Zack Furman NARAM-60 R&D Summary

Previous research on active, electronic control of model rockets has mostly centered on attitude stabilization, especially to a vertical orientation. However, in full-scale rocketry, active control of a rocket's position (as well as attitude) is ubiquitous. In this work, we develop an aerodynamic control system capable of autonomously adjusting a rocket's vertical trajectory, so as to fly to a targeted apogee altitude. We first explain the theoretical model of an active rocket, and the necessary components of a successful system. The rocket must have some means of affecting its altitude, by either adding or subtracting energy, such as air brakes. The rocket must predict the target altitude before apogee, and determine the error from its target apogee. This error can then be fed into a PID loop to determine the magnitude of the control response. Finally, the rocket must have some means of determining and representing its current state, such as via a finite-state machine. We then discuss the implementation of this theoretical model into a physical rocket, and the novel or important solutions used in doing so. Unlike many other model rocketry projects, the flight computer came with most sensors integrated, and had a large pre-existing codebase that included features like logging and SIL/HIL simulation. We determine the sensors necessary to produce the estimation of apogee altitude, and use a Kalman filter to fuse them. Finally, we create a model to represent the drag produced by air brakes, and show how to physically implement them on the rocket. The results show some improvement compared to most passively controlled systems, but lack consistency for reasons that are difficult to determine with current hardware.

-Zach

An Autonomous Air Braking System for Precision Altitude Control

Zach Furman, Giles Beebe, Finn Bjerknes, Daniel Montgomery
Menlo-Atherton High School

Abstract

Previous research on active, electronic control of model rockets has mostly centered on attitude stabilization, especially to a vertical orientation. However, in full-scale rocketry, active control of a rocket's position (as well as attitude) is ubiquitous. In this work, we develop an aerodynamic control system capable of autonomously adjusting a rocket's vertical trajectory, so as to fly to a targeted apogee altitude. First, we discuss the theory behind such a system, including estimation, control methods, and state representation. We then discuss its implementation, including the sensors, flight computer, and drag model. The results show improvement compared to most passively controlled systems, but lack consistency for reasons that are difficult to determine with current hardware.

Nomenclature

g	Acceleration due to gravity
ρ	Air density
C_d	Coefficient of drag
A	Frontal area
m	Mass
v_0	Current velocity
v_T	Terminal velocity
ln	Natural logarithm function

1. Introduction

The motivation for the design of this system comes from the Team America Rocketry Challenge (TARC), which asks competitors to, among other goals, fly their rocket as close as possible to a designated apogee altitude target.

Simulations showed that engine impulse variation (as given by NAR motor datasheets) alone was enough to significantly impact the apogee altitude of the rocket with a standard deviation of around 20 feet (much too far from a winning score). Even if a team could successfully correct for other factors such as wind, launch tilt, and air density, engine impulse variation is very difficult to predict, meaning that there is a minimum amount of error inherent in predicting the apogee altitude. Without some form of in-flight control of the rocket to correct for this error after launch, it is theoretically impossible to win the competition with consistency.

However, the applicability of the work extends beyond the competition alone, and furthers the development of future guidance, navigation, and control (GNC) systems in model rocketry.

The objectives of the work are twofold:

- To create an altitude control system for use in TARC competition, and
- To further the development of actively controlled model rockets via a single-axis trajectory control system

Because only the second objective has broader significance to model rocketry as a whole, it is the one focused on in this paper. Subsequent discussions of the TARC competition will be limited to its role in design restrictions and considerations.

Finally, it should be noted that the work for this project extended across two years (and two rockets). However, information in this paper is not presented in chronological order. Description of the rocket design generally refers either to both rockets or to the final rocket only.

1.1 Related Work

Much of the techniques and topics used in this paper have been previously described at NARAM, including apogee attitude prediction, Kalman filters, and active control systems.

Previous work has been presented on apogee prediction for precision altitude events, including applying simulation software (e.g. RockSim) to predict rockets' apogee altitude prior to launch [1]. However, as mentioned earlier in this section, engine impulse variation limits the possible accuracy of this pre-flight prediction, even without the potential inaccuracy of the simulation's drag model.

Kalman filters, which are used in this project, have also been presented previously. David W. Schultz' 2002 and 2004 NARAM reports demonstrate the implementation of a Kalman filter for altitude

smoothing and apogee detection [2][3]. In our system, both acceleration and altitude (as opposed to just altitude) are used as inputs to a Kalman filter.

Active systems have been previously deployed on model rockets before, but usually only for stabilization to a vertical orientation, as in Brian J. Guzek's 2009 NARAM report [4]. Our system does not actively stabilize orientation, instead attempting to change the rocket's vertical trajectory.

2. Theory

Fundamentally, any closed-loop control system must have some method of estimating a system's current state, and a way of adjusting that state. In this case, the controlled variable is the rocket's apogee altitude. In other words, the system estimates what the apogee the apogee altitude will be, and actively adjusts (using methods discussed in 2.2) the rocket towards the target apogee altitude.

2.1 Estimation

Because apogee altitude can only be directly measured at apogee, it is necessary to predict the altitude in advance. This estimate is necessary to determine the predicted error from the target altitude.

The apogee altitude can be calculated from the rocket's current altitude and velocity as follows:

$$y = \frac{{v_T}^2}{2g}\, ln\, (\frac{{v_0}^2+{v_T}^2}{{v_T}^2})$$

where

$$v_T = \sqrt{\frac{2mg}{C_d A \rho}}$$

The process error is the difference between the predicted apogee altitude and the target altitude.

2.2 Control

In a passive system, a rocket with a given thrust profile will tend to reach a given apogee altitude. In order to change this altitude, an active system must be either *additive* or *subtractive*.

An additive system starts at a lower predicted apogee altitude and adds energy in order to increase the apogee towards the target altitude. For example, igniting extra engines mid-air.

A subtractive system starts at a higher predicted apogee altitude and bleeds off energy in order to lower the apogee towards the target altitude. E.g. increasing drag on the rocket mid-air.

