Team BFR - Final Report

David Aleman. Tommy Beres. Preston Burke.

Brendan Morgan. Vlad Rosca. James Tseng.

Table of Contents

Acronyms, Abbreviations, and Symbols

α	Angle of Attack	ρ	Air Density
CAD	Computer-Aided Design	V	Velocity
C_d	Drag Coefficient	W	Weight
CG	Center of Gravity	m	Mass
C_l	Lift Coefficient	$\overline{m}$	Mass Flow Rate
D	Drag Force	F	Force
a	Acceleration	z	Altitude
T	Thrust	t	Time
I	Impulse	g	Acceleration of Gravity
θ	Angle	PLA	Polylactic Acid
PVC	Polyvinyl Chloride	A	Cross-sectional Area

1. Introduction

The realm of rocket building stands as a testament to humanity's unyielding spirit of exploration and the relentless pursuit of knowledge. Rockets have served as the gateways to the unknown, propelling us beyond Earth's atmosphere and opening the doors to the vast wonders of space. In this final paper, we examine the design principles and components that form the backbone of our rocket, the BFR.

The following design report details the design, manufacturing, testing, and optimization of the BFR. According to mission parameters, the team was tasked to build a rocket capable of launching and returning the payload, an egg, in an undamaged state while at the same time optimizing for maximum altitude. To best meet these requirements, BFR was designed as a 21.25 in. long single-stage rocket with active control capabilities post-burnout to continuously correct the trajectory. Throughout our analysis phase, we estimated an apogee of 1050ft and a 9.79ft/s landing velocity, deemed safe enough for the payload.

In order to achieve active control capabilities, accelerometer, gyroscope, and magnetometer measurements were analyzed through a PD controller. The onboard flight software is operated in a state machine to ensure the proper progression of states and sensor readings.

The testing phase involved a drop test for the recovery system yielding a terminal velocity of 9.5-15 ft/s, a wind tunnel test that resulted in a drag coefficient between 0.95 and 1.05, and passive and active stability tests, which the BFR passed. These results were later used to optimize and refine our final design parameters.

The rocket launch performed in the desert yielded unfavorable results, mostly due to structural integrity failure, but the final result was satisfactory because the payload survived with only a few cracks.

2. Conceptual Design

2.1. Open Rocket Design

The initial layout of the rocket was done in Open Rocket, as shown in Figure 2.1 below. The rocket was about 23.25” long in total, with a 1.5 body tube. The fins were chosen to be three in number to save on weight. Each fin had a 5” root chord, a 2” tip chord, and a 3” height. The sweep of the fins was selected so that the bottom edge would be horizontal - this made manufacturing easier, as the bottom section of the fins was cut from piano hinges, of which the bottom edge was simply left intact. Total stability, as shown in Open Rocket, was 1.91. This was selected to ensure adequate stability off the rail during the burn but low enough that the actively stabilized fins could have some effect on the trajectory of the rocket.

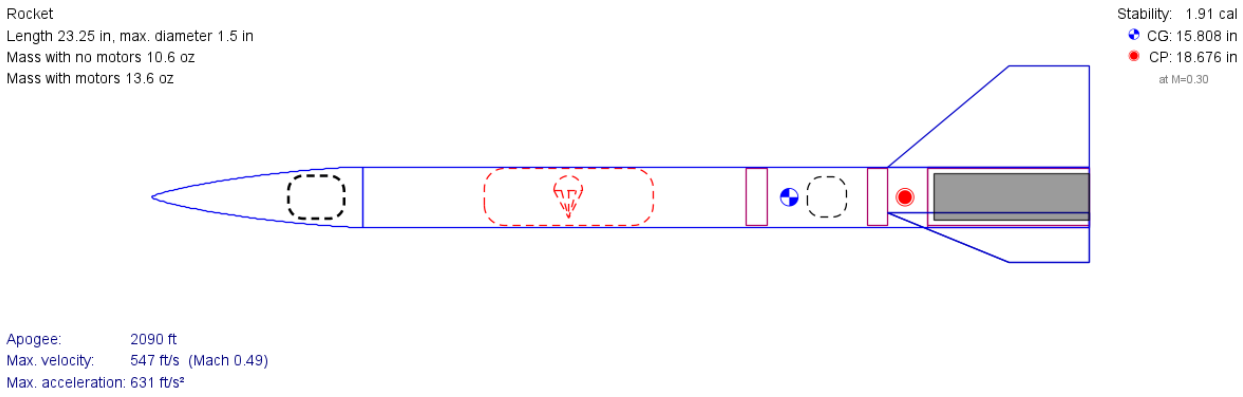

Figure 2.1: Open Rocket Design

2.2. CAD Design

Our CAD design, shown in Figure 2.2 below, was our initial design for the rocket and served as a guide for the early part of the manufacturing process. Our final design is shown in Figure 2.3, highlights the differences between the two designs. Some design changes to highlight are the servo motors and electronics location. Initially, the servo motors were going to be placed between two bulkheads with a tube that would allow the ejection charge to pass by the servos safely. One such concept for this is shown in Figure 2.2. Instead, for simplicity, the tubes were dropped in favor of gluing the servos in with no protection since the servo motors were not a priority to recover. Unlike the servo motors, the electronics (Teensy, altimeter, etc.) had to be

recovered and were moved above the parachute into a separate body tube connected to the nose cone to protect them from the ejection charge and for its accessibility. Many of these design changes and choices were influenced by our ability to manufacture and reliably assemble accurately. The dimensions of our final design can be found in Figure 2.4.

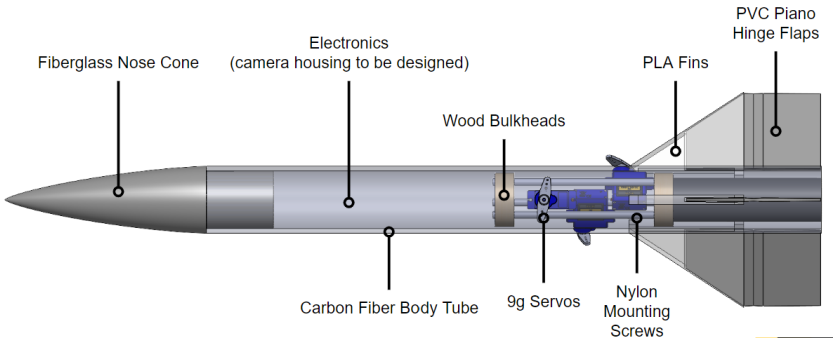

Figure 2.2: Initial Design Concept

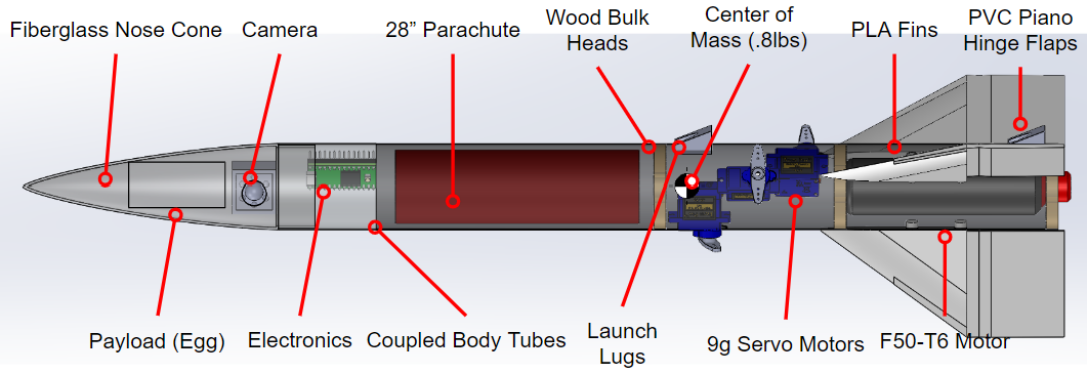

Figure 2.3: Final Design

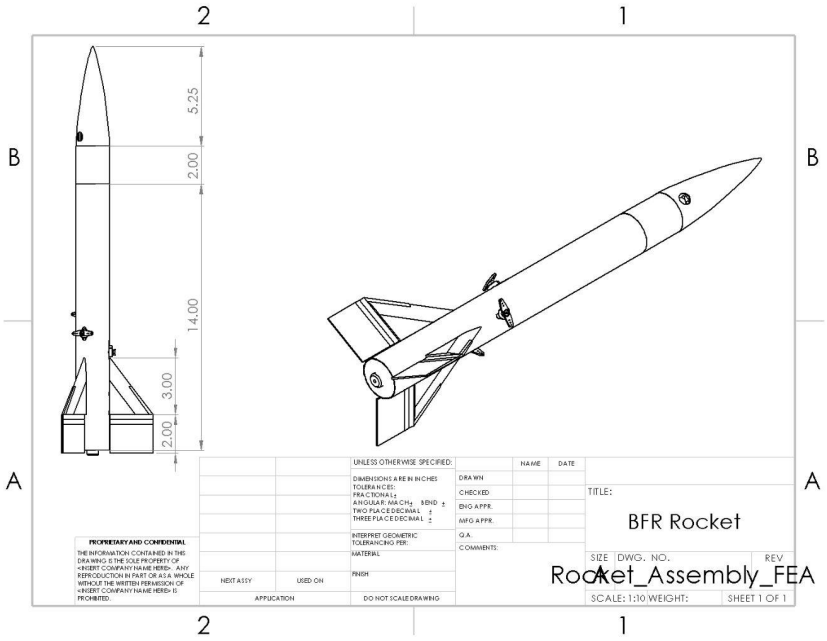

Figure 2.4: Final Design Dimension Drawing

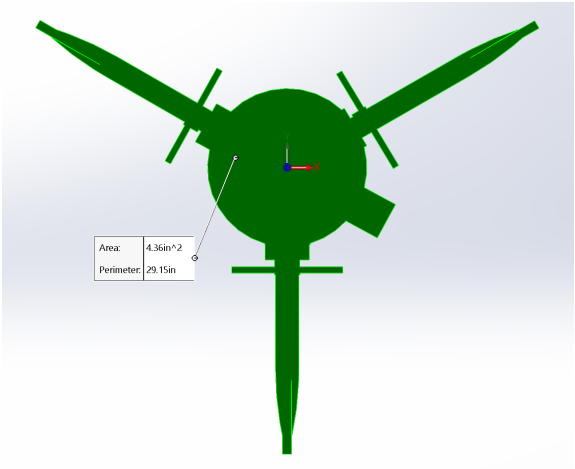

Figure 2.5: Projected Rocket Area

3. Analysis

3.1. Stability Analysis

Hand calculations for stability, which used dimensions from the Solidworks CAD for reference, showed a wet stability margin of 2.03. Open Rocket predicted a similar result at 1.91. This margin was designed to be sufficiently high that the rocket would be stable off the rail during the burn but low enough that the active control system (which switches on post-burn), would have sufficient control authority to modify the trajectory of the rocket during its coast phase.

3.2. Trajectory Analysis

3.2.1. Theory

Using Newton's Second Law, we can solve for the instantaneous velocity of the rocket if we know the forces on the rocket as a function of time. In our case, we can assume that the only forces acting on the rocket are thrust, drag, and weight, where thrust and drag act along the axis of the body. This simplifies Newton's Second Law to:

$$\sum F = ma = \frac{T-W-D}{m}$$

Incorporating a launch angle, θ, the forces are split into axial and lateral directions:

$$F_{net,\,axial} = T - W\cos\theta - D$$

$$F_{net,\,lateral} = W\sin\theta$$

where drag is assumed to be negligible in the lateral direction. It is important to note that drag acts opposite to the direction of motion, so post apogee, the force equation will be:

$$F_{net,\,axial} = D - W\cos\theta$$

Thrust, T, is given by the thrust curve of our motor, Aerotech F50-T6, and can be found on Thrustcurve.org.

