2016-06

Missile demonstrator for counter UAV applications

Rydalch, Fletcher D.

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/55204

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

MISSILE DEMONSTRATOR FOR COUNTER UAV APPLICATIONS

by

Fletcher D. Rydalch

June 2016

Thesis Advisor:	Christopher M. Brophy
Second Reader:	David Dausen

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2016	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE MISSILE DEMONSTRATOR FOR COUNTER UAV APPLICATIONS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Fletcher D. Rydalch	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

An autonomously guided rocket-powered delivery vehicle has been under development at the Naval Postgraduate School. Designed to eventually counter UAV swarm attacks, the vehicle made advances toward reaching a target in the sky. These advances reduced the time needed to launch, modify, and relaunch the rocket, while adding capabilities such as data transfer along the vehicle axis and the rapid download of flight data. Improving the vehicle included reconfiguring the guidance, navigation, and control (GNC) strategy. Advancements included the design, implementation, and evaluation of electronic servo control, actuating fins, and the mechanical coupling design. The forward compartment in the vehicle's nose cone was structurally modified for the GNC equipment and to support electronics under high-g launch conditions. Modifications included innovative designs for managing heat transfer requirements. Using off-the-shelf subsystem components kept the advancements fiscally mindful.

After implementing the design features, two final test launches were performed: one demonstrated a control spin rate of 8.5 rad/sec; the other showed the vehicle's ability to execute pitch maneuvers on a single axis. The test results can be used to improve the GNC software and servo control parameters. Continued development will allow the system to become a viable option for countering UAV swarms.

14. SUBJECT TERMS unmanned aerial vehicle (UAV), rocket, counter-swarm	**15. NUMBER OF PAGES** 163
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

MISSILE DEMONSTRATOR FOR COUNTER UAV APPLICATIONS

Fletcher D. Rydalch
Ensign, United States Navy
B.S., United States Naval Academy, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2016

Approved by: Christopher M. Brophy
Thesis Advisor

David Dausen
Second Reader

Garth Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

An autonomously guided rocket-powered delivery vehicle has been under development at the Naval Postgraduate School. Designed to eventually counter UAV swarm attacks, the vehicle made advances toward reaching a target in the sky. These advances reduced the time needed to launch, modify, and relaunch the rocket, while adding capabilities such as data transfer along the vehicle axis and the rapid download of flight data. Improving the vehicle included reconfiguring the guidance, navigation, and control (GNC) strategy. Advancements included the design, implementation, and evaluation of electronic servo control, actuating fins, and the mechanical coupling design. The forward compartment in the vehicle's nose cone was structurally modified for the GNC equipment and to support electronics under high-g launch conditions. Modifications included innovative designs for managing heat transfer requirements. Using off-the-shelf subsystem components kept the advancements fiscally mindful.

After implementing the design features, two final test launches were performed: one demonstrated a control spin rate of 8.5 rad/sec; the other showed the vehicle's ability to execute pitch maneuvers on a single axis. The test results can be used to improve the GNC software and servo control parameters. Continued development will allow the system to become a viable option for countering UAV swarms.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AMRDEC	U.S. Army Aviation and Missile Research, Development, and Engineering Center
AoA	angle of attack
CUAS	counter unmanned aerial system
DAQ	data acquisition board
DOF	degree of freedom
EKF/AKF	extended or adaptive Kalman filter
EO	electro optical
FCC	flight control computer
GNC	guidance, navigation, and control
GPS/INS	global positioning system/inertial navigation system
GUI	graphic user interface
IMU	inertial measurement unit
IPT	integrated product teams
MES	Marine Expeditionary Security
NED	north east down
PM	plant model
PWM	pulse width modulated
RFI	request for information
TTPs	tactic, techniques, and procedures
UAV	unmanned aerial vehicle
WRC	wireless remote control
ZUPT	auto-zero update options

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all, I would like to thank my wife, Cassandra Sullivan, for putting up with long hours, weekends away, and conversations about rockets.

I would also like to thank the Naval Postgraduate School Rocket Lab support faculty members: Lee Van Houtte, Robert Wright, and Andrew Chaves. Their advice and skill made this work possible.

I would also like to thank ENS Brent Aldridge, USN, Wee Yeow Lim, Alejandro Garcia Aguilar, Maj. John Dirk, USA, and LT Travis Hartman, USN. They were members of the missile design course ME4704 that contributed designs, knowledge, and innovation.

I would also like to thank John Mobley, Naval Postgraduate School. He was the chief machinist for personally machined pieces used in development. His commitment for enabling research and testing cannot be overstated.

I would also like to thank Dave Dausen for his consultation on the direction of the project. He was able to keep the work headed in the right direction.

Finally, I would also like to thank Dr. Chris Brophy, my adviser, for his guidance and mentorship throughout the entire research process. His overall vision and ability to teach at the appropriate level allowed the work to truly flourish.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The role of unmanned aerial vehicles (UAVs) in warfare has increased over the past three decades. Originally, UAVs were developed to be capable of fulfilling multi-platform missions [1]. Their high cost in development and implementation created a barrier to entry. Over the past decade, the utilization of UAVs fulfilling multiple missions has changed and expanded.

A. UAV SWARM THREATS

Both countries and independent entities are now developing and deploying less expensive UAVs that are still capable of causing catastrophic damages [1]. The progression of UAV capabilities has brought the evolution of UAV swarms. Similar to swarms of animals, a swarm of UAVs allows singular inexpensive units to work in unison, creating a more capable whole.

1. Current Danger of a Swarm Attack

The evolution of UAVs and the expansion of their uses are driven by the commercial sector. An attacking swarm of UAVs is now more likely to be used [2] simply because UAVs are more efficient and cost-effective as swarms but still have the likelihood of delivering damaging attacks. UAV swarms are viewed as capable reconnaissance machines, and they are most capable in attacking and damaging aircraft carrier battle groups as well as land-based operations. In a study conducted by Loc Pham et al. at the Naval Postgraduate School [3], the counter UAV swarm defense systems of U.S. Navy ships proved lacking. In a simple scenario involving a Navy destroyer and small swarms of attacking drones with current defense systems, including the Aegis air defense system and machine guns, the study determined that on average, 2.8 out of 8 attacking drones were able to get by the defense systems [4]. As swarm numbers increased, the current defense systems were able to affect only the first seven UAVs, leaving open the door for UAV swarm attacks. The potential threat that UAV swarms pose has prompted the U.S. Navy to invest in the research and development of strategies and countermeasures against large-scale swarm attacks. Likewise, there has to be a push for the development of counter UAV swarm methods to improve defense abilities against swarms while attempting to diminish the cost of defense [4]. Phillip Finnegan, a UAV expert who consults on research for the U.S. Air Force, recently commented that the UAV is "an important area of research because it offers the potential to provide new ways of attacking an adversary at lower cost. It is also important to understand that an adversary might wish to use swarms against the United States—so this has an offensive and defensive character" [5]. As Finnegan alludes to potential threats from UAV swarms, news sources and reports have noticed other countries' growing interest in the development of deployable UAV swarms. As recently noted in the *U.S. News and World Report* [6], China is leading the charge in developing UAVs capable of attacking as swarms and causing noticeable damage. The Chinese government is encouraging aerospace technology companies to develop and test the capabilities of UAV swarm attacks [6].

The potential threat of UAV swarms has reached the upper echelons of strategy development. Portions of "A Cooperative Strategy for 21st Century Sea Power" make clear the need to develop methods to combat swarms of UAV attacks. The plan states the need to "evolve our counter-small boat swarm tactic, techniques, and procedures (TTPs) to include the use of innovative technologies such as lasers, advanced guns, and remotely piloted 'smart' vehicles to counter this threat" [7]. In "Sea Power 21: Projecting Decisive Joint Capabilities," Admiral Vern Clark, USN, expands on the concept of swarm drone attacks by discussing the potential for increased attacks [8]. Because of the need to improve drone countermeasures, military units have been given direction to counter threats. Marine Expeditionary Security (MES) is one such organization. MES is supposed to assist sea- based operations by providing protection for the transit of supplies and assets needed to continue operation. MES must "detect, identify, engage, and destroy Level I, and Level II hostile air, surface, subsurface, and ground targets, day and night, and in most weather conditions in the littoral battle space" [9]. Developing a smart rocket capable of delivering smart submunitions to a specific location could fulfill the request for support.

The Navy is not the only military branch focusing on developing counter UAV swarm platforms. The U.S. Army also has made this subject a priority and asked for information and research on the subject in a recent request for information (RFI) [10]. In it, the U.S. Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC) called for information for advancing an "affordable Counter Unmanned Aerial System (CUAS) material solution" [10]. The request was for "technical data describing the proposed concept and innovative methodology information to assist in programmatic planning" [10].

2. Method for Countering UAV Swarms

Multiple methods of how to counter UAV swarms have been proposed. Current countermeasures include small arms defense or shipboard systems such as the AEGIS, but these are not appropriately scaled for the multitude of targets. The advent of less expensive UAVs has exposed the need for new countermeasures. The vehicle under development is intended to bridge the cost disparity between current countermeasures, which range up to hundreds of thousands of dollars, to capable systems produced for thousands of dollars. The vehicle described for this effort is intended to apply this counter-swarm method. Current counter methods to UAVs do not include a delivery of counter submunitions for attacking and countering a UAV swarm. The UAV delivery vehicle is rocket-propelled and equipped with active guidance and seeker subsystems; it is designed to operate at a fraction of the cost of current counter methods. Building the vehicle with off-the-shelf materials reduces the cost drastically. The method applied in the development of the missile demonstrator is shown in Figure 1. Due to the inclusion of active guidance and seeker subsystems in the delivery vehicle, the terms "missile" and "vehicle" are used interchangeably.

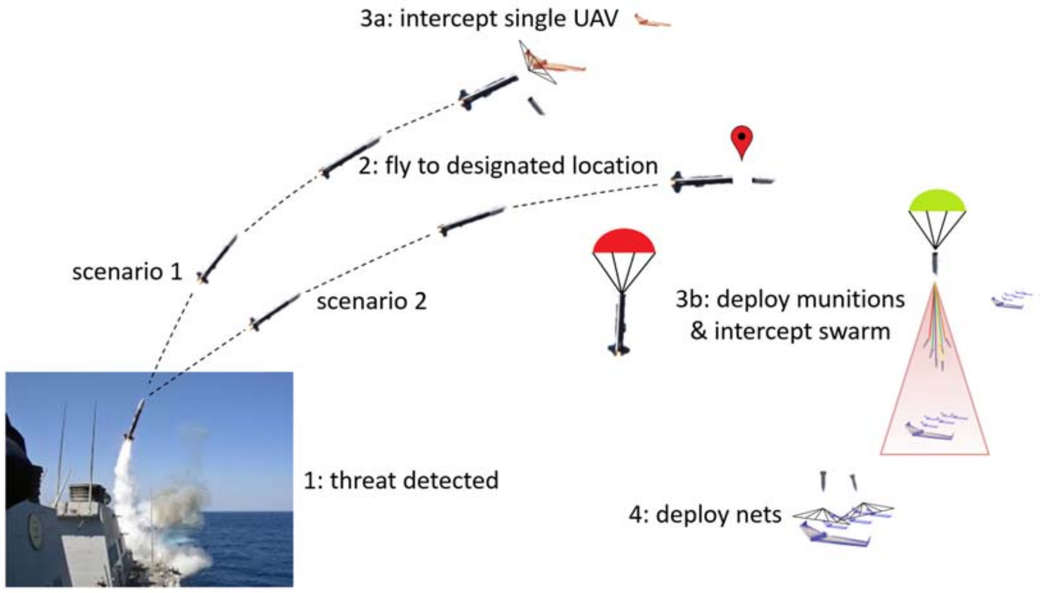


Figure 1. Possible Mission Scenarios of the Vehicle in Counter UAV Swarm Applications

The method under development consists of multi-step processes; the eventual goal is to disable or destroy an approaching swarm of inexpensive, autonomous vehicles. The delivery vehicle is being designed for two scenarios. Scenario 1 is intended to eliminate a single delivery vehicle. Since the independent operable distance of autonomous vehicles is limited, disabling a delivering vehicle would be an effective way to stop a swarm attack. Scenario 1 includes the following steps:

1. The launch of the missile is the first step in activating the counter UAV system. The design of the rocket considers multiple launching platforms from either ships or land.
2. Guidance, navigation, and control (GNC) would be activated within the vehicle, providing commands to change flight trajectory. Guidance for this scenario is intended to directly intercept a single vehicle at high speeds.
3. Closing in and eliminating a single system is the final stage. The single system could include a "mother ship" type of vehicle. Therefore, the intended use of the vehicle would be to disable the UAV prior to swarm deployment.

Scenario 2 is intended to disable an oncoming swarm that is away from its location of origin. The origin of the swarm is independent from how intercepting the swarm occurs. Scenario 2 includes the following steps:

1. The missile is launched.
2. Guidance within the vehicle is activated with the intent to aim the missile to a point above the approaching UAVs.
3. The rocket reaches the area of interest in a controlled flight (regardless of the guidance method).
4. The rocket separates (about 2000 ft. above the swarm of UAVs) and assumes a controlled descent under a parachute canopy.
5. The controlled descent of the nose cone allows the vehicle to use image tracking to locate each vehicle in the swarm. The computer in the nose cone then communicates with submunitions also located in the descending nose cone section of the rocket.
6. Once each of the UAVs is located, commands are given to individual submunitions to deploy attacking the UAVs.
7. Impact of the submunitions with oncoming UAVs is substantial enough to disable the oncoming UAV, eliminating the swarm by a single UAV at a time.

The development of the delivery vehicle is essential for the application and testing of the submunitions. This thesis focuses on the development of a vehicle with the potential to deliver the submunitions. This test vehicle is not currently being designed for a particular set of submunitions but rather to test methods and requirements that may eventually support and complement the submunition approach.

B. HISTORICAL DEVELOPMENT

The ME4704 Missile Design course is a one quarter intensive module at the Naval Postgraduate School (NPS) for students to design a vehicle or missile to meet a specific objective. Beginning in 2012, the focus shifted to developing a vehicle to implement design concepts from the classroom into an actual flight system. Originally, the plan was to research and produce a rocket-powered vehicle that could deliver a particular payload to a specific location for minimal cost. The theory was that the capability of delivering a payload could be used, and this theory proved correct. Over time, more exact applications have been uncovered; one of the forefront applications is that of a counter-swarm UAV delivery vehicle. Over the past year, the work on the vehicle has been aligned. The Gantt chart of the general progress is shown in Figure 2. This portion of the research is considered Phase I development.

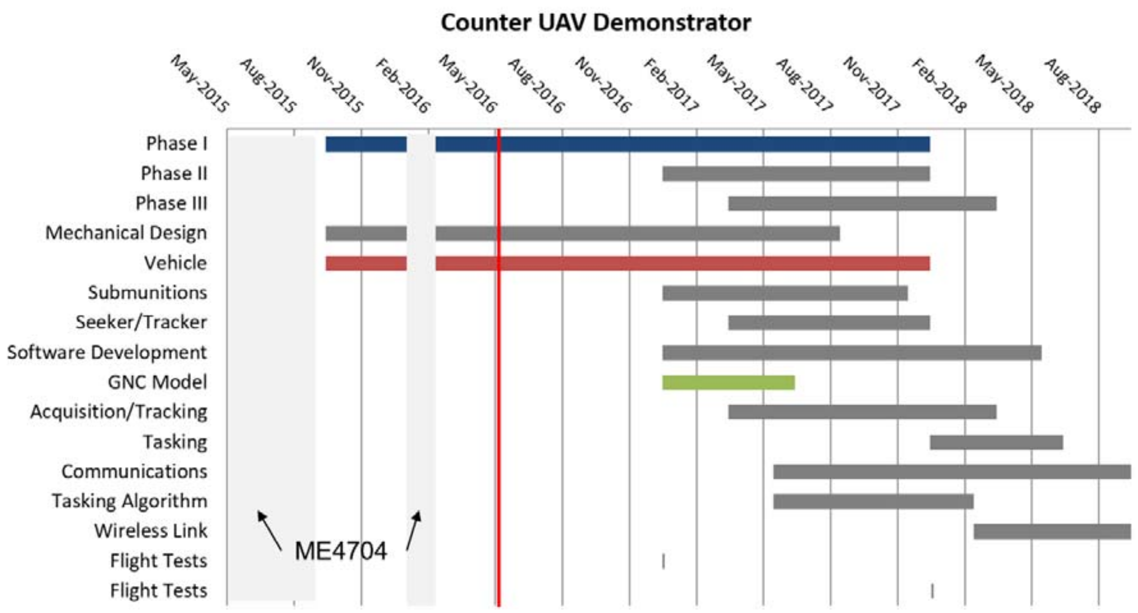


Figure 2. Counter UAV Phase Development

The chart shows the association of Phase I development with mechanical design and delivery vehicle development. This research and thesis is focused on the implementation of proposed concepts and design modifications of the test vehicle. Although the mechanical design and delivery vehicle development is included Phase I, it will continue to be revisited and improved through the end of the project and actual implementation. Follow-on work will target the improvement of the vehicle's guidance and control system. Once successful, the rocket will be able to include the implementation of submunitions. Those submunitions and individual UAV interceptors need to be developed and will be prepared in Phase II for inclusion in Phase III. Phase III will be the implementation of the submunitions proving the capability of the counter-swarm vehicle. Mindful of the overall effort, this research and development exclusively focused on the rocket-powered vehicle. Successful work on the vehicle includes working with off-the-shelf materials and mitigating risks of failure during testing.

Over the past four years, collaborated work has produced a rocket that can launch over 1500 m but is not capable of being controlled. Thought and innovation had been put into designs for control, but these designs were not completely implemented. This thesis intended to advance the rocket toward the end goal of having an operable delivery vehicle. A single thesis project minimizes the challenges that come with class turnover and accompanying documentation of work done. Professed intentions of image-guided flight and control were theorized but never actually installed on the vehicle and tested. At the onset of this thesis research and development, the project was evaluated and reconsidered.

Establishing bench points and mapping the direction of the research was the first step. The vehicle would be developed in separate phases. This research is mostly confined to Phase I of development.

Development of the vehicle began with a review of high-power rocketry. The basic schematic of a model rocket's flight is shown in Figure 3. Included in the figure are annotations that show different stages.

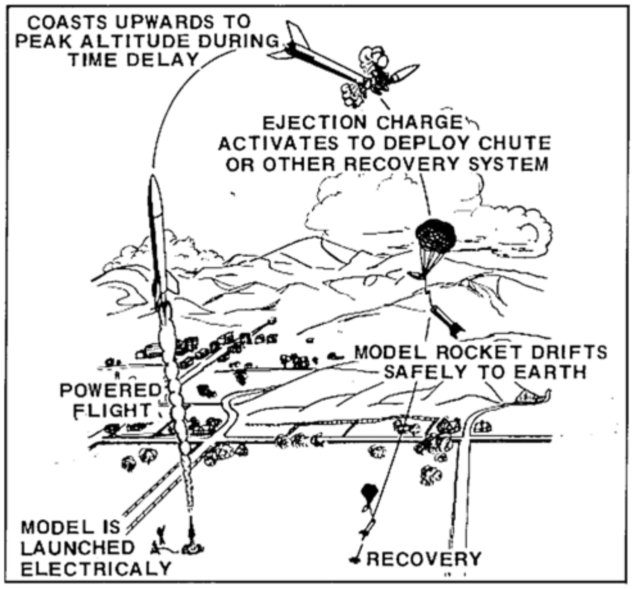

COASTS UPWARDS TO PEAK ALTITUDE DURING TIME DELAY
EJECTION CHARGE ACTIVATES TO DEPLOY CHUTE OR OTHER RECOVERY SYSTEM
MODEL ROCKET DRIFTS SAFELY TO EARTH
POWERED FLIGHT
MODEL IS LAUNCHED ELECTRICALY
RECOVERY

Figure 3. Simplified Model Rocket Launch and Recovery. Source [11]

Amateur and model rocketry has been underway since the 1950s [12]. The advent of rocketry contributed to the independent development of small rockets and the growth in understanding and the popularity of the subject. Model rocketeers continued to improve their skills, and many eventually became working professionals. This project uses many principles that were developed through amateur rocketry but diverges in the advancement being made by using off-the-shelf materials and equipment. The goal is to use that technology and advancement to produce a more capable vehicle at a cost significantly below that of conventionally fielded systems, such as AEGIS.

1. The Product from Past Work

The onset of research and development presented a rocket comprising three nominal compartments. To some extent, each component received individual attention. Past efforts created a rocket with preallocated space for new designs. The compartments are annotated in Figure 4.

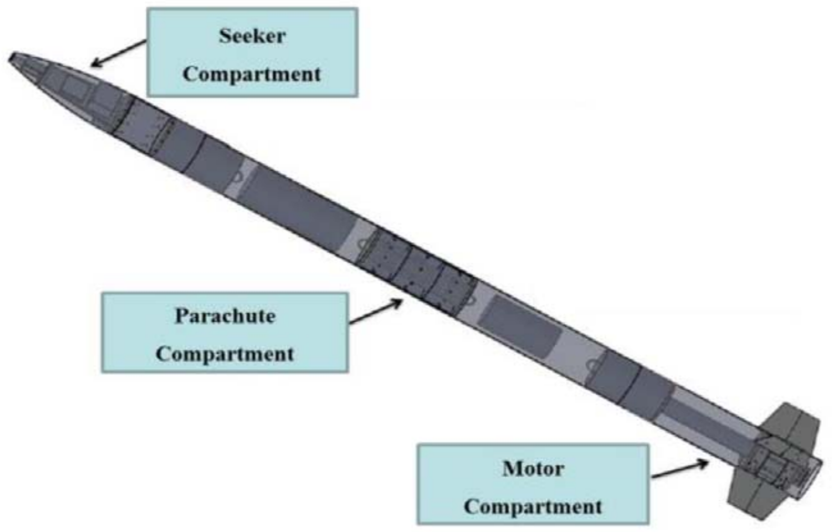


Figure 4. Basic Solidworks of the Previous Design. Source [13]

The former rocket [13] was 3.66 m (12 ft.) tall with a fuselage diameter of 19.69 cm (7.75 in.) The entire system weighed 29.9 kg (66 lbs.) and was propelled by a Cesaroni M1300 rocket motor (see the thrust profile in Appendix A). The rocket was capable of reaching 1524 m (5000 ft.) when launched vertically. The delivery vehicle was built using a redundant parachute deployment system to ensure recoverability and was successfully launched and recovered using a single parachute. The previous version could effectively launch once in a single day. The final product of past work included a rocket launching with fixed fins that was capable of reaching Mach 0.8 off the rail. The parachute deployment techniques used electronic equipment and provided some autonomous function within the vehicle.

a. Seeker and Control Components

At the onset of research and development, the fins were fixed. Servos were planned for but had not been evaluated, tested, or installed. Because of the inability to actuate the fins, no guidance or control of the system was attempted. Further research and development using input from past work and designs was needed to develop a new rocket, test its performance and operating boundaries, and evaluate how the rocket could be used in combating swarms of UAVs. The seeker components were designed to fit within the nose cone of the rocket with aft-located controllable fins on the outside of the motor casing, as shown in Figure 4. Included in the seeker compartment development was the GNC hardware.

Guidance and control investigation in previous work included research into the communications and fin development. That research focused on image processing for guidance and mid-fin actuation for control implementation. Prior work prioritized the hierarchy of needs and areas to consider when developing the rocket. Following are the main design requirements that govern the selection of the servos developed by earlier research [13].

- Missile to fly at Mach 0.8
- Fins to deflect to a maximum of 3 degrees (to limit torque and reduce stall potential)
- Weight of servos to not exceed 2.27 kg (5 lbs.) each
- Position of servos to be as forward as possible to improve flight stability authority
- Worst case situation to be taken as straight and level flight of missile
- Signals from servos must be input from PC/104

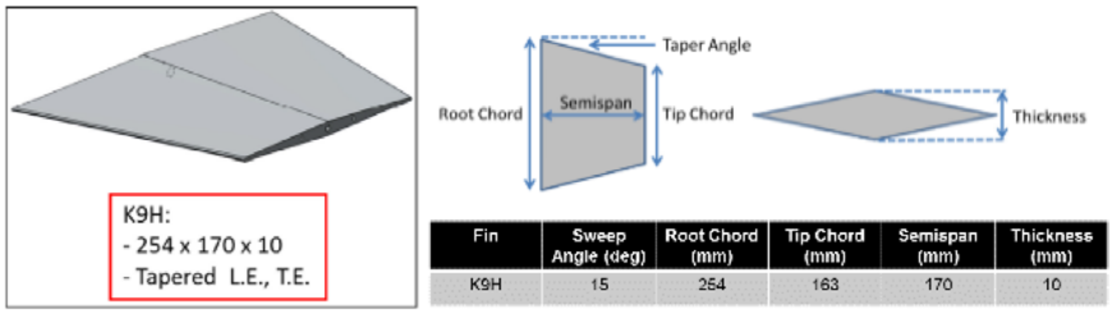
- Servos to operate on DC voltages
- Design to be scalable for future teams
- Future upgrade of rocket motor possible

Previous work was able to correctly size the torque requirements needed from the servos to possibly control the rocket. These calculations were used in initial designs. The calculations were made using base fin and airfoil theory:

$$C_L = \frac{L}{\frac{1}{2}pV^2S} = 2\pi\alpha$$

L = Lift Force (normally acting at quarter-chord, c/4, for subsonic speeds)
ρ = Atmospheric density
V = Velocity of missile
S = Reference Area
α = Angle of Attack (AOA)

The existing fins were analyzed because they were present, and a determination had to be made of whether the fins could continue to be of use. These fins and dimensions are shown in Figure 5.



Fin	Sweep Angle (deg)	Root Chord (mm)	Tip Chord (mm)	Semispan (mm)	Thickness (mm)
K9H	15	254	163	170	10

Figure 5. Prior Designed Fin Dimensions. Source: [13]

The lift force (L) and Torque (T) acting on a fin were calculated using thin airfoil theory. Calculations used a 3° angle of attack (AoA) fin deflection at Mach 0.8 (264 m/s). The calculations are as follows:

$$C_L = \frac{L}{\frac{1}{2}\rho V^2S} = 2\pi\alpha$$

$$L = \frac{1}{2}\rho V^2 S \times 2\pi\alpha = \rho\pi\alpha V^2 S$$

$$= 1.225 \times \pi \times 264^2 \times 0.05236 \times 0.03544$$

$$= 497.79\text{N (112 lbf)}$$

$$Torque, T = 497.79 \times \frac{c}{4}$$

$$= 497.79 \times \frac{0.254}{4}$$

$$= 31.6\text{ N-m (23.3 ft-lbf)}$$

These calculations were carried out for differing fin deflections. The results of the calculations for each fin level are found in Table 1.