While technically possible, additive systems are difficult to execute (ex. the logistical challenges of reliably igniting engines mid-air, with deterministic delay and quantized impulse). In most cases, a subtractive system appears to be the more attractive option.

Once settled on a subtractive system, the question turns toward the physical method of control for the rocket. In other words, the mechanism with which the rocket will use to subtract impulse. There are multiple ways to accomplish this (detaching mass, opening the parachute early, etc.), but addition of aerodynamic drag seemed easiest. This can be realized by simply increasing the frontal area (and drag coefficient) of the rocket, via servo-actuated flaps, which we referred to as "drag brakes".

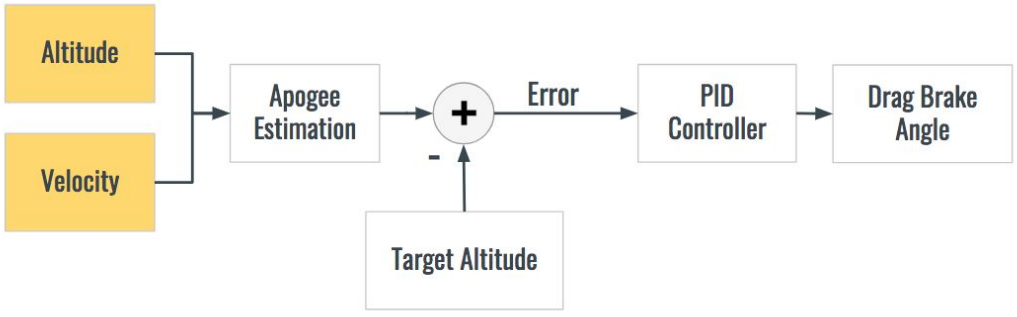
It should be noted that a significant downside to this drag-based method is that the system has more control authority the faster the rocket is going, meaning that it has relatively high control of apogee altitude farther from apogee and low control closer to apogee. Hence the rocket has the least control authority when the computer's apogee estimate is most accurate (close to apogee). However, we decided that with a reasonably accurate apogee estimator, the system would still converge to the target altitude, and so the ease of implementation outweighed the downside of this method.

One final definition: the rocket is considered inside of the *control range* when the drag brakes are neither fully closed nor fully deployed. This effectively measures whether the active system still has control over the rocket. If the entire flight stays within the control range, and the PID is properly tuned, the altitude error should theoretically be zero. This makes it an important metric for measuring flight performance.

A PID controller can be used to connect the process error (calculated in 2.1) to the control method. Given the process error, the controller produces the magnitude of correction from the active system. The PID constants are tuned manually.

In order to provide a more constant flap angle, the PID controls the angular rate of the flaps, rather than their absolute angular position. This is desirable because it keeps the flaps closer to the center of the control range for the duration of the flight.

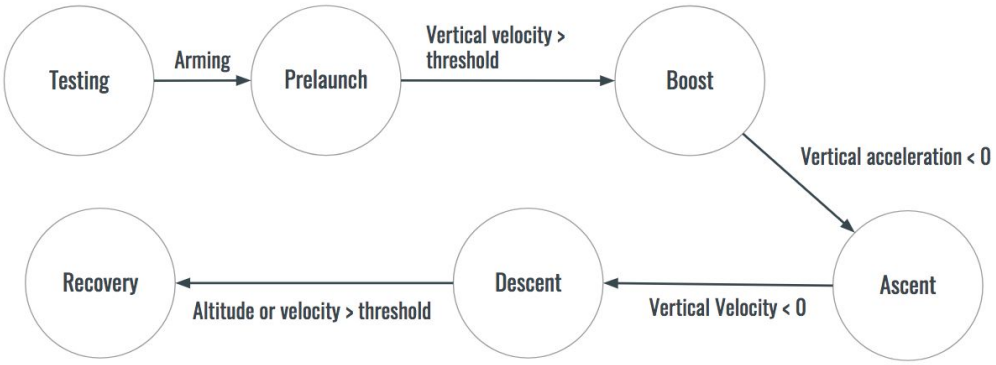
The entire process can be visualized as follows:



2.3 State

In order for the flight code to determine when to begin actuating the drag brakes and to deploy the parachute it must detect when the motor is firing, when the rocket is free-falling, etc. We used a finite-state machine to represent these states and detect transitions between these states. Because the system is reset after every flight and flight states are strictly ordered chronologically, the state machine for this rocket can be linear. In other words, there is one and only one transition between every state, and they proceed in sequential order.

The state machine for the rocket, in abstract, appears as follows:



State	Description
Testing	5-second exercise of drag brakes and recovery system trigger
Prelaunch	Awaiting launch
Boost	Launch detected
Ascent	After engine burnout; drag brake servo power enabled
Descent	After rocket apogee
Recovery	Parachute deployment; altitude/velocity threshold

3. Implementation

In order to understand the rocket's design, it is first important to discuss the considerations that influenced it. The design was constrained by several factors, both arbitrary and practical. First, TARC limits on weight (650 g) and motor impulse (80 Ns) required a low-weight, low-drag rocket in order to meet the the altitude target. Second, the availability of funds limited the budget that could be spent on the rocket, and the deadline limited the time that could be spent on the rocket. Finally, and most obviously, the rocket was designed to consistently perform well at flying close to the altitude target.

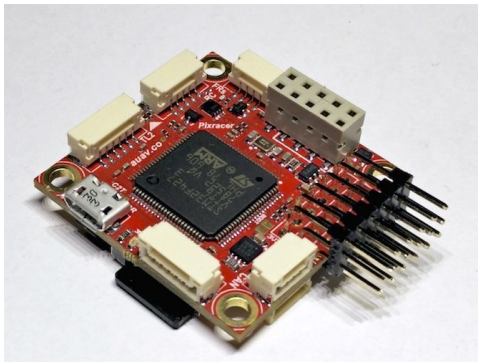
A second important point to note is that this section does not cover the entire physical construction of the rocket. Solutions not unique/essential to our rocket (3D printing, aero fairings, etc.) or too specific to the exact implementation (attachment points, assembly methods, etc). are not covered. The aim of this section is to succinctly cover the critical components of our implementation, rather than provide an exact guide for construction of the rocket.

