_Transmit_Interrupt_Mode_Select U2STAbits.UTXISEL
#define UART_2_Receive_Interrupt_Flag IFS1bits.U2RXIF
#define UART_2_Receive_Interrupt_Mode_Select U2STAbits.URXISEL
#define UART_2_Error_Interrupt_Flag IFS1bits.U2EIF
#define UART_2_Parity_Error_Status U2STAbits.PERR
#define UART_2_Framing_Error_Status U2STAbits.FERR
#define UART_2_Receive_Buffer_Overflow_Status U2STAbits.OERR
#define UART_2_Interrupt_Priority IPC8bits.U2IP
#define UART_2_Interrupt_Subpriority IPC8bits.U2IS
#define UART_2_Transmit_Interrupt_Enable IEC1bits.U2TXIE
#define UART_2_Receive_Interrupt_Enable IEC1bits.U2RXIE
#define UART_2_Error_Interrupt_Enable IEC1bits.U2EIE

#define UART_6_Vector _UART_6_VECTOR
#define UART_6_Transmit_Interrupt_Flag IFS2bits.U6TXIF
#define UART_6_Transmit_Interrupt_Mode_Select U6STAbits.UTXISEL
#define UART_6_Receive_Interrupt_Flag IFS2bits.U6RXIF
#define UART_6_Receive_Interrupt_Mode_Select U6STAbits.URXISEL
#define UART_6_Error_Interrupt_Flag IFS2bits.U6EIF
#define UART_6_Parity_Error_Status U6STAbits.PERR
#define UART_6_Framing_Error_Status U6STAbits.FERR
#define UART_6_Receive_Buffer_Overflow_Status U6STAbits.OERR
#define UART_6_Interrupt_Priority IPC12bits.U6IP
#define UART_6_Interrupt_Subpriority IPC12bits.U6IS
#define UART_6_Transmit_Interrupt_Enable IEC2bits.U6TXIE
#define UART_6_Receive_Interrupt_Enable IEC2bits.U6RXIE
#define UART_6_Error_Interrupt_Enable IEC2bits.U6EIE
```

```
#define UART_Interrupt(object, full_name, event_type, priority, subpriority, tx_is_enabled, rx_is_enabled,
err_is_enabled, tx_interrupt_mode, rx_interrupt_mode)\
 void object##_Interrupt_Settings() {\
  /*interrupt object##_##event_type##_Event autoIPL(priority) vector VECTOR(object);*/\
  object##_Interrupt_Priority = priority; /* Sets the priority for the interrupt */\
  object##_Interrupt_Subpriority = subpriority; /* Sets the subpriority for the interrupt */\
  object##_Transmit_Interrupt_Flag = false; /* Sets the initial flag state for the interrupt */\
  object##_Transmit_Interrupt_Mode_Select = tx_interrupt_mode; /* Sets the rx interrupt mode */\
  object##_Receive_Interrupt_Flag = false; /* Sets the initial flag state for the interrupt */\
  object##_Receive_Interrupt_Mode_Select = rx_interrupt_mode; /* Sets the rx interrupt mode */\
  object##_Error_Interrupt_Flag = false; /* Sets the initial flag state for the interrupt */\
  object##_Transmit_Interrupt_Enable = tx_is_enabled; /* Sets the enable bit for the interrupt */\
  object##_Receive_Interrupt_Enable = rx_is_enabled; /* Sets the enable bit for the interrupt */\
  object##_Error_Interrupt_Enable = err_is_enabled; /* Sets the enable bit for the interrupt */\
 }\
 __attribute__((vector(VECTOR(full_name)),interrupt(PRIORITY(priority)), nomips16))\
 void object##_##event_type##_Event()

#endif    /* INTERRUPT_CONFIG_UART_H */

-----------------------------------------------------------------------------------------
/*
 * File:   interrupt_config_UART_LoRa.h
 * Author: cstgeo
 *
 * Created on February 11, 2018, 2:35 PM
 */

#ifndef INTERRUPT_CONFIG_UART_LORA_H
#define  INTERRUPT_CONFIG_UART_LORA_H
#include "config.h"