Table 1. Lift Force and Torques for Different Angles of Fin Deflection

Angle of Attack $Degree$ (°)	Mach No.	Lift N	Lift lbf	Torque N-m	Torque ft-lbf
3	0.8	498	112	31.6	23.3
5	0.8	830	187	52.7	42.2
7	0.8	1162	261	73.8	54.4

Investigation and determination of the needed servos and accompanying hardware was needed to account for the amount of torque. These requirements were taken into account for the prior group's designs and proposed solutions. However, advancements were never made on those designs. Recommendations were applied in the development of servo design and fin control in the new version.

b. Parachute Compartment

This compartment consisted of two parachutes with a triple redundancy system for deployment. The three-deployment system used is the Raven3 altimeter, the MARSA4 altimeter, and the Wireless Remote Control (WRC+) for manual deployment during emergencies. The setup of the rocket included two separate parachute bays.

c. Motor Compartment

The motor compartment consisted of the motor casing, a Cesaroni M1300 motor, and space for eventual fin control hardware. With an average of 1303 N (293 lbf.) of thrust, the missile could launch straight up and reach nearly 1524 m (5000 ft.). Once the launch took place, the only potential control from the launch site was through the WRC+, and that was only for redundancy on parachute deployment.

II. FLIGHT SYSTEM IMPROVEMENT

The rocket powered launch vehicle was improved during the 2015 and 2016 calendar years. Broad range improvements were intended to improve structural integrity, guidance, mission fulfillment, recoverability, control, modularity, and repeatability. While attempting these improvements, the approaches were made using off the shelf or readily available means; decreasing the cost of the vehicle dramatically when compared to other current defense systems. The objectives of this thesis were to incorporate design aspects from the missile design class, ME4704, of 2016 [14]. Realizations of designs were meant to improve the functionality of the following subsystems such that the flight vehicle would become reusable, reconfigurable, and adaptive to future research and development needs.

- Servo motor control design
- Structural design and machining
- INS/GPS recording and integration
- Aero analysis
- GNC algorithm development and simulation
- Parachute deployment and altimeters

A. SERVO MOTOR CONTROL DESIGN

Developing a vehicle with the potential to be guided necessitated that the rocket have internal control ability. Previous designs have considered multiple possibilities for control [13]. Brent Aldridge, student at the Naval Postgraduate School, contributed to servos motor control. After considering recommendations as well as new ideals, it was decided the method of control was to use servos. They would be installed with reduction gears to transfer and amplify the torque. The fins' actuation would then control the rocket. This section focuses primarily on the hardware, software, and coding required to send the right control signals to the servos for the desired mechanical output. The ultimate goal for the rocket is to have the flight control computer (FCC) operated from the PC/104 located in the nosecone of the rocket. The PC/104 board is intended to use real time data with guidance algorithms to provide guidance command to the vehicle. Advancements to the vehicle required a demonstration of the capability of servos to actuate within the rocket with the installed fins during flight. Therefore, to mitigate risk and retrieve reliable data, alternate means of control were explored apart from the PC/104 board for controlling the vehicle.

1. Background

Servo motors are small DC motors that have an encoder and potentiometer. They have an input voltage and ground connections in addition to a third control signal connection. The encoder, potentiometer, and control signal allow the servo to be rotated to a specified radial position and generally at a specified rate. For the purposes of the flight vehicles, the servos actuation is relayed to the fins, positioning them in a desired angular position to control the flight of the vehicle.

The servos are controlled by a pulse width modulated (PWM) signal. How different pulse widths correspond to different angular positions on the motor is shown in Figure 6. In the case of the HITEC motors used for this rocket, the period of the signal was 20ms, update speed was 50 Hz, with a pulse ranging from 1ms to 2ms.

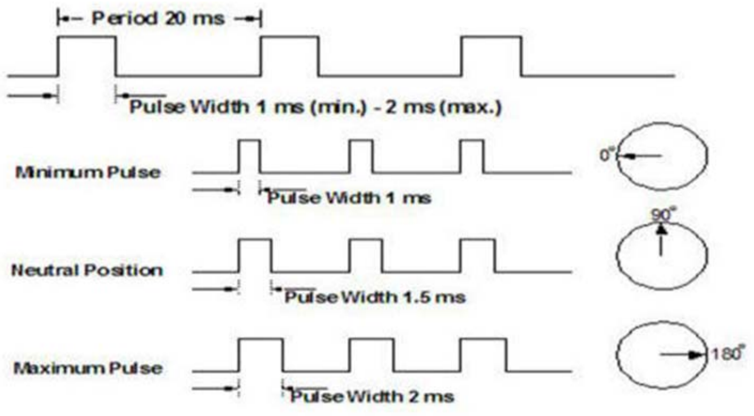


Figure 6. Servo Control Signal. Source: [15]

2. Design Process DAQ and Arduino

The first step in the design process was to find a way to send the required PWM signal to the servos. The flight control code was written in Simulink because of the ability to perform real time processing. The ideal solution was to use hardware to be able to send the PWM signals through Simulink or MATLAB. The previous design had acquired a data acquisition board (DAQ) from Measurement Computing (MC-USB-1608FS-Plus). This piece was capable of sending the PWM signals through the digital I/O ports as well as record flight data from other sensors onboard the flight vehicle using the analog inputs. The flight data could then be compared the INS/GPS recorded flight data for duplicity and verification.

The DAQ from measurement computing did have some support in the data acquisition toolbox for MATLAB and Simulink, but the support was considered legacy hardware by MATLAB. This meant that there were very few blocks or tutorials in Simulink to help with finding a way to send the required signals to the servos. Faced with the need to develop new code using Simulink, alternate routes were considered. At this time, a new and much more capable GPS/INS sensor was acquired. The capabilities of the new sensor were better than the previously planned sensors that the DAQ was intended to operate with. Therefore, alternate control theories were explored. The simplest and most cost effective alternative that was found was the Arduino. There were numerous tutorials online for controlling servos as well as Simulink blocks for servo control using an Arduino. Another benefit was that the Arduino could be run independently from the PC/104. Although the PC/104 needs to communicate to the Arduino to control the servos from the flight control code, it would be valuable to be able to test both independently on a launch to reduce to points of possible failures. Due to advantages of the simplicity of design, the plan was changed to include using the Arduino as the servo control hardware, at least for initial actuated flight tests.

3. Servos

In addition to finding the correct hardware to send the signals to the servos, tests were also performed to measure the performance of the HITEC Servos. The operating characteristics of the servos were needed for the implementation of the model of the flight vehicle. The biggest concern was the speed at which the servos actuate and if the servo would be enough to control the flight of the vehicle. The manufacturer's specifications on the servos stated that it could rotate 462°/sec. The servo used was attached to a gear box, which reduced the speed, by a factor of 7. With the gear box, the predicted speed of the large gear was 66°/sec. The servo was tested in a controlled environment and found to considerably slow considerably through low angles of movement. Between 0 and 5 degrees the measured speed of the servo was 3°/sec. Although different than what expected, the servo was suited for controlling the fins.

The servo rotation speed through low angles of travel beckoned investigation into alternate performing servos. One considered servo manufacturer was Futaba. Their servos were built to maintain rotation speed even through small angles of travel. The Futaba servo also had more capabilities like a programmable update rate and other features that could be beneficial should the flight vehicle need them in future launches. There were delays in ordering the Futaba servo pieces and machining parts for the servo connections. As a result, the HITEC Servos that were originally tested and narrowly fulfilled the needs of the launch were used.

4. Launch Objectives

The main objective for the Arduino microprocessor and servos was to actuate the fins to perform a roll and counter roll after motor burnout. A secondary launch objective was to record flight data in order to verify data from the GPS/INS sensor as well as match the time at which the servo was actuated to the time recorded on the GPS/INS sensor. If possible on a second launch, the primary objective was to actuate the fins to perform a pitch maneuver of the flight vehicle. More launch objectives are discussed in the testing chapter of this report.

5. Hardware

The main piece of hardware was the Arduino Uno board shown below in Figure 7. The Arduino Uno was selected for its wide use in a variety of applications as well as its USB connection and compatible hardware like the SD card data logging shield.

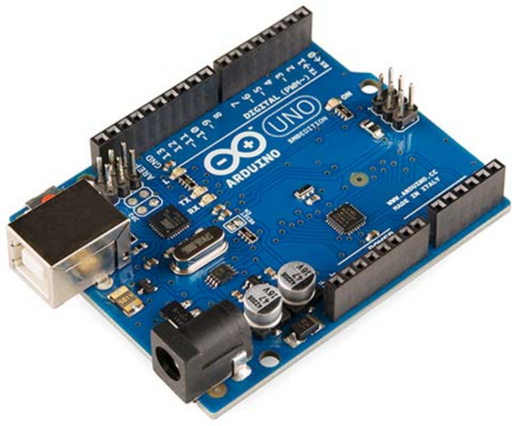


Figure 7. Arduino Uno

The **Adafruit** data logging shield is shown mounted on the Arduino UNO in Figure 8. This shield has the capability to write data to an SD card while allowing access to all of the ports on the Arduino Uno. It hosts a native clock that runs on a small battery to that the time keeping function for the board does not have to be reset every time the Arduino loses power. The shield also had a prototyping area to mount additional hardware.

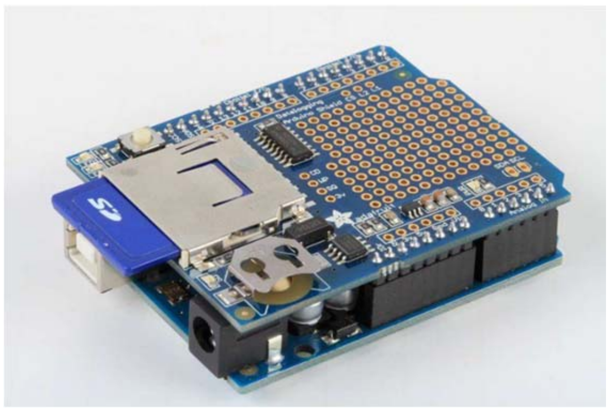


Figure 8. Adafruit Data Logging Shield

The final piece of hardware was the Adafruit 10 DOF IMU Breakout. This sensor has an accelerometer, magnetometer, and gyroscope oriented in three directions as well as a temperature and pressure sensor. This sensor was selected for its ability to measure altitude through the temperature and pressure sensors as well as acceleration and gyro to verify the GPS/INS system connected to the PC/104. The Adafruit is shown in Figure 9.

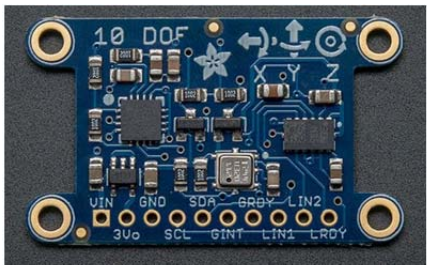


Figure 9. Adafruit 10 DOF IMU Breakout

6. Software

The Arduino Uno is programmed in software called the Integrated Development Environment (IDE). The coding language is most similar to C++ with a few variations. All Arduino scripts have a setup portion of the code, which runs through once, and then a loop section that will run as long as the board has power. The programs written for the launch of the flight vehicle can be opened on any computer with the IDE software and be compiled and loaded on any Arduino.

7. Launch Code

Two launch codes were developed. They were almost identical except for the actual position commands sent to the servos. The first section of code includes the libraries for each of the sensors used on the Arduino. The launch code snips are shown in Appendix B. Arduino Launch Code. These libraries must be installed on the computer that is compiling the code onto the Arduino. The libraries from the Adafruit website [16] were downloaded on a computer and adapted for the events. The code assigned a unique ID to each of the sensors on the 10 DOF IMU. The 10 DOF IMU is the sensor attached to the Adafruit data logging shield that reads specifics including acceleration, gyro, and other navigational aids. This allowed the code to call specific commands to each of these sensors later on. After each senor was assigned, a real time clock was established on the data logging shield.

Initiation of roll and or pitch commands came next in the coding sequence. The roll command assigned different servo positions for each fin. Initial zero for each fin was set with one position being at 92° and the other at 86°. The roll and pitch commands differed. For the roll the command the first position command was based on as altitude attainment of 610 m. The servo position one for each fin was 94° and 88° respectively. After staying at the new position for an allotted time period, three seconds, the follow on positions were initialized. For the pitch maneuver, the servo position was similar, but instead of moving from 92°-94°-90°-92° and 86°-88°-84°-86° the commands for the servos were to go 92°-94°-92° and 86°-84°-86°.

Following the servo position command was the initialization of variables for altitude calculations, the servos (one and two), and the data logging shield. The variables were to be used and logged from the IMU to the SD card. The seaLevelPressure variable must be updated with the pressure at ground level prior to launch, giving a reference pressure for the altitude calculation later in the code. Each manufacturer of data logging hardware for Arduino has a unique ID and Adafruit’s is 10.

The section of code which created a function called initSensors initiated the sensors on the 10 DOF IMU followed variable initiation. This function was called later in the setup portion of the code. In the function, if statements were used with the begin command to start each of the four sensors. If the begin command did not return a true value then the code would go into an infinite while loop that will print an error message to the serial port if the Arduino was connected to a computer.

The setup loop for the operation followed. Everything in the setup loop only ran one time after the code started. The first command in the setup loop established serial communication through the USB at a baud rate of 115200. This command script can remain in the code even if the Arduino is not be connected to USB while the code is running. The next command establishes communication through the I2C protocol which the 10 DOF IMU uses. Following is the real time clock on the data logger. The servo commands link the servo objects defined earlier to the pins which were connected to on the physical Arduino. For this specific setup, servo 2 was connected to pin 9 and servo 4 was connected to pin 3. The final line of code calls the initSensors function explained earlier.

Initiation for the unique filename follows. The first step was to make an array of characters called "LOGGER00.CSV." The for loop and if statement then incrementally increase the numbers in the filename until a file of that name did not exist on the SD card at which point it created a file and exited the loop. This method for naming the files only works for up to 100 files on the SD card. The final bracket in this section of code ends the setup portion.

After the setup portion came the continuous loop that created an event which later called to get data from the 10 DOF IMU. The altitude variable was then updated with the altitude calculated from the measured temperature and pressure, based on the information from the 10 DOF IMU.

Following the logging loop is an if statement, which considers the logged altitude. If the altitude is above 610 m then the servo position commands were then executed while acceleration and gyro roll rates were recorded. This was where the codes for the pitch and roll commands differed. The algorithm included a failsafe where the codes for the position commands will only run one time during flight.

If the altitude was greater than (610 m), and the codes had not previously ran, the commands were sent to the servos to actuate to the positions specified, servo2pos1 and servo4pos1. The flight plan for both the pitch and roll commands was to give the servos a position and have them hold for 3 seconds. In order to achieve this while still logging data simultaneously, a delay command could not be used to achieve the desired amount of time. Instead, a for loop was created that would record the desired data that would run after the servo command was given. Each loop of logging data took roughly 15 milliseconds, so the for loop was run 200 times to achieve the 3 seconds of servo actuation desired.

In the data logging loop, the new command grabbed the current time from the real time clock. The logfile.print command printed information to the variable logfile, which was specified earlier as the name of the file on the SD card. The loop recorded the minute, milliseconds since the program was started, the gyro in the x direction, acceleration in the x direction, and the command being sent to one of the servos. This was determined to be the minimum amount of information required to verify the GPS/INS sensor connected to the PC104. The final command, logfile.flush finished the line of data printed and closed the file on the SD card.

The final section of code is what the program skips to if it never reached 610 m or when it finished the roll or pitch commands. The section commanded the servos to go back to their neutral positions and continued to log data. This eliminated actuation from the fins.

8. Recommendations for Future Work

The code was close to the maximum size for the memory onboard the Arduino Uno. For future work using the Arduino, it will not be necessary to log data using the data logger or the 10 DOF IMU, except for redundancy purposes. Making the code to run the servos much simpler. Long term servo work includes reaching the ultimate goal of having a forward control computer run a real time executable to provide guidance, navigation, and control to the rocket. The PC/104 board can communicate to the servos within the vehicle via the raceway and USB connection which was tested during experimentation. Full application of this guidance will be included in follow on work.

B. STRUCTURAL DESIGN AND MOUNTS

Integrating the advancements together required machining and making mountings for individual pieces. There were two specific areas of the vehicle that required machining for advancements to be applied.

1. Nose Cone Mounting Design

Prior work with the vehicle had considered nose cone design and required electronics, but had not materialized the mountings to support designs. It was required to redesign and implement mounting techniques to mount, run, and protect the PC/104, GPS/INS system, camera, and associated electronic. In order to achieve INS/GPS recording and confirm the viability of installing smart capabilities within the vehicle, fixing the nose cone electronics within the system was essential. The nose cone design structure included temperature monitoring, secure housing, and data transfer capabilities.

a. Design

Nose cone design required optimizing stability, weight, and space. In order to have structural sound mounting locations, materials for design needed to have those attributes. Initial building materials considered were wood, steel, and aluminum plates. Prior designs had considered using a wooden mounting board. Rigidity considerations pushed for the notion that the initial wood build was not strong enough to maintain the integrity of the structure. Wood exhibits desirable cost and weight characteristics, but not the strength and modulus that was required. Steel plates would fulfill the need, but expense and weight density were considered to be unnecessary. Therefore, aluminum plates were used as base boards in design to mount pieces. The base design mounts are shown in Figure 10.

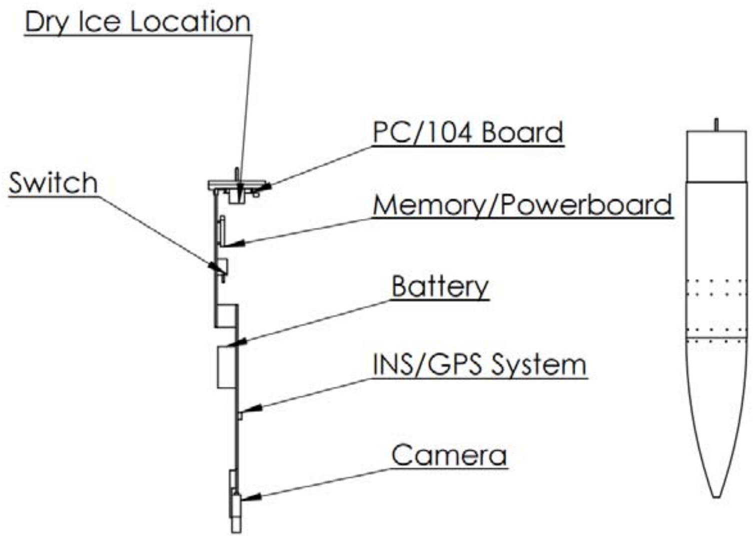


Figure 10. Nose Cone Mounting Board Layout

Launching the rocket at accelerations up to 12 g, required that rigidity and support of the electrical equipment be considered. The first challenge was securely mounting every item to the aluminum support board. This included locations for the PC/104 board, memory drive, power board, INS/GPS sensor, battery, and camera for future optical tracking. In order to align the acceleration vector in a somewhat favorable direction for the PC/104 board, the mount was planned on a circular bulkhead that would absorb the direct force of the g. This bulkhead was reinforced with high density foam to further cushion the mount without allowing for too much movement of the mounting board; ensuring integrity in connections. The power board and the memory drive were stacked together and mounted to the board to minimize needed space and maintain relatively close proximity to the PC/104 board. The INS/GPS sensor and the battery were both fastened directly to the forward board, as shown in Figure 10. The two mounting boards provides for further modifications and modularity of the system. The battery was fastened firmly in place by zip ties and surrounded by high density foam; while the GPS/INS sensor was mounted using spacing screws.

At 12 g of acceleration, the cooling fan accompanying the PC/104 would not function. The mounted fan needed to be removed and alternate cooling methods would be needed for the PC/104 to function through the launch, recovery, and following flight between optimal operating temperatures (15℃ and 55℃). To determine the cooling requirements, different operations were ran on the PC/104 board without it being cooled apart from the aluminum mount and heat sink. When performing at anything greater than 45% capacity CPU demand, the computer would eventually heat up to near damaging levels (70℃), after running for 30–40 minutes; about the time needed for launch and recovery. Therefore, the setup required that an alternate cooling strategy be developed. Alternate cooling sources were explored by calculations and evaluating the heat removal needed. The desire was to keep the operating temperature between 15℃ and 55℃. Based on the estimate there was about 900 W of power being dissipated from the CPU, a simple heat transfer calculation revealed that there would need to be over 100℃ degree fluctuation between the contacting aluminum 6061 piece to the CPU and the external piece of the heat sink. All the material used in the heat conduction was 6061 aluminum. In order to achieve that amount of temperature change, and still maintain a clean working environment, dry ice (CO_2) was evaluated. In order to prove the viability of dry ice, the piece would need to be small enough to fit in the nose cone and on the press area of the heat sink. But large enough to be able to function as a coolant for at least 45 minutes. Assuming that the inside temperature of the nose cone only reached to 35℃ and that the sublimation rate was constant at about 4.5 kg (10 lbm) per 24 hours at room temperature. Through using the density and fastening area, it was found that the piece ought to be 4.57 cm (1.8 inches) by 7.62 cm (3 inches) and 2.41 cm (0.95 inches) thick. This piece would theoretically keep the computer cooled and last the entirety of the launch and recovery sequence. Dry ice was determined to be the best solution available.

Determining the viability of dry ice through calculations was followed by ground testing the cooling process. The eventual design consisted of conducted heat through an aluminum block, a heat sink, another sheet of steel, and then the dry ice was pressed in contact with the sheet of aluminum. This allowed for cooling to occur from launch to recovery of the vehicle. The final machined holding area for the electronics in the nose cone is shown in Figure 11.

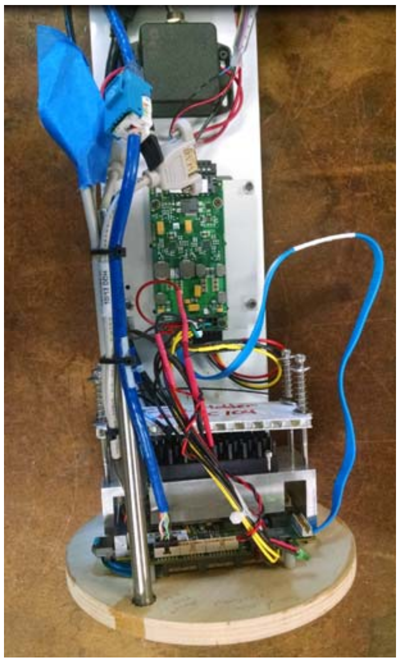


Figure 11. PC/104 Board with Dry Ice Holding Area Shown

The dry ice was a 1.9 cm (3/4 inch) thick rectangular (5 cm x 7.6 cm) piece for the first launch. The dry ice piece was completely sublimed upon recovery. For the second launch the piece was 2.54 cm (1 inch) thick rectangular block; a small amount of it was still present upon recovery.

b. Machining

Machining for the nose cone mounting structure was done exclusively by Robert Wright of the NPS Rocket Laboratory. Modularity considerations were made to retain access to the electronics board both within the nose cone and when it was removed. The entire nose cone mounting board is shown next to the nose cone in Figure 12.

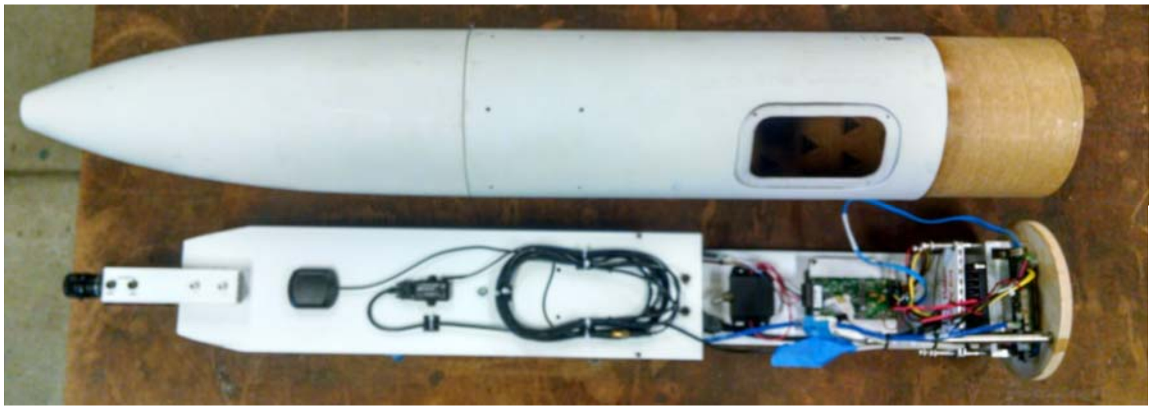


Figure 12. Complete Nose Cone Mounting Board

2. Fin Actuation Hardware

Prior to the December 2015, all launches of the rocket included stationary fins locked into vertical positions. The outlook for this new version of the rocket was to perform a controlled maneuver of the rocket by actuating the fins. This is not the pinnacle of Phase I, but rather an essential step in reaching full control and being able to perform the theory discussed for countering UAVs. For this version, two of the fins were controlled.

a. Design

Space to install servos for fin actuation was already allocated in the previous vehicle design. Within that space, fin attachments were designed to be controlled by the Hitech servos, installed one meter above the fins. The general spacing between the fins and the servo region on the rocket is shown in Figure 13. The design required the transfer of rotating motion from the servo, an initial torque, to a push rod, and then transferred to the fin setup in the bottom of the rocket.

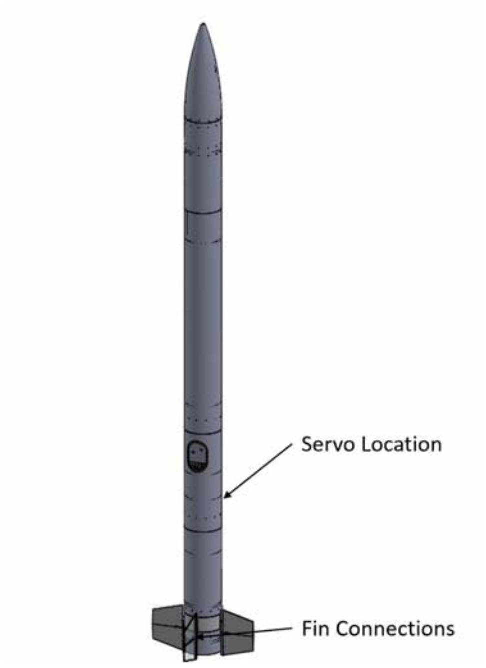


Figure 13. Rocket with Annotated Servo to Fin Distance

Originally the fins were designed for supersonic conditions and to be rotated around the mid chord location. Since the vehicle was now being designed for subsonic flight, optimal designing included shifting the rotation axis to the ¼ chord location due to lower pitching movement. But since the fins were originally designed for center mounting, a fin mounting sled was needed to change the point of rotation. Designing the fin sled required ingenuity in being able to make the preexisting fins adaptable to the new preferred flight condition. The design for the sled along with the fin connection to the pushrod is shown in Figure 14.

