M6: Design Implementation

RTLS Team Purple

Prepared by: Aaron Guenther (Aero), Alex Koromilas (Aero), Nathaniel Root (Aero), Luigi Wellner (Mech), Adam Zarak (Mech)

Advisor: Michael Busque

Submitted on 04/21/2025

Executive Summary

The RTLS (Return-to-Launch-Site) model rocket project aims to design and develop a high-performance Level 1 model rocket capable of autonomous navigation and precise return to a predetermined launch site. This project addresses challenges in current model rocketry, such as limited recovery accuracy and reliance on traditional recovery methods, by integrating modern technologies like GPS guidance, altimetry, and real-time telemetry systems.

The final design incorporates a 4-inch diameter 3D printed PETG body tubes and advanced electronics including the Esp-32 microcontroller, NEO-6 GPS module, Berry IMUv3 inertial measurement unit, and RYLR896 LoRa module for communication. Additional components such as the Spektrum A6390 servo, TF Card Reader with a PNY 32GB card, URGENEX battery, Tip 31C transistor with diode, and a 3-12V to 3.3V and 5V DC-DC module support efficient operation and system reliability.

The project's outcomes promise significant benefits, including enhanced recovery reliability, reduced environmental impact, and the ability to conduct more complex and reusable missions. Through risk assessments, detailed engineering analysis, and advanced flight simulations, the team successfully navigated technical feasibility and resource limitations, facilitating skill development and practical engineering experience.

Currently, the project is finalizing its integration and testing phases, preparing for comprehensive flight tests designed to validate and refine system performance. This report provides a complete overview of the project's final status, methodologies utilized, and critical accomplishments, as well as recommendations and insights for future iterations of the RTLS model rocket system.

Table of Contents

Table of Figures

List of Tables

List of Equations

1. Introduction

1.1. Project Overview

The RTLS (Return-to-Launch-Site) model rocket project aims to design and develop a high-performance Level 1 model rocket capable of autonomous navigation and precise return to a predetermined launch site. This project addresses challenges in current model rocketry, such as limited recovery accuracy and reliance on traditional recovery methods, by integrating modern technologies like GPS guidance, altimetry, and real-time telemetry systems.

The project's outcomes promise significant benefits, including enhanced recovery reliability, reduced environmental impact, and the ability to conduct more complex and reusable missions. By leveraging risk assessments, intricate engineering analysis, and system testing, the team aims to address challenges related to the design of RTLS systems and produce a functional, reliable product.

This project has progressed beyond concept development into the critical design phase. This Milestone 6 report will review the content covered in the preliminary design review (M3) and will also include final designs and specifications, detailed test results, and issues with the product and its development.

1.2. Customer Needs

The user establishes the needs of a project. There are multiple types of users, ranging from the stakeholders, who are the project investors, to the end users, who are the ones purchasing or using the given product. To ensure that a project is successful on all levels, including the final product, all needs must be met. Detailed customer requirements will be presented in Appendix A.

The targeted end users of this project are researchers and colleges that require a subsonic test vehicle. By providing a model rocket system that always returns to a designated position near the launch site, the stakeholders' risk of losing expensive test equipment is reduced significantly compared to current methods. Future customers may be rocketry clubs or the amateur model rocket community; however, they are not the specific target of the project. The stakeholders, the University of Central Florida, provided a list of needs and functions at which the product must perform. The team conducted a needs analysis, component decomposition, and functional decomposition to ensure that all customer needs and functions are met.

1.3. Report Section Summaries

1. Introduction

Outlines the background, problem, customer needs, and rationale for the project. Also includes a summary of the report's structure. Refers to detailed info in appendices.

2. Project Objectives & Scope

States both long-term and current-semester objectives in bullet points. Clarifies what's included and excluded from the project scope.

3. Assessment of Relevant Existing Technologies and Standards

Summarizes prior work, relevant standards, and competitor analysis.

4. Professional and Societal Considerations

Explores how the project affects broader issues like ethics, safety, sustainability, economics, and society.

5. System Requirements and Design Constraints

Details functional requirements and constraints (technical, regulatory, etc.).

6. System Concept Development

Describes the final system concept, including visual aids and justification for chosen design versus alternatives.

7. Design Analysis

Explains analytical and modeling techniques (e.g., simulation, prototyping) used to develop the design. Includes subsystem specs.

8. Final Design and Engineering Specifications

Presents the complete design using drawings and metrics. Addresses cost and manufacturability, with detailed info in appendices.

9. System Evaluation

Summarizes testing and validation: how the design meets performance, safety, and reliability requirements. Full test details go in an appendix.

10. Significant Accomplishments and Open Issues

Reflects on goals met or unmet, unexpected results, unresolved issues, and future improvement suggestions.

11. Conclusions and Recommendations

Summarizes project findings and provides action-oriented next steps.

References

Lists all cited sources in a consistent format (e.g., [Author, Year]), including electronic and print sources.

Appendices Overview:

A: Customer Requirements – Full list with unique IDs and scope categorization.

B: System Evaluation Plan – Detailed test methods, tools, and settings.

C: User Manual – Operation steps and safety instructions.

D: Cost & Manufacturability Analysis – Cost estimate for manufacturing

E: Expense Report – Tabulated project expenses

F: Manuals & Documents List – All relevant manuals and docs for components used.

G: Design Competencies – Competency matrices for AE/ME programs.

H: Code – Electronics code and flight algorithm

2. Project Objectives & Scope

2.1. Project Goals

At the beginning of the project, the objective was to build, test, and demonstrate a rocket with an RTLS system. The customer needs were…

1. Rocket shall follow all NAR and SRA safety standards and guidelines.
2. Rocket shall reach an apogee of 2,200 feet AGL.
3. Rocket shall reach a target no further than 150m from the launch site.
4. Rocket shall land no further than 800 feet from the target.
5. Rocket shall not be actively guided during the ascent phase of flight.
6. Rocket shall use an electronic ignition system.
7. Rocket shall not use a combustion landing system.
8. Rocket shall not produce more thrust than its weight on descent.
9. At least one member shall be L1 certified.
10. Rocket shall not use a “Sparky” motor.
11. Motor impulse shall not exceed Level 1
12. Rocket shall record FPV during descent and should transmit data to a ground station.
13. Rocket should have a sound emitting device (screamer) on board.

Due to several issues with software and hardware integration, procurement and safety, the rocket will no longer be tested with an RTLS system. The RTLS system will instead be demonstrated through hardware-in-the-loop and flight path simulations. These issues will be discussed in greater detail in Section 10. With these considerations, the project goals were refined as follows…

Develop a Dual Deployment Rocket with RTLS Integration:

Successfully design and construct a dual deployment rocket platform that supports integration with an RTLS system. This includes physical compatibility for mounting servos and control surfaces, as well as software integration with onboard guidance algorithms.

Conduct a Successful Flight Test:

Execute at least one flight test using a conventional dual deployment configuration to validate the reliability of the airframe, deployment mechanisms, and recovery sequence independent of the RTLS functionality.

Collect Real-Time Telemetry from a Customized Electronics Bay:

Design and implement a custom electronics bay capable of transmitting real-time telemetry data during flight. This system will monitor and log key flight parameters such as altitude, velocity, orientation, and deployment events to support post-flight analysis and RTLS development.

Demonstrate RTLS Algorithm Performance via Monte Carlo Analysis:

Simulate and validate the RTLS flight control algorithm using Monte Carlo analysis. This statistical approach will evaluate the robustness and accuracy of the RTLS system under a wide range of environmental and launch conditions, ensuring it can reliably guide the vehicle back to a predefined recovery location.

These objectives collectively support the project's broader mission to explore autonomous recovery methods and enhance reusability in amateur rocketry systems. Future work

could be completed to develop the RTLS system to a point of flight readiness. Details of this work will be outlined in Section 10.

2.2. List of Objectives

The major remaining project milestones and their dates of completion are as follows…

1. Testing and building will occur during 3/13-4/7
2. Critical design review (M6) will be completed during 4/8-4/14
3. Final launch will be on 4/12
4. The showcase video will be finalized on 4/12
5. Project will be presented at the virtual showcase from 4/14-4/15
6. Final presentation will occur during finals week between 4/21-4/25

3. Assessment of Relevant Existing Technologies and Standards

3.1. Manufacturing Methods

This section focuses on current manufacturing methods related to rocketry. Selecting the appropriate manufacturing method is critical to prevent component failure. Proper manufacturing methods can reduce costs, production time, and the likelihood of product failures associated with structural design. There are three primary component manufacturing processes: additive manufacturing, subtractive manufacturing, and casting. Each of these processes has its own strengths and weaknesses [1].

3.1.1. Additive Manufacturing

In additive manufacturing, components are built by layering materials such as fiberglass, carbon fiber, plastics (often using a 3D printer), and metals. Examples of components produced using additive manufacturing include fiberglass rocket body tubes, carbon fiber rocket body tubes, and 3D-printed nose cones [2, 3].

Figure 1: Fiberglass rocket components (left); carbon fiber rocket body (center); 3D-printed nose cone (right)

The primary method of additive manufacturing is **Fused Deposition Modeling** (FDM). To create a part using FDM, the engineer first generates a CAD (Computer-Aided Design) file, such as an STL file, and inputs it into slicing software to determine the tool path. The internal geometry of the component significantly affects its strength characteristics. Within the slicing program, the engineer must adjust variables such as layer height and width, infill percentages, and internal geometry.

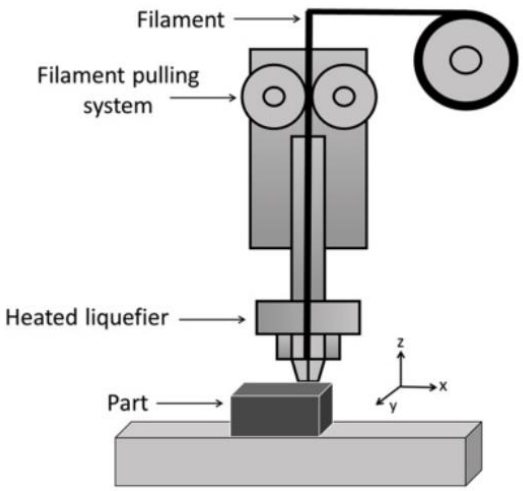


Figure 2: FDM Process

FDM printers primarily use cylindrical nozzles, meaning layer height and width are typically the same (Figure 2). Layer height and width refer to the thickness of each material layer, which can be modified through flow speed or nozzle size. The strength of bonds between layers is critical since failure often occurs at the weakest bonded layer. To improve layer bonding, engineers can [4-6]:

- Increase layer width to enhance adhesion surface area.
- Increase printing temperature to encourage better layer mixing.

- Allow the print to cool slowly, reducing voids and cracks

The **percent infill** setting determines the volume of material inside the part. Adjusting the infill percentage has a substantial impact on strength characteristics. Higher infill percentages increase part strength but also add weight and extend print time (Figure 3). Advanced computational analysis or physical testing is necessary to determine the ideal infill percentage to meet strength requirements while considering the weight-to-strength ratio. Additional weight may require more thrust from the motor to achieve the desired altitude.

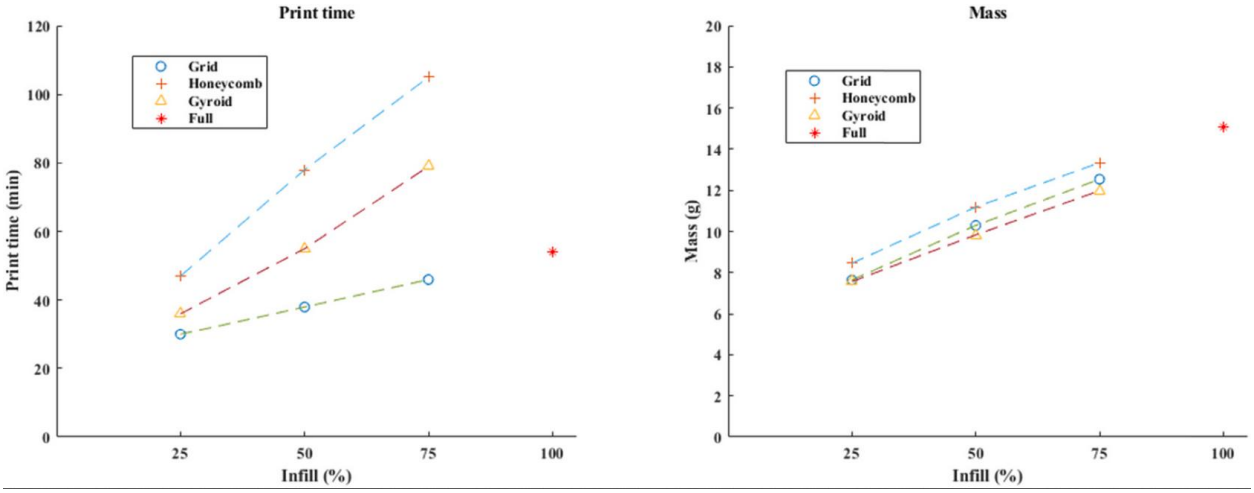


Figure 3: Plots of print time vs. infill (left); mass vs infill (right)

The **internal geometry** of the infill also affects the part's strength. Since 3D printing is a relatively new manufacturing method, limited studies are available on the effects of internal structures. This paper examines three common geometries: Grid, Honeycomb, and Gyroid (Figure 4). Each geometry distributes force differently. Testing was conducted to determine which geometry provides the best strength characteristics (Figure 5).

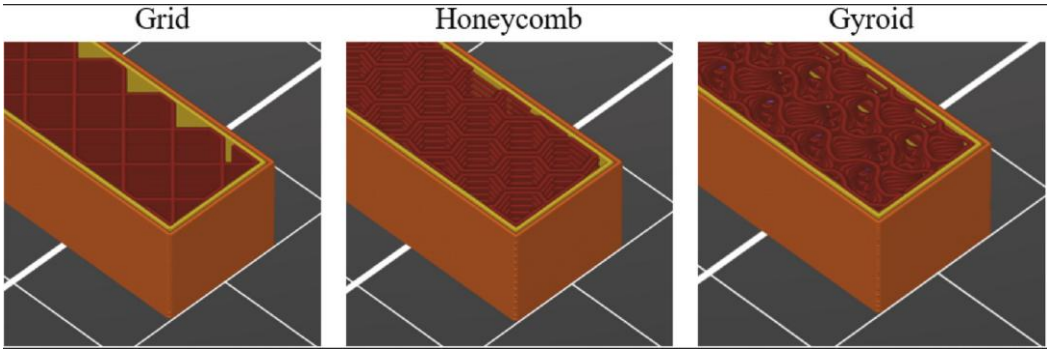


Figure 4: Internal geometry types

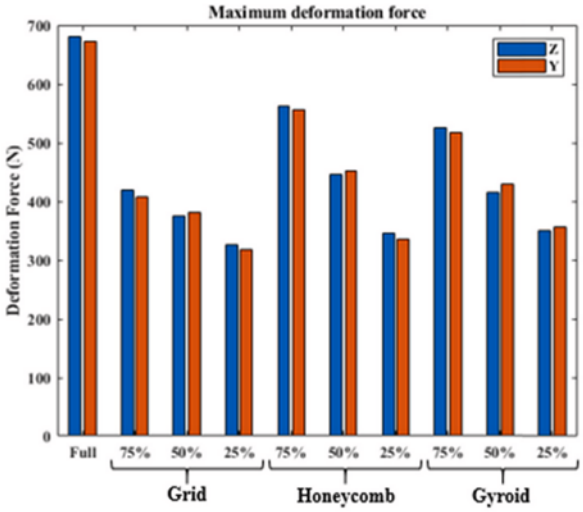


Figure 5: Maximum deformation force vs. infill percentage for different internal geometries

3.1.2. Subtractive Manufacturing

In subtractive manufacturing, a solid block of material is shaped into the desired component by removing excess material. Examples include CNC milling, electrical discharge machining (EDM), laser cutting, waterjet cutting, plasma cutting, and routing. An example of subtractive manufacturing in rocketry is the use of laser cutters or routers to produce rocket fins (Figure 6). This process allows for the use of stronger materials, such as wood and metals [7, 8].

Figure 6: Laser cut rocket fins

3.1.3. Casting Manufacturing

In casting, heated liquid material is poured into a mold, allowed to cool, and then removed to create the desired component. Common casting methods include sand casting, cold chamber die casting, and injection molding. While casting could be used to fabricate rocket motors, UCF guidelines prohibit the team from manufacturing motors for this project. One viable method for the team is injection molding (Figure 7) [9-11].

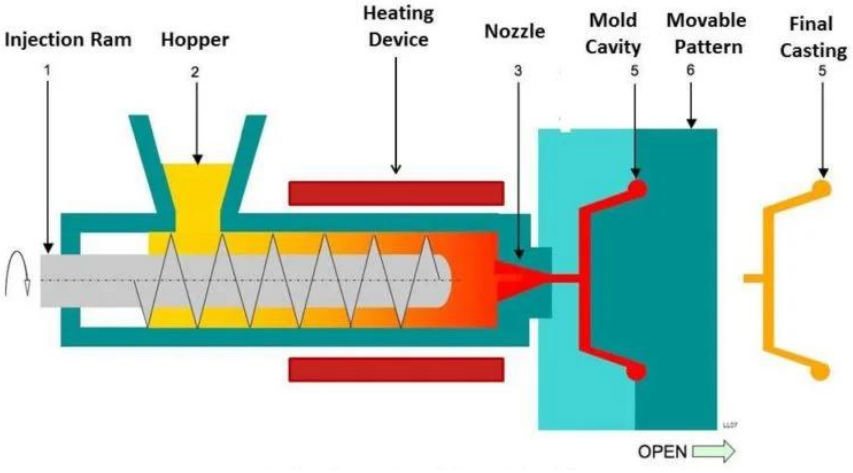


Figure 7: Injection molding device

3.1.4. Summary

Manufacturing Method	Advantages	Disadvantages
Additive	- Easily create complex geometries - High accuracy to CAD model - Enables rapid prototyping - Low material waste - Cost-effective for small runs	- Limited material options for 3D-printed parts - Lower strength of 3D-printed parts - Surface finish requires post-processing - Size limitations - Slowest production time
Subtractive	- High precision and accuracy - Wide range of materials, including high-strength options - Good surface finish	- High material waste and cost - Requires specialized skills - More expensive than additive methods - Limited ability to create complex shapes
Casting	- Fast and cost-effective for large-scale production - Can create complex shapes - Wide range of materials	- Expensive for small runs - Requires mold creation and specialist skills - Risk of defects such as air bubbles

Table 1: Summary of manufacturing methods

3.2. Aerodynamics/Propulsion Systems

When designing a rocket, it is critical for engineers to understand that propulsion and aerodynamics are intimately related. In order to fulfill mission requirements, a rocket must produce enough thrust to overcome drag while in Earth's atmosphere. Drag is especially significant at high speeds and lower altitudes, which can easily be seen in Equation 1 . An effective and efficient design will have minimal drag and high specific impulse. Specific impulse is defined as the amount of impulse per unit of fuel burned.

$$D = C_D S(0.5\rho u^2)$$

Equation 1: Drag force calculation

This study will outline the aerodynamic characteristics of rocket components available in literature. It will also include information on common rocket propulsion system configurations and fuels. The focus of these topics will be for small-scale applications.

3.2.1. Aerodynamics

For the applications of this project, the team does not expect airspeeds to reach speeds even close to Mach 1. Because of this, the aerodynamic calculations will be greatly simplified. However, the team engineers must still be vigilant and ensure all variables are accounted for.

A rocket is considered a streamlined body. On a streamlined body, the effects of viscous drag are more significant than pressure drag because of the large surface area and relatively small wake region. In order to ensure that a rocket is streamlined, the geometry must be thoughtfully chosen. Research on various rocket geometries will be presented later in this study.

When a rocket yaws or pitches off course, rocket fins generate lift. This lift is referred to as a restoring force because it steers the nose back toward the flight direction. This is why fins are essential to rocket stability, and their design will be discussed in more detail in a later section. However, the lift force is only restoring if the center of pressure is located below the rocket's center of gravity (Figure 8) [12]. This is an essential rocket design criterion. The center of pressure is defined as the average location where the pressure (aerodynamic) force is applied. This concept is easily understood as the point where lift and drag act through and where there is no aerodynamic moment [13].

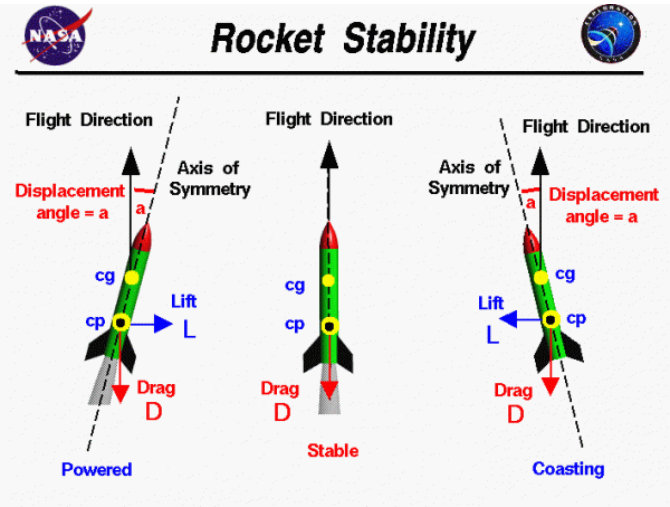


Figure 8: Rocket Stability [12]

Nose Cone Designs

Nose cone geometry is an important choice a rocket engineer must make when considering aerodynamics. There are several nose cone geometries which have been thoroughly studied in the literature and used on missions. Different nose cone geometries are optimized for their vehicle's flight conditions.

In subsonic model rockets, a short, blunt, smooth nose cone geometry is best. A study by Iyer and Pant showed that minimal drag coefficient values were attained for an elliptical nosecone at low Mach number [14]. However, it should be noted that common nose cone shapes show very little deviance in performance for low-speed flight. A much more important factor to consider is that nose cone geometries can significantly alter the location of a rocket's center of pressure. E.g., a larger nose cone would move the center of pressure further toward the tip. This design factor will be carefully considered in the later development stages.

Fins

As discussed earlier, fins are essential to a rocket's stability because of the restoring lift force they produce. There are several different fin shapes available, with a few illustrated in Figure 9. A study by Pektas et al. revealed that the parameters span length and thickness dominated in terms of increasing apogee, and span length dominated in terms of stability [4].

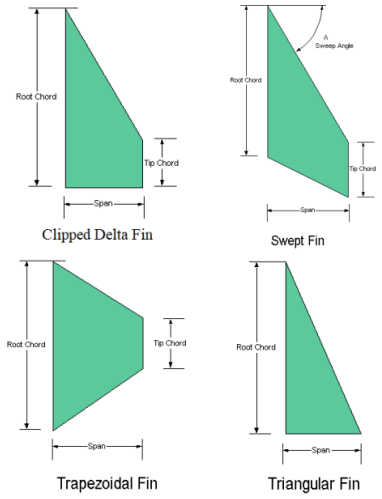


Figure 9: Model Rocket Fin Shapes [15]

A magazine article by Apogee Components also stated that adding an airfoil profile to a rocket's fins significantly reduces drag and increases stability. Elliptical fins are the most optimal shape for stability, but clipped delta fins perform almost the exact same and can have airfoils incorporated much easier [16]. The reason for this is that the lift force acts closer to the airframe for these swept shapes, which reduces vortex formation around the edges and induced drag. Induced drag is the drag component that results from the generation of lift.

3.2.2. Propulsion Systems

There are several different classes of solid rocket motors available on the market. For this project, a level 1 class motor will be used. Level 1 includes H and I class motors, which require an L1 certification through NAR or TRA to purchase and operate [17].

Rocket motors specifications are designated by a three-part code. An example is shown in Figure 10. The three parts define motor parameters as…

1. A letter that represents **total impulse** ("C")
2. A number that represents **average thrust** ("6")
3. A number specifying **delay time** between burnout and ejection charge activation ("3")

Figure 10: Motor Code Example [17]

Specific impulse is another important motor parameter which can be represented as the total amount of power produced per unit of propellant mass. The specific impulse is used to analyze a motor's efficiency. For example, a high specific impulse translates to high efficiency.

High-powered rocket motors (level 1 or higher) generally have the same components. These components are outlined in Figure 11. The propellant, most often aluminum powder and ammonium perchlorate (APCP), burns and exits through the nozzle to produce thrust. The delay charge then burns to allow the rocket to coast after burnout, then the ejection charge ignites to deploy the recovery system. All of these components are housed in a case. Some rocket motors are "plugged", meaning they contain no ejection charge. These are usually used for larger rockets that activate their recovery system using an altimeter or accelerometer [18].

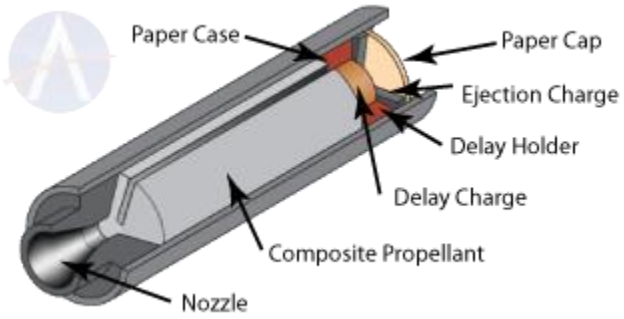


Figure 11: Rocket Motor Cross Section [19]

A thrust curve can be used to analyze a motor's performance. A thrust curve is a plot of a motor's thrust over time (Figure 12) and can be used to derive average thrust. To determine the motor specifications needed for a certain mission, several factors must be considered.

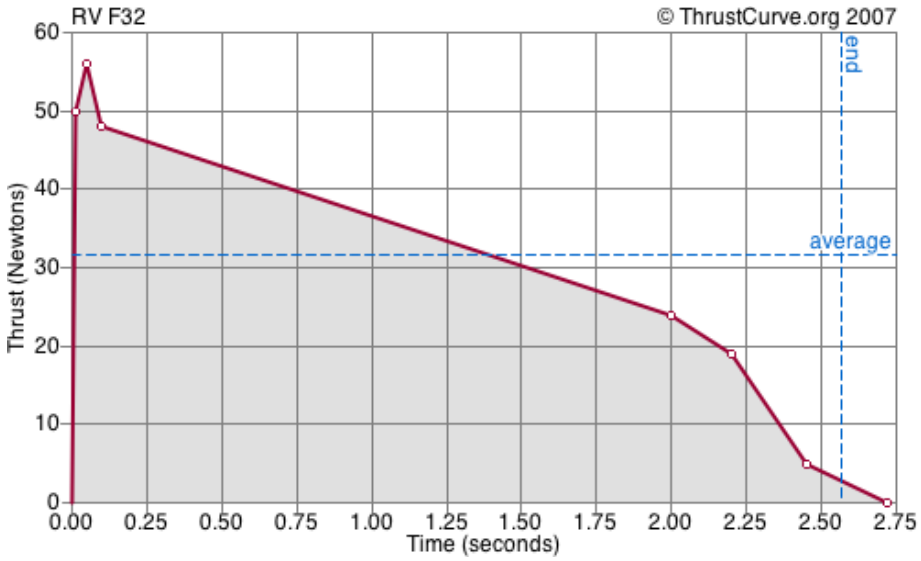


Figure 12: Thrust Curve Example [20]

Selecting a Proper Motor

The most important factors to consider when selecting a rocket motor are weight and target apogee. This will determine how much energy is required to complete the mission. Other characteristics to consider are the rocket's drag coefficient and the weather conditions on launch day, which are highly variable. Drag effects reduce the altitude the rocket is able to reach, and bad weather may cause the rocket to go off course. To mitigate these issues, rockets should be optimized for aerodynamics and should launch under clear skies with little to no wind.

Determining the exact motor specifications needed is almost impossible to do analytically because of complicated dynamics. The rocket is constantly reducing in weight, the velocity is changing, the weather conditions are variable, and the heading is changing. Because of this, an effective process for selecting a proper motor involves first using guidelines and intuition to select a group of candidates, then performing simulations to find the best candidate [21]. Some things to consider when selecting a group of candidates are…

1. Motor casing diameter and length
2. Thrust should generally be five times the weight of the rocket [18]
3. Rocket drag coefficient

Once candidates are selected, they can be imported into the OpenRocket software. OpenRocket can conduct simulations on a user-specified rocket configuration to determine parameters such as maximum velocity, apogee, and flight time.

3.3. Structures and Materials

This section outlines key structural design and material considerations for the development of model rockets. The information presented in this document is based on recent research into advanced materials and what parts would be best suited for the rocket. The team's focus is on using lightweight, durable materials that can withstand the forces during launch, flight, and recovery, and exploring the ways that the qualities of these materials can be tested.

3.3.1. Materials for Model Rockets

Material choice is crucial in determining the performance and durability of model rockets. Based on current research, the most promising materials include:

Polylactic Acid (PLA)

PLA, a biodegradable thermoplastic, is widely used in 3D-printed model rockets due to its ease of use and environmental benefits. Despite being heavier than traditional plastics like ABS, it offers good durability and cost-effectiveness, especially for educational or experimental use [22]. However, PLA has much lower heat resistance than other plastics such as ABS and PETG.

Polyethylene Terephthalate Glycol (PETG)

PETG is widely used in 3D printing because of its high strength, rigidity, heat resistance and weather resistance. PETG shares similar properties to ABS but produces very little toxic fumes and does not require an enclosure to print. PETG also shares the same low warping characteristics with PLA, but without all the durability issues.

Composite Materials

Carbon fiber and fiberglass are frequently employed in aerospace applications due to their high strength-to-weight ratio. Carbon fiber offers excellent rigidity, making it suitable for critical structural components like the airframe and fins [22]. Fiberglass, while slightly less strong, is cost-effective and durable, often used for airframes and fins in larger model rockets [23].

Balsa Wood

Balsa wood is a lighter material than plastic, reducing the amount of power needed for the rocket to reach a 2,200 ft apogee. It is also the strongest wood with respect to its weight. Balsa wood is commonly used in fins for low-power rockets due to its high strength-to-weight ratio.

Epoxy Resins

Strong bonding is essential for rocket construction, and epoxy resins provide the necessary adhesive strength for attaching fins and other structural elements. When used with composites like carbon fiber, epoxy can ensure high-strength joints [24]. JB Weld performs well under high-heat applications; however, it is heavier compared to other epoxies [25]. There have been applications where acetone is mixed with epoxy to thin it for weight reduction. Loctite clear epoxy is best suited for gluing wood but still works well with other materials that are not smooth [25].

3.3.2. Modular Design and Additive Manufacturing

The application of additive manufacturing (AM) offers a flexible approach to rocket design by enabling the production of modular, multi-functional parts [22]. AM allows for

complex geometries in design that are either difficult or impossible to create using other manufacturing methods. The structures often involve lattice designs that can reduce weight while maintaining or possibly enhancing strength [26]. It also allows for the easy production of models using materials that perform well under conditions of rocketry, such as PLA, ABS, or PETG

3.3.3. Structural Design Considerations

The structural design of a model rocket must balance lightweight materials with the ability to withstand aerodynamic and thermal stresses. Key considerations include the airframe, fins, and nose cone. An example of a generic rocket configuration is shown in Figure 13.

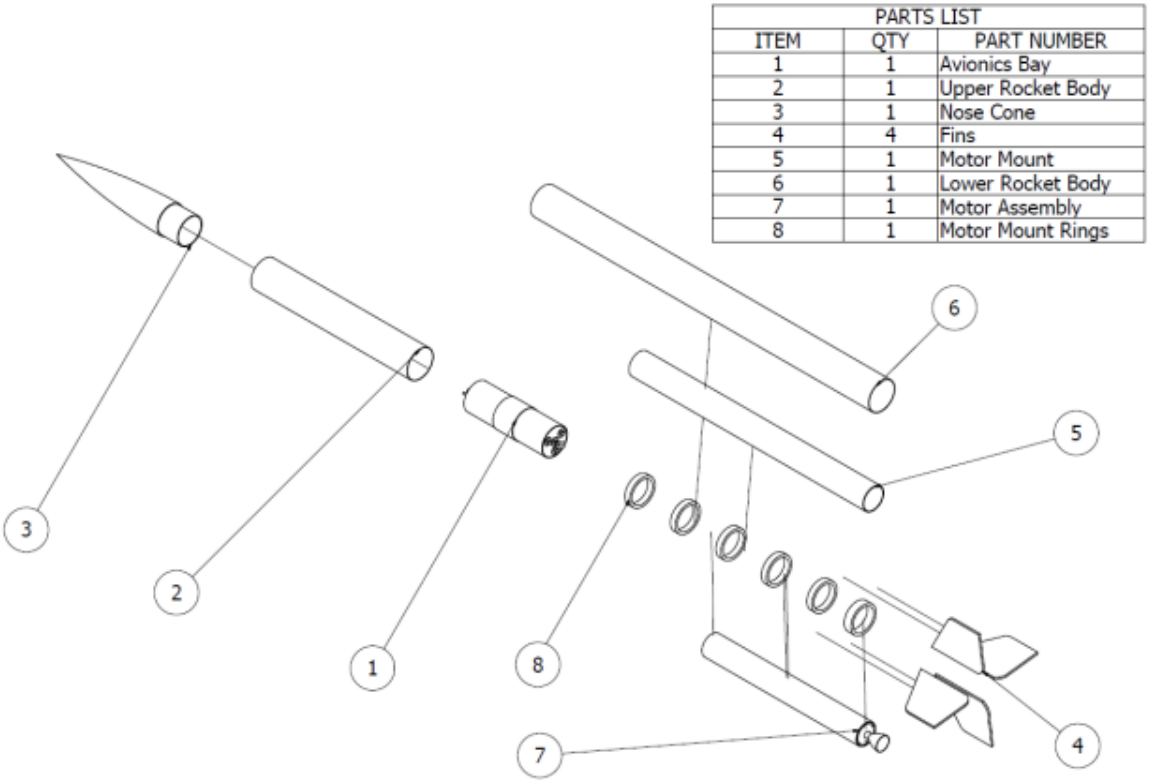


Figure 13: Parts of a model rocket [27]

Airframe

The airframe serves as the main structural body of the rocket and must be capable of withstanding the stresses during flight. Stress in the airframe is highest during powered flight, as this is when the motor is burning. Low powered rockets traditionally use Kraft paper for body tubes. Kraft paper is very lightweight, but highly prone to zippering when contacting the shock cord during flight. Many high-powered rockets use fiberglass or plastics (PETG, ABS, ASA) for their body tubes. These materials are heavier but offer superior strength to Kraft paper. Carbon fiber composites are a much more expensive option, but have a superior strength to weight ratio to all of these materials [24].

Fins

Balsa wood is a very good material for fins on low powered rockets due to its high strength-to-weight ratio. Balsa tends to dent or split if it's handled roughly and needs to be thinner than 3mm. In that case, basswood has been used as a stronger alternative. For high powered rockets, 3D printing fins is a common choice among designers. 3D printed fins can easily incorporate complicated geometry like tapered cross sections and airfoiled edges. A common challenge with installing fins is alignment. 3D printing eliminates this concern, as fins can be fixed to the model on the printer. Fiberglass is also a strong contender for fin construction, as it has the same strength qualities and great heat resistance [28].

Nose Cone

Nose cones for model rockets are commonly made of plastic and manufactured with 3D printers [28]. This is because nose cone geometry is difficult to construct by hand. 3D printers also allow designers to easily add weight to the nose cone without affecting the geometry, which

is often necessary for stability. A heat-resistant filament such as PETG or ASA would be good for rocketry applications to avoid the risk of warping. Research also suggests that a parabolic or ogive nose cone made from lightweight composites such as fiberglass is optimal for durability and low weight while maintaining structural integrity [22].

Simulation Testing

Finite Element Analysis (FEA) is a powerful tool to test the structural integrity of materials used in model rockets. By breaking down the rocket's structure into smaller, manageable elements, you can simulate how different materials and designs behave under various conditions like stress, temperature, and vibration. Physical launch tests can also be used to provide real-world data on stability and performance under flight conditions.

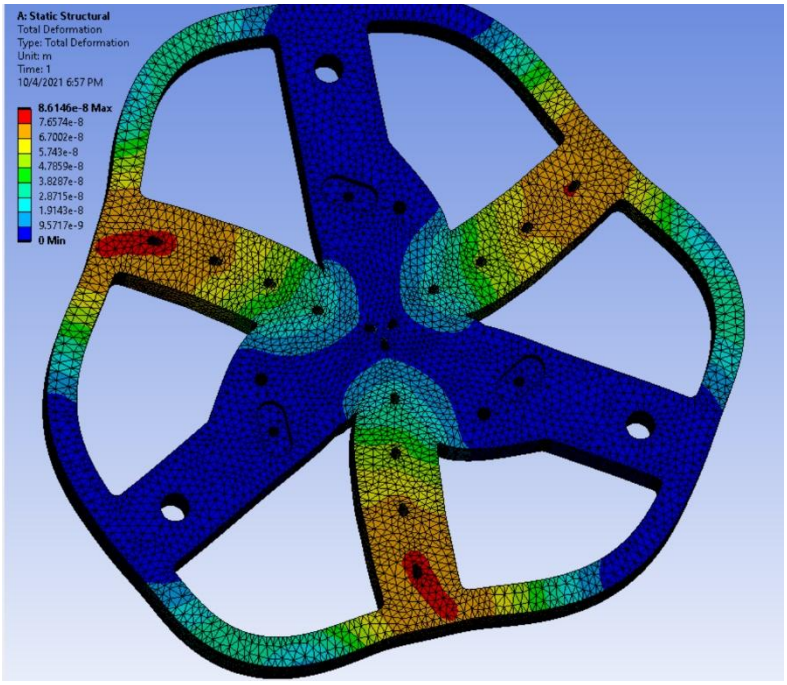


Figure 14: FEA of airbrakes by Cyclone Rocketry [29]

3.4. Guidance Systems and Avionics

When analyzing the key components that make a successful rocket, they can be divided into four sections: structural systems, payload, propulsion system, and finally, the avionics system. The rocketry avionics system will be the main point of focus for this technology study memo. The avionic control system of a rocket is the epicenter of the electrical systems that are crucial for a successful flight [30]. Figure 15 shows an example of what the guidance system onboard a model rocket could look like and the electrical components that are part of the system [31]. These electrical systems are then programmed with software that ensures the rocket is headed in the right trajectory. The avionics system works hand in hand with other components of the rocket, such as the engines, to guide the rocket in the intended path. This technology study memo will delve into the following topics: guidance systems, data acquisition/telemetry, and power management and distribution.

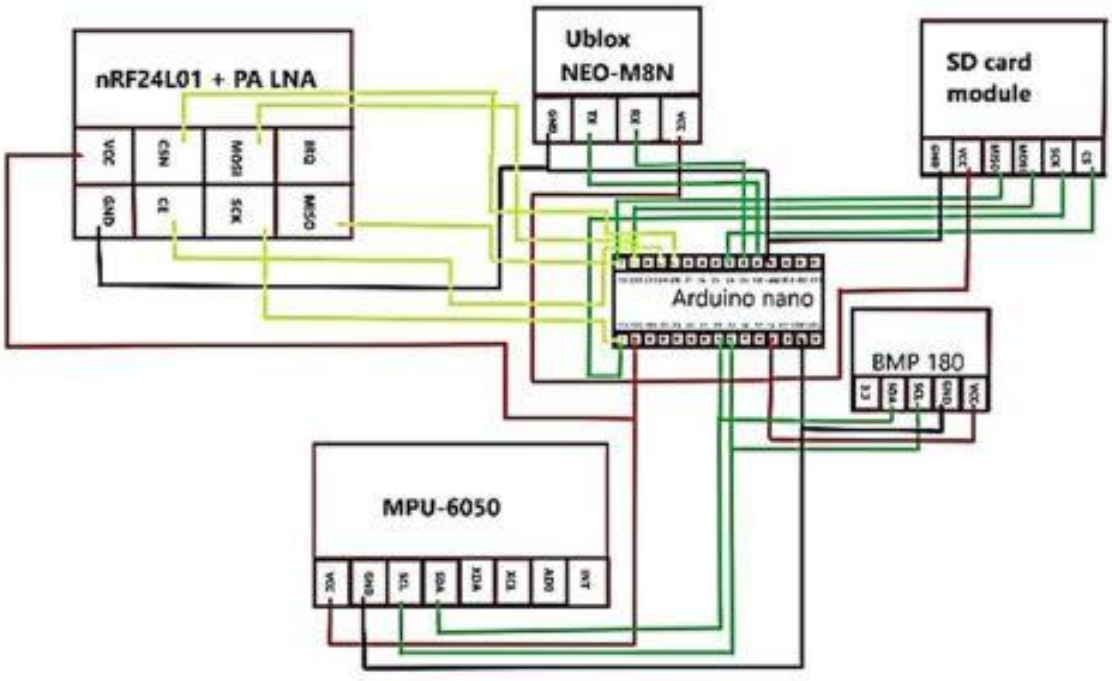


Figure 15: Avionics System of a Model Rocket

3.4.1. Guidance System

A critical component of the avionics system is the guidance system. The guidance system is a subsystem of the avionics that handles the stability and precision of the rocket's intended trajectory [32]. Consisting of components such as the GPS and the altimeter, the guidance system is responsible for determining the location, altitude, and orientation of the rocket during ascent. In the proposed system for this project, the guidance system becomes a component of the recovery system and focuses on guiding the rocket back to the desired site using mechanisms to manipulate the parafoil of the rocket. Figure 16 illustrates the intended path of an RTLS rocket [33]. This path is carried out by utilizing a guidance system within the rocket.

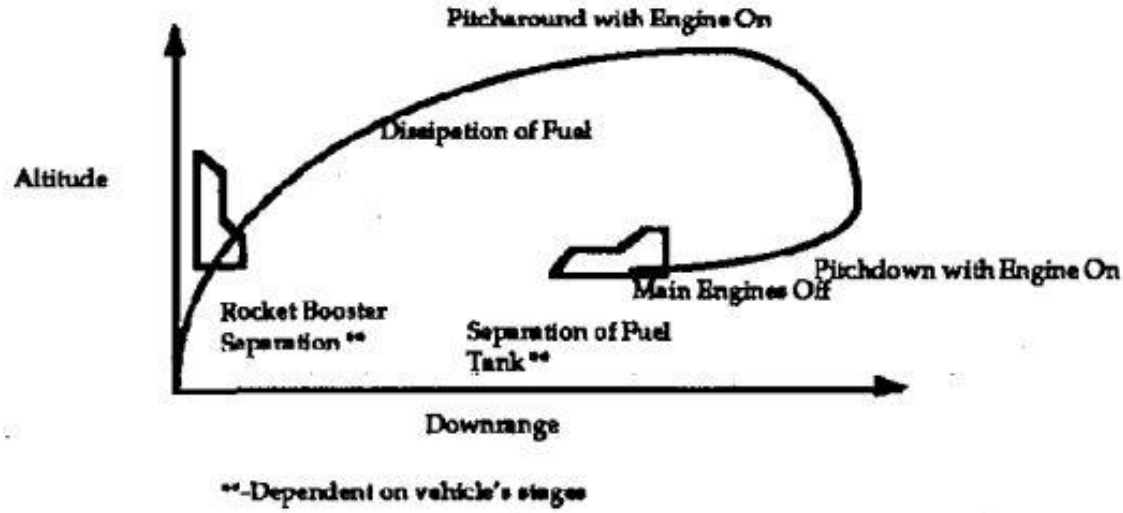


Figure 16: General Path of an RTLS Rocket

As the guidance system collects data on factors such as the position or altitude of the rocket, this data isn't utilized to its full potential if it is not compiled and transmitted back to a ground station for the team to analyze. The guidance subsystem works with other components of the avionics system to collect this data, compile it, and transmit it for the team to utilize. This is

done through another process within the avionics system that handles data acquisition and telemetry.

3.4.2. Sensor Fusion & Filtering

Sensor fusion is the process of combining data from multiple onboard sensors to produce more accurate and reliable information about the rocket's state. During flight, sensors such as the GPS, barometer, and inertial measurement units (IMUs) generate data with varying levels of noise, latency, and accuracy. These imperfections can lead to inconsistencies in the system's perception of position, velocity, and orientation.

To address this, the avionics software applies filtering techniques to reduce noise and reconcile discrepancies between sensors. One of the most used methods is the Kalman filter, a recursive estimator that provides statistically optimal predictions of system states by incorporating both system dynamics and sensor noise models [33]. Kalman filters are especially effective in dynamic environments where continuous adjustments are needed in real time.

Figure 17 demonstrates a simplified Kalman filter flow diagram. In this model, sensor observations (e.g., from GPS and IMUs) are filtered through prediction and correction loops to yield real-time estimates of the rocket's position and velocity.

$$\text{pdf}(w) = \frac{1}{(2\pi)^{n/2}|W|^{1/2}} \exp\left\{-\frac{1}{2} w' W^{-1} w\right\}$$

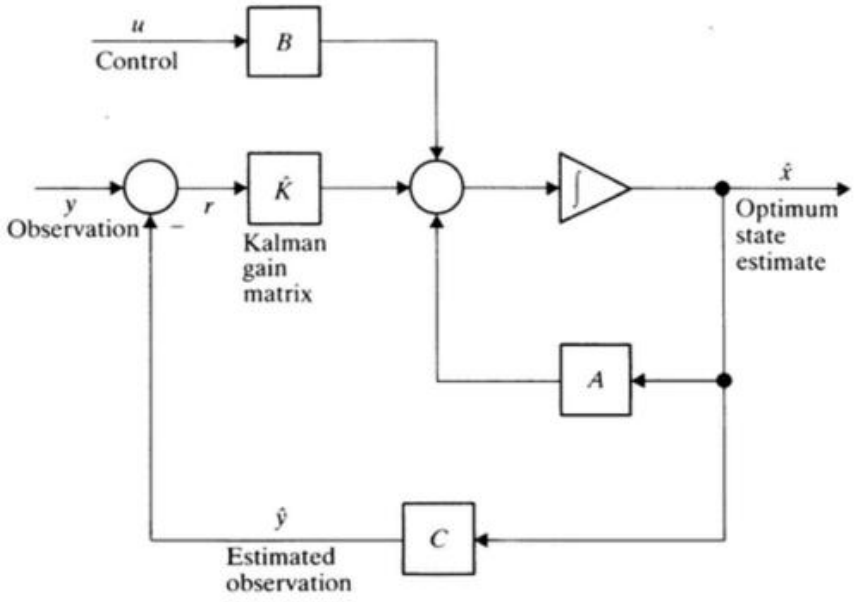


Figure 17: Kalman filter estimation structure demonstrating prediction and correction using multi-sensor input.

In addition to Kalman filtering, low-pass filters are often used to eliminate high-frequency noise from accelerometer or gyro data. Complementary filters may also be employed to combine the long-term stability of one sensor with the short-term responsiveness of another, such as fusing magnetometer and gyroscope data for accurate orientation tracking [34].

3.4.3. State Estimation

State estimation refers to the process of reconstructing the rocket's full dynamic state—such as position, velocity, and orientation—using filtered sensor data. Because individual sensors

are limited in the types of data they provide and may suffer from signal degradation or error, state estimation provides a consistent and complete picture of the rocket's motion during all phases of flight.

This process relies on dynamic models that predict how the rocket is expected to behave based on physical laws and known control inputs. These predictions are then corrected using filtered sensor measurements in real time. Techniques like the Extended Kalman Filter (EKF) are commonly used in aerospace applications to account for nonlinearities in motion and sensor behavior [35].

Figure 18 illustrates the structure of an EKF as applied to aerospace state estimation. The diagram highlights how predicted and observed states are fused using system and observation models, producing a highly accurate estimate of the rocket’s actual flight conditions.

