# Practical Application of Dynamic Air Braking System for Atmosphere Reentry

Judson Upchurch

## Problem/Question

Objects entering the atmosphere of planets require something to slow them down from their orbital velocity down to a safe speed. The most common way is for the vehicle to apply aerodynamic braking. Vehicles such as the Apollo Command Module used a blunt body and heatshield for just this purpose. In atmospheres that are significantly less dense, like Mars's which is 100 times less dense than Earth's.

What would be a practical way to slow vehicles down during reentry? Is there a solution that reduces the chance of catastrophic failure from heat?

## Goals and Criteria

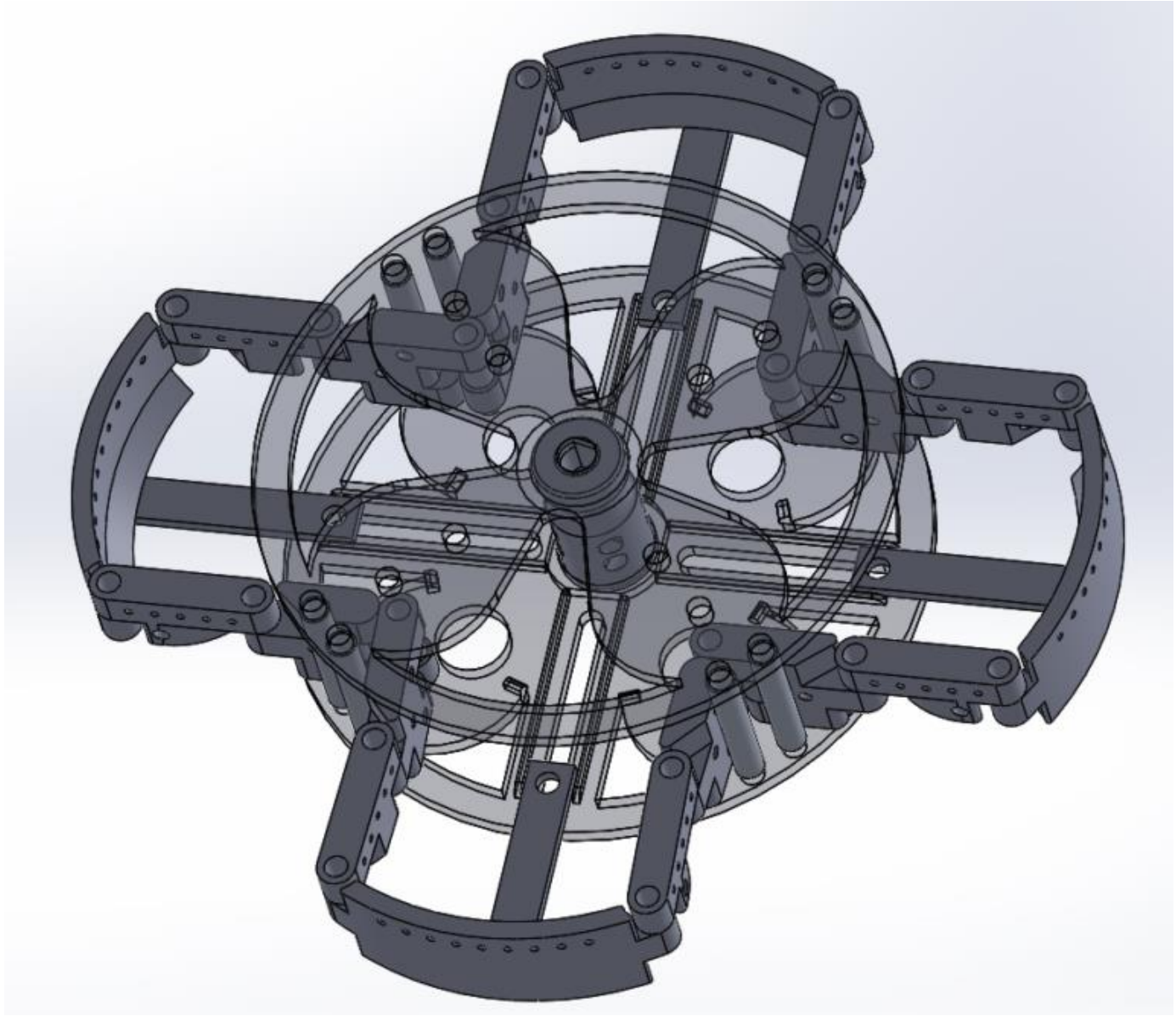
My project's goal is to test the application of aerodynamic braking in rocket launches from the surface. If the project is successful, the system will be able to precisely control the drag on the rocket in a safe and reliable manner.