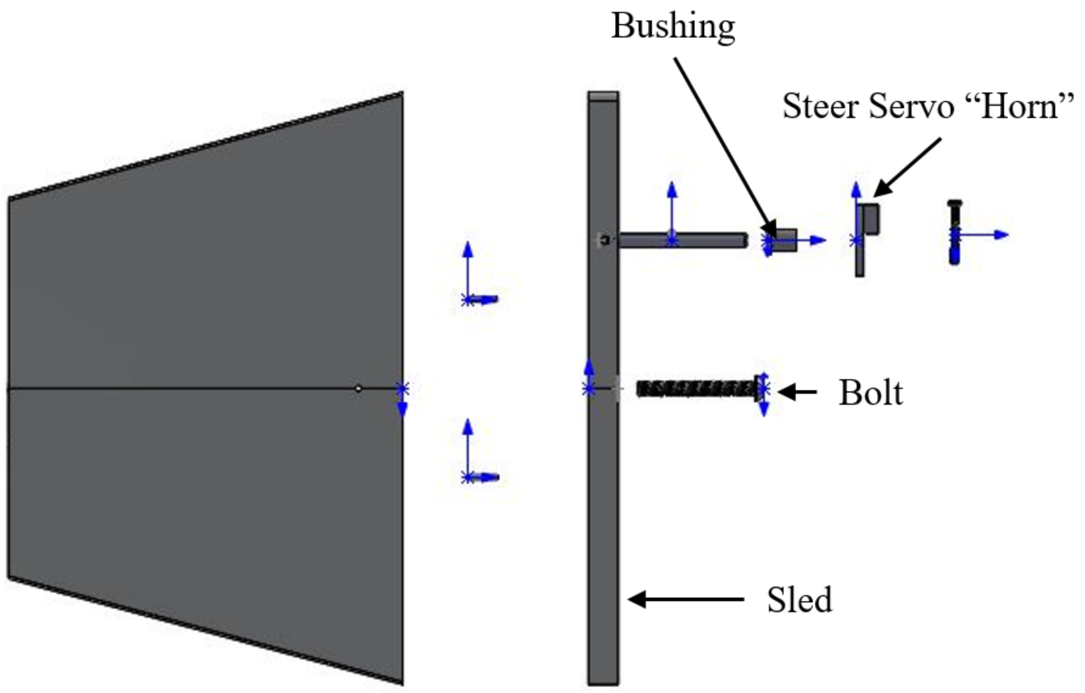


Figure 14. Exploded View of Fin Design

Each element of design was essential to allow the fin to turn around its ¼ chord point instead of mid chord location. The figure shows that the servo "horn" is connected to the sled axle and that the pushrod that connected to the servo gears.

b. Hardware and Machining

The fabrication and installing of the design was also accomplished prior to the April 2 launch date. The fabrication work was mostly accomplished by John Mobley of the Naval Postgraduate School, and by Rocket Laboratory support staff and students. The completed mounting sled, which allowed the fin to actuate along the ¼ point rather than the midpoint, and fin are shown in Figure 15.

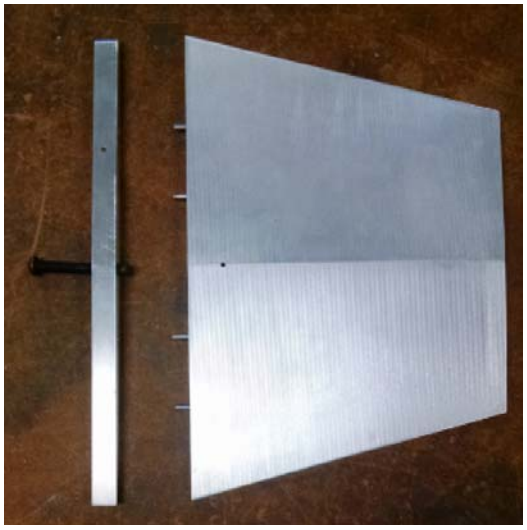


Figure 15. Machined Fin with Fin Sled

As explained previously, the steer horn connected to the pushrods, coming down from the servos above. The finished parts fit together within the motor casing area of the rocket is shown in Figure 16.

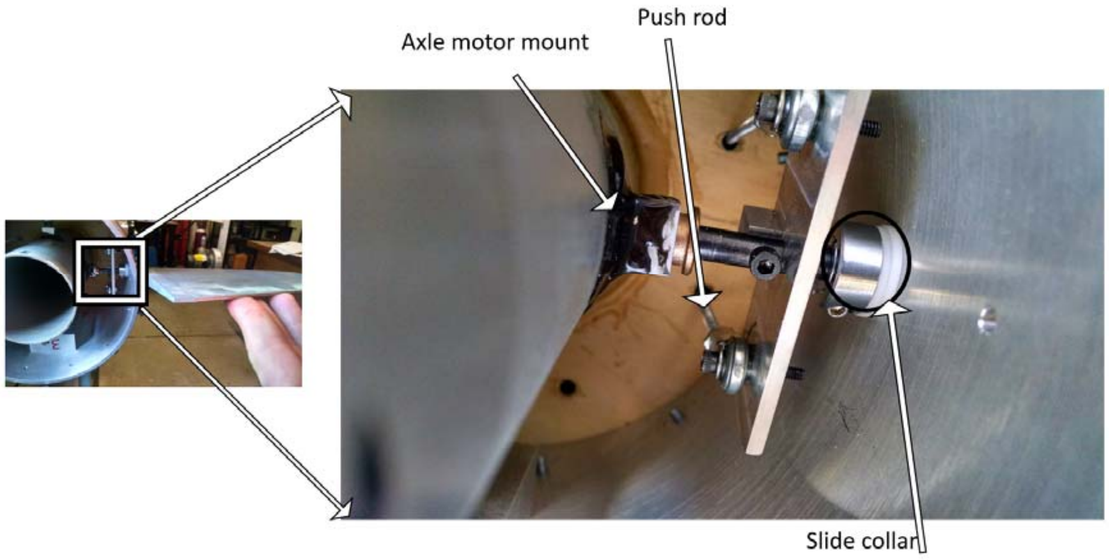


Figure 16. Finished Inner Can Fin Connections

One of the challenges and objectives during the development of this iteration of the vehicle was to produce a vehicle that was essentially off the shelf, or reproducible for minimal cost with readily available materials. Therefore, the production of the rocket would be repeatable in a relatively short amount of time. To make the vehicle in such a timely matter, allowances were made for tolerances. Each of the tolerances added up to produce a final fin which was not as stable as desired. The “too much play,” or instability of the fin, was the $\pm$ 1° rotational play in the fin combined with the approximately $\pm 0.5°$ of yaw in the fin and body relationship.

Originally constructed for fixed fins, the fin attachment can proceeded to present a design problem for tolerance control going forward. The original can was intended to have fixed fins held in place through a 1.03 cm (13/32 inch) diameter hole. The hole was going to be reused to set up the actuatable fins. This fit was not ideal since the only bearings available to fit “off the shelf” were 0.95 cm (3/8 inch) outer diameter, giving way to 0.0813 cm (0.032 inches) of space for extra play. This area was fed with shim stock to a seemingly tight fit, but determination of usability allowed for the vehicle to progress even though the remained undesirable tolerance levels in the bearing and fin fit. Future work should include machining the fin can to the exact dimensions for the bearings and coinciding axle to feed through. The fitting could be made to heat/cold press into position.

The axle for the fins was a 0.635 cm (1/4 inch) steel rod produced at 0.000254 cm (0.0001 inch) tolerance on the diameter. That tolerance was combined with the 0.0025 cm (0.001 inch) tolerance on the bearing. Since it was ordered at this tolerance and size, the only machining needed was cutting down to length (from the fin and sled connection to the motor casing mount). The tolerances for locally designed and cut pieces are shown in Appendix C. Each of the pieces took approximately two weeks of designing, measuring, and detailed design drawing prior to being cut. The machining was finished locally. The process could be reasonably duplicated with simple machining capabilities.

One of the limiting pieces in the design was the fin sled, mentioned earlier. The sled was intended to allow for the fins to actuate around the ¼ chord length rather than the half chord the fixed fins had been originally designed for. The shift necessitated that a sled be designed to fasten to the fin while changing the rotating point. The small source of error that was minimized but still present was the connection of a dowel pin through the sled and axle. The pin essentially stopped the forward back slide of the axle in the fin slot, but allowed for the few thousandths of tolerance to add to the amount of remaining play, and this allowed yaw in the fin. The greatest source of play came though from the pushrod assembly. From the servos connection in the midsection parachute electronics bay, the servo rotation was connected to approximately 83 cm aluminum pushrods. The push rods were verified to have the compression and tensile strength required to withstand the load, but were still quite flexible. The pushrods were published to be 6–32 thread. This design put the outer diameter of the thread at 0.336 cm. These rods presented excessive tolerance because the guide holes were 0.635 cm diameter holes. Guiding from the base of the parachute electronic bay into the tip of the fin can section. Within the excessive guides, the rod experiences small bows that were the supported through the guides. This play could not be remedied with shimming or other off the shelf techniques, but future versions ought to consider having a tighter fit between the push rods and guide holes. The servo with attached push rods are shown in Figure 17.

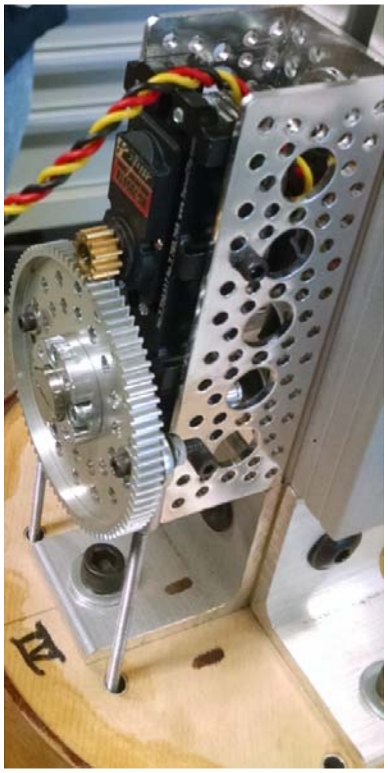


Figure 17. Servo Mounting with Push Rods

Decreasing the tolerance in the fin actuation subsystem would include heat pressed bearings to guide the rods through. Since the design was essentially capable of being made off the shelf, each of the tolerance designed for were amplified up until the fin's fluctuation. Slight decreases in tolerance would be allowable in many of the designs, especially axle to bearing, and bearing to can.

3. Rocket Separation

In order to have a controlled and repeatable separation, there were many changes and additions needed. Prior versions of the rocket found greatest hindrances to modularity and repeatability because of the lack of advancement in this area. Within the parachute electronics bay the prior design had individual wires running from the altimeters to the black powder charges. The charges in the parachute bay were three individually wrapped black powder sacks. In the new version, extensive planning and electrical analysis optimized the setup to have limited free wires, with mounted connection points. The most advancing item was the installation of four and six pin connections. This allowed for consolidation of wiring and simple connections post assembly and through the door. The final setup of the mounting is shown in Figure 18. The deployment charges to release the parachutes are shown in Figure 19. The wiring interfaces are significantly improved in this version of the vehicle, making a second launch possible within a two hour window.

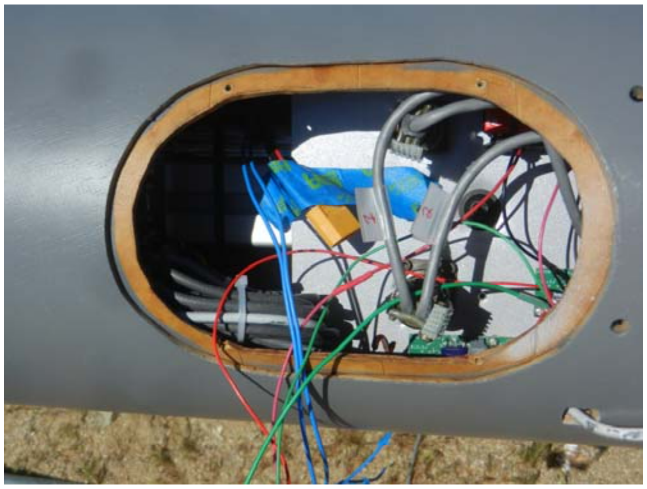


Figure 18. Wiring Inside the Parachute Electronics Bay

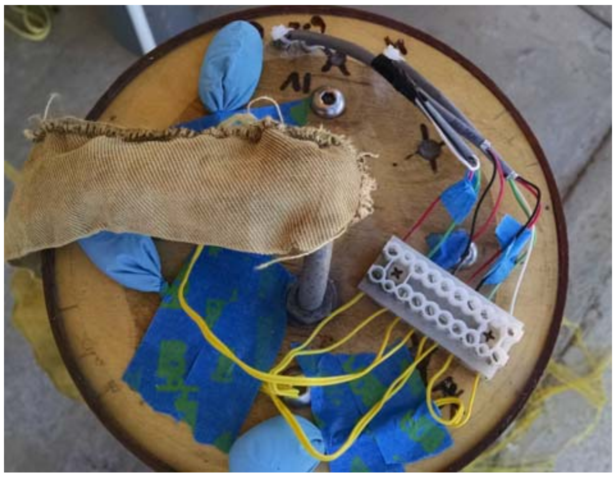


Figure 19. Parachute Deployment Location

The separation point in the phenolic fiberglass tubing had been a compromised location in prior launches. The most harmful locations occurred near the shear locations. To improve the separation dynamics and reduce damage to the tubing, shearing copper shim stock was fastened to the inside of the outer tube and the inner tube. The shim were set to slide across each other. A shearing pin was drilled through both layers and set to hold the separation point together until the black powder charges fired, creating a clean shear event due to the copper shim. This shear resulted in a clean separation with no damage to the rocket body.

C. INS/GPS RECORDING AND INTEGRATION

The goal of the development of seeker capability for the missile was to have a dual mode seeker, which used a GPS/INS for initial guidance and an EO sensor for terminal guidance. The focus included in this portion of work was to understand and be able to configure, record and parse outputs from the sensor acquired during the thesis. Wee Yeow Lim contributed to this section of work during ME4704. The flow pattern for the development of seeking abilities is shown in Figure 20.

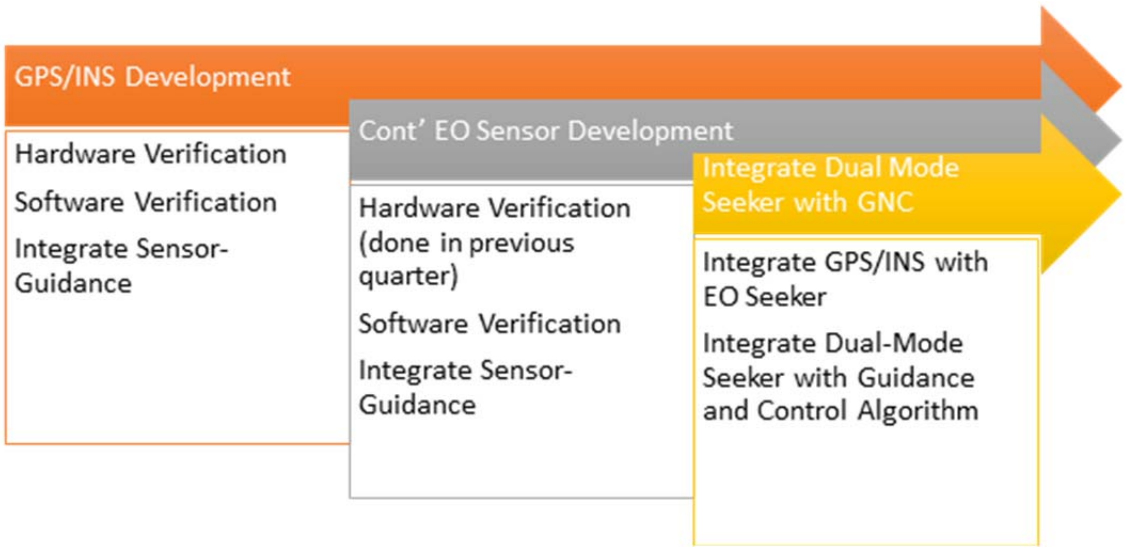

GPS/INS Development
Hardware Verification
Software Verification
Integrate Sensor-Guidance
Cont' EO Sensor Development
Hardware Verification (done in previous quarter)
Software Verification
Integrate Sensor-Guidance
Integrate Dual Mode Seeker with GNC
Integrate GPS/INS with EO Seeker
Integrate Dual-Mode Seeker with Guidance and Control Algorithm

Figure 20. Outline of Dual Mode Seeker Development

a. *GPS/INS Development Phase*

The development of the integrated system between the GPS/INS Sensor and Guidance and Control Algorithm was separated into three stages - Hardware Verification, Software Verification, and Integrate Sensor-Guidance. This decision was made to reduce risk and ensure a systematic development that could verify various portions of the integration at launch conditions. Full development of this integration will require three launches. Details of this spiral development are shown in Figure 21.

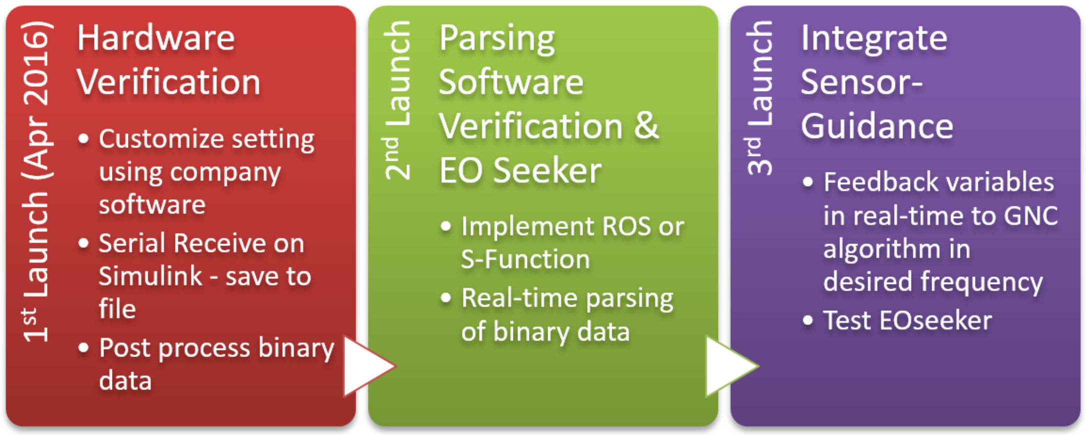

1st Launch (Apr 2016)
Hardware Verification
• Customize setting using company software
• Serial Receive on Simulink - save to file
• Post process binary data
2nd Launch
Parsing Software Verification & EO Seeker
• Implement ROS or S-Function
• Real-time parsing of binary data
3rd Launch
Integrate Sensor-Guidance
• Feedback variables in real-time to GNC algorithm in desired frequency
• Test EOseeker

Figure 21. Three Stage Spiral Development of GPS/INS Sensor—Guidance Integration

(1) First Launch - Hardware Verification (Completed April 2016)

In this launch, the main objective was to achieve streaming of desired data, and to recover these data for post-processing offline. The desired variables, offset for the sensor and GPS antenna position and orientation, calibration of the magnetometer, and capture of gyro bias were done using the MIP Monitor software, and the settings were saved.

(2) Second Launch - Software Verification (Completed April 2016)

In the second launch, the real time parsing software was developed and tested independently. The values were then recorded with to be verified with Arduino. The results are discussed in the data collection portion.

(3) Third launch - Integrate Sensor-Guidance (To be completed)

In the third launch, assuming the software for the GPS/INS has been verified during the second launch, the vehicle can begin to utilize real time GPS/INS sensor data for initial tracking. Afterwards, work may continue on development the EO for terminal phase and integrate both sensors with the GNC algorithm for the complete dual mode seeker can be initiated.

b. Current Hardware - LORD MicroStrain 3DM-GX4-45

The GPS/INS Sensor acquired for the rocket was the MicroStrain 3dm-GX4-45. The sensor is advertised to be able to sustain and compute measurements for maneuvers up to 16Gs and is supported with a user friendly and intuitive software. The dimensions of the sensor are shown in Figure 22.

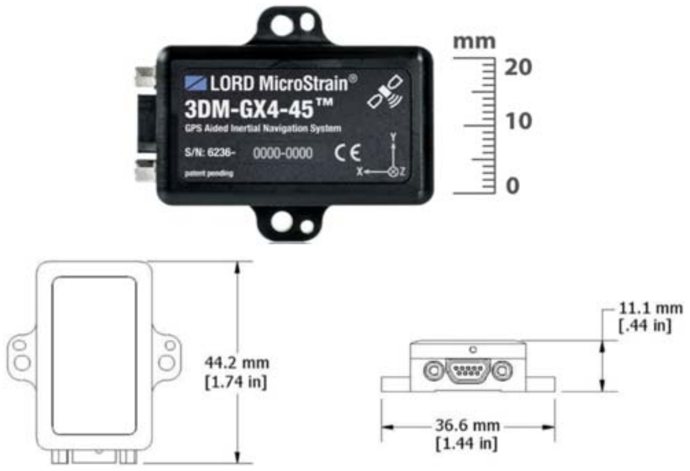


Figure 22. Dimensions of GPS/INS Sensor. Source [17]

Included with the sensor is the user manual. As per the manufacturer's manual [17], the IMU works similar to other IMU devices by using acceleration and angular rates. The captured data inputs are processed internally to give nearly exact outputs that can be fed into the computer system. The data readings are processed using the adaptive Kalman filter (AKF) prior to being fed into the computer. Many of the features of the sensor are found within this filter. The sensor and the connected GPS can be and are read using the LORD MicroStrain MIP Monitor software [17]. A block diagram flow of data within the device is shown in Figure 23.

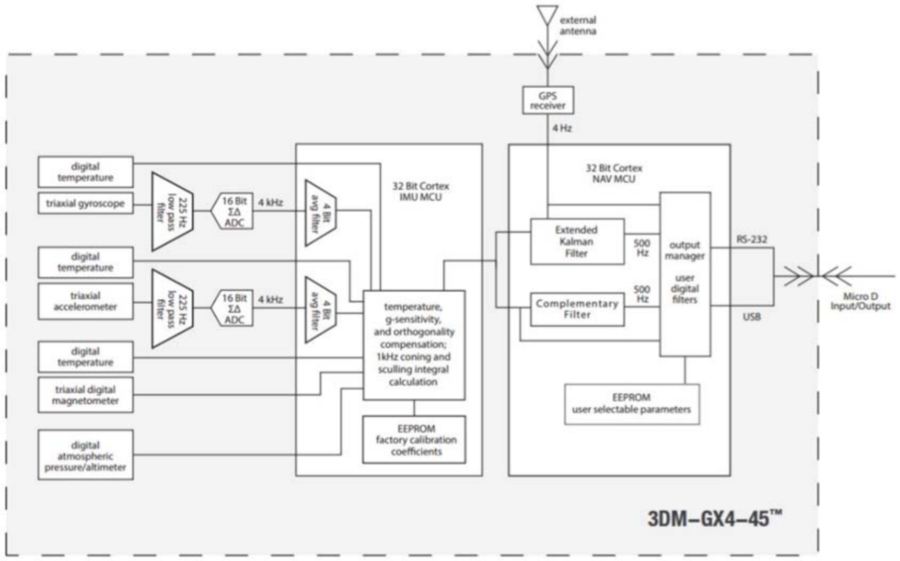


Figure 23. Block Diagram of 3DM-GX4-45 GPS/INS Sensor. Source [17]

c. Sensor Data Acquired from Launch

The settings used for the launch are shown in Table 2. The data gathered using the INS/GPS system are shown in the Appendix D. Raw Data from Flight and discussed in the data collection and results sections.

Table 2. Settings for the GPS/INS Sensor for Missile Launch

Settings	Values / Mode
Sensor Offset	
GPS Offset	
Magnetometer Calibration	<u>Launch Site - Friends of Amateur Rocketry</u> 35°20’49.6”N 117°48’30.9”W A value - F value -
Operation Mode	High G
Drift Compensation	GPS
Gyro Bias	Captured
Variables acquired	<u>Kalman Filter</u> 1. Angular rate 2. Compensated Linear Acceleration 3. Euler RPY 4. GPS Time 5. Position (LLH)

d. Recommendations for Future Development

Recommendations for future development in the nose cone include work on the GPS/INS sensor and on image processing and tracking. Much of the work will need to be viewed as inter disciplinary.

(1) Development of Parsing Software for GPS/INS Sensor

The development of the parsing software for the sensor will be a technical task requiring substantial work. The parsing software could be done through ROS toolkit in Simulink. Other possible avenues could involve looking into S-Function in Simulink and C++ programming for parsing task.

(2) Development of Camera Seeker

The EO seeker was not the focus for this version since this version was targeted at developing and installing the GPS/INS Sensor. Previous work on the EO seeker was done to track the sun specifically, which had unique features of a round shape, and a size that does not change significantly with closing distance [13]. MATLAB scripts were developed in the past but not applied in this portion of development.

D. AERO ANALYSIS

"Aerodynamics is the study of how gases interact with moving bodies" [18]. The forces are exhibited as drag and lift caused by the air flowing over the control surfaces and rocket as it flies. The Aero Analysis section of this report includes calculations for the most significant aerodynamics of the new designed rocket, Alejandro Garcia Aguilar contributed to this section of the ME4704 report, portions of that work were determined to be applicable to further vehicle development and are repeated here to completeness of documentation.

1. Dynamics Derivatives

Dynamic derivatives of flight are based on different force coefficients that effect the flight of the rocket. Calculating the dynamic derivative was done to understand the impact of fin actuation on forces on the rocket. The coefficients used were the angle of attack and sideslip angle, lift force or side force, Mach number, center of gravity, and altitude. An example of the calculations are as follows:

$$Cd = Cd(\alpha, \beta, M, q \dots)$$

By taking the partial derivatives, it can be found that:

$$dCd = \left(\frac{\partial Cd}{\partial \alpha}\right) d\alpha + \left(\frac{\partial Cd}{\partial \beta}\right) d\beta + \left(\frac{\partial Cd}{\partial M}\right) dM + \left(\frac{\partial Cd}{\partial q}\right) dq \dots$$

Following the derivatives, by applying $Cd_0 = \left(\frac{\partial Cd}{\partial \alpha}\right)$ with $\alpha = 0, Cd_\alpha = \left(\frac{\partial Cd}{\partial \alpha}\right)$, etc., and assuming the other minimal changes in other variables, such a small angle approximation results is the following:

$$Cd = Cd_0 + Cd_\alpha \alpha$$

where $\mathrm{Cd}_0$is the total drag coefficient evaluated at α=0. Cd_α is the total drag coefficient variation with angle of attack. And α is equal to the angle of attack. The same process is applied for the other coefficients with the objective to find the total coefficient that affects the body [19]. The drag coefficient for the varying α is shown in Figure 24.