Drag is given by the equation:

$$D = \frac{1}{2}\rho V^2 C_D A$$

where A is the cross-sectional area of the rocket. Using the CAD, a projected area of $0.03028 ft^2$ was found. A drag coefficient, C_D, for the rocket body of 1.00 was found experimentally through wind tunnel testing (discussed in Section 6). Similarly, a drag coefficient of 1.55 for our parachute was found experimentally through a drop test (discussed in Section 6).

Lastly, to calculate weight as a function of time, we need to calculate the mass flow rate of the propellant, $\overline{m}$.

$$\overline{m} = \frac{T_{avg}}{Isp_{avg} \star g} = \frac{w_{prop} T_{avg}}{I_{tot} g} = \frac{m_{prop} T_{avg}}{I_{tot}}$$

Using the mass flow rate we can calculate the instantaneous mass of the rocket, m.

$$m = m_o - \overline{m} t$$

Where t is the time that has passed since ignition. Once the burn time, t_{burn}, has been reached, the mass is simply the dry weight of the system. So the weight force on the rocket is given by:

$$W = (m_o - \overline{m} t) g$$

if $t \leq t_{burn}$ and

$$W = (m_o - m_{prop}) g$$

if $t > t_{burn}$.

Using Euler's method we can analytically solve for the velocity and position as a function of time:

$$v_2 = v_1 + a_1 \Delta t$$

$$z_2 = z_1 + v_1 \Delta t$$

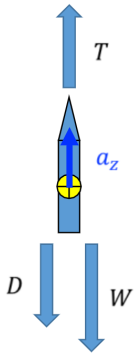

Figure 3.1: Forces on Rocket while Gaining Altitude with no Deflection Angle

3.2.2. Analytical Results

Plugging the theory in the section above into an iterative Matlab script (Appendix 11.1), we obtain an approximate trajectory for our rocket, neglecting the effects of wind. Using the CAD projected area with a launch angle of 10 degrees, our rocket is expected to achieve an apogee of 1050ft and 9.79ft/s landing velocity. The graph of this simulation is shown in Figure 3.2. A much less accurate simulation using only the projected area of our rocket body (projected area of the nose cone), our rocket achieves an apogee of 1480ft, shown in Figure 3.3.

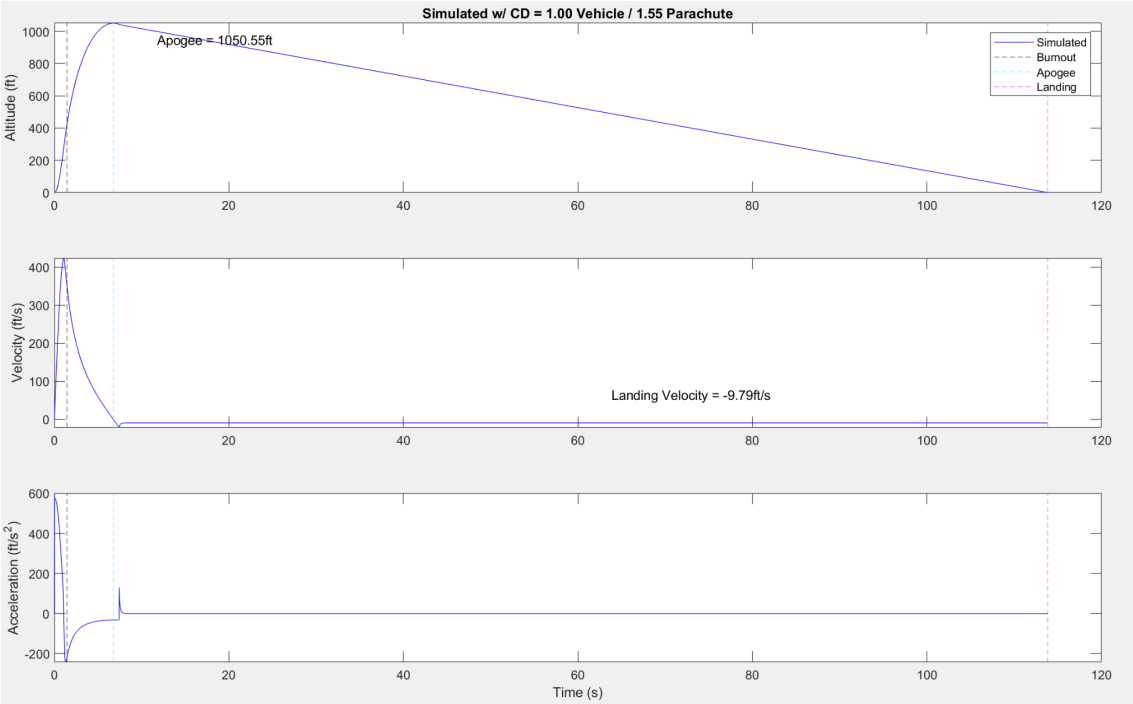

Figure 3.2: Trajectory Analysis with CAD Projected Area ($0.03028 ft^2$)

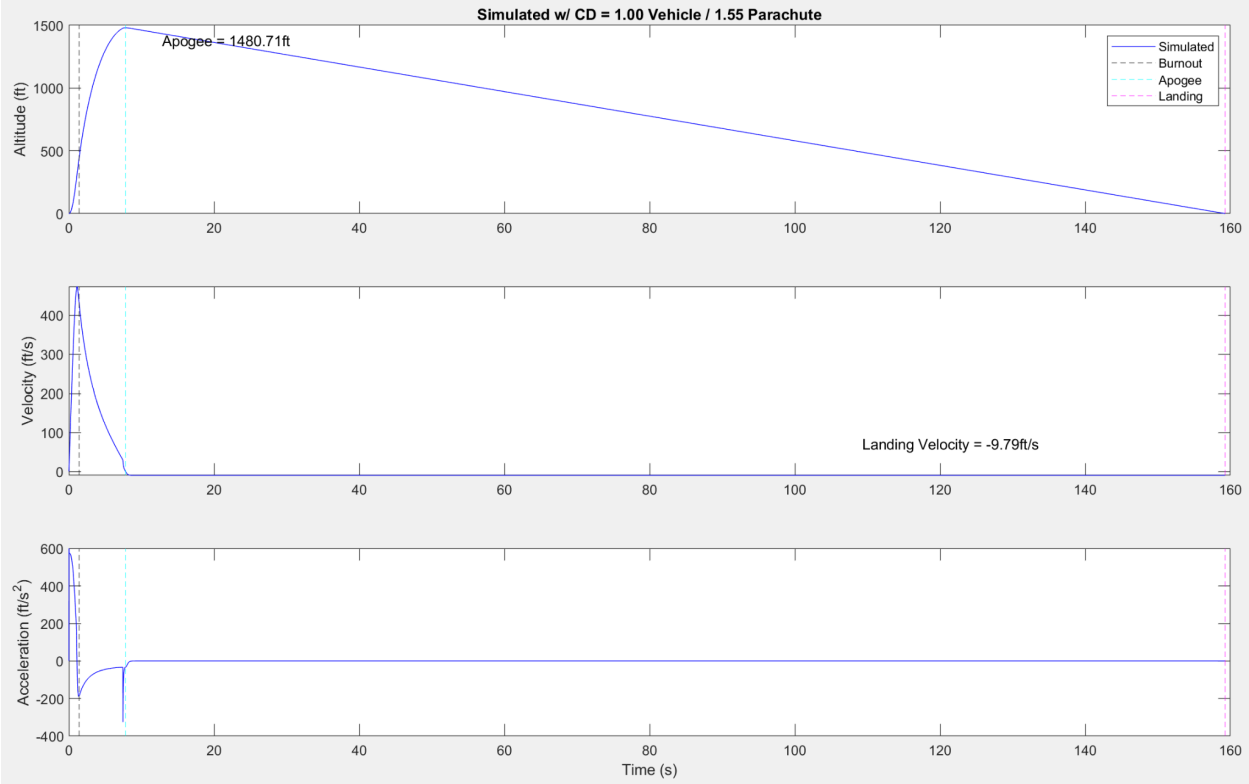

Figure 3.3: Trajectory Analysis with Nose Cone Projected Area ($0.0167 ft^2$)

3.3. CFD Analysis

3.3.1. 2D Nose Cone Analysis

Ansys Fluent was used to simulate a 2D Nose Cone Analysis to determine the max drag force on the nose cone of the rocket. The shape of the model was derived from Solidworks and was built using the equation of the nose cone, which is shown below:

$$y = 0.875\sqrt{\frac{1}{\pi} * (cos^{-1}(1 - 2(\frac{x}{5.25})) - sin(2 * cos^{-1}(1 - 2(\frac{x}{5.25}))))}$$

This equation was sketched and imported into Fluent to create half of a 2D cross-sectional representation of the nose cone. Because the nosecone is axisymmetric, the software was used to approximate the results of a three-dimensional figure from the 2D sketch. The simulation was conducted at the approximated max velocity of 115 m/s. The resulting drag coefficient was:

$$C_D = 0.318$$

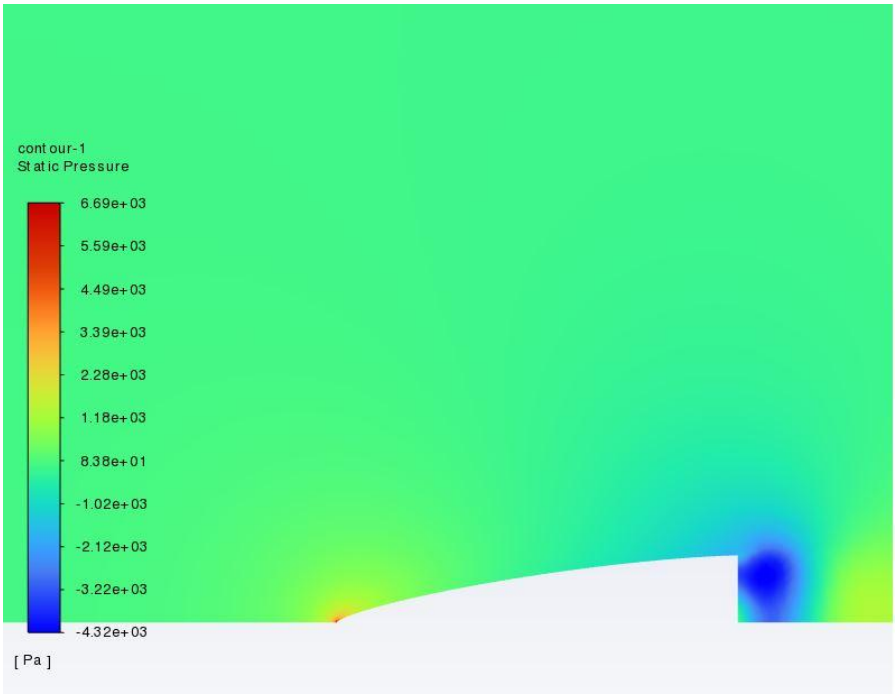

Table 3.4: Static Pressure Plot of 2D Nose Cone

Both the coefficient of drag as well as the static pressure plot are in line with the expected results. A common value for the coefficient of drag of a parabolic nose cone lies around 0.29. From this, it can be concluded that the simulation is fairly realistic. Similarly, the pressure plot shows a high-pressure point at the tip of the nose cone and a large low-pressure zone at the back end of the nose cone where the geometry suddenly falls off. The low-pressure zone may attribute slightly more drag to the nose cone than in reality since that area will be replaced with the body tube.

3.3.2. 3D Fin Analysis

COMSOL was used to approximate the coefficient of drag and lift produced by our fins at different deflection angles and velocities. A 3D model using an extremely fine mesh was selected over a 2D model for increased accuracy. The 3D model accounts for the deflection of the fin as well as the thickness. The fin was placed coincident with the bottom of the flow field because it will be attached to the rocket body tube, meaning there will be no flow in that area.