The project can be considered a success if the rocket's apogee can be controlled to within ±1m. In addition, the parachute control system will be considered successful if it can control the total flight time to ±1.5s.

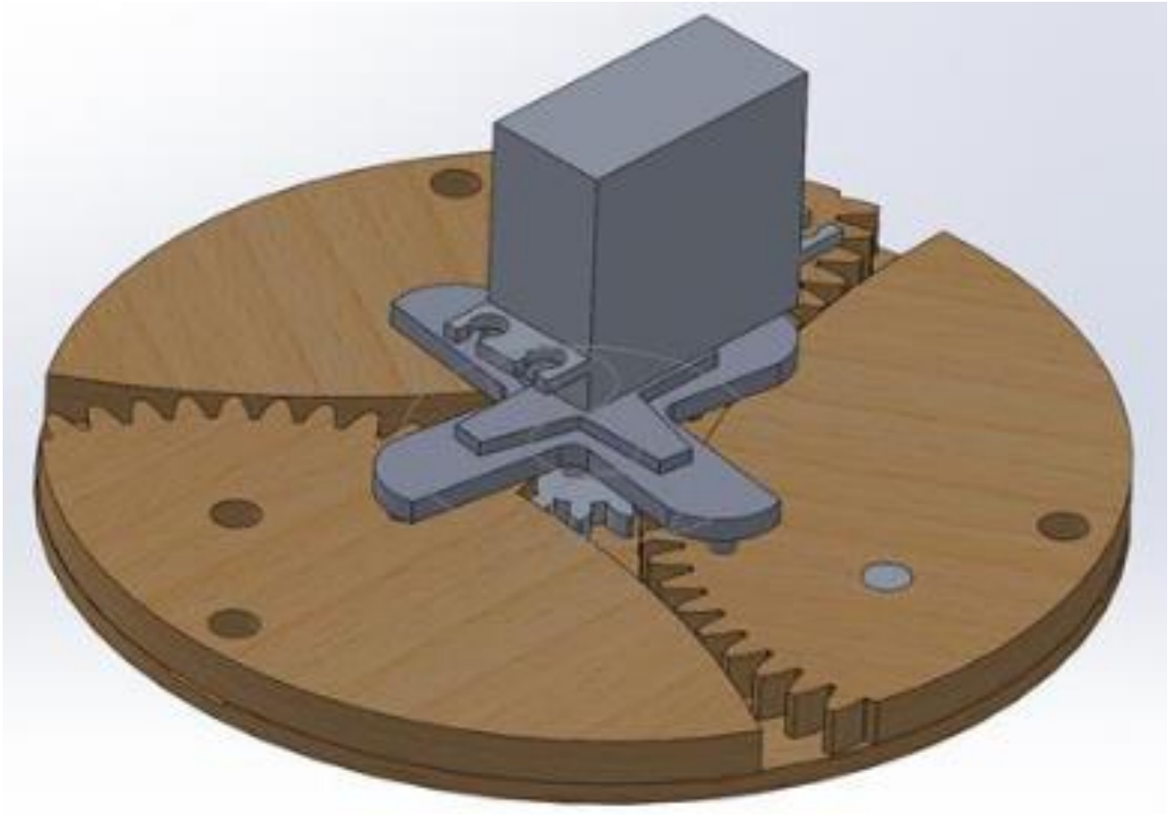
## Background Research

I first began my research by trying to find other people's efforts in rocketry to control the apogee of their rockets. Many schools compete in competitions that emphasize precise apogee deployment so I felt like there would be some good sources.

My efforts first led me to Casey Wilson's website (1): https://caseywilsonaero.com
I was able to find his first design using the "flap" system. However, he explained in detail how it was inefficient. Instead, he moved towards the "sail" design. His CAD design is pictured below.



Casey Wilson's design to felt more complicated than what I was going for. So, I continued searching and producing my own designs when I came across the 2018 University of Ottawa Student Team of Aeronautics and Rocketry design (3). There design implemented a more streamlined and efficient mechanism for deploying out a surface to create drag. Their CAD design is pictured below.