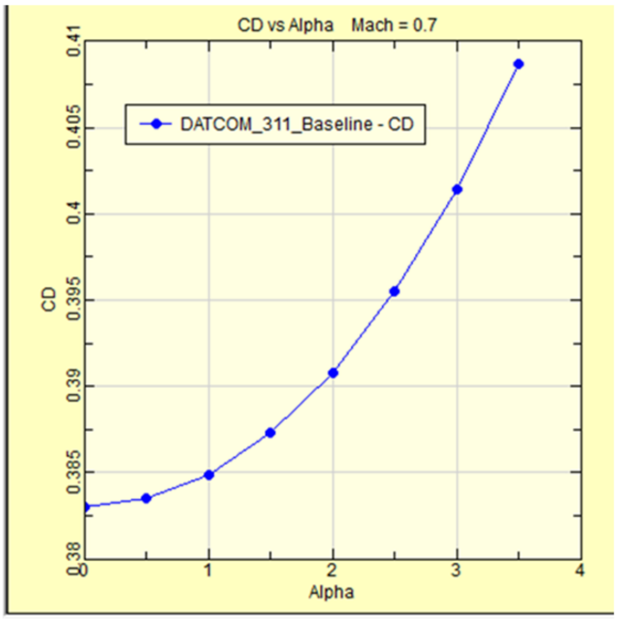


Figure 24. CD versus Alpha at Mach 0.7

2. Missilelab

Missilelab [20] is a Graphic User Interface (GUI) developed in windows environment that allows to the user to run missile prediction codes. By creating a geometry and initial conditions, the prediction code gives to the user the following outputs:

- 3D triangular surface mesh geometry.
- Aerodynamic behavior in given initial conditions.
- 2D line sketch.
- 3D solid model of the input geometry.
- Plotting of the aerodynamic outputs.

a. New Project

Under a new project option, the geometry of the rocket, version 3.0 was created. The dimensions of the rocket body geometry are found in Table 3.

Table 3. Body Geometry Inputs for Simulation

Field	Choice	Inputs
Cross-section Shape	Axisymmetric	Circular-Diameter=19.68 cm (7.75 in)
Nose Geometry	Ogive	Length from origin=51.82 cm (20.4 in) Base diameter=19.68 cm 7.75 in
Nose tip	Truncated	Tip diameter=4.1 cm (1.654 in)
Number of body segments	1	Coordinates from origin= 51.82 cm - 405.18 cm (20.4 in-159.52 in)
Center body geometry	Cylinder	Circular diameter=19.68 cm (7.75 in)
Aft body geometry	None	None

The geometry inputted into the GUI was a close approximation of the rocket. The simulation was intended to guide the development of the guidance and control. The fin type could be varied between planar and polyhedral fin. The airfoil section used was a simple hex design since it most closely approximated the actual fin. The fins designed in the GUI are shown in Figure 25. The geometric defining characteristics of the fin are shown in Table 4. The properties of the fins are shown in Table 5.

Table 4. Geometry Inputs for Fins

Field	Input
Fin sets	1
Fin type	Planar
Define by	Leading edge location
Airfoil section	Simple Hex
Semi-span	0 cm,16.42 cm (0 in,6.67 in)
Chord length	25.4 cm,16.33 cm (10 in,6.43 in)
Ratio of flap chord to fin chord	[0,0] m
Leading edge radius	[0.254,0.254] cm
Distance to chord leading edge	324.886 cm, 333.93 cm (127.90 in,131.47 in)
Fin thickness	1.016 cm, 1.016 cm (0.4 in,0.4 in)
Length of flat section	none
Orientation	[0°,90°,180°,270°]

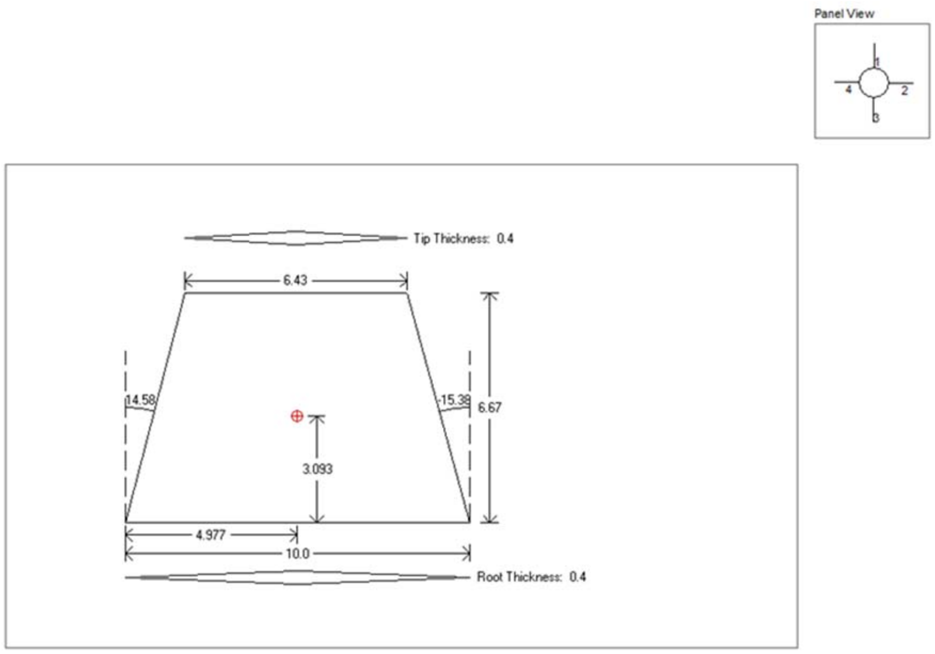


Figure 25. Fin Geometry Depiction. Source [20]

Table 5. Mass Properties of the Fin

	Fin Set	Panel	XLE	Span	chord	thickness	Weight lb.	Xcg	Ycg	Ixx	Iyy	Izz
1	Vert	2	147.9	6.67	8.215	0.4	1.424	4.976	3.093	43.30	53.82	10.59
2	Horz										10.59	53.82

b. *Defined Run Conditions*

The conditions for the flight on the rocket were made based published data for the rocket motors and projected Mach number. Included are also the inputs for the angle of attack and fin deflection. The inputs that were used during the simulations are contained in Table 6.

Table 6. Flight Conditions

Field	Input
Velocity	Mach number [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8] Altitude: 2400 feet for all Mach’s
Angle of attack	3,5,7,10
Apxx Input	Roll configuration

For the fin deflections, deflections varied for each of the runs. The initial setting had the fin set for [-3, 0, 3, 0] at 3° of deflections. Other runs included changes to $5°, 7°,\ and\ 10°$.

Along with a MissileLab simulation a model of the fin was constructed using ANSYS. The objective was to find the normal force experienced on the fin due an angle of deflection during flight. The simulation showed that the greatest normal force was when the fin was actuated 10°. This resulted in 502.6 N (113 lbf.) in the y direction. The simulation also confirmed the plan to actuate the fin only 3° for initial testing.

E. GNC ALGORITHM DEVELOPMENT AND SIMULATION

To solve the GNC problem implies a solution to three distinct problems: guidance, navigation, and control. A fourth problem, developing a reliable and accurate Plant Model (software representation of the missile that would simulate the INS/GPS inputs and reaction of the air vehicle to commands) was also required. The developments in this section were not applied during the launches, but the INS/GPS Sensor was configured to support eventual installation on guidance and control on the rocket.

1. Introduction and Experimental Method

The GNC solution was pursued using MATLAB SIMULINK. The issue of missile navigation was solved first, followed by the guidance solution and finally the control (autopilot). To solve the navigation problem the question of how the missile knows where it is in space had to be answered. To provide for guidance the missile must be able to know where the target is and compute the target's position relative to its own position to chart an intercept course. To solve for control, the missile must be able to read accelerations in all three body axes and command control surface deflections to arrest them. The plant model was developed alongside the GNC solution and greater fidelity added as additional information and insights from test were provided. In all four of the solutions, the rocket was constrained by the physical environment within which it existed, the limitations of its hardware (both in terms of body/structure, but also processing power and speed), and how intelligently its software was tuned (i.e., how much it "knows" about itself and how appropriately the requisite filters and gains are tuned to account for is physical and aerodynamic properties). Ultimately, assessments of the quality of the GNC solution are constrained by the accuracy and quality of the plant model, which itself is constrained by the accuracy of the aerodynamic data and assumptions upon which it was built. Without previous wind tunnel or flight test data to validate assumptions, gains, linearization's, etc., the fidelity of the final solution is in question and it must be recognized as a "first draft" that can serve as the foundation for future iterations that incorporate the lessons learned of earlier attempts. The launch of the completed version of the rocket acquired data that could validate and improve the model; leading to successful a successful and accurate GNC solution.

Regarding the scope of the problem, it is worth noting several simplifications and assumptions. The problem was simplified by specifying a stationary target, theoretically a specific location above the incoming swarm, or even a single UAV of the theoretical swarm; complexity was added by imposing a vertical launch condition. The missile itself is axisymmetric, contains an INS/GPS for assessment of its position in three dimensional space and the measurement of body accelerations, and employs a PC/104 card upon which the GNC solution would be stored and run. Fin deflections were limited to ±3 degrees but the specific characteristics of the servo-actuators (damping, natural frequency, freeplay, speed and precision of movement) were unknown at project initiation. An estimate of the aerodynamic coefficients discussed in the aero analysis section were used for specific points in the theoretical enveloping in defining the missile would behave. Given the short range of the missile, a projection of the navigation solution onto a flat earth plane was used and there was no accounting for the earth's rotation or Coriolis effects. Additionally, given the relatively low altitudes considered, no effects from other gravitational bodies were considered and gravity was assumed to be linear.

A brief walk through of some a modeling and development will be included in this section of the report, but for a more complete review of GNC modeling and development see Appendix A. GNC Algorithm Development and Simulation Results.

2. Brief Overview of Results and Discussion

The initial guidance solution involved the continuous application of calculated angles to calculations of three turn rate commands ($\dot{\Phi}$ [XY], $\dot{\Theta}$ [XZ], $\dot{\Psi}$ [YZ]) applied with gains to the missile state through the basic $\boldsymbol{\omega} \times \mathbf{V}$ relationship to align the missile with the target. The solution evolved from executing suboptimal turns to target to eventually being able to optimally function. The incorrect unwrapping is shown in Figure 26. The correct unwrapping is shown in Figure 27.

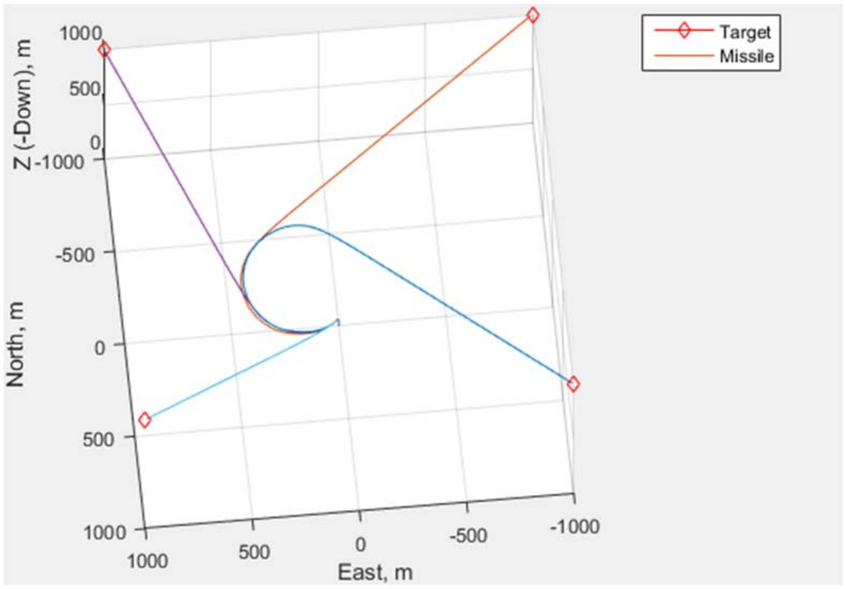


Figure 26. Incorrect Unwrapping and Suboptimal Turns

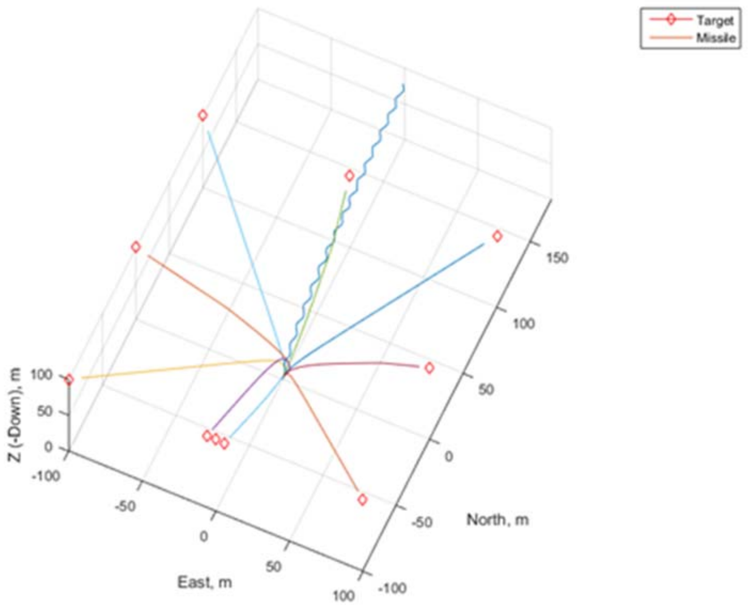


Figure 27. Correct Unwrapping with More Optimal Turns

Finally, a Global Model was created that integrated the guidance and navigation model with the control and plant models. The global model transforms the calculated acceleration commands into the body axis system (p, q, r) and inputs those as reference signals to the autopilot. It includes the beginnings of the required mechanization to remove the plant model and tie the system into the actual GPS/INS. The Global Model is shown in Figure 28. The more comprehensive discussion of the development and results of each flow block is included in Appendix A. GNC Algorithm Development and Simulation Results.

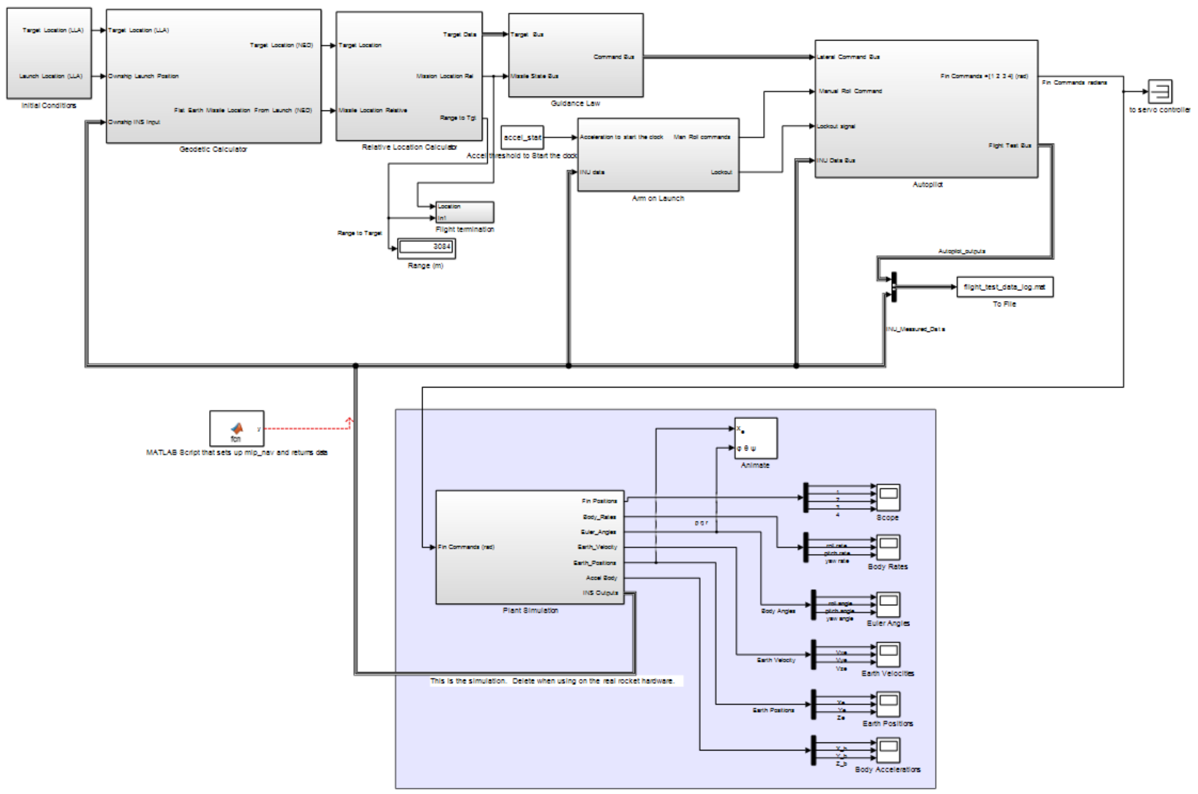


Figure 28. Global Model (Flight Program and Plant Simulation Blocks)

3. Recommendations for Future Work

Advancements to the GNC algorithm has advanced the vehicle closer to being operable. Test objectives for launching the rocket were constructed around the need to validate and provide data for modeling and guidance. Along with the need for flight data, the following are recommendations for further development of the GNC solution.

1. Hardware / Software Integration – Processing of the Simulink as an executable from the flight control computer (FCC) is critical task. Following verification, the next step is processing the data flow through the fin Digital-to-Analog (DAC) control mechanism and the GPS/INS within the Simulink environment. Once data flow and behavior is verified, an embedded coder should be used to develop the flight release of the software. Once loaded onto the flight standard FCC simple tests can verify functionality (i.e., rolling the sensor and looking for correct fin deflection) [21].

2. Tuning –Use the flight data to tune the filters and gains in the GNC program. The goals is to increase speed in order to estimate the LOS rate and avoid noise through the control signals. Another means for tuning the model would be from simulating simpler plants (i.e. 3DOF) in order to verify autopilot through having easier interpreted models. These results can then be used with the more extensive modeling [22].

3. Plant – A thorough peer review of the plant model and its underlying assumptions should be conducted. Aerodynamic coefficients should be updated in the model to include the real data collected from all launches versus the coefficients found through the aero analysis.

F. PARACHUTE DEPLOYMENT AND ELECTRONICS

The performance of parachute primary deployment devices was important because it presented a single point of failure within the vehicle. The installation and improvement of new altimeters and deployment electronics added to the vehicle to provide up to three redundant deployment events. The featherweight "Raven" altimeter was selected to be used and the setup that was given by the company is displayed in Figure 29.

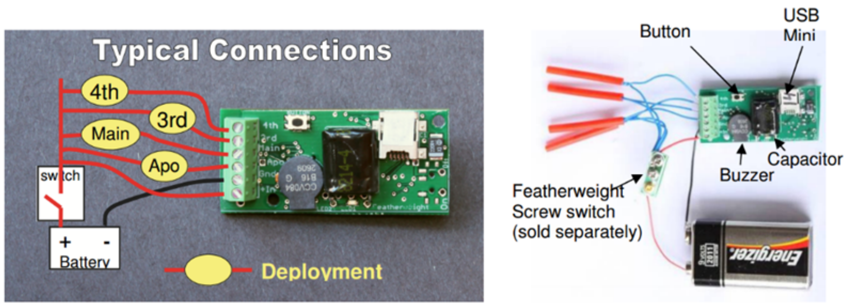


Figure 29. Setup of Raven Used in Parachute Deployment [23]

Pretests were conducted to verify setup and function of the altimeters as deployment mechanisms. The Ravens demonstrated reliability for both launches that took place. The setup for testing the altimeters as situated on the board is shown in Figure 30. The connections that relayed the output of the altimeters to where the ejection charges would be located are shown in Figure 31. After the tests were verified, the devices were installed within the vehicle. Precaution was taken to ensure that the devices could survive launch.

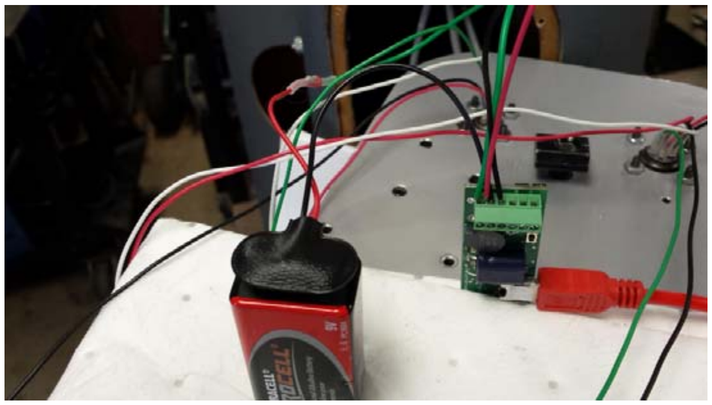


Figure 30. Wiring for Raven

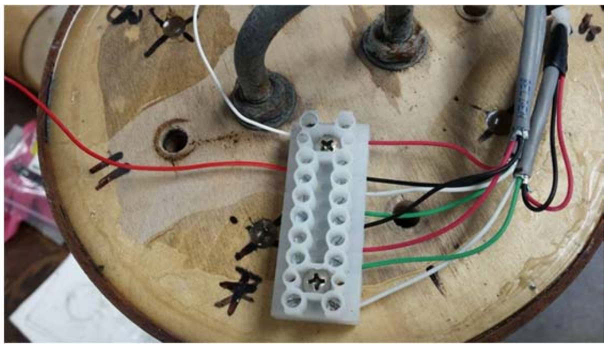


Figure 31. Connection from Output of Raven to Deployment Location

Confirmation that the altimeter performed correctly can be found in the log file of the measurements recorded by the Raven; which included the voltage applied to the black powder charges. It should be noted that the shared lead with the battery is in the positive socket of the Raven. This causes the voltage across the terminals to indicate a value between 8V and 9V, until the preset deployment scenario is met and the deployment signal is sent, when the voltage drops to zero. Viewing the log files is one of the most effective ways to determine if the Raven performed as expected.

III. TESTING

After making improvements on the vehicle, test launches were conducted to evaluate design and draw conclusions. Multiple launches took place over this thesis with the most significant being the final two launches.

A. OBJECTIVES

The most recent test for the vehicle was April 2, 2016 at the Friends of Amateur Rocketry launch site in the Mohave Desert near Tehachapi, California. The purpose of this test was to launch, demonstrate control, and recover the rocket. The objectives for the launch were as follows:

- **Test Objective 1:** Activate and record GPS/INS system, recover flight generated data for use in evaluating future guidance system
- **Test Objective 2:** Control fin actuations using Arduino command sequence
- **Test Objective 3:** Record vehicle response to engaged servos (camera recorded views)
- **Test Objective 4:** Clean separation; including the raceway connection
- **Test Objective 5:** Parachute deployment and rocket recovery

Test objectives were established prior to developing the Gantt planning chart for advancing the vehicle development. This allowed for focused efforts directed at set objectives.

1. Launching Descriptions

Explicit planning and descriptions of the adjustments needed to fulfill the objectives are included in the advancements of the rocket section. A brief overview of the system improvements for the April 2 launch is provided in this section. The flow chart for the control system for the vehicle is shown in Figure 32.

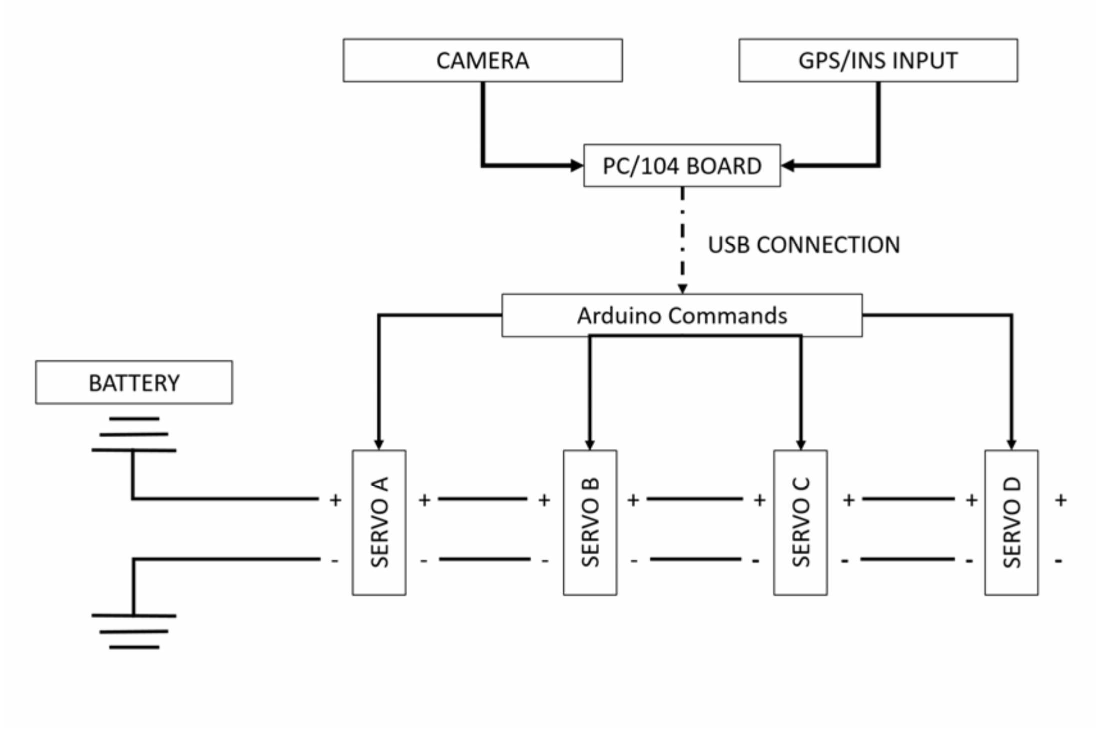


Figure 32. Flight Control System Flow Chart for Launch

Control of the vehicle was through a programmed sequence of events on the Arduino. The commands were then relayed to actuate the fins. The commands were preloaded onto the Arduino computers and then logged using the Arduino and the INS/ GPS system on the PC/104 board. Creating two independent and redundant sets of logged data. For the launch, a camera was installed on the parachute electronics bay to fly with the rocket, and record the response of the fins to the commands from the Arduino. The evaluation of this response fulfils the correlating objective.

Takeaways from the control system flow chart for the launch show the number of servos for the system which will eventually be installed. For the launch there were only two of the servos installed to mitigate risk. Each of the servos controlled one fin independently.