3.1 Flight computer

The most common flight computers for model rocketry are microcontrollers (e.g. Arduinos) with off-the-shelf expansion boards ("shields") or custom electronics added. However, designing a custom embedded flight controller is time-intensive, and requires creation and integration of hardware and development of low-level software.

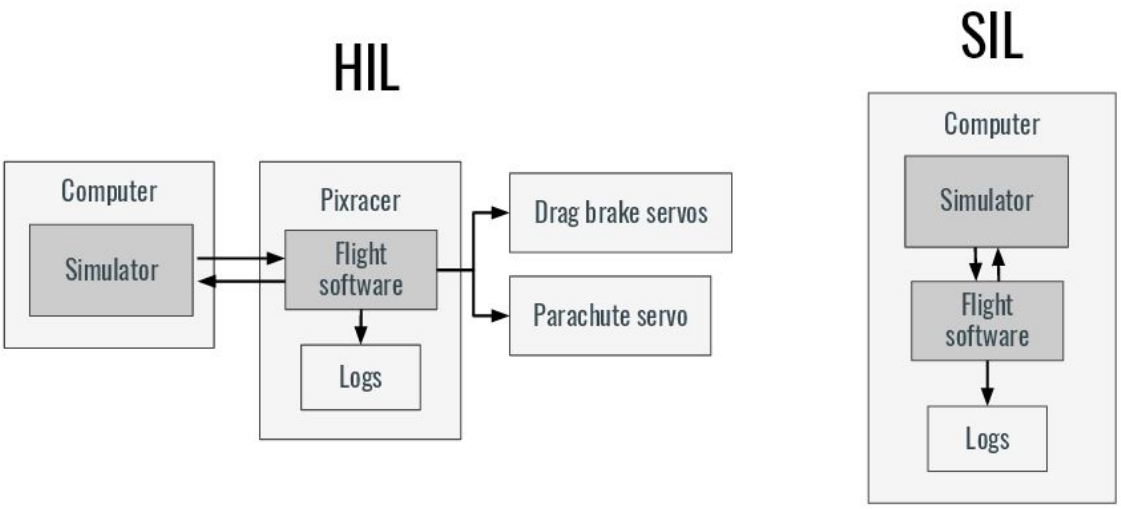
Luckily, it turns out that electronics designed for hobbyist aerospace already exist, in the form of flight computers aimed at multi-rotor and fixed-wing RC aircraft. These come with necessary sensors and servo control hardware built-in.

In our case, we used the Pixracer, a flight computer that is a member of a family of open-hardware platforms that share a common codebase (PX4) [5]. It weighs only 10.5 grams, and features a 180 MHz ARM processor with 256 KB of SRAM. It incorporates an onboard barometer, 6-DOF inertial sensor, a 3-axis magnetometer and interfaces to external sensors, such as GPS.



The PX4 open-source software framework, which has wide community support, provides the means to actuate servos in response to sensor data. The PX4 code also provides useful infrastructure, such as logging of flight data to onboard storage, download/graphing of that log data and real-time monitoring/calibration of sensors through ground control station (GCS) software that can run on a laptop.

Finally, the PX4 software environment provides hardware-in-the-loop (HIL) and software-in-the-loop (SIL) simulation, which uses a Java program running on a connected PC to simulate the entire flight. HIL was especially useful, because it connected the simulator to the Pixracer, feeding it simulated data and allowing it to move servos as it would in flight. It allowed ground-testing of the actual flight code on the target hardware, including servo actuation, without requiring time-intensive, costly flight testing. It also allowed verification of essential software behaviors such as parachute deployment without risking damage to the rocket.



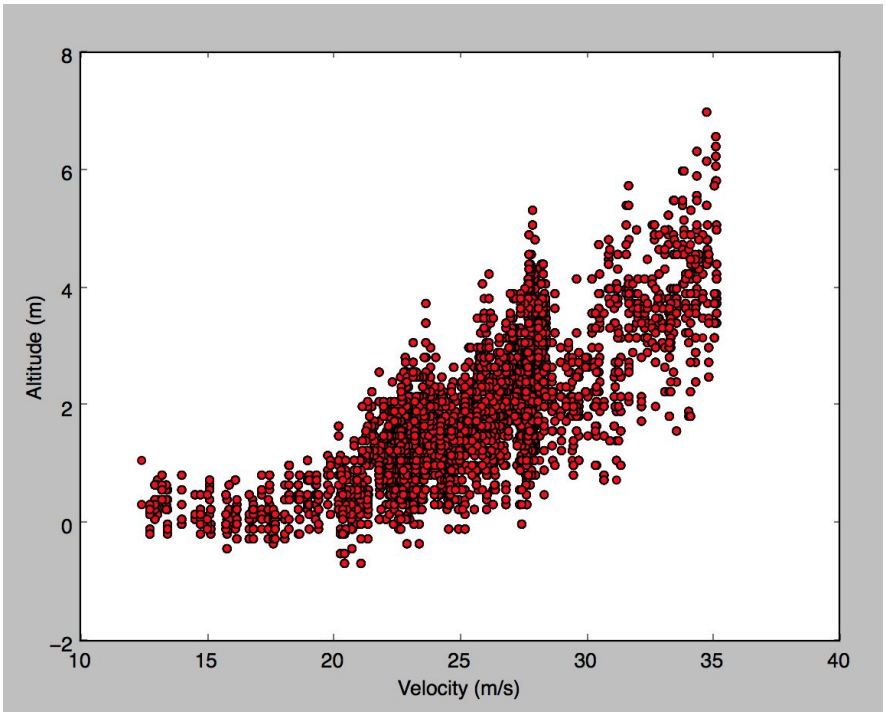
3.2 Sensors

As mentioned in section 2.1, the measurements necessary for apogee altitude prediction are altitude and velocity. These must be gathered directly by sensors or synthesized from indirect measurements.