This design greatly improved upon Casey's. Once I felt like I was confident that this design would work well, I worked on modeling my own and making my own improvements in order to make it as effective as possible.

## Works Cited

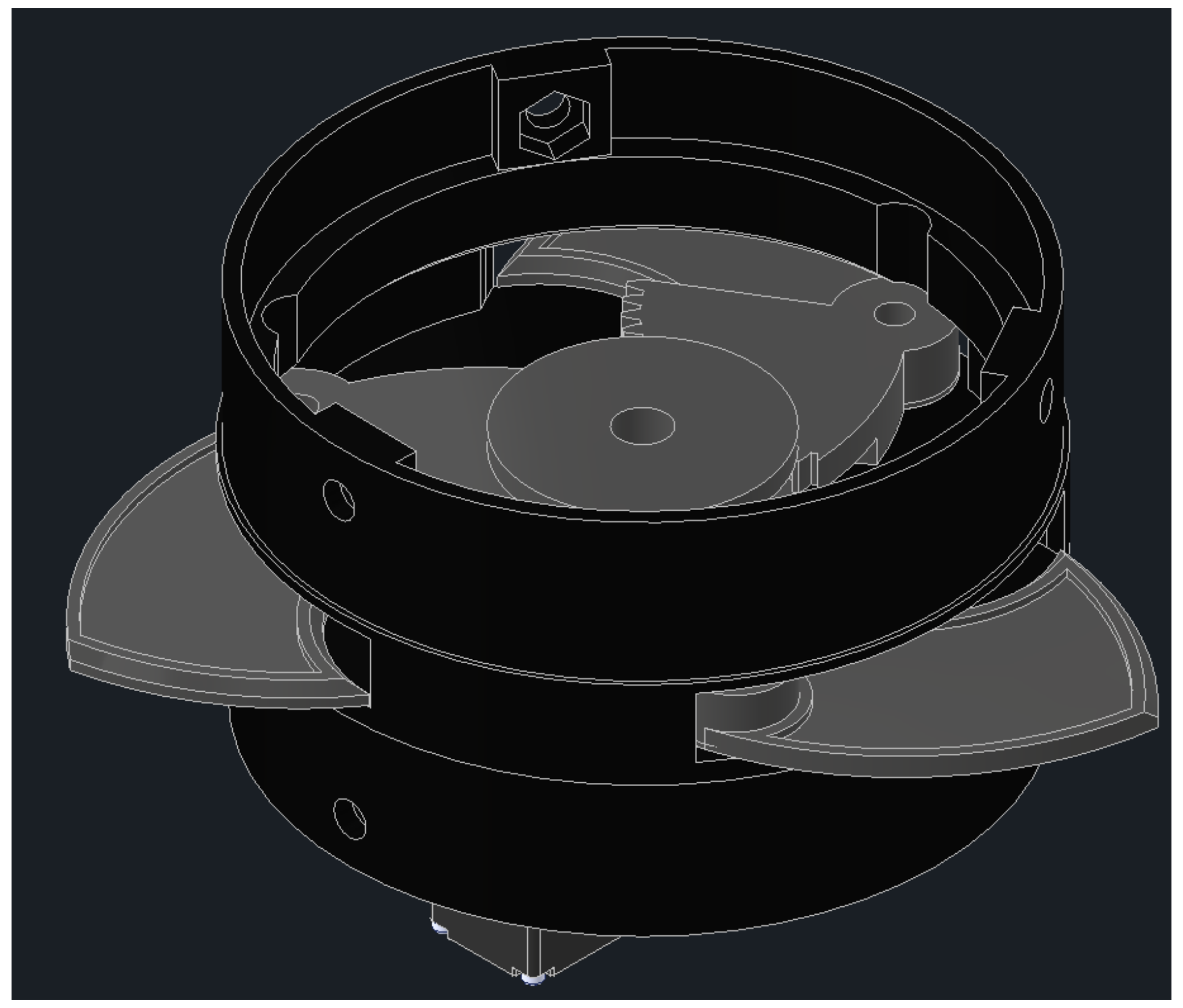
(1) Wilson, Robert. Sail Brake. https://caseywilsonaero.com/rocketry/active-drag/sail-brake/

(2) Culp, Randy. (2008). Rocket Equations Quick Reference, RocketMime. http://www.rocketmime.com/rockets/qref.html