The results of the simulation are shown in Table 3.1. The results indicate that a single one of our fins will produce a CD of 0.24 with no deflection and 0.265 with 5 degrees of deflection. The increase in CD when the fin is deflected agrees with our hypothesis that as the fins deflect, they will slow the rocket down due to the increased surface area that the flow sees. Unfortunately, other experimental results in fin drag coefficients could not be found to compare our results to. Using a simple approximation that our total rocket drag coefficient will equal the sum of the fin and nose cone drag coefficients:

$$C_{D,rocket} = 3 \star C_{D,fin} + C_{D,nose\ cone} = 1.038$$

This is close to our experimental rocket drag coefficient of 1.00 found through wind tunnel testing and discussed in section 6 of this report. This is a very crude approximation that works under the assumption that drag coefficients can be summed together when parts are added together. In reality, the addition of parts leads to interference drag, which could be positive or negative (Cannon, 2004). To account for this, there are empirical formulas (found in Brent Cannon's Report) that utilize the skin friction coefficient and wetted area to approximate the drag interference between two parts, but that is beyond the scope of our project.

The lift coefficients, on the other hand, are not as accurate. As expected, the fins produce zero lift when not deflected because they are symmetric (ideally), but when the fins are deflected 5 degrees, they only produce a coefficient of lift of 0.004. This value is far too low and may be due to user error or a bad mesh. In the future, it would be beneficial to perform these simulations for many different fin angles and compare the results to wind tunnel testing on just the fins. A full table of lift coefficients at various fin angles and velocities can be used to create a fit equation for the active control, which allows for a mapping of the fin deflection angle to generate lift. This, however, was not utilized due to the inaccuracies in the CFD.

Velocity (m/s)	Fin Angle (degrees)	Drag Coefficient	Lift Coefficient
50	0	.241	0
50	5	0.267	0.004
100	0	.24	0
100	5	0.266	0.003
115	0	0.24	0
115	5	0.26	0.0038

Table 3.1: 3D Fin CFD Analysis Results for CD and CL

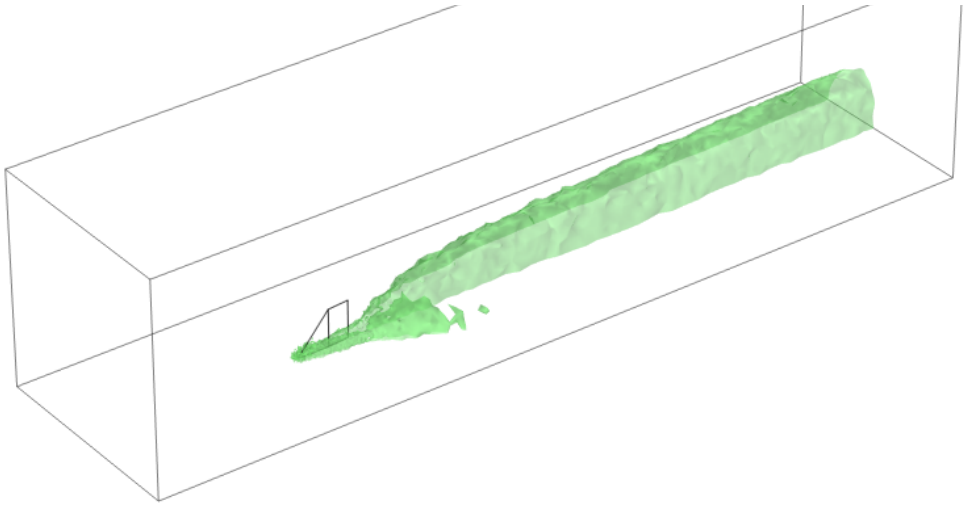

Figure 3.5: 3D Fin Isosurface Plot with Trailing Vorticities

3.3.3. 3D Rocket Analysis

We used the Ansys Fluent software to estimate the drag coefficient for our whole rocket design and to understand the overall behavior when in flight for different values of angle of attack and velocity. A 3D model with high-quality mesh was used. The rocket was divided into a total number of 5200 divisions for improved quality. The results of this simulation are shown in Tables 3.2 and 3.3.

Velocity (m/s)	Angle of Attack (degrees)	Drag Coefficient
50	0	0.45055
60	0	0.45049
70	0	0.45038
80	0	0.45031
90	0	0.45025
100	0	0.45021
110	0	0.45018
120	0	0.45015

Table 3.2: 3D CFD Results for 0° AoA

Velocity (m/s)	Angle of Attack (degrees)	Drag Coefficient
50	5	0.43964
60	5	0.43351
70	5	0.42924
80	5	0.42603
90	5	0.41872
100	5	0.41550
110	5	0.41093
120	5	0.40695

Table 3.3: 3D CFD Results for 5° AoA

These results indicate that the rocket will produce a total drag coefficient of 0.45 on average for 0° angle of attack and a drag coefficient of 0.42 on average for 5° angle of attack. For the same

value of the angle of attack, the drag coefficient decreases as the speed is increasing because the rocket experiences increasing dynamic pressure that compresses the air in front, leading to a decrease in the relative density of the air. This behavior was expected. Moreover, for these small angles of attack, the drag coefficient remains fairly constant, with bigger differences occurring at higher angles of attack.

However, as we tried to perform the 3D simulations for higher angles of attack, the results were inconsistent with the theoretical predictions because the resultant drag coefficient was significantly lower than the predicted one. More exactly, we can easily observe that the 3D drag coefficient is fairly low compared to the 3D fin drag coefficient or compared to the testing one. This might occur due to inaccurate mesh sizing or due to model infidelities, such as not having an accurate representation of the real rocket due to its geometric complexity. Another factor that might have an effect on this discrepancy is the imperfect manufacturing process of the rocket. Thus, surface roughness and imperfection might create flow turbulence that is not taken into account when performing CFD simulations.

3.4. FEA Analysis

Finite element analysis was performed on multiple components of the rocket as well as a simplified assembly of the completed rocket. Structural analysis was performed on these components as they were deemed vital to flight as well as had to be manufactured by our group. Therefore, finite element analysis was used to ensure that the components would hold up during flight. The major components in question are the nose cone, fins, and body tube.

Starting with the nose cone, the coefficient of drag was pulled from the 2D Nose Cone CFD analysis. From the coefficient of drag, a maximum drag force could be extrapolated using the following equation:

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

Using the max velocity of 115 m/s along with the density of air at sea level and the maximum cross-sectional area of the nose cone, a drag force of 33 N was calculated. In performing finite element analysis, the drag force of 33 N was imparted axially on the nose cone while the

shoulder was held fixed to simulate the compression that the nose cone would face. Looking at the stress plot, the maximum Von Mises stress achieved in the simulation was multiple magnitudes of order lower than the yield stress that the nose cone could withstand; therefore, the part was not expected to fail. The results are shown in the figure below:

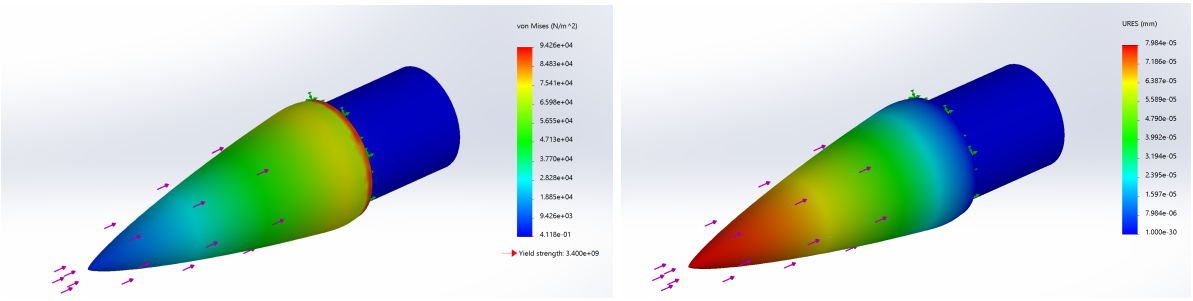

Figure 3.6: Nose cone stress plot (left). Nose cone displacement plot (right)

Moving on to the structural analysis for the fins, the analysis was performed to discover if there was possible bending in the PVC portion of the fin as it was deemed issues could arise here. On top of that, for active stabilization to work properly, the fins would need to hold up and perform as they were expected to. From the 3D Fin CFD analysis, a drag force of 0.27 N was found to be imparted on the nose-facing surface of the fin assembly. This force was rounded up to 0.3 N to account for margin and further anomalies. The 3D printed portion of the fin was held fixed at the bottom as well as the PVC portion in the fin well of the 3D printed part was held fixed to account for the epoxy connection. The maximum deflection was found at the location in red on the deflection plot shown below. The value of maximum deflection is 0.69x10^-3 mm which is lower than predicted. This deflection is practically negligible and would not affect flight in the passive regime where the total fin deflection was 0°.

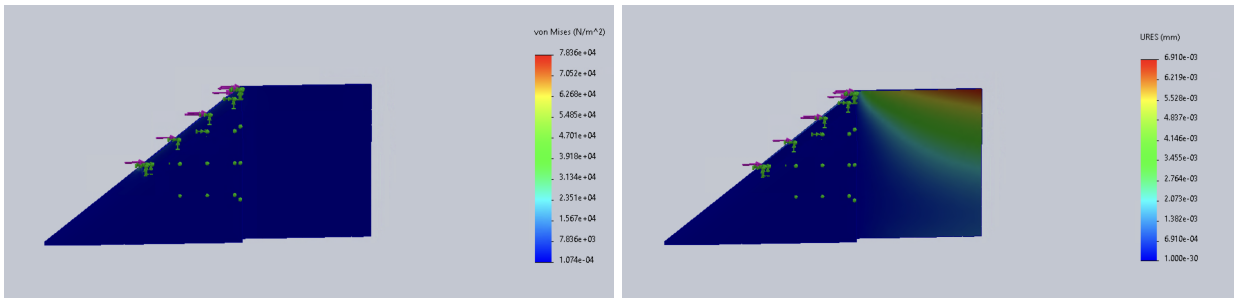

Figure 3.7: Fin stress plot (left), Fin displacement plot (right)

Finally, structural analysis was performed on the basic assembly of the rocket consisting of the nose cone, fins, and body tube, which are the three most essential parts that comprise the body. The same maximum force of 33 N has been applied axially to the nose cone. Using the given statistics of the motor, a force of 79.6 N was applied axially to the bottom of the body to simulate the maximum thrust achieved by the motor. From these two forces, a compression is created along the rocket. The body tube was held fixed as a reference frame. From the stress plot shown below, it can be seen that the highest stress will be located at the shoulder between the nose cone and body tube, as well as at the bottom of the body tube where the thrust will be applied. Looking at the plot of the factor of safety, it can be seen that the minimum factor of safety is well above a conservative threshold of 4, and, therefore, none of the components should be in danger of failing due to structural forces.