One solution for measuring altitude is obvious - barometric altimeters are frequently used in model rockets for this purpose and, as previously mentioned, one was integrated onto the flight computer PCB. However, the use of the altimeter was not as simple as we had initially assumed.

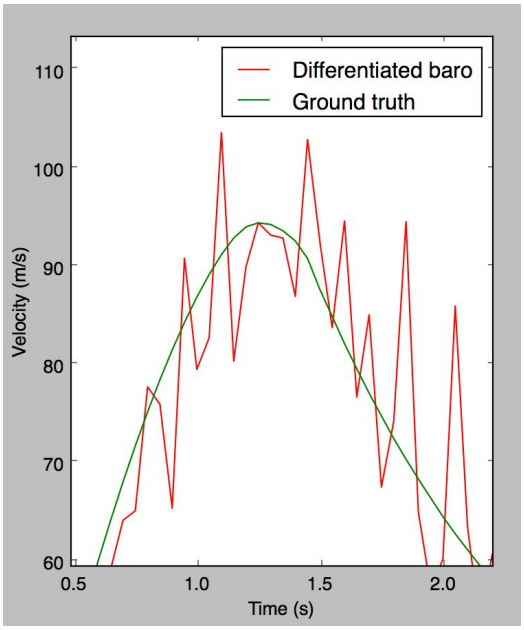
Upon review of flight data, a polynomial curve was fitted to the rocket's altitude, and it was noted that at some points during ascent the acceleration dropped significantly below the level of gravity (9.8 m/s^2), which is impossible once the motor has stopped firing. We hypothesized that a Venturi effect was

depressurizing the chamber beyond the true static pressure for that altitude. This effect is proportional to velocity squared, meaning that the sensor is accurate at apogee but not while the rocket is moving. In order to test this theory, the rocket was placed on a pole (far enough away to be outside the vehicle's boundary layer) and stuck outside a vehicle as it accelerated to 80 mph on flat ground. The results from that test appear below:



The altimeter reported an increase in altitude by almost 8 meters, despite the true altitude remaining constant. A polynomial curve was fitted to the test data to create a correction function to compensate for this effect in flight. The successfulness of this correction is discussed in section 5.

The source of velocity data was much less obvious. Several options were considered and rejected, including pitot tubes, optical flow, and Doppler radar. The first option seriously considered was differentiated barometric altitude, but, according to simulation, the sensor was too noisy to produce a smooth velocity estimate. See graph below:



Next, we attempted to use GPS, which is theoretically capable of providing high-accuracy (± 0.1 m/s) estimates for velocity using the Doppler shift on the carrier signal. Unfortunately, virtually all consumer GPS units are rated to a maximum 4Gs of acceleration. When deployed on our rocket, it often provided erroneous and erratic results, sometimes reporting physically non-plausible data, as shown by the velocity graph below (covering the entire ascent of the rocket):

Finally, we decided to eliminate a velocity sensor altogether. Instead, we estimated velocity by using measured acceleration and altitude as inputs to a 1-D Kalman filter. (We initially attempted to use PX4's built-in Kalman filter, but had severely inaccurate results). With state variables of position, velocity, and acceleration, the state transition matrix was as follows:

$$F = \begin{bmatrix} 1 & dt & \frac{1}{2}dt^2 \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix}$$

Other matrices and constants, such as the process, measurement, and estimate covariances, were tuned manually. The effectiveness of this method for generating velocity is discussed in section 5.

3.3 Drag Model

The aerodynamic drag of the rocket was assumed to be of the form:

$$D_T = D_F + D_B \, sin^2(\theta)$$

where

D_T is the rocket's total drag
D_F is the rocket's fixed drag
D_B is the brakes' maximum drag
θ is the angle of the drag brakes from vertical

Experimental support for such a sin^2 proportionality is found in Sheldahl & Klimas' 1981 Sandia National Laboratories report [6].

If D_B is assumed to follow the drag equation, we can calculate the minimum torque necessary for any drag brake servos, given the drag characteristics and pivot distance.