Test objective four addressed the clean separation point in the rocket. The required separation of the rocket is pivotal for not only deploying the parachutes for the development vehicle, but also for being able to control the submunition deployment portion of the launch in the future. The clean separation was planned to be accomplished at the USB connection, the same point as the nose cone and parachute bay connection point.

Included in achieving test objective four and test objective five was improving the design and installation shear pins. Shear pins were drilled through the parachute bay and nose cone coupler connection in prior launches, but the separation was not clean and damaged the rocket as separation occurred. Therefore, 2.5 cm^2 of brass shim stock, 0.0154 cm thick, was used to sheer the sheering pins so that there would be no scrapping or harm to the body. The metal and phenolic tubing were both sanded and prepped to create a flush connection and for a good bond to improve repeatability of separation. Prior to traveling to the launch, the modified structural design was evaluated by firing an ejection charge with the assembled system to test the raceway and sheer pin separation.

In order to test and then to verify performance of control system, two launch and recovery events were attempted. The testing included the execution of two preprogrammed fin actuation sequences. The approximate stages of the fins during the first flight are shown in Figure 33.

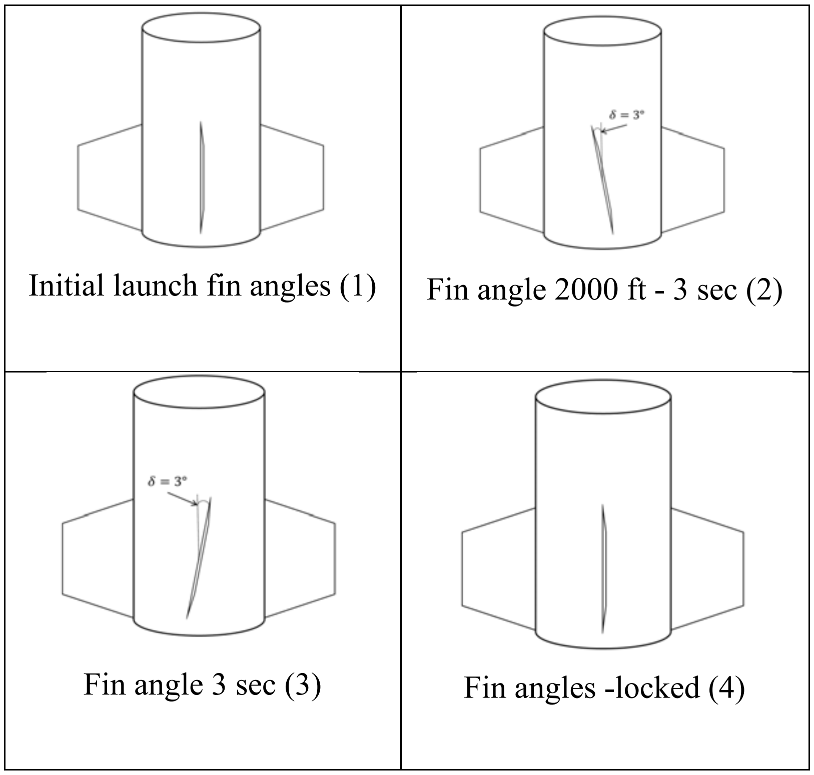


Figure 33. Programmed Fin Oscillation During Flight

Each of the fin actuations took place during the allotted time space. For each angle displacement, the opposing fin would rotate such that a roll rate would be induced upon the vehicle. The duration of each fin rotation was planned for an allotted amount of time. Test flight one executed an induced roll rate and then a counter spin command to return the roll rate to near-zero.

Test flight 2 had a similar command sequence, but the fins actuated in a coordinated manner. The command requested was intended to cause a pitch command for the rocket which was then going to be straightened by returning the fins to an aligned position. Changing the two input commands not only verifies repeatability but tested the integrity of the fin hardware and also the structural stability of the rocket.

B. LAUNCH ACTIVITIES

Originally targeted for March 5, the launch of the modified vehicle commenced on April 2, at the Friends of Amateur Rocketry launch site in the Mohave Desert. The day was successful in being able to launch and recover the rocket twice. The planned objectives and the correlating launch objectives are contained in Table 7.

Table 7. Test Objective Fulfilment from April 2 Launches

Test Objective	**Test Scenario**	**Launch 1**	**Launch 2**
1	Activate and record GPS/INS system, recover usable data for future guidance system	**X**	**X**
2	Control fin actuations using Arduino commands; executing roll and pitch maneuvers	**X**	**X**
3	Record vehicle response to engaged servos (Camera recorded views)	**X**	**X**
4	Clean separation; including the raceway connection	**X**	**X**
5	Recover the rocket	**X**	**X**

Each of the essential objectives were addressed to some degree, and results were achieved. In order to have the most use of results from the data recorded, certain measurements were needed of the rocket. These measurements are shown in Table 8.

Table 8. Rocket Measurements for Launch

Area	**Weight (lbs/kg)**	**Length(inches/m)**
Base with fin	33.2/15.1	-
Nose cone	18.5/8.4	-
Parachute tube	6.0/2.7	-
Parachute 1	3.2/1.5	-
Parachute 2	4.2/1.9	-
Kevlar	1.0/0.45	-
Motor casing	2.8/1.3	-
Overall length	-	137.5/3.49
Length to COG	-	57.4/1.46

Along with weight and COG measurements, different specifics were associated with each launch. Some of these are provided in Table 9. The specifics for the motor are contained in Table 9.

Table 9. Specifics for Each Launch [24], [25]

	Launch 1	**Launch 2**
Motor	Cesaroni M3100	Cesaroni M1540
Wind	<5 mph	8 mph
Temp (℃)	18	20
Conditions	Clear	Clear
Launch Rail	NPS (1515 steel)	NPS (1515 steel)
Config	Roll Program	Pitch Program

1. Launch Photos

The rocket was successfully launched and recovered twice during the day. The turnaround time from first launch recovery to second launch was one hour and forty-five minutes. This shows how the rocket's modularity has improved, extremely advancing the rocket by simplifying the steps needed to reach launch readiness. Figure 34, Figure 35, Figure 36, Figure 37, Figure 38, Figure 39, Figure 40, Figure 41, and Figure 42 were all taken at the launch site and show the setup, launch, and recovery of the rocket.

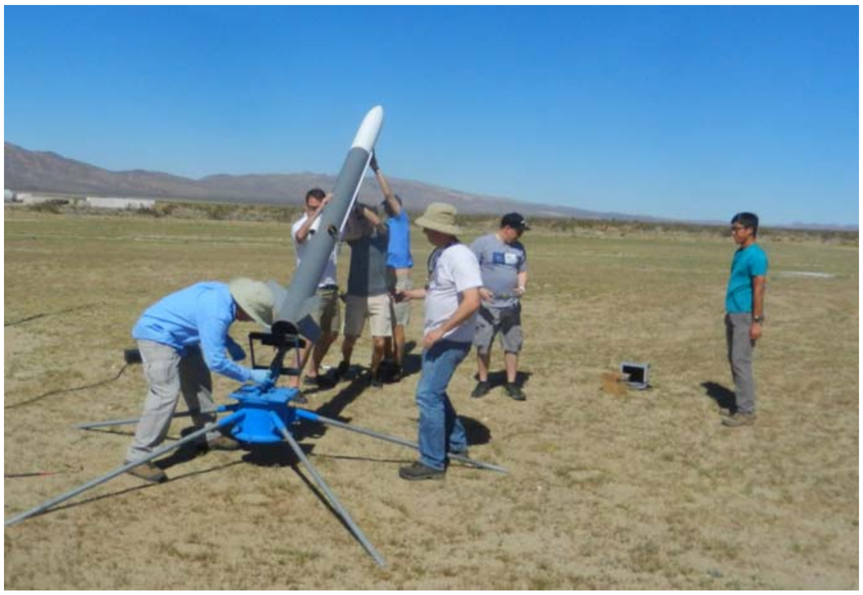


Figure 34. Setup for Launch Listed Left to Right: Robert Wright, Fletcher Rydalch, Andrew Chaves, Brent Aldridge, Chris Brophy, Lee Van Houtte, and Wee Yeow Lim

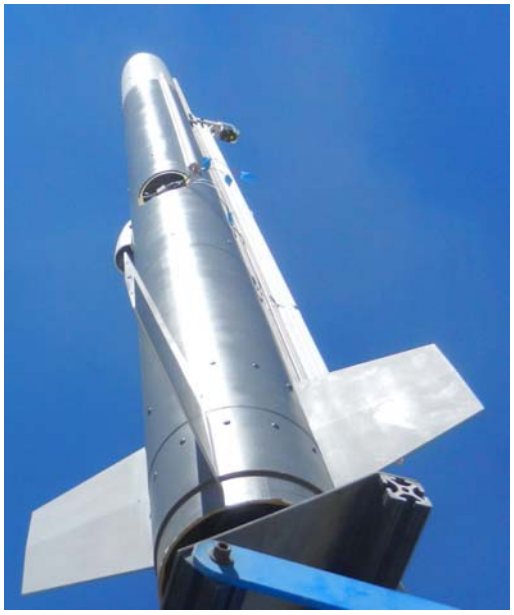


Figure 35. Rocket on Launch Rail

Figure 36. Rocket Prior to Launch

Figure 37. Launch of Rocket

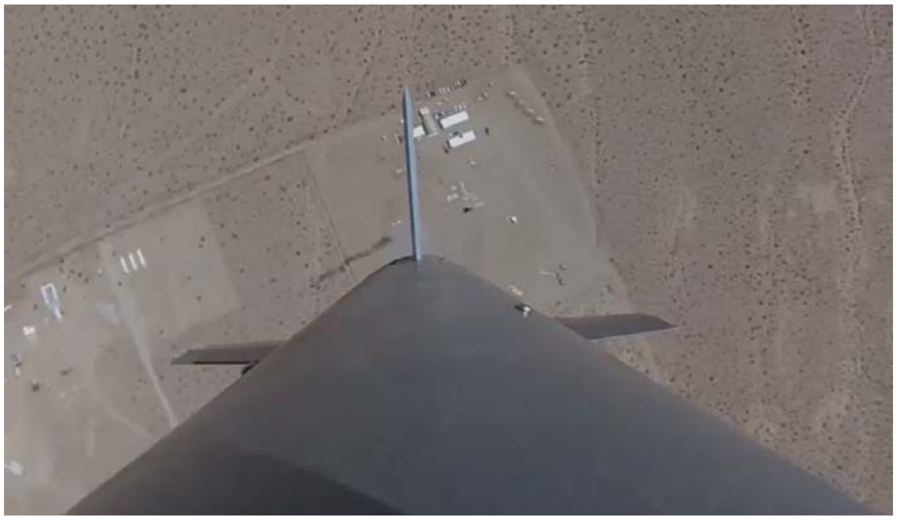


Figure 38. Pre-actuated Fins (0° Deflection)

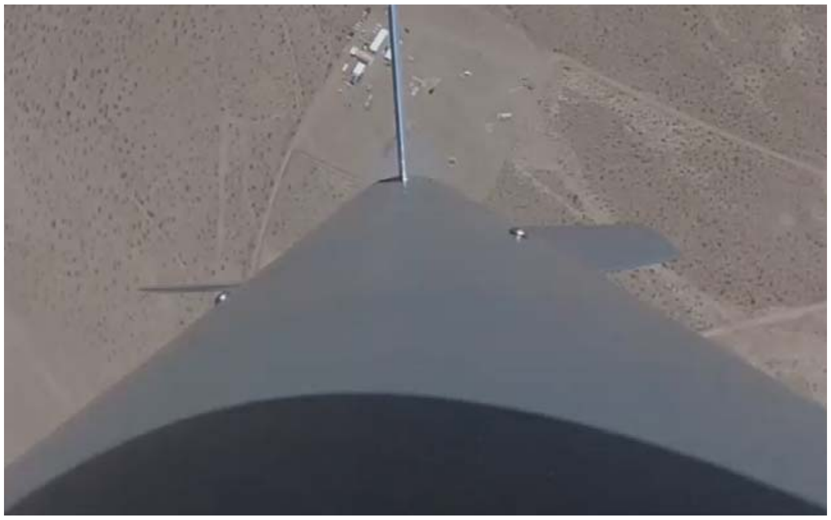


Figure 39. Fin Position One (±3° Deflection)

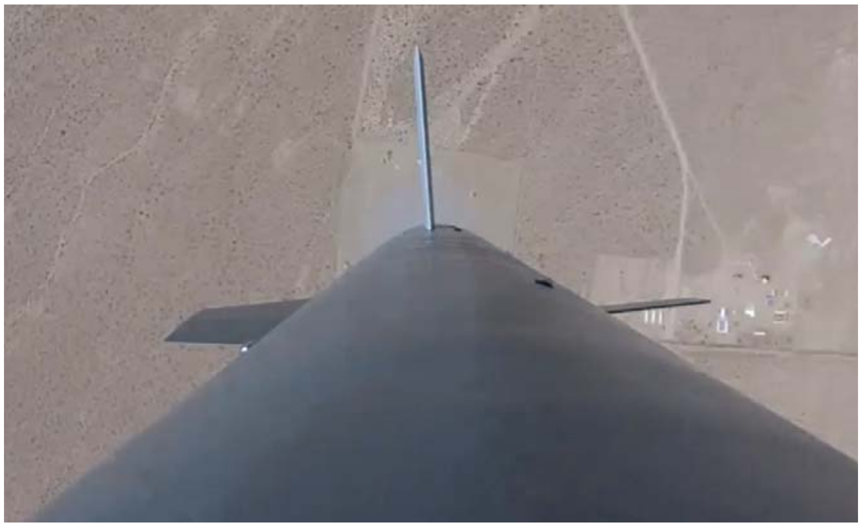


Figure 40. Fin Position Two (±3°)

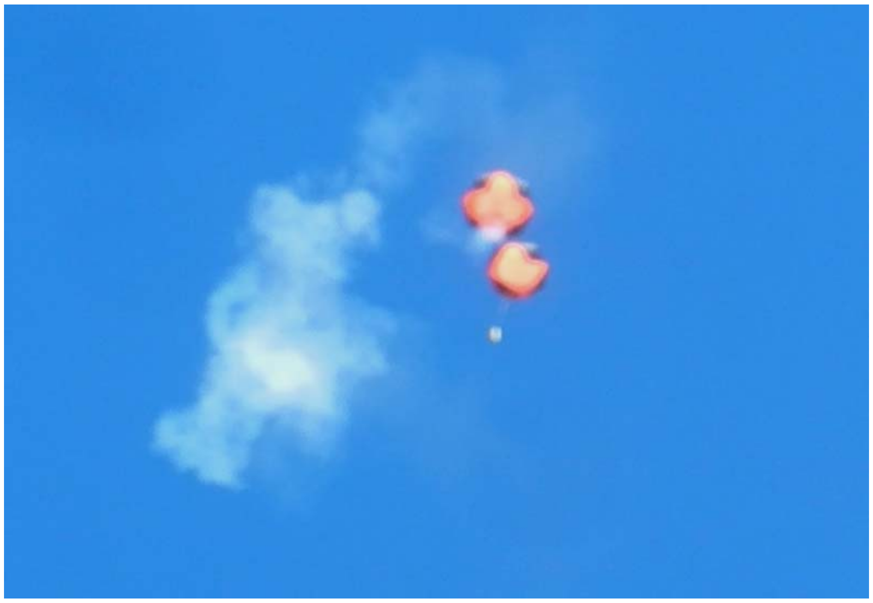


Figure 41. Parachute Deployment

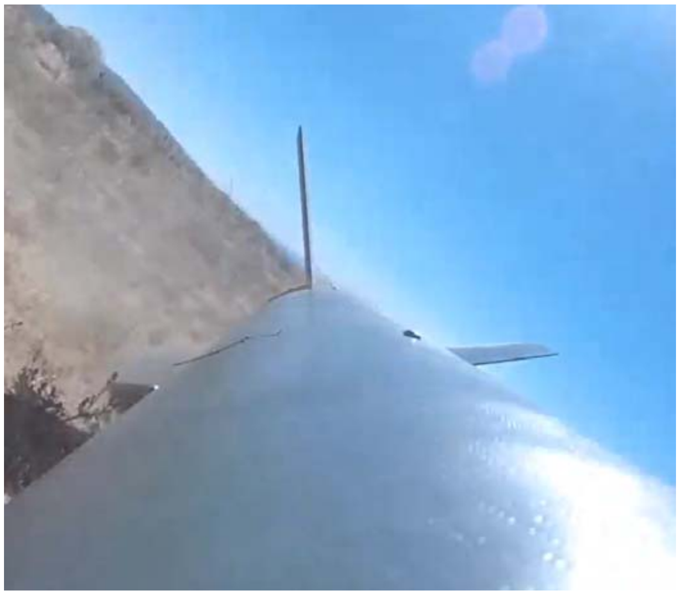


Figure 42. Landing of Rocket

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DATA COLLECTION AND RESULTS

Successfully launching and recovering the rocket resulted in multiple sets of data being recovered. The data had logged the performance of the rocket. One set of data was recorded in parachute electronics bay through the Arduino microprocessor, the other was recorded on the PC/104 in the nose cone through the INS/GPS system. There was also data recorded by the Featherweight altimeters, located in the parachute electronics bay.

A. FEATHERWEIGHT ALTIMETER RESULTS

For each of the launches, two Featherweight altimeters were calibrated and set to record data and then send ejection signals to the charges set in the parachute bay. The altimeters were each able to obtain reliable data for each launch. Launch one was a success and the results of the launch as recorded by the Raven number one are shown in Figure 43.

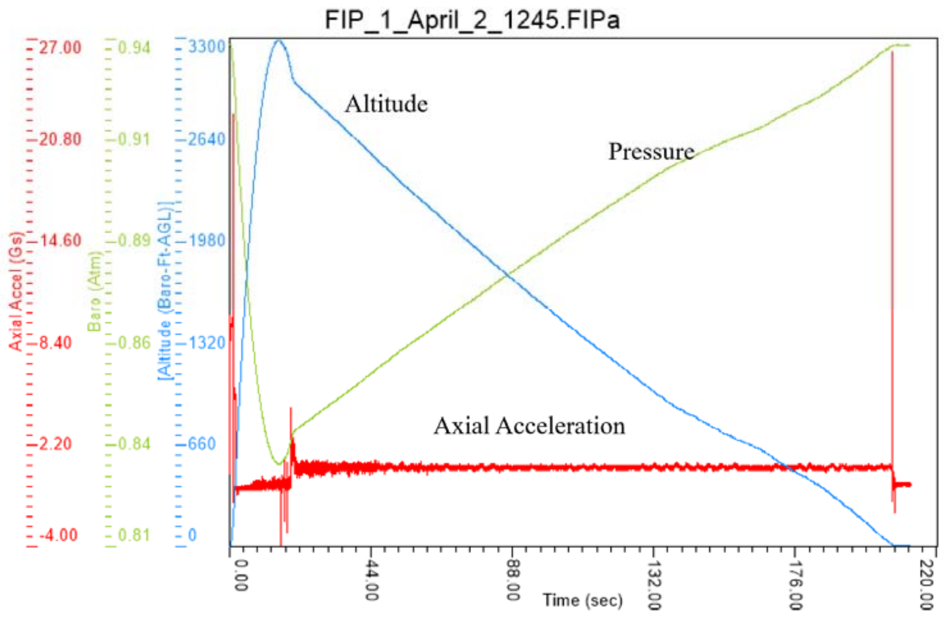


Figure 43. Results for Raven #1, First Launch

The rocket reached an altitude of 1006 m (3300 ft) high. The highest force recorded was nearly 21 g's of force, but it appeared to be an outlier. Upon comparison to the recorded forces by the second Raven there was a large spike only a couple seconds into the launch (the second Raven placed the spike as high as 46 g's). A zoomed in view of the same data is shown in Figure 44.

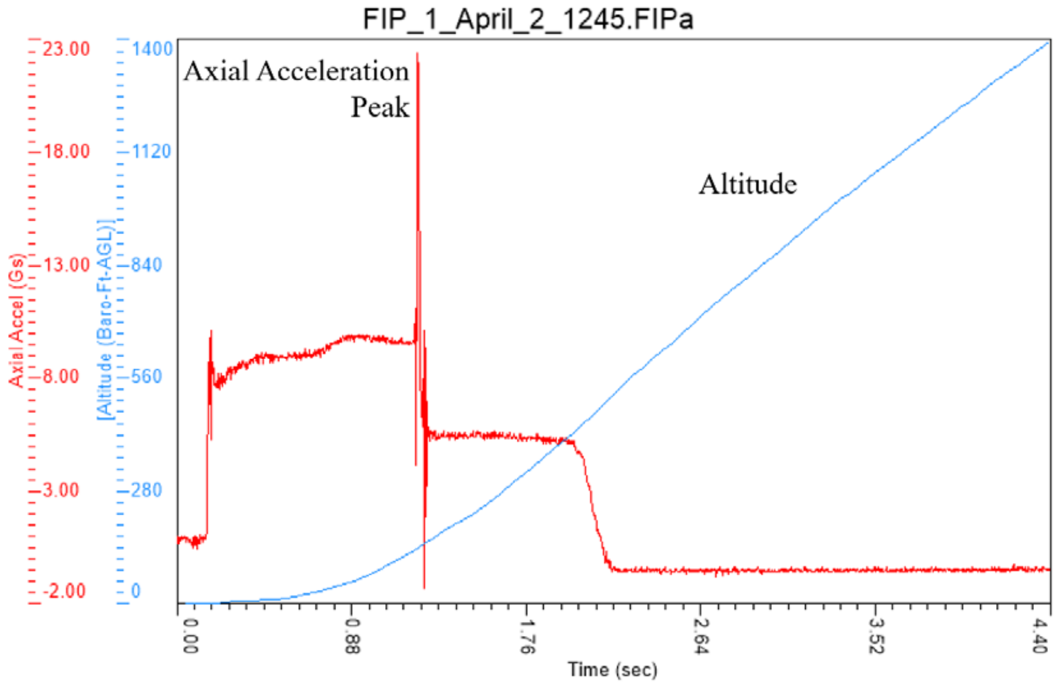


Figure 44. Results for Raven #2, First Launch (Zoomed In)

This zoomed in view does confirm that there appears to be a legitimate spike in the acceleration experienced by the rocket about 1 second into the launch. The maximum value is greater than 21 g. Comparable to the Raven number one, the results for the second Raven are shown in Figure 45.

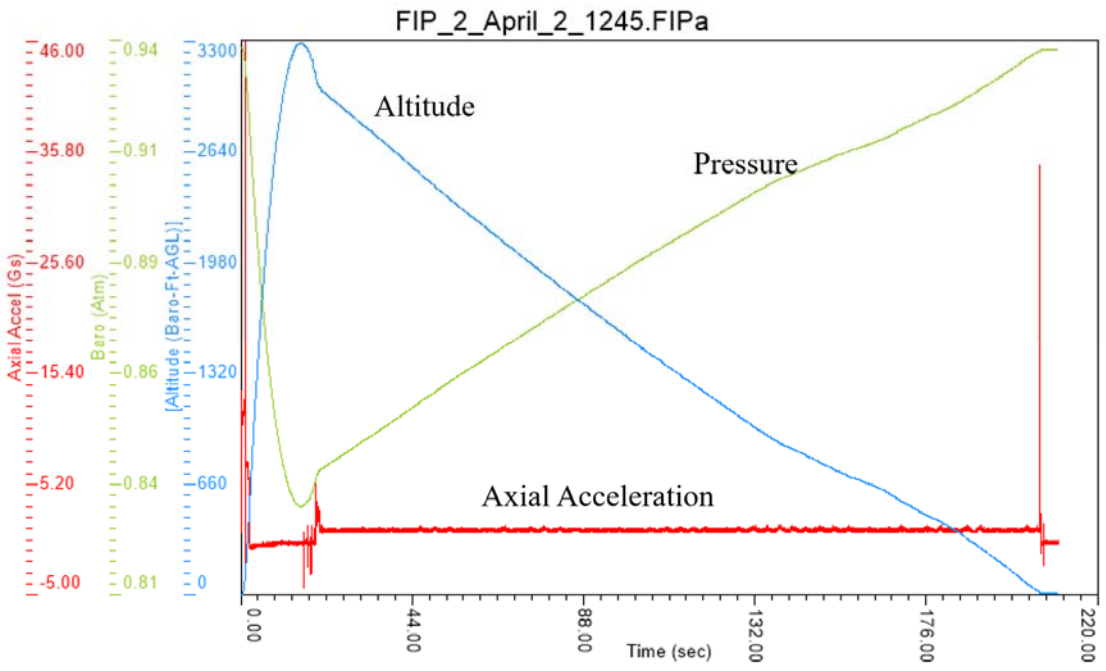


Figure 45. Results from Raven #2, First Launch

Very similar to Raven #1, the data confirms the rocket reached about 1006 m (3300 ft) and had a steady descent from 19.8 seconds into the launch until it returned to ground. The steady fall alludes to the time under successfully deployed parachutes.

In the second launch, both altimeters performed as expected. The launched differed from the first in the installed motor and the actuating commands. Based on the specifications of the motor used, it was suspected that the rocket would launch higher and faster. The second launch also had two featherweight altimeters. The results taken from Raven number 1 are contained in Figure 46. Both of the altimeters had matching data.

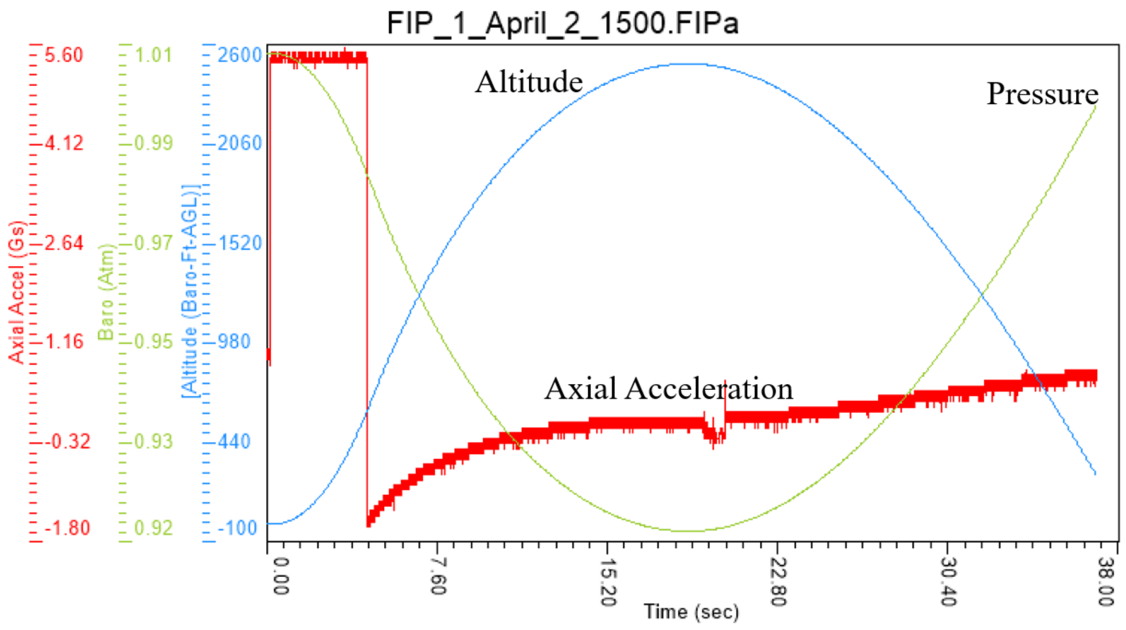


Figure 46. Results Recorded by Raven #1, Second Launch

It is apparent that the second launch only reached 792.5 m (2600 ft) while the first launch went to 1005.8 m (3300 ft). There are two likely reasons for this, but the lack of height primarily stemmed from the use of an incorrect nozzle geometry. The nozzle was fitted for the first motor and not switched for the second motor. The first one was 2.54 cm diameter throat nozzle and the second one was supposed to be 1.9 cm diameter throat. Associated with the lower thrust levels, the rocket only experienced a maximum of 5 g during flight, dramatically decreased from the first launch. Along with the alternate motor, the second launch experienced different fin actuation than the first launch.