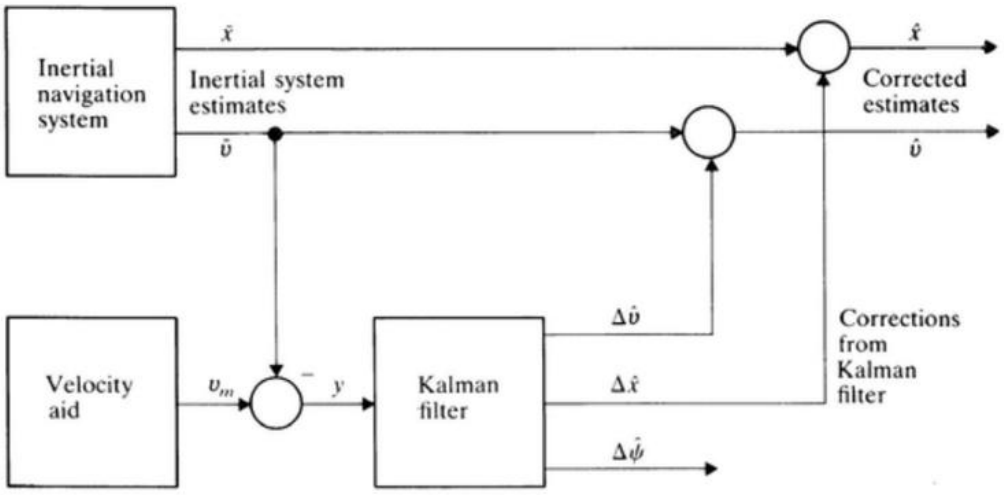


Figure 18: Use of a Kalman filter for fusing inertial and non-inertial navigation data, demonstrating state estimation in aerospace systems.

Accurate state estimation is crucial for executing the guidance strategy, particularly during descent. The estimated state serves as input for feedback control, enabling the rocket to respond intelligently to changes in its environment and maintain its course back to the designated landing site.

3.4.4. Closed-Loop Control

Closed-loop control, also known as feedback control, is essential to the rocket's ability to correct its flight path during descent. In a closed-loop system, the current state of the rocket—estimated through sensor fusion and state estimation—is continuously compared to the desired state. The control system then computes adjustments to actuators or steering mechanisms to minimize the error between the two.

For the RTLS system, a Proportional-Derivative (PD) controller is implemented. The proportional term reacts to the magnitude of the error, providing a corrective response based on how far off-course the rocket is. The derivative term accounts for the rate of change of this error, dampening overshoot and improving system stability [36]. This combination ensures responsive but smooth corrections, which is essential during descent under parafoil guidance.

Figure 19 shows a standard block diagram of a PD controller in a closed-loop system. This layout emphasizes the feedback loop, where the error signal is continuously calculated and minimized to control the rocket’s descent path.

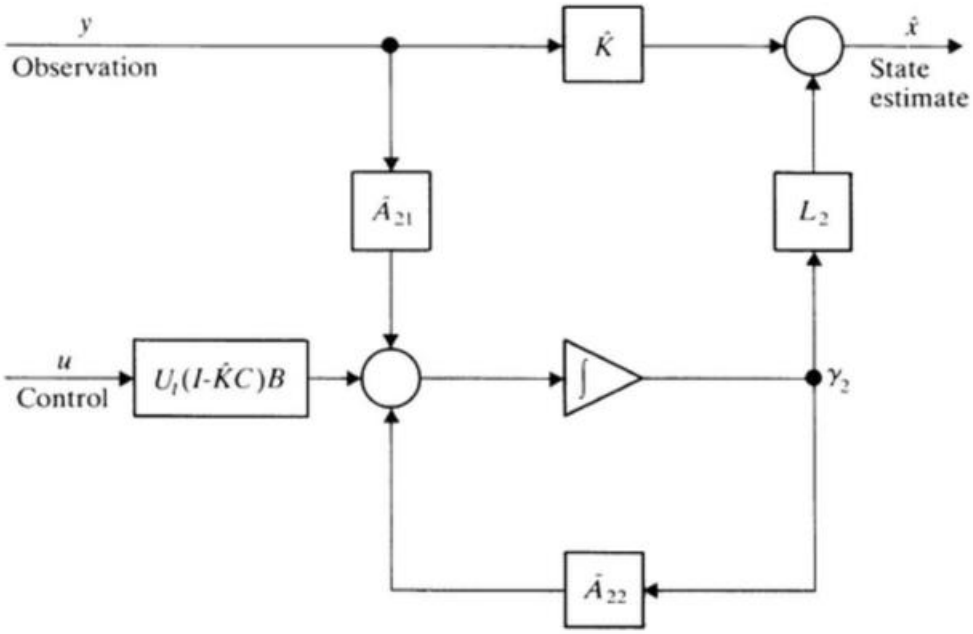


Figure 19: PD-like control architecture shown as a reduced-order observer in a closed-loop system.

The feedback loop operates until the rocket reaches the landing zone, allowing the system to adapt to environmental disturbances such as wind or thermals. This approach enhances the precision and reliability of the rocket's return trajectory.

3.4.5. Data Acquisition and Telemetry

Telemetry is the act of collecting measurements to analyze and follow the progress of the system undergoing action [34]. Within the scope of this project, telemetry applies to the rocket's motion, position, altitude, and a plethora of other data points. Telemetry data is collected from a system of sensors and transmitters that measure, process, and finally transmit the data to the ground station for the launch engineers to analyze. The main goals of the telemetry system, after the data is acquired, are to alert the team of the status of the rocket (speed, position, altitude, etc.)

and to alert the team of any failures through inconsistencies in the transmitted data. Figure 20 shows an example of how the acquired data can be analyzed and displayed to the team at the ground station [35].

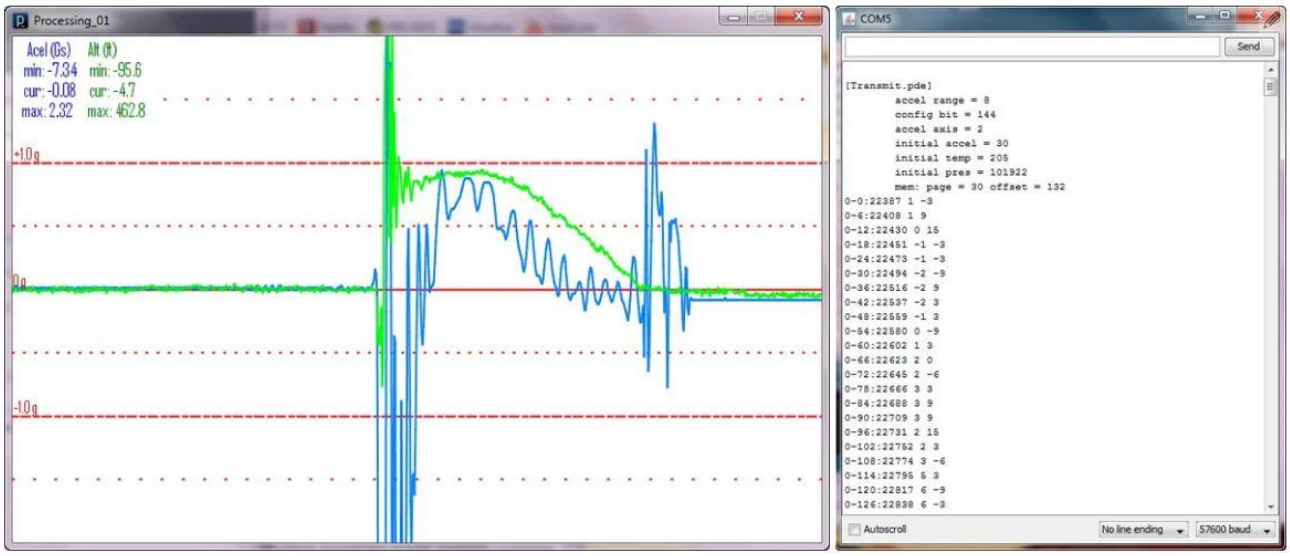


Figure 20: Telemetry Flight Data

As Figure 20 shows, the team will benefit immensely from telemetry data, as it provides a live view of the performance of the rocket. Any discrepancies or failures will immediately alert the team as the data will display these in real-time. The data acquisition/telemetry subsystem of the avionics is a critical system that processes the performance of the rocket in real-time and can also be analyzed after launch. With reliable data acquisition and telemetry systems ensuring real-time monitoring and communication, the rocket's avionics rely on efficient power management and distribution to sustain these critical operations throughout the flight.

3.4.6. Power Management and Distribution

To mitigate issues and electronic failure during ascent and recovery, it is critical for the team to consider a power management system that manages power efficiently. The power

management system of the avionics ensures that all the electrical components receive the necessary power during flight. Important considerations for this system are to utilize efficient power distribution networks and ensure redundancy to handle any potential failures. When analyzing an efficient power distribution system for avionics, the key components include the following [36]…

1. **Power Sources**: Typically, for a model rocket, batteries such as Lithium-polymer (LiPo) or Lithium-ion (Li-ion) are utilized due to their high energy density and lightweight properties.
2. **Power Distribution Network**: This network ensures that power is delivered to all the components of the avionics system, such as the sensors, flight computers, and communication/telemetry system.
3. **Voltage Regulation**: Used to maintain consistent voltage levels, protecting any sensitive electronics from issues.
4. **Redundancy & Safety**: Incorporating redundant power paths and safety mechanisms helps prevent system failures during the critical flight phases (ascent and recovery).

With these four key components within the power management system, the possibility of any electronics-related failures is heavily reduced for the RTLS team.

3.4.7. Summary

The avionics system of a model rocket is essential for achieving mission success, integrating precise guidance systems, reliable data acquisition and telemetry, and efficient power management and distribution. These interconnected systems ensure stability, monitor performance, and mitigate potential failures during critical flight phases. As the RTLS rocket

progresses, thorough testing and simulations will remain crucial for enhancing system reliability and advancing innovative rocketry capabilities.

3.5. Recovery Systems

3.5.1. Overview

The recovery system is one of the most essential parts of any rocket. Without the use of a recovery system, the rocket would be subject to rapid, uncontrolled descent that could possibly injure others, destroy property, and make the rocket unable to relaunch. Depending on the scale of the rocket, there are multiple recovery systems that can be applied, from streamers to parachutes, and even thrust-vectored descent. In the case of model rocketry, in order to receive each certification level, there is a recommended/required recovery system. For a Level 1 (L1) certification, a single-deployment recovery is required, but dual-deployment recovery is allowed [37].

A dual-deployment recovery system utilizes two separation points within the rocket's structure, enabling sequential deployment of two parachutes. To facilitate this design, the rocket's body is divided into three main components: an upper body tube, a coupler, and a lower body tube. The coupler serves as a connection between the upper and lower body tubes and separates at both deployment stages.

Shear pins are installed at the junctions where the tubes attach to the coupler. These pins provide a temporary, secure connection, preventing unintended separation during flight. When separation is required, a black powder charge generates a substantial force to shear the pins. Additionally, the coupler uses a friction fit to ensure stability during flight, minimizing wobble or unintended movement between the connected sections. This combined approach allows the parachutes to deploy smoothly while enabling the body tubes to detach cleanly from the coupler, ensuring reliable recovery [38].

3.5.2. Dual-Deployment Recovery

As for the timing of the separation, the first black powder charge is fired at apogee, the highest point the rocket achieves, thus beginning its descent, and the drogue parachute is deployed. The drogue parachute is smaller compared to the main parachute, as it isn't meant to slow the descent by a large margin, but it allows the rocket to drift less during its descent [38]. An alternative for a drogue in lightweight model rockets is a streamer. A streamer is a long, thin piece of fabric designed to catch air and generate drag. This drag force helps slow down the rocket during its descent.

Once the rocket drops into the 500-800 ft altitude range, the lower body tube is separated from the coupler via a black powder charge, and the main parachute slides out. The application and benefit of a dual-deployment system compared to a single-stage parachute recovery system can be seen in Figure 21. The use of the drogue parachute greatly assists in reducing the rocket drift distance relative to the launch pad due to the wind. Although a drogue parachute will have more drag than a streamer, a streamer will result in much lower drift distance [39].

To prevent damage to the parachutes or streamer, a Nomex blanket can be used as a protective measure. This blanket is typically a square sheet of flame-resistant Kevlar that is large enough to fully wrap around the parachute while allowing it to remain loose. Ensuring the blanket is not tightly wrapped is critical, as a tight fit could cause the parachute to jam inside the body tube during deployment.

The Nomex blanket is securely attached to the shock cord along with the drogue and main parachutes or streamer, offering complete protection during ejection. In a dual-deployment rocket, there are two shock cords. One is in the upper body tube, connecting the nose cone

bulkhead to the coupler bulkhead. The other is in the lower body tube, linking the bottom coupler bulkhead to the motor mount bulkhead. This arrangement ensures proper attachment points for the recovery system components, allowing the rocket to remain securely together throughout the entire flight.

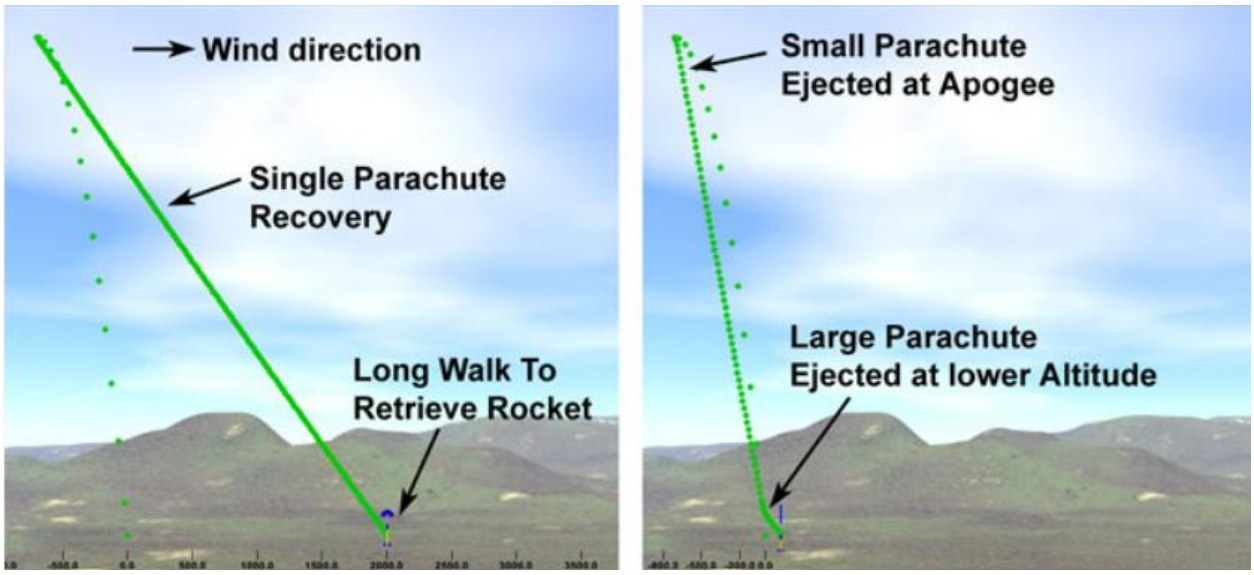


Figure 21: Rocket drift comparison [38]

An important part of designing any rocket system is the concept of operations, otherwise known as CONOPs [40]. These CONOPs often include or solely comprise drawings that illustrate the entire launch-to-recovery system for a rocket, as shown in Figure 22. Figure 22 also shows the drogue and main parachute deployments. Additionally, all the events or numbered points in CONOPs tend to indicate the expected times and heights at which each event occurs.

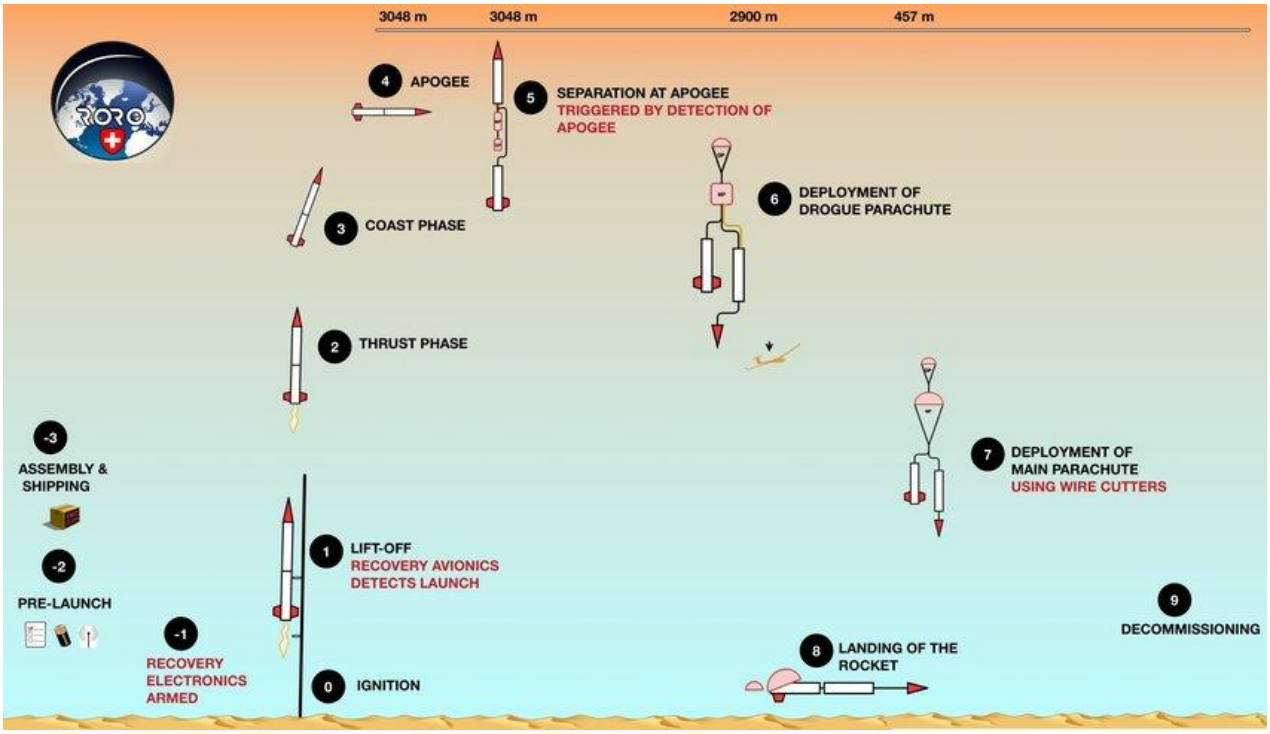


Figure 22: CONOPs Diagram [40]

3.5.3. Pre-Flight Requirements

Before launching a rocket, one of the most important steps is going through the preflight checklist, which all rockets should require before launching. This involves considerations such as weather conditions, rocket assembly, altimeter calibration, black powder charge assembly, and recovery system assembly. All steps must be written out and physically checked off every time to avoid errors, skipped steps, and ensure overall safety for both the launching crew and bystanders.

An additional important part of a preflight checklist is the Go-No Go procedure. This involves one person running through a premade procedure list and confirming with each member of the team that each part of the assembly was completed. This is verified step by step to ensure no steps were missed.

3.5.4. Recovery System Components Summary

Component	Function
Coupler	Holds the upper and lower body tubes, which in turn hold the parachutes; the coupler also contains the sled.
Sled	Secured in the coupler, the sled holds the altimeter, wires, black powder charge, and the batteries.
Altimeter	Records the altitude and flight/glide path throughout the rocket's launch. Also sends signals to ignite the black powder charges at apogee and the designated main deployment height.
Wires/Black Powder Charges	Wires transmit the voltage signal from the altimeter to ignite the black powder charges, causing the shear pins to break and enabling body tube separation.
Bulkhead	Connects the coupler and ties the body tubes, coupler, and parachutes together, ensuring nothing separates during flight.
Shock Cord	Provides a strong rope that ties the rocket together and gives the parachute room to fully expand while holding the rocket.
Nomex Blanket	Wraps the parachutes to keep them tight and bundled in the body tubes while protecting them from black powder charge explosions.
Parachute	The drogue and main parachutes slow the rocket down and enable a safe, controlled descent.
Streamer	Slows the rocket’s descent with minimal drift but decreases in efficiency as weight increases.

Table 2: Recovery Component Summaries

4. Professional and Societal Considerations

The Return-to-Launch-Site (RTLS) rocket features a low-cost guidance system designed to enable controlled descent and targeted landings. Traditionally, guided rocket descent has been limited to large scale aerospace applications due to the complexity and high costs of the technology involved. By contrast, our system demonstrates that it is possible to achieve precise recovery using affordable, compact components that are accessible to hobbyists, educators, and smaller research teams.

This innovation has several important implications beyond the immediate technical scope. First, it promotes sustainability by significantly increasing the likelihood of rocket recovery and reuse, reducing waste and material consumption.

Additionally, the manufacturability of our guidance system supports scalable production using off-the-shelf electronics and 3D-printed components, lowering the barrier to entry for those interested. This can help expand access to advanced rocketry concepts, encouraging engagement among students and hobbyists.

From a safety and ethical standpoint, the ability to guide rockets back to a designated area helps minimize the risk of uncontrolled landings, thereby protecting people, property, and the environment. As small-scale rocketry grows in popularity, ensuring safe, responsible flight and recovery becomes a societal priority one our system directly addresses.

In summary, our RTLS rocket project not only solves a technical challenge, but also contributes to broader goals of sustainability, affordability, education, and public safety in the field of model rocketry.

5. System Requirements and Design Constraints

This section will focus on describing how the guidance and recovery system will operate, since the primary objective of this project is to engineer a system that guides a rocket to a ground target during its descent. The team has proposed a parafoil system that can be controlled from the ground station.

5.1. Theory of Operation and Benchmarking

A parafoil deploys in a similar fashion to a parachute but has an airfoil profile and multiple chords that can manipulate its shape. By manipulating the shape of the parafoil, the aerodynamic forces change. This allows for a controlled descent. Airplane wings change shape similarly by moving flaps, slats, and ailerons, which change the aircraft heading. Prior to committing to this parafoil design, the team conducted benchmarking on RTLS systems developed by other teams. These are summarized in Table 3.

Project	Summary	Benchmark

1. Miniature Autonomous Rocket Recovery System (MARRS) [41]	• Three-stage system
	• Reached an apogee of 7674 feet
	• Three avionics systems
	• Activated BPCs at 0 g's
	• Referenced the snowflake aerial delivery system
	  ○ System is powered by an autonomously guided parafoil
	  ○ Snowflake was designed to airdrop supplies to warzones
		• Snowflake system inspired the guided parafoil concept for the recovery and guidance system
		• This system was used to deliver small payloads into active combat zones, which are comparable in size to high-powered rockets

2. Patent: Guided Parafoil System for Delivering Lightweight Payloads [42]	• Design for a guided parafoil system to deliver lightweight payloads • Parafoil is controlled by a single motor • Parafoil is maneuvered perpendicular to the flight path to travel in a circle above target	• Prompted single motor configuration for rocket recovery and guidance system • Prompted programming the guidance system to travel in a circle above target once within a certain radius

Table 3 Benchmarking Summaries

5.2. Standards

The standards of the National Association of Rocketry (NAR), the Tripoli Rocketry Association (TRA) and the guidelines set by UCF were used when formulating the design parameters [43, 44]. The key guidelines considered during the development of the rocket design parameter were…

1. Rocket must use an electric ignition system
2. Rocket must not use a combustion landing system
3. Operators must be Level 1 certified.
4. Rocket must not be actively guided during ascent.

5.3. Key Requirements and Design Parameters

The key system functions were established based on customer needs. Establishing standards for system functions then allowed the team to determine the design requirements of the guidance and recovery system. These were derived through preliminary engineering analysis in Section 6 of the M3 preliminary design review. These system requirements will be confirmed through testing in Section 9.

Key Requirements

1. The rocket shall return to a specified target.
2. The rocket shall return to the designated area safely.
3. The rocket shall be reusable.

Function	Design Requirement	Specification	Analysis, Testing, Inspection, Demonstration
Rocket shall return to a specified target	Distance from target.	D <= 800 ft	Testing
	Distance of target from launch site.	D <= 490 ft	Inspection
Rocket shall reach apogee	Apogee height	H >= 2200 ft	Testing
Rocket shall land safely	Impact velocity	V <=25 ft/s	Testing
Rocket shall have a two-stage recovery system	Streamer 1 deployment height	H = 2200 ft	Testing
	Streamer 2 and paraglider deployment height	~500-700 feet	Testing
	Main chute descent velocity	V <= 25 ft/s	Testing

Table 4: Key System Function Limits

Key Components

1. Guidance System
2. Recovery System
3. Ground Station

Component	Sub-Component	Design Requirement	Specification	Analysis, Testing, Inspection
1. Guidance system	1. GPS	1. Accuracy 2. Power 3. Output 4. Dimensions 5. Weight	1. 24 - 100ft 2. 3-5 V 3. UART 4. 3 - 5in x 1in x 0.5 in 5. ~0.6 oz	1. Demonstration 2. Demonstration 3. Demonstration 4. Demonstration 5. Demonstration

	2. Altimeter	1. Accuracy 2. Power 3. Output 4. Dimensions 5. Weight	1. 24 - 100ft 2. 3-5 V 3. I2C 4. 3 - 5in x 1in x 0.5 in 5. 0.07-0.6 oz	1. Demonstration 2. Demonstration 3. Demonstration 4. Demonstration 5. Demonstration
	3. Compass	1. Accuracy 2. Power 3. Output 4. Dimensions 5. Weight	1. 2 deg 2. 3.3-5v 3. UART 4. 0.5x0.5x0.16 in 5. 0.3-0.4 oz	1. Demonstration 2. Demonstration 3. Demonstration 4. Demonstration 5. Demonstration
	4. Flight and Guidance Computer	1. Power 2. Calculation speed 3. Memory storage 4. Dimensions 5. Weight 6. Connections	1. 3-5v 2. 240 MHZ 3. 4MB 4. 2in x 2in 5. ~0.2oz 6. All forms of SPI, and analog.	1. Demonstration 2. Demonstration 3. Demonstration 4. Demonstration 5. Demonstration 6. Demonstration
	5. Radio Transmitter	1. Frequency 2. Power requirement 3. Range 4. Dimensions 5. Weight 6. Connection	1. 915 MHz 2. 3-5v 1.2-2a 3. 1-6 miles 4. 0.25in x 0.05in 5. 0.1 - 0.2 oz 6. UART	1. Demonstration 2. Demonstration 3. Testing 4. Demonstration 5. Demonstration 6. Demonstration
	6. Servo	1. Accuracy of movement 2. Produce x force 3. Dimensions 4. Input voltage 5. Weight 6. Connection	1. 4 deg 2. 0.3 - 0.8 oz 3. 1-1.25 in x 0.5-0.8 in 4. 5v - 6v 5. 2.4 oz 6. PWM	1. Demonstration 2. Testing 3. Demonstration 4. Demonstration 5. Demonstration 6. Demonstration
	7. Battery	1. Voltage 2. Dimensions 3. Weight 4. mAh 5. Time of operation	1. 3-9V 2. .7in x 1.91in x 1.05in 3. 1.2 oz 4. 2000-4000s	1. Demonstration 2. Demonstration 3. Demonstration 4. Demonstration 5. Testing

2. Recovery system	1. Screamer	1. Weight 2. Time 3. Dimensions	1. 0.3-1 oz 2. 2 hours 3. 3.5in x 1.125in	1. Demonstration 2. Demonstration 3. Demonstration
	2. Streamer One	1. Withstand x force 2. Drag force 3. Dimensions 4. Material 5. Weight	1. 3-5 lb 2. 0.5-0.1 lb 3. 8-10 in 4. Nylon 5. 2-3 oz	1. Testing 2. Testing 3. Demonstration 4. Demonstration 5. Demonstration
	3. Streamer Two	1. Withstand x force 2. Drag force 3. Dimensions 4. Material Weight	1. 3-5 lb 2. 0.5-0.1 lb 3. 8-10 in 4. Nylon 5. 2-3 oz	1. Testing 2. Testing 3. Demonstration 4. Demonstration 5. Demonstration
	4. Paraglider	1. Withstand x force 2. Drag force 3. Wing Span 4. Material 5. Weight 6. Glide Ratio	1. 30 lbs 2. 6 lbs 3. 60 in 4. Ripstop Nylon 5. 2 lbs 6. 10:1	1. Testing 2. Testing 3. Testing 4. Testing 5. Testing
3. Ground Station	1. Radio receiver	1. Frequency 2. Power requirement 3. Range 4. Dimensions 5. Weight 6. Connection	1. 915 MHz 2. 3-5v 1.2-2a 3. 1-6 miles 4. 0.25in x 0.05in 5. 0.1 - 0.2 oz 6. UART	1. Demonstration 2. Demonstration 3. Demonstration 4. Demonstration 5. Demonstration 6. Demonstration
	2. USB to UART Dongle	1. Power 2. Connection	1. 3-5V 2. USB to UART	1. Demonstration 2. Demonstration

Table 5: Key Components

6. System Concept Development

6.1. Methodology

System concepts were generated by the team using the currently available technology in the field of rocketry. The RTLS system, which is the focus of this project, will be a new system inspired by relevant technology in rocketry and other areas. The team eliminated a large portion of design options through engineering intuition and project constraints. For the rest, extensive trade studies were conducted to determine the best concept for components without obvious solutions.

In order to appropriately identify all rocket components, sizes, weights, and specifications, a specific sequence is needed. The determination process is as follows: all necessary software drives the criteria and choices for hardware with respect to compatibility. The hardware dimensions and sizes then decided the necessary length and material for the body of the rocket. This, in turn, determined the aerodynamics of the rocket, including the drag and the size of the fins. Ultimately, these factors influenced our choice for motor selection, including the motor class, thrust, and the required impulse.

6.2. Overall System Concept Selection

Before the paraglider was selected to control the RTLS system, several other design concepts were considered. These included rolleron fins, a motorized main parachute, and a control line placed at the landing target that can pull the rocket toward the ground.

Paragliders are often used for controlled, directed descents in recreational activities. A paraglider system can be thought of as a glider as it moves forward in one direction. Applying this principle as an alternative to a parachute enables a controlled descent with a slower velocity due to the lift generated by the paraglider. Unlike a parachute, which primarily creates drag, a paraglider produces lift. In this concept, the paraglider would be connected to two points on the rocket and controlled by servos to direct its descent. The two control wires from the paraglider would run to the electronics bay, which houses the servos. These servos would be guided by the flight computer to manage and adjust the rocket's descent path. The benefits of using this system are that the user can specify an exact landing position. The slower decent speed provides a softer landing, protecting the sensitive internal components. The challenges of using this system include that paraglider wing control cords can easily tangle together cause the glider to not fully deploy. The nose cone shock cord attachment location would need to be moved below the control system of the wing.

Overall, the paraglider is the superior design choice for several reasons. Paraglider technology is well established, being used in areas such as recreational flight and military air deliveries. A paraglider system could also deploy in a similar fashion to traditional dual deployment high powered rockets, replacing the main parachute. This would avoid a lot of uncertainty that would come with designing an entirely new recovery system configuration.

Lastly, the paraglider would allow the rocket to descend slowly. This would allow electronics time to make necessary adjustments to guide the rocket to its target.

6.3. Subsystem Breakdown

The rocket was broken down into 5 major subsystems that contain all major components: software, hardware, airframe, propulsion system, and recovery system. These are listed in Table 6. The choices for each component will be provided with brief explanations. More detailed explanations can be found in Section 5 of M3. It is worth noting that a few design changes have occurred since M3 was written. These changes are present in this section, and all changes are summarized in Section 10.

Subsystem	**Components**
1. Software	- PWM Servo Control - MATLAB-based PD Flight Algorithm - Target Heading Calculation - 1D Kalman Filter for Yaw Estimation - Adaptive Gain Tuning - PD Controller - Sensor & Motion Model - Wind Disturbance Modeling - Data Logging to CSV - Monte Carlo Analysis for Validation

Subsystem	Components
2. Hardware	- ESP32 Microcontroller - u-blox NEO-6M GPS Module - Adafruit BMP388 Barometric Sensor - BerryIMUv3 (LIS3MDL) Magnetometer - Micro SD Card Module - LoRa Radio Module
3. Airframe	- 3D-Printed PETG Rocket Body Tubes - Ogive PETG Nose Cone - Clipped Delta PETG Fins - Surface-Mounted Fin Attachments - Apogee 1515 Rail Buttons - PETG Bulkheads - PETG Coupler
4. Propulsion	- Aerotech I285R-A14 Rocket Motor - 3D-Printed PETG Motor Mount Assembly
5. Recovery	- Synapse 170 Paraglider with Servo-Actuated Brake Lines - Ripstop Nylon Streamer with Kevlar Blanket - Electronically Activated Triple Seven Black Powder Charge - High-Strength Kevlar Shock Cord

Table 6: Subsystem component lists

6.3.1. Software

Servo Control

Design Selection: PWM signal generation

Reasoning:

The ESP32 controls the paraglider's directional lines through a PWM-driven servo. The control logic applies a heading correction algorithm that actuates left or right turns based on angular deviation from the target bearing.

Flight Algorithm

Design Selection: MATLAB based PD flight algorithm with adaptive gain tuning, wind modeling, and trajectory simulation

Reasoning:

The flight algorithm simulates post-apogee guidance for the RTLS rocket. It calculates yaw error between current heading and the bearing to a fixed GPS target. An adaptive PD controller adjusts servo commands to steer the paraglider accordingly. The algorithm accounts for altitude, yaw, velocity, and external disturbances such as wind, enabling repeatable and tunable return simulations prior to embedded implementation on the ESP32.

Target Heading Calculation

Design Selection: using Euclidean distance and bearing computation

Reasoning:

Determines the heading from current position to target using Cartesian coordinates. This ensures precise directional awareness for course correction during flight. The bearing output serves as the target yaw for PD correction.

Yaw Estimation

Design Selection: implementing a 1D Kalman filter for yaw state refinement

Reasoning:

Filters noisy yaw measurements using adaptive Kalman gain. This prevents instability in servo commands due to sensor fluctuations and ensures smooth heading estimation. Filter stability is enforced by bounding NaN or Inf values.

Adaptive Gain Tuning

Design Selection: Distance-based PD gain scaling

Reasoning:

Uses proximity to target to dynamically tune PD gains. Higher gains are applied when far from the target to enable aggressive corrections, and lower gains are used when close to minimize overshoot, ensuring smooth and precise landings.

PD Controller

Design Selection: PD controller for control of yaw error

Reasoning:

Computes servo command based on current and previous yaw error. The control output is clamped within ±15° to reflect realistic servo constraints. This approach enables responsive yet bounded control input to the paraglider's brake lines.

Sensor and Motion Model

Design Selection: Simulates yaw dynamics and positional updates

Reasoning:

Models motion by integrating yaw-based heading over time with added noise for realism. The function also updates vertical descent, simulating barometric altitude. This mimics the behavior expected from physical sensors and motion response.

Wind Disturbance Modeling

Design Selection: Stochastic wind speed and direction modeled directly in main script

Reasoning:

Random wind disturbances are applied at each timestep to test controller robustness. Wind vectors influence lateral drift and test the algorithm's ability to maintain an accurate return path under uncertain conditions.

Data Logging

Design Selection: Outputting flight telemetry to a CSV file

Reasoning:

Records time-stamped yaw, altitude, servo commands, and position data for each timestep. This enables performance review and supports further tuning and debugging through post-simulation analysis.

Validation of System

Design Selection: Monte Carlo Analysis

Reasoning:

Runs 100+ randomized trials to assess flight algorithm robustness across various initial conditions. Final distances from the target are analyzed to verify performance consistency and validate control logic under realistic variability.

6.3.2. Hardware

Microcontroller

Design Selection: ESP32 WROOM

Reasoning:

The ESP32 was selected for its dual-core processing capability, allowing concurrent execution of sensor polling and control algorithms. It offers native support for I2C, SPI, UART,

and PWM interfaces, all essential for integrating the rocket's sensors, radio, SD card, and servo motor. Its compact footprint, built-in Wi-Fi/Bluetooth (for potential future use), and low power consumption make it ideal for embedded aerospace applications.

GPS

Design Selection: u-blox NEO-6M GPS module via UART (Serial 2)

Reasoning:

The GPS provides real-time position, velocity, and time data with an accuracy of ~2.5 meters. This serves as the primary navigation reference for RTLS descent logic. Data is parsed using the TinyGPS++ library for latitude, longitude, satellite count, and HDOP values.

Altitude Estimation

Design Selection: Adafruit BMP388 barometric pressure sensor via SPI

Reasoning:

The BMP388 sensor is used to determine the rocket's altitude AGL, enabling logic transitions such as recovery deployment and servo activation thresholds. Its SPI interface ensures fast and reliable data acquisition under flight conditions.

Orientation Estimation

Design Selection: BerryIMUv3 magnetometer (LIS3MDL) via I2C

Reasoning:

The LIS3MDL provides compass-based heading data, which is used to determine the direction to the target landing zone. This heading is continuously compared with the desired bearing to compute correction signals for steering.

Data Logging

Design Selection: Micro SD card module via SPI

Reasoning:

The ESP32 logs flight data to an SD card with files named using GPS-provided timestamps. Data includes GPS position, altitude, heading, and control outputs for post-flight analysis.

Telemetry Transmission

Design Selection: LoRa-based radio module via UART (Serial1)

Reasoning:

The system transmits downlinked telemetry to a ground station for live tracking. The packets include GPS coordinates, AGL altitude, magnetic heading, and other flight state indicators. This enables the team to monitor rocket performance in real time.

6.3.3. Airframe

Rocket Body Tubes

Design Selection: PETG body tubes

Reasoning:

PETG is strong, impact resistant, and heat resistant. It also has advantages in manufacturing other materials that share its properties such as low warping, open-bed printing, low toxicity, and lack of abrasion on printer nozzles. PETG is slightly heavier than cardboard, but cheaper than fiberglass. Fiberglass has a higher strength to weight ratio but was not selected due to its high cost. PETG offers the advantages of high strength and customizability at a price point within the project budget.

Nose Cone

Design Selection: Ogive nose cone

Material: 3D-printed PETG

Reasoning:

At low speeds, the exact profile of a rocket nose cone does not significantly affect drag. The most important characteristic of a nose cone is how it affects the location of the center of gravity and center of pressure (for stability) on a rocket. Therefore, the choice of profile is arbitrary and highly based around aesthetics for low-speed rockets. For this reason, the team opted to use the ogive profile as a reference to the film, *The Dictator*.

Fins

Design Selection: Clipped delta

Material Selection: 3D-printed PETG

Reasoning:

As for nose cones, drag characteristics of fins are not as significant at low speeds. Fins are mainly important for flight stability. The clipped delta shape is easy to manufacture and incorporate airfoiled edges into the design. The surface area of these fins will be the most important aspect of their design, as this significantly affects the location of the center of pressure.

Fin Attachment Mechanism

Design Selection: Surface mounting

Reasoning:

The fins were chosen to be surface mounted as opposed to through-the-wall to save weight. The extra rigidity was not found to be necessary. Printing the fins onto the body tube instead of using an adhesive is significantly stronger.

Guide Buttons:

Design Selection: Apogee 1515 Rail Buttons

Reasoning:

These rail buttons were chosen to provide stability during takeoff. These were found to be sufficient for a heavy rocket, where extra support is needed.

Bulkheads:

Material Selection: 3D-printed PETG

Reasoning:

Bulkheads are used to separate individual bays in a rocket and provide rigidity to body tubes. PETG was chosen as the material for the bulkheads because of its high bending strength. Most bulkheads are subject to bending loads due to snatch forces seen during deployment. 3D printing also allows the bulkheads to be tightly fit into the body tubes, preventing leaks of ejection charges into other rocket bays.

Coupler

Material Selection: 3D printed PETG

Reasoning:

The coupler is designed to connect the upper and lower body tubes while serving as a separator, allowing each tube to store different mission-critical components. PETG was selected as the material for the coupler because of its high rigidity. 3D printing this component allows it to be tightly friction fitted and for ejection charges to be incorporated into the design.

6.3.4. Propulsion System

Rocket Motor

Design Selection: Commercial off-the-shelf Aerotech I285R-A14 rocket motor

Reasoning:

The Aerotech I285R-A14 motor was selected due to its performance characteristics

closely aligning with the project's specific mission requirements. This motor provides a total impulse of approximately 635 Newton-seconds (Ns), classifying it comfortably within the Level 1 certification limits. Its peak thrust of 436.7 N and an average thrust of 285 N effectively meet the thrust-to-weight ratio necessary for stable and reliable ascent past the targeted apogee of 2,200 feet AGL. The motor's total burn time of roughly 2.23 seconds supports controlled acceleration and predictable flight trajectory, crucial for accurate trajectory simulations conducted with OpenRocket software.

Motor Mount

Design Selection: PETG motor mount assembly

Reasoning:

The motor mount assembly, including motor tube and structural stiffeners, is manufactured using PETG. PETG was selected due to its excellent strength, impact resistance, and thermal resistance, which is essential for withstanding both mechanical stresses and heat generated by the motor. PETG’s ease of printing enables precision in producing complex geometries such as integrated fin slots and structural stiffeners without requiring additional machining or tooling. Alternative materials, like fiberglass, were considered due to their superior strength-to-weight ratio; however, they were not selected because of significantly higher costs and manufacturing complexity.

6.3.5. Recovery System

Paraglider

Design Selection: Synapse 170 airfoil profile with ripstop nylon canopy and servo-actuated brake lines

Reasoning:

The Synapse 170 paraglider was selected due to its airfoil profile. The canopy is constructed from ripstop nylon, a lightweight and durable material resistant to tearing, making it ideal for repeated deployments and exposure to atmospheric drag forces. Control is achieved through servo-actuated brake lines connected to the paraglider's control lines, enabling autonomous directional adjustment during descent. This system allows for targeted landings and recovery, improving post-flight data collection and vehicle reuse. Other parachute configurations were considered; however, they lacked the maneuverability and precision offered by a paraglider setup.

Streamer

Design Selection: Ripstop nylon streamer with Kevlar protective blanket

Reasoning:

A ripstop nylon streamer was chosen over a conventional drogue parachute to minimize the risk of tangling with the main recovery system, particularly the paraglider. The streamer provides sufficient drag to decelerate the rocket while maintaining a straight trajectory, facilitating clean separation and deployment of the main recovery system. Ripstop nylon offers a balance of low weight and high tear resistance. Additionally, a Kevlar-Nomex blanket is used to shield the streamer from the black powder ejection charge, ensuring safety during deployment. A

drogue chute was considered but eliminated due to increased complexity and potential interference with the paraglider.

Black Powder Charge

Design Selection: Electronically activated Hodgdon Triple Seven FFFG granular powder charge

Reasoning:

The deployment system uses Hodgdon Triple Seven FFFG black powder, which is a sulfur-free, high-energy propellant. It is electronically activated for precise timing and reliable ejection of recovery components. The granular form ensures consistent burn characteristics and pressure buildup inside the deployment bay. Triple Seven was chosen for its clean burn and reduced residue, which aids in maintaining the performance of nearby mechanical components such as servos and linkages. Other powders were evaluated but found to produce more residue or require more stringent handling protocols.

Shock Cord

Design Selection: High-strength Kevlar shock cord

Reasoning:

Kevlar was selected for the shock cord due to its exceptional tensile strength, thermal resistance, and minimal elasticity. This makes it highly effective in withstanding parachute snatch forces during deployment, particularly when transitioning from high-velocity descent to parachute inflation. Its resistance to abrasion and heat ensures durability over multiple flight cycles. Alternative materials like nylon were not used due to their higher elasticity and potential to rebound or fail under extreme loads.

7. Design Analysis

7.1. Key System Parameters Analysis

This section presents a comprehensive overview of the Computational Fluid Dynamics (CFD), Finite Element Analysis (FEA), and OpenRocket flight simulations conducted for the vehicle. These simulations were essential for estimating critical performance parameters that are either impractical or highly complex to compute analytically, including flight velocity profiles, apogee prediction, aerodynamic drag, and deployment loads. Each simulation type played a distinct role in evaluating the vehicle's aerodynamic behavior, structural integrity, and flight trajectory.

For each model described herein, the simulation methodology, input assumptions and boundary conditions, and key results will be detailed. This includes descriptions of the software environments, mesh resolutions, material properties, and physical models employed, as well as validation steps where applicable. The results of these simulations directly informed design decisions, ensuring the vehicle meets performance, safety, and reliability requirements. The final design will be presented in Section 8.

7.2. Flight Simulations (OpenRocket)

7.2.1. Obtaining Motor Impulse Range

Accurately determining the total impulse range is crucial for meeting performance and safety criteria. Software such as OpenRocket simulates the dynamic behavior of rocket motors by accounting for variable burn rates, aerodynamic drag, and atmospheric conditions. This method provides a comprehensive total impulse value and serves as the primary tool for this analysis.

To determine the optimal impulse for the rocket, a range of I-class motors was selected arbitrarily, each with different impulse values. This approach allowed testing of various impulse levels to see which would yield the best performance in terms of apogee and overall flight stability. The chosen motors provided valuable insights into how different impulse outputs affect the rocket's performance. The rocket's calculated weight of 1888 g (excluding the motor) was factored into the analysis to ensure that the thrust-to-weight ratio remained sufficient for a stable launch. Incorporating this weight consideration into the simulations helped more accurately assess how each motor's impulse would overcome gravitational forces and contribute to the desired apogee.

Motor	Velocity of Rod	Apogee	Optimum Delay	Max. Velocity
I285R-14	71.3ft/s	2523 ft	10.4 s	513 ft/s
I345-15A-15	78.6ft/s	2464 ft	10.4 s	525 ft/s
I305FJ-14	75.2ft/s	2652 ft	10.4 s	526 ft/s

Table 7: Flight Simulation Data 1 for Tested Motors via OpenRocket

Motor	Max. acceleration	Time to apogee	Flight time	Ground Hit
I285R-14	15.3 G	11.9 s	82.9 s	14.1 ft/s
I345-15A-15	16.8 G	11.6 s	174 s	15.4 ft/s
I305FJ-14	16.2 G	12 s	187 s	15.4 ft/s

Table 8: Flight Simulation Data 2 for Tested motors via OpenRocket

From the data in Table 7, all three motors achieved above the 2200 ft minimum with a 10% safety factor (2420 ft), demonstrating that an impulse range of 400 Ns to 450 Ns would be effective. Motors with impulses below 400 Ns risk underperformance, while impulses above 450 Ns could be deemed unnecessary, as the goal is to reach over 2200 ft apogee. Table 8 provides more important performance data on the motor candidates. For example, acceleration is important for determining the forces experienced by the airframe during flight. Flight time is also important, as shorter flight times help prevent drift from wind

Motor Selection: I285R

Despite the superior apogee of the I305FJ-14, its configuration would have placed it too close to the electronics bay (Figure 23). risking exposure of sensitive components to excessive heat and vibration. The shock cord will also be tied to the bottom of the electronics bay, so there needs to be sufficient space for the configuration. In addition, this motor comes with a delay charge that would need to be removed. The I285R, on the other hand, provides a balanced total impulse within the target range while ensuring safe clearance from the electronics bay. Thus, the I285R was chosen as it meets both the performance requirements and safety constraints for the model rocket design.