// LoRa UART (UART 2)
#define LoRa_UART_Transmit_Interrupt_Flag UART_2_Transmit_Interrupt_Flag
#define LoRa_UART_Transmit_Interrupt_Mode_Select UART_2_Transmit_Interrupt_Mode_Select
#define LoRa_UART_Receive_Interrupt_Flag UART_2_Receive_Interrupt_Flag
#define LoRa_UART_Receive_Interrupt_Mode_Select UART_2_Receive_Interrupt_Mode_Select
#define LoRa_UART_Error_Interrupt_Flag UART_2_Error_Interrupt_Flag
#define LoRa_UART_Parity_Error_Status UART_2_Parity_Error_Status
#define LoRa_UART_Framing_Error_Status UART_2_Framing_Error_Status
#define LoRa_UART_Receive_Buffer_Overflow_Status UART_2_Receive_Buffer_Overflow_Status
#define LoRa_UART_Interrupt_Priority UART_2_Interrupt_Priority
#define LoRa_UART_Interrupt_Subpriority UART_2_Interrupt_Subpriority
#define LoRa_UART_Transmit_Interrupt_Enable UART_2_Transmit_Interrupt_Enable
#define LoRa_UART_Receive_Interrupt_Enable UART_2_Receive_Interrupt_Enable
#define LoRa_UART_Error_Interrupt_Enable UART_2_Error_Interrupt_Enable
```

```
UART_Interrupt(LoRa_UART,UART_2,Interrupt,6,1,
       /*tx interrupts*/disabled,
       /*rx interrupts*/enabled,
       /*err interrupts*/disabled,
       UART_Transmit_Interrupt_Mode.Buffer_Just_Emptied,
       UART_Receive_Interrupt_Mode.Buffer_Not_Empty) {
// pc_uart_just_printed_debug_info = true;
// printf("\n\rvoid LoRa_UART_Interrupt_Event()");
 if(LoRa_UART_Transmit_Interrupt_Enable && LoRa_UART_Transmit_Interrupt_Flag) // Transmitter-buffer-
empty interrupt
 {
  printf("\n\r\tUART Transmit Interrupt");
  LoRa_UART_Transmit_Interrupt_Flag = false; // clear flag
 }
 if(LoRa_UART_Receive_Interrupt_Enable && LoRa_UART_Receive_Interrupt_Flag) // Receiver-data-available
interrupt
 {
  char c = getU2();
  lora_uart_state = (c == '\n') ? LORA_UART_STRING_RECEIVED : LORA_UART_RECEIVING_STRING;
  if (c != '\0') {
   received_string[received_string_index++] = c;
   if (lora_uart_state == LORA_UART_STRING_RECEIVED) {
    received_string[received_string_index] = '\0';
    received_string_index = 0;
   }
  }
  LoRa_UART_Receive_Interrupt_Flag = false; // clear flag
 }
 if(LoRa_UART_Error_Interrupt_Enable && LoRa_UART_Error_Interrupt_Flag) // UART-error interrupt
 {
  printf("\n\r\tUART-error interrupt");
  if (LoRa_UART_Parity_Error_Status) // Parity Error
  {
   printf("\n\r\t\tParity Error");
  }
  if (LoRa_UART_Framing_Error_Status) // Framing Error
  {
   printf("\n\r\t\tFraming Error");
  }
  if (LoRa_UART_Receive_Buffer_Overflow_Status) // RX Buffer Overflow
  {
   printf("\n\r\t\tRX Buffer Overflow");
  }
  LoRa_UART_Error_Interrupt_Flag = false; // clear flag
```

```
 }
}

#endif    /* INTERRUPT_CONFIG_UART_LORA_H */
------------------------------------------------------------------------------------------------------------------------------------------------------------
/*
 * File:   lidarHeader.h
 * Author: John
 *
 * Created on February 4, 2018, 2:12 PM
 */

#define numCounts 500
void initializeLidar(void);
int readLidar(void);
int readLidar(void){
   int disCount = 0;
   unsigned char byte;
   unsigned char check = 0xFF;
   unsigned char disHigh;
   unsigned char disLow;
   initializeLidar();
   while (check != 0x00){
      startI2C();
      putI2C(lidar_write & 0xFE);
      putI2C(0x01);
      stopI2C();
      startI2C();
      putI2C(lidar_read | 0x01);
      byte = getI2C(1);
      check = byte & 0b00000001;
      stopI2C();
   }
   startI2C();
   putI2C(lidar_write & 0xFE);
   putI2C(0x8f);
   stopI2C();
   startI2C();
   putI2C(lidar_read | 0x01);
   disHigh = getI2C(0);
   disLow = getI2C(1);
   int distance = disHigh << 8 | disLow;
   printf("Current distance from target is %d cm \n\r",distance);
   stopI2C();
   return distance;
}
```