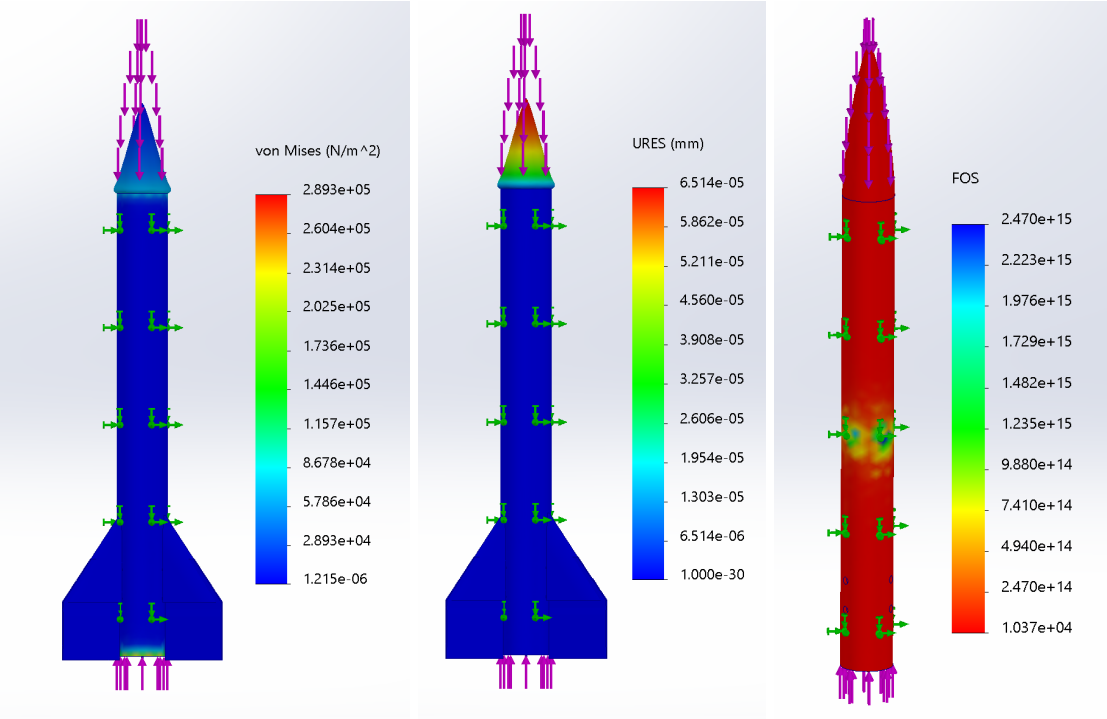

Figure 3.8: Body stress plot (left). Body displacement plot (center). Body FOS plot (right)

4. Active Controls

The objective of our BFR rocket is to achieve active control over the trajectory of our launch. Once the motor burn is complete, the fins at the tail of the rocket actuate to provide attitude control in all three axes (roll, pitch, and yaw). This allows for complete control of our rocket orientation, which we utilize to maintain a straight-up trajectory on launch, correcting for any wind perturbations or aerodynamic imperfections on our rocket surface and fin alignments.

4.1. Controls Paradigm

This section discusses the overall controls paradigm for active control: estimation, control, and actuation. Figure 4.1 shows the block diagram schematic for the process. The sections below discuss the components in greater detail.

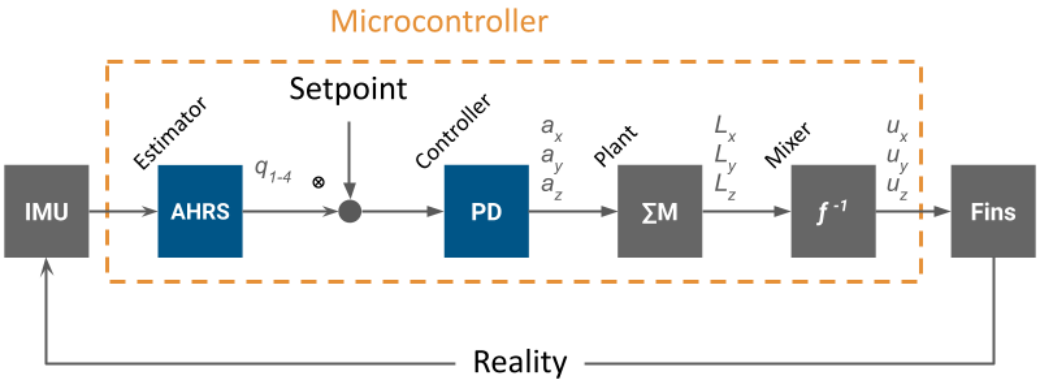

Figure 4.1. Active Controls Block Diagram.

4.1.1. IMU and AHRS Estimator

The attitude of the rocket is determined using sensor data from the Pololu AltIMU-10 v5 IMU module, which provides accelerometer, gyroscope (angular velocity), and magnetometer measurements in all three axes. These measurements are fused using a Madgwick AHRS (attitude and heading reference system) filter, implemented in the SensorFusion library, and outputs the attitude as a quaternion. From the use of the magnetometer, the identity quaternion ($1 + 0\hat{\imath} + 0\hat{\jmath} + 0\hat{k}$) coincides with the axes of the NED (North-East-Down) coordinate frame.

Due to the noisiness of measurements, both the accelerometer and gyroscope measurements are filtered through a moving average. In addition, the gyroscope suffers from drift that induces a steady state offset in the gyroscope measurements. In an effort to reduce this, the steady-state gyroscope measurement is determined when the rocket is stationary and upright on the pad and subtracted. In implementation, however, the gyroscopic drift is not constant, and thus the drift accumulates over time. Hence, the magnetometer was required in the AHRS filter to correct for this drift.

4.1.2. PD Controller

The error, which would be the rotation quaternion from the attitude to the setpoint quaternions, is utilized in a quaternion-based PD controller, a basic framework kindly provided by Professor Lopez. The setpoint quaternion is determined when the rocket is stationary and upright on the launch pad. The error rotation quaternion is calculated using the Hamilton product as follows:

$$\Delta q_{err} = q_{sp} q_{att}^{-1}$$

where Δq_{err} is the error rotation, q_{sp} is the setpoint quaternion, and q_{att}^{-1} is the reciprocal of the attitude quaternion determined from the AHRS. In the case of unit quaternion, the reciprocal is the same as the conjugate, $q^{-1} = q^{*}$ which simplifies computation.

The PD (proportional-derivative) controller output is governed by:

$$a = \; - \; \sigma K_P \Delta q_{err} - K_D \omega$$

where a is the output of the controller, in angular acceleration moments, σ is either 1 or -1 to address unwinding in the Δq_{err} rotation quaternion, K_P is the proportional gain, K_D is the derivative gain, and ω is the angular velocity, which is obtained from the gyroscope measurement. a has the angular acceleration components for each axial direction.

4.1.3. Plant Dynamics

With the desired angular accelerations, this can be converted to the necessary lift that each of the three fins on the BFR rocket needs to produce. For the discussion in this and the following section, we will utilize a body coordinate frame that aligns the z-axis longitudinally with the

rocket, which is *not* the same coordinate frame as NED. Roll, pitch, and yaw correspond to rotations about ϕ, ψ, and θ, respectively. The lifts determined here in the plant dynamics simply consider lifting in the x, y, and θ cylindrical directions, as opposed to each individual fin.

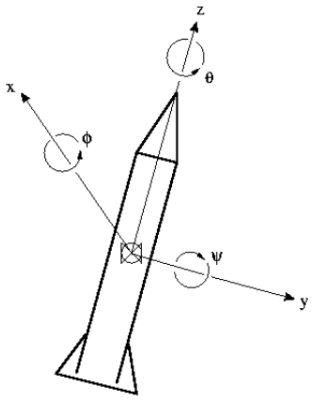

Figure 4.2. Rocket Body Frame Coordinates.

The roll and pitch components can be translated from angular acceleration to fin lift by considering a simple moment arm model by neglecting the drag force. The roll component is modeled cylindrically, assuming that all three fins contribute equal amounts of lift.

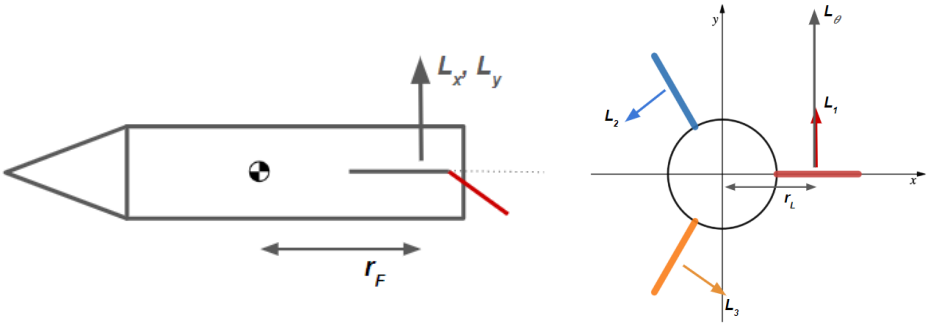

Figure 4.3. Moment Diagrams. Roll and Pitch on left, Roll on right.

Thus, the summation of moments:

$$\Sigma M_x = I_x a_x = r_F L_x,\ \Sigma M_y = I_y a_y = r_F L_y, \text{ and } \Sigma M_z = I_z a_z = r_L L_\theta$$

where a is the desired angular acceleration output from the PD controller. Isolate for lifts:

$$L_x = \frac{I_x a_x}{r_F},\ L_y = \frac{I_y a_y}{r_F}, \text{ and } L_\theta = \frac{I_z a_z}{r_L}$$

We now have a desired lift that is required to command the rocket to the desired setpoint.

4.1.4. Mixer, Allocation, and Fins

The mixer converts the lift in the body frame coordinates to the 3 fins on the rocket, as well as converts lift to the output signal that will be sent to the servo. The lifts in the body frame need to be converted into lifts of each fin, as shown below:

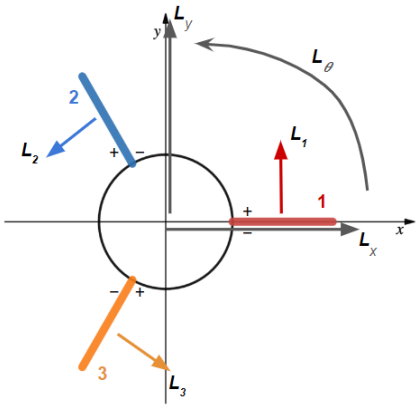

Figure 4.4. Fin Lifts Diagram.

This mapping is achieved through simple inverse kinematics, where:

$$\begin{bmatrix} L_x \\ L_y \\ L_\theta \end{bmatrix} = \begin{bmatrix} \cos(0+\frac{\pi}{2}) & \cos(\frac{2\pi}{3}+\frac{\pi}{2}) & \cos(\frac{4\pi}{3}+\frac{\pi}{2}) \\ \sin(0+\frac{\pi}{2}) & \sin(\frac{2\pi}{3}+\frac{\pi}{2}) & \sin(\frac{4\pi}{3}+\frac{\pi}{2}) \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix}$$

and by inverting to obtain the allocation matrix for L_1, L_2, and L_3:

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 0 & 2 & 1 \\ -\sqrt{3} & -1 & 1 \\ \sqrt{3} & -1 & 1 \end{bmatrix} \begin{bmatrix} L_x \\ L_y \\ L_\theta \end{bmatrix}$$

The lift is then converted to a servo input through a fit function. This fit is ideally determined using CFD and/or experimental testing of fin deflection and lift obtained at various forward velocities V, $L_{fin} = f(u, V)$. We expect that the lift should scale quadratically according to the lift equation. This fit would model only the incremental lift from the fin deflection, assuming the lift of the body and passive portion of the fins remained constant. This model is shown in the diagram below:

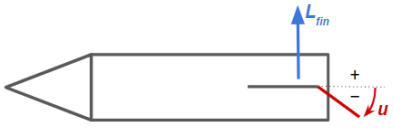

Figure 4.5. Deflection and Lift Diagram.

Thus, by inverting this lift equation with a measured rocket forward velocity, the function to obtain servo inputs can be obtained for each fin, $u = f^{-1}(L_{fin}, V)|_V$, for u_1, u_2, and u_3.

4.1.5. Implementation Changes

Implementation wise, portions of the aforementioned control blocks were not implemented due to time and testing constraints. Particularly, the plant dynamics and the inverse fin lift fitting components were not implemented as we lacked complete knowledge of the rocket and its aerodynamic properties. Determining the rocket moment of inertia and the complete lift profile would have been very time intensive for very little marginal benefit, as the rocket would only be under active control for roughly 6 seconds. Gains were instead chosen to be small and underdamped in actuation to prevent over-correcting and causing instability.