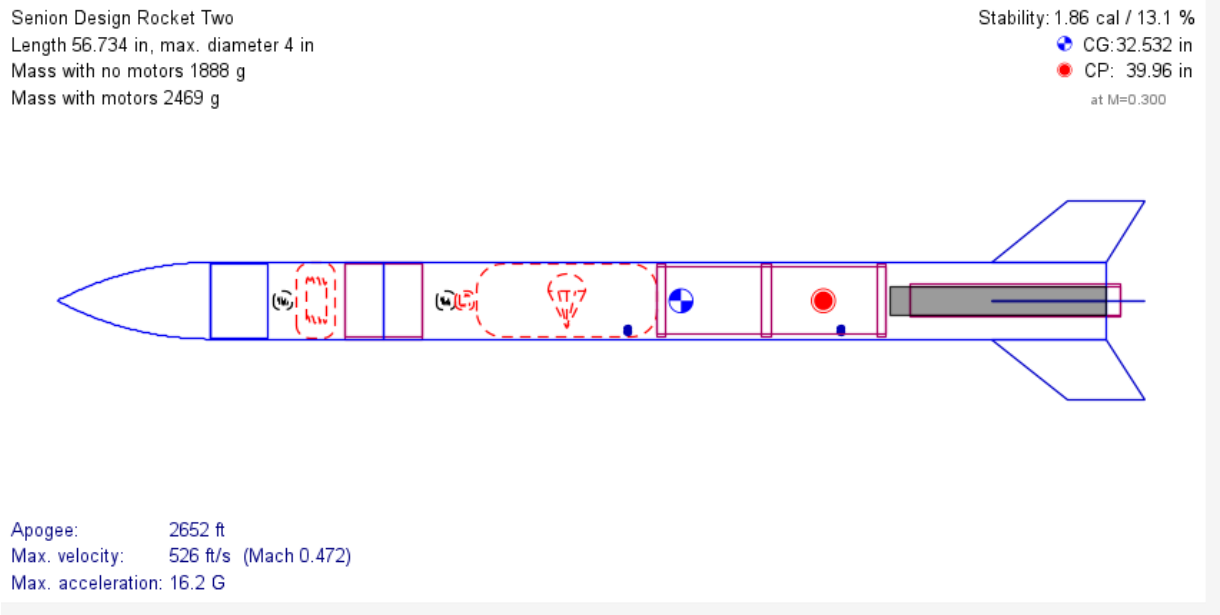


Figure 23: Configuration of rocket with the I305FJ-14

7.2.2. Extended Flight Simulation data with I285R

	Warnings	Name	Configuration	Velocity off rod	Apogee	Velocity at deploy...	Optimum delay	Max. velocity	Max. acceleration	Time to apogee	Flight time	Ground hit velocity
✔	1 ⚠	Simulation 1 (0 mph)	[I285R-14]	71.3 ft/s	2520 ft	72.5 ft/s	10.7 s	513 ft/s	15.3 G	12.2 s	83.1 s	14.1 ft/s
✔	1 ⚠	Simulation 2 (15 mph)	[I285R-14]	71.3 ft/s	2478 ft	75.6 ft/s	10.2 s	512 ft/s	15.3 G	11.7 s	82.9 s	22.2 ft/s
✔	1 ⚠	Simulation 3 (20 mph)	[I285R-14]	71.3 ft/s	2441 ft	79.4 ft/s	10.1 s	510 ft/s	15.3 G	11.6 s	82.4 s	30.7 ft/s

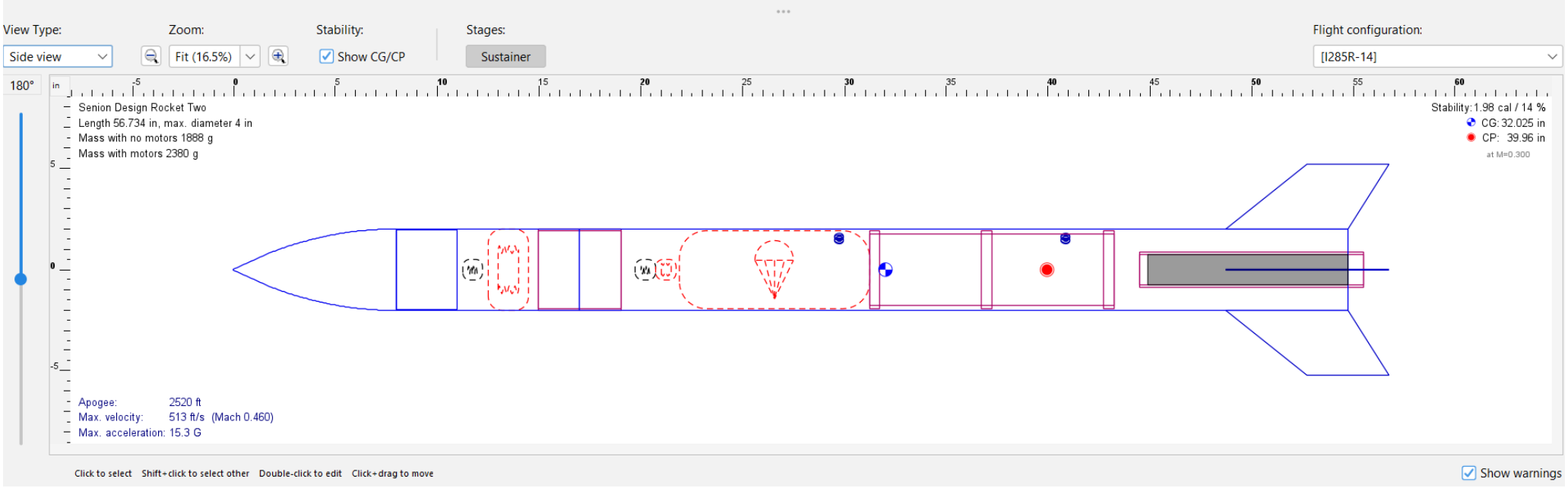


Figure 24: Data of Rocket flight with the rocket configuration via OpenRocket

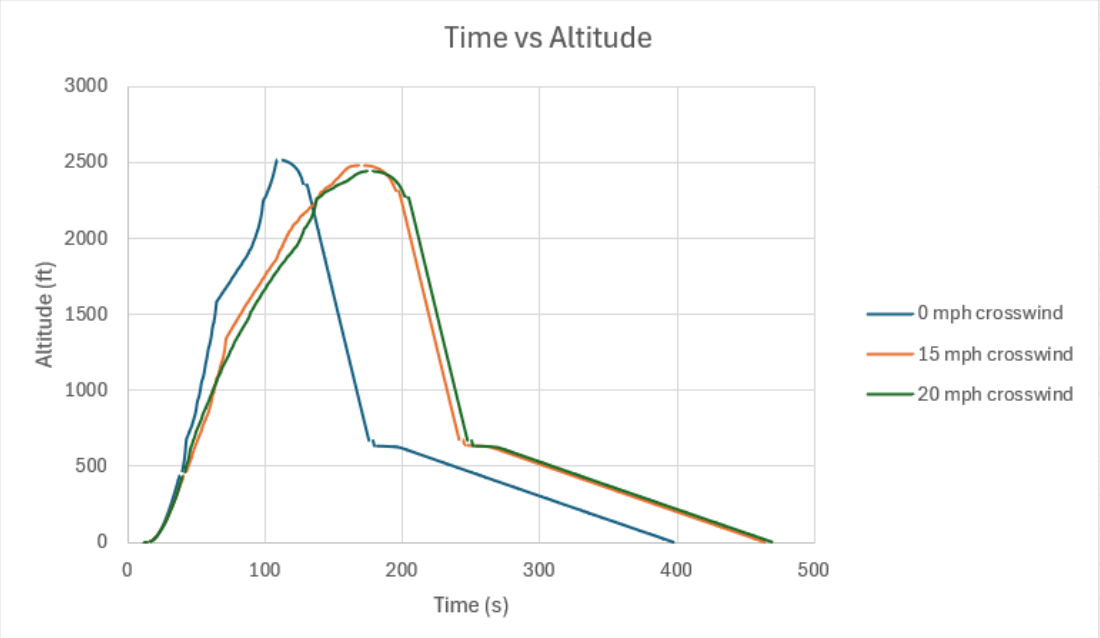


Figure 25: Time vs Altitude (via OpenRocket)

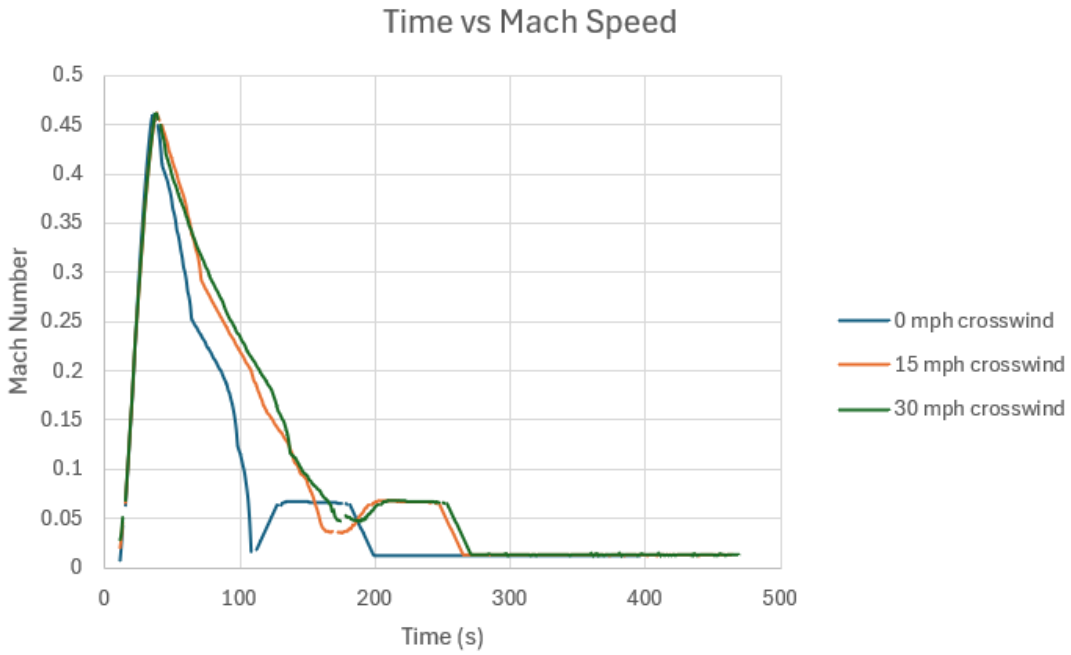
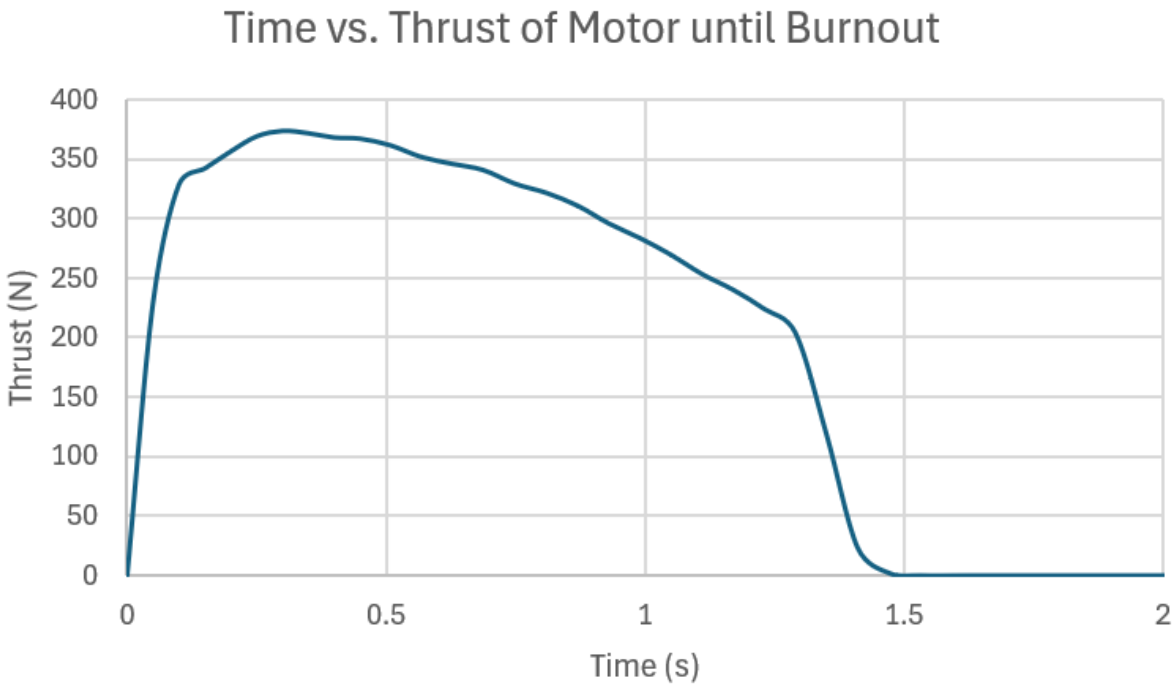


Figure 26: Time vs. Mach number



Event	Time (s)
Launch	0
Liftoff	0.05
Off Rail	0.2
Burnout	1.53
Apogee	12.217
Streamer 1	12.218
Streamer 2	38.033
Paraglider	38.534
Hit Ground	83.133

Figure 27: Simulated Time History of Launch Events for 0 mph Crosswinds (via OpenRocket)

Event	Time (s)
Launch	0
Liftoff	0.05
Off Rail	0.2
Burnout	1.53
Apogee	11.644
Streamer 1	11.645
Streamer 2	36.983
Paraglider	37.484
Hit Ground	82.356

Figure 28: Simulated Time History of Launch Events for 30 mph Crosswinds (via OpenRocket)

The OpenRocket simulation predicts that the rocket will reach an apogee of 2520 feet in 0 mph crosswinds, while in 30 mph crosswinds, the apogee decreases slightly to 2441 feet. Both values exceed the minimum required altitude of 2200 feet, confirming that the rocket meets the necessary performance criteria for L1 certification. The simulation also provides insights into the descent and landing speeds, showing that the rocket is expected to land at 14.1 ft/s in 0 mph crosswinds and 30.7 ft/s in 30 mph crosswinds. This variation emphasizes the need for precise parafoil guidance to control drift and ensure safe landing.

Regarding the recovery system, the first streamer is expected to deploy approximately 10 milliseconds after the rocket reaches apogee. While this is an extremely precise deployment in the simulation, real-world factors such as ejection charge delays, sensor response times, and aerodynamic inconsistencies may introduce variations. Additionally, higher crosswinds not only reduce apogee but also increase descent velocity, which could impact both the parafoil deployment and the guidance system's ability to counteract wind drift. Given these factors, real-

world testing will be essential to validate the accuracy of the simulation and refine the recovery system's performance under actual flight conditions.

7.3. FEA

FEA was used to determine the airframe stress during the deployment of the first-stage recovery system. Although simulating airframe stress during paraglider deployment would be beneficial as well, the geometry and configuration is too complicated to model within the scope of this project. Joints will be designed to resist forces significantly stronger than those seen during flight.

7.3.1. Model Setup

The MSC Apex modeler and MSC Nastran solver were used for the FEA simulations. First, the geometry of the model was simplified to a plate inside of a hollow cylinder (Figure 29). The model can be simplified this way because the deployment stresses are concentrated primarily around these areas. This model was created in SolidWorks and imported into the MSC Apex modeler. Next, the geometry was mid-surfaced. Mid-surfacing allows for the creation of 2D elements in an MSC Nastran FEA model. Each of these elements is assigned a property card in a NASTRAN input file, which specifies their thicknesses and material properties. These can be used when stress in one dimension is negligible, which is usually seen in thin-walled structures. Using 2D elements greatly simplifies models where they are applicable.

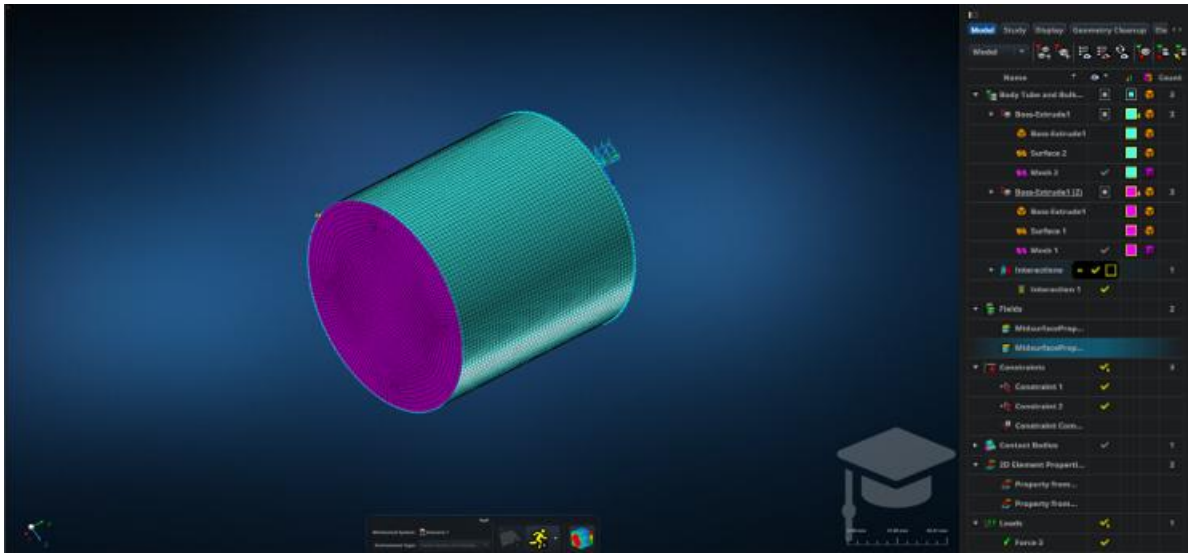


Figure 29: MSC Apex FEA model setup

Next, the model was surface meshed to be moderately coarse. After that, material properties were assigned to the elements. For the simplicity of the model, the behavior of the PETG material was assumed to be linear-elastic and isotropic. The Young's modulus and shear modulus of PETG were found to be 2.4 GPa and 0.85 GPa, respectively. Next, a glued contact was created between the bulkhead and tube to simulate the behavior of epoxy. Finally, loads and constraints were applied to the model. The bottom of the tube was constrained in all six degrees of freedom, and a point load was applied to the center node on the face of the bulkhead to replicate the force concentrated on the eye bolt.

The magnitude of the opening force is difficult to calculate analytically due to a non-inertial frame of reference and varying deceleration rates of the rocket. An accelerometer could be used to experimentally determine the snatch force, but that is out of the scope of this project. Instead, **a factor of safety of 5 was applied to the steady state drag force** (weight) to approximate the snatch force.

$$F_{Snatch} = F_{Drag,SS} * F.S.$$

Equation 2: Snatch force approximation

7.3.2. Results

The following results are from the SOL101 linear static solution type in MSC Nastran. These were analyzed directly in the MSC Apex post-processor.

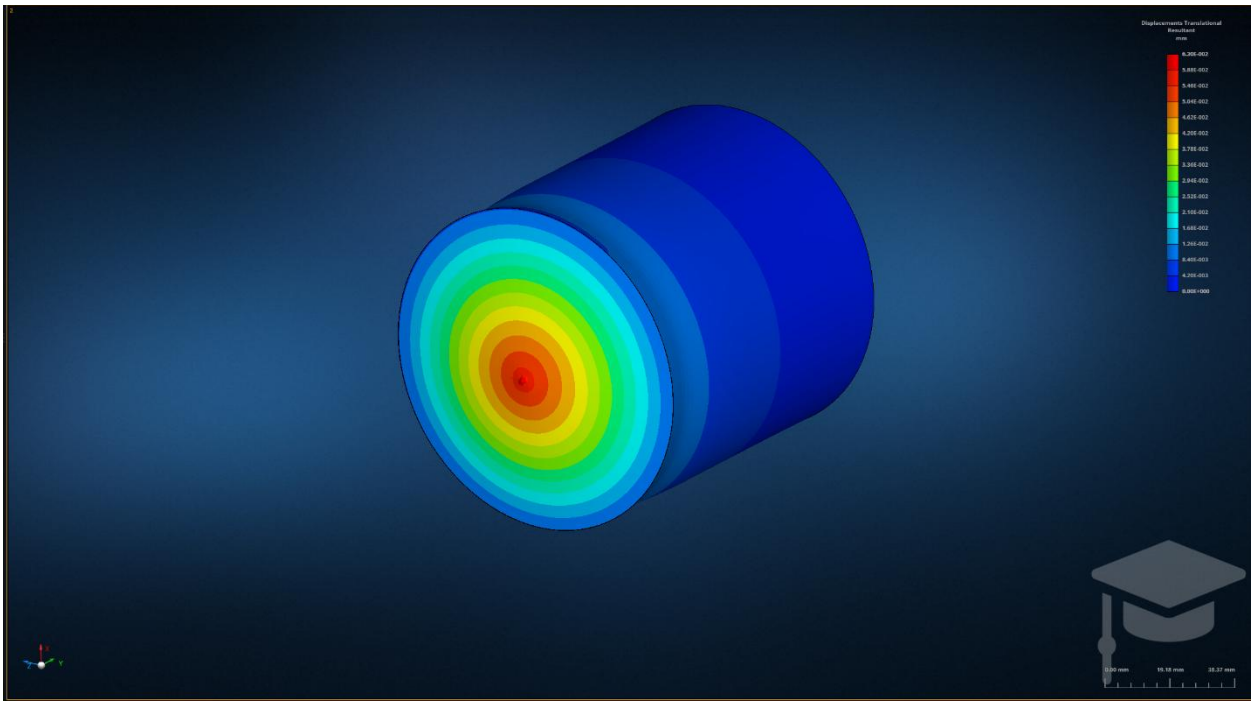


Figure 30: Element displacement contour plot

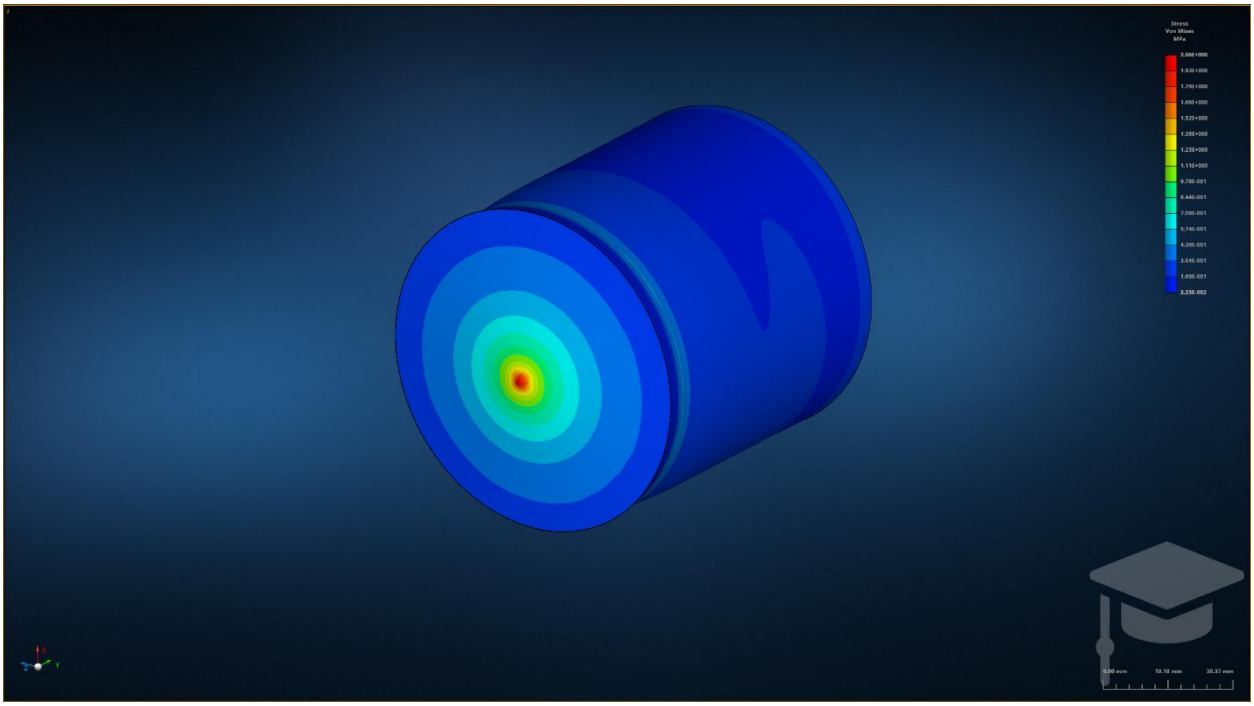


Figure 31: Element Von Mises stresses contour plot

These results show the expected stresses at deployment. Intuition from mechanics of materials theory says that the maximum stress occurs at the edges of the bulkhead because bending stress is at its highest there. However, this does not account for local deformations due to concentrated loads.

7.3.3. Design Considerations

Because of the high stress at the point of attachment for the shock chord, extra thickness was added to the bulkheads to carry the loads. Sturdy eye bolts will be used with thread locker to prevent pull-out failure.

7.4. CFD

CFD was used to calculate the pressure and velocity fields around the rocket body during max-q flight. 2D CFD simulations were used on half of the rocket to save computation time due to symmetry. However, the fins cannot be accurately modeled using 2D simulations. 3D CFD simulations were deemed too complex for the scope of this project. The results gained from a 3D CFD simulation would not critically affect the design of the system. The only potential advantage would be optimizing the fins and nose cone for optimal drag, but these reductions in drag are not aligned with the goals of this project, which are recovery and reusability.

7.4.1. Model Setup

All simulations were calculated using Ansys Fluent with a viscous flow model. The control volume was meshed fine using Ansys Mechanical (Figure 32). The rocket profile was defined as a wall with a no-slip condition. The boundary conditions were a velocity inlet (at max flight velocity) and a pressure outlet (at atmospheric pressure). Simulations converged after 250 iterations.

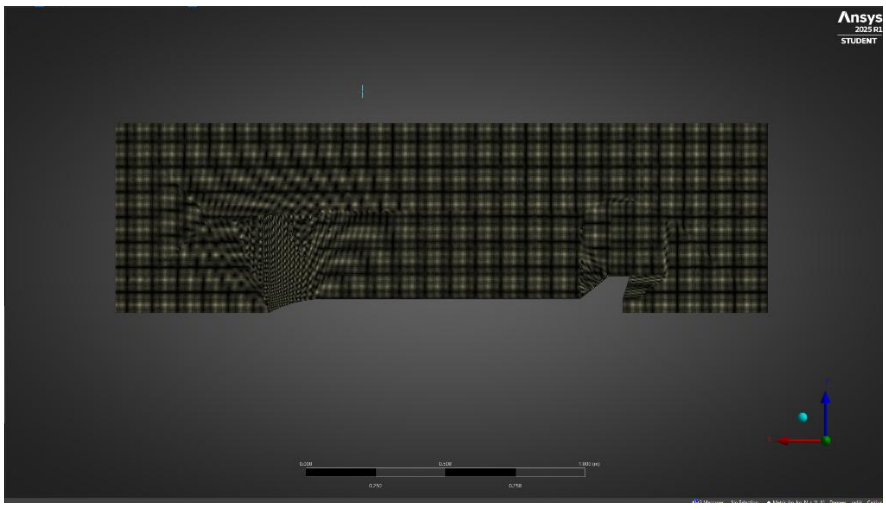


Figure 32: Control volume mesh setup

7.4.2. Results

The results showed that the airframe is optimized for flight. There was no flow separation around the nose, and a short boundary layer present around the body tubes. These simulations confirm that the rocket will perform as expected and provide the team with confidence to proceed with manufacturing for final design.

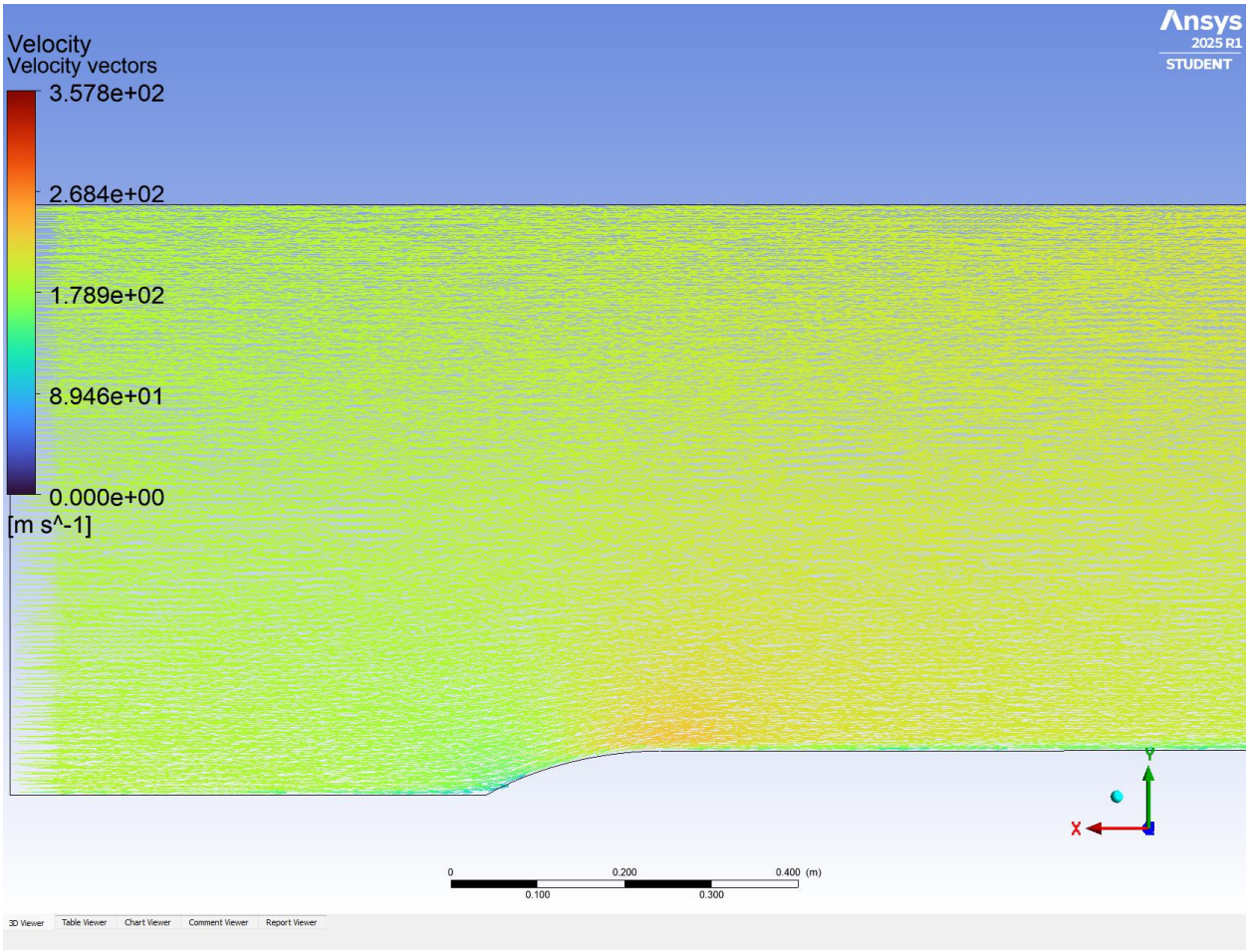


Figure 33: Velocity vectors

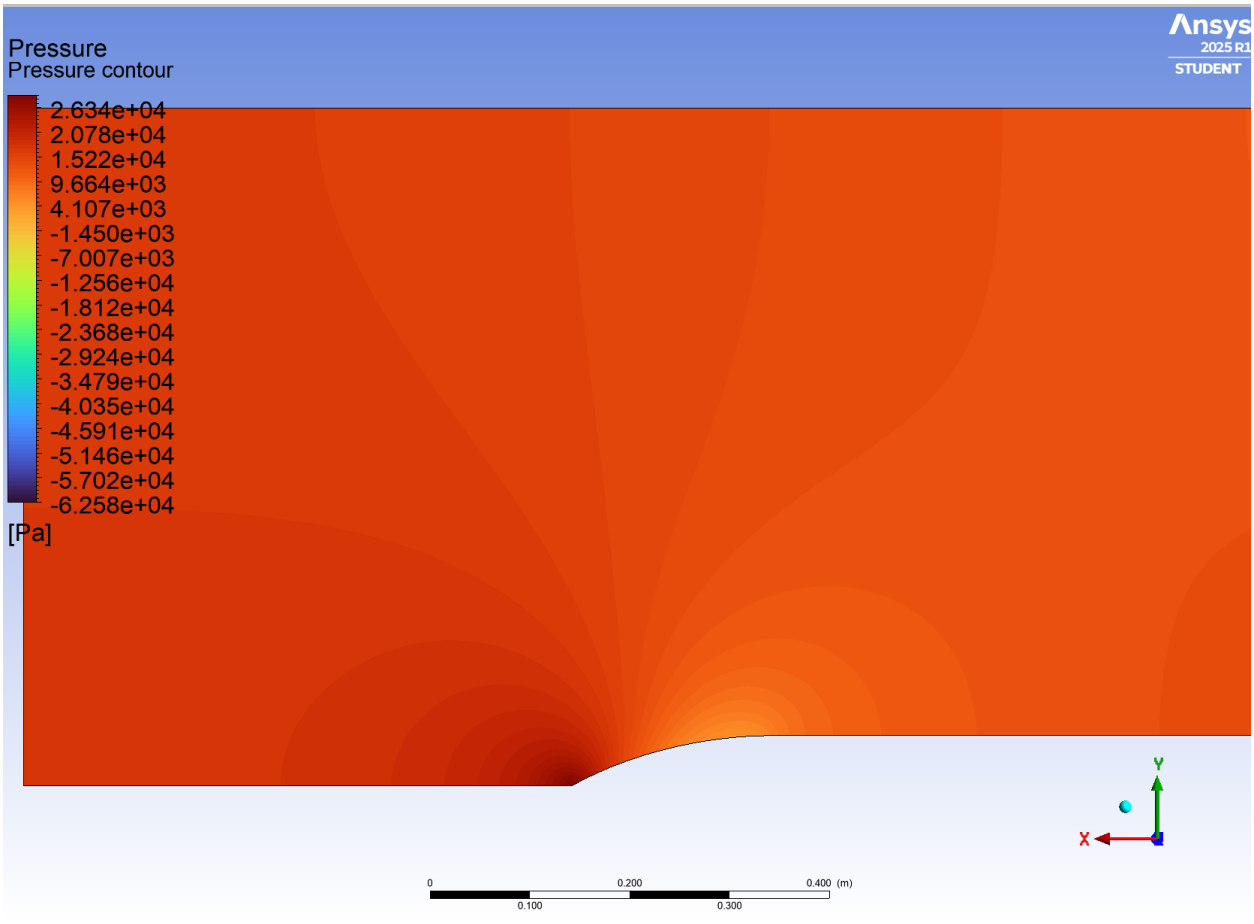


Figure 34: Pressure contours

7.5. Guidance System Simulations

7.5.1. Introduction to RTLS Descent and Guidance Systems

The success of the Return-To-Launch-Site (RTLS) rocket relies on precise descent control and landing accuracy. This section outlines the development of two key models: a MATLAB simulation for analyzing descent behavior under wind disturbances and a Simulink-based guidance system that processes real-time sensor data to adjust the rocket's trajectory. Together, these models enable trajectory correction and controlled landing, ensuring the rocket can autonomously navigate to its target zone.

7.5.2. MATLAB Simulation for RTLS Descent

To evaluate and refine the rocket's descent control strategy, a MATLAB-based simulation (test3D.m) was developed. This simulation models the descent trajectory by incorporating servo-controlled adjustments and wind disturbances while tracking the rocket's positional deviations from a predefined landing target. The simulation initializes the rocket at 700 feet altitude, with a constant descent rate of -5 ft/s, and applies real-time corrections to guide the rocket toward the intended landing site at (X=150ft, Y=50ft).

7.5.3. Trajectory and Control Adjustments

As shown in Figure 35, the simulation tracks the X and Y positions over time, illustrating how the rocket's lateral movement changes throughout the descent. The altitude profile confirms a steady decline, while the servo angle adjustments demonstrate the control system's effort to maintain an optimal landing trajectory. The servo angle initially increases to correct for wind-induced drift, then gradually stabilizes as the rocket approaches the target.

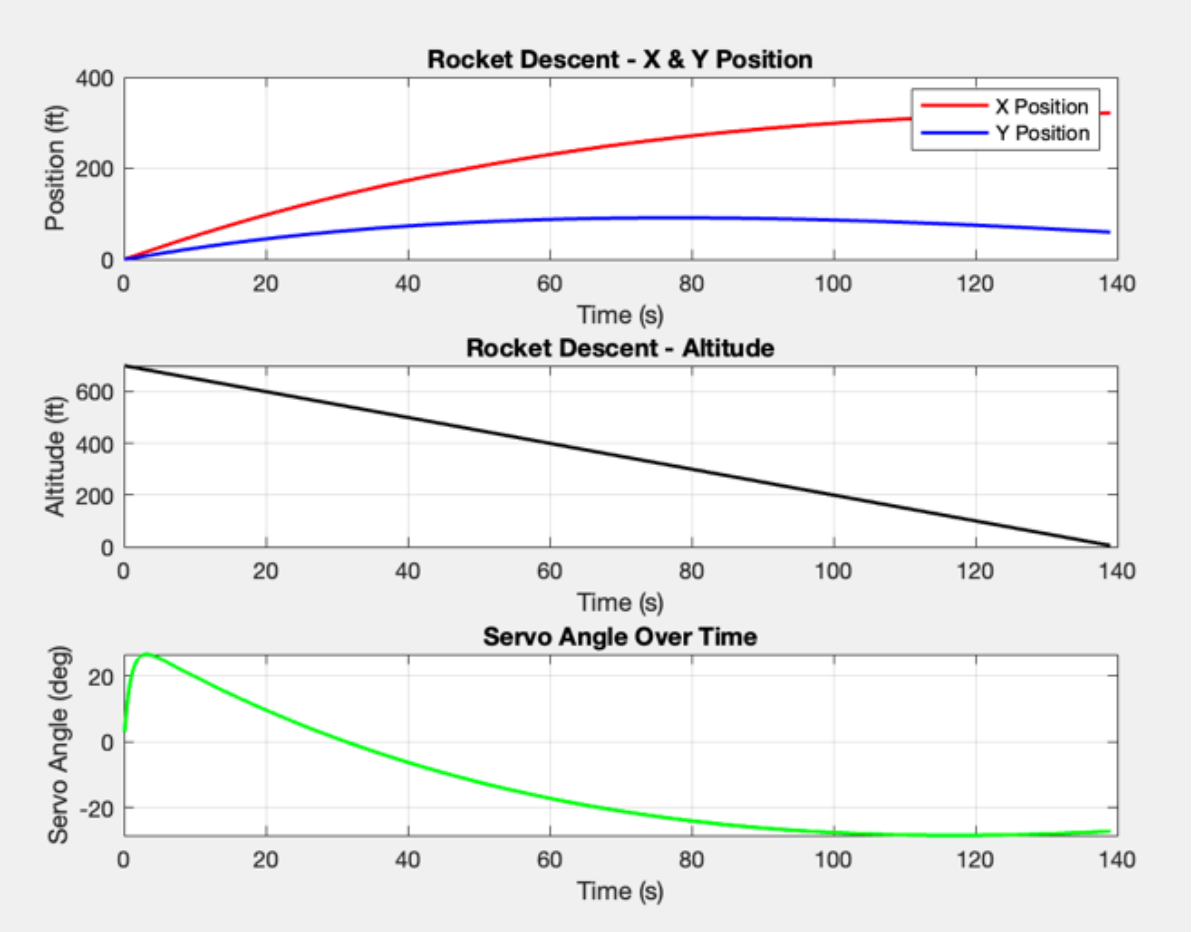


Figure 35: Visualization of Descent 2D

The **3D trajectory visualization** shown in Figure 36 further illustrates the rocket's path from deployment to touchdown. The curved descent profile highlights the influence of wind forces and the effectiveness of servo adjustments in guiding the rocket towards the target. The start and end points are marked, providing a clear visualization of the deviation from the intended landing site.

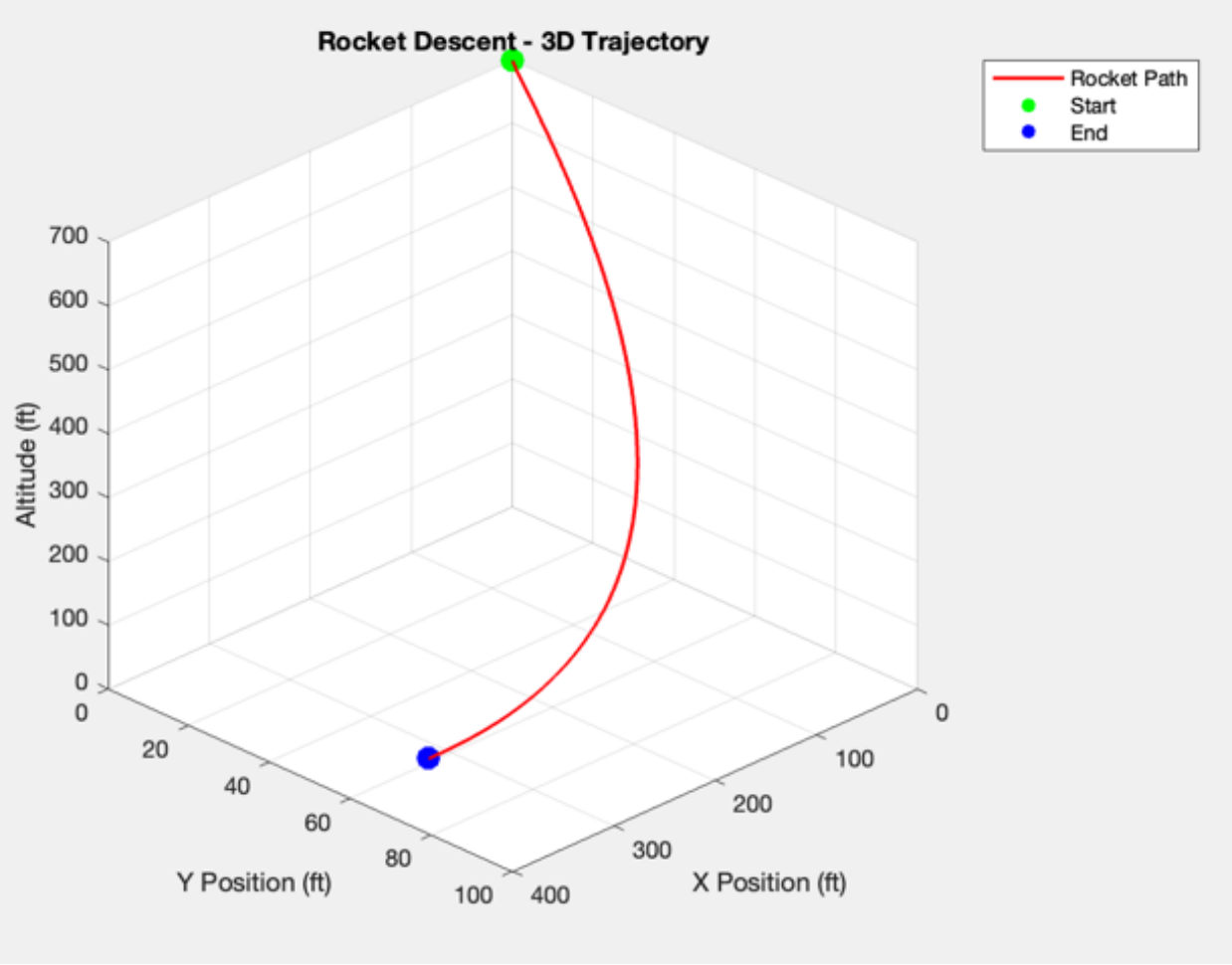


Figure 36: Rocket Trajectory 3D

7.5.4. Data Logging and Landing Accuracy

Throughout the simulation, key flight data—including position, altitude, and servo angles—were recorded for analysis. A sample of the real-time data output is shown in the figure below, displaying incremental updates of the rocket's position and control inputs during descent. The final output in the figure below reports a landing deviation of 171.54 feet from the target, primarily caused by wind effects and control system limitations.

```
Simulation Starting...
Time (s)  |   X (ft)   |   Y (ft)   |  Altitude (ft)  |  Servo Angle (°)
-------------------------------------------------------------------
   0.1    |    0.46    |    0.19    |     699.50      |       3.00
   0.2    |    0.93    |    0.38    |     699.00      |       5.69
   0.3    |    1.41    |    0.58    |     698.50      |       8.10
   0.4    |    1.89    |    0.79    |     698.00      |      10.26
   0.5    |    2.38    |    1.00    |     697.50      |      12.20
   0.6    |    2.87    |    1.23    |     697.00      |      13.93
   0.7    |    3.37    |    1.45    |     696.50      |      15.47
   0.8    |    3.88    |    1.68    |     696.00      |      16.85
   0.9    |    4.39    |    1.92    |     695.50      |      18.08
   1.0    |    4.90    |    2.15    |     695.00      |      19.18
   1.1    |    5.41    |    2.39    |     694.50      |      20.16
   1.2    |    5.93    |    2.64    |     694.00      |      21.02
   1.3    |    6.45    |    2.88    |     693.50      |      21.79
   1.4    |    6.97    |    3.13    |     693.00      |      22.47
```

Figure 37: Initialization of Simulation

```
 139.0    |  321.21    |   59.63    |       5.00      |     -27.24
 139.1    |  321.22    |   59.54    |       4.50      |     -27.23
 139.2    |  321.23    |   59.45    |       4.00      |     -27.22
 139.3    |  321.25    |   59.36    |       3.50      |     -27.20
 139.4    |  321.26    |   59.27    |       3.00      |     -27.19
 139.5    |  321.27    |   59.18    |       2.50      |     -27.18
 139.6    |  321.28    |   59.09    |       2.00      |     -27.17
 139.7    |  321.29    |   59.00    |       1.50      |     -27.16
 139.8    |  321.30    |   58.91    |       1.00      |     -27.15
 139.9    |  321.31    |   58.82    |       0.50      |     -27.14
-------------------------------------------------------------------
Final Position: X=321.31ft, Y=58.82ft, Altitude=0.00ft
Intended Landing: X=150.00ft, Y=50.00ft
Deviation from Target: ΔX=171.31ft, ΔY=8.82ft, Total Deviation=171.54ft
```

Figure 38: Finalization of Simulation

The MATLAB simulation provides a valuable analytical tool for evaluating the RTLS rocket's descent performance. While the servo-controlled adjustments help correct deviations, the final landing accuracy suggests that further tuning of the control parameters, including increased responsiveness to wind disturbances, may be necessary to improve precision. This

simulation serves as a foundation for refining the Simulink-based real-time control model before full-scale physical testing.

The Simulink-based Return-to-Launch-Site (RTLS) Guidance System is a real-time model designed to process sensor data, execute a flight control algorithm, and output servo commands to autonomously guide a descending rocket to a precise landing. This model is structured to replicate the RTLS Control System Diagram exactly, ensuring that the simulation mirrors actual hardware performance. The system integrates multiple sensor inputs, applies real-time data filtering, processes control logic computations, and generates servo control outputs, all within a strict 10ms update cycle to ensure system stability and accuracy.

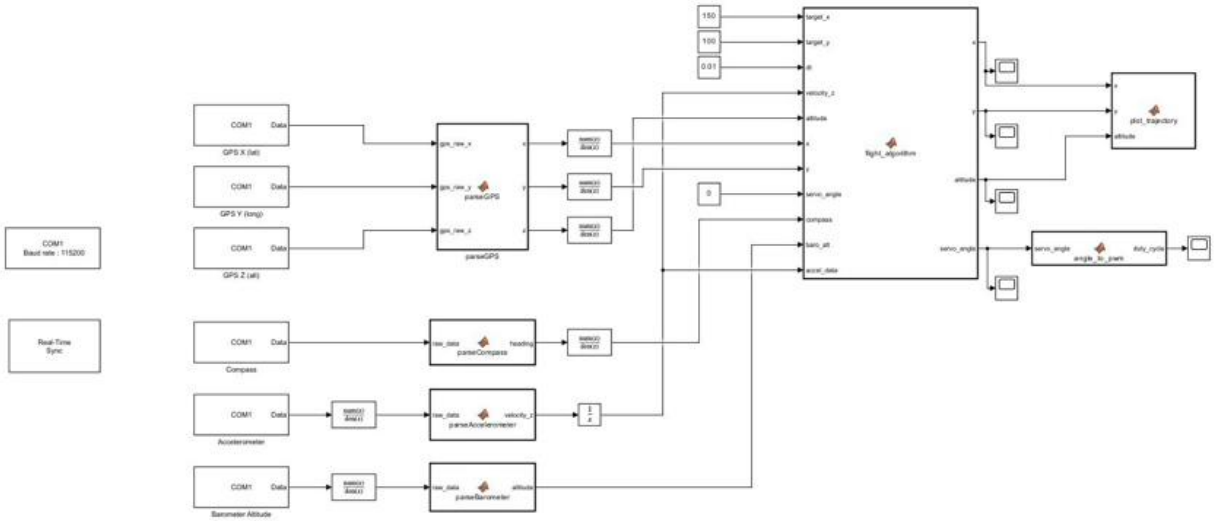


Figure 39: Entire Simulink Block

7.5.5. Sensor Data Acquisition and Processing

At the core of the system lies the sensor data acquisition pipeline, responsible for capturing critical real-time flight parameters such as position, heading, altitude, and acceleration. This information is sourced from multiple sensors: GPS, compass, barometric altimeter, and accelerometer. To ensure accurate control decisions, sensor readings undergo filtering before being used by the flight algorithm. Noise, high-frequency fluctuations, and transient disturbances can interfere with control accuracy, making it crucial to apply low-pass filters implemented as discrete transfer functions. Each sensor's filtering is customized based on its noise characteristics and operation:

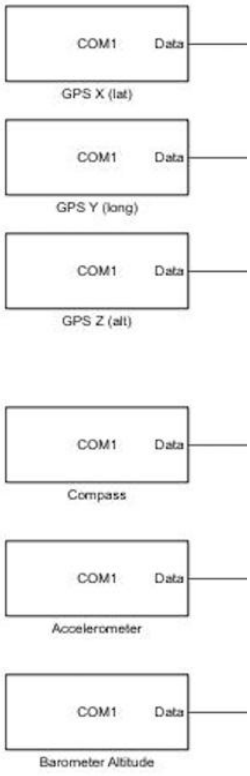


Figure 40: Serial receivers for sensors

1. **GPS Module (X, Y, Z Positioning):** The GPS system captures raw latitude, longitude, and altitude data via three independent Serial Receive blocks. This data is parsed through the parseGPS MATLAB Function Block, converting it into a usable numerical format. However, GPS signals are prone to positioning jitter and satellite errors, which may cause fluctuations in the reported position. A first-order low-pass filter (discrete transfer function) is applied post-parsing to smooth out inconsistencies, preventing erratic course corrections and ensuring gradual, stable position updates.
2. **Compass (Heading Information):** The compass sensor, interfaced through a Serial Receive block, undergoes processing via the parseCompass MATLAB Function Block. Magnetometers are susceptible to electromagnetic interference (EMI) from surrounding electronics, leading to erratic heading changes. To mitigate this, a low-pass filter is applied post-parsing to ensure stable, drift-free heading data, reducing unnecessary mid-flight trajectory corrections.
3. **Barometric Altimeter (Altitude Estimation):** Altitude data is received through a Serial Receive block and parsed using the parseBarometer MATLAB Function Block. Barometric altitude readings fluctuate due to air pressure instability, leading to inconsistent altitude estimates. A post-parsing low-pass filter eliminates transient variations, ensuring smooth altitude tracking and accurate descent velocity estimation.
4. **Accelerometer (Velocity Estimation):** The accelerometer transmits raw acceleration data, which is parsed via the parse Accelerometer MATLAB Function Block. The accelerometer readings contain high-frequency noise from structural vibrations, which can distort velocity calculations. To prevent this, a low-pass filter is applied before integration. The filtered acceleration is integrated using a Backward Euler method within

an Integrator Block, converting acceleration into velocity along the Z-axis for descent estimation.