```
void initializeLidar(void) {//initialize LIDAR and get first reading
   startI2C();
   putI2C(lidar_write & 0xFE);
   putI2C(0x00); //write device command
   putI2C(0x04); //take distance measurement without receiver bias correction
   stopI2C();
}

-----------------------------------------------------------------------------------------------
/*
 * File:   lora_config.h
 * Author: cstgeo
 *
 * Created on March 1, 2018, 11:10 PM
 */

#ifndef LORA_CONFIG_H
#define  LORA_CONFIG_H
#include "config.h"


typedef enum { INITIALIZING,
            CONFIGURING,
            TRANSCEIVER_ENABLED,
            INITIALIZATION_ERROR,
            CONFIGURATION_ERROR
} LORA_STATE;
LORA_STATE lora_state = INITIALIZING;

typedef enum { LORA_UART_RECEIVING_STRING,/* -> */LORA_UART_STRING_RECEIVED,/* ->
*/LORA_UART_STRING_READ,
} LORA_UART_STATE;
LORA_UART_STATE lora_uart_state = LORA_UART_RECEIVING_STRING;

typedef enum { NOT_CONFIGURED,
            PAUSING_LORAWAN,
            SETTING_RADIO_POWER,
            SETTING_WATCHDOG_TIMER,
            CONFIGURATION_COMPLETE,
            PAUSE_LORAWAN_ERROR,
            SET_RADIO_POWER_ERROR,
            SET_WATCHDOG_TIMER_ERROR
} LORA_CONFIGURATION_STATE;
LORA_CONFIGURATION_STATE lora_configuration_state = NOT_CONFIGURED;

typedef enum { NOT_ENABLED, IDLE,
            TRANSMISSION_STARTING,/* -> */TRANSMITTING,
```

```
        RECEPTION_STARTING,/* -> */RECEIVING
} LORA_TRANSCEIVER_STATE;
LORA_TRANSCEIVER_STATE lora_transceiver_state = NOT_ENABLED;


typedef enum { CONTINUOUS_RECEPTION,
        CONTINUOUS_TRANSMISSION,
        TRANSMIT_DATA_ONCE_EVERY_N_TIMER_CYCLES_ELSE_RECEIVE
} LORA_TRANSCEIVER_MODE;


#define received_string_length 250
#define received_data_offset 10
unsigned char received_string[received_string_length];
unsigned int received_string_index = 0;
#define lora_initialized_condition received_string[0] == 'R'
#define lora_mac_paused_condition received_string[0] == '4'
#define lora_radio_power_set_condition received_string[0] == 'o'
#define lora_watchdog_timer_set_condition received_string[0] == 'o'
#define lora_rx_receiving_condition received_string[0] == 'o'
#define lora_tx_transmitting_condition received_string[0] == 'o'
#define lora_rx_received_condition received_string[0] == 'r'
#define lora_tx_transmitted_condition received_string[0] == 'r'

#define mac_pause lora_configuration_state = PAUSING_LORAWAN;\
            printf("[PIC32 -> LoRa]:\t" "mac pause\r\n"); __XC_UART = 2;\
            printf("mac pause\r\n"); __XC_UART = 1
#define radio_set_pwr(n) lora_configuration_state = SETTING_RADIO_POWER;\
                printf("[PIC32 -> LoRa]:\t" "radio set pwr " #n "\r\n"); __XC_UART = 2;\
                printf("radio set pwr " #n "\r\n"); __XC_UART = 1
#define radio_set_wdt(n) lora_configuration_state = SETTING_WATCHDOG_TIMER;\
                printf("[PIC32 -> LoRa]:\t" "radio set wdt " #n "\r\n"); __XC_UART = 2;\
                printf("radio set wdt " #n "\r\n"); __XC_UART = 1
#define radio_tx(n) lora_transceiver_state = TRANSMISSION_STARTING;\
              printf("[PIC32 -> LoRa]:\t" "radio tx " #n "\r\n"); __XC_UART = 2;\
              printf("radio tx " #n "\r\n"); __XC_UART = 1
#define radio_rx(n) lora_transceiver_state = RECEPTION_STARTING;\
              printf("[PIC32 -> LoRa]:\t" "radio rx " #n "\r\n"); __XC_UART = 2;\
              printf("radio rx " #n "\r\n"); __XC_UART = 1

void configure_lora();
void handle_lora_output();
void function_that_does_something_with_data(unsigned char data[]);

void LoRa() {
 if (lora_transceiver_state == IDLE && lora_uart_state == LORA_UART_STRING_READ) {
   radio_rx(0);
```