3.4 Drag Brakes

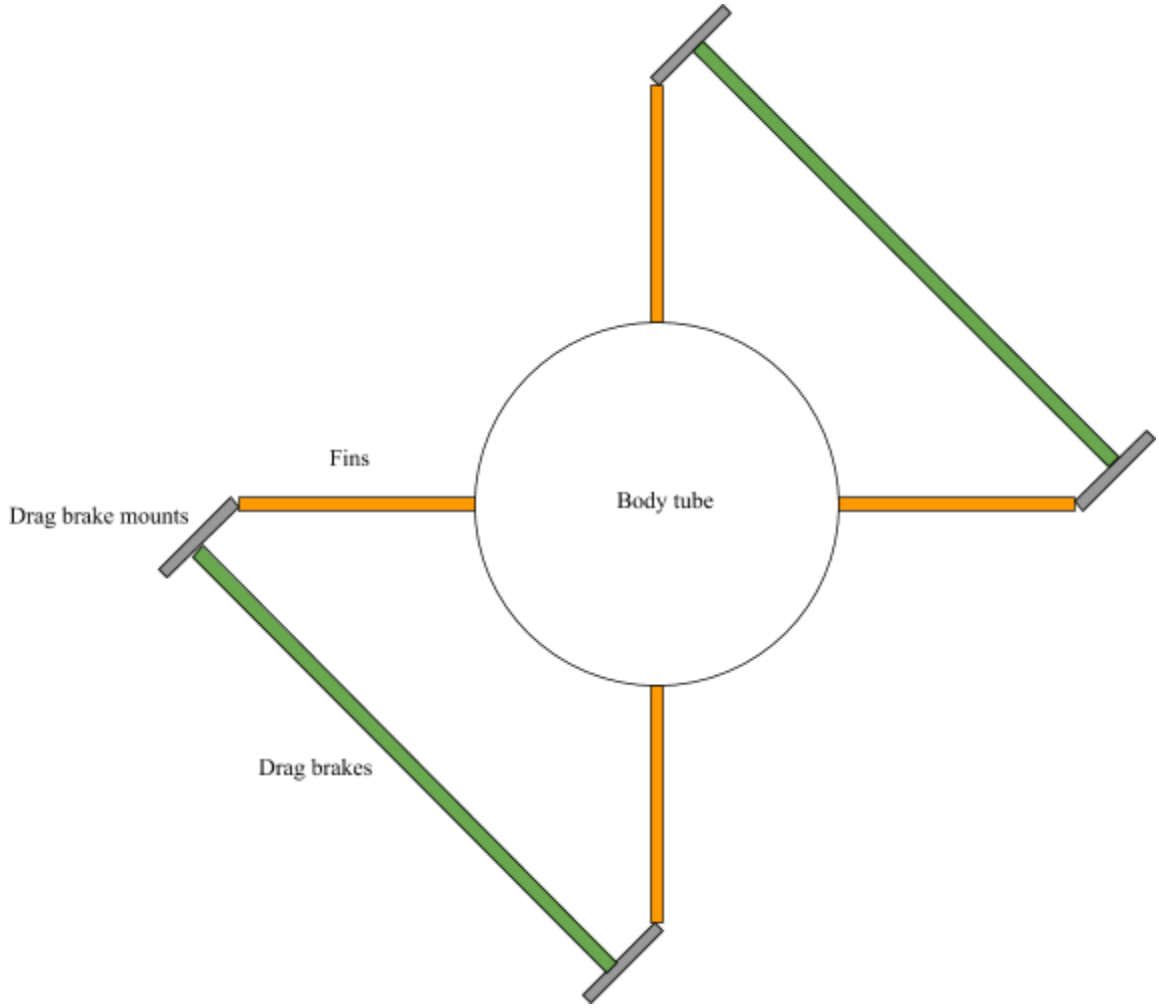
Although avoiding overly implementation-specific details, it is important to discuss the design of the drag brakes given how essential they are to the rocket.

As discussed in the theory section, we knew our drag brakes would fundamentally consist of some number of fixed plates extended into the airstream at a controllable angle. The difficulty lies in creating an efficient, lightweight system to do this given the confined dimensions of a rocket.

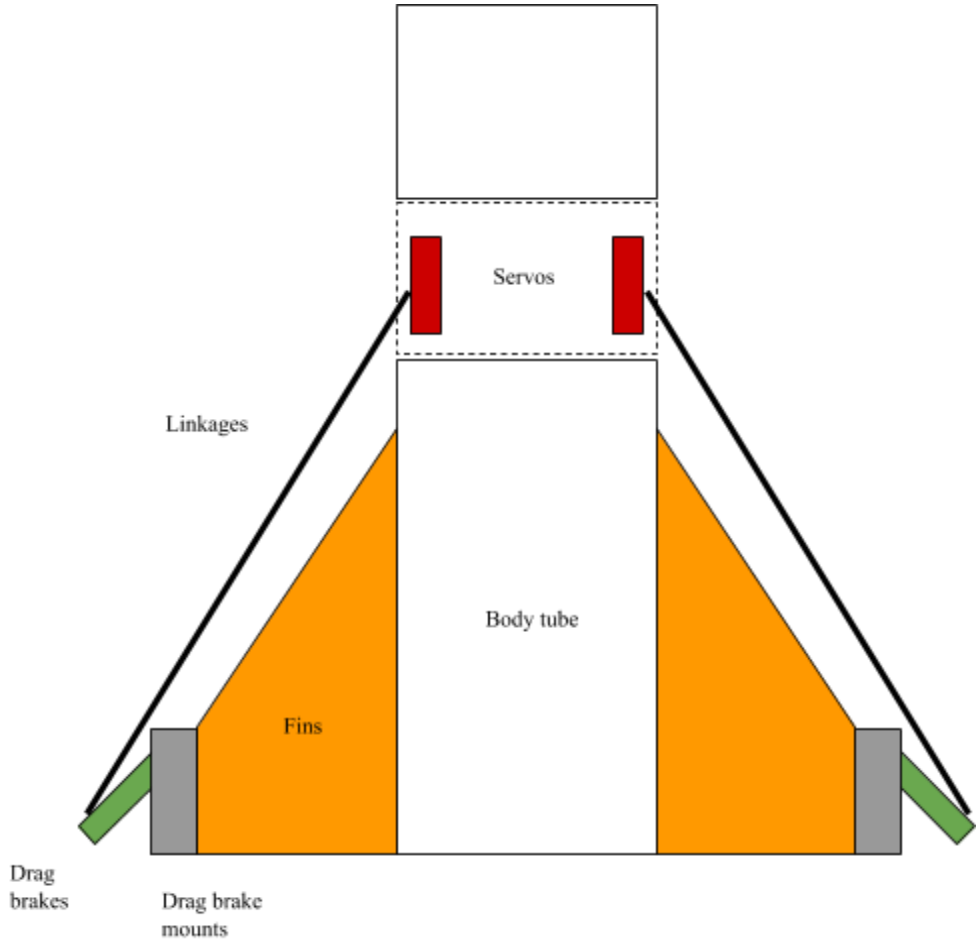
First, the issue of what the shape of the drag brake would actually be: because the body of a model rocket is cylindrical, the shape of the drag brakes must either be semi-cylindrical or stick outside of the rocket. It was decided that the structural sturdiness and high maximum drag of a flat plate outweighed the low minimum drag of a semi-cylinder.