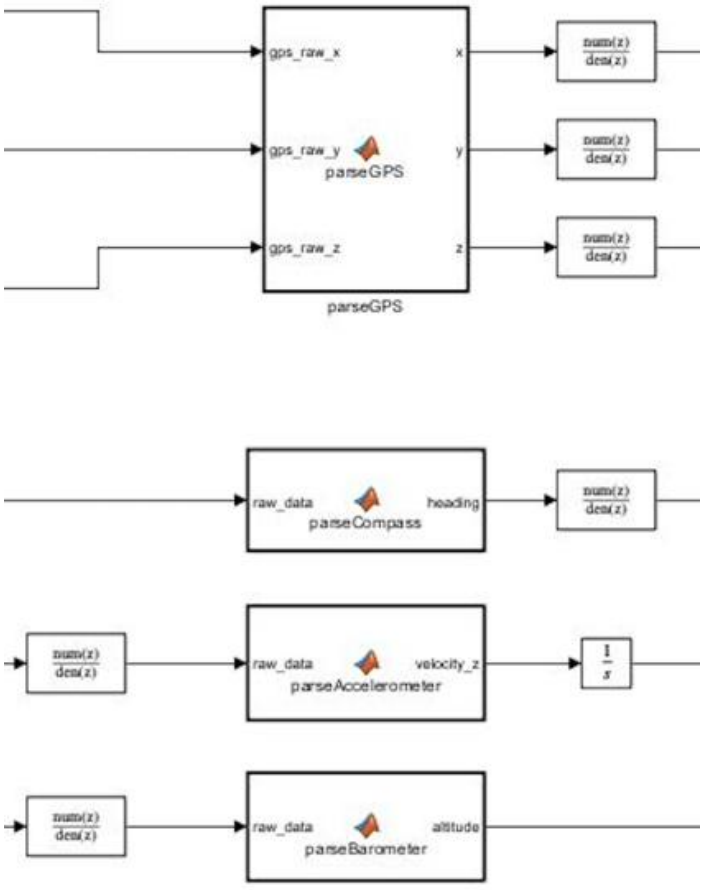


Figure 41: Processing and data filtering from serial receivers

All sensor data follows a strict 10ms update rate, ensuring real-time synchronization and minimizing latency between sensor readings and control decisions. The filtered outputs are fed into the flight control algorithm, ensuring smooth and precise trajectory corrections.

7.5.6. Flight Algorithm Execution

The Flight Algorithm MATLAB Function Block serves as the decision-making core of the guidance system. It processes filtered sensor data: position, altitude, velocity, and heading, while continuously adjusting the control surfaces to align the rocket with its intended landing coordinates. The algorithm performs the following key computations:

1. **Error Calculation and Servo Adjustment:** The first step in the algorithm is computing the difference between the current position (X, Y) and the target landing coordinates (X_target, Y_target). This position error determines the necessary servo adjustments. The servo angle is updated based on the difference between the current and target positions while factoring in the rocket's heading from the compass data. To prevent excessive oscillations or overcorrections, the servo angle is constrained within [-30°, 30°].
2. **Position and Altitude Updates:** The X and Y positions are updated based on the servo angle influence and velocity estimates, accounting for accelerometer and compass data. Altitude is adjusted using barometric sensor data, incorporating velocity_z to estimate descent rate while ensuring that the rocket's altitude never drops below zero
3. **Trajectory Refinement:** By continuously updating its trajectory based on real-time sensor inputs, the algorithm ensures that the rocket makes progressive course corrections rather than abrupt, unstable adjustments. This predictive control mechanism improves landing precision while minimizing oscillations.

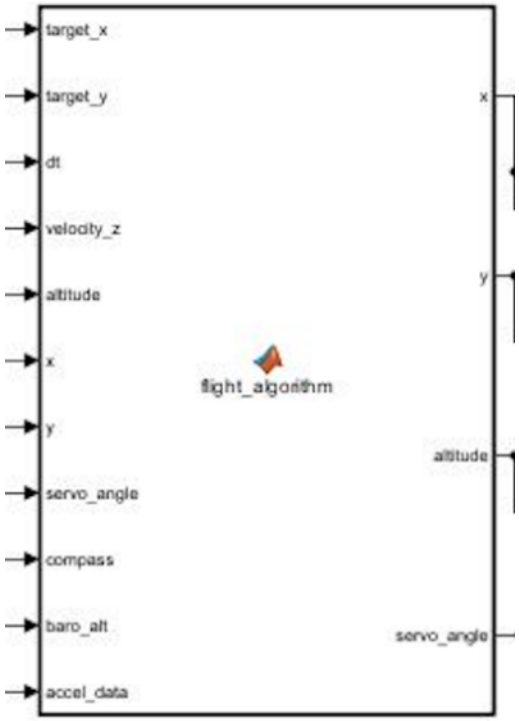


Figure 42: Flight algorithm block

7.5.7. Servo Control and PWM Conversion

Once the flight algorithm determines the necessary servo angle, this value needs to be converted into a PWM (Pulse-Width Modulation) duty cycle to properly drive the servo motor. This conversion is performed by the Angle-to-PWM MATLAB Function Block, which translates the servo angle into a corresponding PWM signal suitable for motor actuation.

Why PWM?

Servo motors require PWM signals to interpret positional commands. By mapping the servo angle into an appropriate duty cycle, the system ensures that the control surfaces move proportionally to the computed corrections.

Verification of PWM Output:

The PWM output is connected to a Scope block, allowing real-time visualization of the generated signal. This ensures that the signal aligns with expected servo movements before deploying to hardware.

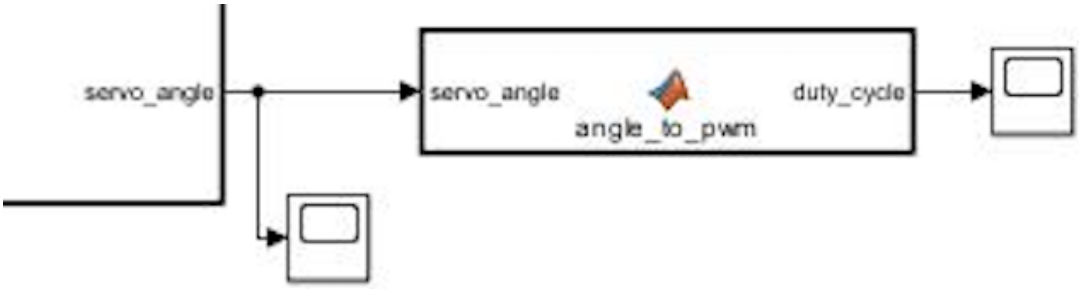


Figure 43: Servo processing & visualization

7.5.8. Visualization and Data Analysis

To monitor system performance, multiple Scope blocks and plotting subsystems are used for real-time analysis of the rocket's descent trajectory and control actions.

1. **X, Y, and Altitude Scopes:** These Scopes track the position and altitude evolution over time, verifying whether the trajectory follows the planned descent path.

2. **Plot_Trajectory Subsystem:** This subsystem visualizes the rocket's descent path in real-time, helping engineers verify whether the guidance corrections align with the intended trajectory.
3. **Servo Angle Scope:** By tracking servo actuation in real time, this Scope ensures that control surface movements remain within expected limits.

Together, these visualization tools offer immediate feedback, allowing for tuning and debugging of the guidance model before real-world deployment.

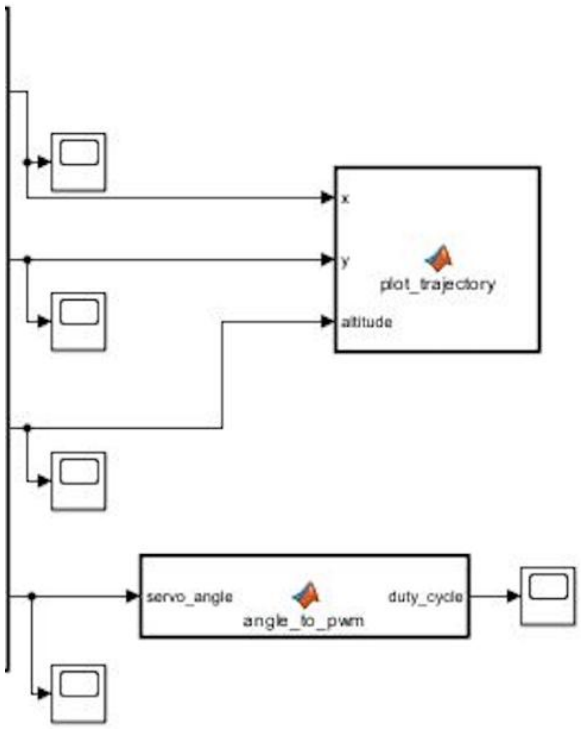


Figure 44: Scopes and plotting data

This Simulink model provides a fully functional, real-time representation of the RTLS Control System Diagram, incorporating sensor inputs, data filtering, flight control logic, and

servo actuation within a structured and synchronized computational framework. The integration of real-time signal processing, trajectory corrections, and predictive control adjustments allows for an accurate and practical simulation of the rocket's descent guidance. By filtering sensor data, ensuring stable position updates, and executing real-time servo actuation, this model serves as an ideal test environment for validating RTLS guidance algorithms before hardware implementation.

7.6. Ignition System

The RTLS ignition system is engineered to use an external 9V battery to reliably fire black-powder charges, since neither the ESP32's 3.3V logic output nor the onboard 7V system battery can deliver enough energy to initiate an E-match and achieve full combustion. The schematic implements two independent ignition channels, each with its own continuity-monitoring circuit.

- **Standby (continuity check):** With the ESP32 output low, the E-match leads form a closed loop that allows a nominal 3V bias current to flow; this voltage is continuously sampled by a dedicated GPIO continuity-detect pin represented by Figure 46.
- **Ignition:** When the ESP32 asserts a 3.3V trigger signal, it drives the base of a TIP-series switching transistor. The transistor then closes the 9V ignition circuit, delivering the required current pulse to the E-match for reliable ignition represented by Figure 46.

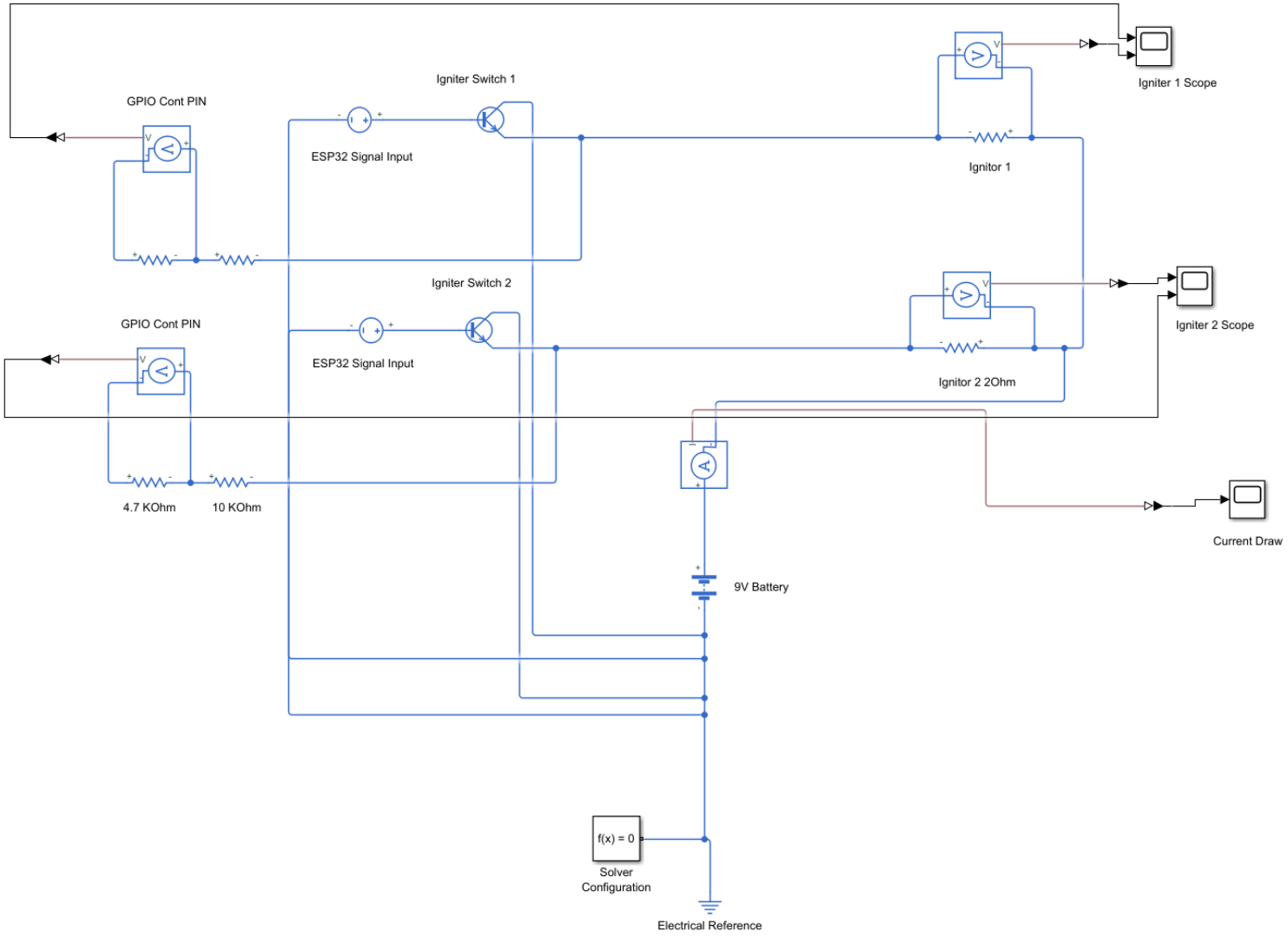


Figure 45: Ignition System

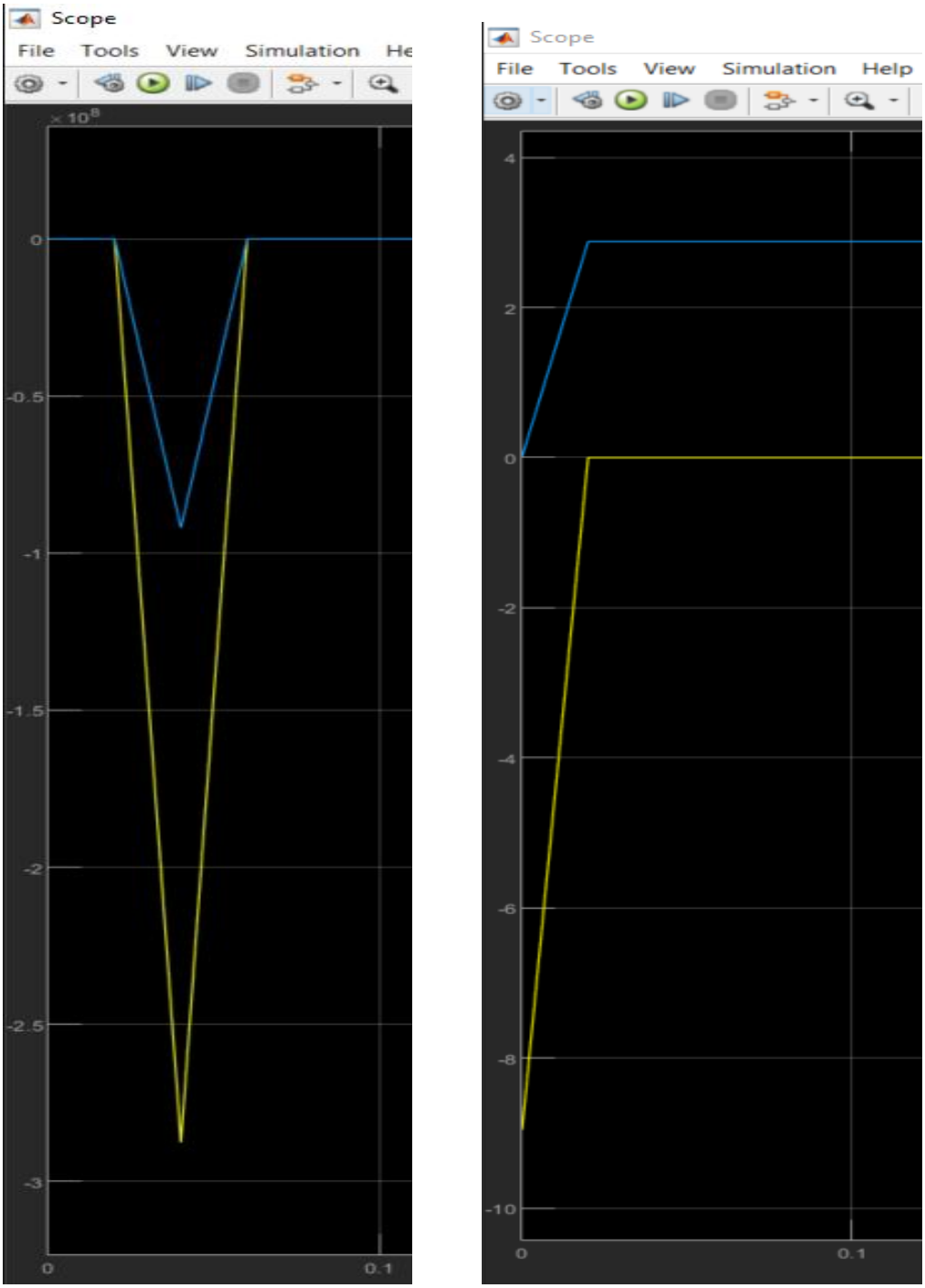


Figure 46: Ignition system continuity and ignition

8. Final Design and Engineering Specifications

8.1. Overview

The rocket is designed to achieve an apogee of approximately 2,200 feet and execute an autonomous guided return to the launch site, operating through three primary stages: launch, deployment, and guidance. During the launch phase, the rocket is secured to a launch rail via rail buttons, with an electronic ignition system igniting the motor. Aerodynamic stability during powered ascent is maintained by strategically designed fins and a weighted ogive-shaped nose cone.

At apogee, a primary ejection charge activates, separating the nose cone from the coupler and deploying a streamer. This streamer serves to stabilize the rocket vertically and slow its descent. Upon reaching an altitude of approximately 750 feet, a secondary ejection charge fires, shearing retention pins behind the coupler and deploying the RTLS system paraglider. To avoid tangling, the shock cord is externally routed along the rocket's body and secured with tape during launch preparations.

Once the main parachute is deployed, an onboard dual-core computer manages the active guidance phase, which is currently undergoing testing to gather critical performance data. Core 0 processes real-time sensor data, including GPS coordinates, altimeter readings, and compass orientation, transmitting continuous telemetry to the ground station. Core 1 operates the onboard control algorithms, which would actuate servo motors connected to the control lines of the paraglider. This dual-core configuration enables real-time adjustments to flight trajectory, ensuring precise navigation toward the predetermined landing site.

Additionally, the rocket includes a screamer unit to facilitate recovery efforts should radio communication become compromised during flight. Through this design, the rocket fulfills altitude, staging, and guidance requirements, achieving precise, autonomous landing capability.

This section will cover the final design of each subsystem of the rocket. This is an extension of Section 6 of this report, which outlines the design concepts. Through simulation, analysis and testing, the team concluded that a few design changes would be necessary to satisfy the project requirements. These changes will be reflected here and thoroughly explained in Section 10.

8.2. Key Performance Parameters

Function	Design Requirement	Specification	Analysis, Testing, Inspection, Demonstration
Rocket shall return to a specified target	Distance from target.	824 ft	Testing/Analysis
	Distance of target from launch site.	0 ft	Inspection
Rocket shall reach apogee	Apogee height	2794 ft	Testing/Analysis
Rocket shall land safely	Impact velocity	16.8 ft/s	Testing/Analysis
Rocket shall have a two-stage recovery system	Streamer 1 deployment height	2794 ft	Testing/Analysis
	Streamer 2 and paraglider deployment height	750 ft	Testing/Analysis
	Main chute descent velocity	16.8 ft/s	Testing/Analysis

Table 9: Key System Function Specifications

Due to deployment failure during a flight test, many of these specifications were based on simulations of the final vehicle design on OpenRocket and flight simulations developed by the team. The vehicle was lost, so no data could be recovered from the flight.

8.3. Airframe

8.3.1. Overview and Design Methodology

All CAD models presented in this section were developed using SolidWorks 2023 Student Edition. These models provide a detailed representation of the rocket's internal geometry, including the layout of all subsystems, component placement, and structural joints. The primary purpose of the CAD model is to clearly communicate the design decisions made to effectively house all internal components while maintaining structural integrity under flight conditions. Emphasis was placed on efficient space utilization, secure mounting of critical hardware, and the overall assembly workflow. These models also support assessments related to manufacturability and ease of integration during final assembly.

The dimensions of the rocket are as follows…

- Length: 54.8 inches
- Width: 10 inches
- Outer Diameter: 4 inches
- Weight: 6.49 lb

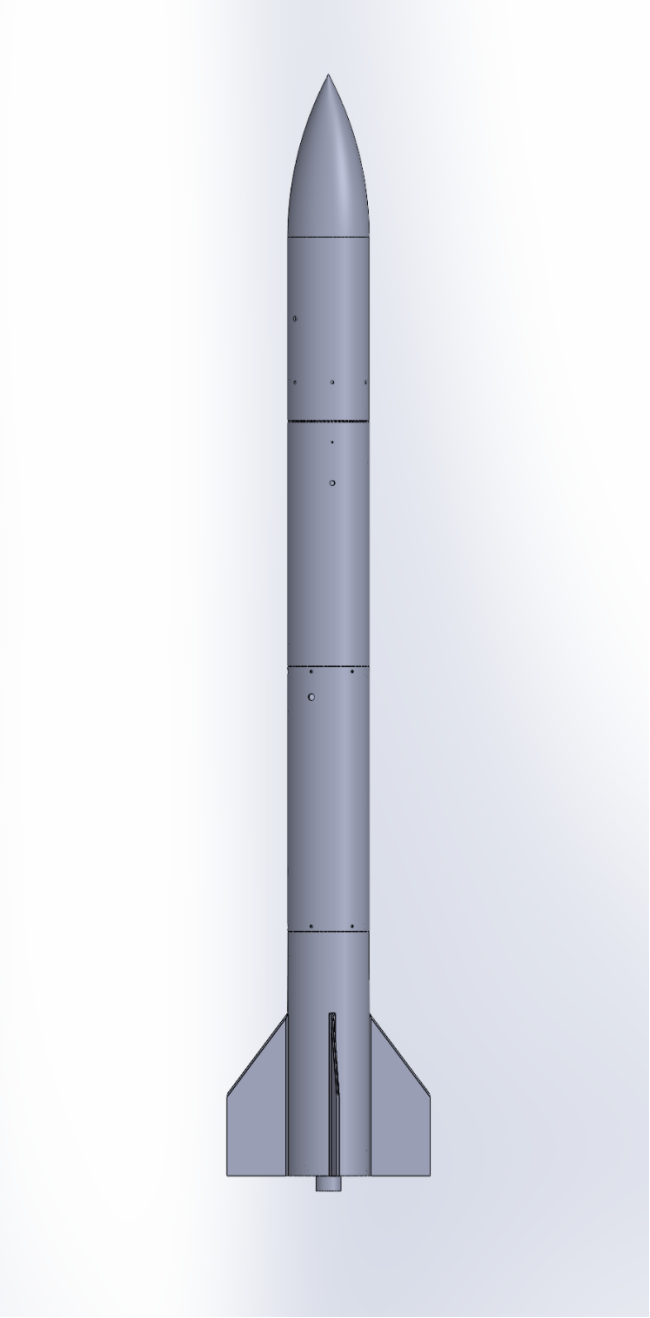


Figure 47: Rocket prototype CAD (side view)

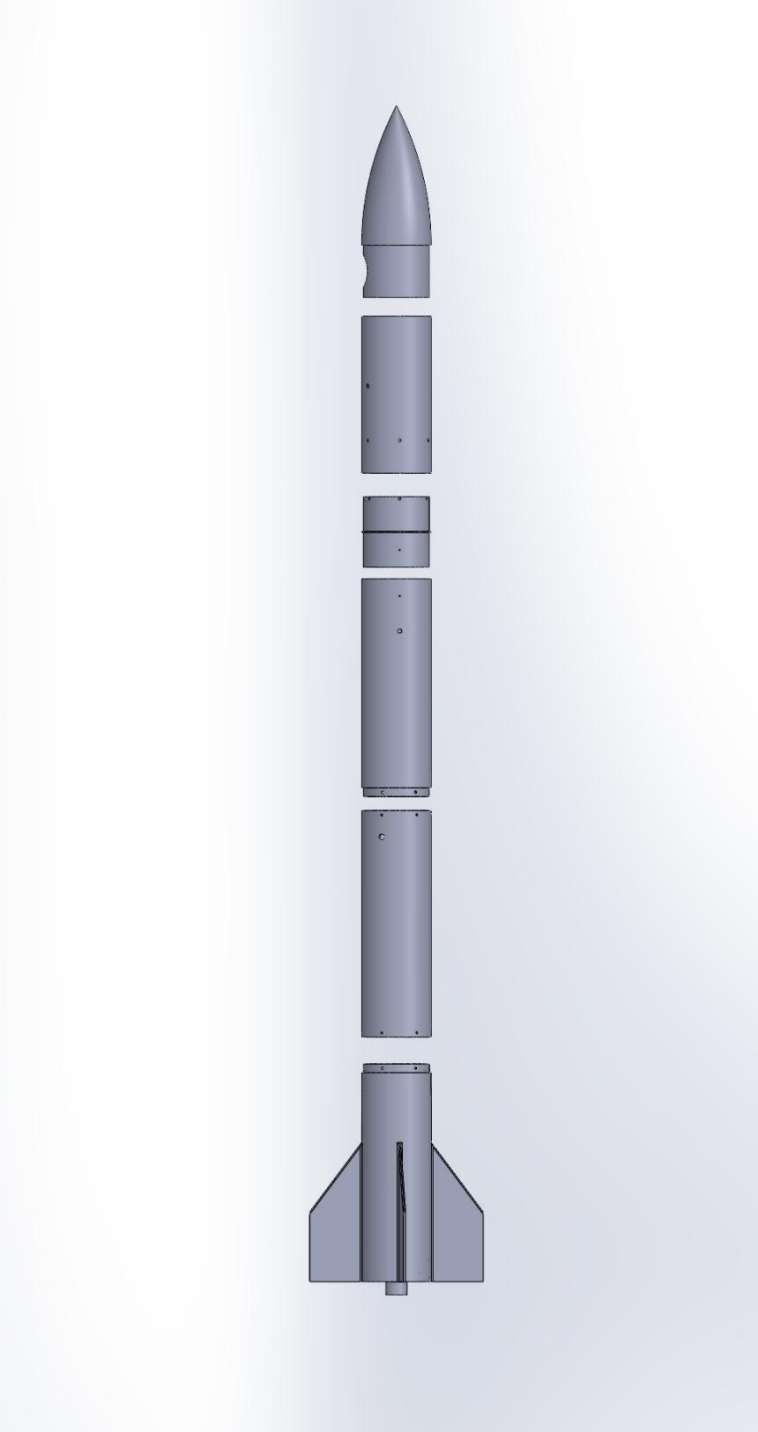


Figure 48: Rocket prototype CAD (exploded view)

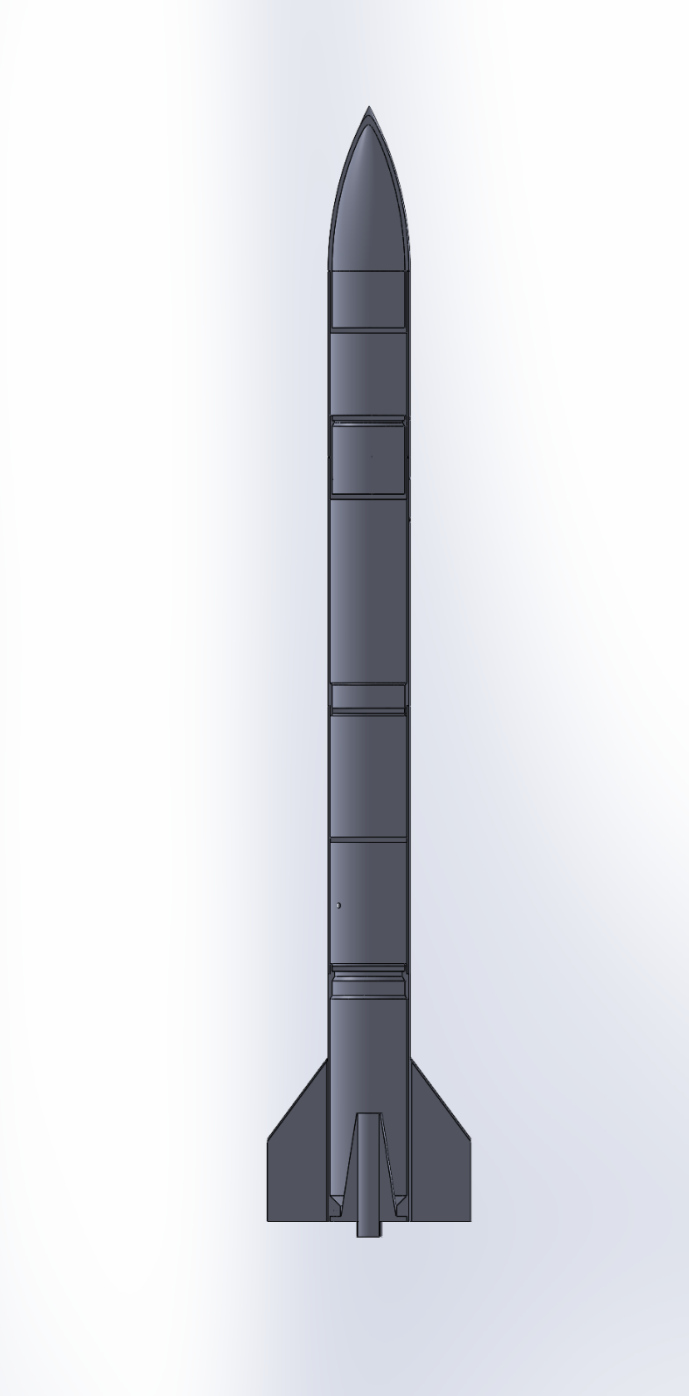


Figure 49: Rocket prototype CAD (section view)

8.3.2. Airframe Component Specifications

This part will outline the function and design rationale for each airframe component in order from foremost to aftmost (nose cone to motor assembly). All components in this section will be 3D printed out of PETG filament.

Nose Cone:

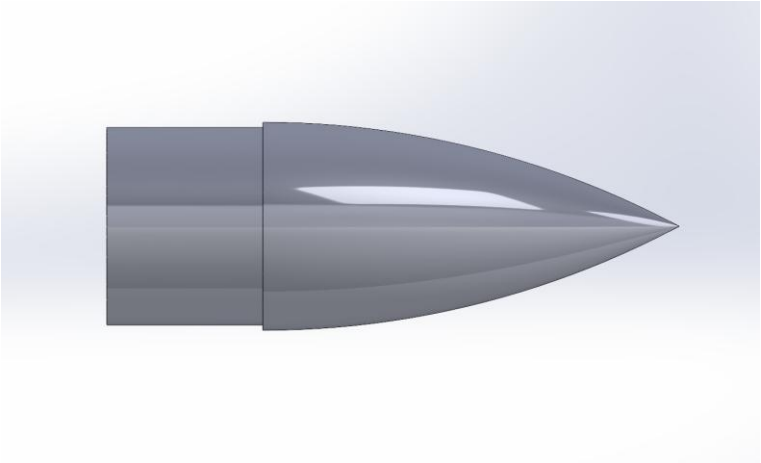


Figure 50: Nose cone CAD (side view)

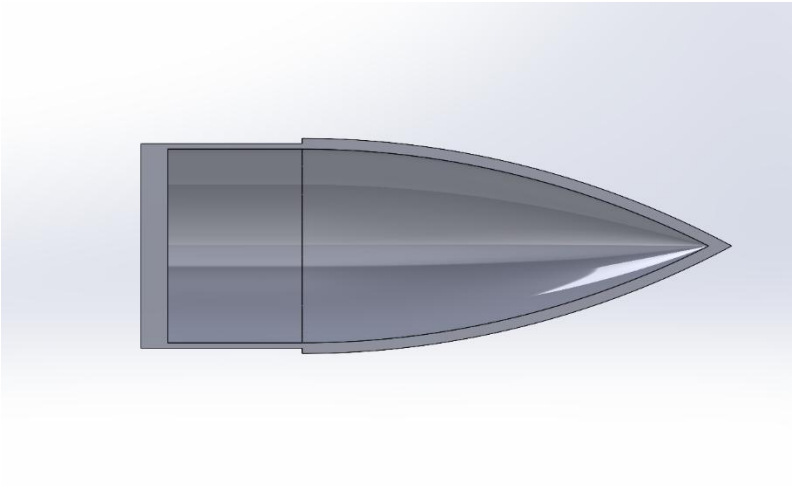


Figure 51: Nose cone CAD (section view)

The ogive nose cone shape was chosen because of its aesthetics and its seamless integration into the airframe. The ogive curve terminates tangent to the body tube, allowing for a smooth transition. A shoulder joint was added to be friction fitted into the upper body tube.

At apogee, the nose cone will be blown out of the upper body tube by an ejection charge to deploy the streamer. This will require an eye bolt to be mounted to the bottom of the shoulder for the shock chord to tie to (not pictured). Extra thickness was added to the eye bolt attachment surface to ensure the nose cone can withstand deployment forces. Both a hole for a ¼”-20 eye bolt and an access port were added to the model.

Upper Body Tube (Streamer Bay):

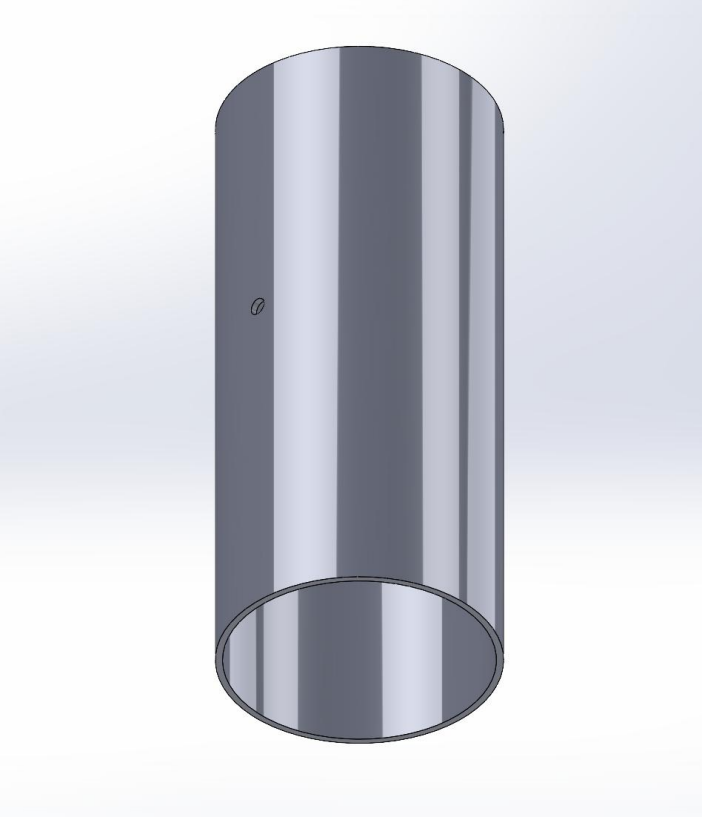


Figure 52: Upper body tube CAD

The upper body tube will house the streamer and its associated ejection charge. The tube was designed with 1/10” thick walls to resist compressive flight forces. A pressure relief hole was also added to allow the ejection charge gases to escape. All body tubes in the airframe have these two features. The tube will be friction fitted to both the nose cone and bolted to the coupler.

Coupler:

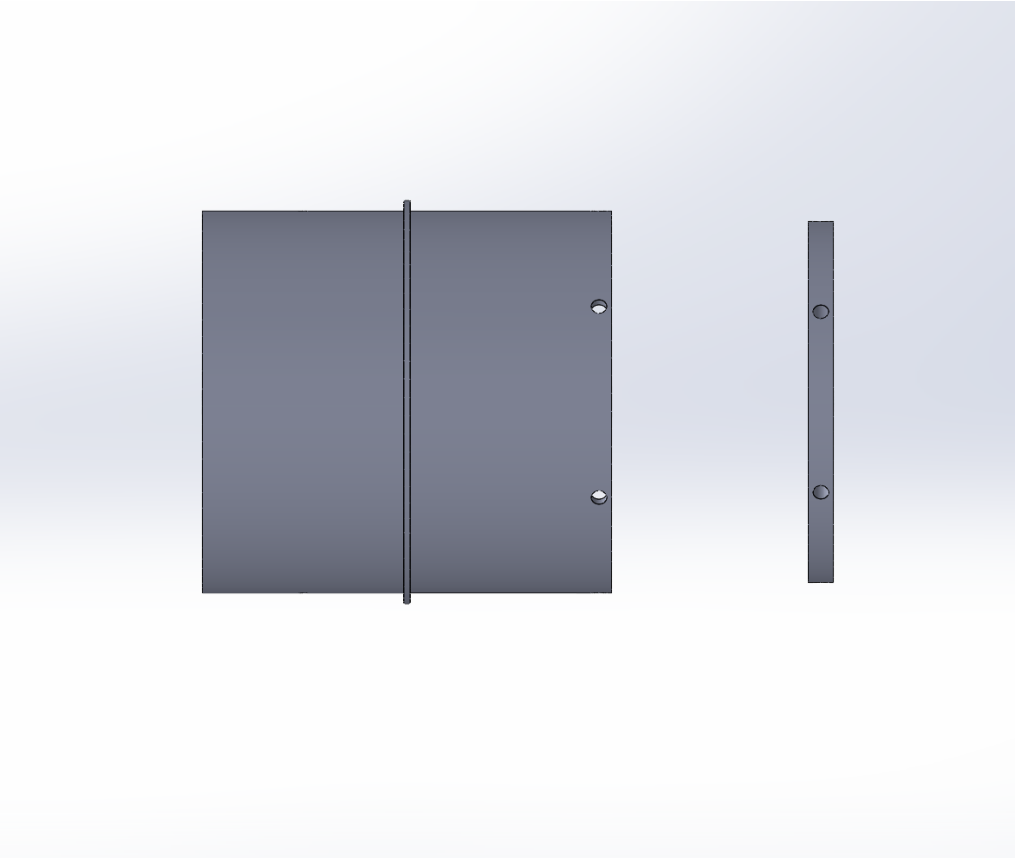


Figure 53: Coupler CAD (side view)

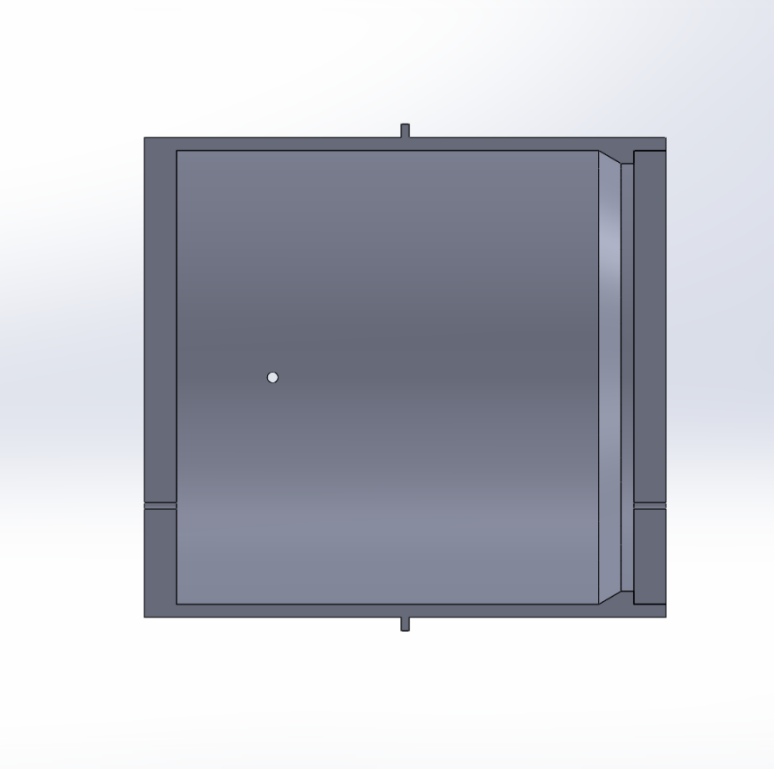


Figure 54: Coupler CAD (section view)

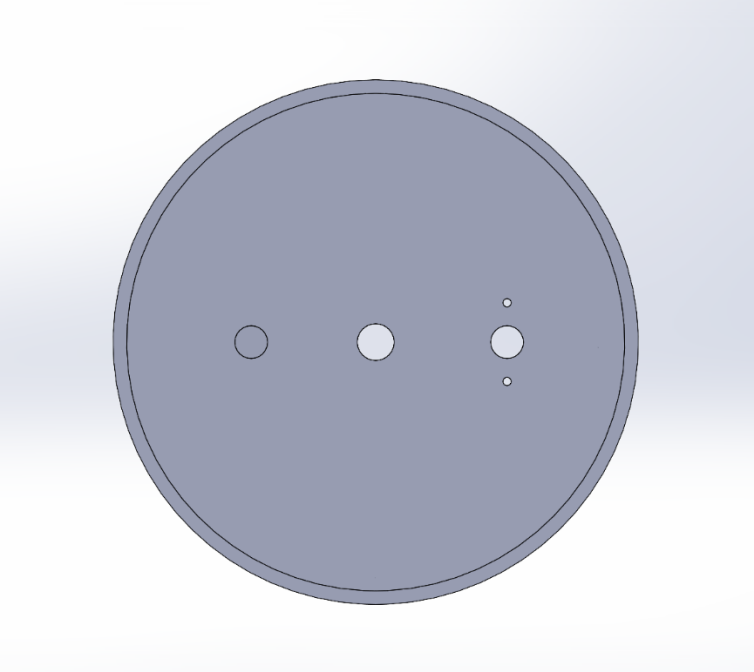


Figure 55: Coupler CAD (top view)

The coupler will join both the upper body tube and middle body tube by friction fit and a fastened joint. It will also serve as the attachment point for both shock chords in the rocket. The coupler was designed into two separate pieces so that extra weight is not spent on support material when 3D printing, and eye bolts can be attached and preloaded on the top and bottom plates (not pictured). Preloading the eye bolts will allow the tension forces from deployment to be distributed through the coupler and will therefore mitigate the risk of pull-out failure. Extra thickness was added to the top and bottom plates for added bending stiffness. Holes were also added to mount the ejection charges and igniters on the top and bottom bulkheads. The removable lid allows users to easily load new charges for vehicle reuse.

Middle Body Tube (Paraglider Bay):

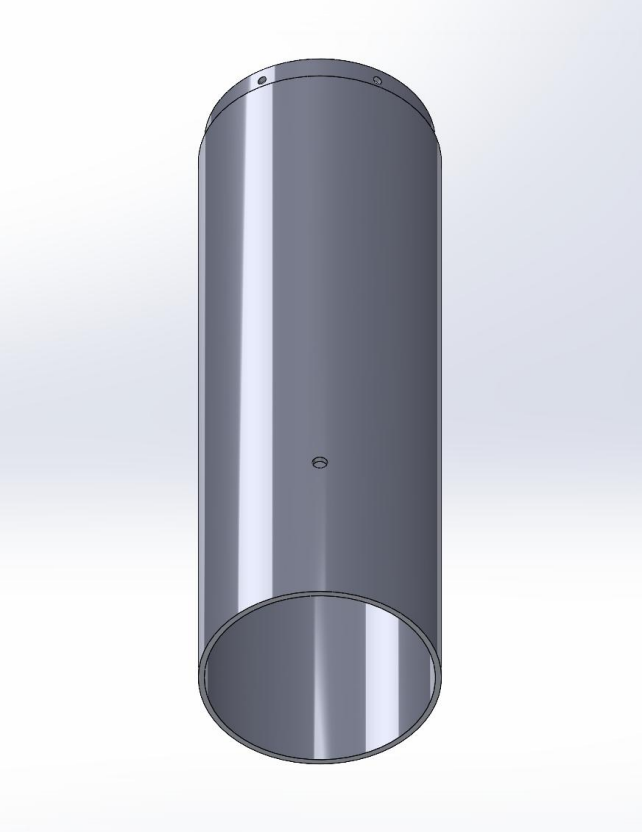


Figure 56: Middle body tube CAD

The middle body tube will house the parafoil and the streamer used to eject it. The coupler will be inserted into the side without holes. One side will be friction fitted to the coupler to allow paraglider deployment, and the other will be fastened to the electronics bay and motor assembly. This is to ensure that the electronics bay remains attached to control the paraglider during descent.