(3) University of Ottawa Student Team of Aeronautics and Rocketry. (2018). It's Not Rocket Science; It's Simple Latte. https://www.soundingrocket.org/uploads/9/0/6/4/9064598/55_project_report.pdf

## My Design and Parts List

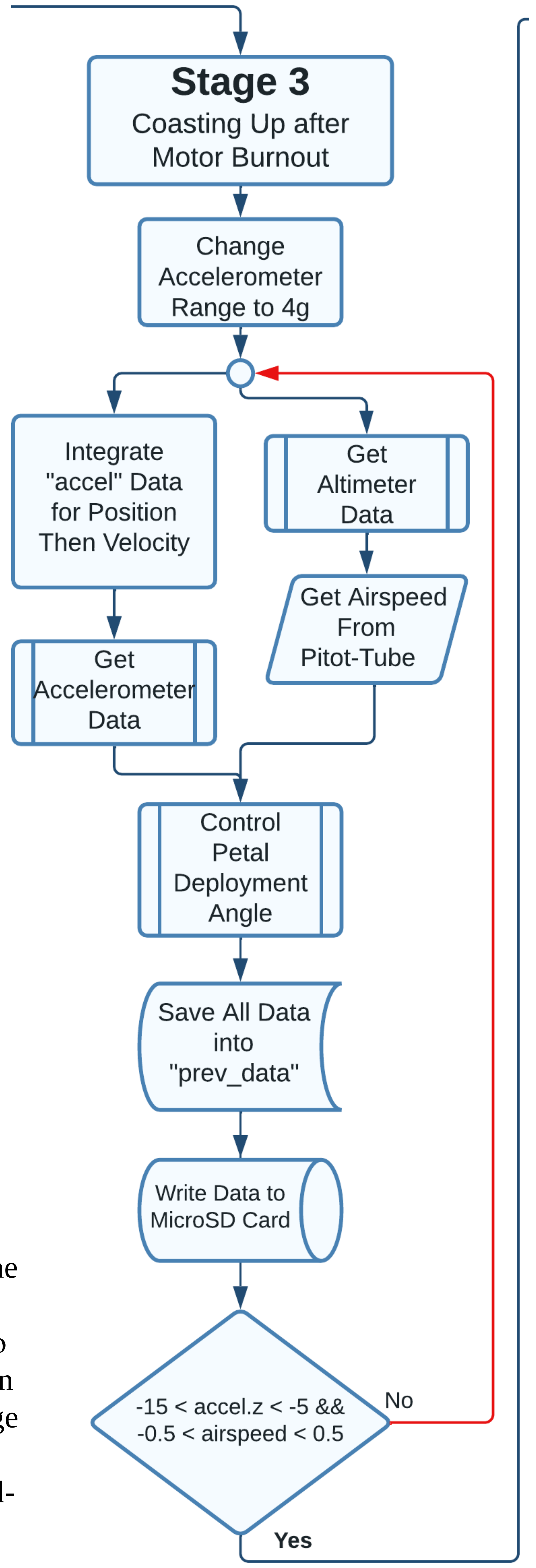
The structural design incorporates a single servo with a range of motion of 270 degrees attached to a center gear. There are 3 petals on the periphery with gearing on them so that every degree turn of the servo is 1/3rd of a degree turn of the petals. Here is a CAD image of the system:



On the flight computer side, there consists of the hardware:

1. Accelerometer (ICM20948)
2. Altimeter (BMP280)
3. Pitot-Tube + Differential Pressure Sensor (XGMP3v3)
4. Servo (HS-70MG)
5. Microcontroller (Seeeduino XIAO)
6. 11.1V 3S LiPo Battery
7. 5V Regulator (7805 MOSFET)
8. MicroSD Card Reader/Writer
9. Sound Buzzer

However, the most important part of the design is the software for the flight computer. The code is written in Arduino (C/C++) for the Seeeduino XIAO. It is split up into many "stages" that correspond to different parts of the launch. Stage 3 is where the flight computer can control the petals and the drag of the rocket. A block diagram of stage 3 is pictures to the right. None of this would be possible without the equation needed to dynamically predict the apogee of the rocket mid-flight. Randy Culp and his website http://rocketmime.com were instrumental in helping me derive an equation (2). After finding his generic equation, I was able to manipulate it in order to cancel out terms such as air density and coefficient of drag which would have made the equation nearly useless. The equation can be seen in the "Control Petal Deployment Angle Function" block diagram.