Second, after deciding that the brakes would be planar, it was necessary to determine how to mount such flat plates to the rocket's cylindrical surface. The solution chosen was to mount the brakes in between the fins, pivoting about holes through the fin tips. Diagrams follow (with matching color-codes):

Top view:



Side view:



3.5 Budget

Expenses for 2016-2017:

Description	Cost
2x motor casings	$140.38
SuperLube, Pnut altimeter	$76.62
3x F motor refills	$32.39
Apogee (rocket parts)	$127.30
GPS receiver	$34.99
2x F40 motor refills, motor wrench	$48.57
F39-9T (3-Pak)	$32.39
Fruity Chutes	$49.05

PixRacer + case	$134.00
Stratologger altimeter	$53.86
Apogee (rocket parts)	$37.84
Apogee (rocket parts)	$84.15
Fruity Chutes	$49.25
Bay Area Rocketry - Engines	$44.26
Apogee (rocket parts)	$77.68
Apogee (rocket parts)	$78.31
Apogee (rocket parts) - refund unused shipping	-$24.43
Bay Area Rocketry - Engines	$145.03
Bay Area Rocketry - Engines	$120.87
Apogee (rocket parts)	$14.26
F52 reloads (wildman hobbies	$228.17
Aloft Hobbies Servos etc	$70.00
Apogee Components	$90.00
Total	$1744.94

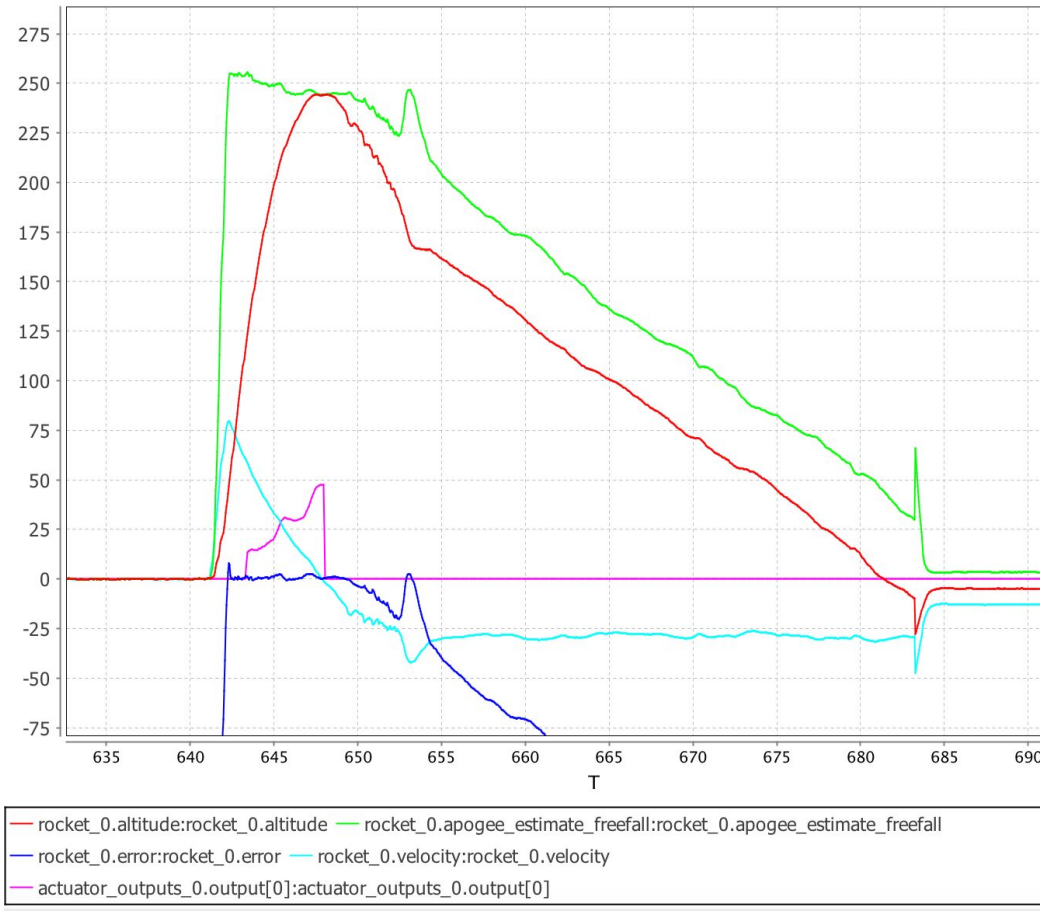
Expenses for 2017-2018:

Description	Cost
Animal Motor Works (2x fly-away rail guides)	$113.00
Apogee Order #96945 (airframe parts & engines)	$277.59
Aloft Hobbies #102050002 (3x servos)	$129.73
Fruity Chutes	$55.26
Apogee Order #97826 (airframe parts)	$22.07
Amazon #114-9292000-3073007 (epoxy, mixing nozzles)	$46.35
J&M Hobby (wood, brass rods, adhesives, etc.)	$102.89
Bay Area Rocketry (engines)	$159.66
Apogee Order #99513 (egg protector, nose cone, rail buttons)	$36.89
Amazon #114-3429376-0049061(adhesives)	$17.61
Additive Aero (2x fly-away rail guides)	$105.89
HobbyKing (adhesives)	$19.39
Performance Hobbies (Engines for Nationals)	$47.97
Total	$1134.3