B. RECORDED DATA

There were two readings of inertial data taken during each flight. The more telling data was recorded in the nose cone by the MicroStrain sensor. The next set of data was measured in the parachute electronics bay by the sensors attached to the Arduino microprocessor.

1. First Launch

One of the additions to the rocket included the installation of the PC/104 computer board in the nose cone of the rocket. Included in the nose cone was a GPS/INS system; installed to accurately and reliably record flight data that could be used to guide the system in the future. The devices were successfully installed and tested, as alluded to in Section II. The flight included two launches and two different flight profiles. The first launch executed a spin maneuver at about 609.6 m (2000 ft) high in the air. The results of that launch as recorded by the GPS/INS system installed in the nose cone of the rocket are found in Appendix D. Raw Data from Flight. The INS/GPS system confirmed the momentarily jump in acceleration experienced. The rocket experienced 21.9 g (215.3 m/s^2) of acceleration at approximately 1.08 seconds into the launch. The spike in acceleration lasted for less than 0.04 seconds (25 Hz) because the spike was only read in one data line. The angular rates recorded during flight are shown in Figure 47. It is clear that there are apparent variations throughout the flight, especially in the regards to the X angular rate

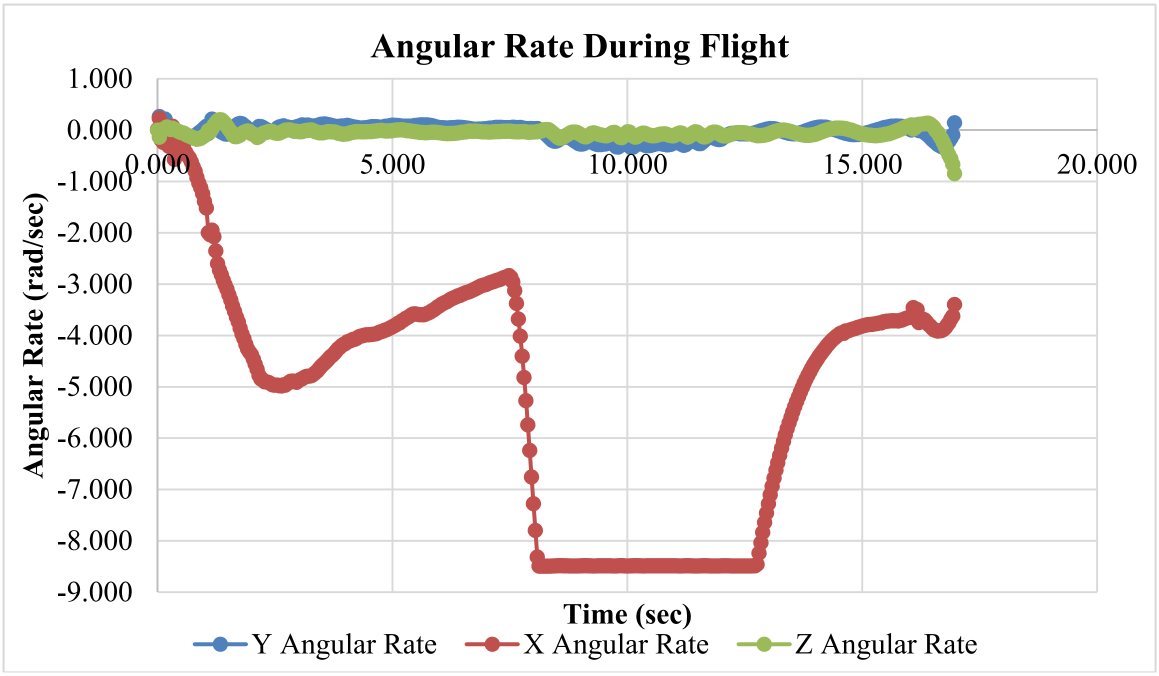


Figure 47. Angular Rates for Three Axis, First Flight

The data reveals that there was an obvious change in the X angular rate (rad/s) during the launch of the rocket where the angular rate increased from 0 rad/s to -8.5 rad/s. The data shows that there was a gradual increase in angular rate at the beginning, then a steadying of the rocket followed by a sharp increase from -2.8 rad/s to -8.5 rad/s, the maximum achieved. The data recorded through the IMU on the Arduino is shown in Figure 48. The data is opposite from the date recorded on the INS/GPS sensor because the orientation setting for angular rates were set opposite for the X angular rate direction.

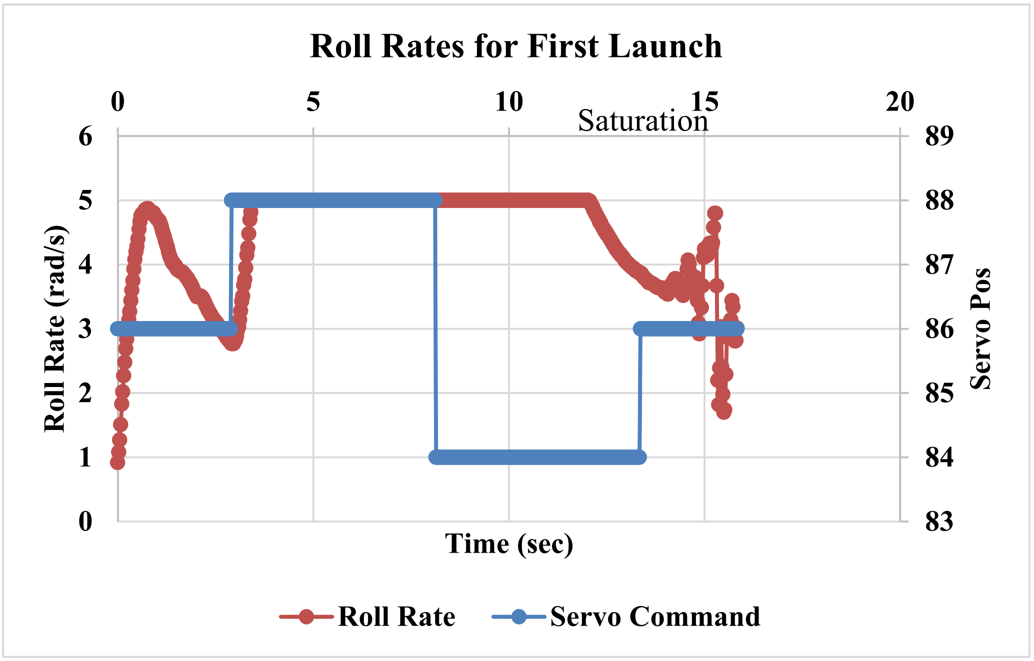


Figure 48. Angular Roll Rate Recorded by Arduino, First Launch

The data displayed shows the rate at which the vehicle responded to the servo commands. The first command to shift from the vertical position (88°) to the first position (86°) was given at about 2.9 seconds into the launch. It took about 0.546 seconds for the vehicle to surpass the 5 rad/sec roll rate. It also was discovered and later verified in the second launch that the Arduino IMU sensors cannot record roll rates greater than 5 rad/sec. Because of this limitation the response of the rocket to the change in servo position about half-way through launch goes unobserved in the data, until the roll rate decreased to less than the 5 rad/sec. The ending oscillations of the angular rate corresponds with the parachute deployment.

The IMU/GPS sensor that was installed in the nose cone of the rocket did successfully measure roll rates and accelerations throughout the launch, this showed that the IMU/GPS could function adequately with the PC/104 as the communicating equipment. It also showed that the rocket did respond timely to the input commands from the Arduino, and relayed to the fins. The most revealing portions of recorded data is shown in Appendix D. Raw Data from Flight. Overall, the data recorded told the story that the first launch was great success in regards to preserving electronics and capability of the rocket. It was also successful in being able to observe the response of the rocket to attempted controls.

2. Second Launch

In the second launch, the input for the commands were changed from executing a spin maneuver to a pitch maneuver. The plan was to actuate the fins in a motion that would cause the rocket to turn and then to align back to center. The commands were planned to begin at the end of rocket acceleration, like the first launch. The data points that correspond to the same data taken in the first launch are in Figure 49. The Y angular rate of the rocket corresponds to the pitch while Z angular rate corresponds to the yaw of the rocket. The figure shows varying oscillations for the Y and Z angular rates, this may be a result of a compromised set screw installed between the servo horn and the fin axel, set to diminish extra tolerance allowance.

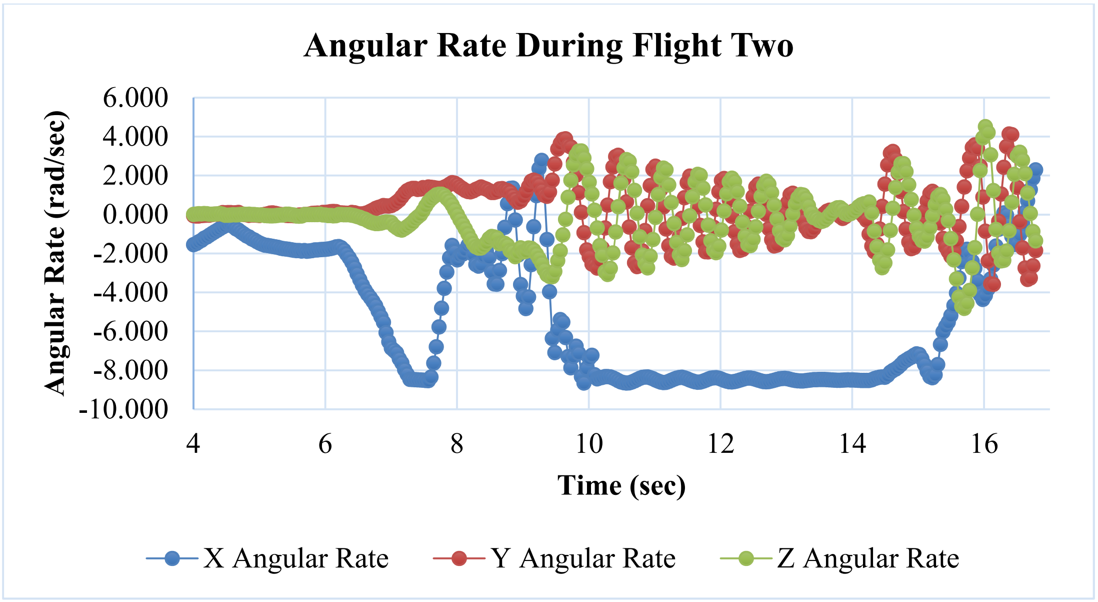


Figure 49. Angular Rates for Three Axis, Second Flight

During the second launch data was recorded by the IMU on Arduino along with the INS/GPS sensor. By having data, confirmation of successful installation of the Arduino was realized. The IMU attached to the Arduino was successfully mounted so that it could run the whole time. The data recorded by the Arduino is shown in Figure 50.

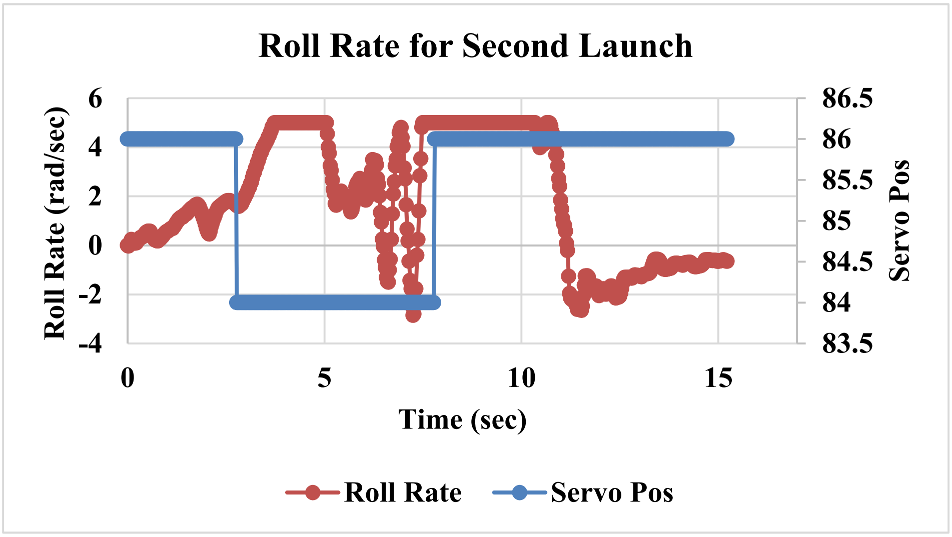


Figure 50. Angular Roll Rate Recorded by Arduino, Second Launch

The second launch did not see the rocket respond to input servo commands entirely as expected. The servo positions included only two movements, from starting position of 86°, and the middle position of 84°. With the intention being that servo 1 and servo 2 would mirror each other and cause the rocket to perform a pitch maneuver. Based on the data and the video that was recorded of the control surfaces, the servos began its commands prior to motor burn out. Combining this aspect of the flight profile with the incorrect thrust levels, the rocket was not able to achieve the expected thrust levels and altitude. Correct commands were given through the Arduino, but did not result in a correct response for the entire flight. Upon review of the video, it is apparent that one of the locking collars of the fin shafts came loose, causing the rocket to go out-of-control. The out-of-control flight included small corkscrew motions at the tail end of the motor burn and up until the parachute was deployed. As a result, the pitch and yaw readings were erratic oscillations; as shown in the Y and Z angular rates in Figure 49.

Along with numerical data to record the launch performance. Visual recording of the flight was also attempted. The rocket did not fly optimally once the fins began to actuate. Photographs of the flight of the rocket are shown in Figure 51.

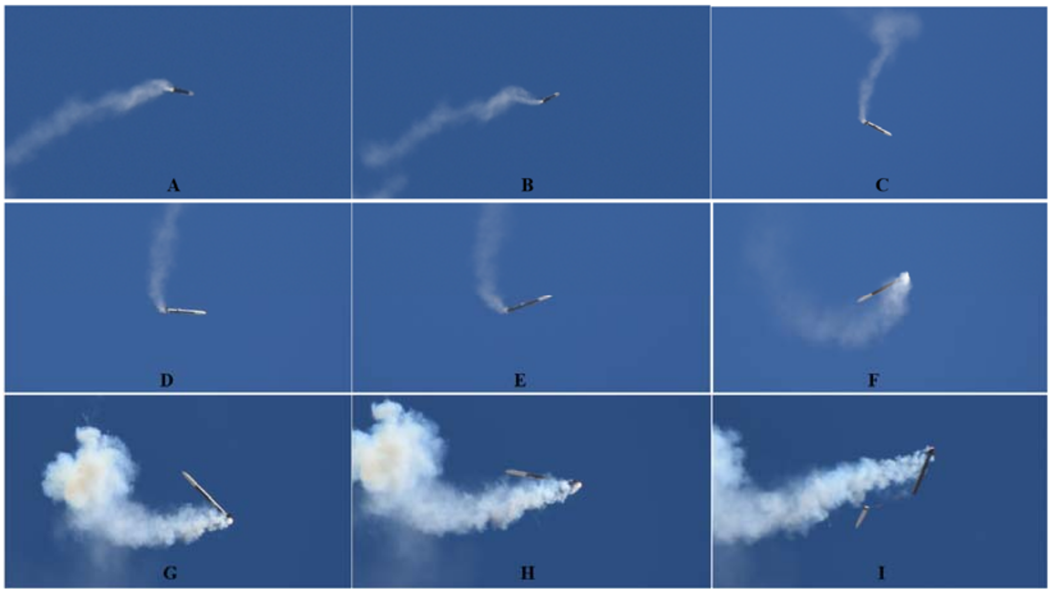


Figure 51. Photographs of Flight of Rocket from Second Launch

While still burning the rocket motor, the beginnings of actuation caused the vehicle to begin a pitch maneuver and turn the rocket. Once the maneuvering began, the fin began to flutter and the rocket began to corkscrew as pictured. The loose fin and random movements may have been the cause of the corkscrew. The corkscrew flight of the rocket was confirmed in the frames recorded by a side mounted camera. Frames showing fin actuation are shown in the Figure 52 and Figure 53.

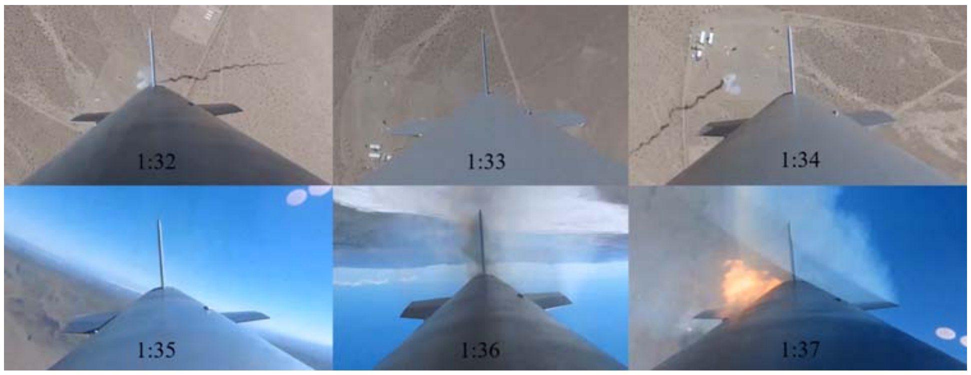


Figure 52. Fin Facing Camera during Corkscrew Motion (1/2)

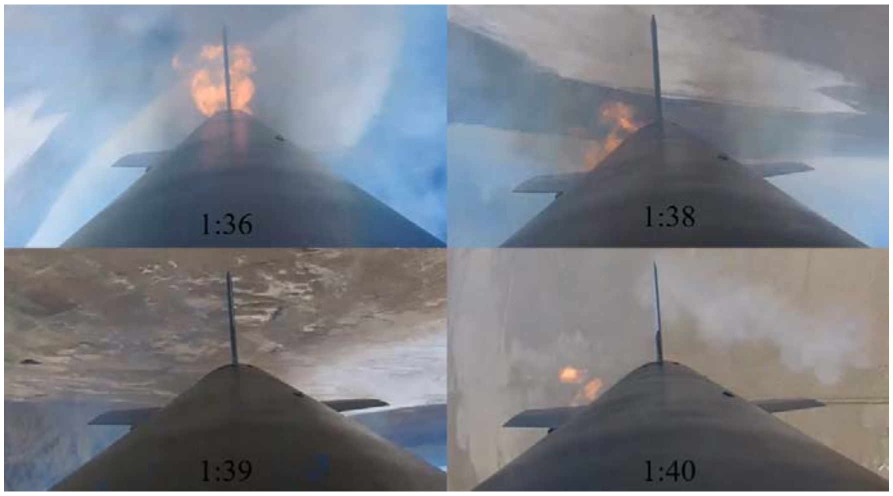


Figure 53. Fin Facing Camera during Corkscrew Motion (2/2)

The flutter which occurred for the left fin (based on the figures) contributed to the lack of control. The erratic performance was due to a loose connection between the servo horn and actuating rods. Increasing the shaft diameter and designing a better technique to lock the shaft to the control arm should eliminate this failure mode from future launches.

V. SUMMARY AND CONCLUSIONS

Each of the objectives for advancing the vehicle were met. The design, materialization, installation, and overall advancements of individual subsystems provided a vehicle that was successful in multiple testing scenarios. A manifestation of the modularity and reusability of the vehicle occurred during the April 2, 2016, test when the rocket was recovered and relaunched in only one and a half hours. The modularity and improvements of the system stem from the creation of a clean separation design capable of mitigating damage. Along with modularity advancements, a new GPS/INS system was installed with supporting equipment to communicate with the PC/104 computer; this system accurately recorded all the vehicle's forces and rates experiences. The design of a new thermal management system for the nose cone was demonstrated and allowed valuable flight data to be obtained for future guidance and control development. The data confirmed that two active fins could be used as control surfaces to control the vehicle. Image and video recording showed that the fins were controlled effectively and that the selected servo motors were functional through multiple launches. The actuated fins successfully induced a desired roll rate during one test flight and a preliminary pitch command on a following flight.

A. FUTURE WORK

More in-depth suggestions and discussions on how the rocket can continue to be improved are included throughout this report; an overview of recommendations is included in this chapter. As outlined in the overall schedule of work, the mechanical designing and machining began in phase I. The work presented here must be included and built upon going forward; building upon past designs should continue through the entirety of vehicle development.

The design and development process revealed new challenges that must be addressed in continued efforts. The hardware and machining should include machining a new fin design to the exact dimensions needed for press fits to be usable. Building the can will require time and incur more cost, but areas where tolerance must be minimized have been found and—the most pressing of which is within the fin can and servo connections. Hardware should also be worked on to activate all control surfaces of the vehicle.

For future software development, control of the servos needs to shift from the Arduino Uno board to the PC/104 board installed in the nose cone. The code used in this launch was nearly the maximum size the Arduino board could handle. The USB chord that was installed and separation tested on the raceway of the rocket should be used to transfer data between the servos and the PC/104 board.

To be useable as a counter UAV delivery vehicle, the rocket needs to be able to be self-guiding. Initial development of guidance was added in the GNC section, but future work needs to include interdisciplinary efforts to guide both the rocket and future submunition deployments. Development should include the recorded data from previous launches in the GNC modeling. Image processing also needs to be investigated to provide communications and guidance between the nose cone of the rocket and submunition deployments.

APPENDIX A. GNC ALGORITHM DEVELOPMENT AND SIMULATION RESULTS

This section of the paper documents the experimental method and notes the numerous problems, solutions, and lingering issues encountered during development. It is included to maintain completeness in the report. The GNC solution and much of this appendix was constructed by Maj. John Dirk, USA, and LT Travis Hartman, USN, in phases with the assistance of Vladimir Dobrokhodov [13].

1. Overview of GNC Algorithm

The first portion of the GNC solution, navigation, was approached by constructing a converter that converted GPS-style geodetic outputs into a local tangent frame. After converting both rocket and target locations into a North, East, Down (XYZ) tangent plane, calculations were be made to support navigation.

A previously constructed 2-D pursuit guidance model was modified via the addition of non-linear matrices for turns in the XY, YZ and XZ planes to provide the initial 3-D Guidance solution () [25]. The initial model was crude, involving a constant velocity point mass with acceleration limitations imposed independently in each axis and no flight termination criterion. The decision to use a heavily modified existing model rather than constructing a model from the "ground up" precluded testing each incremental addition and made it considerably more difficult to quickly identify problem components. Thus, a considerable amount of time was devoted to simplifying and vetting the model piece-by-piece.

To facilitate further development the geodetic frame portion of the model was removed in favor of a local tangent plane, north-east-down (NED), reference for navigation and the calculation of bearing to target and target elevation angle. Later the geodetic calculator was reintroduced to support the complete model.

The initial guidance solution involved the continuous application of calculated angles to calculations of three turn rate commands ($\dot{\Phi}$ [XY], $\dot{\Theta}$ [XZ], $\dot{\Psi}$ [YZ]) applied with gains to the missile state through the basic $\boldsymbol{\omega}$ x $\mathbf{V}$ relationship to align the missile with the target. It was quickly realized that only two turns were necessary to affect a three dimensional intercept and the phi-dot command was removed. Not only was phi-dot unnecessary, but commanding both psi and phi resulted in interference during some intercepts and the missile effectively "fighting" itself to chart an efficient course.

Other complications involved the use of a vertical launch and singularities encountered near the cardinal headings. The rocket initially would only consummate intercepts in the northern hemisphere. A single plane analysis was employed to isolate and better understand the specific issues encountered during certain intercepts. Due to the vertical launch condition, the missile's initial heading is undefined, which resulted in the bearing error calculation rapidly transitioning between $\pm\pi$. Rather than tilting the rocket, this singularity issue (a product of SIMULINK implementation of psi-dot), was eventually minimized via an unwrapping function that would run continuously between $\pm\infty$ to preclude rapid sign convention changes between $\pm\pi$ (the imposition of feedback with high gains was first attempted unsuccessfully). Several iterations were required to determine in which axes, and at what point in the calculations to unwind the tangent function via MATLAB code block to prevent a step change between two pi and zero as the rocket maneuvered through various headings. The initial 3-D implementation is shown in Figure 54.

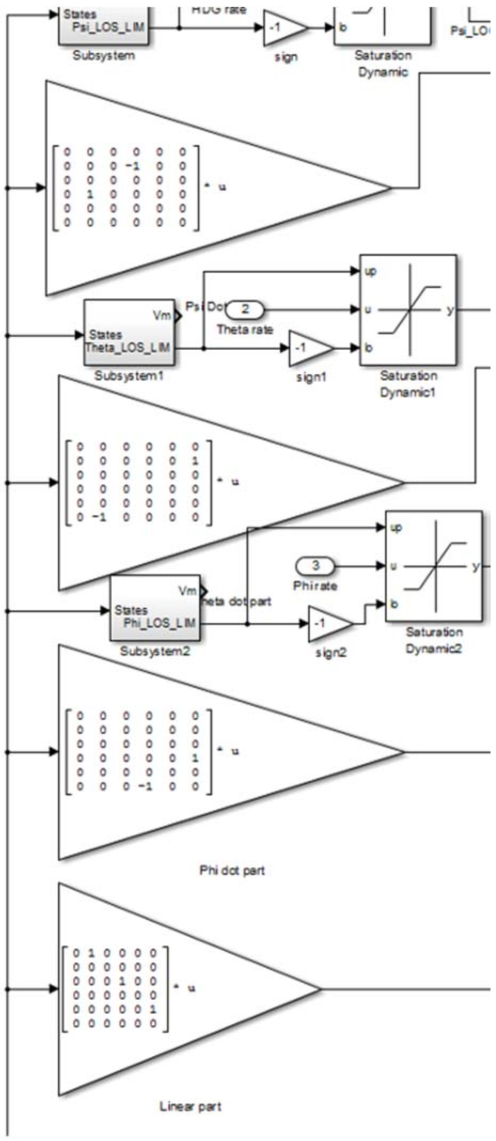


Figure 54. Initial 3-D Implementation

When done incorrectly, the rocket would favor a position theta-dot and pursue a sub-optimal path (for example: often preferring to turn 260 degrees vice 100 degrees). The sub-optimal path is shown in Figure 55. When done correctly, the problem was minimized but not removed and remains a fundamental limitation of employing pursuit guidance in a vertically launched missile without using quaternions.