Stage 3
Coasting Up after Motor Burnout

Change Accelerometer Range to 4g

Integrate "accel" Data for Position Then Velocity

Get Accelerometer Data

Get Altimeter Data

Get Airspeed From Pitot-Tube

Control Petal Deployment Angle

Save All Data into "prev_data"

Write Data to MicroSD Card

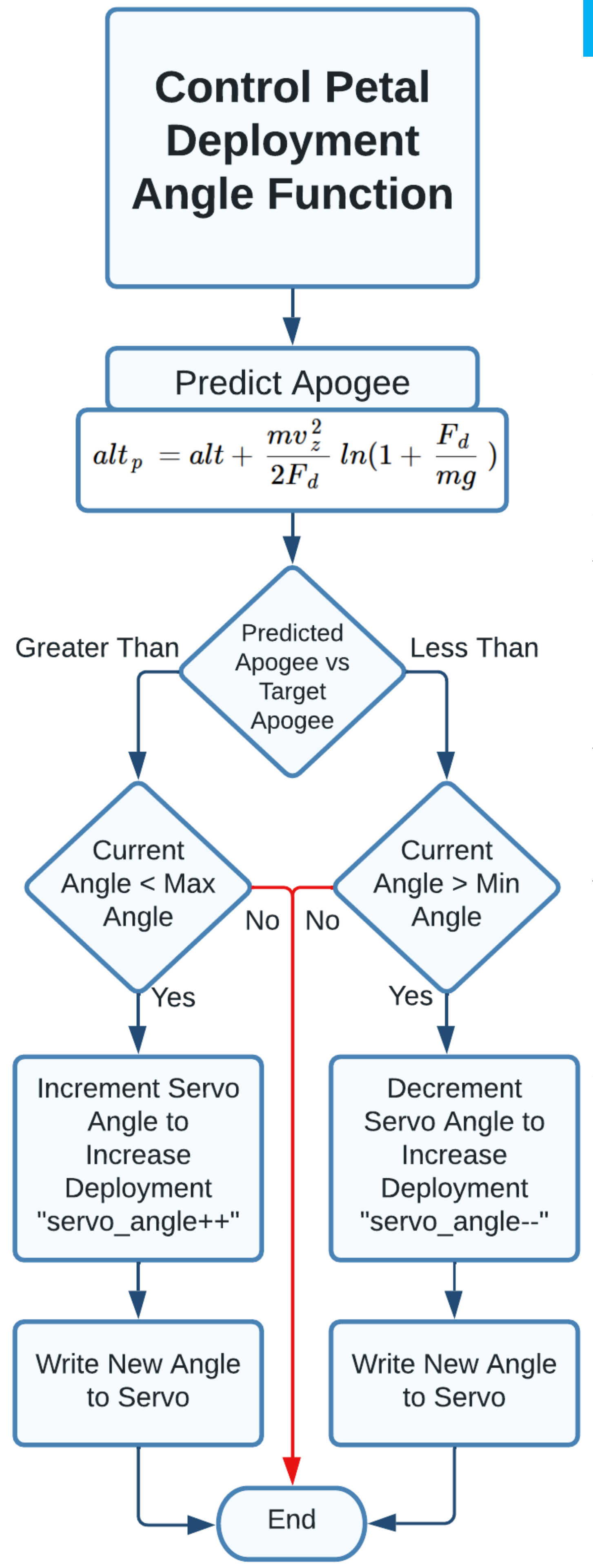
-15 < accel.z < -5 && -0.5 < airspeed < 0.5

No

Yes

## How it Works

In general, after the motor ignites and completely burns, the rocket will take in data from its pitot-tube, accelerometer, and altimeter in order to predict an apogee and then follow the logic below to control the deployment of the petals.



Control Petal Deployment Angle Function

Predict Apogee

$$alt_p = alt + \frac{mv_z^2}{2F_d} ln(1 + \frac{F_d}{mg})$$

Predicted Apogee vs Target Apogee

Greater Than

Less Than

Current Angle < Max Angle

Current Angle > Min Angle

No

No

Yes

Yes

Increment Servo Angle to Increase Deployment "servo_angle++"

Decrement Servo Angle to Increase Deployment "servo_angle--"

Write New Angle to Servo

Write New Angle to Servo

End

## Testing

Testing the system has been difficult due to lack of availability of rocket motors and days available to launch.