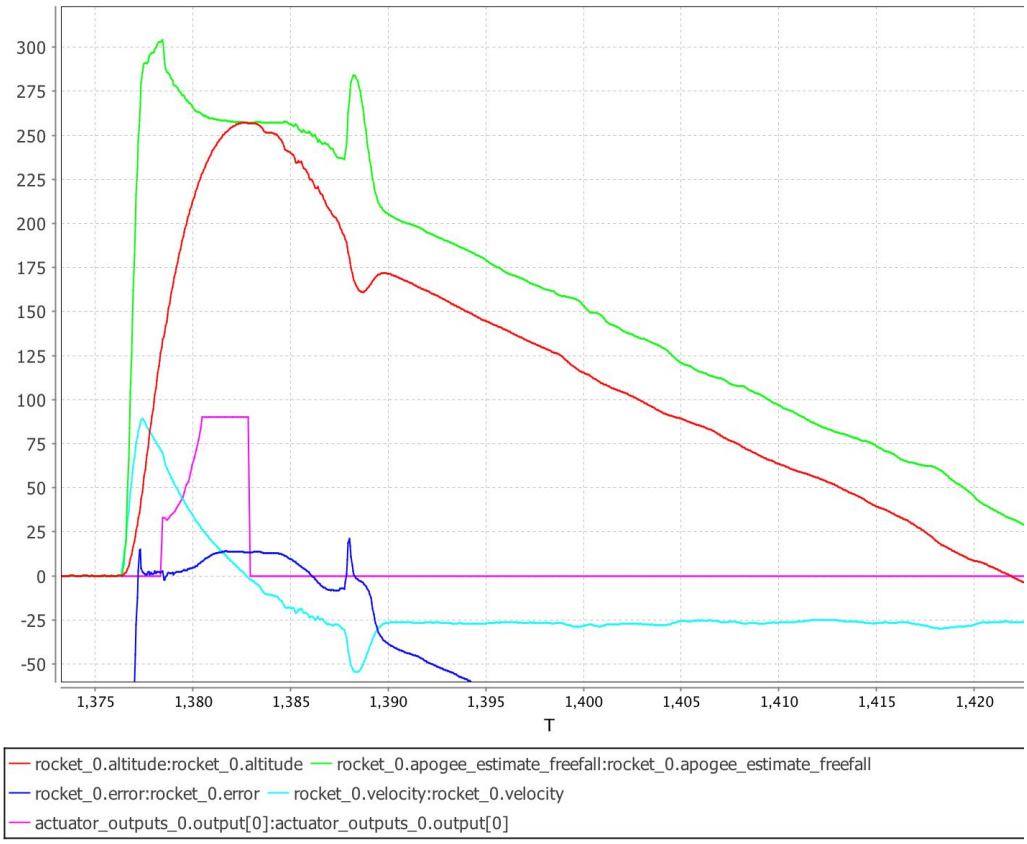
For two actively-controlled rockets, with a combined 50+ F-motor-powered flights, this project was relatively inexpensive. The use of commercial, off-the-shelf components allowed for relatively low costs despite the fairly complex rockets, and the use of software-controlled parachute deployment (as opposed to via ejection charge) allowed for higher reliability. Some of the fixed costs necessary for the project, such as avionics and altimeters, were expended in the first year, allowing for cheaper construction in the second year.

4. Results

Towards the end of development, there were multiple highly-performing flights, with altitude errors at or near zero, and the entire flight within the brakes' control range. For example, this flight:



However, the performance was not consistent. Other flights had much more mediocre performance:



This flight flew 43 feet too high, and the brakes exited the control range approximately two seconds before apogee.

The difficulty of launching at a high frequency and limited availability of test launches prevented acquiring a statistically significant number of test flights and control flights. As such, the results are not considered definitive, but rather a point for further investigation and continuation.

5. Conclusion

There is a large difficulty in identifying the actual issues causing performance problems. The lack of ground truth sensors and drag characteristics makes it difficult to spot where error is introduced.

The altimeter Venturi effects mentioned in section 3.2 have called into question the accuracy of the altimeter before apogee, leaving only the accelerometer as a trusted sensor, which is insufficient for most validation purposes. With no direct sensor measuring velocity, it is very hard to determine whether the filter-synthesized velocity is accurate.

It is also unknown if the drag model for the rocket is accurate. The sin^2 assumption is based on findings for a tilting flat plate in isolation, and does not cover the complex drag interaction effects that could result

in a drag brake system with more complex geometry. Estimates for the drag of the vehicle, which were based on flight data, varied by up to ±20%.

Several solutions could potentially clear up this uncertainty. The velocity could be found by simply adding an existing high-dynamics GPS that can handle large, sustained accelerations; Unfortunately, such GPS units are prohibitively expensive, typically costing upwards of a thousand dollars. The drag could be found via wind tunnel testing, but again having an issue with cost.

Other, cheaper solutions could include using a ground-based transmitter to provide Doppler velocity, adding a pitot tube for airspeed velocity, or drop testing for drag data.

Future work could potentially implement such solutions and produce more consistent and significant results.