```
  } else if (lora_uart_state == LORA_UART_STRING_RECEIVED) {
   lora_uart_state = LORA_UART_STRING_READ;
   printf("[LoRa -> PIC32]:\t%s", received_string);
   switch (lora_transceiver_state) {
    case NOT_ENABLED: switch (lora_configuration_state) {
                    case NOT_CONFIGURED:        if (lora_initialized_condition       ) { mac_pause;         } break;
                    case PAUSING_LORAWAN:       if (lora_mac_paused_condition        ) { radio_set_pwr(20); }
break;
                    case SETTING_RADIO_POWER:    if (lora_radio_power_set_condition   ) { radio_set_wdt(0);  }
break;
                    case SETTING_WATCHDOG_TIMER: if (lora_watchdog_timer_set_condition) {
                     lora_configuration_state = CONFIGURATION_COMPLETE;
                     printf("\t...LoRa configuration complete...\n\r");
                     lora_transceiver_state = IDLE;
                    } break;
                   }
                            break;
    case TRANSMISSION_STARTING: if (lora_tx_transmitting_condition) { lora_transceiver_state =
TRANSMITTING; } break;
    case RECEPTION_STARTING: if (lora_rx_receiving_condition) {
                                        lora_transceiver_state = RECEIVING;
                                        printf("\t...LoRa module waiting to receive...\n\r");
                                       } break;
    case TRANSMITTING: if (lora_tx_transmitted_condition) {
                                    lora_transceiver_state = IDLE;
                                } break;
    case RECEIVING: if (lora_rx_received_condition) {
                                lora_transceiver_state = IDLE;
                                function_that_does_something_with_data(received_string + received_data_offset);
                               } break;
   }
  }
}

#endif    /* LORA_CONFIG_H */
-----------------------------------------------------------------------------------------------
/*
 * File:   LORATest.h
 * Author: John
 *
 * Created on February 20, 2018, 11:45 PM
 */
void uartTest(void);
void LORATest(void);
void uartTest(void){
    unsigned char userChar;
    userChar = getU1();
```

```
    putU1(userChar);
    printf("\n\r");
}
#define maxIndex 20
void wait(unsigned char message[]){
    do {
        int index=0;
        unsigned char userChar = 'a';
        while ((userChar != '\n') && index < maxIndex){
            userChar = getU2();
            message[index++]=userChar;
        }
        message[index++] = '\0';
    }  while (strcmp(message,"busy\n\r"));
    __XC_UART = 1;
    printf(message);
    __XC_UART = 2;
}
void LORATest(void){
    unsigned char message[maxIndex];
    printf("mac pause\r\n");
    wait(message);
    printf("radio set pwr 14\r\n");
    wait(message);
    printf("radio tx 0123456789ABCDEF\r\n");
    wait(message);
}
```

```
/*
 * File:   setup_devices_file.h
 * Author: John
 *
 * Created on February 4, 2018, 2:19 PM
 */

void setup_devices()
{
// MUXbits
 DDPCONbits.JTAGEN = 0; // turn off JTAG Controller Pins
 AD1PCFG = 0xFFFF; // set ANx Pins to Digital mode
 LATB = 0x0000;
 LATE = 0x0000;
 LATF = 0x0000;
 /* TRISB_MASK */ TRISB = 0x8000; // sets RB(2-4,15) to input mode
 /* TRISC_MASK */ TRISC = 0x6000; // sets RC(13-14) to input mode (Altimeter Interrupt Pins)
 /* TRISD_MASK */ TRISD = 0x00F0; // sets RD(4-7) to input mode
 /* TRISE_MASK */ TRISE = 0x0000; // sets all RE to output mode
```

```
 /* TRISF_MASK */ TRISF = 0x0000; // sets all RF to output mode
 /* TRISG_MASK */ TRISG = 0x0000; // sets all RG to output mode
              LORA_nRESET = low;
 LATB = 0x0000;
 LATE = 0x0000;
 LATF = 0x0000;
}
-------------------------------------------------------------------------------------------
/*
 * File:   timer_config.h
 * Author: Chris St. George
 */

#ifndef TIMER_CONFIG_H
#define  TIMER_CONFIG_H

#include "config.h"