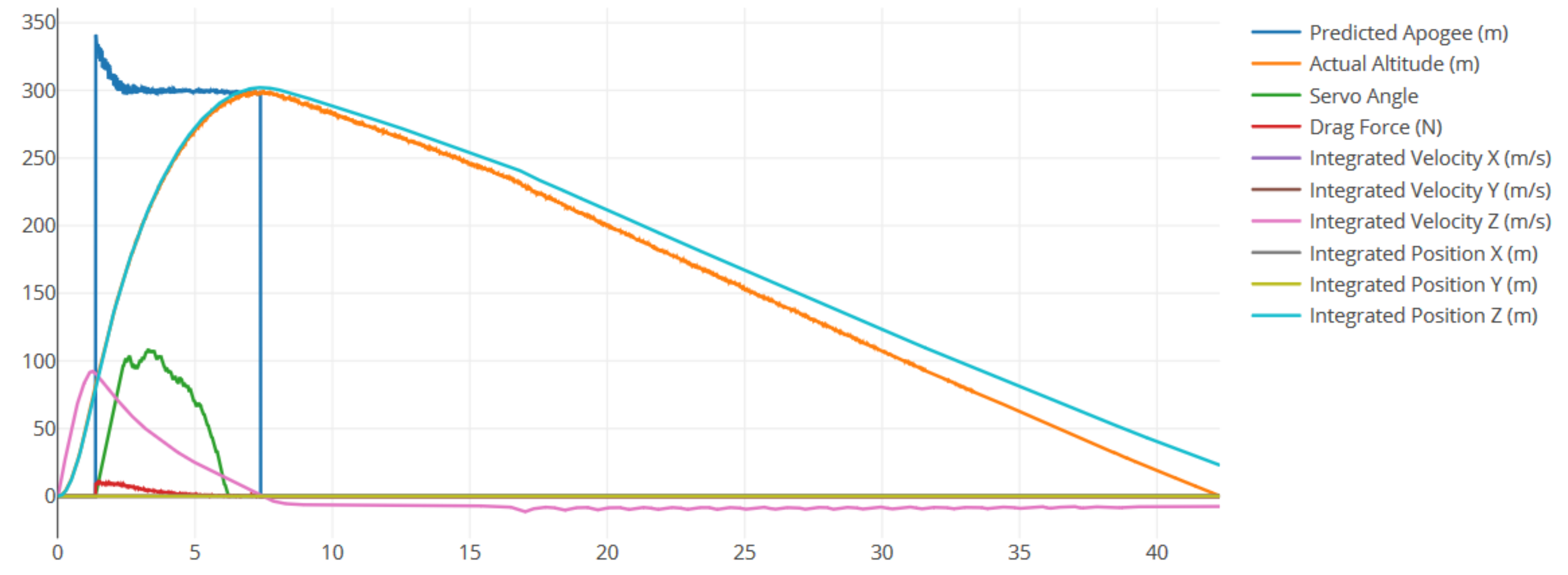
In order to get an idea of how the system should perform, I programmed a physics engine in NodeJS that simulates all of the forces on the rocket that it would experience in a launch. The simulator is designed to insert error into the "readings" of the sensors that the rocket would typically have to get itself.

With an applied error of 5% for altitude, 10% for acceleration, and 5% for airspeed, the rocket was able to control its apogee to within 0.75m, coming all the way down from 340m with no petals to a target of 300m.

## Conclusion/Future

Based solely upon the physics engine testing that I was able to complete, I believe that my project has been a success. It has proven without a shadow of a doubt that it can manipulate its own drag characteristics in order to maintain a predicted apogee with ever changing conditions.

I plan on testing the rocket with an actual launch in the next coming weeks in order to prove its flight capability. I would also like to implement this design on a high-powered rocket that can reach many kilometers in the sky. This would allow it a chance to control not only its ascent but also its descent., Therefore, it could provide a more realistic picture of the systems ability to control a vehicle on reentry.



Predicted Apogee (m)
Actual Altitude (m)
Servo Angle
Drag Force (N)
Integrated Velocity X (m/s)
Integrated Velocity Y (m/s)
Integrated Velocity Z (m/s)
Integrated Position X (m)
Integrated Position Y (m)
Integrated Position Z (m)