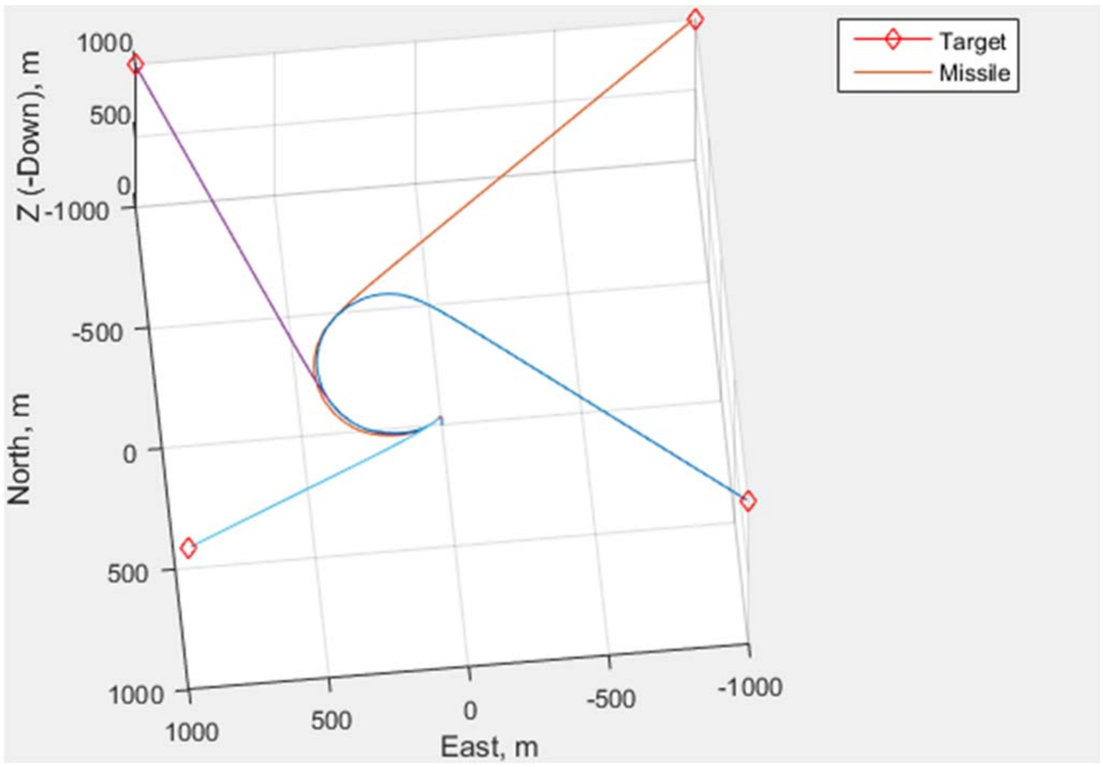


Figure 55. Incorrect Unwrapping and Sub-Optimal Turns

In conjunction with the now rudimentary but functioning pursuit guidance and navigation models, the initial construction of a stabilizing autopilot was attempted and the first draft of a plant model commenced. The first plant attempted to model basic actuator and gyro dynamics but was built without actual lab data on either system and functioned as a stand-in, with the intention that it would be modified significantly once a full aerodynamics based, linear-time-invariant model was available.

In order to project the force of gravity onto the air vehicle, the gravity vector for earth was transformed from the navigation frame to the body frame using the Euler angles of the vehicle. This was a three axis rotation that used the transpose of the body to navigation frame [26]. The result was then multiplied by magnitude of gravity near the earth's surface (g). Yaw angle has no effect on the solution.

$$\begin{bmatrix} \ddot{\boldsymbol{x}}_{\boldsymbol{b}} \\ \ddot{\boldsymbol{y}}_{\boldsymbol{b}} \\ \ddot{\boldsymbol{z}}_{\boldsymbol{b}} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\theta s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 9.8 \end{bmatrix} = \mathbf{9.8} \begin{bmatrix} \boldsymbol{-s\theta} \\ \boldsymbol{s\phi c\theta} \\ \boldsymbol{c\phi c\theta} \end{bmatrix}$$

During this period the navigation and guidance models had been integrated together, and the plant and first draft of the controller had been integrated together, but these two models had not been tied together into a single, complete GNC solution.

The next iteration of the guidance model, intended to remove the lingering negative effects of a pursuit guidance solution combined with a vertical launch and optimize the intercept path (as well as provide for future growth potential when the target is moving through space) was to institute a proportional navigation (PN) solution. Although a simple (ideal) PN solution would use the derivate dU/dt in its calculations of line-of-sight (LOS) rate, a real-world solution must consider the practical issues of implementing derivatives and avoid mathematical complications via the imposition of a low pass filter. While the filter increases the potential for variations in flight path during the initial turn towards the target it removed the greater risk of mathematical singularities. This adds the requirement of appropriately tuning the filter. Additionally, a Coulomb viscous friction block was added to compel the missile to pursue a greater lead angle during the intercept and keep the closing velocity oscillating around a small, nonzero number thereby reducing the number of zero crossings.

This iteration of the guidance model also imposed a fin lock command which employed a trigger mechanism that cued a countdown timer after the missile achieved a targeted acceleration in the x-axis. Following completion of the countdown, the fin locking command is removed and the control and guidance becomes active. This effectively would ensure that the boost-sustain rocket motor had burnt out prior to the missile guiding towards the target; acting as a safety to ensure that separation was achieved from the launch platform and negating any issues with thrust offset that might couple into guidance commands. This delay also served to ensure that sufficient velocities were present when the guidance law activated to remove issues with derivative calculations.

The plant model was modified to include an accurate representation of the rocket motor thrust profile and the associated missile accelerated accordingly. Additionally, mass characteristics (Ixx, Iyy, Izz) were added, and soon thereafter the first aerodynamic coefficients $C_{m\alpha}$ and $C_{m\beta}$ (both 1.3 per degree) were provided. Thrust, gravity, and air vehicle characteristics were accounted for, but lift and drag were yet implemented.

The initial attempt at a PN solution in 3-D was essentially a failure due to errors in understanding that resulted in an incorrect implementation, shown in Figure 56. The rocket would drive itself to a specific LOS relative to the target, but not necessarily one required to intercept or even the opposite one, shown in Figure 57. Additionally, a corkscrew/spiral motion was present in the flight path, oscillations observed in the terminal phase just prior to intercept, and the rocket would not track on targets directly east or west of the launch position. This is shown in Figure 58. It is also shown in Figure 59. One major note, because the state equation implements acceleration demands in an inertial LTP frame the LOS turn rate had to also be calculated in the same frame of reference. The initial corkscrew motion was the result of GN model still not having a stabilizing autopilot/controller (it was essentially still a point mass that was being pulled in two orthogonal directions by two uncoordinated forces). The oscillatory motion in the terminal phase of flight was a result of the tuning of the derivative filters and also a function of closing velocity. The east/west guidance anomalies were due to the numerical issue of attempting to take ARCTAN(dZ/dX) with dX approaching zero. One course of action considered was a blended guidance solution that transitioned to PG when the target was located within a certain radial distance of a cardinal heading, but this concept was not executed/encoded into the model. However, further investigation and consultation determined that once the GNC models were integrated the mathematical anomaly would be corrected.

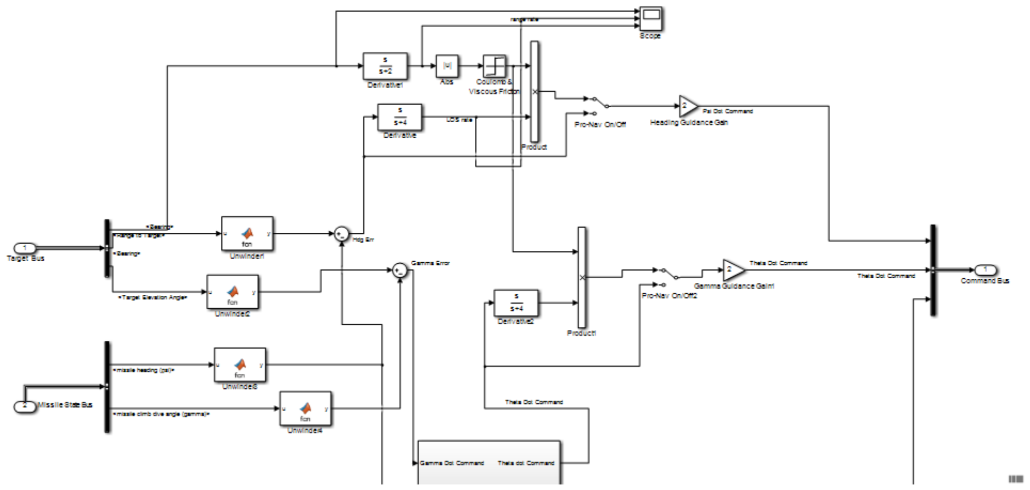


Figure 56. Initial SIMULINK PN Guidance Model

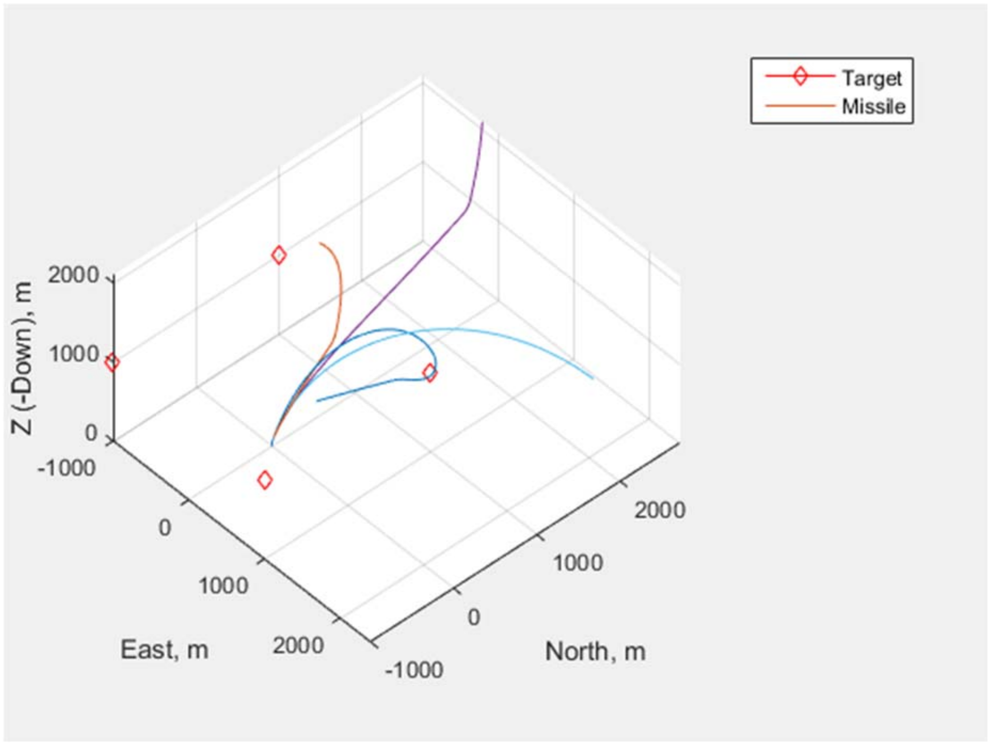


Figure 57. Resulting Intercepts from First PN Guidance Attempt

\

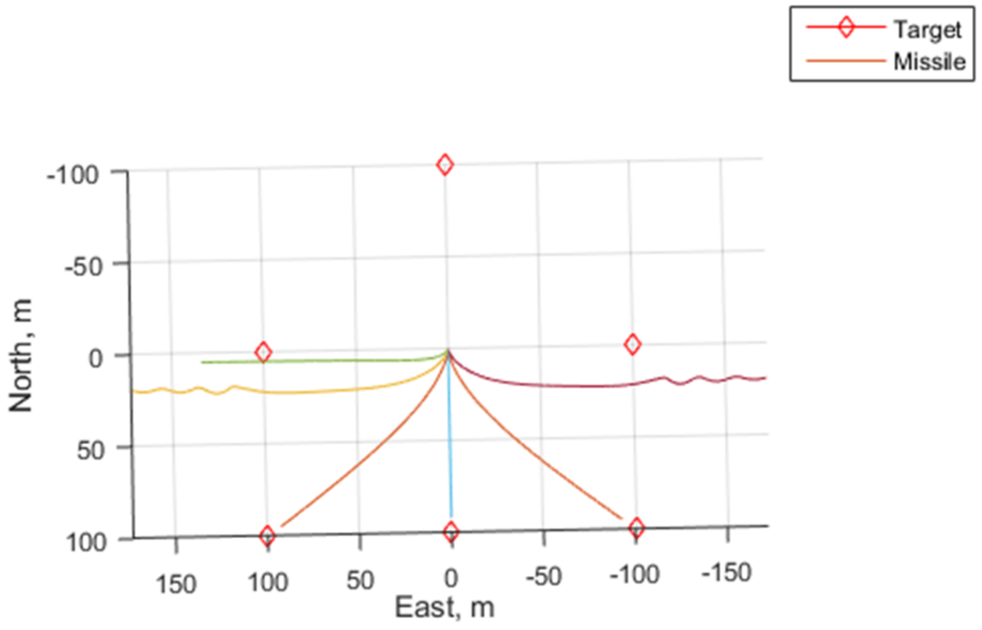


Figure 58. PN Guidance Model Induced Corkscrew and Cardinal Heading Intercept Restrictions (1 of 2)

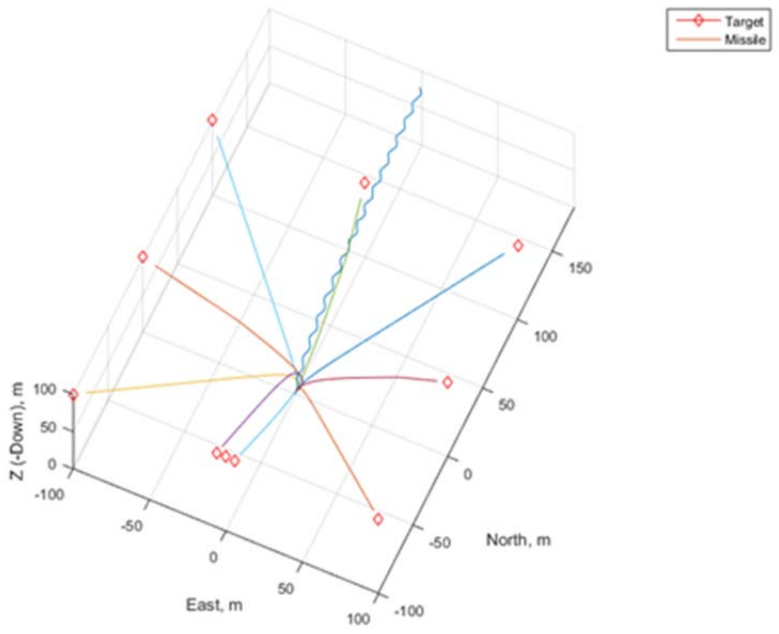


Figure 59. PN Guidance Model Induced Corkscrew and Cardinal Heading Intercept Restrictions (2 of 2)

Greater fidelity was added to the controller, essentially creating the first functioning representation of an autopilot rather than solely an approximation of actuator dynamics. It was decided that there required a separate roll channel PI controller and pitch/yaw PID controllers. These would eventually be tuned based on simulations run through the Plant (thus, filter tuning is dependent upon the fidelity of the plant itself…and as later determined, the rocket is considerably dependent on the quality of the filter tuning). Implementation of the pitch/yaw channels involved calculation of the "felt" acceleration in each axis to determine a measured pitch/yaw rate that is subtracted to generate the error the controller drives to zero. A filter was imposed such that the low frequency component of the measured pitch rate was subtracted from the controller output (effectively acting as a rate damper). It was recognized that in each axis, regulating an error to zero would exhibit a plethora of zero crossings in a dynamic flight profile. Either an "adaptive zero cross detection" feature or a low-pass filter (similar to the Coulomb Viscous Friction) had to be imposed to reduce the zero crossings and create a better behaving controller.

During initial tests of the controller, the roll channel behaved predictably but the pitch/yaw axes exhibited issues. These issues were not investigated or solved at this time due to a re-scoping of the project to emphasize purely a roll control implementation during the upcoming launch. The potential for control coupling resulting in out-of-control flight still exists and the discussion of safety features based upon air vehicle attitude or rates that would drive surfaces to zero deflection positions was explored. While the mass characteristics and initial aerodynamic coefficients demonstrate a strongly stable missile body, dynamics have not been thoroughly modeled. The issue of single axis control remains an active discussion point. A single axis solution is possible but inadvisable given the current unverified version of the rocket. The risk of dynamic coupling and departure is significant based on the current level of knowledge about the system. However, delaying actuations until a safe altitude is achieved and a rapidly functioning flight termination/parachute system was thought to reduce the negative consequences of a departure (assuming the departure characteristics do not compromise the parachute systems functionality), and reducing the magnitude and duration of control actuations would lower the probability of a coupling induced departure. There exist two issues that needed to be considered; the fidelity of the plant cannot be increased without flight/wind tunnel data but the reaction of the missile to inputs cannot be accurately predicted without a more thoroughly vetted plant model.

Additionally, the Euler kinematic equations in the gravity transform use angles that are one integrator behind the rates. At high roll rates, the projected location of gravity may be significantly different from its actual position. This phase lag in the gravity force acts to create lateral accelerations which are purely an artifact of the model. Thus, rolls in any attitude other than pure vertical result in alpha and beta excursions that the plant interprets as inducing forces upon the air vehicle in the pitch and yaw channels that must be controlled. Reducing the maximum step size of the plant (reducing step/ sampling size) reduces this issue at the expense of greater processing power. It remains to be seen whether this effect is present solely in the mathematical approximation of reality that is the plant or whether these conditions would be experienced during flight. The previous paragraph's discussion on plant fidelity applies. The maximum acceptable step size during simulations might be evaluated by considering the maximum expected roll rate and choosing a maximum step size that would ensure no greater than a maximum acceptable phase mismatch, for example 10 degrees. Recovered data from the flight provided actual inputs for future iterations of GNC. Never running a simulation at larger step size would assure the effect of this simulation phase lag is minimized, but it cannot be completely eliminated.

A more robust mixer was instituted within the control section. It combines inputs into yaw/roll and pitch/roll. A single channel is only permitted to allocate 75% of control authority to a given command to preclude saturation that inhibits blending from the commands coming from other channels. This replaced the original mixer design, which was a simple addition with saturation applied only after mixing.

Following the arrival of the remaining aerodynamic coefficients, the drag polar of the air vehicle was calculated as well as the total lift force. Both values were added to the plant model as functions of dynamic pressure and angle of attack (AoA).

Finally, a Global Model was created that integrated the guidance and navigation model with the control and plant models. The global model transforms the calculated acceleration commands into the body axis system (p, q, r) and inputs those as reference signals to the autopilot. It includes the beginnings of the required mechanization to remove the plant model and tie the system into the actual GPS/INS. Additionally, it includes a Flight Test Data bus to extract time histories of numerous parameters for the post flight analysis that will be critical to optimize future iterations. The Global Model in its current form behaves poorly, due primarily to the interrelated dynamics and limitations discussed in prior paragraphs. Specifically, it is worth reiterating the need to tune filters and gains, validate aerodynamic coefficients, and linearize the plant model. Roll damping was added to the model and the behavior improved, but the aforementioned items remain as work currently outstanding.

The final hurdle involved integration of components. Given the rapidity of a rocket engagement, the GNC flight program ought not to be run as a Simulink model within the MATLAB workspace, but must be compiled into an executable that runs on the implemented flight control computer (FCC). The planned FCC was a PC/104 form factor PC, but the implementation could also include a RasberryPi or other similar devices. A basic executable was created, however, without hardware in the loop it was not possible to verify that the flight control software was functioning correctly or that its execution interprets GPS/INS inputs correctly and commands the desired deflections of control surfaces. Due to this outstanding integration challenge, the Global Model stands as a foundational exercise that is currently useful in providing low fidelity predictions of missile behavior, but was not incorporated into the rocket for this round of flight test. The Global Model is shown in Figure 60.

a. Recommendations for Future Work

The following GNC Team recommendations, if successfully followed, should provide for actual guided flight of the NPS missile on a future flight test event:

1. Hardware / Software Integration – Processing of the Simulink based Flight Program for use on the final standard flight control computer is a critical task that should top the priorities during follow-on development of this

system. Integration of the fin Digital-to-Analog (DAC) control mechanism and the GPS/INS within the Simulink environment should be the first step. Once data flow and behavior is verified, then the Real Time Workshop or an embedded coder (if target hardware for FCC were to change), should be used to develop the flight release of the software. Once loaded onto the flight-standard FCC simple tests can verify functionality (i.e., rolling the sensor and looking for correct fin deflection). Documentation of this process MUST be robust, but once a repeatable and reliable process is established for moving from the Simulink model to the flight-standard FCC, a rapid FLY-FIX-FLY developmental loop will be established allowing changes to flight control software literally from launch to launch [21].

2. Tuning – Use the specs of the sensors and the flight test data gathered during this iteration of testing (which should enable construction of the missile transfer function [bode diagrams]). Use this information to tune the filters and gains in the GNC program. The goal is for the filters to be fast enough to estimate the LOS rate but band limited to not pass noise through to the control signals. Another means of investigating/ accomplishing the tuning would be via the generation of a simpler plant (3DOF) such that you could verify the autopilot on an easier to interpret model and then apply the lessons and tuning to the more complex nonlinear model. Anytime the plant is used for tuning the rocket should be placed in un-accelerated flight (thrust and gravity turned off, flying in free space). Highly non-linear mechanisms (such as drag due to fin deflection) may need to be disabled as well. Another option is to generate a linear model of the plant using the MATLAB control systems toolbox. Gains might be tuned using the gross characteristics of the missile as described in the references [22].

3. Plant – A thorough peer review of the plant model and its underlying assumptions should be conducted. Aerodynamic coefficients should be updated in the model to include the real data collected from all launches. Execute linear analysis (SCD Toolbox) of plant model (as discussed above), then use the higher fidelity plant to more accurately predict flight behavior with more complex control inputs. The full non-linear plant should be used for simulations, although tuning must be completed using a linear model. Testing recommendations for further flight test progression is shown in Table 10.

Table 10. Possible Flight Test Progression for Plant / Autopilot Tuning

Flight	Test	Goal
1-2	Roll command	Observe plant response
3	Single Fin step input (small)	Parameter identification of Aero Coeeff
4	Paired Fin step input (small)	Parameter identification of Aero Coeeff

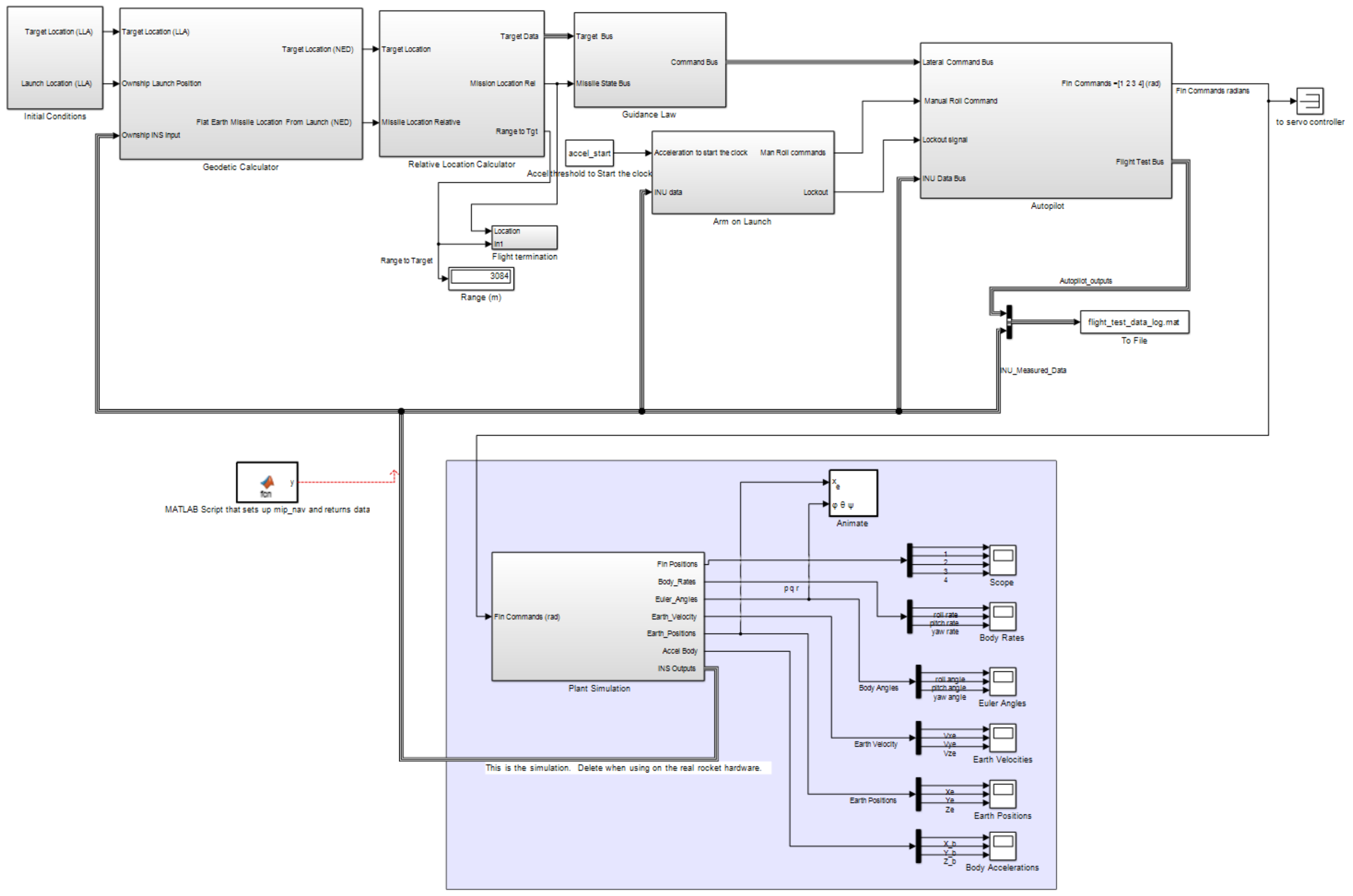


Figure 60. Global Model (including Flight Program and Plant Simulation Blocks)

2. Standards and Conventions

Below are several figures that document the conventions used throughout the GNC SIMULINK model. These are important references for understanding the signs on many of the blocks within the global model and sub-blocks. The coordinate frames are shown in Table 11. After the table are multiple figures that show the coordinate frames, conventions, and orientations used during the simulations. The fin number system is shown in Figure 61. The launch orientation for the simulation is shown in Figure 62. The positive fin deflection is shown in Figure 63. The body axes positive directions are shown in Figure 64.