// Timer 1
#define Timer_1_On T1CONbits.TON
#define Timer_1_Gated T1CONbits.TGATE
#define Timer_1_Source_Select T1CONbits.TCS
#define Timer_1_Prescale_Select T1CONbits.TCKPS
#define Timer_1_Prescale_1 Timer_1_Prescale_Select = 0b00
#define Timer_1_Prescale_8 Timer_1_Prescale_Select = 0b01
#define Timer_1_Prescale_64 Timer_1_Prescale_Select = 0b10
#define Timer_1_Prescale_256 Timer_1_Prescale_Select = 0b11
#define Timer_1_Count TMR1
#define Timer_1_Period PR1

// Timer 2
#define Timer_2_On T2CONbits.TON
#define Timer_2_Gated T2CONbits.TGATE
#define Timer_2_Source_Select T1CONbits.TCS
#define Timer_2_Prescale_Select T2CONbits.TCKPS
#define Timer_2_Prescale_1 Timer_2_Prescale_Select = 0b000
#define Timer_2_Prescale_2 Timer_2_Prescale_Select = 0b001
#define Timer_2_Prescale_4 Timer_2_Prescale_Select = 0b010
#define Timer_2_Prescale_8 Timer_2_Prescale_Select = 0b011
#define Timer_2_Prescale_16 Timer_2_Prescale_Select = 0b100
#define Timer_2_Prescale_32 Timer_2_Prescale_Select = 0b101
#define Timer_2_Prescale_64 Timer_2_Prescale_Select = 0b110
#define Timer_2_Prescale_256 Timer_2_Prescale_Select = 0b111
#define Timer_2_Count TMR2
#define Timer_2_Period PR2

// Timer 3
```

```
#define Timer_3_On T3CONbits.TON
#define Timer_3_Gated T3CONbits.TGATE
#define Timer_3_Source_Select T1CONbits.TCS
#define Timer_3_Prescale_Select T3CONbits.TCKPS
#define Timer_3_Prescale_1 Timer_3_Prescale_Select = 0b000
#define Timer_3_Prescale_2 Timer_3_Prescale_Select = 0b001
#define Timer_3_Prescale_4 Timer_3_Prescale_Select = 0b010
#define Timer_3_Prescale_8 Timer_3_Prescale_Select = 0b011
#define Timer_3_Prescale_16 Timer_3_Prescale_Select = 0b100
#define Timer_3_Prescale_32 Timer_3_Prescale_Select = 0b101
#define Timer_3_Prescale_64 Timer_3_Prescale_Select = 0b110
#define Timer_3_Prescale_256 Timer_3_Prescale_Select = 0b111
#define Timer_3_Count TMR3
#define Timer_3_Period PR3

// Timer 4
#define Timer_4_On T4CONbits.TON
#define Timer_4_Gated T4CONbits.TGATE
#define Timer_4_Source_Select T1CONbits.TCS
#define Timer_4_Prescale_Select T4CONbits.TCKPS
#define Timer_4_Prescale_1 Timer_4_Prescale_Select = 0b000
#define Timer_4_Prescale_2 Timer_4_Prescale_Select = 0b001
#define Timer_4_Prescale_4 Timer_4_Prescale_Select = 0b010
#define Timer_4_Prescale_8 Timer_4_Prescale_Select = 0b011
#define Timer_4_Prescale_16 Timer_4_Prescale_Select = 0b100
#define Timer_4_Prescale_32 Timer_4_Prescale_Select = 0b101
#define Timer_4_Prescale_64 Timer_4_Prescale_Select = 0b110
#define Timer_4_Prescale_256 Timer_4_Prescale_Select = 0b111
#define Timer_4_Count TMR4
#define Timer_4_Period PR4

// Timer 5
#define Timer_5_On T5CONbits.TON
#define Timer_5_Gated T5CONbits.TGATE
#define Timer_5_Source_Select T1CONbits.TCS
#define Timer_5_Prescale_Select T5CONbits.TCKPS
#define Timer_5_Prescale_1 Timer_5_Prescale_Select = 0b000
#define Timer_5_Prescale_2 Timer_5_Prescale_Select = 0b001
#define Timer_5_Prescale_4 Timer_5_Prescale_Select = 0b010
#define Timer_5_Prescale_8 Timer_5_Prescale_Select = 0b011
#define Timer_5_Prescale_16 Timer_5_Prescale_Select = 0b100
#define Timer_5_Prescale_32 Timer_5_Prescale_Select = 0b101
#define Timer_5_Prescale_64 Timer_5_Prescale_Select = 0b110
#define Timer_5_Prescale_256 Timer_5_Prescale_Select = 0b111
#define Timer_5_Count TMR5
#define Timer_5_Period PR5
```