4.2. State Machine Design

The BFR rocket onboard flight software operates in a state machine to ensure proper progression of states and sensor readings and to prevent the fins from actuating prior to motor burnout. The rocket utilized four states: IDLE, PAD, LAUNCH, and FLIGHT. Regardless of state, the data is continuously logged at 10 Hz. The flowchart for the state progression is shown below:

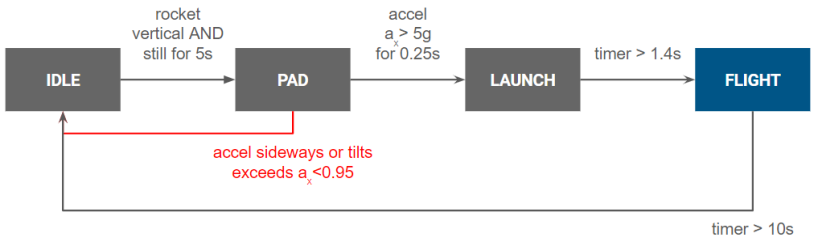

Figure 4.6. State Machine Diagram.

In the IDLE state, only the AHRS estimator is active while the fins remain stationary. When the rocket is placed upright (vertical) and remains still for 5 seconds, it is switched to PAD state for when the rocket is on the launch pad.

In the PAD state, the steady state offsets are measured and removed from the gyro measurements, and the current attitude for which the rocket is stationary is saved as the attitude setpoint. When the rocket experiences greater than 5G's in the vertical direction for 2.5 consecutive seconds, it is switched to the LAUNCH state for the launch burn portion of the flight. Otherwise, if the rocket experiences acceleration sideways or tilts that exceed the vertical acceleration $a_x < 0.95$ G's or exceeds the acceptable normalized acceleration $0.95 < a_{norm} < 1.05$, the failsafe is triggered, and the rocket returns to the IDLE state.

In the LAUNCH state, only the AHRS estimator is active with the steady state offsets (determined from the PAD state) removed. Fins remain stationary. After 1.4 seconds, the rocket transitions to a FLIGHT state.

In the FLIGHT state, the active control paradigm is functional, and fins are allowed to actuate. After 10 seconds, the rocket transitions back to the IDLE state.

The complete flight software code implementing both the state machine and the controls paradigm is included in Appendix 11.3.

5. Manufacturing

This section provides a comprehensive overview of the manufacturing processes and operations employed for building our rocket. Our primary focus was on efficiently transforming the materials used into the finished components that form our rocket while maintaining high quality. This section will delve into the various stages of our manufacturing processes, encompassing everything from production planning to equipment utilization and quality control measure.

5.1. Planning

We began with a detailed analysis of the rocket design and specifications, which served as the foundation for creating a comprehensive manufacturing plan. The plan shown in Figure 5.1 outlines the sequence of activities, resource allocation, and timelines required to transform the design into a fully functional rocket. This schedule encompasses crucial steps such as material procurement, component fabrication, subsystem integration, and testing procedures for quality assurance. Each step was carefully scheduled to ensure a smooth workflow and minimize potential delays. Adhering to this manufacturing plan was vital to ensure the efficient and timely completion of each stage, ultimately leading to a successful rocket launch.

Week	Start	End	Tasks	Location	Material	Part Quantity
3	17-April	21-April	Motor Selection	Lab IV-114	Vender	1
			3D-Print Nose Cone Cast	MakerSpace	PLA/3D Printed	1
4	24-April	28-April	Nose Cone	Lab IV-114	Fiberglass	1
			Aerodynamic Stability	Computational		
			Aerodynamic Drag	Computational		
			Aerodynamics CFD	Computational		
			PLA Fins	MakerSpace	PLA/3D Printed	3
5	1-May	5-May	Iterate Design, Analysis, and Build Plan			
			Bulkheads	WaterJet/Makerspace	Wood	2
			Test Servos	Lab IV-114	Vender	4
6	7-May	12-May	Preliminary Stability Check	Wind Tunnel		
			Pressure & Altitude	Computational		
			Order Body Tude	Carbon Fiber		
7	14-May	19-May	Machining	Machine Shop		
			Start Rocket Build	Lab IV-114		
8	21-May	26-May	reTest	Lab IV-114		
9	28-May	2-June	Final Assembly	Lab IV-114		
			Final Stability Check	Wind Tunnel		
10	5-June	9-June	Packaging	Lab IV-114		

Figure 5.1: Manufacturing Plan

5.2. Body Tube

The body tube was manufactured during week six, as outlined in the planning chart. The process began with the selection of a release film, carbon fiber sheets, perforated release film, peel ply, a breather, and a vacuum bag, which were then cut into precise dimensions. These sheets were carefully layered onto a custom-built mandrel, following the enumerated order to achieve optimal strength and rigidity. The vacuumed assembly shown in Figure 5.2 was then placed in an oven, where heat and pressure were applied to create a solid carbon fiber composite. Once cured, the tube was carefully removed from the mandrel, and excess material was trimmed and sanded to achieve the desired shape and finish. The final product is a three-layer carbon fiber body tube with a diameter of 1.75 in. and a length of 18 in.

Figure 5.2: Body Tube Manufacturing Assembly

5.3. Fins

Our design requires three fins for optimal stability and maneuverability. We proceeded by 3D printing the fins from high-quality PLA. We chose this material because of its durability, low weight, and ease of printing. These fins were printed with two pegs each that snap into the carbon fiber body tube for easy fixation. Since we are using active control, we decided to use PVC piano hinges as our control surfaces. Thus, we cut the piano hinge into the dimensions

specified in our design and fixed them with epoxy into the 3D-printed fins. These hinges will allow for angle of attack changes and ease of maneuverability. Once the epoxy had dried, we sanded everything down to the needed measurement. The fins were then snapped and glued to the carbon fiber body tube. In Figure 5.3, the resulting fins are pictured, while in Figure 5.4, the final fin and body tube assembly is shown.

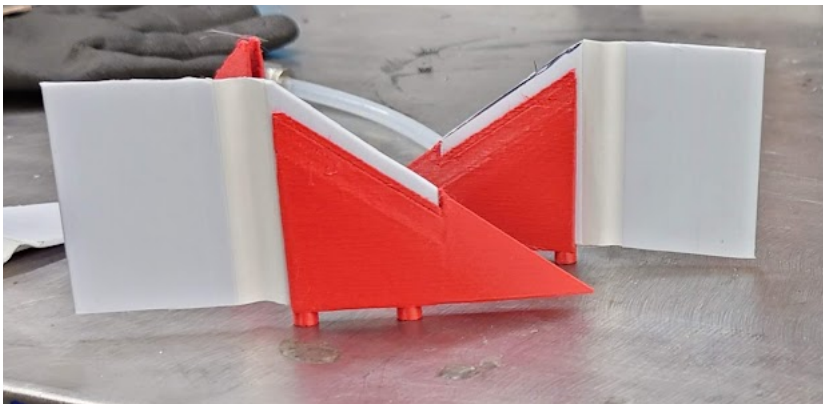

Figure 5.3: Manufactured Fins

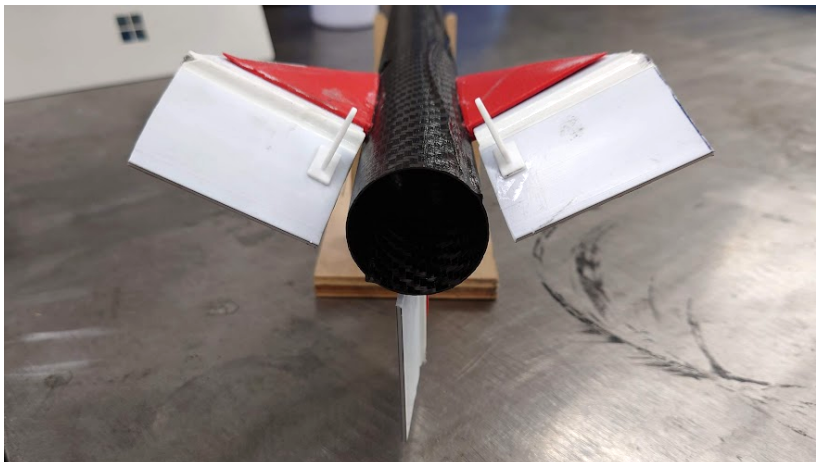

Figure 5.4: Fins Assembled on Rocket

5.4. Control Mechanism

Each fin was independently driven by a 9g servo motor, each of which can output a torque of 1.6 kg cm. The size constraint of the 1.5” diameter body tube meant the servos could not be arranged next to each other. Instead, they were arranged in a vertical row, each facing outwards in a

different direction towards the respective fin. They were attached by gluing them to the carbon fiber wall of the body tube from the inside, with the central part of the servo poking through small holes drilled in the body tube. Each servo had an arm that connected with a fin below via two metal drive rods, as can be seen below in Figure 5.5. The piano hinges of the fins each had two arms adhered with loctite glue for these drive rods to attach to. In hindsight, this attachment mechanism may have been a large reason for launch failure: the fin arms broke off during flight, unable to withstand the lift focus generated on the piano hinges. This left the active control system unable to actually drive the fins.

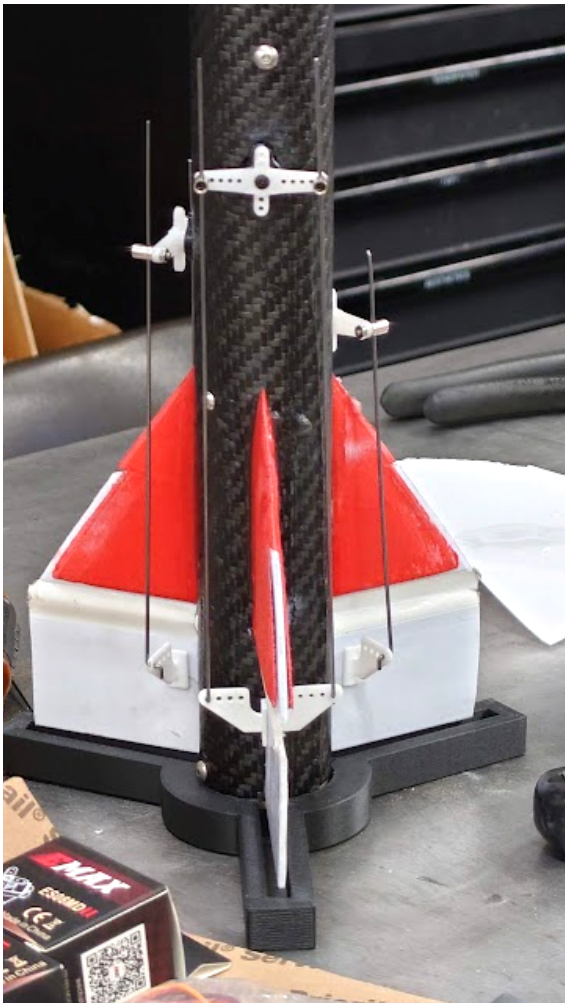

Figure 5.5: Control Mechanism

5.5. Motor

Motor was secured at the bottom of the rocket by two water-jetted wooden bulkheads. The bulkheads were manufactured larger than intended to manually sand them down and ensure a tight fit for the motor once it was implemented in the rocket. Holes were drilled into the carbon fiber body tube and the bulkhead, which were then fastened by screws to keep the motor secured during launch. The motor used was an Aerotech F50-6T, with a total impulse of 76.8 Nsec and a 6-sec delay charge that produces a maximum thrust of 79.6N.

Figure 5.6: Aerotech F50-6T Motor

5.6. Recovery System

Initially, we purchased a 12-in-diameter parachute, but after a careful review of its characteristics and deceleration capabilities, we deemed it too small for our purposes. Thus, we decided to hand-make our own parachute. This was cut from a piece of fabric existent in the lab and had a diameter of 28 inches, much larger than the initially purchased one. We also perforated a 3-in-diameter top vent in the center of the parachute in order to allow for an organized airflow out of the parachute to ensure that it inflates properly after deployment. A picture of our parachute is shown in Figure 5. 7.

In addition to the parachute, the recovery system also consisted of a fixing mechanism for the egg. Thus, we filled the nose cone with stuffing and wrapped the egg in a soft sponge to

minimize the damage produced by the landing on the egg.