Lower Body Tube (Electronics and Servo Control Bay):

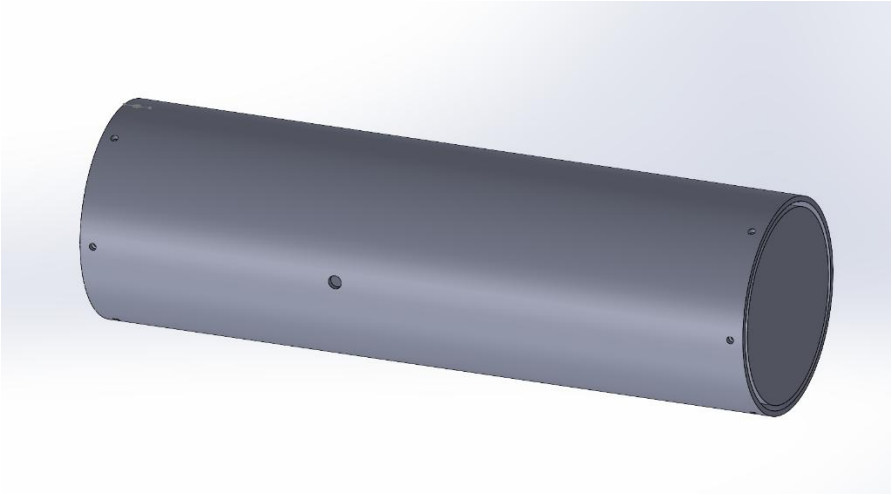


Figure 57: Lower body tube CAD (side view)

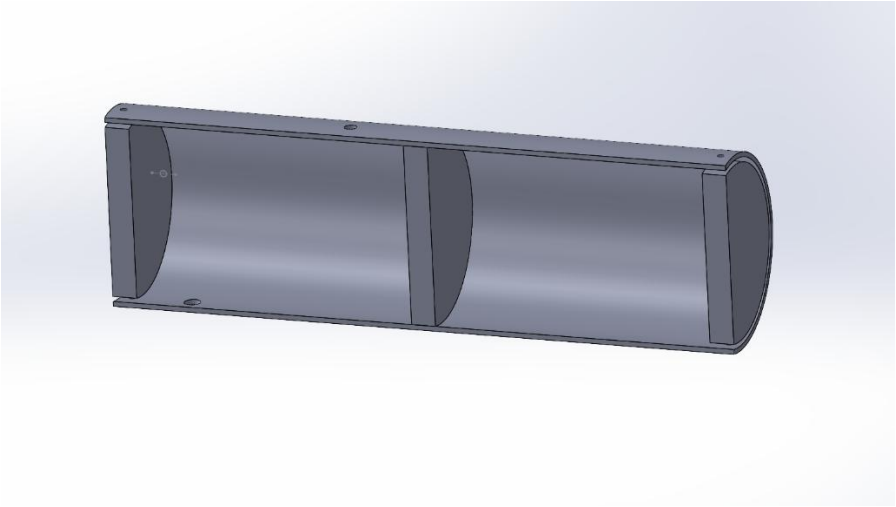


Figure 58: Lower body tube CAD (section view)

The lower body tube's primary function is to house the electronics and servo control system for the parafoil. These will be separated by the bulkhead in the middle but will be connected by wires through a small hole (not pictured) The setup is shown in Figure 59. The control lines from the parafoil that attach to the servo will also be passed through the upper bulkhead into the middle body tube through a hole (not pictured). A pressure relief hole is also featured in the electronics bay to allow the altimeter to see the outside air pressure.

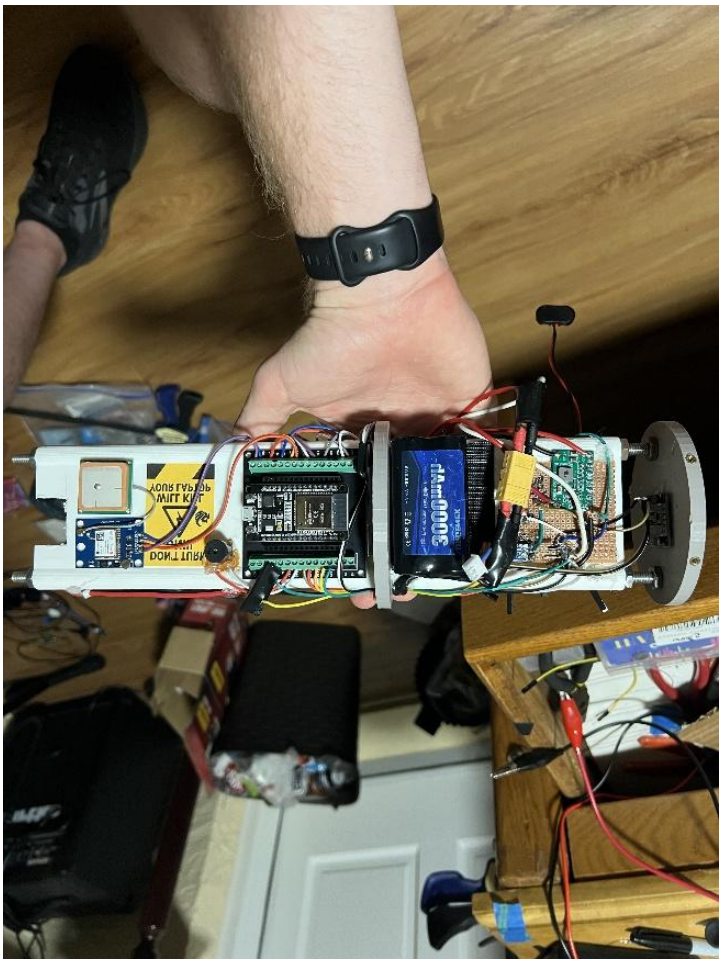


Figure 59: Electronics Bay

Another function of the lower body tube is to serve as an attachment point for the lower shock chord holding all the components from the middle body tube and above. The shock chord will be tied to an eye bolt attached to the lower bulkhead (not pictured) and passed through a hole in the wall of the body tube to the outside of the rocket. The reason for attaching the lower shock chord to the aft portion of the lower body tube is to prevent tangling with the parafoil lines.

Motor and Fin Assembly:

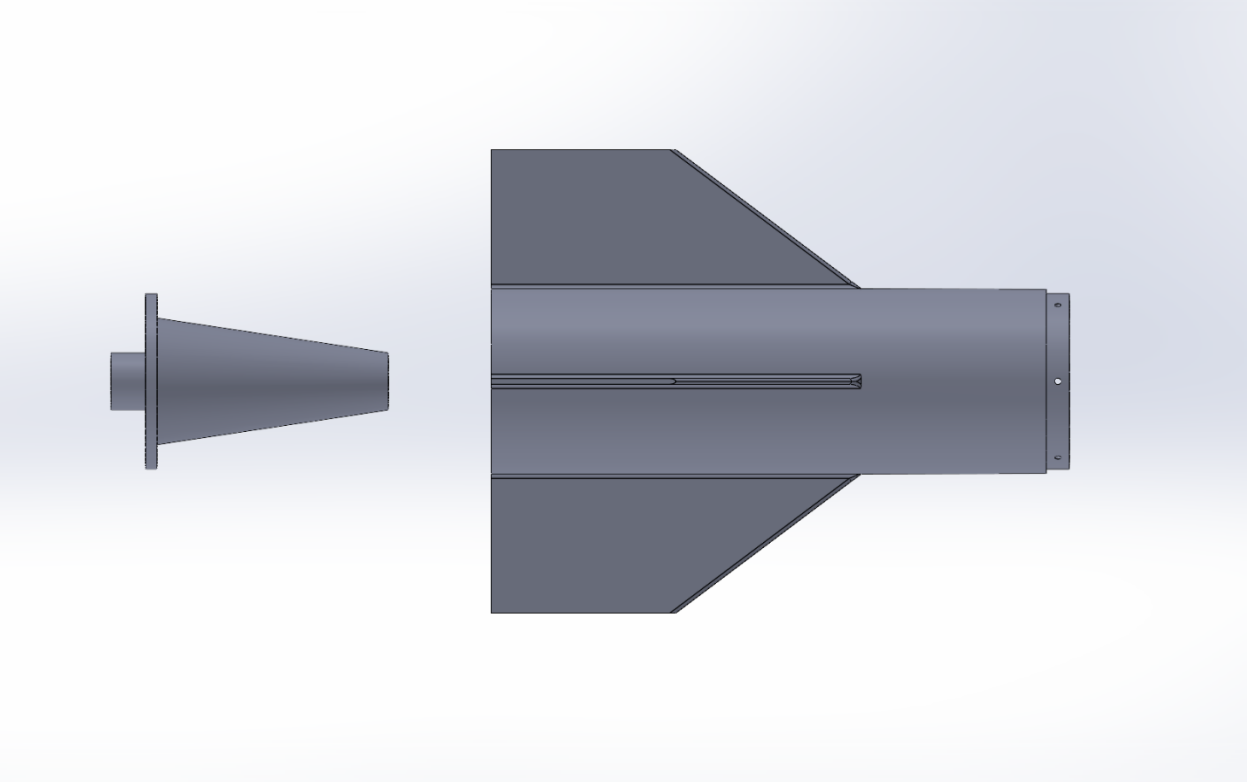


Figure 60: Motor and fin assembly CAD (side view)

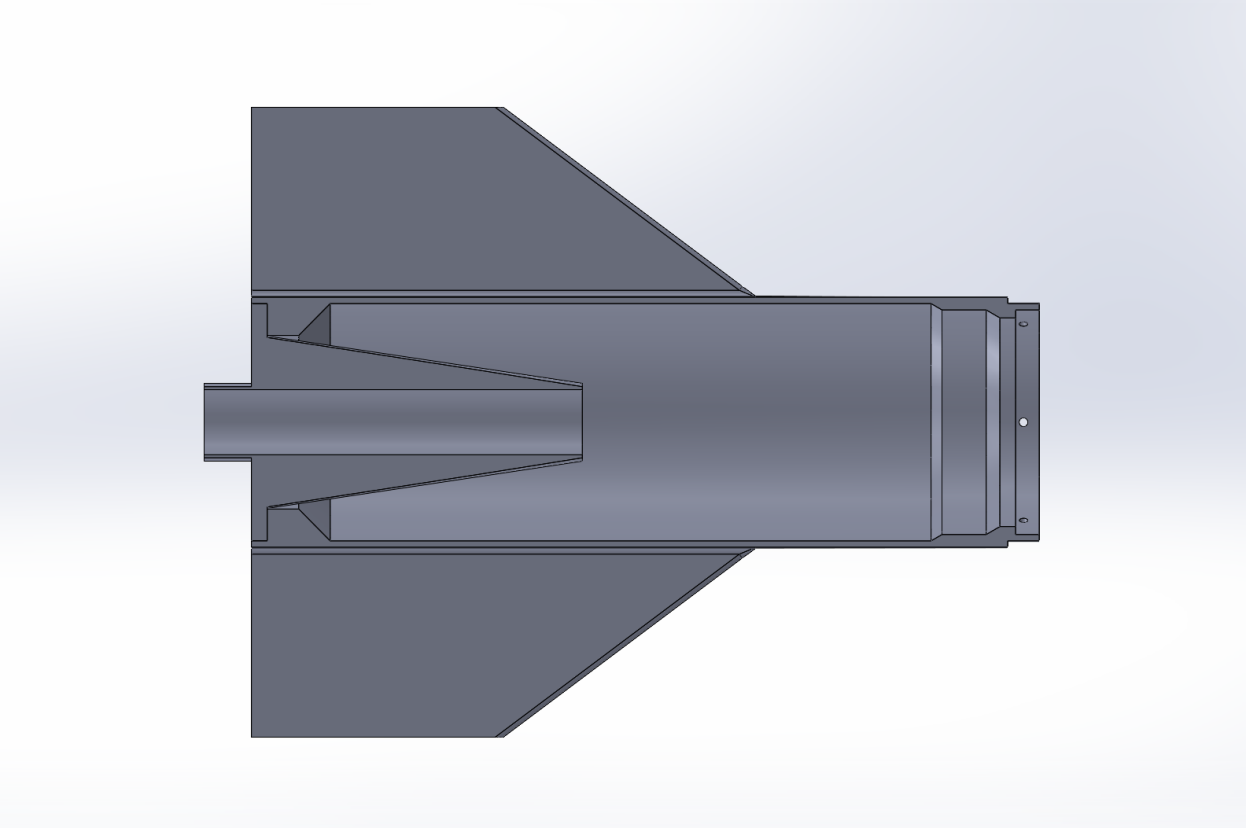


Figure 61: Motor and fin assembly CAD (section view)

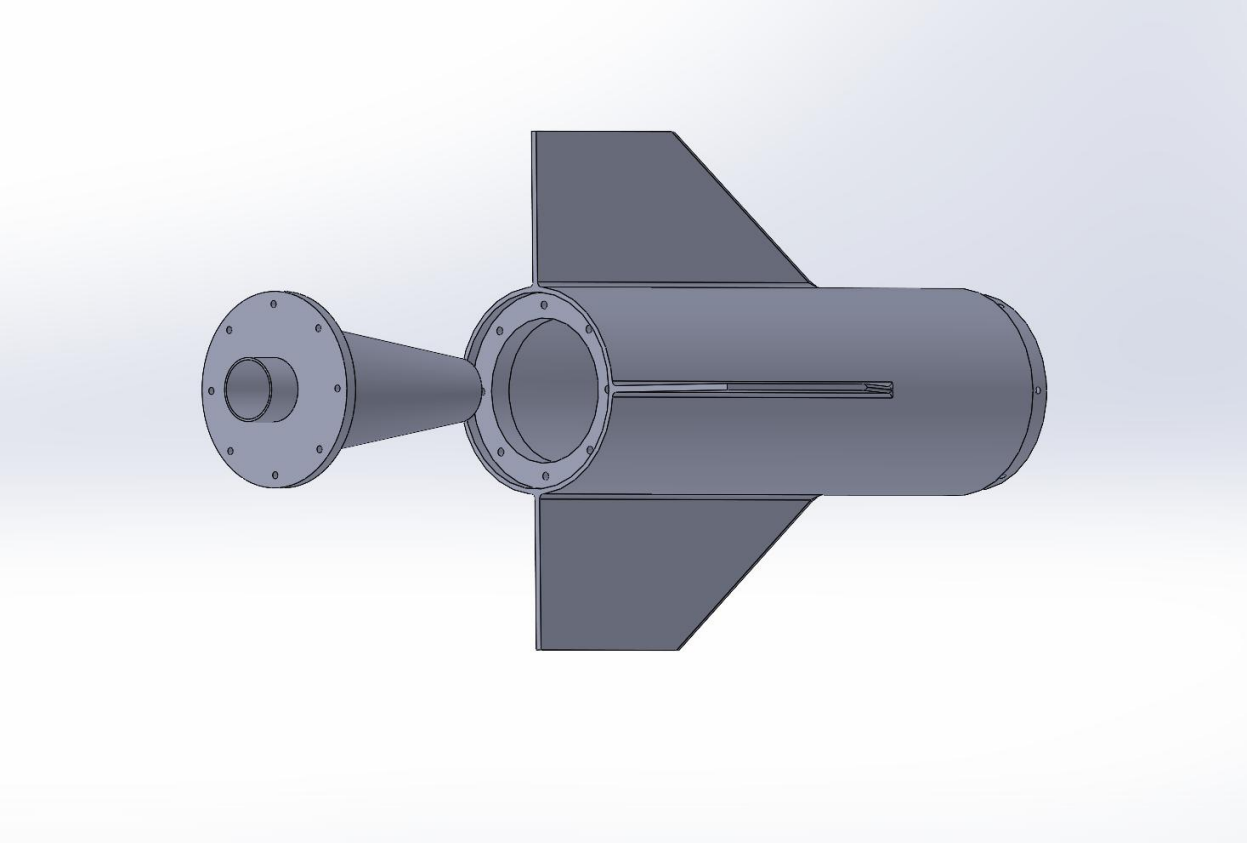


Figure 62: Motor and fin assembly CAD (rear view)

This assembly contains the fins and motor tube for the rocket. The clipped delta fin shape with rounded corners was chosen for high stability and low drag. The motor tube includes a tapered stiffener to resist lateral motor vibrations during flight. The fins include fillets at their corners to distribute the shear stress from flight loads to the body tube more efficiently. A motor retainer will be adhered to the outlet of the motor tube. The bottom bulkhead inside the lower body tube will insert into the top of this assembly.

8.3.3. Airframe Manufacturing

An FDM 3D printer will be used to manufacture all airframe components. The printer chosen for this project is the Prusa XL (Figure 63). This printer was chosen for its high accuracy

and large print volume (14” x 14” x 14”). All components will be printed with PETG filament. PETG was chosen for this application because of its high strength, rigidity, heat resistance, and weather resistance. ASA was also considered; however, it requires an enclosure and its tendency to string while printing. Due to these issues, PETG is the superior option.

Figure 63: Prusa XL 3D printer [45]

3D printing was chosen as the manufacturing method for this project for several reasons. First, 3D printing offers the advantage of full customization to engineers designing parts. Another reason is that 3D printing is highly affordable and allows users with access to 3D printers to easily repair broken or worn parts on their own. This capability is crucial in creating sustainable and reusable rockets. Finally, 3D printing mitigates many of the errors encountered when building high powered rockets. Since most users of rockets in the L1 class do not have access to advanced manufacturing equipment, there can be several inaccuracies in their builds. An example is uneven fin mounting due to measurements and building being done by hand. A properly calibrated 3D printer allows these parts to be manufactured almost effortlessly with

minimal error. However, there are a few complications to consider when manufacturing with 3D printing.

The complication that most significantly impacts designs is the incapability of 3D printers to build overhangs greater than 45 degrees. This was considered in the design of all airframe components. The optimal solution to this problem is to create fillets that gradually transition into overhangs (Figure 64), but support material can also be used. The issue with support material comes when it needs to be used in areas of components that are inaccessible, leaving it unable to be removed and resulting in excess weight of the part. Since excess weight is highly undesirable in rocketry, the components were designed to avoid inaccessible support material at all costs.

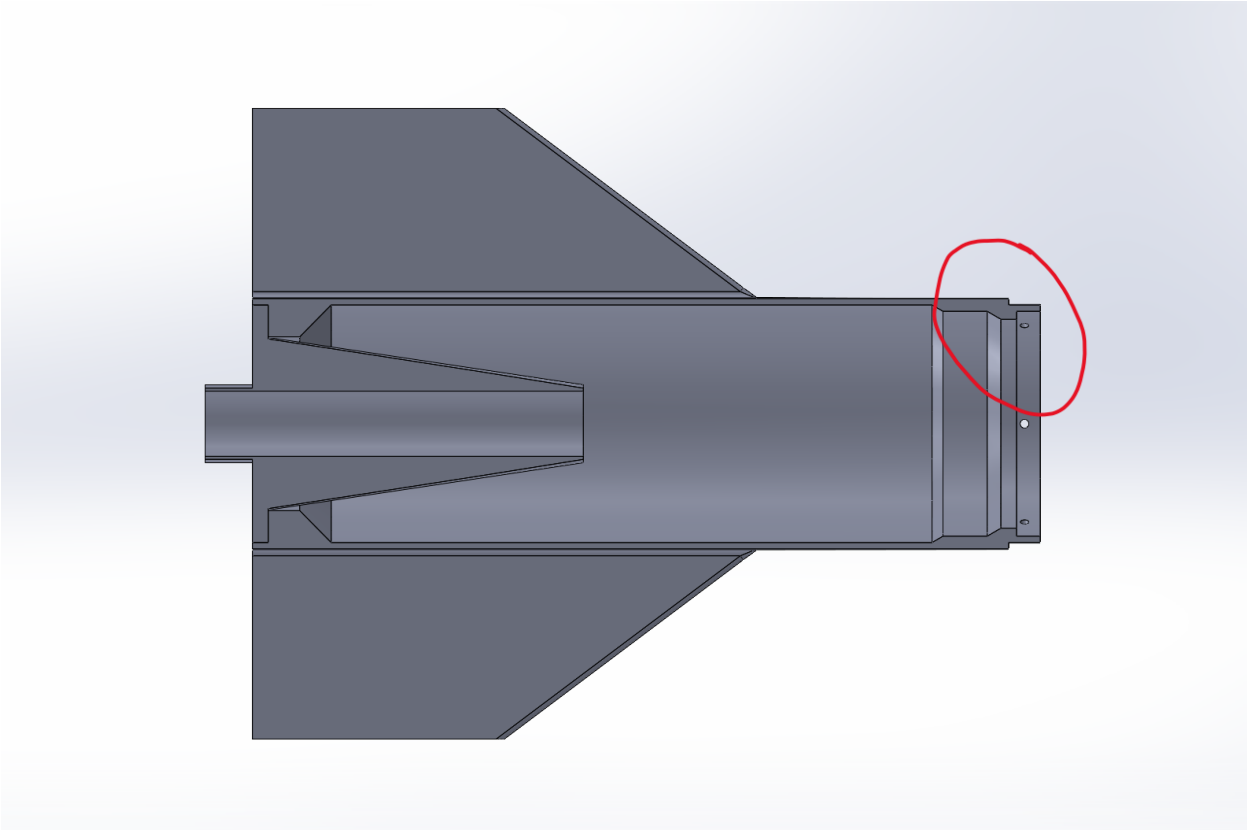


Figure 64: Filleted overhangs on the motor and fin assembly (circled in red)

Another consideration that needs to be made when designing large 3D printed parts is the maximum print volume. Since the size of the rocket far exceeds that of Prusa XL's maximum print volume, many components had to be printed in sections. To build a sturdy vehicle that can withstand high flight loads, strong joints must be designed to connect these sections. In sections where deployment is required (e.g., upper body tube), friction-fitted joints were used. These joints will be tested to ensure proper tolerancing between the male and female joint component. In sections that will remain connected throughout flight, lap joints were used with a ring of fasteners (Fig. 17). Six M3 x 5 bolts will be used to secure each joint. Because printing threaded holes is unreliable, holes will be created to fit threaded inserts. These joints will safely carry the shear and bending loads during deployment, as well as the compressive stresses during flight.

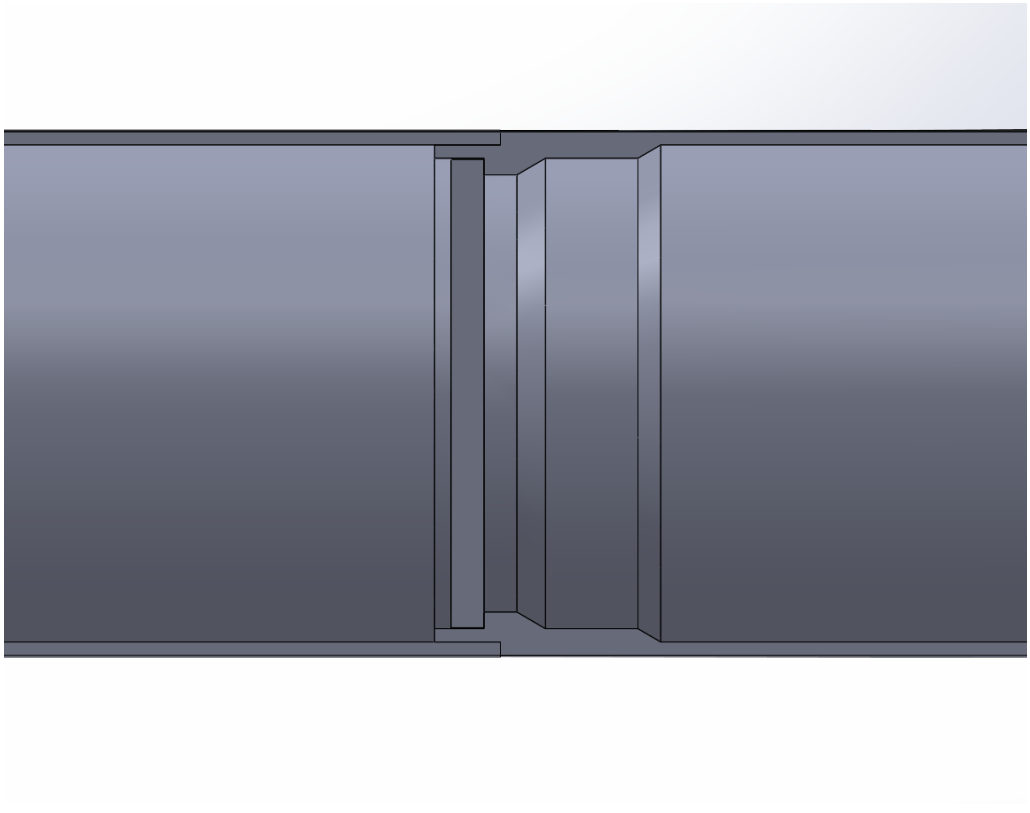


Figure 65: Lap joint cross section

8.4. Hardware

8.4.1. Introduction

Every hardware decision in our design was driven by budget constraints, ensuring the most cost-effective yet high-performance components were selected. Although several consumer-grade products were available, their price points exceeded our allocated resources. Instead, we strategically integrated a suite of budget-conscious components, namely an IMU, GPS, SD card, servo, ignition circuit, radio, power distribution board, servo, and microcontroller—to create an efficient and economically viable system for the RTLS project without compromising on functionality.

8.4.2. Hardware Control System

The ESP-32 controls all functions of the RTLS system including reading all input data, running the RTLS algorithm, initiating the ignition system, and outputting the data both to the onboard memory and the ground station for GPS location and some of the data to avoid loss of the rocket and data. For an explanation of the control system look at Figure 22 below.

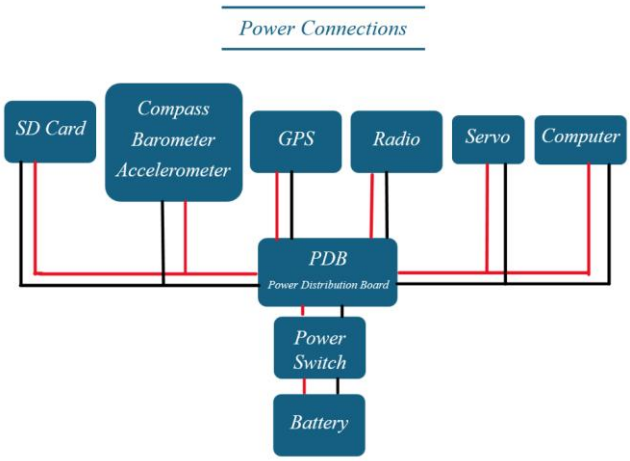


Figure 66: Power Connection Diagram

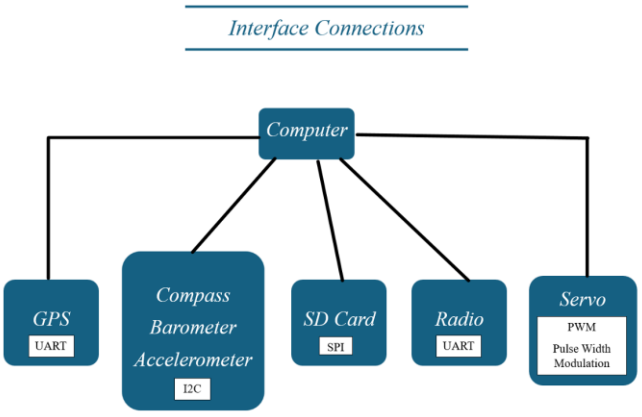


Figure 67 Interface Connections

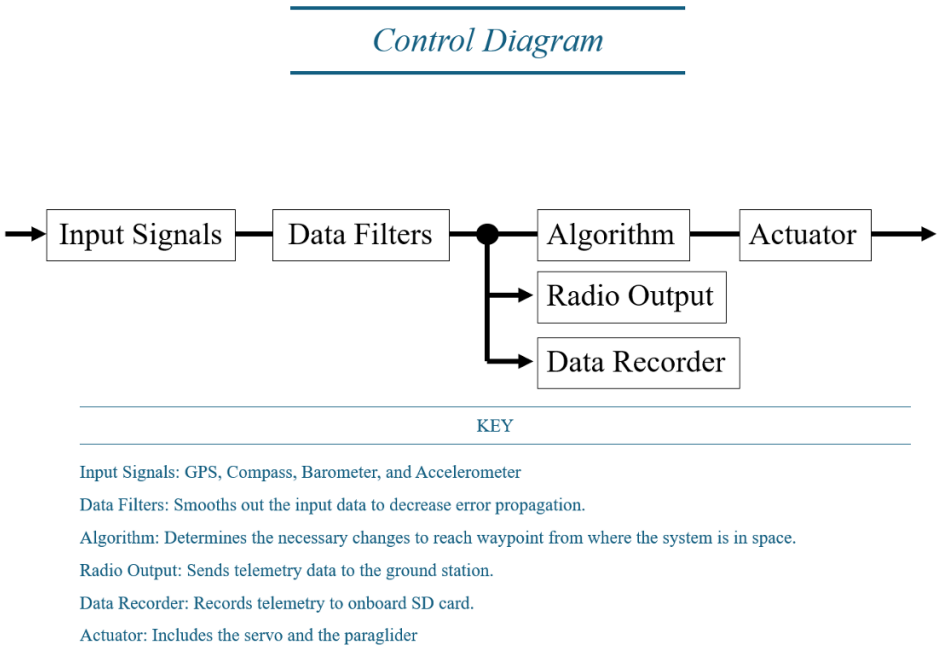


Figure 68: Control Diagram

8.4.3. Component Break Down and Specification

1) IMU (Inertial Measurement Unit) Berry IMU V3: This module combines a magnetometer, gyroscope, and accelerometer, furnishing precise measurements of angular velocity, linear acceleration, and magnetic orientation.

- Gyroscope & Accelerometer (LSM6DSL):

  Measurement Range (Gyroscope): ±125, ±250, ±500, ±1000, and ±2000 dps

  Sensitivity Options (Gyroscope): 4.375, 8.75, 17.50, 35, and 70 mdps

  Accelerometer Scales: ±2g, ±4g, ±8g, and ±16g

  Combined sensor output rate up to 6.7 kHz (6,664 samples per second)

- Magnetometer (LIS3MDL):

  Full-scale Magnetic Field Options: ±4, ±8, ±12, and ±16 gauss

- Barometric Sensor (BMP388):

  Pressure Range: 300 – 1250 hPa

  Pressure Resolution: 0.18 Pa (equivalent to measuring altitude changes of under 10 cm)

  Absolute Pressure Accuracy: ±50 Pa

  Relative Pressure Accuracy: ±8 Pa (≈±0.6 m altitude accuracy)

  Temperature sensor integrated; operating range: -40°C to +85°C

2) GPS Module: Employed for six degrees-of-freedom orientation, the GPS not only complements the IMU but also provides essential positioning data that is critical during the dynamic phases of flight.

- NEO-6

  Horizontal Accuracy: Circular Error Probable (CEP) of 2.5 meters.

  Vertical Accuracy: 4–7 meters.

Time-to-First-Fix (TTFF): 30 seconds for a cold start.

Performance Conditions: Verified under clear, unobstructed sky conditions.

3) SD Card: Serving as an onboard data-logging tool, the SD card records telemetry and sensor readings throughout the mission, facilitating.

- HW-125

  Operating Voltage: 4.5V to 5.5V DC

  Write speed: ~80 to 250 KB/s

  Read speed: ~150 to 400 KB/s

4) Radio Module: In tandem with the SD card, the radio module supports real-time data transmission, ensuring that vital telemetry can be monitored remotely during flight.

- RYLR896

  Frequency Range: Minimum: 862-1020 MHz

  RF Output Power Range: –4 dBm to +15 dBm

  RF Sensitivity: –148 dBm (very sensitive — great for long-range)

  RF Input Level (Max): +10 dBm

  Filter Insertion Loss: 1–3 dB

  Frequency Accuracy: ±2 ppm

  Communication Range: Minimum: 4.5 km Maximum: 15 km

- Super 8 Antenna

  Frequency Range: 860 – 930MHz (LBT)

  Peak Gain: 3dBi@900Mhz

  Feed Impedance: 50 ohm

  RF Cable: RG405 Semi-flexible

5) Power Distribution Board: Used to take in the battery voltage 9-7v and step down to the correct voltage for the components.

- MP1495 DC-DC Step Down Module Voltage

  Input Voltage: 5V to 16V DC

  Output Voltage Options: 1.25V, 1.5V, 1.8V, 2.5V, 3.3V, 5V (default is 1.25V)

  Maximum Output Current: 3A

  Conversion Efficiency: Up to 90%

  Switching Frequency: 500 kHz

6) Servo: Used to control and actuate the paraglider.

- Spektrum A6390

  Input Voltage: 4.8V to 6.0V DC

  Torque: 48.6 oz-in (3.5 kg-cm) @ 4.8V - 66.7 oz-in (4.8 kg-cm) @ 6.0V

  Speed: 0.17 sec/60° @ 4.8V - 0.14 sec/60° @ 6.0V

7) Ignition Circuit: The ignition circuit is dedicated to shoot deployments

   Made with a 10Kom and 4.7Kom resistor, Tip 120, and diodes.

8) Microcontroller (ESP32): Operating under the Arduino IDE framework, our custom code efficiently initializes and manages the interconnection among sensors and control elements, forming a responsive and reliable control system for the rocket.

- ESP32-Wroom

  Clock Speed: Up to 240 MHz

  Performance: Up to 600 DMIPS

  Internal Ram: 520 KB SRAM

  Internal Rom: 448 KB

External Flash: SPI Flash (commonly 4MB–16MB supported)

Operating Voltage Range: 2.3V to 3.6V (3.3V typical)

Current Consumption:

Wi-Fi active: ~160–260 mA

Light sleep: ~1 mA

Deep sleep: ~10 µA

8.4.4. Electronics Bay

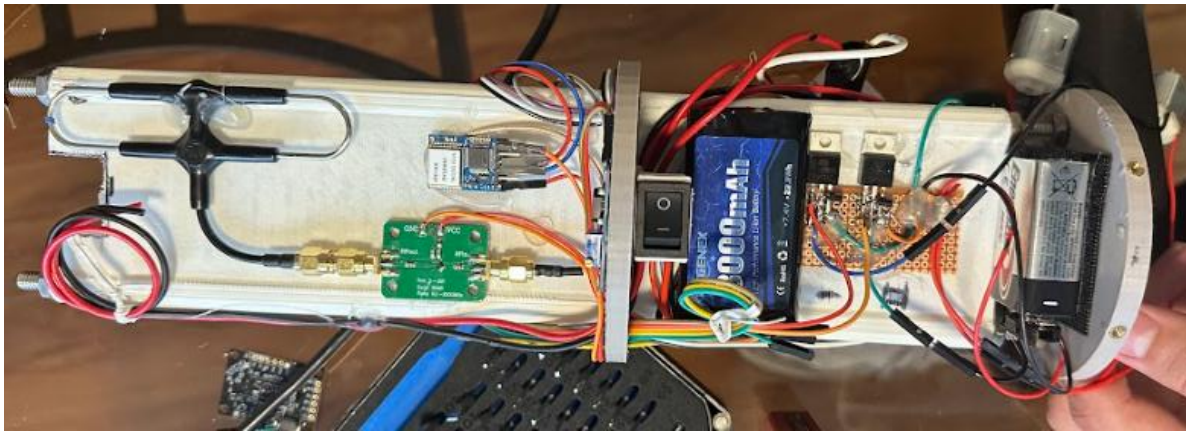


Figure 69: Front of E-Bay

Figure 70: Back of E-Bay

8.5. RTLS Software Architecture – Final MATLAB Simulation and Hardware Integration

The RTLS software system was developed to support autonomous descent, guidance, and return-to-launch-site navigation using a heading-based control strategy. The software architecture combines simulation-based modeling in MATLAB and a real-time hardware-in-the-loop (HIL) implementation using an ESP32 microcontroller. This section provides a breakdown of the final MATLAB control system and its corresponding HIL sensor fusion and actuation system.

8.5.1. Overview and Architecture

The simulation is executed in discrete time steps within a central script that handles initialization, control logic, environmental modeling, and telemetry. The architecture is modular, with each function representing a critical control or feedback subsystem. At each time step, the system computes the yaw error between the current rocket orientation and the direction to the target. This yaw error is passed through an adaptive proportional-derivative controller, which generates a servo command used to modify the rocket's trajectory. The simulation terminates when the altitude reaches zero or the time limit is reached, and it logs the full trajectory and control history for post-analysis.

8.5.2. Main Simulation Script – main.m

The main.m script acts as the primary simulation controller. It initializes all state variables including starting position, altitude, yaw, descent velocity, and wind parameters. Within a time-stepped loop, the script calculates the bearing to the target and the yaw error,

applies adaptive PD control, and updates the rocket's position and heading using a sensor-motion model. Wind drift is applied at each time step to simulate realistic environmental disturbances. The script also logs all relevant telemetry data and generates a 3D plot at the end of the simulation to visualize the trajectory, including start and end points, target location, and landing accuracy.

8.5.3. Motion and Sensor Model – get_sensor_data.m

The get_sensor_data.m function simulates both the movement of the rocket and the feedback provided by its onboard sensors. The servo command modifies the yaw rate, capped at ±120 degrees per second, and small Gaussian noise is introduced to replicate sensor inaccuracies. The rocket's position is updated based on the new heading and a fixed forward step size, while the altitude is decreased by a fixed descent rate. This function returns updated values for X and Y position, yaw, altitude, and vertical velocity, simulating the physical behavior of the vehicle during descent.

8.5.4. Target Navigation – compute_yaw_target.m and Euclidean_distance.m

These functions calculate the directional and distance-based metrics needed for navigation. The Euclidean_distance.m function computes the straight-line distance between the rocket and the fixed landing target, and returns both the distance and the bearing angle to the target. The compute_yaw_target.m function acts as a wrapper to streamline this process and provide the current yaw target. These outputs are used by the controller to determine how to reorient the rocket during descent.

8.5.5. Yaw Control – pd.controller.m

The pd_controller.m function computes the yaw correction required to align the rocket with the target direction. It uses the proportional-derivative (PD) control law, incorporating both current yaw error and its rate of change. The resulting output is a servo command constrained to ±15 degrees, reflecting the physical limits of a standard servo. This function enables the system to smoothly and responsively rotate toward the target bearing at each time step.

8.5.6. Adaptive Gain Scaling – get_adaptive_gains.m

The get_adaptive_gains.m function enhances the PD controller by adjusting its gains based on the rocket's distance from the target. At greater distances, higher gains provide aggressive correction, while at closer ranges, lower gains reduce the risk of overshooting. This dynamic tuning improves control stability and landing precision across varying approach stages.

8.5.7. Yaw Estimation Smoothing – adaptive_kalman.m

The adaptive_kalman.m function applies a one-dimensional Kalman filter to refine yaw target estimates by reducing measurement noise. It executes a predict-correct cycle using specified process and measurement noise covariances, and adjusts the estimate based on observed errors. This function is particularly useful for hardware-in-the-loop testing or noisy simulations where sensor stability is critical.

8.5.8. Telemetry Logging – log_telemetry.m

The log_telemetry.m function captures real-time simulation data and writes it to a CSV file (`telemetry_log.csv`). Logged parameters include time, yaw, yaw target, yaw error, servo

command, position, altitude, vertical velocity, and distance to the target. The function writes headers only once and ensures only valid data is logged, allowing for clean post-processing and performance analysis.

8.5.9. Statistical Analysis – monte_carlo_runner.m

The monte_carlo_runner.m script runs the RTLS simulation repeatedly under randomized conditions to evaluate landing consistency. For each run, it records the final landing distance and determines whether the result falls within the 800 ft success threshold. The script plots a histogram of landing distances and computes statistical metrics such as mean, standard deviation, and success rate. This module provides critical insight into the robustness and reliability of the RTLS guidance algorithm.

```
Monte Carlo Results (n = 100):
Mean Distance: 824.85 ft
Max Distance: 1110.85 ft
Min Distance: 524.27 ft
Std Deviation: 135.13 ft
Successful Landings (<800 ft): 37 / 100 (37.00%)
```

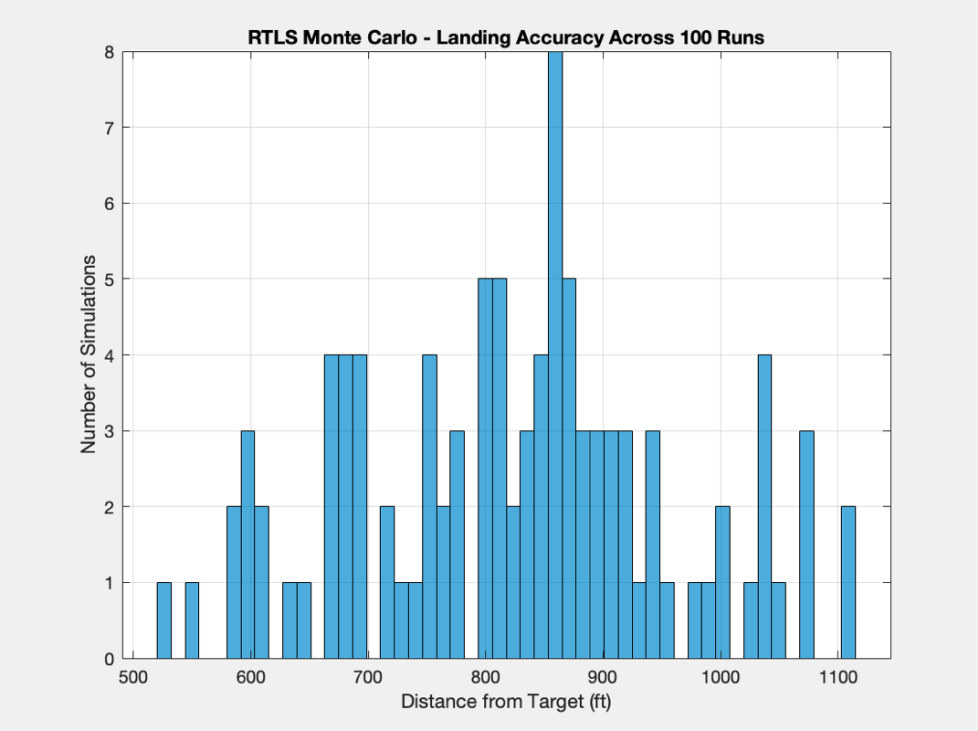


Figure 71: Monte Carlo simulation results

8.5.10. Hardware Integration

To validate the RTLS algorithm under realistic conditions, a hardware-in-the-loop (HIL) test system was implemented on an ESP32 microcontroller using FreeRTOS. This dual-core setup emulates flight conditions and provides real-time actuation and telemetry, ensuring the software control logic is compatible with the physical hardware components used during the final launch.

The system is split across two processor cores. Core 0 is reserved for sensor acquisition tasks (GPS, barometer, and IMU), while Core 1 handles the simulated descent logic and RTLS actuation. Specifically, Core 1 executes `RTLSloop()` every 100 ms, which loads a time–altitude profile from an SD card (using `Time vs Altitude.csv`) and simulates rocket descent from

apogee to the 750 ft trigger altitude. Upon reaching 750 ft, the RTLS system is activated and begins steering toward a target GPS coordinate using adaptive heading-based control.

The core guidance logic resides in rtls_controller.cpp and is called via the `updateRTLS()` function. This module calculates the bearing and distance to the target using haversine geometry, determines yaw error, and applies adaptive proportional-derivative control based on distance. The output is a servo deflection angle, clamped to ±15°, which is converted to a PWM command and actuated through the servo. The system logs all simulation events, servo commands, and state data to `FlightLog.csv` and outputs real-time telemetry via both USB serial and radio (Serial2).

The `RTLS_HIL_FinalV3.ino` sketch bootstraps both FreeRTOS tasks using `xTaskCreatePinnedToCore()` and initializes all necessary hardware including the SD card, servo motor, and deployment charge GPIOs. The dual-core separation ensures that the simulated descent can proceed independently from real-time sensor validation, enabling consistent, repeatable testing. While Core 0 was not used to feed live sensor data into the RTLS logic for this version, it was essential in verifying the electrical and software integrity of the sensor stack prior to full deployment.

Ultimately, this setup provided a critical testbed that confirmed the RTLS flight algorithm could respond in real time to altitude-based events, generate directional corrections based on GPS-derived position and yaw, and reliably actuate the parafoil control surfaces through servo movement. This simulation-driven testing approach bridged the gap between software simulation and full field deployment, contributing to a successful RTLS demonstration.

8.6. Recovery System

8.6.1 Overview and Design Methodology

The recovery system is designed to ensure the safe retrieval of the rocket following flight. It combines conventional recovery elements such as black powder ejection charges and shock cords with a novel paraglider-based descent mechanism and an electronically controlled dual-deployment strategy. The system is centered around the Featherweight Synapse170 deployment controller, enabling precise timing of ejection events and redundancy in recovery sequencing.

Key recovery components include FFFg black powder, custom ejection capsules, nickel-chromium wire igniters, Kevlar shock cords, and the Synapse170 flight computer. The design process prioritized safety, simplicity, and reliability to ensure consistent deployment with minimal failure risk.

8.6.2 Recovery Component Specifications

Each component of the recovery system was selected based on strength-to-weight performance, reliability under dynamic loads, and compatibility with the overall airframe design. Key components include:

- **Synapse170 Deployment Controller**: A precision altimeter-based dual-deployment controller responsible for triggering both apogee and main ejection events. Offers redundant pyro channels and built-in safeguards against false triggering.
- **Kevlar Shock Cord:** High-strength, heat-resistant material used to absorb shock loads during ejection and recovery (10 feet drogue; 15 feet main). Secured to internal hardware using ¼-20 eye bolts.

- **FFFg Black Powder:** Granulated black powder used in ejection charges for initiating deployment of the recovery system. Charge mass was selected based on calculated volume and pressure required for clean separation.
- **Ripstop Nylon Streamers:** Durable, lightweight streamers used to slow descent and increase visibility. Chosen for their resistance to tearing and high drag efficiency.
- **Nickel-Chromium Wire Igniters:** Custom-wound igniters used to initiate the black powder charges. These were fabricated manually to ensure consistent burn characteristics and reliable ignition.
- **Ejection Capsules:** 3D printed enclosures used to contain and direct the ejection charge. Designed for easy packing and consistent orientation during deployment.
- **¼-20 Eye Bolts:** Steel fasteners providing secure anchor points for the shock cord within the airframe.
- **Kevlar Blanket:** Flame-resistant sheath wrapped around the recovery components to protect them from heat and debris generated during ejection.

8.6.3 Manufacturing and Assembly

Most components were sourced from certified vendors compliant with high-power rocketry standards. The Synapse170 controller was mounted internally within the electronics bay and pre-programmed with altitude-based deployment thresholds. Wiring for the igniters was routed to the controller's pyro output terminals to ensure reliable signal transmission. Igniters were manually assembled using nickel-chromium resistive wire and insulated leads—allowing for precise control over ignition timing. Mechanical components such as eye bolts and shock

cord loops were affixed to 3D-printed internal mounts, reinforced to handle dynamic loads. The shock cords were routed internally and tied to these mounts to create a robust and continuous recovery linkage. Special care was taken to eliminate friction points and minimize the potential for entanglement or abrasion during deployment.

8.7. Propulsion System

8.7.1 Overview and Design Methodology

The propulsion system is a critical subsystem designed to securely house the rocket motor and effectively transfer thrust loads to the airframe. This design prioritizes reusability, mechanical robustness, and precise load distribution. While the motor itself—an Aerotech I366—was purchased off-the-shelf, the surrounding motor assembly and integration hardware were fully designed and customized by the team. The propulsion system serves not only as the mounting interface for the motor but also as a structural element capable of handling dynamic forces throughout the launch and ascent phases. Special consideration was given to vibration resistance, axial load transfer, and modularity for future launches.

8.7.2 Propulsion Component Breakdown

The propulsion system was integrated directly into the fin can assembly to ensure a compact and structurally unified rear section. Key design features include:

- **Eight M3 Bolts:** A ring of M3 bolts secures the motor mount to the surrounding structure. These fasteners distribute mechanical loads evenly, resist vibrational loosening, and provide redundancy in case of individual fastener failure.
- **Ring Stiffener:** A rigid structural ring is used to transfer thrust loads from the motor tube to the surrounding body tubes. This component enhances structural stiffness and reduces stress concentration.
- **Tapered Motor Tube:** The motor tube geometry includes a taper that minimizes lateral movement of the rocket motor during acceleration, maintaining alignment and flight stability.

- **Interchangeable Motor Tube Design:** The propulsion system supports modular motor tubes, allowing the rocket to be flown with both 29mm and 38mm motors. This provides flexibility for different flight profiles and launch conditions.
- **3D Printed Motor Retainer:** A custom motor retainer, sourced from a design by Alex Ward (via Printables), was 3D printed to secure the motor in place.
- **AeroTech I366R Motor:** This rocket is powered by an AeroTech I366R, a high-power solid composite reloadable motor featuring…
    - Total Impulse: 539 N·s (Newton-seconds)
    - Average Thrust: 366 N
    - Peak Thrust: 507 N
    - Burn Time: 1.0 second
    - Motor Class: High-power “I” class (HPR certified by NAR, TRA, and CAR)

8.7.3 Manufacturing and Assembly

The motor assembly components—including the motor tube, ring stiffener, and mounting interfaces—were fabricated using a Prusa XL 3D printer. Material selection focused on high-strength, heat-resistant filaments suitable for handling mechanical stress and brief thermal exposure during motor burn, so PETG was selected.