```
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/

// LED Timer (Timer 2)
#define PWM_Timer_On Timer_2_On
#define PWM_Timer_Gated Timer_2_Gated
#define PWM_Timer_Source_Select Timer_2_Source_Select
#define PWM_Timer_Prescale_Select Timer_2_Prescale_Select
#define PWM_Timer_Prescale_1 Timer_2_Prescale_1
#define PWM_Timer_Prescale_4 Timer_2_Prescale_2
#define PWM_Timer_Prescale_8 Timer_2_Prescale_8
#define PWM_Timer_Prescale_16 Timer_2_Prescale_16
#define PWM_Timer_Prescale_64 Timer_2_Prescale_64
#define PWM_Timer_Prescale_256 Timer_2_Prescale_256
#define PWM_Timer_Count Timer_2_Count
#define PWM_Timer_Period Timer_2_Period

#define MEASURE_Timer_On Timer_4_On
#define MEASURE_Timer_Gated Timer_4_Gated
#define MEASURE_Timer_Source_Select Timer_4_Source_Select
#define MEASURE_Timer_Prescale_Select Timer_4_Prescale_Select
#define MEASURE_Timer_Prescale_1 Timer_4_Prescale_1
#define MEASURE_Timer_Prescale_8 Timer_4_Prescale_8
#define MEASURE_Timer_Prescale_64 Timer_4_Prescale_64
#define MEASURE_Timer_Prescale_256 Timer_4_Prescale_256
#define MEASURE_Timer_Count Timer_4_Count
#define MEASURE_Timer_Period Timer_4_Period

#define DRIVE_Timer_On Timer_3_On
#define DRIVE_Timer_Gated Timer_3_Gated
#define DRIVE_Timer_Source_Select Timer_3_Source_Select
#define DRIVE_Timer_Prescale_Select Timer_3_Prescale_Select
#define DRIVE_Timer_Prescale_1 Timer_3_Prescale_1
#define DRIVE_Timer_Prescale_8 Timer_3_Prescale_8
#define DRIVE_Timer_Prescale_64 Timer_3_Prescale_64
#define DRIVE_Timer_Prescale_256 Timer_3_Prescale_256
#define DRIVE_Timer_Count Timer_3_Count
#define DRIVE_Timer_Period Timer_3_Period

#define DEPLOY_Timer_On Timer_5_On
#define DEPLOY_Timer_Gated Timer_5_Gated
#define DEPLOY_Timer_Source_Select Timer_5_Source_Select
#define DEPLOY_Timer_Prescale_Select Timer_5_Prescale_Select
#define DEPLOY_Timer_Prescale_1 Timer_5_Prescale_1
#define DEPLOY_Timer_Prescale_8 Timer_5_Prescale_8
#define DEPLOY_Timer_Prescale_64 Timer_5_Prescale_64
#define DEPLOY_Timer_Prescale_256 Timer_5_Prescale_256
#define DEPLOY_Timer_Count Timer_5_Count
```

```
#define DEPLOY_Timer_Period Timer_5_Period

#define PWM_Period 24975
#define MEASURE_Period 40000
#define DRIVE_Period 3000
#define DEPLOY_Period 10000

void configure_PWM_timer() {
   printf("\r\nconfigure_timer_2()");
   PWM_Timer_Prescale_8;
   PWM_Timer_Count = 0;
   PWM_Timer_Period = PWM_Period;
   PWM_Timer_Source_Select = 1;
   PWM_Timer_On = false;
}
void configure_measure_timer() {
   printf("\r\nconfigure_timer_4()");
   MEASURE_Timer_Prescale_256;
   MEASURE_Timer_Count = 0;
   MEASURE_Timer_Period = MEASURE_Period;
   MEASURE_Timer_Source_Select = 1;
   MEASURE_Timer_On = false;
}
void configure_drive_timer() {
   printf("\r\nconfigure_timer_3()");
   DRIVE_Timer_Prescale_1;
   DRIVE_Timer_Count = 0;
   DRIVE_Timer_Period = DRIVE_Period;
   DRIVE_Timer_Source_Select = 1;
   DRIVE_Timer_On = false;
}
void configure_deploy_timer() {
   printf("\r\nconfigure_timer_5()");
   DEPLOY_Timer_Prescale_256;
   DEPLOY_Timer_Count = 0;
   DEPLOY_Timer_Period = DEPLOY_Period;
   DEPLOY_Timer_Source_Select = 1;
   DEPLOY_Timer_On = false;
}
void configure_timers() {
   configure_PWM_timer();
   configure_measure_timer();
   configure_drive_timer();
   configure_deploy_timer();
}
void timer_test(void) {
   MEASURE_Timer_On = true;
```