Figure 5.7: Parachute

5.7. Electronics Assembly

Due to additional power requirements from the servos, which operate at 5V, voltage regulators were required to power all components properly. As each of the available 1-cell Li-Po batteries nominally operated at 3.7 volts, we utilized 2 Li-Po batteries in series to achieve 7.4V, which is stepped down to 5V to power both the servos and the Teensy microcontroller. The complete electronics schematic of the electronics assembly on the BFR rocket is shown below:

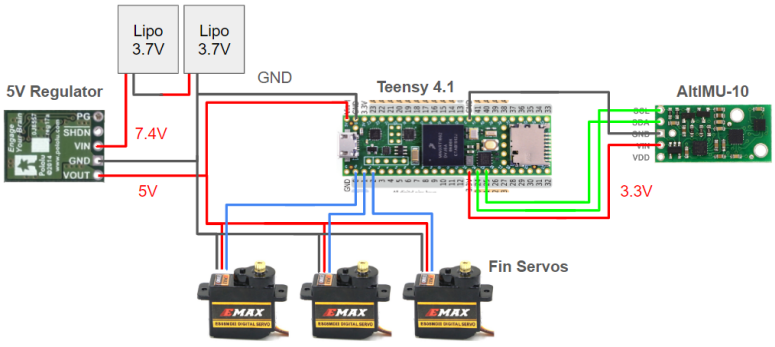

Figure 5.8: Electronics Assembly Schematic.

Wires were soldered into place, and solder joints were covered in hot glue to prevent contact with the carbon fiber body tube that would cause shorting. Further care was taken by wrapping each PCB board with electrical tape when applicable. The AltIMU, which has a pressure sensor, was not wrapped but instead had an insulating double-sided foam tape that rigidly secured it to the body tube wall.

Since the electronics assembly is housed in the detaching portion of the nose cone, the wires connecting to the servo must all be detachable to allow proper separation for the parachute to eject. Thus, jumper wires were used to connect the servos, voltage regulator, and the Teensy microcontroller.

5.8. Nose Cone

The nose cone of the rocket was the same nose cone that our group fabricated for lab 3. The nose cone is a fiberglass layup that was created using a 3d printed mold. The equation for the design of the nose cone is shown below:

$$y = 0.875\sqrt{\frac{1}{\pi} * (cos^{-1}(1 - 2(\frac{x}{5.25})) - sin(2 * cos^{-1}(1 - 2(\frac{x}{5.25}))))}$$

The nose cone is part of the family of Haack series nose cones. It was constructed by cutting out multiple cross-sections of the nose cone and using an epoxy-resin matrix to act as a binding agent. The materials were fitted to a mold and cured for 48 hours.

A few adjustments were made to the nose cone. Firstly, weight was added inside of the nose cone at the tip. While the BFR's passive stabilization was good, it was deemed important to increase it slightly. Adding mass to the front achieved that. Next, there was a quarter-inch diameter hole drilled into the nose cone. It was decided that the optimal place to fit that camera would be in the nose cone. This machined hole acted as a window for the camera to shoot video of the launch. Finally, the nose cone became the housing for the payload. Instead of a regular egg, a quail egg was used as the payload. The quail egg was padded in foam and stored in the nose cone.

5.9. Budget

Overview of the budget includes property costs and common items. Property cost pertains to the active control system that was implemented in the rocket with items such as the motor, servo motors, hinges, etc. Common items were required for this project to help build the rocket, with materials such as carbon fiber and fiberglass were used to create the body tube and nose cone. Other items were given but still needed to be implemented into the budget plan.

Property Costs				
Item	Description	Link	Quantity	Amount
Motor	F50T-6	https://www.apogeerockets.com/Rocket_	2	$84.72
Servo Motors	9g digital servo (4 pack)	https://www.amazon.com/diymore-MG90	1	$15.99
Servo Linkages	10Pcs Adjustable Pushrod Connector	https://www.amazon.com/Adjustable-Co	1	$11.99
Hinges	Plastic Piano Hinge without Holes White	https://www.mcmaster.com/11195A46/	1	$18.83
Common Items				
Carbon Fiber	Surface Area: 87.7 in^2	https://www.mcmaster.com/8181K14/	1	$60.25
Fiberglass	Surface Area: 17 in^2	https://www.mcmaster.com/products/fibe	1	$27.89
Parachute	12" Elliptical Parachute	https://the-rocketman.com/chutes-html/	1	$5.78
Microcontroller	Teensy 4.1		1	
Altimeter/IMU	Pololu AltIMU		1	
Camera	Small Spy Camera		1	
Battery	Lipo		1	
			Total	**$225.45**
			Total w/o CF/FG	**$131.53**

Figure 5.9: Budget Chart

6. Testing

6.1. Recovery System

We decided to test the decelerating capabilities of our parachute by performing a drop test. We chose the bridge between Engr. IV and Engr. VI buildings as our testing place because it provides sufficient clearance for the parachute's descent. In order to not destroy our only rocket, we used a water bottle filled with enough water to replicate the actual weight of our rocket, 0.82 lbs. We also added the rocket's sensors to this water bottle in order to obtain acceleration and pressure results. Right before dropping the assembly of the bridge, we activated the sensors. Based on this test, we estimated that if the parachute deploys fully, the rocket will have a terminal velocity between 3-5 m/s and a terminal coefficient of drag between 1 and 1.5. A picture from the drop test is attached below.

Figure 6.1. Recovery System Testing

6.2. Wind Tunnel

Wind tunnel testing was used to test the drag, stability, and active controls of our rocket. As illustrated in Figure 6.3, the rocket's center of gravity was mounted to a vertical rod that freely

rotates. A turbine at the end of the wind tunnel was activated at various speeds to achieve uniform flow over the rocket.

Figure 6.2. Wind Tunnel

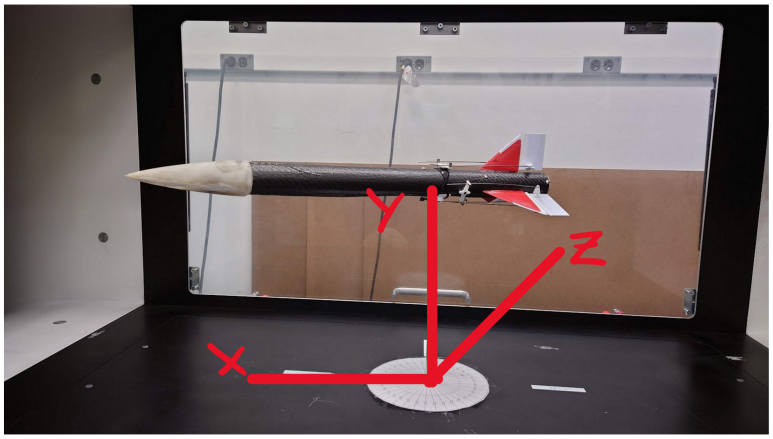

Figure 6.3. Wind Tunnel Setup

6.2.1. Coefficient of Drag

The drag force experienced by the rocket was measured using a load cell under the mounting rod. Results from our final testing are presented in Figure 6._, where velocity is the speed of the flow, Fx is the drag force (to the left in Figure 6._), and Fy and Fz are forces in their respective directions. Assuming the density of air in the wind tunnel is $1.204\,kg/m^3$, the rocket's coefficient of drag can be found with the following formula:

$$C_D = \frac{2D}{\rho V^2 A}$$

The resulting drag coefficients are highlighted in Figure 6._ below. Based on the data, our drag coefficient is nearly unity. The average drag coefficient for a model rocket is typically 0.75. The

fins, servos, fin rods, and launch lugs are key factors that contributed to our high drag coefficient.

Constants				Velocity (m/s)	Fx	Fy	Fz	CD
rho	1.204	kg/m^3		0	0	0	0	0
mu	1.83E-05	kg/m/s		14.771	0.365	-0.017	-0.059	0.988
Area	0.002812898	m^2		14.946	0.362	-0.016	-0.057	0.957
	(projected area from CAD)			25.025	1.062	-0.004	-0.113	1.002
				25.104	1.068	-0.005	-0.114	1.000
				25.013	1.066	-0.003	-0.116	1.006
				25.092	1.067	-0.004	-0.113	1.001
				25.021	1.065	-0.004	-0.112	1.005
CD = 2*Fx/(rho*V^2*Area)				25.099	1.068	-0.003	-0.114	1.001
				25.040	1.064	-0.003	-0.111	1.002
				30.040	1.637	0.240	-0.026	1.071
				29.860	1.621	0.240	-0.039	1.074
				29.692	1.519	-0.213	0.043	1.017

Figure 6.4. Coefficient of Drag Calculations

6.2.2. Passive Stability

Passive stability was the first test done in the wind tunnel to ensure that our rocket was stable and ready for launch. The rocket was tested in the wind tunnel at speeds of 15, 20, 30, and 40 m/s. The lowest velocity is most critical for stability due to the least amount of corrective aerodynamic force. We corroborate this phenomenon with greater stability at higher speeds. Overall results were satisfactory as the rocket did pass all speed tests in the wind tunnel.

Another passive stability test was implemented where more weight was added to the rocket, which moved the center of gravity. 0.8 grams were added to the rocket to account for things that were not added in our initial test, such as the weight of the egg and camera. Similar trends occurred in this test, one different observation is that our rocket became more stable with the added weight that was implemented. In conclusion, both passive stability tests passed, and it allowed for more weight to be added into the rocket if needed leading into launch. Links to video documentation of these tests can be found in Appendix 11.3.

6.2.3. Active Stability

Active stability was the final test conducted in the wind tunnel to validate the feasibility of our controls paradigm. The only control in one axis, roll, was able to be tested as a constraint of the

testing apparatus. A custom testing state machine and code were utilized to bypass the state transition logic.

The stability was tested at a freestream velocity of 30 m/s. In response to perturbations, the rocket remains stable. It does return to the setpoint faster but has more prominent oscillations about the setpoint. It is likely that the phenomenon is due to the 3 fin configuration that causes coupled dynamics in the other axes, where roll and pitch corrections induce moments in the other. The underdamped oscillations are also likely due to the low tuning gains as well as phase delay in the controls paradigm from the AHRS measurements, where the accelerometer and gyroscope measurements are filtered on a moving average. Links to video documentation of these tests can be found in Appendix 11.3.

7. Launch Synopsys

7.1. Analysis vs. Reality

Initially, at launch, the rocket did clear the rail and reached an apogee of 500 ft. Coming down, the parachute did not deploy as there was a large impact on the ground. When analyzing the rocket, the servo motor linkages came off the fins. It is undetermined what exactly occurred, but there are two possibilities, the servo motor linkages could have come off during launch because the force of the motor was too powerful, or they became undone upon impact. The high lift forces generated on the piano hinges during launch tore the arms off of the piano hinges. That led to fins that would have wildly flapped in the windstream and very strongly reduced the stability margin (likely less than 1 if the upper PLA portions of the fins were the only parts providing significant stability damping force). This would have resulted in a very unstable rocket that spiraled on its ascent - which is what was observed during flight. The rocket was especially unable to control itself post-burn as the active control mechanism switched on, but it could not actually move the fins without the push rods being attached. This led to the nosecone coming off prematurely long before the ejection charge went off because the rocket, at some portions of the flight, was traveling perpendicular to the windstream, and the coupler attachment method holding the nose cone in place was not strong enough to resist the high aerodynamic forces here. Thus the parachute was deployed long before the planned apogee, but because the unstable rocket was not actually traveling that fast and there was no gas pressure from an ejection charge, the parachute did not unfurl and did not have sufficient time to do so during the rocket's very quick descent back to the ground.

Internally most of the electronics and servo motors were intact, the only thing corrupted was the camera and the internal measuring unit data. Lastly, the overall structure of the rocket was intact, as there was no damage to the body tube, nosecone, or fins. Unfortunately, this was not the result we wanted, as the active control was unsuccessful during the launch, but there is a lot to learn from our launch.