All components were assembled by hand and thoroughly inspected for tolerance fits and alignment. The complete propulsion system was then secured to the rocket’s airframe using the integrated fastener ring, ensuring a rigid and vibration-resistant connection. The assembly process was designed for repeatability, allowing for straightforward disassembly and reconfiguration for future flights.

Page intentionally left blank.

9. System Evaluation

9.1. Airframe Stability Test

What is this test for:

Check the stability level of the rocket by measuring the Center of Gravity (CG) position in the rocket and subtracting it from the Center of Pressure (CP) obtained from open rocket simulations.

Performance Requirements:

The rocket must maintain a caliber similar to the OpenRocket's caliber (1.8 cal) with a Safety factor of 1.2.

Tolerance and Sensitivity:

1.8 cal $\pm$ 0.36 cal

Reliability and Safety:

Poor stability of the rocket may result in the rocket losing control during flight, which can cause major complete destruction of the rocket. Proper stability is key to maintain a stable flight path until the recovery system deploys.

Failure Modes:

If the rocket is not like the OpenRocket model, the model will have to be reevaluated to obtain a result more like the actual product. If the stability is too unstable (less than 1 cal or more than 2.5 cal), the rocket will have to be redesigned to improve stability.

Testing Procedures:

1. Setup up the rocket to be ready for launch.
2. Balance rocket over your hand and see what point the rocket balances and mark it.
3. Subtract the position of the CG and CP and divide it by the diameter of the rocket

Test Results:

The CG was measured to be 30.5 in from the rocket tip, and the CP is 37.696in from the rocket tip. CP – CG = 7.19in. The diameter of the rocket is 4in. The stability metric is found by...

$$\frac{CP - CG}{4in} = 1.799\ cal$$

The measured stability is almost identical to the one calculated by open rocket (1.8 cal). This ensures a strong likelihood of a stable flight, given that weather is favorable.

9.2. Deployment Ground Test

What is this test for:

This test was conducted to ensure the recovery deployment system is functioning as intended. This test was conducted on the ground in case of a failure, which would be catastrophic in-flight.

Performance Requirements:

1. Electronics system must detect apogee and automatically deploy the drogue chute
2. Electronics bay must detect an altitude of 750 feet and deploy the main parachute

Tolerance and Sensitivity:

1. Drogue deployment must occur $\leq\leq$ 50 feet post-apogee
2. Main deployment must at occur ± 50 feet of target altitude

Reliability and Safety:

This test is crucial to the safety of surrounding people and the vehicle. A deployment failure will result in a “lawn-dart” scenario, where the rocket falls toward the ground with nothing to slow it down. This would certainly destroy the rocket, and severely harm someone if they were hit.

Failure Modes

1. Drogue deployment fails due to failed ejection charge, but main successfully deploys
    a. Outcome: Main could fail due to extremely high snatch force

2. Drogue successfully deploys, but main fails due to failed ejection charge
    a. Outcome: Rocket would hit the ground at too high of a speed, severely damage components, and possibly injure people in the area
3. Both ejection charges fail
    a. Outcome: Rocket would accelerate toward the ground to terminal velocity, resulting in destruction of the rocket and possible injury to those around
4. Drogue successfully deploys and main ejection charge successfully ignites, but rocket does not separate
    a. Outcome: Rocket would hit the ground at too high of a speed, severely damage components, and possibly injure people in the area. Also, the main parachute would likely be burned by the black powder gases
5. Drogue ejection charge successfully ignites, but rocket does not separate and main successfully deploys
    a. Outcome: Rocket would hit the ground at too high of a speed, severely damage components, and possibly injure people in the area. Also, the drogue parachute would likely be burned by the black powder gases
6. Both ejection charges ignite, but rocket does not separate
    a. Outcome: Rocket would accelerate toward the ground to terminal velocity, resulting in destruction and possible injury to those around

Testing Procedures:

1. Arm electronics bay
2. Measure appropriate level of black powder and load it into ejection canister
3. Install main parachute and fireproof blanket into main bay
4. Install drogue parachute and fireproof blanket into drogue bay. Secure nose cone and attach shear pins
5. Attach shop-vac to electronics bay pressure hole
6. Plug in vacuum from a safe distance
7. Unplug vacuum after drogue deploys. Main will deploy as pressure vents back to atmosphere.

Test Results:

Both ejection charges went off but failed to deploy the 1st and 2nd stage recovery systems. The first ejection charged was able to move the nose cone slightly but not enough for it to release fully. The second ejection failed to break the shear pins and deploy the main recovery system. Both failures are likely to be due to insufficient black powder to create a strong enough force to deploy the systems. To correct the errors found through this test, the black powder was doubled before launch.

9.3. Body Tube Compression Test

What is this test for:

Establish maximum compression load limits for 3D-printed body tubes. Compression is the dominant load that the vehicle will experience throughout flight. This test is being conducted to make sure the vehicle can withstand flight loads

Performance Requirements:

Body tubes can withstand max compression forces during flight. These forces are found by using data from open rocket to solve the equation below.

$$F_{compression,max} = G_{max}W$$

$$F_{compression,max} = 104\ lbf$$

Tolerance and Sensitivity:

A factor of safety of 1.5 will be applied to account for variances in flight acceleration and structural imperfections

Reliability and Safety:

A body tube failing would likely result in complete mission failure. The rocket would experience rapid unscheduled disassembly during flight.

Failure Modes:

1. Body tubes fail in crippling
2. Body tubes fail in buckling

Testing Procedures:

1. Place body tube onto solid, flat surface
2. Load weights onto body tube in increments every 5 seconds until double the maximum compressive load is reached.

Test Results:

The body tube was put under a maximum load of 150lbf and withstood the force with no visible deflection. From that point the was no further weight added due to the body tube supporting well over the 104 lb it would go under.

9.4. Paraglider Drop Test

What is this test for:

Measuring the descent speed of the paraglider under a weight of 6 pounds

Performance Requirements:

Paraglider must deploy in time before landing, and maintain stability while landing below 25ft/s.

Tolerance and Sensitivity:

Paraglider cannot exceed 25ft/s and must fully deploy before landing.

Reliability and Safety:

Excessive drop speed from the paraglider may cause damage to the airframe and the electronics of the rocket. It can also possibly cause injury to people near the rocket's landing.

Failure Modes:

1. If the rocket falls above 25ft/s it can cause damage to the rocket's components.
2. If deploy time is too sudden the snatch force may be too high and tear the fabric of the paraglider
3. If deployment time is too long it may result in a hard landing, damaging the rocket and its components.

Testing Procedures:

1. Attach a weight to the paraglider that is about the same weight as the rocket (6lbs).
2. Drop the paraglider from a height that allows the paraglider enough distance to stabilize before landing. (50ft)
3. Use a stopwatch to time how long it takes for the parachute to deploy.
4. Analyze data calculated from accelerometer.

Test Results:

When released the paraglider had a complete aerodynamic collapse when deployed, resulting in an unstable, rapid descent to ground level. It fell at around 30ft/s above the maximum landing velocity desired. This data concludes that the paraglider is not safe for launch and another method must be used to ensure safe recovery during launch.

Solutions:

Increase the size of the paraglider:

Upgrading to a larger or more robust parafoil rated for the expected payload (5.5 lbs or more) ensures that the glide and descent characteristics stay within safe limits.

Use an alternative emergency recovery system:

In the worst-case scenario, if repeated failures occur, switching to a conventional round parachute that has more stability may be necessary to guarantee a safe recovery. Although the round parachute would not be very effective with the guiding system, it can be deployed if there is a failure with the paraglider to mitigate damage done to the rocket.

9.5. Ripstop Nylon Material Strength Test

What is this test for:

To measure the strength of the materials used to construct both the streamers and paraglider

Performance Requirements:

Nylon must remain intact under the highest forces experienced during flight (snatch force)

Tolerance and Sensitivity:

Snatch Force = 5 * Weight

Reliability and Safety:

Paraglider and streamer material not being able to withstand deployment forces can cause the rocket vehicle experiencing rapid descent. This would be a total system failure.

Failure Modes:

If the paraglider material fails below 30lbs it will result in a failed recovery.

Testing Procedures:

1. Attach a 6” wide piece of the ripstop nylon which is used in paraglider to the end of a force gauge
2. Pull on material until the force gauge is measuring maximum force (500N/112lbf), or the material rips

3. Observe for any warping or tearing

Test Results:

The ripstop nylon was able to withstand the maximum force allowed on the force gauge which is 112.4 lbf. The paragliders ripstop nylon has more than enough strength to withstand the froes experienced during flight. These results deem it durable enough for the forces it will experience.

9.6. Battery Longevity Test

What is this test for:

The purpose of conducting this test is to verify the endurance of the onboard battery under full system conditions. A battery longevity test is crucial, as it ensures that the battery has the capabilities to sustain all the components of the electronics bay for the entirety of the mission timeline. The battery must be able to continue powering the electronics bay not only for the duration of the recovery aspect of the mission, but for pre-launch and post-launch tracking as well. This test will simulate how long the RTLS electronics bay can be powered for with the battery of choice.

Performance Requirements:

1. The battery must provide power for at least 2.5 hours of continuous operation
2. Voltage must remain above minimum safe cutoff during the entire test (3.3V)
3. All electronics, including telemetry, sensors, and control systems, must function without interruption during the test.

Tolerance and Sensitivity:

1. Minimum acceptable runtime: 2.5 hours
2. Ideal operational buffer: >= 5 hours
3. System behavior must remain stable and responsive at low voltage levels until automatic shutdown

Reliability and Safety:

Battery life is crucial for maintaining reliable operation of mission criteria such as telemetry, control, and the recovery systems during the entirety of flight. A power failure mid-flight could half telemetry, disable recovery logic, and render onboard data unrecoverable. Ensuring that the battery system is appropriately sized prevents in-flight cutoffs and system resets.

Failure Modes:

1. Battery drains before mission completion
    a. Outcome: Potential data loss or failure of recovery system activation
2. Battery voltage drops below cutoff mid-mission
    a. Outcome: System resets or powers off. Loss of flight data
3. Servo actuation causes voltage dips under load
    a. Outcome: Brownout conditions, reduced reliability.
4. Battery overheats or swells due to over-discharge
    a. Outcome: Permanent damage and potential safety risk.

Testing Procedures:

1. Fully charge battery to manufacturer specifications
2. Connect entire electronics suite as assembled in final payload configuration
    a. Components include:
        i. ESP32 Microcontroller
        ii. GPS module
        iii. Barometer

iv. LoRa radio transceiver

v. SD card logger

3. Power the system using the onboard battery
4. Begin timing the test at system power-on using a stopwatch
5. Allow the system to run until the battery is fully depleted and the electronics shut down
6. Record total operational time and note any behavioral anomalies such as voltage drops, resets, or interruptions

Test Results:

Total Runtime: Approximately 11 hours

The electronics suite functioned nominally throughout the test, with all systems remaining online and responsive for the duration of the battery discharge. Telemetry continued to transmit and servo actuation was logged throughout the entire run. Shutdown occurred gradually as battery voltage fell below system thresholds. The battery system significantly exceeded the mission requirement of 2.5 hours, providing an operational margin of over 4x the minimum duration. This validates that the current battery configuration is more than sufficient for full flight operations, including pre-launch preparation, descent phase, and extended post-landing tracking if needed. No unexpected system resets or failures were observed.

9.7. Radio Communication Range Test

What is this test for:

This test is conducted to ensure that the radio telemetry system can maintain a stable connection with the ground station over the required operational distance. During flight, the rocket will transmit real-time telemetry including GPS location, barometric altitude, and system state. If communication is lost at any point during descent, recovery efforts may be delayed or jeopardized.

Performance Requirements:

1. Maintain a stable, bidirectional communication link up to 1 mile minimum.
2. RSSI values must remain above the manufacturer's minimum threshold.
3. No more than 10% packet loss at maximum range.

Tolerance and Sensitivity:

1. Communication range must be ≥ 1 mile, with an optimal goal of 6 miles.
2. RSSI ≥ -110 dBm considered acceptable for reliable reception

Reliability and Safety:

Ensuring a consistent radio connection is essential for ground-based tracking of the rocket’s landing site. Failure to maintain signal could result in lost equipment and loss of flight data.

Failure Modes:

1. Radio fails to initialize or pair with ground station
    a. Outcome: Entire system inoperable for telemetry and tracking.
2. Signal degradation occurs prematurely before target distance
    a. Outcome: Shortened range; recovery data may be lost.
3. Packet collisions or dropped signals due to environmental interference
    a. Outcome: Inconsistent tracking or data loss.

Testing Procedures:

1. Upload test firmware to ESP32 that sends a “Ping” message over LoRa every 2 seconds.
2. Mount transmitting radio on an elevated surface (UCF Libra Parking Garage) to simulate rocket elevation.
3. Using a laptop and Hterm software, log packet reception in a vehicle equipped with the receiver module.
4. Drive at a constant speed (~20mph) away from the elevated surface down an adjacent road.
5. At each interval, record:
    a. GPS location

   b. RSSI value
   c. Number of successfully received packets out of 10

6. Continue testing until signal is no longer reliable (packet loss > 50%).

7. Record and analyze RSSI degradation across the route

Software and Hardware Used:

1. Transmitter: Microcontroller paired with LoRa module.

2. Receiver: Identical LoRa module connected to a second USB-to-UART bridge.

3. Development Environment: Arduino IDE for uploading test firmware and serial communication

4. Logging Interface: Hterm serial terminal used to monitor incoming packets in real-time and assess signal quality (RSSI) and error handling

5. Power Source: Battery-powered ESP32 for transmitter, ground station connected to laptop USB

Test Results:

Location: UCF Libra parking Garage

The test was conducted by taping the transmitting LoRa radio module to the top floor of the UCF Libra parking garage (Figure 72), simulating the rocket's airborne elevation. A vehicle

equipped with the ground station receiver and a laptop was then driven down the adjacent road at a consistent speed of 20 MPH, logging live telemetry data using Hterm.

Figure 72: Radio Test Setup

The radio began by transmitting a “Ping” packet every 2 seconds via LoRa, which was successfully received by the ground station as the vehicle moved away. Reception quality gradually declined, and after reaching approximately 0.3 miles (1584 feet) from the transmitter, the connection was lost.

The text below represents sample packets received during the test:

+OK<\r><\n>

+RCV=1,7,Working,-145,-71<\r><\n>

+RCV=1,7,Working,-133,-26<\r><\n>

+RCV=1,7,Working,-123,-11<\r><\n>

+RCV=1,7,Working,-109,7<\r><\n>

+RCV=1,7,Working,-123,-5<\r><\n>

+RCV=1,7,Working,-131,-21<\r><\n>

+ERR=12<\r><\n>

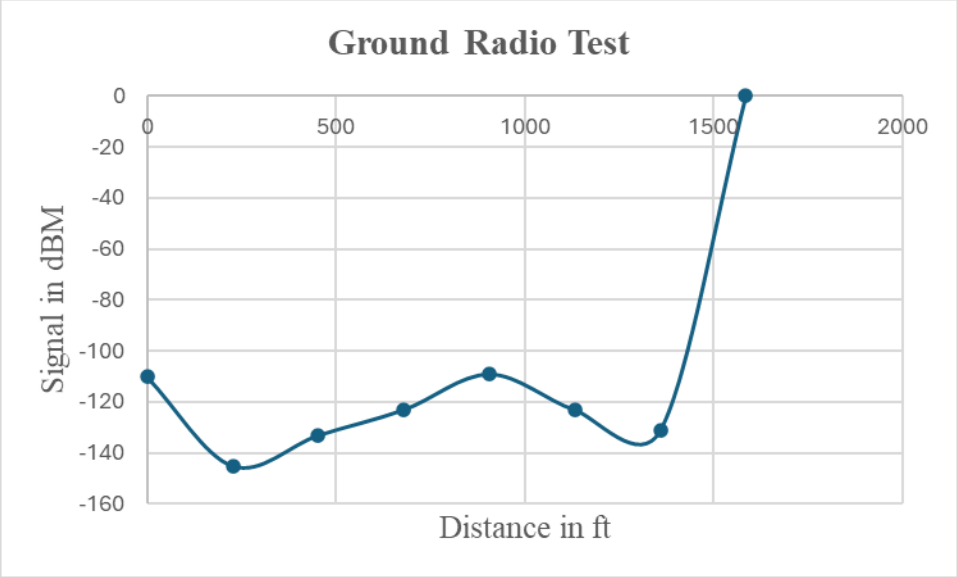


Figure 73: Signal vs distance for ground radio test

RSSI values recorded ranged from -109 dBm (strong signal) to -145 dBm (signal drop-off threshold). As seen in Figure 73, beyond -133 dBm, packet delivery became erratic and eventually failed completely.

The test confirmed that the radio telemetry system is capable of maintaining stable communication up to 0.3 miles under semi-urban conditions with line-of-sight limitations (. While functional at close ranges, this falls short of the 1-mile minimum range set by project performance requirements.

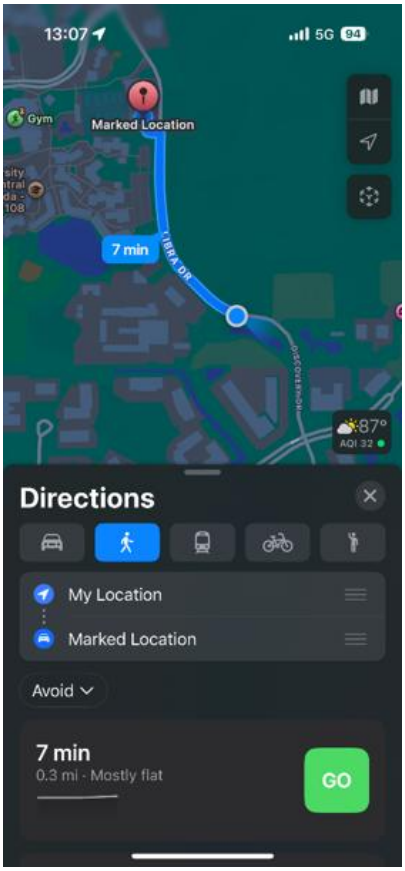


Figure 74: Map of radio range test

Factors contributing to early signal loss may include:

- Low antenna gain
- Ground clutter and building reflections

- Radio module configuration not optimized for long-range

Future Recommendations:

- Repeat the test using higher-gain or directional antennas.
- Conduct range trials in an open field or rural environment to reduce interference.
- Investigate LoRa module settings such as spreading factor, bandwidth, and output power to improve range.
- Elevate both the transmitter and receiver to further minimize obstacles and simulate high-altitude descent conditions.

Despite the shortfall in range, the test validated the system's baseline functionality and provided valuable insight into signal behavior over distance. Future tests would focus on refining hardware and signal tuning to meet or exceed the 1-mile communication range required for flight-readiness.

9.8. Hardware in the Loop (HIL) Testing

What is this test for:

This test was conducted to validate the RTLS algorithm's behavior on embedded hardware using pre-recorded flight data. Hardware-in-the-Loop (HIL) testing allows the RTLS logic to be verified in real-time without requiring live sensor input or an actual launch. The test confirms system functionality including servo actuation, algorithm logic sequencing, and telemetry output based on flight simulation data.

Performance Requirements:

1. The algorithm must successfully read simulated flight data from a CSV file in real time.
2. Servo response must correspond to RTLS yaw error correction logic
3. LED indicators for black powder events must activate at correct altitude thresholds
4. The system must transmit GPS, yaw, and altitude data via radio telemetry

Tolerance and Sensitivity:

1. RTLS activation must occur at or near 750 ft AGL
2. Servo commands must update based on computed yaw error and reduce as distance to target decreases
3. CSV parsing must maintain real-time pace with consistent altitude steps

Reliability and Safety:

HIL testing provides an essential validation step for embedded control systems before integration into the full rocket. By simulating descent using actual flight profiles and outputting physical control signals, the team can verify that the RTLS logic responds predictably and safely under time-driven conditions. This avoids costly errors during flight and confirms servo functionality under deterministic logic.

Failure Modes:

1. CSV altitude readings fail to update or feeze
    a. Outcome: RTLS algorithm does not activate; servo remains static
2. Servo oscillates excessively or responds incorrectly
    a. Outcome: Faulty yaw correction; unstable descent control
3. LED indicators do not trigger at altitude thresholds
    a. Outcome: Incorrect timing for simulated recovery system events
4. Telemetry does not appear in Hterm
    a. Outcome: Loss of GPS/yaw/altitude tracking at ground station

Testing Procedures:

4. Configure ESP32 to run the corresponding .ino file on Core1 using xTaskCreatePinnedToCore(). Core0 is disabled
5. Place Open Rocket flight data (down sampled) onto SD card as a CSV file with altitude as the triggering variable
6. Connect servo motor to flight pin and two LEDs to simulate BP1 and BP2 ejection triggers

7. Simulated logic:
    a. BP1 (apogee) LED activates around 2200 ft AGL
    b. RTLS algorithm activates at 750ft AGL
    c. Servo yaw command is calculated from GPS and yaw angle to fixed target
    d. BP2 LED activates when simulated altitude drops below 700 ft
8. All telemetry (GPS, yaw, altitude) is sent to the ground station over Serial2 and monitored using Hterm
9. Evaluate servo movement consistency, LED trigger accuracy, and telemetry output timing

Test Results:

1. The servo motor moved responsively according to RTLS yaw correction commands. Movements were aggressive at large heading errors and gradually reduced as the simulated rocket approached the target.
2. The first black powder LED (BP1) was triggered near 2200 ft as expected. The second LED (BP2) activated consistently when altitude dropped below 750 ft.
3. CSV altitude parsing functioned correctly for the full descent. Prior issues with data skipping below 750 ft were resolved before this test.
4. HTerm displayed live output of GPS coordinates, altitude, and yaw at 2-second intervals without delay.
5. The simulated trajectory brought the vehicle within ~10–15 meters of the target by 500 ft AGL. Servo output reduced to 0 degrees near this point, indicating successful convergence on the landing site.

The HIL test demonstrated successful real-time integration of the RTLS algorithm into embedded hardware. All mission-critical components, including servo control, ejection triggers, and telemetry output, performed as intended when driven by pre-recorded flight data. The test confirmed that the algorithm can actuate the servo in response to dynamic GPS and yaw inputs with high repeatability. With this validation, the RTLS system is approved for integration into future closed-loop testing and field demonstrations.

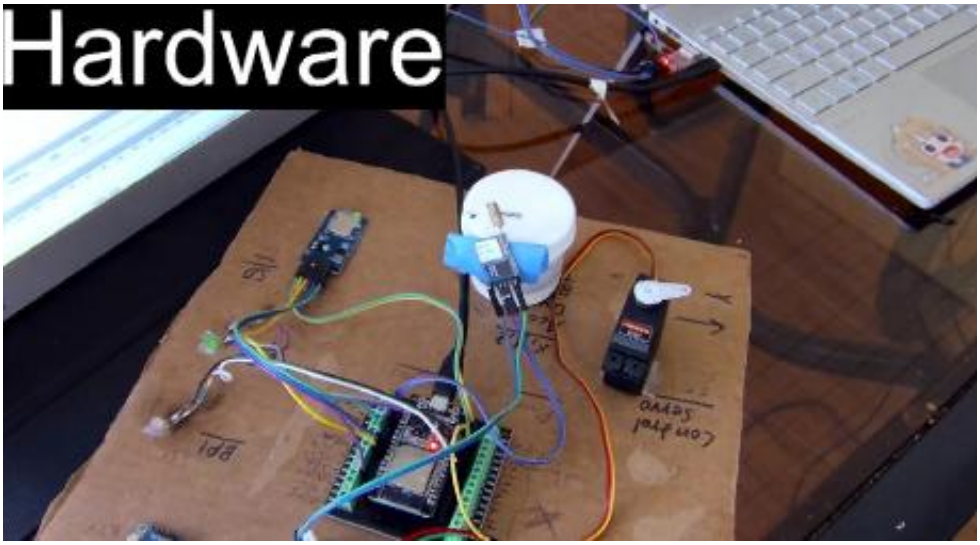


Figure 75: HIL Hardware

10. Significant Accomplishments and Open Issues

10.1. Accomplishments and Results

In Section 2, customer needs, project goals, and team objectives were outlined. The purpose of this section is to reference back to those and report on their success or failure.

10.1.1. Customer Needs

Customer Requirement	**Success/Failure**
Rocket shall follow all NAR and SRA safety standards and guidelines.	Success
Rocket shall reach an apogee of 2,200 feet AGL.	Failure
Rocket shall reach a target no further than 150m from the launch site.	Failure
Rocket shall land no further than 800 feet from the target.	Failure
Rocket shall not be actively guided during the ascent phase of flight.	Success
Rocket shall use an electronic ignition system.	Success
Rocket shall not use a combustion landing system.	Success
Rocket shall not produce more thrust than its weight on descent.	Success
At least one member shall be L1 certified.	Failure

Rocket shall not use a "Sparky" motor.	Success
Motor impulse shall not exceed Level 1.	Success
Rocket shall record FPV during descent and should transmit data to a ground station.	Successfully recorded video, failed at transmission
Rocket should have a sound emitting device (screamer) on board.	Success

Table 10: Customer Requirements Compliance

10.1.2. Project Goals Results

Develop a Dual Deployment Rocket with RTLS Integration:

The vehicle developed is capable of dual deployment with an RTLS system. Ground testing results showed that both ejection charges will trigger automatically during descent – drogue at apogee, and main at 750 feet. The electronics suite can easily be wired to a servo, and the main parachute bay can house a paraglider.

Conduct a Successful Flight Test:

A flight test was conducted, but deployment failed for both drogue and main despite charges successfully deploying during ground testing. Extra black powder beyond the calculated values was added to both the drogue and main charges to ensure deployment, as ground testing showed black powder mass was insufficient for deployment. Despite this, the deployment system still failed. Due to the vehicle being lost with no communications, a failure mode cannot be concluded. It is suspected that the altimeter was damaged during ground testing, as the barometer is sensitive to rapid changes in pressure.

Collect Real-Time Telemetry from a Customized Electronics Bay:

The electronics bay was successfully developed to support full real-time telemetry collection, a capability that was demonstrated through extensive ground testing using a custom-built development environment. Core subsystems—including GPS, IMU, SD card logging, radio transmission, and black powder deployment control—were fully integrated and verified during bench testing.

Demonstrate RTLS Algorithm Performance via Monte Carlo Analysis:

To assess the reliability and robustness of the RTLS (Return-to-Launch-Site) algorithm under realistic and variable descent conditions, a Monte Carlo analysis was conducted using MATLAB. A total of 100 randomized descent trials were simulated, each introducing variability in drop location, altitude, and wind conditions. The objective was to evaluate how effectively the flight algorithm could guide the vehicle to land within the 800-foot target zone mandated by competition rules.

The RTLS algorithm employed adaptive PD control based on the real-time heading angle between the rocket's current position and a fixed landing coordinate. Each simulation trial tracked the final ground distance from the landing point to the target, with results compiled to analyze accuracy and consistency.

The statistical results of a sample run of the simulation are summarized as follows:

- **Mean Distance from Target**: 824.85 ft
- **Minimum Distance**: 524.27 ft
- **Maximum Distance**: 1110.85 ft

- **Standard Deviation**: 135.13 ft
- **Successful Landings (<800 ft)**: 37 / 100 (37.00%)

A histogram of the landing distances demonstrates a broad spread of outcomes, with most landings clustering between 700–950 ft from the target. While the algorithm achieved some degree of consistency, the success rate of 37% reveals that further refinement is needed, particularly under adverse wind and offset conditions.

These results indicate that although the RTLS guidance logic is functional, additional improvements to the adaptive gain tuning, heading correction logic, or wind estimation model are necessary to consistently meet the performance threshold. Nonetheless, the Monte Carlo analysis provided valuable insight into control stability and established a clear performance baseline for future embedded testing.

Hardware-In-the-Loop (HIL) Testing

This test was conducted to validate the RTLS algorithm's behavior on embedded hardware using pre-recorded flight data. Hardware-in-the-Loop (HIL) testing allows the RTLS logic to be verified in real-time without requiring live sensor input or an actual launch. The test confirms system functionality including servo actuation, algorithm logic sequencing, and telemetry output based on flight simulation data.

Dev

10.2. Design Changes

10.2.1. Software – Hardware Integration

As the RTLS project progressed from simulation to real-world implementation, the software system underwent several critical design changes to accommodate new requirements, hardware limitations, and observed performance issues.

The most substantial shift was the transition from single-threaded MATLAB simulation to a dual-core embedded architecture on the ESP32 microcontroller. Initially, all control logic, sensor processing, and telemetry operations were run within MATLAB using static input parameters. However, once Hardware-in-the-Loop (HIL) testing began, the software was divided across ESP32's two cores using the C-++ function, `xTaskCreatePinnedToCore()`. Core 0 was originally designated to handle sensor initialization and real-time data acquisition, while Core 1 executed the RTLS guidance logic, black powder trigger simulation, and servo actuation. This design allowed parallel execution and reduced latency in timing-sensitive operations.

Later in the development cycle, Core 0 was fully removed from the system following the loss of the electronics bay during the final launch. With no access to live sensors, the team pivoted to a CSV-driven simulation framework running exclusively on Core 1. This new framework parsed flight log files in real-time to emulate descent conditions and continued to actuate the servo, LED indicators, and simulated telemetry output without needing physical sensor input. This decision preserved the ability to demonstrate the algorithm's logic and performance in the absence of full hardware capability.

Another major design evolution was the inclusion of adaptive gain scheduling within the RTLS PD controller. Early versions of the algorithm used static proportional and derivative gains, which often led to overshoot or sluggish heading correction under varying wind magnitudes. The adaptive controller dynamically scaled its response based on heading error, significantly improving convergence rate and descent stability during simulation.

To aid in debugging and improve real-time feedback, visual indicators were added to the software. These included LED triggers for apogee (BP1) and 750 ft (BP2) events, allowing the team to physically verify state transitions during HIL tests. These were essential for validating timing accuracy given the reliance on CSV altitude parsing.

Lastly, the telemetry system underwent reconfiguration, with outputs redirected to Serial2 for radio transmission. While the system successfully printed data to the ESP32's serial bus, unresolved issues in baud rate matching and signal transmission prevented consistent reception on the ground station. Though not fully resolved due to improper calibration of measurement devices, the radio was lost. This reconfiguration laid the groundwork for future telemetry integration.

In summary, the software architecture evolved significantly to adapt to real-world constraints. From redesigning control task distribution to implementing fallback simulation tools and adding adaptive control logic, the system matured into a flexible and modular framework capable of functioning even under partial hardware failure scenarios.

10.2.2. Hardware Changes

Hardware changes were divided into power supply and control system upgrades. The team initially selected a 7V main battery, which was sufficient for running the servos, microcontrollers, and sensors. However, testing showed it could not reliably ignite the black powder charges. To resolve this, a separate 9V battery was added specifically for the ignition system, ensuring consistent and independent charge deployment during flight.

After M3, a major control system redesign replaced the original dual-microcontroller setup—Teensy and Raspberry Pi—with a single ESP32. This change simplified the hardware, reduced system complexity, and allowed for dual-core processing to handle all functions including sensor data, logging, and communication, without compromising performance.

10.2.3. Motor Changes

The initial propulsion plan for the L1 rocket involved the use of an I285 motor to achieve the target apogee of 2200 feet. However, the electronics bay weight was undershot by around half a pound in original estimates. To ensure the rocket met the apogee requirement, the I366 motor was selected.

10.2.4 Paraglider changes

Originally, the recovery system for the rocket was designed around the Synapse 170 kite, intended to function as a guided parafoil controlled by a servo motor for return-to-launch-site capability. However, during testing, the kite experienced complete aerodynamic collapse upon deployment, leading to rapid and unstable descent. These results indicated that the Synapse 170 lacked the necessary stability and reliability for safe recovery under the expected flight conditions.

10.3. Project Issues

This section will outline all the major challenges the team encountered during both Senior Design I and Senior Design II. These include issues with communication with administration, budget issues, design errors, and manufacturing failures.

10.3.1. Paraglider Selection

One of the earliest and most significant issues was the mistaken assumption that the paraglider purchased would provide load-bearing capabilities. In reality, it was a recreational kite designed for aerodynamic lift but not structural support. The team initially selected it due to cost constraints. A true load-bearing paraglider designed for this application was almost 300 dollars, nearly half the team's total budget. By the time the discrepancy was identified, the recovery system had already been integrated into the vehicle design, making a full redesign impractical.

Although the team considered alternatives such as custom fabricating a canopy, limitations in time, materials, and technical capability made this approach unfeasible. The solution was launch with a conventional parachute and drogue system, which sacrificed steerability but maintained the core requirement of recoverability. Launching with the paraglider system would induce unnecessary risk to the safety of vehicle and surrounding people.

10.3.2. Procurement Challenges

Procurement-related issues significantly impacted the project timeline and technical development. A 60-foot streamer ordered in late February failed to arrive in time for integration and testing, which prevented the team from validating the drogue deployment system. For this reason, a 24" drogue parachute was used for the test flight. Additionally, a shock cord order was

incorrectly fulfilled, with only 10 feet delivered instead of the requested 25, which compromised recovery layout and forced late-stage modifications.

Along with procurement problems, multiple components—including the motor tube, servo, and force gauge—were marked as in stock by the senior design lab but were unavailable at the time of checkout. These setbacks introduced redesign requirements and forced the team to rerun OpenRocket simulations, losing valuable development time.

10.3.3. Administrative and Communication Issues

Communication gaps between the project team and university administration created significant delays and inefficiencies. Scheduled launch opportunities were missed due to weather-related cancellations. February due to wind, March due to wildfires, and a final opportunity in April was discovered informally through another advisor, just two days before the event. This resulted in a rushed and condensed integration effort that compromised preparation quality. Further, the senior design lab's inventory records were not consistently updated, leading the team to make critical design decisions based on components that were ultimately unavailable. Additionally, the propulsion system was initially designed without access to a complete motor inventory. These issues underscore the importance of accurate and timely information sharing between administrative bodies and student teams.

10.3.4. Electronics Integration and Hardware Failures

In the final days of assembly, multiple avionics issues emerged that significantly impacted system capability. Two Berry IMUv3 sensors were lost—one due to reversed polarity during soldering, and the other likely due to mishandling caused by impaired judgment following exposure to lead-based solder. These failures occurred too late for replacements, resulting in the

loss of real-time orientation data for the flight. Consequently, the rocket relied solely on GPS and barometric altitude data.

The ESP32 microcontroller also exhibited unexpected electrical behavior, including a high-current surge when powered simultaneously via USB and battery. This anomaly, undocumented in official datasheets, nearly damaged the programming computer and raised safety concerns. Additionally, several commercial components underperformed, most notably the RYLR896 LoRa module, which failed to meet its specified range during testing. To mitigate this, a radio range extender was integrated at the last minute to maintain telemetry integrity. A final critical issue arose with the ejection charge system—following the loss of the IMUs, a backup altimeter was hastily integrated, but continuity and sequencing checks were incomplete prior to launch despite limited vacuum testing.

10.3.5. Propulsion System and Mounting Rework

During the final stages of launch preparation, our team encountered significant challenges related to motor procurement and approval. Initially, when attempting to pick up the previously checked-out motor from procurement, we were incorrectly informed that it was unavailable. Upon follow-up, another employee confirmed the motor was in fact available and issued it to us.

However, the day before launch, we were unexpectedly informed that the reusable motor system (I-366) we had planned to use was not permitted with our 3D-printed motor tube. This restriction had not been communicated to us at any point prior, forcing the team to make a last-minute decision. Given the time constraints, a complete redesign of the motor case was unfeasible.

As a result, we had to switch to a disposable motor system, the I-205, for the launch. This motor offered significantly reduced performance, reaching an estimated apogee of 1,500 feet compared to the I-366's projected 2,500 feet. Despite these limitations, the team adapted quickly and proceeded with the launch using the I-205 motor.

10.3.6. Delayed Testing and Limited Validation

Final integration of the rocket occurred within 48 hours of the scheduled launch due to cumulative delays across multiple subsystems. When the backup altimeter was installed after the primary altimeter was damaged, a ground test was conducted during the day leading up to launch. Both charges ignited, but neither drogue nor main deployed because the force was not strong enough. Due to the lack of time, the team decided not to test the system again and load the recovery and main ejection capsules with double the black powder to ensure ejection occurred. Despite this, the recovery systems still failed to deploy during flight, leading to loss of the vehicle. If the team had the option to delay the launch, further testing would have been conducted to ensure a successful flight.

10.3.7. Budget Constraints

Throughout the project, budget limitations significantly influenced design decisions and overall system performance. The most impactful constraint was the team's inability to afford a larger paraglider capable of safely supporting the rocket's weight. As a result, a less expensive and smaller paraglider was selected, which introduced safety and stability concerns during recovery.

Additionally, financial restrictions affected the design of the electronics system. Rather than purchasing more costly, off-the-shelf electronics, the team chose to source individual

electronic components to reduce expenses. While cost-effective, this approach required extensive programming and integration work. Much of this technical effort extended beyond the typical scope of coursework in the mechanical and aerospace engineering curriculum, necessitating a substantial amount of self-teaching. This learning curve created a significant bottleneck in the project timeline.

With additional funding, more advanced and integrated electronic systems could have been acquired, reducing development time and allowing for a more sophisticated and reliable final design.

10.3.8. Software Issues

While the RTLS software system demonstrated foundational success in simulation and hardware-in-the-loop environments, several open issues remain that limit full mission reliability.

Foremost, the most recent Monte Carlo analysis revealed that only 37% of descent trajectories resulted in landings within the 800 ft scoring radius. With a mean landing distance **of** 824.85 ft, and distances ranging from 524.27 ft to 1110.85 ft, the system failed to consistently meet the competition performance requirement. This variability, highlighted by a standard deviation of 135.13 ft, indicates that the adaptive control logic, while directionally effective, is not sufficiently robust under randomized wind conditions and offset trajectories. Future work will require improved gain scheduling, wind compensation models, and possibly real-time correction feedback to enhance performance consistency.

Additionally, while the heading angle was successfully calculated based on simulated GPS coordinates, the current implementation lacks yaw rate damping. Without integration of live

gyroscopic data, the system cannot compensate for overcorrection or oscillatory behavior, potentially degrading trajectory convergence.

Telemetry performance also remains unresolved. Although GPS and altitude data were correctly sent to Serial2, no consistent data appeared on the HTerm ground station during testing. This could be due to radio baud mismatches, improper wiring, or weak antenna transmission, each of which requires isolated debugging.

Finally, the system currently relies on CSV-based altitude simulation, and inconsistencies in parsing altitude values have led to delayed triggering of key state transitions, particularly around the 750 ft deployment event. This affects both RTLS activation and simulated black powder charge logic. A future implementation using real-time barometer data would eliminate this dependency and improve temporal precision across the control sequence.

These open issues highlight the need for further refinement in both simulation fidelity and embedded robustness before the RTLS software system can be deployed in a real-world flight environment.

10.4. Failure Analysis (Five Whys)

This section will outline the root causes of vehicle failure during the flight test. During the flight test, the rocket's parachutes failed to deploy, causing it to fly far from the launch site and hit the ground at a speed high enough to render it unsalvageable.

Problem: The flight test failed, and the rocket was unsalvageable.

Why?

Both the drogue and main parachutes failed to deploy.

Why?

The altimeter did not fire the ejection charges, or the ejection charges forces were too weak to separate the vehicle.

Why?

If the altimeter failed, this was likely due to the igniter cables becoming disconnected. If the black powder charges were not strong enough, this was due to insufficient ground testing by the team.

Why?

The cables could have potentially disconnected due to the high g-forces or vibrations experienced by the vehicle.

Why?

The team did not do their due diligence to ensure that the connections were strong enough to withstand in-flight forces but loose enough to allow the rocket to separate when the main black powder charge deployed.

Overall, the failed flight test proved the importance of through system testing before conducting a flight test. The team strongly recommends that the force required to disconnect the igniter cables be thoroughly tested and that all electronic components be checked for functionality before flight. This would likely have prevented the loss of the vehicle.

10.5. Future Plans and Recommendations

The primary goal moving forward is the successful full-scale integration of the Return-to-Launch-Site (RTLS) system into a high-powered rocket. This includes validating the full functionality of the dual-deployment system, with streamer deployment at apogee followed by the autonomous paraglider guidance sequence during main deployment. Achieving this requires a paraglider robust enough to handle forces greater than the rocket's total weight, along with aerodynamic drag during descent. The control lines must be securely attached to the servos and designed to withstand friction along the rocket body tube without snapping or detaching.

To enhance system performance, future iterations of the project will include the integration of longer-range radio modules to support greater flight distances. Additional funding would enable the purchase of higher-performance GPS systems with faster update rates, supporting the implementation of a proportional-derivative (PD) controller for real-time correction during descent. These upgrades aim to improve responsiveness and accuracy of the RTLS system during flight. Moreover, replacing the current servo with a continuous rotation servo will improve the paraglider's steering smoothness and turning efficiency, contributing to more precise landing near the original launch location.

10.4.1. Recommendations

The challenges encountered throughout this project highlight the importance of early-stage verification, clear communication, and rigorous systems integration. One of the key takeaways was the critical need to validate all assumptions—especially those involving flight-critical components—before progressing past the design phase. The decision to rely on a kite in place of a proper load-bearing canopy, while financially motivated, illustrates how unverified

assumptions can have cascading effects on system architecture, test planning, and overall feasibility.

Also, the failed flight test proved the importance of through system testing before demonstration. The team strongly recommends that the force required to disconnect the igniter cables be thoroughly tested and that all electronic components be checked for functionality before flight. This would likely have prevented the loss of the vehicle.

Another major lesson involved procurement and component sourcing. Delays and mismatches in ordered materials created substantial setbacks and forced last-minute redesigns that undermined schedule stability and technical reliability. Future teams would benefit from ordering critical hardware and consumables as early in the semester as possible, with built-in contingency plans and reserve funds to accommodate unexpected failures or substitutions.

Communication with administrative staff and inventory oversight bodies also emerged as a limiting factor. Design decisions were repeatedly impacted by inaccurate inventory data or uncommunicated changes in material availability. Establishing formal protocols for inventory verification, as well as clear lines of communication for component requests and approvals, would help prevent misalignment between project planning and real-world constraints.

From an engineering and systems integration standpoint, late-stage failures in electronics and propulsion design underscore the importance of modular testing and timeline padding. Multiple redesigns within the final week left little time for ground validation, leading to a flight-ready system with only partial verification of charge deployment, telemetry, and sensor behavior. Future teams should schedule dedicated windows for end-to-end subsystem testing, with redundant instrumentation and interface checks conducted well in advance of launch readiness.

Lastly, despite these limitations, the team demonstrated strong adaptability, creative problem solving, and resilience under pressure. The final assembled vehicle represented the culmination of interdisciplinary coordination, iterative design, and hands-on troubleshooting. While the RTLS system's full functionality was constrained by component failures and time limits, the framework established through this project lays the groundwork for future development. Enhancements such as fully steerable parachute systems, robust fault-tolerant avionics, and modular testing harnesses can all evolve from the lessons documented here.

By maintaining meticulous records of both our successes and setbacks, this project contributes valuable insight to the continued development of autonomous, recoverable, and reusable model rocket systems.

11. Conclusions and Recommendations

The Return-To-Launch-Site (RTLS) rocket project set out to develop a Level 1 model rocket capable of autonomous navigation and guided recovery using GPS, servo-actuated control, and a dual-stage deployment system. While the final prototype did not achieve full functionality or complete a successful flight, the project served as a valuable demonstration of complex systems integration, multidisciplinary design, and the challenges of building a novel aerospace recovery system within tight time, budget, and resource constraints.

Despite the ultimate failure of the prototype, the team accomplished several important milestones. A full-scale 3D-printed airframe was designed and assembled; key subsystems including GPS telemetry, barometric sensing, and servo control were integrated and tested in isolation; and simulations were carried out in MATLAB, OpenRocket, CFD, and FEA to validate structural and aerodynamic performance. A complete RTLS control algorithm was developed and tested in Simulink, offering real-time sensor processing and servo actuation logic. A thorough failure modes and effects analysis (FMEA) and a robust set of ground tests were also conducted, helping to characterize the system's limitations and inform future improvements.

However, multiple critical issues prevented a successful launch and recovery. The use of a low-cost kite in place of a proper load-rated paraglider resulted in a recovery system that could not support the rocket's weight. Procurement errors and delays severely impacted integration timelines, with key items such as streamers and shock cords arriving late and incorrect quantities. Several IMUs were damaged due to soldering mishaps, and power routing issues nearly resulted in hardware failure. In addition, poor communication with administration and inconsistent inventory documentation led to missed opportunities for test flights and frequent last-minute

design changes. Ultimately, the RTLS system could not be fully validated, and the final vehicle was unable to demonstrate autonomous guided return as intended.

Nonetheless, the project offered critical real-world experience in systems engineering, design trade-offs, and interdisciplinary coordination. The lessons learned will benefit future teams pursuing similar goals. Moving forward, it is strongly recommended that future iterations secure a verified parafoil early in the project, increase redundancy in sensor design, transition to longer-range radio systems, and implement structured testing milestones well in advance of launch. Improved communication with lab advisors and more accurate inventory tracking would also reduce late-stage disruptions and rework.

In summary, although the RTLS rocket failed in demonstrating its intended functionality, the effort laid a solid foundation for future research in autonomous recovery systems. The technical work, documentation, and testing processes undertaken throughout the project provide a detailed roadmap for identifying and avoiding critical pitfalls. With better resources and longer development cycles, a future version of this system has strong potential to achieve a successful return-to-launch-site demonstration. Having said this, it is the strong recommendation from this team that the RTLS project is not continued during future Senior Design semesters without serious readjustments to the budget and scope of the project, as well as requiring teams to include students with expertise in electronics and software.

Appendix A: Customer Requirements

All customer requirements below were provided to us by the customer in the form of a detailed project description…

14. Rocket shall follow all NAR and SRA safety standards and guidelines.
15. Rocket shall reach an apogee of 2,200 feet AGL.
16. Rocket shall reach a target no further than 150m from the launch site.
17. Rocket shall land no further than 800 feet from the target.
18. Rocket shall not be actively guided during the ascent phase of flight.
19. Rocket shall use an electronic ignition system.
20. Rocket shall not use a combustion landing system.
21. Rocket shall not produce more thrust than its weight on descent.
22. At least one member shall be L1 certified.
23. Rocket shall not use a “Sparky” motor.
24. Motor impulse shall not exceed Level 1
25. Rocket shall record FPV during descent and should transmit data to a ground station.
26. Rocket should have a sound emitting device (screamer) on board.

Justification for Ranking:

1. **Safety Standards**: Adhering to all safety standards and guidelines is the most important aspect of this project, as any violation will not be tolerated and could potentially put others in danger.
2. **Minimum Altitude**: The rocket must reach a minimum altitude of 2,200 feet above ground level. Any maximum height below this threshold will result in disqualification.
3. **Target Landing**: The rocket must land within **800 feet (approximately 244m)** of the center of the target.

4. **Ignition System**: An electronic ignition system shall be used, as required by the SRA/Palm Bay launch site.
5. **Combustion Landing System**: The rocket cannot use a combustion landing system, as this would violate NAR regulations.
6. **Guidance System**: The rocket cannot be actively guided during flight, as this would violate UCF regulations.
7. **Certification Requirement**: One team member must be Level 1 certified, as required by both NAR and UCF, since the rocket is classified as an L1 (Level 1) rocket due to its motor impulse. **Note:** This is not a certification flight.
8. **Video Streaming/Recording**: The rocket must be capable of recording or transmitting video footage. While this requirement is important to UCF, it is not critical to the project's success, leading to a lower priority.
9. **Screamer**: The rocket should carry a screamer onboard. Like the video requirement, this is not a legal mandate nor critical to the project’s success, and thus is ranked lower in priority.