```
   DRIVE_Timer_On = true;
   DEPLOY_Timer_On = true;
   PWM_Timer_On = true;
}
#endif    /* TIMER_CONFIG_H */
-----------------------------------------------------------------------------------------
/* File:   uart_config.h
 * Author: cstgeorg
 */

#ifndef UART_CONFIG_H
#define  UART_CONFIG_H
#include "config.h"



// UART Definitions
#define baudPC 57600
#define baudGPS 9600
#define baudLoRa 57600
#define baudBLE 115200
#define BR1 baudPC
#define BR2 baudLoRa
#define BR6 baudBLE
#define BRG1 FPB/4/BR1 - 1
#define BRG2 FPB/4/BR2 - 1
#define BRG6 FPB/4/BR6 - 1
#define GPS_SetBaudRateCommand "$PMTK251,115200*27\r\n"

//#define printfToUART(x) __XC_UART = (x == 2 ?  2 : 1) /* directs output of printf function through UARTx */
//
//#define printfToPC printfToUART(1)
//#define print(String,...) if(__XC_UART != 1) {unsigned char saved_uart_state = __XC_UART; printfToPC;
printf(String,##__VA_ARGS__); __XC_UART = saved_uart_state;}\
//                      else {printf(String,##__VA_ARGS__);}
//#define printfToLORA print("Printing to LoRa\r\n"); printfToUART(2)
//#define printfToMUX(x) print("Printing to MUX(%i)\r\n",x); muxToInput(x); printfToUART(6)
//#define printfToBLE1 print("Printing to BLE1\r\n"); muxToBLE1; printfToUART(6)
//#define printfToBLE2 print("Printing to BLE2\r\n"); muxToBLE2; printfToUART(6)
//#define printfToBLE3 print("Printing to BLE3\r\n"); muxToBLE3; printfToUART(6)
//#define printfToGPS print("Printing to GPS\r\n"); muxToGPS; printfToUART(6)

#define printUARTsettings(x) printf("\n\r" "UART " #x " Setup:"\
                            "\n\r\t" "BRG:\t%i"\
                            "\n\r\t" "BR:\t%i bps"\
                            "\n\r\t" "SCLK:\t%i MHz"\
                            "\n\r\t" "PBF:\t%i MHz", U##x##BRG, BR##x, SCLK/1000000, FPB/1000000)
```

```
//void _mon_putc (char c)
//{
// switch (__XC_UART)
// {
//  case 1:
//   while (U1STAbits.UTXBF); // Wait till transmission is complete
//   U1TXREG = c;
//   break;
//  case 2:
//   while (U2STAbits.UTXBF); // Wait till transmission is complete
//   U2TXREG = c;
//   break;
//  case 3:
//   while (U3STAbits.UTXBF); // Wait till transmission is complete
//   U3TXREG = c;
//   break;
//  case 4:
//   while (U4STAbits.UTXBF); // Wait till transmission is complete
//   U4TXREG = c;
//   break;
//  case 5:
//   while (U5STAbits.UTXBF); // Wait till transmission is complete
//   U5TXREG = c;
//   break;
//  case 6:
//   while (U6STAbits.UTXBF); // Wait until transmission is complete
//   U6TXREG = c;
//   break;
//  default: exit(1);
// }
//}

int u1rx_count = 0;
unsigned char getU1()
{
 while(!U1STAbits.URXDA);
// u1rx_count++;
 return U1RXREG;
}
void putU1(unsigned char c)
{
 while(U1STAbits.UTXBF);
 U1TXREG = c;
}

int u2rx_count = 0;
```

```
unsigned char getU2()
{
 while(!U2STAbits.URXDA){
 }
// u2rx_count++;
 return U2RXREG;
}
void putU2(unsigned char c)
{

 while(U2STAbits.UTXBF);
 U2TXREG = c;
}

int u6rx_count = 0;
unsigned char getU6()
{
 while(!U6STAbits.URXDA);
// u6rx_count++;
 return U6RXREG;
}
void putU6(unsigned char c)
{
 while(U6STAbits.UTXBF);
 U6TXREG = c;
}

void configure_uart() {
  __XC_UART = 1;
 /* Setup UART 1 */
                        // Enable UART
                        U1MODEbits.UARTEN = true;
                        U1STAbits.URXEN = true;
                        U1MODEbits.BRGH = true;
                        U1BRG = BRG1;