As shown throughout this report, the results are not what we predicted from the apogee to the active control. Although we did do testing, the lab was a more controlled environment to predict

what was going to occur at launch. There is always room for improvement as there could have been further analysis done on the rocket, but there were time constraints restricting any further testing. Overall, the results done prior were satisfactory.

Figure 7.1: Left: Pre-Launch Setup; Right: Post-Launch Rocket

8. Future Plans and Improvements

While the rocket managed to launch and functioned well until apogee, there are multiple adjustments that could be made if we were to do this project over again. Firstly, we would have liked to expand our timeline. Given that the BFR implemented active stabilization for post-burnout control, more complete wind tunnel testing would have helped to define the stabilization code parameters.

A major adjustment that we would have made to our rocket is to increase the diameter of the body tube. The current diameter of the body tube is 1.75 in; however, a body tube with a diameter between 3 to 4 in would have been ideal. Having a small body tube posed various issues that were not accounted for in the beginning. One of these limitations was that a normal-sized egg did not fit into the body tube. While the egg was housed in the nose cone, the size of the nose cone was directly correlated to the diameter of the body tube. Increasing the diameter of the body tube would have allowed a regular egg to fit into either the body tube or nose cone instead of having to resort to placing a quail egg in the nose cone. Another issue that arose from having a small diameter tube was the lack of space. Doing active control, the body tube had to house 3 servo motors, extra wiring, extra electronic components, a camera, the parachute with shock cords, the motor, and the payload. Trying to stuff all of these parts into the body tube caused issues in testing and setting up the rocket. The electronics had to be housed pretty close to the motor and had created complications when the ejection charge went off. Furthermore, the lack of parachute deployment can be attributed to the small body tube diameter. In order to fit it properly, the parachute had to be tightly wrapped in its own chords. While the parachute deployed, it did not unfurl since it was too tightly wrapped. On the topic of parachute deployment, we also feel that while a 28-in-diameter parachute sufficed, a larger parachute might have been safer for the increased amount of avionic components that we had.

Another interesting idea that could be explored would be to have the stabilization system all be housed within the body tube. There was various clutter on the outside of the rocket that contributed to parasite drag and affected the apogee of the rocket. The parts that stuck out were the servo motors, the rods, the connecting components for the rods, and the launch lugs. This

disrupted the flow around the body tube and increased drag. Housing the stabilization system in the body tube would be complicated and an interesting challenge to take on. Regarding the launch lugs, fly-away launch lugs could be explored as well.

For this project, we were not allowed to explore active stabilization during burnout due to the dangers of the rocket turning into a missile. However, this would have been interesting to try and could potentially increase the apogee of the rocket. The times when the rocket is least stable are when it is moving at the lowest velocities. These times are at launch and at apogee. From the launch video, we can see that the rocket did not take off very straight off the launch rail. Active control would have been necessary and would have helped. Thus, if given permission for a future rocket launch, potentially having active stabilization during burnout would create interesting data to compare with this launch.

9. Summary

9.1. Conclusion

While the results of the launch were less than ideal, we are proud of the rocket that we have built. Given the window of opportunity for planning, developing, manufacturing, and assembling the BFR, achieving a rocket with active stabilization features was thought to be a long shot and highly unlikely. However, only the BFR team accepted the challenge and came together to make it work. Looking back on it, there were multiple changes in the design phase that could have been made to make manufacturing the rocket easier, but those hurdles were eventually overcome. Creating a rocket that had the capacity for active stabilization and succeeded in active wind tunnel tests was a great feat and one that should not be overlooked. The BFR is the first actively stabilized rocket to be built throughout MAE 157A's long history. Hopefully, the BFR will pave the way for more actively stabilized rockets and can serve as a model in terms of both shortcomings and strong points for future UCLA students that want to follow in our footsteps.

9.2. Report Contributions Breakdown

Report Contribution	Team Member					
	David Aleman	**Tommy Beres**	**Preston Burke**	**Brendan Morgan**	**Vlad Rosca**	**James Tseng**
Introduction					Introduction	
Conceptual Design			CAD	Open Rocket & CAD		
Analysis		FEA & 2D Nose Cone CFD	Trajectory 3D Fin Analysis	Stability	3D Rocket Analysis	
Active Controls						Controls Paradigm, State Machine
Manufacturing	Budget & Motor	Nose Cone		Control Mechanism	Planning, Body Tube, Recovery System, Fins	Electronics
Testing	Passive stability		Wind Tunnel Coefficient of Drag		Recovery System	Active Stability
Launch Synopsis	Launch Synopsis			Launch Synopsis		
Future Plans and Improvements		Future plans + improvements				
Summary		Summary				

10. References

Cannon, Brent. “MODEL ROCKET SIMULATION WITH DRAG ANALYSIS.” Brigham Young University, Apr. 2004.

11. Appendix

11.1. MATLAB Trajectory Code

```
%% MAE 157A Rocket Trajectory Group 3
clc; clear; close all;

% Load thrust curve files (from ThrustCurve.org)
Thrust = table2array(readtable('AeroTech_F50T.csv'));

% Time Parameters
t = 0; % Initial time [sec]
dt = 0.005; % time step [sec]
t_final = 200; % final time [sec]
Time = (t:dt:t_final);

% Interpolate data to time step
Thrust = rmmissing(interp1(Thrust(:,1), Thrust(:,2), Time));

% Set rocket specifications
rocket_Cd = 1.00; % rocket coefficient of drag
para_D = .7112; % diameter [m] (28 in)
launch_angle = deg2rad(10); % initial launch angle [rad]
para_Cd = 1.55; % parachute coefficient of drag
rocket_D = 0.0444; % diameter [m] (1.75in)
para_A = pi*(para_D/2)^2; % parachute area [m^2]
% rocket_A = rocket_D^2/4*pi; % rocket cross-section area [m^2] Body Tube
rocket_A = 0.002813;% rocket cross-section area [m^2] CAD
M_prop = 37.9/1000; % propellant mass [kg]
M_motor = 84.9/1000; % mass of motor with prop [kg]
M_o = .375; % initial mass [kg]
M_f = M_o - M_prop; % final mass [kg]
% M_f = .495 + M_motor - M_prop; % final mass (dry mass) [kg]
% M_o = M_f + M_motor; % initial mass (wet mass) [kg]
rod_legnth = 1; % Launch rod length [m]
eng_delay = 6; % Engine delay charge time [sec]
burn = Time(length(Thrust)); % Burn time
delay = eng_delay;
ejection = burn + delay;
avg_thrust = 53.7; % average thrust [N]
tot_impulse = 76.8; % total impulse [N-s]
m_dot = avg_thrust*M_prop/tot_impulse; % mass flow rate [kg/s] (assume constant)

%Atmospheric specifications
rho = 1.225; %atmospheric density [kg/m^3]

% Preallocate Arrays
alt = zeros(1,length(Time)); % Altitude [m]
vel = zeros(1,length(Time)); % Velocity [m/s]
acc = zeros(1,length(Time)); % Acceleration [m/s^2]

%% Outer Simulation Loop %%
```

```
k = 1; % tracks time step

% Loop until a set final time
while t < t_final-dt

    % Identify what phase of flight it's in based on burn time and delay time
    if (t <= burn)

        % Burn Phase
        mass = M_o - m_dot * t; % update mass in burn phase
        T = Thrust(k); % current thrust force
        D = 0.5*rho*(vel(k)^2)*rocket_Cd*rocket_A; % current drag force
        if alt(k) > 0
            W = mass * 9.81; % current gravity force
        else
            W = 0; % force is canceled with the ground
        end
    elseif (burn < t) && (t < ejection)

        % Coast Phase
        mass = M_f; % no propellant
        T = 0; % motor is burnt out
        D = 0.5*rho*(vel(k)^2)*rocket_Cd*rocket_A; % current drag force
        if vel(k) < 0
            D = -D;
        end
        W = mass * 9.81; % current gravity force
    elseif (t >= ejection)

        % Recovery
        mass = M_f; % no propellant
        T = 0; % motor is burnt out
        D = 0.5*rho*(vel(k)^2)*para_Cd*para_A; % current drag force
        if vel(k) < 0
            D = -D;
        end
        if alt(k) > 0
            W = mass * 9.81; % current gravity force
        else
            W = 0; % force is canceled with the ground
        end
    end

    F = T*cos(launch_angle) - W - D*cos(launch_angle); % Current force on rocket
    acc(k) = F/mass; %  acceleration
    vel(k+1) = vel(k) + acc(k)*dt; % velocity
    alt(k+1) = (alt(k) + vel(k)*dt); % altitude

    % Prevent the rocket from going through the ground
    if alt(k+1) < 0
        alt(k+1) = 0;
        vel(k+1) = 0;
        index = k;
```

```
        break;
    else
        index = length(alt);
    end

    % Advance time step
    t = t + dt;
    k = k + 1;
end
%% Plots

% shorten arrays
alt = alt(1:index);
vel = vel(1:index);
acc = acc(1:index);
Time = Time(1:index);

% apogee
apogee = max(alt);
apogee_time = Time(alt == apogee);
apogee = 3.28084*apogee; % in ft
% landing
land_time = Time(end);
land_vel = vel(end);
land_vel = 3.28084*vel(end); % in ft

figure
subplot 311
plot(Time, 3.28084*alt, 'b')
xline(burn, 'k--')
xline(apogee_time, 'c--')
xline(land_time, 'm--')
ylabel('Altitude (ft)')
apogee_str = sprintf('Apogee = %.2fft', apogee);
text(apogee_time + 5, apogee - 100, apogee_str)
legend('Simulated', 'Burnout', 'Apogee', 'Landing')
str = sprintf('Simulated w/ CD = %.2f Vehicle / %.2f Parachute', rocket_Cd,
para_Cd);
title(str)

subplot 312
plot(Time, 3.28084*vel, 'b')
xline(burn, 'k--')
xline(apogee_time, 'c--')
xline(land_time, 'm--')
vel_str = sprintf('Landing Velocity = %.2fft/s', land_vel);
text(Time(end)-50, max(vel)/2, vel_str)
ylabel('Velocity (ft/s)')

subplot 313
plot(Time, 3.28084*acc, 'b')
xline(burn, 'k--')
xline(apogee_time, 'c--')
```

```
xline(land_time, 'm--')
xlabel('Time (s)')
ylabel('Acceleration (ft/s^2)')
```