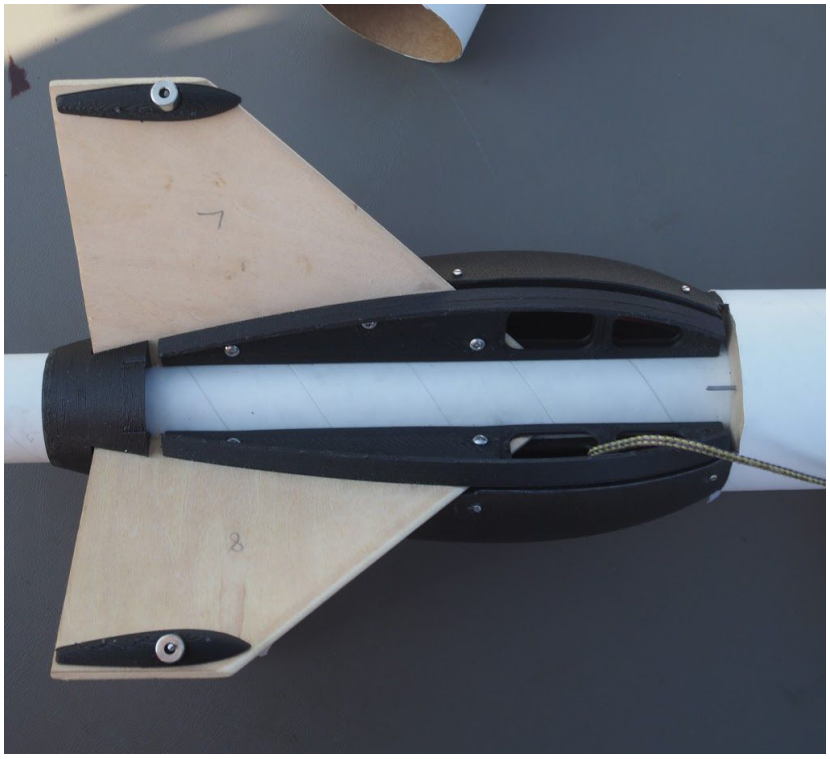
6. References

1. Konshak, M., “Calibrating a Model Rocket for Competitive Precision Altitude Events using Rocksim 9.” National Association of Rocketry, Marion, IA; August, 2009. https://nar.org/member-only-reports/rd/NARAM_51_Reports_TCR22/Calibrating%20Rocket%20for%20Precision%20Altitude%20(M.Konshak,%20NARAM%2051,%202009).pdf
2. Schultz, D., “Application of the Kalman Filter to Rocket Apogee Detection,” National Association of Rocketry, Marion, IA; August, 2004. https://nar.org/member-only-reports/rd/NARAM_46_Reports_TCR17/Kalman%20Filter%20for%20Altimeters%20(D.%20Schultz,%20NARAM%2046,%202004).pdf
3. Schultz, D., “Barometric Apogee Detection Using the Kalman Filter,” National Association of Rocketry, Marion, IA; August, 2002. https://nar.org/member-only-reports/rd/NARAM_44_Reports_TCR16/Kalman%20Filter%20for%20Apogee%20Detection%20(D.%20Schultz,%20NARAM%2044,%202002).pdf
4. “A Work in Progress: Active Stabilization Flight Computer” National Association of Rocketry, Guzek, B; August, 2009. https://nar.org/member-only-reports/rd/NARAM_51_Reports_TCR22/Active%20Stabilization%20Flight%20Computer%20(B.Guzek,%20NARAM%2051,%202009).pdf
5. Pixracer. (n.d.). Retrieved July 20, 2018, from https://docs.px4.io/en/flight_controller/pixracer.html
6. Sheldahl, R. E., & Klimas, P. C. (1981). *Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines*. Albuquerque, NM: Sandia National Laboratories.

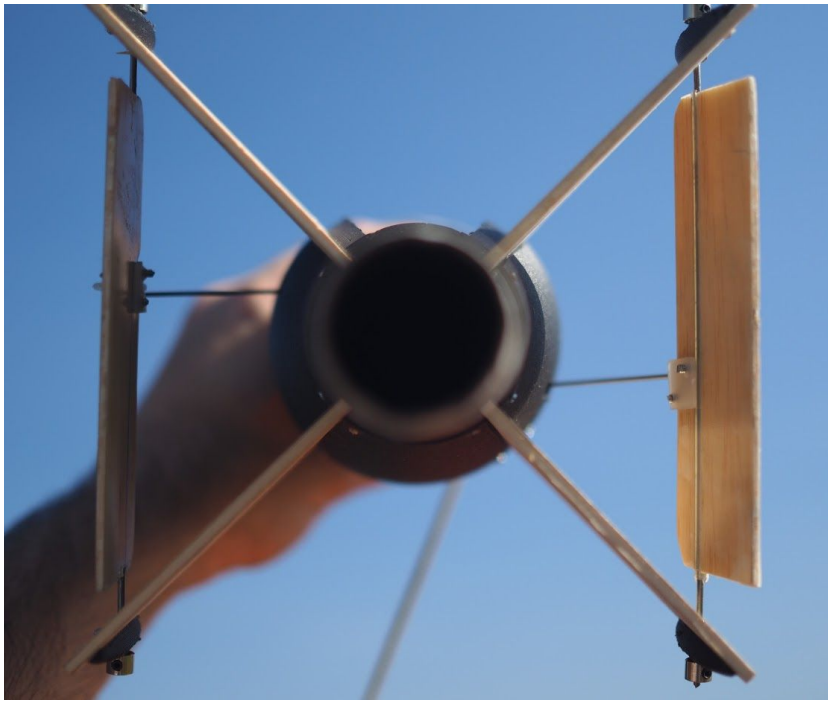
Appendix A (Photos)

Rocket on pad, in flight configuration (orange is pop-lug rail):

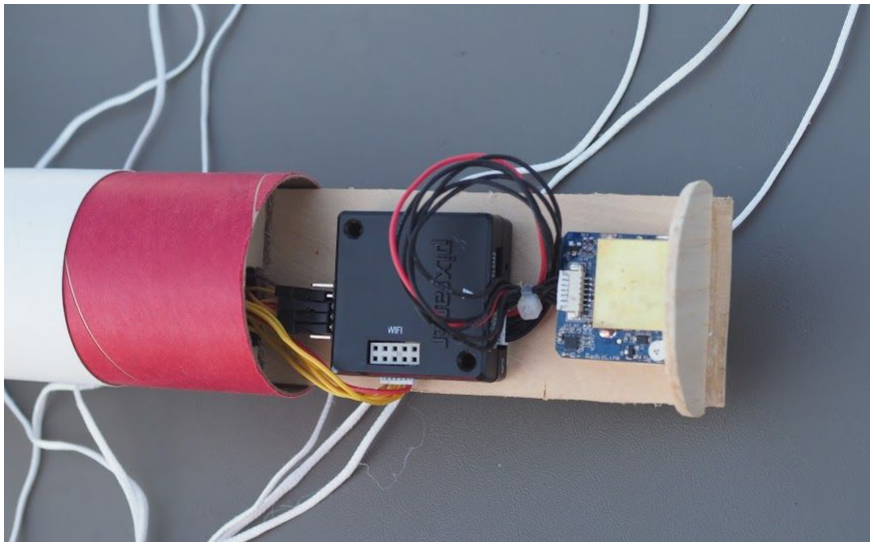
Lower section (parachute compartment opened):



Bottom view of rocket:



Avionics bay (with GPS on right, no longer used)::



Appendix B (Software)

Preliminary Python simulator:
https://github.com/zfurman56/rocket-simulator

Flight software (C/C++):
https://github.com/zfurman56/Firmware

SIL/HIL simulator (Java):
https://github.com/zfurman56/jMAVSim

This R&D Report provided as a membership bonus for joining the National Association of Rocketry at http://nar.org

Check out the other membership bonuses at http://nar.org/members/

Thank you for joining the National Association of Rocketry!