 /* Setup UART 2 */
                        // Enable UART
                        U2MODEbits.UARTEN = true;
                        U2STAbits.URXEN = true;
                        U2MODEbits.BRGH = true;
                        U2BRG = BRG2;

 /* Setup UART 6 */
                        // Enable UART
                        U6MODEbits.UARTEN = true;
                        U6STAbits.URXEN = true;
```

```
                U6MODEbits.BRGH = true;
                U6BRG = BRG6;
}

#endif    /* UART_CONFIG_H */
```

SHELL SCRIPTS:

```
#!/bin/bash

# start_LoRa_A: function to start the LoRa_A screen
function start_LoRa_A() {
screen -R -S LoRa_A /dev/cu.usbserial-F* $*
}
alias sa="start_LoRa_A"
alias SA="sa"


# start_LoRa_B: function to start the LoRa_B screen
function start_LoRa_B() {
screen -R -S LoRa_B /dev/cu.usbserial-A* $*
}
alias sb="start_LoRa_B"
alias SB="sb"


# LoRa_A: function to communicate with LoRa_A screen
function LoRa_A() {
if [[ -z $2 ]]; then
        echo "\"`printf "AT+$*"`\""
        screen -S LoRa_A -X stuff "`printf "AT+$*"`"
else
        echo "\"`printf "$*\r\n\r"`\""
        screen -S LoRa_A -X stuff "`printf "$*\r\n\r"`"
fi
}
alias a="LoRa_A"
alias A="a"


# LoRa_B: function to communicate with LoRa_B screen
function LoRa_B() {
if [[ -z $2 ]]; then
        echo "\"`printf "AT+$*"`\""
        screen -S LoRa_B -X stuff "`printf "AT+$*"`"
else
        echo "\"`printf "$*\r\n\r"`\""
        screen -S LoRa_B -X stuff "`printf "$*\r\n\r"`"
fi
}
alias b="LoRa_B"
```

```
alias B="b"


# sys: function to communicate with LoRa screens
function sys() {
echo "\"`printf "sys $*\r\n\r"`\""
screen -S LoRa_A -X stuff "`printf "sys $*\r\n\r"`"
screen -S LoRa_B -X stuff "`printf "sys $*\r\n\r"`"
}
alias SYS="sys"


# mac: function to communicate with LoRa screens
function mac() {
echo "\"`printf "mac $*\r\n\r"`\""
screen -S LoRa_A -X stuff "`printf "mac $*\r\n\r"`"
screen -S LoRa_B -X stuff "`printf "mac $*\r\n\r"`"
}
alias MAC="mac"


# radio: function to communicate with LoRa screens
function radio() {
echo "\"`printf "radio $*\r\n\r"`\""
screen -S LoRa_A -X stuff "`printf "radio $*\r\n\r"`"
screen -S LoRa_B -X stuff "`printf "radio $*\r\n\r"`"
}
alias RADIO="radio"

function scom() {
mac pause
radio set pwr 20
radio set wdt 0
}
alias SCOM="scom"


#ab: function to send value from LoRa a to LoRa b
function ab() {
b radio rx 0
if [[ -z $1 ]]; then
        a radio tx 0123456789ABCDEF
else
        a radio tx $*
fi
}
alias AB="ab"


#ba: function to send value from LoRa b to LoRa a
function ba() {
a radio rx 0
if [[ -z $1 ]]; then
```

```
        b radio tx 0123456789ABCDEF
else
        b radio tx $*
fi
}
alias BA="ba"


#at: function to communicate with the JDY-08 BLE screens
function at() {
echo "\"`printf "AT+$*"`\""
screen -S JDY_A -X stuff "`printf "AT+$*"`"
screen -S JDY_B -X stuff "`printf "AT+$*"`"
}
alias AT="at"


#start_host: function to communicate with the JDY-08 BLE screens. Sets attached modules as hosts and scans.
function start_host() {
AT HOSTEN1; sleep 0.25; AT RST; sleep 0.25; AT SCAN1
}
alias START_HOST="start_host"

#at: function to communicate with the JDY-08 BLE screens
function scan() {
AT SCAN1
}
alias SCAN="scan"
```