11.2. Onboard Flight Software

This code is implemented on the Teensy 4.1 microcontroller that interfaces the AltIMU sensor and runs the active controls code. This code is also available on GitHub.

```
#include <Wire.h>
#include <LPS.h>
#include <LSM6.h>
#include "SparkFunLSM6DS3.h"
#include <LIS3MDL.h>
#include <SD.h>
#include <Servo.h>
#include <Math.h>
#include "SensorFusion.h"
#include <quaternion_type.h>

const int led = LED_BUILTIN;
int ledIter=0;
LPS ps;
LSM6DS3 imu;
LIS3MDL mag;
SF ahrs;
const int chipSelect =  BUILTIN_SDCARD;
Servo fin1; Servo fin2; Servo fin3;

String filename;

// controls
enum bfr_state{IDLE,PAD,LAUNCH,FLIGHT};
enum bfr_state state = IDLE;

// PD controller
#define Kp 15
#define Kd 2
// mixer gains
#define Lx 3
#define Ly 3
#define Lz 2
// servo limits
#define contr_min -100
#define contr_max 100
#define servo_min 1200 // microsec
#define servo_max 1800

float L1, L2, L3;
uint16_t u1, u2, u3;

// q [1 0 0 0] is flat, starside down facing North (NED)
quat_t q_set = {1,0,0,0};
```

```
// AHRS
float ax_0=0; float ay_0=0; float az_0=0;
#define s_0v 128
float gx_0v[s_0v]; float gy_0v[s_0v]; float gz_0v[s_0v]; uint8_t i_0v=0;
#define s_rv 32
float ax_rv[s_rv]; float ay_rv[s_rv]; float az_rv[s_rv];
float gx_rv[s_rv]; float gy_rv[s_rv]; float gz_rv[s_rv]; uint8_t i_rv=0;
float gx_r=0; float gy_r=0; float gz_r=0;
float gx_0=0; float gy_0=0; float gz_0=0;

// timers
long int time_0=0; // initial time
long int time_p=0; // previous step time
long int time_zero=0;
long int time_print=0;
long int time_led=0;
long int time_pad=0; bool debounce_pad=false;
long int time_burn=0; bool debounce_burn=false;
long int time_launch=0;
long int time_flight=0;

//helper functions
void update_offsets();

void setup() {
  // put your setup code here, to run once:
  pinMode(led, OUTPUT);
  Serial.begin(38400);
  Wire2.begin();

//  initialize Pressure
  if (!ps.init()) {
    while(1){
      Serial.println("Failed to autodetect pressure sensor!");
      digitalWrite(led, HIGH); delay(100);
      digitalWrite(led, LOW ); delay(100);
    };
  }
  ps.enableDefault();

//  initialize IMU
  if (imu.begin() != 0) {
    while(1){
      Serial.println("Failed to detect and initialize IMU!");
      digitalWrite(led, HIGH); delay(100);
      digitalWrite(led, LOW ); delay(100);
    };
  }
  imu.settings.accelRange = 16;      //Max G force readable.  Can be: 2, 4, 8, 16
  imu.settings.gyroRange = 2000;   //Max deg/s.  Can be: 125, 245, 500, 1000, 2000
  imu.begin();

// initialize mag
```

```
  if (!mag.init()) {
    while(1){
      Serial.println("Failed to detect and initialize magnetometer!");
      digitalWrite(led, HIGH); delay(100);
      digitalWrite(led, LOW ); delay(100);
    };
  }
  mag.enableDefault();

// initialize servo
  fin1.attach(0); fin2.attach(1); fin3.attach(2);
  fin1.writeMicroseconds(servo_min); fin2.writeMicroseconds(servo_min);
fin3.writeMicroseconds(servo_min);
  delay(500);
  fin1.writeMicroseconds(servo_max); fin2.writeMicroseconds(servo_max);
fin3.writeMicroseconds(servo_max);
  delay(500);
  fin1.writeMicroseconds(1500); fin2.writeMicroseconds(1500);
fin3.writeMicroseconds(1500);

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    while (1) {
      // No SD card, so don't do anything more - stay stuck here
      Serial.println("Card failed, or not present");
      digitalWrite(led, HIGH); delay(100);
      digitalWrite(led, LOW ); delay(100);
    }
  } else {
    int i=0;
    filename="BFR_log_"+String(i)+".txt";
    while (SD.exists(filename.c_str())) {
      i+=1;
      filename="BFR_log_"+String(i)+".txt";
    }
//     Serial.println("Card initialized, logging to: "+ filename);
  }
  File dataFile = SD.open(filename.c_str(), FILE_WRITE);
  if (dataFile) {
    dataFile.println("t [ms], p [mbar], alt [m], T [c], ax [g], ay [g], az [g], gx
[rad/s], gy [rad/s], gz [rad/s], q1, q2, q3, q4, state, u1, u2, u3");
    dataFile.close();
  } else {
   while (1){
    Serial.println("Error opening " + filename);
    digitalWrite(led, HIGH); delay(100);
    digitalWrite(led, LOW ); delay(100);
    }
  }

  // init timers
  time_0=millis(); time_p=millis();
}
```

```
void loop() {
  // // // // // // // // //
  // READ SENSORS
  // // // // // // // // //
  float pressure = ps.readPressureMillibars();
  float altitude = ps.pressureToAltitudeMeters(pressure);
  float temperature = ps.readTemperatureC();
  ax_rv[i_rv]=imu.readFloatAccelX();
  ay_rv[i_rv]=imu.readFloatAccelY();
  az_rv[i_rv]=imu.readFloatAccelZ();
  // smooth out noisy gyro measurements
  gx_rv[i_rv]=imu.readFloatGyroX();
  gy_rv[i_rv]=imu.readFloatGyroY();
  gz_rv[i_rv]=imu.readFloatGyroZ(); i_rv=(i_rv+1)%s_rv;
  gx_r=0; gy_r=0; gz_r=0;

  // average across array
  float ax_r=0; float ay_r=0; float az_r=0;
  for(uint8_t i=0; i<s_rv; i++) {
    gx_r+=gx_rv[i]; gy_r+=gy_rv[i]; gz_r+=gz_rv[i];
    ax_r+=ax_rv[i]; ay_r+=ay_rv[i]; az_r+=az_rv[i];
  }
  gx_r/=s_rv; gy_r/=s_rv; gz_r/=s_rv;
  ax_r/=s_rv; ay_r/=s_rv; az_r/=s_rv;
  float ax=ax_r-ax_0; float ay=ay_r-ay_0; float az=az_r-az_0;
  float gx=(gx_r-gx_0)*DEG_TO_RAD;
  float gy=(gy_r-gy_0)*DEG_TO_RAD;
  float gz=(gz_r-gz_0)*DEG_TO_RAD;

  mag.read(); float mx=mag.m.x; float my=mag.m.y; float mz=mag.m.z;

  // AHRS update
  float dt = ahrs.deltatUpdate(); time_p = millis();
  ahrs.MadgwickUpdate(gx,gy,gz,ax,ay,az,mx,my,mz,dt);
//  ahrs.MadgwickUpdate(gx,gy,gz,ax,ay,az,dt);
  float* q_r = ahrs.getQuat();
  quat_t q_imu = quat_t(q_r).norm();

  // // // // // // // // //
  // STATE TRANSITION LOGIC
  // // // // // // // // //

// IDLE TO PAD
  float a_norm = sqrt(ax_r*ax_r + ay_r*ay_r + az_r*az_r);
  if (state == IDLE && ax_r > 0.95 && a_norm > 0.95 && a_norm < 1.05) {
    if (debounce_pad) {
      if (millis()-time_pad > 5000){
        state = PAD; debounce_pad = false;
      }
    } else {debounce_pad = true; time_pad = millis();}
  } else {debounce_pad = false;}
```

```
//  EXIT PAD TO IDLE
  if (state == PAD && ax_r < 0.95 && a_norm > 0.95 && a_norm < 1.05) {
    state = IDLE;
  }

// PAD TO LAUNCH
  if (state == PAD && ax_r > 5) {
    if (debounce_burn) {
      if (millis()-time_burn > 250){
        state = LAUNCH; time_launch = millis(); debounce_burn = false;
      }
    } else {debounce_burn = true; time_burn = millis();}
  } else {debounce_burn = false;}

//  LAUNCH TO FLIGHT
  if (state == LAUNCH && millis()-time_launch>1400) {
    state = FLIGHT; time_flight = millis();
  }

  //  FLIGHT TO IDLE
  if (state == FLIGHT && millis()-time_flight>10000) {
    state = IDLE;
  }

  // // // // // // // // //
  // STATE IDLE
  // // // // // // // // //
  if (state == IDLE) {
//    light
    if (millis()-time_led>=1000) {digitalWrite(led, HIGH); time_led = millis();}
    else {digitalWrite(led, LOW);}

    fin1.writeMicroseconds(1500); fin2.writeMicroseconds(1500);
fin3.writeMicroseconds(1500);
  }

  // // // // // // // // //
  // STATE PAD
  // // // // // // // // //
  else if (state == PAD) {
    digitalWrite(led, HIGH);
    update_offsets();
    q_set = q_imu;

    fin1.writeMicroseconds(1500); fin2.writeMicroseconds(1500);
fin3.writeMicroseconds(1500);
  }

  // // // // // // // // //
  // STATE LAUNCH
  // // // // // // // // //
  else if (state == LAUNCH) {
```

```
    fin1.writeMicroseconds(1500); fin2.writeMicroseconds(1500);
fin3.writeMicroseconds(1500);
  }

  // // // // // // // // //
  // STATE FLIGHT
  // // // // // // // // //
  else if (state == FLIGHT) {
    //    light
    if (millis()-time_led>=200) {digitalWrite(led, HIGH); time_led = millis();}
    else {digitalWrite(led, LOW);}

    quat_t q_diff = q_set * q_imu.conj();
    int8_t sgn = 1;
    if (q_diff.get(0) < 0) { sgn = -1; }

//    PD controller - diff coordinates based on Controls Scheme
    float Mbx = - sgn * Kp * q_diff.get(2) - Kd * gy;
    float Mby = - sgn * Kp * q_diff.get(3) - Kd * gz;
    float Mbz = - sgn * Kp * q_diff.get(1) - Kd * gx;
//    allocation
    Mbx *= Lx; Mby *= Ly; Mbz *= Lz;
    float sq3 = sqrt(3);
    L1 =        0 + 2*Mby + Mbz;
    L2 = -sq3*Mbx -   Mby + Mbz;
    L3 =  sq3*Mbx -   Mby + Mbz;
//    fin mapping
    u1 = map(L1,contr_min,contr_max,servo_min,servo_max);
    u2 = map(L2,contr_min,contr_max,servo_min,servo_max);
    u3 = map(L3,contr_min,contr_max,servo_min,servo_max);
//    limit
    u1 = min(servo_max,max(servo_min,u1));
    u2 = min(servo_max,max(servo_min,u2));
    u3 = min(servo_max,max(servo_min,u3));
//    servo output
    fin1.writeMicroseconds(u1); fin2.writeMicroseconds(u2);
fin3.writeMicroseconds(u3);
  }

  // // // // // // // // //
  // LOG DATA
  // // // // // // // // //
//  log at 10 Hz
  if (millis()-time_print > 100) { time_print = millis();
  // "t [ms], p [mbar], alt [m], T [c], ax [g], ay [g], az [g], gx [rad/s], gy
[rad/s], gz [rad/s], q1, q2, q3, q4, state, u1, u2, u3"
    String dfs = String(millis())+", "+String(pressure)+", "+String(altitude)+",
"+String(temperature)+", ";
    dfs += String(ax)+", "+String(ay)+", "+String(az)+", "+String(gx)+",
"+String(gy)+", "+String(gz)+", ";
    dfs += String(q_imu.get(0))+", "+String(q_imu.get(1))+",
"+String(q_imu.get(2))+", "+String(q_imu.get(3))+", ";
```

```
    dfs += String(state)+", "+String(u1)+", "+String(u2)+", "+String(u3);

  // open the file.
    File dataFile = SD.open(filename.c_str(), FILE_WRITE);

  // if the file is available, write to it:
    if (dataFile) {
      dataFile.println(dfs);
      dataFile.close();
    }
    String ds = String(u1)+" "+String(u2)+" "+String(u3);
    Serial.println(ds);
  }
}

// // // // // // // // // //
// HELPER FUNCTIONS
// // // // // // // // // //
void update_offsets() {
  gx_0v[i_0v]=gx_r; gy_0v[i_0v]=gy_r; gz_0v[i_0v]=gz_r; i_0v=(i_0v+1)%s_0v;
  gx_0=0; gy_0=0; gz_0=0;
  // average across array
  for(uint8_t i=0; i<s_0v; i++) {
    gx_0+=gx_0v[i]; gy_0+=gy_0v[i]; gz_0+=gz_0v[i];
  }
  gx_0/=s_0v; gy_0/=s_0v; gz_0/=s_0v;
}
```

11.3. Wind Tunnel Testing Videos

11.3.1. Passive Stability – Preliminary

Summary video of the preliminary BFR rocket at speeds of 15, 20, 30, and 40 m/s.

11.3.2. Passive Stability – Final

Video of the complete BFR rocket (sans paint) at 30 m/s with active control off.

11.3.3. Active Stability – Final

Video of the complete BFR rocket (sans paint) at 30 m/s with active control on.