	Customer Needs	Importance Weight	Risk			
			Injury to persons or property	Project Disqualification	Laws and Regulations	Customer wants
			1	2	3	4
1	Rocket shall follow all NAR and SRA safety standards and guidelines	0.18	x	x	x	
2	Rocket shall reach an apogee of 2200 feet AGL	0.15		x		
3	Rocket shall land within 800ft of the center of the target	0.15		x		
4	Rocket shall use an electronic ignition system	0.10		x	x	
5	Rocket shall not use a combustion landing system	0.10		x	x	
6	Rocket shall not be actively guided during the ascent phase of flight	0.10		x		
7	At least one member shall be L1 certified	0.05		x	x	
8	Recovery system shall not produce total thrust less than the burnout mass of the rocket.	0.05		x	x	
9	Rocket shall not use a Sparkiy motor, Dark Matter, Skidmark, and Metalstorm	0.05		x		
10	Motor shall not exceed Level 1 or have an impuse above 640Ns	0.05		x	x	
11	Rocket shall record FPV during descent and/or transmit to ground station	0.01				x
12	Rocket should have a screamer on board	0.01				x

Figure 76: Customer Needs Importance Analysis

A.1. NAR Regulations

	NAR Regulations
1	Launch Site: Rockets shall always be launched in an open area away from buildings, trees, power lines, and other obstacles. The site shall be free of hazards and shall provide enough space for the rocket to ascend and descend safely.
2	Safety Distance: Spectators shall maintain a safe distance from the launch pad, with a minimum distance of 15 feet from the launch site. The launcher shall be positioned at least 100 feet away from spectators and bystanders.
3	Launch Pad and Ignition: The rocket shall be securely mounted on the launch pad and pointed at a safe trajectory. Only electrical launch controllers shall be used to ignite the rocket.
4	Weather Conditions: Rockets shall not be launched in windy conditions, during storms, or when there is lightning. Weather conditions shall be assessed before every launch.
5	No Dangerous Modifications: Dangerous or non-approved modifications shall not be added to the rocket. Flammable materials or explosive devices shall not be added to the rocket.
6	Altitude and Recovery: The rocket shall be properly equipped with a recovery system, such as a parachute or streamer, that will bring it safely back to the ground.
7	Handling Rocket Motors: Proper safety procedures shall be followed when handling rocket motors. Rocket motors shall be kept in their original packaging and shall not be exposed to heat or physical damage.
8	Firing Pin/Launch Button: Only the person launching the rocket shall control the launch button or ignition system, and there shall be no one near the rocket when it is ignited.
9	Post-Launch Safety: After a rocket has landed, it shall not be retrieved until it has fully cooled down, as some motors may still be hot or live for a short period after launch.
10	Launch Site Size: The SRA Rocket Ranch shall be a five mile by five mile cleared launch site, free of major obstructions such as trees and buildings, and shall allow up to "M" class motor use with prior notification and approval.
11	FAA Waiver: The launch site shall have an FAA waiver to 13,500 feet AGL.
12	Launch Controller: The SRA shall use a nine-pad "electric match safe" launch controller that allows up to "K" power class clustered or staged multiple motor configurations and "L" power class single motor configurations with prior approval.
13	Launch Pad Variety: The launch site shall have six launch pads with ¼" through ¾" launch rods and 1010/1515 rails available for use.
14	Hybrid Motor Launcher: The SRA shall have one Hypertek hybrid motor launcher/filler pad with support equipment available for use with prior coordination.
15	HPR Fliers Access: HPR Fliers shall have access to and may conduct flights from the High-Power Launch Area and/or Model Rocket Launch Area.

	NAR Regulations
16	Non-Tripoli Members Access: Non-Tripoli members age 18 and over who are students of an accredited educational institution shall be allowed to participate in joint projects with Tripoli members, with supervision by an HPR Flier or Adult Flier.

Table 11**:** NAR Regulations

Appendix B: System Evaluation Plan

B.1. Overview

The team took a requirements-based approach to system evaluation using the Key Requirements and Design Parameters in M3 Section 4.3 to determine when testing was necessary while individual testing plans are outlined in M5 Section 4. The updated testing plan and results are discussed below.

B.2. Functional Test

Before conducting the final customer demonstration, a physical flight test will be conducted to verify all functional requirements are meet and that the flight data is concurrent to the simulated flight mission. The headings below are the objectives for the physical flight test.

B.2.1. Rocket lands <=800 ft from target

Success Criteria: Rocket lands intact with all systems functioning <= 800ft from the target.

Procedure: After paraglider deployment, the electronics suite guides rocket to the target.

B.2.2. Rocket Reaches Apogee >= 2200 ft AGL

Success Criteria: Rocket reaches Apogee at 2200 ft AGL

Procedure: Rocket motor is ignited, and the powder burns fully, while the rocket remains is steady controlled flight.

B.2.3. Impact Velocity <= 25 ft/s

Success Criteria: The rocket sustains no critical damage upon landing and sustains an impact velocity less than 25 ft/s. This requirement is the same as the paraglider descent velocity

Procedure: All critical events occur on the rocket including streamer 1 deployment, streamer 2 deployment, and paraglider deployment. Velocity data is recorded to the SD card via the accelerometer.

B.2.4. Streamer 1 Deploys at Apogee

Success Criteria: The streamer deploys before vertical velocity is less than -65 ft/s

Procedure: If the barometer indicates a negative Z-axis velocity, or when the acceleration becomes 0 after launch, the black powder charge will be ignited. The nose cone then separates, and the streamer deploys.

B.2.5. Streamer 2 and Paraglider Deploys at 500-700 ft AGL

Success Criteria: The paraglider deploys at 500-700 ft AGL with no critical damage.

Procedure: When the GPS or barometer indicates the altitude is less than 700 ft AGL on decent, the second black powder charge expels the main recovery system.

B.3. Component Tests

B.3.1. Radio Range

Reliable radio communication is essential to ensure the rocket can be recovered during flight testing. A full-scale prototype of the radio system is required to evaluate its performance under real-world conditions. If the rocket's paraglider deploys at apogee, the radio system will provide tracking and recovery capabilities.

Objectives:

1) Verifying software compatibility with the radio system.
2) Measuring the effective communication range.
3) Evaluating power consumption at maximum transmission power.

Success Criteria:

1) Communication between the radio and ground system remains stable and reliable.
2) The communication range falls within the 1–6-mile requirement.
3) Power consumption does not exceed the manufacturer's specified wattage.

Procedure:

1) A simple transmission test will be performed by sending a "hello" message between the radio and receiver to confirm software and hardware compatibility.
2) The radio system will be positioned 36 ft above ground level (AGL) while the receiver is moved away in 500 ft increments. Signal strength, data rate, and power levels will be recorded at each interval.

3) The radio system will be set to maximum transmission power, and a DC power supply with an ammeter will be used to measure average power consumption in watts.

B.3.2. Servo Torque

Reliable servo operation is crucial to ensure proper actuation of control surfaces on the paragliding. A comprehensive test of servo torque is required to validate its performance under expected operational loads.

Objectives:

1) Verifying that the servo meets the specified torque rating.
2) Measuring the servo's ability to maintain torque under load.
3) Evaluating power consumption during peak torque conditions.

Success Criteria:

1) The servo successfully lifts and holds a load corresponding to its rated torque.
2) No excessive overheating or failure occurs under maximum rated load.
3) Power consumption remains within the manufacturer's specifications.

Procedure:

1) Torque Verification: Attach a lever arm of known length to the servo horn. Suspend a known weight at a measured distance from the center of rotation to create a torque equivalent to the servo's rating. Verify that the servo can hold and move the load without excessive deflection.

2) Load Endurance Test: Apply a cyclical load to the servo by oscillating it between two positions under maximum rated torque conditions. Record performance over an extended period to identify potential overheating or mechanical failure.
3) Power Measurement: Use an oscilloscope and power supply to measure voltage and current draw during peak torque conditions. Calculate average power consumption in Watts.

B.3.3. Battery Longevity

The battery must ensure sustained operation of onboard systems throughout the mission. A structured test of battery longevity is required to validate its endurance under expected load conditions.

Objectives:

1) Verifying that the battery meets the expected operational duration.
2) Measuring the battery's discharge characteristics under nominal and peak loads.

Success Criteria:

1) The battery provides sufficient power for the full mission duration of 2.5 hours without failure.
2) Voltage and capacity depletion rates remain within acceptable limits under load.

Procedure:

1) Load Simulation: Connect the battery to a simulated load representing typical mission power consumption. Monitor voltage, current, and temperature throughout discharge.

2) Endurance Test: Fully charge the battery, then subject it to continuous operation until depletion. Record time to failure and energy delivered.

B.3.4. Streamer Max load

Understanding the maximum load capacity of the streamers is critical, as their failure during deployment could result in excessive velocity, leading to catastrophic structural damage to the paraglider. While destructive testing would yield the most accurate data on maximum load capacity, budget constraints necessitate an alternative approach. A representative sample of the streamer material will undergo destructive testing, and the results will be extrapolated using calculated material properties to estimate the maximum load capacity.

Objectives:

1) Determine the maximum tensile load the streamer material can withstand before failure.
2) Validate the streamer's capability to maintain structural integrity under expected aerodynamic forces.
3) Establish a safety margin for operational loads based on experimental data.

Success Criteria:

1) The streamer material withstands forces exceeding the calculated operational load with an appropriate factor of safety.
2) The failure mode of the material is consistent with expected deformation or tearing characteristics, confirming reliability under deployment conditions.
3) Experimental results align with theoretical calculations within an acceptable margin of error.

Procedure:

1) Material Preparation: Cut multiple samples of the streamer material to standardized dimensions for tensile testing. Ensure consistent handling to avoid pre-test damage.
2) Destructive Load Testing: Secure each sample in a tensile test machine and apply increasing force until failure. Record maximum force, elongation, and failure mode.
3) Data Analysis: Compare experimental results with calculated material properties. Identify trends and establish a failure threshold.

B.3.5. Streamer Drag Force

As stated in the overview, simulation-based testing will be used to increase the accuracy of the control algorithm. For accurate simulations, the streamer's drag force needs to be calculated via testing. The test must ensure accurate measurement of the drag force acting on a streamer during free fall.

Objectives:

1) Verifying that the streamer reaches a terminal velocity of <=25 ft/s (F.3.1) under controlled conditions.
2) Measuring acceleration and velocity to determine drag force.

Success Criteria:

1) Terminal velocity is achieved and recorded consistently across multiple tests.
2) Calculated drag force aligns with theoretical expectations within an acceptable margin of error.

Procedure:

1) Drop Test: Securely attach an accelerometer to the streamer and release it from a predetermined height. Record acceleration and velocity throughout the descent.
2) Data Analysis: Process recorded data to identify terminal velocity and compute the drag force using Newton's laws.

B.3.6. Paraglider Max Swing Load

To ensure structural integrity and prevent failure during deployment, it is critical to determine both the maximum static and dynamic loads the paraglider can withstand. This test will evaluate the material strength and dynamic response under simulated deployment conditions.

Objectives:

1) Quantify the maximum static load (swing load) the paraglider can endure before failure.
2) Determine the maximum dynamic load experienced during deployment.

Success Criteria:

1) The maximum swing load is measured within an acceptable margin of error.
2) The maximum dynamic load is determined within an acceptable margin of error.

Procedure:

1) Material Testing: Obtain representative paraglider materials and fabricate a scaled test sample that accurately reflects the paraglider's structural design.

2) Tensile Testing: Conduct controlled tensile tests on the sample to determine the ultimate tensile strength and failure threshold under static loading conditions.

3) Dynamic Load Testing: Perform incremental weighted drop tests to simulate deployment forces and measure the maximum dynamic load experienced.

4) Data Analysis: Analyze collected data to establish load limits and compare with theoretical predictions.

B.3.7. Paraglider Drag Force / Glide Ratio

The need for physical testing of the paraglider is essential to ensure accurate aerodynamic modeling and validation of simulation-based testing. As the paraglider being purchased has no aerodynamic data available from the manufacturer, physical testing was determined to be the fastest method to accommodate the tight deadline.

Objectives:

1) Determine the glide ratio by measuring horizontal and vertical displacement during free fall.

2) Calculate drag force using acceleration and barometric pressure data.

Success Criteria:

1) Glide ratio is consistently measured across multiple test runs.

2) Drag force is consistently measured across multiple test runs.

Procedure:

1) Drop Test: Securely attach an accelerometer and barometer to the paraglider. Release the object from a known height. Record acceleration data throughout the descent. Use the barometer to determine altitude changes. Measure horizontal displacement from the drop point to determine glide ratio.

2) Data Analysis: Process recorded data to identify terminal velocity and compute the drag force using Newton's laws. Determine glide ratio by averaging the vertical distances from testing divided by the height of the drop.

B.4. Additional Tests

B.4.1. Airframe Stability

Testing the stability of a model rocket ensures it flies straight and predictably, preventing dangerous tumbling or veering off course. It also verifies that the center of gravity (CG) and center of pressure (CP) are correctly positioned, optimizing aerodynamic performance and recovery system deployment.

Objectives:

1) Ensure CG and Cp are correctly positioned
2) Rocket does not tumble during flight

Success Criteria:

1) Ensure the CG is around 1.98 calibers ahead of the CP as calculated in OpenRocket
2) Rocket does not tumble during testing

Procedure:

1) Static Balance Test: Find the CG by balancing the rocket on a pivot, then compare with the simulated CG in OpenRocket. To find the CP use a cardboard cutout of rocket's profile and find the balance point. Then ensure the CG is around 1.98 calibers ahead of CP.

B.4.2. Airframe Structural Strength

Testing the structural strength of the airframe is crucial to ensure it can withstand the forces of launch, flight, and recovery without failing. High thrust, aerodynamic pressure, and parachute deployment forces can cause bending, cracking, or complete structural failure if the

materials and joints are not strong enough. By testing for compression, torsion, vibration, and impact resistance, you can confirm the rocket's durability, prevent in-flight failures, and ensure a safe and successful launch and recovery.

Objectives:

1) Ensure the airframe can withstand launch forces, aerodynamic stress, and recovery deployment.
2) Verify that structural joints and materials hold up under mechanical loads.

Success Criteria:

1) The airframe does not crack, warp, or separate under expected launch, flight, and landing stresses.
2) Structural joints remain intact after tensile, vibration, and impact testing.

Procedure:

1) Impact Resistance Test: Drop the airframe from a height of 15ft onto a cushioned but firm surface to simulate landing at 30 ft/s (maximum impact speed with a safety factor of
    a) Then, repeat the test at different angles, and inspect for cracks, dents, or deformations in the structure
2) Tensile test: Secure one end of the rocket and pull on the other end firmly to make sure the body tube joints are secured properly. Do this to the upper and lower body tube with attaching them together.

B.4.3. Coupler Fit and Tightness Test

Testing the coupler fit is essential to ensure the rocket's airframe sections remain securely connected during flight while still allowing for smooth separation at deployment. A coupler that is too loose can cause airframe misalignment or unintended separation, while a coupler that is too tight may prevent proper ejection of the recovery system. This test ensures that the coupler provides a firm but functional fit for reliable performance.

Objectives:

1) Verify that the coupler provides a secure connection without excessive looseness or tightness.
2) Ensure smooth separation at ejection while maintaining structural integrity during flight

Success Criteria:

1) The coupler should require moderate force to insert and remove neither too loose nor excessively tight.
2) The airframe sections should remain connected under vibration and pull tests but separate cleanly when the ejection charge is fired.

Procedure:

1) Friction Fit Test: Insert the coupler fully into the body tube by hand, then attempt to pull it apart using steady force (~5lbs). The coupler should require a firm but manageable force to separate. Do this for both couplers in the rocket.
2) Ejection charge test: Load a small test ejection charge into the rocket. Then fire the charge in a safe outdoor environment with the coupler in place. Observe whether the

coupler separates cleanly without excessive force or jamming. Do this for both couplers in the rocket.

Appendix C: Rocket Assembly and Flight Data Manual

C.1. ASSEMBLY MANUAL

C.1.1. Required Components

External Components:

- Nose cone
- Upper body tube
- Coupler
- Lower body tube
- Electronics bay body tube
- Motor/fin assembly

Internal Components:

- 10 ft shock cord (drogue)
- 15 ft shock cord (main)
- Complete electronics bay (batteries, servo, wiring)
- 3D-printed motor retainer
- Disposable solid rocket motor
- Paraglider and two streamers
- Black powder charges (0.6 g for drogue, 1.75 g for main)
- Blast caps (for black powder)

- E-matches
- M3 screws and shear pins

C.1.2. Assembly

C.1.2.1. Airframe Assembly

C.1.2.1.1. Prepare Nose Cone Assembly

1. Attach one end of the 10 ft shock cord to the nose cone bulkhead.
2. Securely attach the 60 ft drogue streamer to the shock cord by feeding the shock cord through both reinforced holes in the streamer. Tie a secure overhand knot at each exit point to prevent the streamer from sliding along the cord during deployment.
3. Tie the other end of the shock cord to the top bulkhead of the coupler.

C.1.2.1.2. Assemble the Electronics bay to Coupler Shock Cord

1. Attach the 15 ft shock cord from the bottom side of the coupler (facing the main body section) and then to the bulkhead on the underside of the electronics bay.
2. Attach the 5 ft streamers to the paraglider and begin folding and prepping the paraglider.

C.1.2.2. Ignition Wires to Black Powder Wiring

1. Feed the two E-match wires with the ignitor end through the 1/4in hole made in the coupler.
2. Get one of the wires and feed it through the hole that leads to the blast cap that will face the main recovery. Then get tape and label that wire as “M” (for main)

3. Get the other wire and lead it through the blast cap hole in the bulkhead that will cover the other side of the coupler. Attach tape to the end of the wire to label that wire as “D” (for drogue).

C.1.2.3. Black Powder Setup

1. After wiring the ignitor wires (E-matches) into the blast caps, the black powder can be fed into the blast caps.
2. For the drogue blast cap, remove the blast cap gently and fill the cap with 0.6 grams of black powder. Then put the cap back on.
3. For the main blast cap the procedure will be the same but the blast cap will be filled with 1.75 grams of black powder.

C.1.3. Electronics Setup

C.1.3.1. Installing Electronics Bay Into Body Tube

1. Insert the electronics bay fully into the electronics bay tube.
2. Slide the motor and fin assembly into the bottom of the electronics bay tube and secure it with the M3 screws provided.

C.1.4. Paraglider Installment and Prepare Wiring

1. Carefully fold the paraglider into the lower body tube using a blanket fold method (fold in the sides, corners, and top).

2. Ensure the paraglider control wires are connected from the electronics bay to the servo inside the bay.
3. Slide the folded paraglider into the lower body tube (main section).
4. Connect the drogue and main e-match leads to the appropriate ignition channels from the electronics bay.

C.1.5. Stage and Secure the Shock cord

- Route the 15 ft shock cord taut along the outer wall of the lower body tube, from the bottom of the electronics bay bulkhead.
- With painters' tape, tape the taut shock cord to the side of the lower body tube and infinity loop the rest of the shock cord.
- Take the looped shock cord and tape it with painters' tape to the packed paraglider
    - *Purpose: This ensures the cord gently assists with deployment before releasing, preventing tangling.*

C.1.6. Final Assembly

1. Fasten the lower body tube and the electronics bay tube together by lining the holes up with the threads on the top bulkhead of the electronics bay. Then use the M3 screws to screw them together.

2. Line up the 2 holes on the lower body tube and the couplerm then insert the nylon shear pin screws into the aligned holes.

3. Insert and secure the motor using the 3D-printed retainer.

C.1.7. Pre-Launch Activation

1. Power on the electronics bay through the activation pinhole, ensuring the system is armed and ready.

2. Perform any pre-launch checks, verify continuity, and confirm GPS lock and servo readiness.

C.1.8. Ready for Launch

Once all systems are secured, powered, and verified, the rocket is ready for launch.

C.2. ELECTRONICS FLIGHT DATA & IGNITION SYSTEM MANUAL

C.2.1. Introduction.

The electronics in the RTLS rocket work together as an integrated system that manages navigation, flight monitoring, data logging, and recovery deployment. At the heart of the system is the ESP32 microcontroller, which acts as the central processor, collecting data from all sensors and controlling key functions. A GPS module provides continuous updates on the rocket's position, altitude, and time, allowing the system to track its location throughout the flight. Meanwhile, a IMU sensor measures air pressure, temperature, compass heading, to calculate altitude and detect apogee. This data is used in real time to trigger the rocket's recovery events.

As the flight progresses, all information is recorded onto an onboard SD card, functioning as a flight data recorder for post-mission analysis. Simultaneously, a radio module transmits critical flight data to a ground station, enabling real-time monitoring from the ground. The system is also equipped with a buzzer that provides audible feedback, signaling successful startup, system readiness, normal operation, or failures through distinct beep patterns. Two ignition circuits, controlled by the ESP32, deploy black powder charges at apogee and at approximately 750 feet during descent to release the recovery systems—a streamer and a paraglider. The entire system is powered by a dedicated battery and carefully designed to operate autonomously, ensuring reliable recovery and mission tracking with minimal human intervention.

C.2.2. System Process

Power-Up & Startup:

- On powering up, the buzzer immediately sounds a 2-second full-power beep.
- The system then initializes the SD card, IMU, GPS module, and radio module.

Ground Initialization:

- The system waits for a valid GPS fix and records ground conditions (position, time, and altitude).
- A log file is created on the SD card, using the EST date/time.
- A second 2-second beep indicates that the system is fully operational.

Normal Operation:

- The system continuously reads sensor data.
- Flight data are logged and transmitted via the radio module and SD card.
- A periodic 1-second beep sounds every minute before ignition events signal a working system.

Ignition Events:

- PB_1 fires at apogee when a drop of more than 5 m from the maximum altitude is detected along with negative vertical velocity.
- PB_2 fires when the rocket descends to approximately 750 ft AGL (~229 m above launch altitude) during descent.

Post-Ignition Operation:

- After both ignitions have occurred, the buzzer switches its pattern to a 2-second beep every 10 seconds.

Failure Mode:

- If a critical system error occurs (e.g., BMP180 sensor failure), the system enters failure mode.
- In failure mode, the buzzer sounds continuously at half power to alert the operator without excessive noise.

C.2.3. Hardware Installation

C.2.3.1. Special Connection Warning

To safely plug in your laptop, you must disconnect the Vin pin from the ESP32. Leaving the Vin pin connected when your laptop is plugged in can send high voltage back into your computer, potentially destroying it. Ensure that the Vin connection is properly isolated before connecting your laptop for programming or debugging.

C.2.3.2. Required Software Libraries

Install the following libraries via the Arduino IDE Library Manager:

1. Adafruit BMP085 Library
2. TinyGPS++ Library
    - SPI, and FS libraries (typically pre-installed)

3. Install the required libraries if they are not already installed.
4. Open and verify the provided code in the Arduino IDE.
5. Within the file “Core1.cpp”, find the variable declarations of “targetlat” and “targetlon”. These values will be changed by the user to their desired coordinates to return the rocket to.
6. Check that the pin assignments match your hardware configuration.
7. Upload the sketch to your ESP32 board.
    - IMPORTANT: Ensure the Vin pin is disconnected from the ESP32 before connecting your laptop.

C.2.4. Software Installation & Programming

C.2.4.1. Pre-Launch Checks

Before powering on the system, make sure your power supply is stable and reliable. Always disconnect the Vin pin from the ESP32 before plugging in your laptop to avoid damaging it. Then, insert a formatted SD card (FAT16 or FAT32) into the slot so the system can log flight data.

To set up the ground station, use a program called Hterm, which lets you talk to the rocket’s radio. You’ll need to send a few setup commands to make sure the rocket and ground radio can communicate. These commands are:

```
sendCommand(radioSerial, "AT\r\n");
sendCommand(radioSerial, "AT+ADDRESS=2\r\n");
endCommand(radioSerial, "AT+NETWORKID=5\r\n");
sendCommand(radioSerial, "AT+BAND=915000000\r\n");
sendCommand(radioSerial, "AT+PARAMETER=10,7,1,7\r\n");
```

Figure 77: Hterm Commands

Open Hterm, send these commands, and check that the settings match your ground radio's requirements. Set address = 2.

C.2.4.3. Launch Sequence

Starting operation:

- Power up the system via the power button.
- A 2-second full-power beep is emitted at startup.
- Wait for system initialization.
- The system initializes all electrical components.
- Ground conditions (position, time, altitude) are recorded.
- A second 2-second full-power beep signals that the system is fully operational.
- A log file is created on the SD card.

Normal operation:

- Flight data is transmitted via the radio module and logged to the SD card.
- A 1-second beep sounds every minute during normal pre-ignition operation.
- Ignition events:

- PB_1 fires automatically at apogee.
- PB_2 fires automatically when the rocket descends to approximately 750 ft AGL.

Post Ignition:

- The buzzer emits a 2-second beep every 10 seconds to indicate that both ignitions have fired.

C.2.4.4. Failure Mode

If any critical system fail the system will stop normal operation and the buzzer will sound continuously at half power as an alert. Consult the troubleshooting section for further guidance.

C.2.5. Troubleshooting & Maintenance

C.2.5.1. Troubleshooting

SD Card Issues:

- Verify that the SD card is inserted correctly and formatted as FAT16 or FAT32.
- Check the SPI connections.

Sensor Errors:

- Ensure proper I^2C wiring and power.
- Confirm that the sensors are securely connected.

GPS Initialization Delays:

- Allow the GPS module sufficient time for an outdoor fix.
- Ensure the antenna is properly oriented.

Radio Transmission Problems:

- Check the AT command settings using Hterm.
- Verify that radio connections match the pin assignments.

Laptop Safety:

- Always disconnect the Vin pin before connecting your laptop.

C.2.5.2. Maintenance

Regularly inspect sensor connections and calibrate if necessary. Check SD card integrity and log files periodically. Update firmware and software libraries as needed.

C.2.6. Safety Considerations

Electrical Safety:

- Always double-check wiring before powering up.
- Disconnect the Vin pin to protect your laptop from high voltage.

Ignition Safety:

- The system controls high-energy ignition events.
- Test the system in a controlled environment and ensure ignition wiring is secure and insulated.

Buzzer Noise:

- Continuous buzzer operation can be disruptive.
- Adjust mounting or volume as needed based on operational environments.

General Operation:

- Always adhere to local safety regulations when handling pyrotechnic devices and during rocket launches.

C.2.7. Conclusion

This manual provides detailed instructions for the setup, operation, and troubleshooting of the Rocket Flight Data & Ignition System. By following these guidelines, operators can deploy the system to accurately collect flight data, precisely control ignition events, log data for post-flight analysis, and receive real-time radio updates with integrated audible alerts.

C.3. Guidance System Manual

C.3.1. System Overview

The RTLS HIL Test System simulates the descent and guidance phase of a model rocket using a dual-core ESP32 microcontroller. It is designed to validate the performance of the RTLS (Return-to-Launch-Site) control algorithm using a simulated flight profile while operating real hardware components such as servos, an SD card, and telemetry output.

This system separates tasks across the ESP32's two CPU cores using FreeRTOS:

- **Core 0** handles live sensor readings (GPS, IMU, barometer)
- **Core 1** runs the RTLS simulation algorithm based on preloaded altitude data

C.3.2. Hardware Requirements

To run this system, you will need:

- ESP32 (dual-core capable)
- SD card + microSD card reader
- Servo motor (e.g., Injora INJ5035-360)
- GPS module (NEO-6M)
- Barometer (BMP388 or BMP180)
- IMU (BerryIMUv3)
- Radio module (optional for telemetry)

- Power supply (battery or USB)
- Serial monitor or ground station for telemetry

C.3.3. File Structure

Place the following files in your Arduino project folder:

```
RTLS_HIL_Final/
├── RTLS_HIL.ino              → Main file (sets up both cores)
├── Core0.ino                 → Sensor task for Core 0
├── Core1.ino                 → Simulation task for Core 1
├── rtls_controller.cpp       → RTLS algorithm logic
├── rtls_controller.h         → RTLS algorithm header
└── Time vs Altitude.csv      → Place on SD card only (not in the project folder)
```

Figure 78: HTL File set up

C.3.4. File Descriptions

- **RTLS_HIL.ino**

  Launches two FreeRTOS tasks using xTaskCreatePinnedToCore() for Core 0 and Core 1.

- **Core0.ino (Sensor Task)**

  Reads real-time data from GPS, barometer, and IMU to ensure sensor stack functionality.

  This core runs independently and does not feed data into the RTLS algorithm during simulation.

- **Core1.ino (Simulation Task)**

  Loads altitude profile from Time vs Altitude.csv, simulates flight events (apogee and

descent to 750 ft), runs the RTLS controller logic, moves the servo, logs data to FlightLog.csv, and outputs telemetry via Serial and radio.

- **rtls_controller.cpp / .h**

  Contains updateRTLS() and getServoCommand() functions. Uses a PD controller to calculate and output servo angle corrections (±15°) based on target heading.

- **Time vs Altitude.csv**

  Must be placed on the SD card. Provides a simulated flight profile in the format time,altitude.

C.3.5. How to Run the System

1. Flash the entire project to the ESP32 using the Arduino IDE.
2. Insert a **formatted SD card** with Time vs Altitude.csv saved on the root directory.
3. Power on the ESP32 — both cores will begin running.
4. Monitor telemetry through the Serial Monitor and optional ground station radio.
5. After the test completes, retrieve FlightLog.csv from the SD card for review.

Appendix D: Cost Analysis and Manufacturability

D.1. Product Materials Cost

Item	Make	Quantity	Unit Price	Subtotal
Motor Propellant Kit	Aerotech 38mm Propellant Kit - I285R-14A	1	$78.10	78.10
Motor Casing	RMS-38/480 Casing w/Forward Seal Disk	1	$103.36	103.36
3-D Filament	Polymaker 3kg PETG Filament 1.75mm Black (1880g)	1	$38.85	38.85
Paraglider	Synapse 170	1	$95.00	95.00
Screamer	Birdie	1	$29.95	29.95
Epoxy	JB Clearweld Epoxy 8oz	1	$9.49	9.49
M3 x.5 x 12mm Screws	Phillips Pan Head Machine Screw A2 SS	1	$1.08	1.08
Nomex Blanket	LOC/PRECISION Flame Resistant Blanket 36" x 36"	1	$27.08	27.08

Shock Cord	1/2" Nylon Tubular Webbing 25ft	1	$17.00	17.00
Streamer 1	Mylar Streamer 4in X 56in	1	$7.35	7.35
Streamer 2	6" x 60' Streamer	1	$70.50	70.50
Threaded Inserts	M3 Standard	1	$1.54	1.54
Camera	Universal Astrocam	1	$53.49	53.49
Computer	ESP-32	1	$14.00	14.00
GPS	NEO-6 GPS	1	$10.00	10.00
Magnetometer/Barometer	BerryIMUv3	1	$38.00	38.00
Radio	RYLR896	2	$15.90	31.80
Servo	Digital RC Servo	1	$0.00	0.00
SD Card Reader	TF Card Reader	1	$8.00	8.00
Micro SD Card	PNY 32GB	1	$9.00	9.00
Battery	URGENEX 3000mAH Lithium Ion	1	$25.00	25.00
Black Powder Switch	DC 3.3V Relay Module	2	$1.58	3.16
Antenna	Super 8 Antenna (915MHz)	1	$13.48	13.48

Black Powder Ignition	32Ga Nichrome 6C	1	$8.19	8.19
Compass	Eyeskey Military Sighting Navigation Compass	2	$27.99	55.98
Battery Charger	URGENEX	1	$8.99	8.99
USB to UART Dongle	CP2102	1	$4.82	4.82
Ground Station Antenna	7dBi Antenna (915MHz)	1	$8.99	8.99

D.2. Manufacturing Equipment

Item	Make	Quantity	Unit Price	Subtotal
3-D Print Machine	Prusa XL Single Nozzle	1	$1999.00	1999.00
Soldering Iron	Soldering Iron Kit, 80W 110V	1	$8.99	8.99
Philips Screwdriver	Pittsburgh #1 x 3in Philips Screwdriver	1	$1.70	1.70
Level	PITTSBURGH 48 in. I-Beam Level	1	$9.99	9.99
Marker	Sharpie Marker 2-pack	1	$1.99	1.99

D.3. Testing Equipment

Item	Make	Quantity	Unit Price	Subtotal
Force Gauge 500N	Deslands Digital Force Gauge	1	$32.99	32.99
Multimeter	Digital Multimeter, DC AC Voltmeter	1	$9.99	9.99

D.4. Overall Cost Summary

Aspect	**Cost**
Material Cost per Rocket	$772.20
Manufacturing Equipment (One time cost)	$2021.67
Testing Equipment (One time cost)	$42.98
Total Cost:	$2064.65 + $772.20*
	*per rocket

Appendix E: Expense Report

E.1. Electronics Bay Components

Component	Quantity	Cost per Unit (USD)	Shipping (USD)	Total (USD)
GPS Module NEO-6	1	$10.00	$0	$20.00
Magnetometer/-Barometer BerryIMUv3	1	$38.00	$0	$38.00
Flight Controller (ESP 32)	1	$14.00	$0	$14.00
Radio Transmitter (RYLR896)	2	$0.00	N/A	$0
Servo (Spektrum A6390)	1	$0.00	N/A	$0
Battery (URGENEX 3000mAH Lithium Ion)	1	$25.00	$0	$25.00
Personal Safety Alarm (She's Birdie)	1	$0	N/A	$0
Camera (Universal Astrocam)	1	$53.49	$11.29	$64.78
Black Powder Switch (DC 3.3V Relay Module)	1	$0	N/A	$0
Antenna (Super 8 Antenna)	1	$0	N/A	$0
Subtotal				$162.73

E.2. Ground Station Equipment

Component	Quantity	Cost per Unit (USD)	Shipping (USD)	Total (USD)
Battery Charger (URGENEX)	1	$0.00	N/A	$0.00
Antenna (7dBi Antenna)	1	$0.00	N/A	$0.00
USB to UART Dongle	1	$0.00	N/A	$0.00
Subtotal				$0.00

E.3. **Rocket Body and Structural** Components

Component	Quantity	Cost per Unit (USD)	Shipping (USD)	Total (USD)
3-D Filament (Polymaker 3kg PETG)	1	$61.99	$0.00	$61.99
Epoxy (JB Clearweld Epoxy 8oz)	1	$18.98	$0.00	$18.98
U–Bolts (6 pack Stainles Steel)	1	$11.99	$0.00	$11.99
M3 x 12mm Screws	1	$0.00	N/A	$0.00
Threaded Inserts	1	$0.00	N/A	$0.00
Subtotal				$92.99

E.4. Recovery System

Component	Quantity	Cost per Unit (USD)	Shipping (USD)	Total (USD)
Streamer (4in x 56in Mylar)	1	$7.35	$8.19	$32.54
Shock Cord (1/2” Nylon Tubular Webbing 25ft)	1	$17.00		
Paraglider (Synapse 170)	1	$95.00	$0	$95.00
Ripstop Nylon PU coated 1.9oz	5	$9.00	$0	$45.00
Nomex Blanket (36” x 36” Flame Resistant Blanket)	1	$27.08	$8.19	$3
Streamer (6” x 60’ Streamer)	1	$70.50	$0	$70.50
Black Powder Charge	1	$0	N/A	$0
Black Powder Ignition (32Ga Nichrome 6C)	1	$0	N/A	$0
Subtotal				$278.31

E.5. Propulsion

Component	Quantity	Cost per Unit (USD)	Shipping (USD)	Total (USD)
I-285 Motor	1	$0	N/A	$0
Aerotech 38/720mm Motor Case	1	$0	N/A	$0
Subtotal				$0

E.6. Miscellaneous

Component	Quantity	Cost per Unit (USD)	Shipping (USD)	Total (USD)
Force Gauge	1	$0	N/A	$0
Spray Paint	5	$5.98	N/A	$29.90
Subtotal				$29.90

E.7. Total Expenses

Category	Amount (USD)
Electronics Bay	$162.73
Rocket Body	$92.99
Recovery System	$278.31
Propulsion	$0.00
Ground Station	$0.00
Miscellaneous	$29.90
Grand Total	$563.93

Appendix F: List of Manuals and Other Documents

F.1. Software

Kalman's Original Paper on Linear Filtering:

- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering, 82(Series D), 35–45.

State-Space Control Design Textbook:

- Friedland, B. (2005). Control system design: An introduction to state-space methods (Dover ed.). Dover Publications. (Original work published 1986)

Comprehensive Overview of GPS and WAAS Architecture:

- Enge, P., & Misra, P. (1999). Special issue on global positioning system: Signals, measurements, and performance. Proceedings of the IEEE, 87(1), 3–15. https://doi.org/10.1109/5.736345

F.2. Hardware

Electronics Bay:

- Flight Computer (ESP-32): https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
- GPS (NEO-6VGPS): https://content.u-blox.com/sites/default/files/products/documents/LEA-NEO-MAX-6_HIM_%28UBX-14054794%29_1.pdf
- Manometer/Barometer/Accelerometer (Berry IMUv3): https://ozzmaker.com/berryimu-quick-start-guide/
- Radio (RYLR896): https://reyax.com//upload/products_download/download_file/LoRa%20AT%20Command%20RYLR40x_RYLR89x_EN.pdf
- Servo (Spektrum A6390 Mid-Torque, Mid-Speed): https://www.spektrumrc.com/on/demandware.static/-/Sites-horizon-master/default/dwd1230205/Manuals/SPMSA6390-Manual-MULTI.pdf
- 915MHz Antenna (Super 8 Antenna): https://medium.com/home-wireless/testing-and-reviewing-lora-antennas-5b37dfa594a3

Ground Station:

- 915MHz Antenna (7dBi Antenna): Link provides datasheet and installation guide https://www.trendnet.com/support/support-detail.asp?prod=140_TEW-AI77OB

External Camera:

- Camera Unit (Universal Astro Cam):

  https://modelrockets.co.uk/index.php?dispatch=attachments.getfile&attachment_id=120

F.3. Propulsion

- Aerotech Motor Tube Assembly Instructions

  https://www.yumpu.com/en/document/read/30843053/38-480-720w-instructions-aerotech

F.4. Recovery

- Steamer

  https://the-rocketman.com/streamers/

- Synapse 170 Kite

  https://prismkites.com/products/synapse-170?srsltid=AfmBOoqyNKlcTQxQe77rMXtWiN7LBTdw1MOLCsbbdRDam35UKIIxLd43

(There are no manuals for the airframe)

Appendix G: ABET Design Competence Matrices and Topic Competence Criticality Matrix

AEROSPACE ENGINEERING DESIGN COMPETENCE EVALUATION

Project Title: Return to Launch Site – Team Purple

AERONAUTICAL	Critical/Main contributor	Strong contributor	Necessary but not a primary contributor	Necessary but only a minor contributor	Only a passing reference	Not Included in this Design Project
Aerodynamics		X				
Aerospace Materials	X					
Flight Mechanics		X				
Propulsion		X				
Stability & Control	X					
Structures		X				

ASTRONAUTICAL	Critical/Main contributor	Strong contributor	Necessary but not a primary contributor	Necessary but only a minor contributor	Only a passing reference	Not Included in this Design Project
Aerospace Materials						X
Attitude Determination & Control		X				
Orbital Mechanics						X
Rocket Propulsion						X
Space Environment						X
Space Structures						X
Telecommunications				X		

MECHANICAL ENGINEERING DESIGN COMPETENCE EVALUATION

ME Design Areas	Critical/Main contributor	Strong contributor	Necessary but not a primary contributor	Necessary but only a minor contributor	Only a passing reference	Not Included in this Design Project
Thermal-Fluid Energy Systems						X
Machines & Mechanical Systems						X
Controls & Mechatronics			X			
Materials Selection	X					
Modeling & Measurement Systems			X			
Manufacturing			X			

ABET Summary

Topic Competence Criticality Matrix:

Aeronautical and/or Astronautical Topics Utilized in this Senior Design Project:

Topic	Criticality to Project	Section and Page(s)	Comments
Aerodynamics	Strong	Section 6, 7.4	
Aerospace Materials	Critical	Section 6, 7.3	
Flight Mechanics	Strong	Section 7.5	
Propulsion	Strong	Section 6, 7.2, 8.3	
Stability & Control	Critical	Section 7.5, 8.4, 8.5	
Structures	Strong	Section 6, 7.3, 8.3	
Attitude Determination & Control	Strong	Section 7.5, 8.5	
Telecommunications	Not Necessary	Section 7.5, 8.5	*Radio

Mechanical Topics Utilized in this Senior Design Project:

Topic	Criticality to Project	Section and Page(s)	Comments
Thermal-Fluid Energy Systems	Not Included	N/A	
Machines & Mechanical Systems	Not Included	N/A	
Controls & Mechatronics	Necessary	Section 7.5. 8.5	
Materials Selection	Critical	Section 6	
Modeling & Measurement Systems	Necessary	Section 7.2, 7.3, 7.4	
Manufacturing	Necessary	Section 6, 7.2, 8.3	

Appendix H: Code

H.1. Flight Algorithm

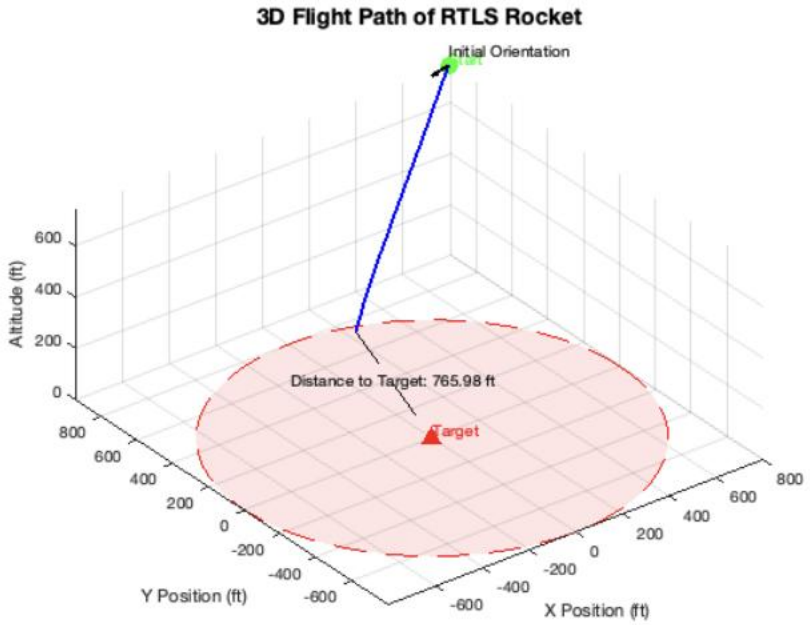
H.1.1. Main.m

```
function main(disable_plots_flag)
%clearvars -except FINAL_DISTANCE results i;

if nargin < 1
    disable_plots_flag = false;
end


global FINAL_DISTANCE

FINAL_DISTANCE = NaN;  % <-- initialize early!


% Initialize data logging for 3D trajectory
x_log = [];
y_log = [];
z_log = [];

% RTLS Flight Algorithm - Heading Based PD Control Only
if exist('telemetry_log.csv', 'file')
    delete('telemetry_log.csv');
end

% Initialize
current_x = randi([700, 1100]);
current_y = randi([700, 1100]);
target_x = 0;
target_y = 0;
dt = 0.1;
t = 0;
max_time = 80;
yaw = randi([0,359]);
initial_yaw = yaw;
yaw_error_prev = 0;
altitude = 750;
velocity_z = -10;

% Wind configuration
wind_speed_mean = 0;        % Mean wind speed in ft/s
wind_speed_std = 1;         % Std deviation of wind gusts
wind_direction_mean = 0;    % Mean wind direction in radians (0 = +X direction)
wind_direction_std = pi/8;  % Directional variability

max_angle = 15;

clear log_telemetry

while t < max_time && altitude > 0
    % Compute bearing and distance to target
    [distance_to_target, target_bearing] = Euclidean_distance(current_x, current_y, target_x, target_y);
    target_bearing = mod(target_bearing, 360);

    % Compute yaw error [-180, 180]
    yaw_error = mod(target_bearing - yaw + 180, 360) - 180;

    % PD Controller
    % Adaptive PD Controller
    [Kp, Kd] = get_adaptive_gains(distance_to_target);
    servo_cmd = pd_controller(yaw_error, yaw_error_prev, dt, Kp, Kd, max_angle);

    yaw_error_prev = yaw_error;

    % Simulate sensor & motion update
    [altitude, velocity_z, current_x, current_y, yaw] = get_sensor_data(servo_cmd, current_x, current_y, altitude, velocity_z, yaw, dt);

    % Apply wind *after* movement is updated
    wind_speed = wind_speed_mean + randn * wind_speed_std;
    wind_direction = wind_direction_mean + randn * wind_direction_std;
    wind_dx = wind_speed * cos(wind_direction);
    wind_dy = wind_speed * sin(wind_direction);
    current_x = current_x + wind_dx * dt;
    current_y = current_y + wind_dy * dt;

    % Log trajectory data
    x_log(end+1) = current_x;
    y_log(end+1) = current_y;
```

```
    z_log(end+1) = altitude;

    % Log data
    log_telemetry(t, yaw, target_bearing, yaw_error, servo_cmd, current_x, current_y, altitude, velocity_z, distance_to_target);

    t = t + dt;

    %display data to command window
    if mod(t, 1) < dt
    fprintf('t=%.1fs | X=%.1f | Y=%.1f | Alt=%.1f | Yaw=%.1f | YawTarget=%.1f | Err=%.1f | Dist=%.1f\n', ...
        t, current_x, current_y, altitude, yaw, target_bearing, yaw_error, distance_to_target);
    end

end

% Store final distance globally (for Monte Carlo runner)
if exist('distance_to_target', 'var') && ~isempty(distance_to_target)
    FINAL_DISTANCE = distance_to_target;
else
    FINAL_DISTANCE = NaN; % simulation exited early
end

if ~disable_plots_flag
    figure;


% 3D Trajectory Plot
figure;

% Plot trajectory
plot3(x_log, y_log, z_log, 'b-', 'LineWidth', 2); hold on;

% Start position
plot3(x_log(1), y_log(1), z_log(1), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
text(x_log(1), y_log(1), z_log(1)+30, 'Start', 'Color', 'g', 'FontSize', 10);

% Target position
plot3(target_x, target_y, 0, 'r^', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
text(target_x, target_y, 30, 'Target', 'Color', 'r', 'FontSize', 10);

% Orientation arrow
arrow_length = 100;
quiver3(x_log(1), y_log(1), z_log(1), ...
        cosd(initial_yaw)*arrow_length, sind(initial_yaw)*arrow_length, 0, ...
        'k', 'LineWidth', 2, 'MaxHeadSize', 2);
text(x_log(1), y_log(1), z_log(1)+60, 'Initial Orientation', 'FontSize', 10);

% Draw 800 ft radius circle around target
r = 800;
theta = linspace(0, 2*pi, 100);
circle_x = target_x + r * cos(theta);
circle_y = target_y + r * sin(theta);
circle_z = zeros(size(circle_x));
fill3(circle_x, circle_y, circle_z, 'r', ...
      'FaceAlpha', 0.1, 'EdgeColor', 'r', 'LineStyle', '--');

% Annotate distance from final position
final_x = x_log(end);
final_y = y_log(end);
final_dist = sqrt((final_x - target_x)^2 + (final_y - target_y)^2);
plot3([final_x, target_x], [final_y, target_y], [0, 0], 'k--');
text((final_x + target_x)/2, (final_y + target_y)/2, 20, ...
     sprintf('Distance to Target: %.2f ft', final_dist), ...
     'HorizontalAlignment', 'center', 'Color', 'k', 'FontSize', 10);

% Formatting
xlabel('X Position (ft)');
ylabel('Y Position (ft)');
zlabel('Altitude (ft)');
title({'3D Flight Path of RTLS Rocket'; ''}, 'FontWeight', 'bold', 'FontSize', 14);
grid on;
axis equal;
end

global FINAL_DISTANCE;


% Create traj_data matrix from simulation logs
```

```
t=1.1s | X=785.1 | Y=940.1 | Alt=739.0 | Yaw=206.0 | YawTarget=230.1 | Err=36.1 | Dist=1225.5
t=2.0s | X=778.5 | Y=937.7 | Alt=730.0 | Yaw=194.0 | YawTarget=230.3 | Err=24.3 | Dist=1219.4
t=3.0s | X=771.3 | Y=935.4 | Alt=720.0 | Yaw=194.0 | YawTarget=230.5 | Err=24.5 | Dist=1213.0
t=4.0s | X=764.2 | Y=932.7 | Alt=710.0 | Yaw=193.9 | YawTarget=230.6 | Err=24.7 | Dist=1206.5
t=5.1s | X=756.2 | Y=929.7 | Alt=699.0 | Yaw=205.8 | YawTarget=230.9 | Err=37.0 | Dist=1199.1
t=6.1s | X=748.8 | Y=926.9 | Alt=689.0 | Yaw=205.8 | YawTarget=231.0 | Err=37.3 | Dist=1192.3
t=7.1s | X=740.8 | Y=924.0 | Alt=679.0 | Yaw=205.7 | YawTarget=231.3 | Err=37.5 | Dist=1185.1
t=8.1s | X=733.5 | Y=921.3 | Alt=669.0 | Yaw=205.7 | YawTarget=231.5 | Err=37.7 | Dist=1178.4
t=9.1s | X=726.0 | Y=918.7 | Alt=659.0 | Yaw=205.8 | YawTarget=231.7 | Err=37.9 | Dist=1171.6
t=10.1s | X=718.3 | Y=916.1 | Alt=649.0 | Yaw=205.8 | YawTarget=231.9 | Err=38.1 | Dist=1164.9
t=11.1s | X=710.7 | Y=913.2 | Alt=639.0 | Yaw=205.8 | YawTarget=232.1 | Err=38.3 | Dist=1158.0
t=12.1s | X=703.1 | Y=910.6 | Alt=629.0 | Yaw=205.8 | YawTarget=232.3 | Err=38.5 | Dist=1151.2
t=13.1s | X=695.9 | Y=907.9 | Alt=619.0 | Yaw=205.8 | YawTarget=232.5 | Err=38.7 | Dist=1144.6
t=14.1s | X=688.1 | Y=905.0 | Alt=609.0 | Yaw=205.8 | YawTarget=232.7 | Err=38.9 | Dist=1137.6
t=15.1s | X=680.9 | Y=902.4 | Alt=599.0 | Yaw=205.8 | YawTarget=232.9 | Err=39.2 | Dist=1131.2
t=16.1s | X=673.5 | Y=899.5 | Alt=589.0 | Yaw=205.7 | YawTarget=233.2 | Err=39.4 | Dist=1124.5
t=17.1s | X=666.3 | Y=896.8 | Alt=579.0 | Yaw=205.7 | YawTarget=233.4 | Err=39.6 | Dist=1117.9
t=18.1s | X=658.8 | Y=894.0 | Alt=569.0 | Yaw=205.7 | YawTarget=233.6 | Err=39.9 | Dist=1111.4
t=19.0s | X=652.0 | Y=891.6 | Alt=560.0 | Yaw=193.7 | YawTarget=233.8 | Err=28.1 | Dist=1105.2
t=20.0s | X=644.3 | Y=888.9 | Alt=550.0 | Yaw=193.7 | YawTarget=234.0 | Err=28.3 | Dist=1098.4
t=21.0s | X=636.5 | Y=886.2 | Alt=540.0 | Yaw=193.8 | YawTarget=234.3 | Err=28.5 | Dist=1091.7
t=22.0s | X=629.4 | Y=883.8 | Alt=530.0 | Yaw=193.8 | YawTarget=234.5 | Err=28.7 | Dist=1085.6
t=23.0s | X=622.2 | Y=881.1 | Alt=520.0 | Yaw=193.7 | YawTarget=234.7 | Err=29.0 | Dist=1079.3
t=24.0s | X=615.0 | Y=878.5 | Alt=510.0 | Yaw=193.7 | YawTarget=235.0 | Err=29.3 | Dist=1073.0
t=25.0s | X=607.5 | Y=875.8 | Alt=500.0 | Yaw=193.7 | YawTarget=235.2 | Err=29.6 | Dist=1066.4
t=26.0s | X=600.4 | Y=873.2 | Alt=490.0 | Yaw=193.6 | YawTarget=235.5 | Err=29.8 | Dist=1060.3
t=27.0s | X=592.6 | Y=870.7 | Alt=480.0 | Yaw=193.7 | YawTarget=235.7 | Err=30.1 | Dist=1053.9
t=28.0s | X=584.8 | Y=868.1 | Alt=470.0 | Yaw=193.7 | YawTarget=236.0 | Err=30.3 | Dist=1047.3
t=29.0s | X=577.5 | Y=865.3 | Alt=460.0 | Yaw=193.7 | YawTarget=236.2 | Err=30.5 | Dist=1041.0
t=30.0s | X=570.1 | Y=862.9 | Alt=450.0 | Yaw=193.8 | YawTarget=236.5 | Err=30.8 | Dist=1034.6
t=31.0s | X=562.1 | Y=860.1 | Alt=440.0 | Yaw=193.7 | YawTarget=236.8 | Err=31.1 | Dist=1028.1
t=32.0s | X=553.9 | Y=857.3 | Alt=430.0 | Yaw=193.7 | YawTarget=237.1 | Err=31.4 | Dist=1021.5
t=33.0s | X=546.7 | Y=854.8 | Alt=420.0 | Yaw=193.7 | YawTarget=237.4 | Err=31.6 | Dist=1015.3
t=34.0s | X=539.2 | Y=852.1 | Alt=410.0 | Yaw=193.8 | YawTarget=237.6 | Err=31.9 | Dist=1009.1
t=35.0s | X=531.7 | Y=849.3 | Alt=400.0 | Yaw=193.8 | YawTarget=237.9 | Err=32.1 | Dist=1002.6
t=36.0s | X=524.3 | Y=846.7 | Alt=390.0 | Yaw=193.8 | YawTarget=238.2 | Err=32.4 | Dist=996.4
t=37.0s | X=516.7 | Y=844.1 | Alt=380.0 | Yaw=193.7 | YawTarget=238.5 | Err=32.8 | Dist=990.2
t=38.0s | X=508.9 | Y=841.6 | Alt=370.0 | Yaw=193.7 | YawTarget=238.8 | Err=33.1 | Dist=984.2
t=39.0s | X=501.1 | Y=839.0 | Alt=360.0 | Yaw=193.7 | YawTarget=239.1 | Err=33.4 | Dist=977.8
t=40.0s | X=493.3 | Y=836.2 | Alt=350.0 | Yaw=193.7 | YawTarget=239.4 | Err=33.8 | Dist=971.4
t=41.0s | X=485.9 | Y=833.4 | Alt=340.0 | Yaw=193.6 | YawTarget=239.7 | Err=34.1 | Dist=965.2
t=42.0s | X=478.2 | Y=830.5 | Alt=330.0 | Yaw=193.6 | YawTarget=240.0 | Err=34.4 | Dist=958.9
t=43.0s | X=470.6 | Y=827.8 | Alt=320.0 | Yaw=193.6 | YawTarget=240.4 | Err=34.7 | Dist=952.8
t=44.0s | X=463.1 | Y=825.0 | Alt=310.0 | Yaw=193.7 | YawTarget=240.6 | Err=35.0 | Dist=946.8
t=45.0s | X=455.4 | Y=822.4 | Alt=300.0 | Yaw=193.7 | YawTarget=241.0 | Err=35.3 | Dist=940.6
t=46.0s | X=447.6 | Y=820.0 | Alt=290.0 | Yaw=193.7 | YawTarget=241.3 | Err=35.6 | Dist=934.7
t=47.0s | X=440.4 | Y=817.2 | Alt=280.0 | Yaw=193.7 | YawTarget=241.7 | Err=35.9 | Dist=928.8
t=48.0s | X=433.1 | Y=814.4 | Alt=270.0 | Yaw=193.7 | YawTarget=242.0 | Err=36.3 | Dist=922.9
t=49.0s | X=425.9 | Y=811.7 | Alt=260.0 | Yaw=193.7 | YawTarget=242.3 | Err=36.6 | Dist=917.1
t=50.0s | X=418.2 | Y=809.1 | Alt=250.0 | Yaw=193.7 | YawTarget=242.6 | Err=36.9 | Dist=911.3
t=51.0s | X=410.6 | Y=806.3 | Alt=240.0 | Yaw=193.8 | YawTarget=243.0 | Err=37.2 | Dist=905.4
```

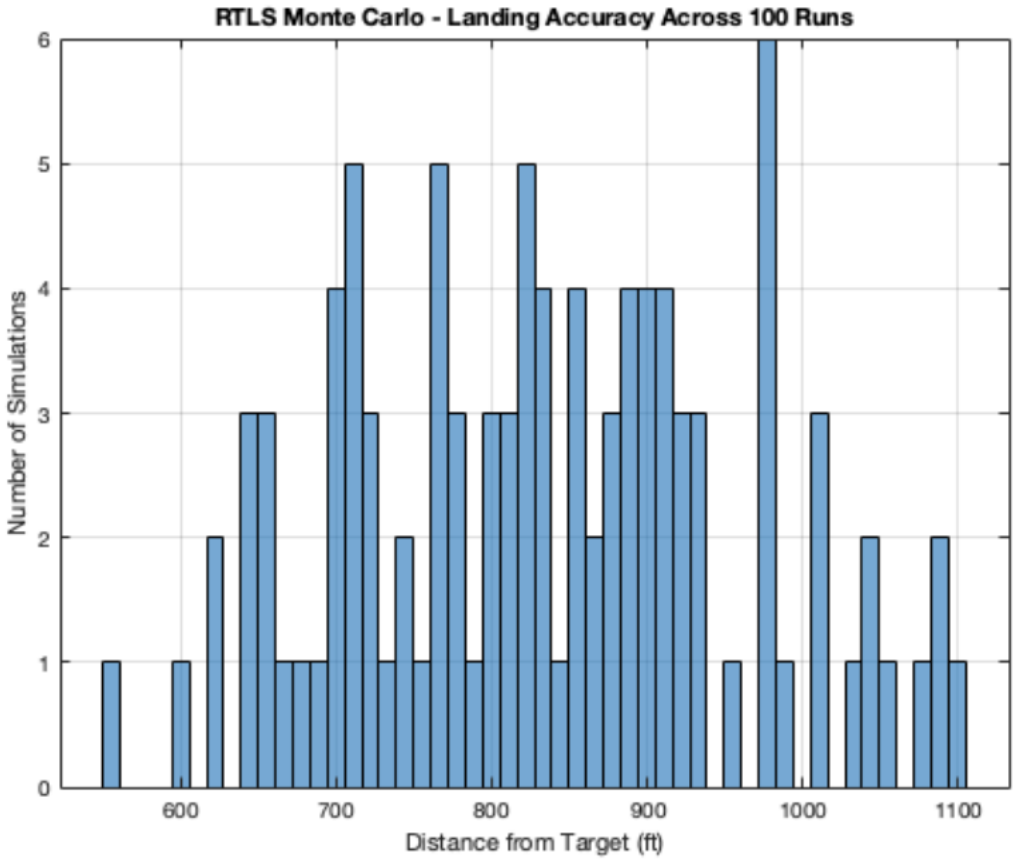
```
t=52.0s | X=403.0 | Y=803.6 | Alt=230.0 | Yaw=193.8 | YawTarget=243.3 | Err=37.5 | Dist=899.6
t=53.0s | X=395.4 | Y=801.1 | Alt=220.0 | Yaw=193.8 | YawTarget=243.7 | Err=37.9 | Dist=893.8
t=54.0s | X=387.6 | Y=798.5 | Alt=210.0 | Yaw=194.2 | YawTarget=244.1 | Err=37.9 | Dist=888.1
t=55.0s | X=379.8 | Y=795.6 | Alt=200.0 | Yaw=194.5 | YawTarget=244.4 | Err=38.0 | Dist=882.2
t=56.0s | X=372.4 | Y=792.8 | Alt=190.0 | Yaw=195.0 | YawTarget=244.8 | Err=38.0 | Dist=876.4
t=57.0s | X=365.7 | Y=789.9 | Alt=180.0 | Yaw=195.2 | YawTarget=245.1 | Err=38.0 | Dist=870.9
t=58.0s | X=358.5 | Y=786.8 | Alt=170.0 | Yaw=195.7 | YawTarget=245.5 | Err=37.9 | Dist=865.2
t=59.0s | X=350.7 | Y=783.8 | Alt=160.0 | Yaw=196.0 | YawTarget=245.9 | Err=37.9 | Dist=859.3
t=60.0s | X=343.5 | Y=780.7 | Alt=150.0 | Yaw=196.3 | YawTarget=246.2 | Err=37.9 | Dist=853.4
t=61.0s | X=335.6 | Y=777.7 | Alt=140.0 | Yaw=196.9 | YawTarget=246.6 | Err=38.0 | Dist=847.6
t=62.0s | X=327.7 | Y=774.6 | Alt=130.0 | Yaw=197.2 | YawTarget=247.0 | Err=37.8 | Dist=841.7
t=63.0s | X=320.1 | Y=771.4 | Alt=120.0 | Yaw=197.5 | YawTarget=247.4 | Err=38.0 | Dist=835.8
t=64.0s | X=312.6 | Y=768.1 | Alt=110.0 | Yaw=197.9 | YawTarget=247.8 | Err=38.0 | Dist=829.9
t=65.0s | X=305.5 | Y=764.9 | Alt=100.0 | Yaw=198.4 | YawTarget=248.2 | Err=37.8 | Dist=824.2
t=66.0s | X=298.1 | Y=761.8 | Alt=90.0 | Yaw=198.7 | YawTarget=248.6 | Err=38.0 | Dist=818.5
t=67.0s | X=290.6 | Y=758.3 | Alt=80.0 | Yaw=199.1 | YawTarget=249.0 | Err=37.9 | Dist=812.6
t=68.0s | X=283.7 | Y=754.9 | Alt=70.0 | Yaw=199.5 | YawTarget=249.4 | Err=37.9 | Dist=807.0
t=69.0s | X=276.8 | Y=751.2 | Alt=60.0 | Yaw=199.9 | YawTarget=249.7 | Err=37.9 | Dist=801.1
t=70.0s | X=269.8 | Y=747.7 | Alt=50.0 | Yaw=200.2 | YawTarget=250.1 | Err=37.9 | Dist=795.3
t=71.0s | X=262.8 | Y=744.3 | Alt=40.0 | Yaw=200.7 | YawTarget=250.5 | Err=38.0 | Dist=789.8
t=72.0s | X=255.1 | Y=740.7 | Alt=30.0 | Yaw=201.0 | YawTarget=250.9 | Err=37.9 | Dist=783.9
t=73.0s | X=248.0 | Y=737.0 | Alt=20.0 | Yaw=201.5 | YawTarget=251.4 | Err=37.8 | Dist=778.1
t=74.0s | X=240.6 | Y=733.2 | Alt=10.0 | Yaw=202.0 | YawTarget=251.8 | Err=37.8 | Dist=772.1
t=75.0s | X=233.7 | Y=729.5 | Alt=0.0 | Yaw=202.3 | YawTarget=252.2 | Err=37.9 | Dist=766.5
```



H.1.1.2. Monte_Carlo_Runner.m

```
% Runs the RTLS flight algorithm 1000 times and records final landing distances

clear; clc;
num_trials = 100;
results = zeros(num_trials, 1);
successful_landings = 0;

for i = 1:num_trials
    try
        % Reset global variable
        global FINAL_DISTANCE
        FINAL_DISTANCE = NaN;


        %run sims
        main(true);

        % Save final distance from current run
        if ~isnan(FINAL_DISTANCE)
            results(i) = FINAL_DISTANCE;

            if FINAL_DISTANCE <= 800
                successful_landings = successful_landings + 1;
            end
        else
            results(i) = NaN;
        end
    catch ME
        warning("Simulation %d failed: %s", i, getReport(ME));
        results(i) = NaN;
    end
end

% Clean up failed runs
valid_results = results(~isnan(results));

% Plot histogram
figure;
histogram(valid_results, 50);
xlabel('Distance from Target (ft)');
ylabel('Number of Simulations');
title('RTLS Monte Carlo - Landing Accuracy Across 100 Runs');
grid on;

% Print stats
fprintf("Monte Carlo Results (n = %d):\n", numel(valid_results));
fprintf("Mean Distance: %.2f ft\n", mean(valid_results));
fprintf("Max Distance: %.2f ft\n", max(valid_results));
fprintf("Min Distance: %.2f ft\n", min(valid_results));
fprintf("Std Deviation: %.2f ft\n", std(valid_results));
fprintf("Successful Landings (<800 ft): %d / %d (%.2f%%)\n", ...
    successful_landings, num_trials, 100 * successful_landings / num_trials);
```

```
Monte Carlo Results (n = 100):
Mean Distance: 834.28 ft
Max Distance: 1101.31 ft
Min Distance: 551.93 ft
Std Deviation: 125.86 ft
Successful Landings (<800 ft): 38 / 100 (38.00%)
```



H.2. Electronics Code

```
// ======================
// Includes & Definitions
// ======================
#include <Arduino.h>
#include <Wire.h>
#include <math.h>
#include <Adafruit_BMP3xx.h>  // BMP388 sensor library
#include <TinyGPS++.h>       // GPS parsing library

// SD Card & Filesystem Libraries
#include "FS.h"
#include "SD.h"
#include "SPI.h"

// --------------------
// GPS Pin Definitions
// --------------------
#define GPS_RX_PIN 16
#define GPS_TX_PIN 17
#define GPS_BAUD 9600

// --------------------
// Radio Definitions
// --------------------
#define RXD1 15           // Radio RX pin
#define TXD1 4            // Radio TX pin
#define RADIO_BAUD 115200
#define Address 2         // Receiver radio address

// --------------------
// IMU Sensor WHO_AM_I Definitions
// --------------------
```

```
#define LSM6DSL_WHO_AM_I    0x0F
#define EXPECTED_LSM6DSL   0x6A    // Expected for LSM6DSL
#define LIS3MDL_WHO_AM_I    0x0F
#define EXPECTED_LIS3MDL   0x3D    // Expected for LIS3MDL

// ---------------------
// Baseline Delay (ms)
// ---------------------
#define BASELINE_DELAY_MS 5000   // Allow sensors to stabilize before capturing baseline

// ---------------------
// Global Variables for Ground Position & Time Initialization
// ---------------------
bool groundPositionCaptured = false;
float groundLat = 0.0;
float groundLon = 0.0;
float groundAltitude_msl = 0.0;
int hours, minutes, seconds;
unsigned long lastMillis = 0;  // For manual time update

// ---------------------
// Global Variables for GPS Altitude Smoothing
// ---------------------
bool gpsAltitudeInitialized = false;
float gpsInitialAltitude = 0.0;
float filteredGpsAltMSL = NAN;
float gpsAltSum = 0.0;
int gpsAltCount = 0;

// ---------------------
// For the Millisecond Offset Column
// ---------------------
unsigned long lastFullSecondMillis = 0;
int lastGpsSecond = 0;
```

```
// --------------------
// BMP388 Setup
// --------------------
Adafruit_BMP3XX bmp;
float initialPressureAltitude = 0.0;      // BMP388 MSL baseline
bool pressureAltitudeInitialized = false;
unsigned long programStartTime;           // When the program started

// --------------------
// IMU (BerryIMUv3) Definitions & Global Variables
// --------------------
#define LSM6DSL_ADDRESS       0x6A
#define LSM6DSL_CTRL1_XL      0x10
#define LSM6DSL_CTRL8_XL      0x17
#define LSM6DSL_CTRL3_C       0x12
#define LSM6DSL_CTRL2_G       0x11
#define LSM6DSL_OUT_X_L_XL    0x28
#define LSM6DSL_OUT_X_L_G     0x22

#define LIS3MDL_ADDRESS       0x1C
#define LIS3MDL_CTRL_REG1     0x20
#define LIS3MDL_CTRL_REG2     0x21
#define LIS3MDL_CTRL_REG3     0x22
#define LIS3MDL_OUT_X_L       0x28

// Timing and Filtering Definitions for IMU Processing:
#define DT      0.01      // Loop interval: 10 ms (100 Hz)
#define AA      0.97      // Complementary filter constant
#define G_GAIN  0.070     // Gyroscope gain

byte buff[6];
int accRaw[3], gyrRaw[3], magRaw[3];
float rate_gyr_x = 0.0, rate_gyr_y = 0.0, rate_gyr_z = 0.0;
```

```
float gyroXangle = 0.0, gyroYangle = 0.0, gyroZangle = 0.0;
float AccXangle = 0.0, AccYangle = 0.0, AccZangle = 0.0;
float CFangleX = 0.0, CFangleY = 0.0, CFangleZ = 0.0;
unsigned long loopStartTime;

// --------------------
// Black Powder (BP) Global Variables
// --------------------
int BP_1 = 0;  // Black Powder Charge 1 state (off)
int BP_2 = 0;  // Black Powder Charge 2 state (off)

// --------------------
// Create the TinyGPS++ Object and Hardware Serial for GPS
// --------------------
TinyGPSPlus gps;
HardwareSerial SerialGPS(2);   // GPS on Serial2

// --------------------
// Create Hardware Serial for Radio
// --------------------
HardwareSerial radioSerial(1); // Radio on Serial1

// --------------------
// Global Variable for Log File Name
// --------------------
char logFileName[32] = "";   // Will hold the EST date/time as filename

// --------------------
// I2C Communication Functions for IMU
// --------------------
void writeTo(int device, byte address, byte val) {
  Wire.beginTransmission(device);
  Wire.write(address);
  Wire.write(val);
```

```
  Wire.endTransmission();
}

void readFrom(int device, byte address, int num, byte buff[]) {
  Wire.beginTransmission(device);
  Wire.write(address);
  Wire.endTransmission();
  Wire.beginTransmission(device);
  Wire.requestFrom(device, num);
  int i = 0;
  while (Wire.available()) {
    buff[i++] = Wire.read();
  }
  Wire.endTransmission();
}

// ---------------------
// Sensor Check Function
// ---------------------
bool sensorCheck() {
  byte buffer[1];
  // Check LSM6DSL
  readFrom(LSM6DSL_ADDRESS, LSM6DSL_WHO_AM_I, 1, buffer);
  if (buffer[0] != EXPECTED_LSM6DSL) {
    Serial.print("Error: LSM6DSL WHO_AM_I incorrect. Expected 0x");
    Serial.print(EXPECTED_LSM6DSL, HEX);
    Serial.print(", got 0x");
    Serial.println(buffer[0], HEX);
    return false;
  }
  // Check LIS3MDL
  readFrom(LIS3MDL_ADDRESS, LIS3MDL_WHO_AM_I, 1, buffer);
  if (buffer[0] != EXPECTED_LIS3MDL) {
    Serial.print("Error: LIS3MDL WHO_AM_I incorrect. Expected 0x");
```

```
    Serial.print(EXPECTED_LIS3MDL, HEX);
    Serial.print(", got 0x");
    Serial.println(buffer[0], HEX);
    return false;
  }
  return true;
}

// ---------------------
// IMU Initialization (BerryIMUv3)
// ---------------------
void enableIMU() {
  // LSM6DSL Accelerometer
  writeTo(LSM6DSL_ADDRESS, LSM6DSL_CTRL1_XL, 0b10011111);
  writeTo(LSM6DSL_ADDRESS, LSM6DSL_CTRL8_XL, 0b11001000);
  writeTo(LSM6DSL_ADDRESS, LSM6DSL_CTRL3_C,  0b01000100);
  // LSM6DSL Gyroscope
  writeTo(LSM6DSL_ADDRESS, LSM6DSL_CTRL2_G,  0b10011100);
  // LIS3MDL Magnetometer
  writeTo(LIS3MDL_ADDRESS, LIS3MDL_CTRL_REG1, 0b11011100);
  writeTo(LIS3MDL_ADDRESS, LIS3MDL_CTRL_REG2, 0b00100000);
  writeTo(LIS3MDL_ADDRESS, LIS3MDL_CTRL_REG3, 0b00000000);
}

// ---------------------
// Sensor Read Functions for IMU
// ---------------------
void readACC(byte buff[]){
  readFrom(LSM6DSL_ADDRESS, LSM6DSL_OUT_X_L_XL, 6, buff);
}
void readGYR(byte buff[]){
  readFrom(LSM6DSL_ADDRESS, LSM6DSL_OUT_X_L_G, 6, buff);
}
void readMAG(byte buff[]){
```

```
  readFrom(LIS3MDL_ADDRESS, 0x80 | LIS3MDL_OUT_X_L, 6, buff);
}

// ---------------------
// Magnetometer Calibration Variables & Function
// ---------------------
float magOffsetX = 0;
float magOffsetY = 0;
float magOffsetZ = 0;

// This routine averages several magnetometer samples.
// IMPORTANT: Hold the sensor with its x-axis pointed toward magnetic north during calibration.
void calibrateMag() {
  const int numSamples = 100;
  long sumX = 0, sumY = 0, sumZ = 0;
  for (int i = 0; i < numSamples; i++) {
    readMAG(buff);
    int mx = (int)(buff[0] | (buff[1] << 8));
    int my = (int)(buff[2] | (buff[3] << 8));
    int mz = (int)(buff[4] | (buff[5] << 8));
    sumX += mx;
    sumY += my;
    sumZ += mz;
    delay(20);  // Adjust delay as necessary
  }
  magOffsetX = sumX / numSamples;
  magOffsetY = sumY / numSamples;
  magOffsetZ = sumZ / numSamples;

  Serial.print("Mag calibration offsets: ");
  Serial.print(magOffsetX);
  Serial.print(", ");
  Serial.print(magOffsetY);
  Serial.print(", ");
```

```
  Serial.println(magOffsetZ);
}

// --------------------
// Function to Send AT Commands to the Radio and Print Responses
// --------------------
void sendCommand(HardwareSerial& radioSerial, String command) {
  Serial.print("Sending command: ");
  Serial.print(command);
  radioSerial.print(command);
  // Wait a moment for a response (if any)
  delay(50);
  if (radioSerial.available()) {
    String response = radioSerial.readStringUntil('\n');
    Serial.println("Response: " + response);
  } else {
    Serial.println("No response received.");
  }
}

// --------------------
// SD Card Logging Function
// --------------------
void logData(const char* data) {
  File file = SD.open(logFileName, FILE_APPEND);
  if (file) {
    file.println(data);
    file.close();
  } else {
    Serial.println("Failed to open log file for appending");
  }
}

// ======================
```

```
// Setup Function
// ======================
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println();

  // --- Wait for SD Card to come online ---
  unsigned long sdStartTime = millis();
  while (!SD.begin()) {
    Serial.println("Waiting for SD card to come online...");
    delay(1000);
    // Optionally print a message if waiting for more than 3 minutes
    if (millis() - sdStartTime > 180000) {
      Serial.println("SD card still not ready after 3 minutes, continuing to wait...");
    }
  }
  Serial.println("SD card online and ready for writing.");

  Wire.begin();
  delay(500);

  // BP Setup: Initialize BP pins and reset states
  pinMode(25, OUTPUT);  // BP_1 + (3.3V)
  pinMode(26, OUTPUT);  // BP_1 - (0V)
  pinMode(27, OUTPUT);  // BP_2 + (3.3V)
  pinMode(14, OUTPUT);  // BP_2 - (0V)
  digitalWrite(25, LOW);
  digitalWrite(26, LOW);
  digitalWrite(27, LOW);
  digitalWrite(14, LOW);

  // Sensor check for IMU devices
  if (!sensorCheck()) {
```

```
  Serial.println("IMU Sensor Check Failed! Halting.");
  while (1);
 }

 // Initialize BMP388 sensor
 if (!bmp.begin_I2C()) {
  Serial.println("Error: Couldn't find BMP388 sensor.");
  while (1);
 }
 Serial.println("BMP388 Barometric Sensor Initialized.");
 bmp.setTemperatureOversampling(2);
 bmp.setPressureOversampling(3);
 bmp.setIIRFilterCoeff(2);
 programStartTime = millis();

 // Initialize BerryIMUv3 sensors
 enableIMU();

 // *** Calibrate Magnetometer ***
 // Hold the sensor with its x-axis pointed toward magnetic north.
 calibrateMag();
 // Print a reminder note:
  Serial.println("NOTE: For compass calibration to work as intended, be sure to keep the sensor stationary in the proper
orientation during startup.");

 // Start GPS module on Serial2
 SerialGPS.begin(GPS_BAUD, SERIAL_8N1, GPS_RX_PIN, GPS_TX_PIN);
 Serial.println("GPS Module Initialized.");

 // --- Radio Initialization ---
 radioSerial.begin(RADIO_BAUD, SERIAL_8N1, RXD1, TXD1);
 delay(500);
 Serial.println("Radio Serial Setup Complete.");
 sendCommand(radioSerial, "AT\r\n");
```

```
sendCommand(radioSerial, "AT+ADDRESS=1\r\n");
sendCommand(radioSerial, "AT+NETWORKID=5\r\n");
sendCommand(radioSerial, "AT+BAND=915000000\r\n");
sendCommand(radioSerial, "AT+PARAMETER=10,7,1,4\r\n");

// ---- Ground Position & Time Initialization ----
Serial.println("Waiting for valid GPS data for ground position and time...");
while (!groundPositionCaptured) {
  while (SerialGPS.available() > 0) {
    gps.encode(SerialGPS.read());
  }
  if (gps.location.isValid() && gps.altitude.isValid() && gps.time.isValid()) {
    groundLat = gps.location.lat();
    groundLon = gps.location.lng();
    groundAltitude_msl = gps.altitude.meters();
    groundPositionCaptured = true;

    // Get UTC time from GPS and convert to EST (UTC-5 adjustment)
    hours = gps.time.hour();
    minutes = (gps.time.minute() - 4);
    seconds = (gps.time.second() + 11);
    hours = (hours + 19) % 24; // Convert UTC to EST

    lastMillis = millis();
    lastFullSecondMillis = millis();
    lastGpsSecond = gps.time.second();

    Serial.println("Ground position captured:");
    Serial.print("Lat: "); Serial.print(groundLat, 6);
    Serial.print(", Lon: "); Serial.print(groundLon, 6);
    Serial.print(", Alt (MSL): "); Serial.print(groundAltitude_msl, 2);
    Serial.println(" m");

    char initTime[9];
```

```
      sprintf(initTime, "%02d:%02d:%02d", hours, minutes, seconds);
      Serial.print("Initial Time (EST): ");
      Serial.println(initTime);

      // ---- Generate the Log File Name using EST Date and Time ----
      if (gps.date.isValid()) {
        int year = gps.date.year();
        int month = gps.date.month();
        int day = gps.date.day();
        sprintf(logFileName, "/%04d%02d%02d_%02d%02d%02d.txt", year, month, day, hours, minutes, seconds);
      } else {
        sprintf(logFileName, "/EST_%02d%02d%02d.txt", hours, minutes, seconds);
      }
      Serial.print("Log file name: ");
      Serial.println(logFileName);

      // Create an empty log file
      File logFile = SD.open(logFileName, FILE_WRITE);
      if(logFile){
        logFile.close();
        Serial.println("Log file created.");
      } else {
        Serial.println("Failed to create log file.");
      }
    }
  }
}

// ======================
// Main Loop Function
// ======================
void loop() {
  loopStartTime = millis();
```

```
// --- GPS Data Parsing ---
while (SerialGPS.available() > 0) {
  gps.encode(SerialGPS.read());
}

// Update manual time every second based on lastMillis
if (millis() - lastMillis >= 1000) {
  lastMillis += 1000;
  seconds++;
  if (seconds >= 60) {
    seconds = 0;
    minutes++;
  }
  if (minutes >= 60) {
    minutes = 0;
    hours = (hours + 1) % 24;
  }
}

// Update GPS location if available
float gpsLatitude = NAN, gpsLongitude = NAN;
if (gps.location.isValid()) {
  gpsLatitude = gps.location.lat();
  gpsLongitude = gps.location.lng();
}

int gpsSatellites = gps.satellites.value();
float gpsHDOP = gps.hdop.value() / 100.0;

// Process GPS altitude (raw MSL)
float gpsAltMSL = NAN, gpsAltAGL = NAN;
if (gps.altitude.isValid()) {
  float newGpsAlt = gps.altitude.meters();
  // For the first BASELINE_DELAY_MS, average the altitude for baseline
```

```
  if (millis() - programStartTime < BASELINE_DELAY_MS) {
    gpsAltSum += newGpsAlt;
    gpsAltCount++;
    filteredGpsAltMSL = gpsAltSum / gpsAltCount;
  } else {
    if (!gpsAltitudeInitialized) {
      filteredGpsAltMSL = gpsAltSum / gpsAltCount;
      gpsInitialAltitude = filteredGpsAltMSL;
      gpsAltitudeInitialized = true;
      Serial.println("GPS Initial Altitude Set from Average.");
    } else {
      // Apply low-pass filter
      filteredGpsAltMSL = 0.2 * newGpsAlt + 0.8 * filteredGpsAltMSL;
    }
  }
  gpsAltMSL = filteredGpsAltMSL;
  gpsAltAGL = gpsAltMSL - gpsInitialAltitude;
}

// Update the millisecond offset column:
if (gps.time.isValid()) {
  int currentGpsSecond = gps.time.second();
  if (currentGpsSecond != lastGpsSecond) {
    lastGpsSecond = currentGpsSecond;
    lastFullSecondMillis = millis();
  }
}
unsigned long msOffset = millis() - lastFullSecondMillis;

// --- BMP388 Sensor Readings ---
float temperature = NAN, currentPressureAltitude = NAN, pressureAltAGL = NAN;
if (bmp.performReading()) {
  temperature = bmp.temperature;
  currentPressureAltitude = bmp.readAltitude(1013.25);
```

```
  // Wait until BASELINE_DELAY_MS has elapsed before setting the baseline
  if (!pressureAltitudeInitialized && (millis() - programStartTime > BASELINE_DELAY_MS)) {
    initialPressureAltitude = currentPressureAltitude;
    pressureAltitudeInitialized = true;
    Serial.println("Pressure Sensor Initial Altitude Set.");
  }
  pressureAltAGL = currentPressureAltitude - initialPressureAltitude;
}

// --- BerryIMUv3 Sensor Readings ---
// Accelerometer
readACC(buff);
accRaw[0] = (int)(buff[0] | (buff[1] << 8));
accRaw[1] = (int)(buff[2] | (buff[3] << 8));
accRaw[2] = (int)(buff[4] | (buff[5] << 8));

// Magnetometer (apply calibration offsets)
readMAG(buff);
magRaw[0] = (int)(buff[0] | (buff[1] << 8)) - magOffsetX;
magRaw[1] = (int)(buff[2] | (buff[3] << 8)) - magOffsetY;
magRaw[2] = (int)(buff[4] | (buff[5] << 8)) - magOffsetZ;

// Gyroscope
readGYR(buff);
gyrRaw[0] = (int)(buff[0] | (buff[1] << 8));
gyrRaw[1] = (int)(buff[2] | (buff[3] << 8));
gyrRaw[2] = (int)(buff[4] | (buff[5] << 8));

// Process Gyroscope Data
rate_gyr_x = (float)gyrRaw[0] * G_GAIN;
rate_gyr_y = (float)gyrRaw[1] * G_GAIN;
rate_gyr_z = (float)gyrRaw[2] * G_GAIN;

gyroXangle += rate_gyr_x * DT;
```

```
gyroYangle += rate_gyr_y * DT;
gyroZangle += rate_gyr_z * DT;

// Compute Accelerometer Angles (in degrees)
AccXangle = (float)(atan2(accRaw[1], accRaw[2]) + PI) * 180.0 / PI;
AccYangle = (float)(atan2(accRaw[2], accRaw[0]) + PI) * 180.0 / PI;
AccZangle = (float)(atan2(accRaw[0], accRaw[1]) + PI) * 180.0 / PI;

// Adjust angles per BerryIMU method
AccXangle -= 180.0;
AccZangle -= 180.0;
if (AccYangle > 90)
  AccYangle -= 270;
else
  AccYangle += 90;

// Complementary Filter
CFangleX = AA * (CFangleX + rate_gyr_x * DT) + (1 - AA) * AccXangle;
CFangleY = AA * (CFangleY + rate_gyr_y * DT) + (1 - AA) * AccYangle;
CFangleZ = AA * (CFangleZ + rate_gyr_z * DT) + (1 - AA) * AccZangle;

// --- Tilt-Compensated Heading Calculation ---
// Calculate roll and pitch from the accelerometer data
float ax = (float)accRaw[0];
float ay = (float)accRaw[1];
float az = (float)accRaw[2];
float roll  = atan2(ay, az);
float pitch = atan2(-ax, sqrt(ay * ay + az * az));

// Use the calibrated magnetometer data for tilt compensation
float mx = (float)magRaw[0];
float my = (float)magRaw[1];
float mz = (float)magRaw[2];
```

```
// Tilt compensated magnetometer measurements
float magX_comp = mx * cos(pitch) + mz * sin(pitch);
float magY_comp = mx * sin(roll) * sin(pitch) + my * cos(roll) - mz * sin(roll) * cos(pitch);

// Compute the heading in degrees
float heading = atan2(magY_comp, magX_comp) * 180.0 / PI;
if (heading < 0)
  heading += 360;

// --- Build Manual Time String ---
char manualTime[9];
sprintf(manualTime, "%02d:%02d:%02d", hours, minutes, seconds);

// --- Build Formatted Output String ---
char outputBuffer[512];
sprintf(outputBuffer,
        "Time: %s.%03lu, GPS Lat: %10.6f, GPS Long: %10.6f, Sat: %2d, HDOP: %5.2f, "
        "GPS(MSL): %6.2f m, GPS(AGL): %6.2f m, "
        "P(MSL): %6.2f m, P(AGL): %6.2f m, Temp: %6.2f C, "
        "AccX: %7.2f, AccY: %7.2f, AccZ: %7.2f, "
        "GyrX: %7.2f, GyrY: %7.2f, GyrZ: %7.2f, "
        "CF_X: %7.2f, CF_Y: %7.2f, CF_Z: %7.2f, "
        "MagX: %d, MagY: %d, Heading: %7.2f, LoopTime: %3lu",
        manualTime, msOffset,
        gpsLatitude, gpsLongitude, gpsSatellites, gpsHDOP,
        gpsAltMSL, gpsAltAGL, currentPressureAltitude, pressureAltAGL, temperature,
        AccXangle, AccYangle, AccZangle,
        gyroXangle, gyroYangle, gyroZangle,
        CFangleX, CFangleY, CFangleZ,
        magRaw[0], magRaw[1], heading, millis() - loopStartTime);

Serial.println(outputBuffer);

// --- Radio Transmission ---
```

```
// Build the radio message using only:
// Time(EST), GPS Lat, GPS Long, GPS(AGL), P(AGL), HEADING.
if (gps.location.isUpdated()) {
  String radioData = "TIME:" + String(manualTime) +
                     ", LAT:" + String(gpsLatitude, 4) +
                     ", LONG:" + String(gpsLongitude, 4) +
                     ", GPS(AGL):" + String(gpsAltAGL, 2) +
                     ", P(AGL):" + String(pressureAltAGL, 2) +
                     ", HEADING:" + String(heading, 2);
  int payloadLength = radioData.length();
  String radioCommand = "AT+SEND=" + String(Address) + "," + String(payloadLength) + "," + radioData + "\r\n";
  sendCommand(radioSerial, radioCommand);
  Serial.println("Radio Message Sent");
}

// --- Black Powder (BP) Control ---
// Trigger BP_1 based on its state
if (BP_1 == 1) {
  digitalWrite(25, HIGH);  // BP_1 + (3.3V)
  digitalWrite(26, LOW);   // BP_1 - (0V)
} else {
  digitalWrite(25, LOW);   // BP_1 + (0V)
  digitalWrite(26, HIGH);  // BP_1 - (3.3V)
}

// Trigger BP_2 based on its state
if (BP_2 == 1) {
  digitalWrite(27, HIGH);  // BP_2 + (3.3V)
  digitalWrite(14, LOW);   // BP_2 - (0V)
} else {
  digitalWrite(27, LOW);   // BP_2 + (0V)
  digitalWrite(14, HIGH);  // BP_2 - (3.3V)
}
```

```
  // --- Log the Output Buffer to SD Card ---
  logData(outputBuffer);

  // Wait until the loop period is complete (10 ms)
  while (millis() - loopStartTime < (DT * 1000)) {
    delay(2);
  }
}
```

References

[1] S. Khan. "DIFFERENT TYPES of MANUFACTURING PROCESSES | the “ALL in ONE” GUIDE." https://www.linkedin.com/pulse/different-types-manufacturing-processes-all-one-guide-sayeed-afzal/ (accessed.

[2] S. A. E. M. Group. "Additive Manufacturing Subtracts from Rocket Build Time." https://www.techbriefs.com/component/content/article/50267-additive-manufacturing-subtracts-from-rocket-build-time (accessed.

[3] P. Exploration. "Evolution Rocket Manufacturing Techniques: From Conventional to Additive." https://www.linkedin.com/pulse/evolution-rocket-manufacturing-techniques-from-conventional-additive/ (accessed.

[4] M. T. Birosz and et al., "Effect of FDM Infill Patterns on Mechanical Properties," *Polymer Testing*, vol. 113, p. 107654, 2022, doi: 10.1016/j.polymertesting.2022.107654.

[5] V. Shanmugam and et al., "The Mechanical Testing and Performance Analysis of Polymer-Fibre Composites Prepared through the Additive Manufacturing," *Polymer Testing*, vol. 93, p. 106925, 2021, doi: 10.1016/j.polymertesting.2020.106925.

[6] Y. P. Shaik and et al., "The Comparison of the Mechanical Characteristics of ABS Using Three Different Plastic Production Techniques," *Open Access Library Journal*, vol. 10, no. 05, pp. 1-18, 2023, doi: 10.4236/oalib.1110097.

[7] H. Worthy. "Subtractive Manufacturing vs Additive Manufacturing: What Are Their Advantages, Disadvantages, Differences." https://www.worthyhardware.com/news/subtractive-manufacturing-vs-additive-manufacturing-what-are-their-advantages-disadvantages-differences/ (accessed.

[8] A. Klvarez. "Subtractive Manufacturing: Past, Present, and Future." https://ketiv.com/blog/subtractive-manufacturing-past-present-and-future/ (accessed.

[9] J. Lie. "Different Types of Casting Process, Die Casting Process." https://www.runsom.com/blog/10-different-types-of-casting-process/ (accessed.

[10] "Metal 3D Printing vs. Casting vs. CNC: Which Is Better?" https://www.thesteelprinters.com/news/metal-3d-printing-vs-casting-vs-cnc-which-is-better (accessed.

[11] Shane. "10 Different Types of Casting Process." https://www.machinemfg.com/types-of-casting/ (accessed.

[12] M. Dasilva, "Rocket Stablility." [Online]. Available: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/rocket-stability/#why-is-flight-direction-vertical

[13] J. Anderson, *Fundamentals of Aerodynamics*, 6 ed. (McGraw-Hill series in aeronautical and aerospace engineering). McGrawHill, 2017.

[14] A. R. Iyer and A. Pant, "A review on nose cone designs for different flight regimes," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 08, pp. 3546-3554, 2020.

[15] A. Pektaş, Z. Demircan, Ü. Hacıabdullahoğlu, N. Ejder, and C. Tola, "Effects of different fin shapes on apogee and stability of model rockets," in *2019 9th International conference on recent advances in space technologies (RAST)*, 2019: IEEE, pp. 193-199.

[16] B. Hennin. (2012) Why Should You Airfoil Your Fins? *Peak of Flight*.

[17] N. A. o. Rocketry. "Standard Motor Codes." https://www.nar.org/standards-and-testing-committee/standard-motor-codes/ (accessed.

[18] G. Dahlke, "TECHNICAL WORKFORCE DEVELOPMENT PROGRAM

STUDENT GUIDE," NASA Kennedy Space Center, 2024.
[19] A. Rockets. "Rocket Motor Basics." https://www.apogeerockets.com/Rocket-Motor-Basics-Quick-Start-Guide (accessed 2024).
[20] thrustcurve.org. "Motor Guide Help." https://www.thrustcurve.org/motors/guidehelp.html (accessed 2024).
[21] A. Rockets. (2004) Selecting Rocket Motors: A Step-by-Step Procedure. *Peak of Flight*. Available: https://www.apogeerockets.com/education/downloads/Newsletter132.pdf
[22] R. N. Hernandez, "Design and Performance of Modular 3-D Printed Solid-Propellant Rocket Airframes," *Aerospace*, vol. 4, no. 2, p. 17, 2017, doi: 10.3390/aerospace4020017.
[23] K. V. Avramov, "Strength of Composite Transport and Launch Container for Rocket Launch," *Journal of Mechanical Engineering – Problemy Mashynobuduvannia*, vol. 26, no. 4, pp. 17-22, 2023, doi: 10.15407/pmach2023.04.017.
[24] C. Barile, "Mechanical Comparison of New Composite Materials for Aerospace Applications," *Composites Part B: Engineering*, vol. 162, pp. 122-128, 2019, doi: 10.1016/j.compositesb.2018.10.101.
[25] "Beginner's Epoxy Question." www.rocketryforum.com/threads/beginners-epoxy-question.158037/ (accessed.
[26] B. R. Murray, K. Matthews, and R. Telford, "Testing Interfaces of Lattice Spacecraft Structures," 2023. [Online]. Available: www.atg-europe.com/wp-content/uploads/ECSSMET2023_Testing-interfaces-of-lattice-spacecraft-structures.pdf.
[27] C. B. Nathan Akers, et al. "Rocket Project." http://ftp.demec.ufpr.br/foguete/bibliografia/Complete%20Project%20Rocket%20+%20Design.pdf (accessed Decmenber 8, 2014).
[28] Ukroc, "Materials for Model Rockets," 2023. [Online]. Available: www.ukroc.com/wp-content/uploads/2023/02/Materials-for-Model-Rockets-2023.pdf.
[29] R. Cyclone. "Mechanical Team." Iowa State University College of Engineering. https://stuorgs.engineering.iastate.edu/cyclone-rocketry/mechanical-team/ (accessed.
[30] NASA. "Avionics: The “Brains” Command NASA’s Deep Space Rocket." https://www.nasa.gov/reference/avionics-the-brains-command-nasas-deep-space-rocket/ (accessed.
[31] J. S. L. Reddy, "Avionics System For Model Rocket," *Journal of Emerging Technologies and Innovative Research*, vol. 10, no. 4, 2014.
[32] NASA. "Guidance System." www1.grc.nasa.gov/beginners-guide-to-aeronautics/guidance-system/ (accessed.
[33] C.-H. Chuang and L. Ledsinger, "First and second variation analysis for return to launch site guidance," 1996: 21st Atmospheric Flight Mechanics Conference, doi: https://doi.org/10.2514/6.1996-3426.
[34] R. J. M. Rao. "What is Telemetry? Definition, Purpose, Applications." https://automationcommunity.com/what-is-telemetry/ (accessed 2024).
[35] MakersBox. "Radio Telemetry for a Model Rocket." https://www.instructables.com/Radio-Telemetry-for-a-Model-Rocket/ (accessed 2024).
[36] jetsman97, "Building Avionic System," vol. 2024, ed, 2022.
[37] A. Tripoli Rocketry. "Level 1 Certification." https://www.tripoli.org/Level1 (accessed.

[38] R. Apogee. "Intro to Dual Deployment in Rocketry." https://www.apogeerockets.com/Intro-to-Dual-Deployment#:~:text=The%20key%20to%20the%20system,the%20altitude%20of%20the%20rocket (accessed.

[39] "Model Rocket Parachute vs. Streamers: When to Use and Avoid." https://themodelrocket.com/model-rocket-parachute-vs-streamers-when-to-use-and-avoid/ (accessed.

[40] "Concept of Operations for Rocket Flight." https://www.researchgate.net/figure/Concept-of-Operations-for-rocket-flight_fig1_330566002 (accessed.

[41] C. W. H. Adam J. Yingling, Thomas A. Seigenthaler, Oleg A. Yakimenko, "Miniature Autonomous Rocket Recovery System (MARRS)," presented at the A Aerodynamic Decelerator Systems Technology Conference and Seminar, Dublin, Ireland, 2011. [Online]. Available: https://nps.edu/documents/106608270/107784480/Snowflake+-+Yingling+-+Miniature+Autnomous+Rocket+Recovery+System+-MARRS.pdf/0a9d66f2-977b-4175-9116-5d5cedd3f96f.

[42] G. R. Bailey, "Guided Parafoil System for Lightweight Payloads," United States, 2002.

[43] N. A. o. Rocketry. "Laws and Regulations." https://www.nar.org/find-a-local-club/section-guidebook/laws-regulations/ (accessed.

[44] T. R. Association. "Tripoli Rocketry Association Safety Code." https://tripoli.org/content.aspx?page_id=22&club_id=795696&module_id=520420 (accessed.

[45] "Original Prusa XL (Assembled) 5-toolhead 3D printer," Accessed 17 Feb 2025 2023. [Online]. Available: https://www.prusa3d.com/en/product/original-prusa-xl-assembled-5-toolhead-3d-printer/.