Table 11. Coordinate Frames

Name	Coordinates	Units	Notes
Geodetic {λ,ϕ,h}	Lat. Long. Height	Decimal Degrees / meters above datum ellipsoid	Must specify if using MSL as datum Origin: equator/PM
Local Tangent Frame{u}	North, East, Down Ψ,Θ,Φ	Meters Radians	Origin: launch point
Navigation Frame{n}	North, East, Down	Meters	Origin: CG
Body Frame {b}	X, Y, Z ϕ,θ,ψ (roll pitch yaw angle) p, q, r (roll pitch yaw rate) p_dot, q_dot, r_dot (roll, pitch, yaw accelerations) L,M,N (roll pitch yaw moments)	Meters Rad Rad/s Rad/s/s Nm	Origin:CG

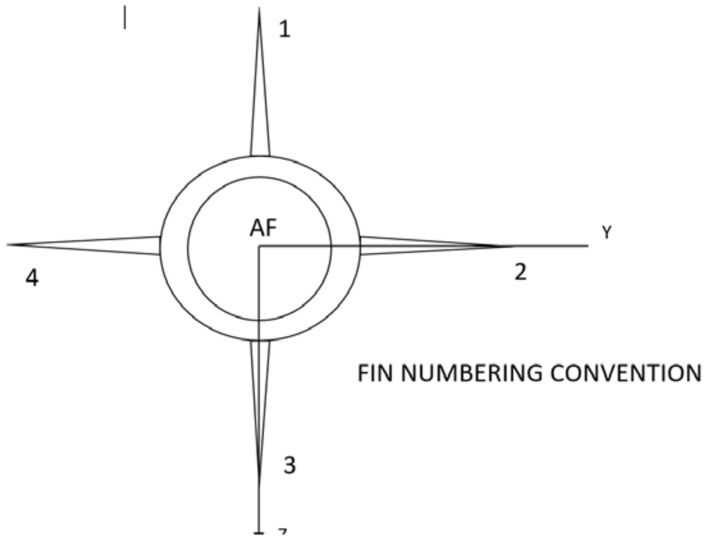


Figure 61. Fin Numbering Convention

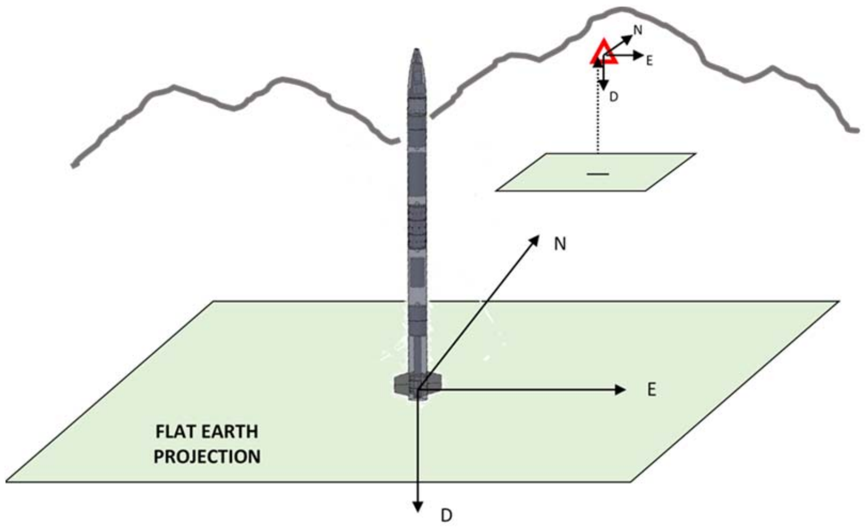


Figure 62. Launching Orientation

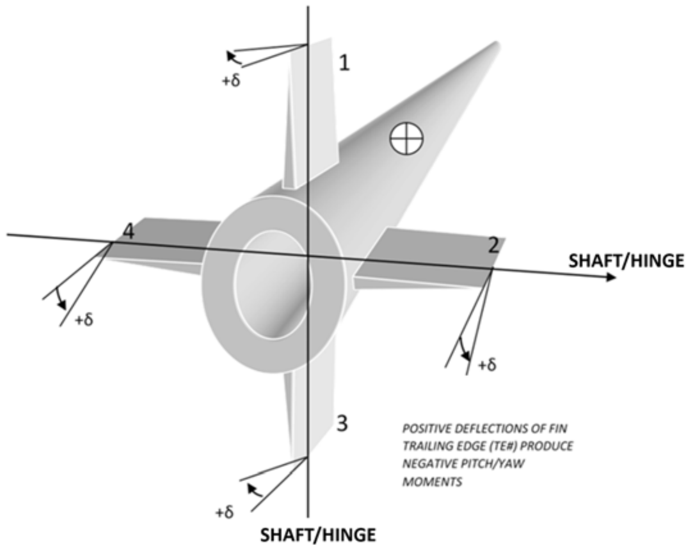

1
+δ
4
2
SHAFT/HINGE
+δ
+δ
3
POSITIVE DEFLECTIONS OF FIN TRAILING EDGE (TE#) PRODUCE NEGATIVE PITCH/YAW MOMENTS
+δ
SHAFT/HINGE

Figure 63. Positive Fin Deflection Convention

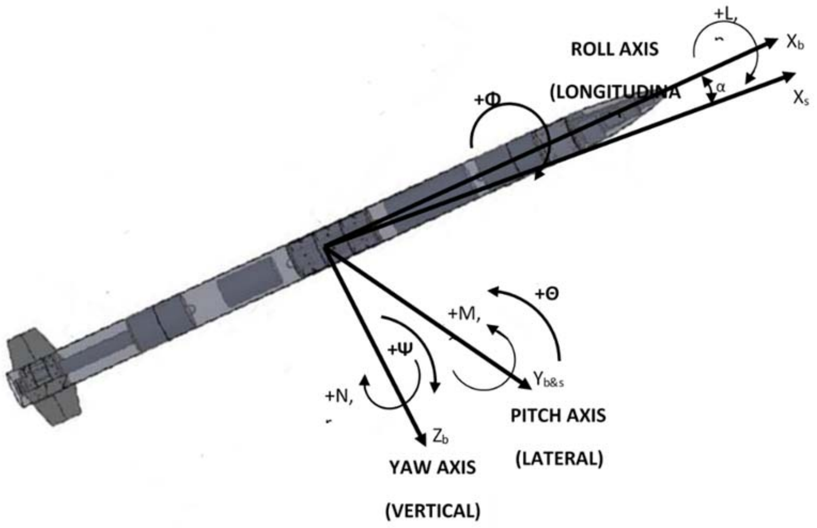


Figure 64. Body Axes Positive Directions

3. Global Variable Setup Script

To facilitate a clear understanding of the assumed values of quantities within the model, most constants within the model are implemented as named variables. Before running the model the variables are set by running the "Global_model_setup.m" script. The script is shown in Figure 65. Changing any of the variables will thus change them throughout the model when they appear in various places and prevent errors due to changing one but not all values of a given quantity. Of note, the current value of all constants is displayed in a condensed format in the MATLAB workspace for convenient reference.

```
%% Missile OFP Setup Script% Originally written by Major John Dirk, LCDR Travis Hartman for NPS
ME4704  Prof. Dr. Chris Brophy % Significant help provided by Prof. Dr. Vlad Dobrokhodov
Target_Location=[38.2 -76.408 2397]; %Lat Long Altitude of target (altitude in meters)
Launch_Position=[38.1831 -76.397703 0]; %LLA of Launch Postion
g=9.80665; %(m/s/s) gravity at earths surface
accel_start=30; %(m/s/s) acceleration threshold to start the clock
fin_unlock_time=5; %time after clock start to unlock fins
roll_control_active_time=7; %time after clock start to enable closed loop roll control
S=.035351; %Reference surface area
c_bar=.208661; %average cord length
S_c_bar=S*c_bar; % m^3 S*c_bar (2 fins) assumed to dimensionalize C_M_alpha
C_M_alpha=-74.485; %(1/rad) change in pitching moment coefficient with alpha (radians) (-1.3)
C_M_alpha_alpha=-257.839; %(1/rad^2) variation of Cma with alpha  (-.05 in alpha degrees)
C_N_beta=C_M_alpha; %axisemetric missile
C_N_beta_beta=C_M_alpha_alpha; %axisemetric missile
C_M_q=-1604.283; %(1/rad/s) change in pitching moment coeff due to pitch rate (Pitch damping) (-28)
C_M_alpha_dot=-101.987;  %(1/rad/sec) change in pitching movement coeff due to alpha rate (-1.78)
C_N_r=C_M_q; %axisemetric missile
C_N_beta_dot=C_M_alpha_dot; %axisemetric missile
C_L_p=-18.9; %(1/rad/sec) change in roll moment coeeff due to roll rate (roll damping) (-.33 in deg)
C_F_delta_fin=13.5218; %coefficient of normal force with fin deflection (radians) (=.236 in degrees)
C_l_alpha=12.8915; %(1/rad) change of coefficient of lift with alpha (.225 in 1/deg);
C_l_beta=C_l_alpha; %axisemetric missile
C_d_zero=.375; %.002
Delta_C_d_zero=.00375; %additional drag per full deflected fin
C_d_alpha_square=6.839; %Change in ceofficient of drag due to alpha squared (alpha in radians) (.0021)
C_d_beta_square=C_d_alpha_square; %axisemetric missile
Mass_full=33.2; %(Kg) Full Mass
Mass_empty=28.245; %(Kg) empty mass
I_xx=.2026;
I_yy=22.94;
I_zz=I_yy; %axisemetric missile
I_empty=[I_xx 0 0; 0 I_yy 0; 0 0 I_zz]; %(kg/m^2) empty weight inertial tensor
I_full=(Mass_full/Mass_empty)*I_empty; %Estimate of Full Weight inertial tensor
Euler_initial=[0 pi/4 0]; %Initial launch angle
Roll_moment_arm=.176987; %(m)
Pitch_moment_arm=1.857171; %(m)
Yaw_moment_arm=Pitch_moment_arm; %axisemetric
specific_force=9503.82; %(N/rad) force per radian of deflection due to one fin
TSMF=0.000641632; %(Kg/s/N) Thrust Specific Mass Flow
Thrust_max=2826.9; %(N) Maximum Thrust of the motor (percent
Fin_max=.052; %(rad) max fin deflection
actuator_freq=31; %(rad/sec) natural frequency of actuators
actuator_damp=1.3; % damping ratio of actuators
density_exp=1+(g*.0289644/(8.31432*-.0065)); %exponant of atmospheric density equation
T_0=288.15; %K temp at sea level (ISA atmosphere)
L_b=-.0065; %K/m standard temp lapse rate
roh_0=1.225; %kg/m^3 standard atmosphere density at sea level
h=11000;  %(m) Height above MSL
roh=roh_0*(T_0/(T_0+L_b*h))^density_exp; %(kg/m^3)density at height h
```

Figure 65. Global_model_setup.m

GNC Block Descriptions

- **Block Name**: Initial Conditions
- **Location**: Top Level
- **Purpose**: This block sets the position of the target and launch position of the missile in Latitude Longitude and Altitude in meters (LLA).
- **Inputs**: Global variables from setup script.
- **Outputs**: Launch position and target current location in LLA.
- Limitations/Assumptions/Simplifications: Assumes target is static.
- **Problems**: None identified.
- **Future Improvements:** This block was left as a sub-function to allow for a more complex target scenario (i.e., a moving target).
- Additional Notes / References / Resources: None

- **Block Name**: Geodetic Calculator
- **Location**: Top Level
- **Purpose**: This block takes the target location, launch location, and INS inputs (all in LLA) and calculates a missile and target relative position from launch in a local tangent plane (Flat Earth) location.
- **Inputs**: Launch position and target current location in LLA and Missile current position in LLA from the INS.
- **Outputs**: Target and missile location (relative to launch) in local tangent plane (North, East, Down).
- **Limitations/Assumptions/Simplifications**: Assumes earth is flat at the Launch Position (3) altitude.
- **Problems**: None identified.
- **Future Improvements**: None identified.
- **Additional Notes / References / Resources:** Conversion from Geodetic (LLA) frame to local tangent frame is accomplished by aerospace blockset converters. More information is available in the Simulink help.

- **Block Name**: Relative Location Calculator
- **Location**: Top Level
- **Purpose**: This block takes the target location and missile location (relative to launch point in NED coordinates of meters) calculates range to target and passes through the input data.
- **Inputs**: Target and missile location (relative to launch) in local tangent plane (North, East, Down).
- **Outputs**: Target Data Bus (consisting of target location (NED) and range to target from missile) and missile location relative launch. Range to target from missile is output separately for simulation termination function.
- Limitations/Assumptions/Simplifications: None.
- **Problems**: None identified.
- **Future Improvements:** This block was significantly simplified as functions from it moved to other blocks. It could be incorporated into other blocks.
- **Additional Notes / References / Resources:**

- **Block Name**: Guidance Law Figure 66.
- **Location**: Top Level
- **Purpose**: This block uses missile and target location in the local tangent plane (NED in meters) to calculate X Y Z acceleration commands in the missile body frame.

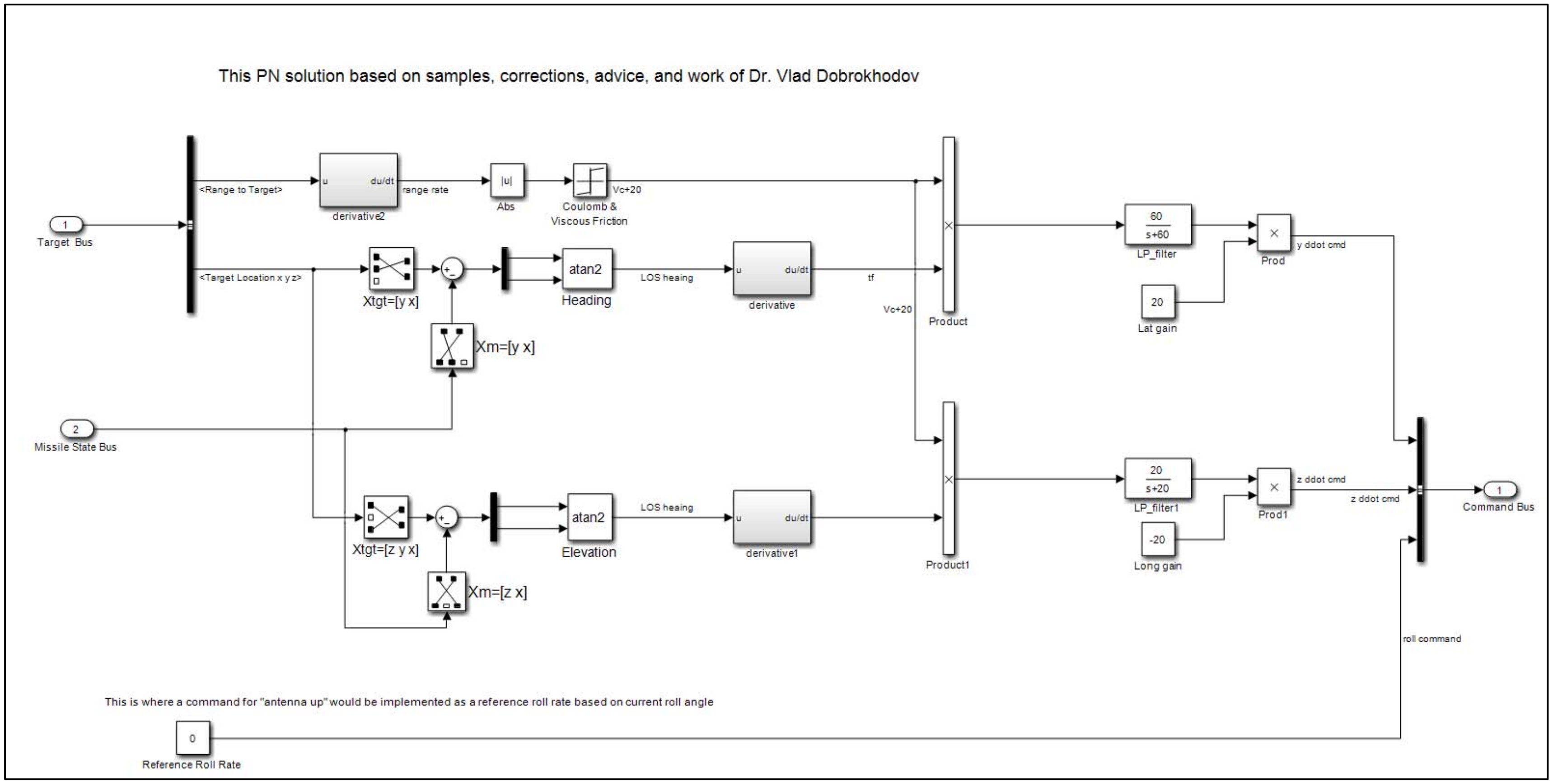


Figure 66. Guidance Law Block Showing an Implementation of Proportional Navigation (PN)

- **Inputs**: Target Data Bus (consisting of target location (NED) and range to target from missile) and missile location relative launch.

- **Outputs**: Command bus (2D vector consisting of $\dot{y}$ and $\dot{x}$ commands in the body frame)

- **Limitations/Assumptions/Simplifications:**

  - **Description:** This block implements a Proportional Navigation guidance solution based on the equation

$$\begin{bmatrix}\dot{y}\\ \dot{z}\end{bmatrix} = \boldsymbol{P}\dot{R}\dot{\boldsymbol{\omega}}$$

where $\begin{bmatrix}\dot{y}\\ \dot{z}\end{bmatrix}$ represents the commanded lateral accelerations, $\boldsymbol{P}$ is the navigational constant of proportionality for each lateral axis, $\dot{R}$ is range rate (or closing velocity), and $\dot{\boldsymbol{\omega}}$ is the line of sight rate vector in elevation and azimuth.

  - **Problems:** The low pass filters and command gains are critical to proper functioning of this block. They will need to be updated as the rocket design progresses. Further, analysis of this block are incomplete. Analysis must include evaluation of target locations on the coordinate axes. Odd behaviors have been noted at locations such as [1000 0 1000] etc. This implementation requires a “fin lock” logic for at least the first few tenths of seconds to avoid bad behavior due to math. The fin lock mechanism was moved from this block to the autopilot, but it must exist.

  - **Future Improvements:** Tune low pass filters, evaluate that the calculated command acceleration rates for desired effect. Roll rate command is included in this block but set to zero. Future improvement could close the loop on roll angle to provide a “GPS ANTENNA SKYWARD” command throughout flight.

  - **Additional Notes:** Originally implemented at pursuit guidance (PG), it was determined in testing that PN was more robust and a better long term (scalable) solution for the system if a single guidance law is used especially for a vertical (pi/2) launch condition. Dr. Vlad Dobrokhodov heavily modified early versions to find a workable solution. Other solutions tried included systems that identified the target location quadrant and then used a logic switcher mechanism to change the law slightly based on the quadrant. The implemented solution here is a more robust calculation which is less likely to cause location “windup” and crossing of 2π. Note that the “atan2” Simulink blocks used within this block are “4 quadrant arctangents and carry with them data from the calculated quadrant. The Coulomb and Viscous Friction block adds a flat 20 m/s to the calculated range rate to help prevent zero crossings which slow down the calculation. See also the Navigation Evaluation Model which can be used in conjunction with this block to evaluate its performance.

- **Block Name**: Arm on Launch, Figure 67.
- **Location**: Top Level
- **Purpose**: This block uses sensed acceleration to start the flight sequence, including initiation of a pre-planned roll maneuver and timed FIN command unlock.
- **Inputs**: INS acceleration ($\dot{x}$), Acceleration threshold to start the clock (external from *accel_start*).
- **Outputs**: Lockout Boolean commands (1 is fins enabled) and manual roll commands. Roll commands are output from -1 to 1 as a fraction of maximum fin deflection defined by *Fin_max* global variable.
- **Limitations/Assumptions/Simplifications**: Due to limitation to scope during this cycle, the arming block can only send a roll channel manual command. Also, FIN LOCK is enabled along with a ROLL CHANNEL lock for the same reason. Although YAW and PITCH channels cannot be individually locked, they are de facto locked if the FIN channel is set to '0'.
- **Problems**: None identified.
- **Future Improvements:** Potential area of growth to add option for manual pitch and yaw channel commands for aerodynamic parameter identification during future test flights to further define the air vehicle behavior. Other parameters could be tied to this "clock starting event." Add one or two more lockout channels to allow individual lockout control of YAW/PITCH in addition to ROLL command channels.
- **Description:** The current body X accel from the accelerometer is compared to the value set by the global variable *accel_start*. When the sensed acceleration exceeds the preset value the block "acceleration trigger" is enabled. Within this block is a step function which goes from 0 to 1 at .01 seconds after being enabled and remains in that state. It acts as a memory to allow the clock to continue running even when X accel falls below the threshold. This trigger block enables the manual roll command block containing a custom signal that sends a normalized roll channel command based on time. The trigger block also enables the roll channel
- **Additional Notes / References / Resources**:

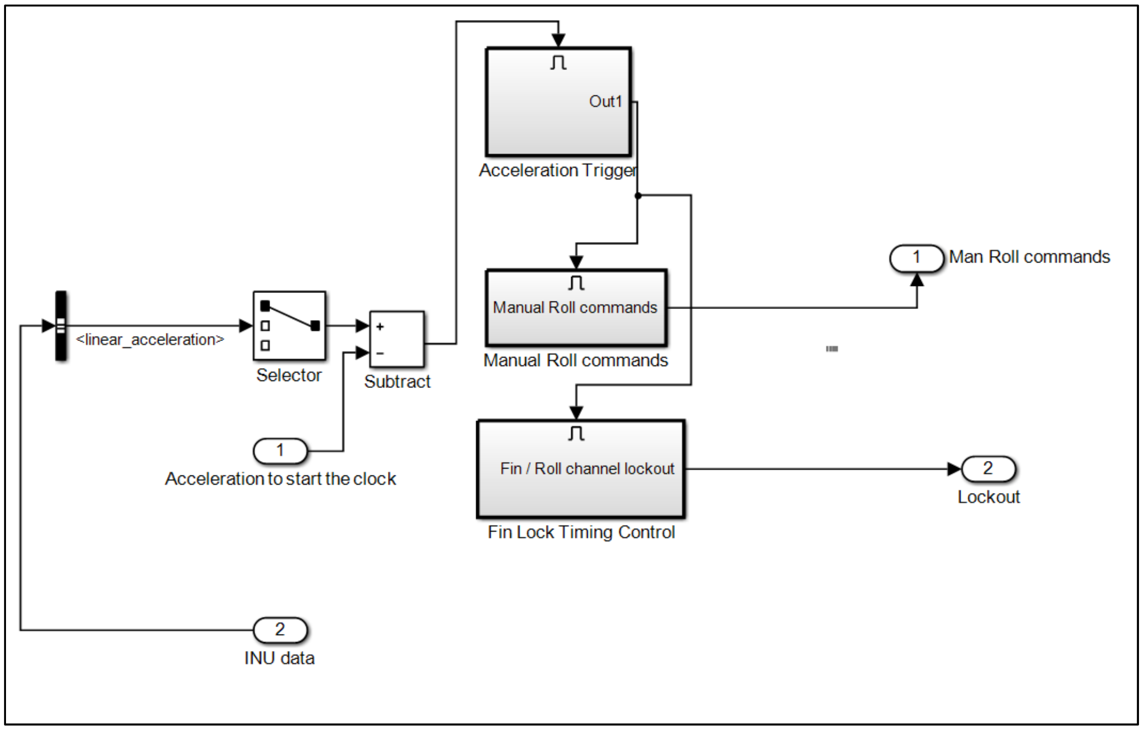


Figure 67. Arm on Launch Block

- **Block Name**: Flight Termination, Figure 68.
- **Location**: Top Level
- **Purpose**: This block stops the simulation when the missile closes within 20m of the target or the missile flies below the launch altitude.
- **Inputs**: Range to Target, Target Location
- **Outputs**: none (Simulink STOP command)
- Limitations/Assumptions/Simplifications:
- **Problems**: None identified.
- **Future Improvements:** This block could be used to take a certain action in flight, however, it would normally be removed during implementation of the OFP in the flight control computer and is used for simulation only.
- **Description:** If the range is less than 20m or the relative altitude to launch is underground (i.e., positive as positive is DOWN), then stop the simulation.
- **Additional Notes / References / Resources:**

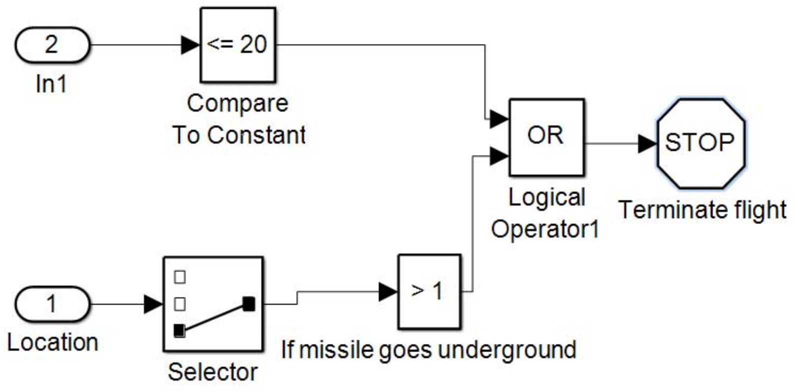


Figure 68. Flight Termination

- **Block Name**: Autopilot, Figure 69.
- **Location**: Top Level
- **Purpose**: This block implements a stabilizing autopilot control system in Roll, Pitch, and Yaw outputting FIN commands to the missile fin controller (or simulated plant for simulation) and flight test data to the FLIGHT TEST BUS for record.
- **Inputs**: Lateral Commands from COMMAND BUS, Manual Roll commands, Lockout signals, INU data.
- **Outputs**: Fin commands (in radians), flight test data on FLIGHT TEST BUS for recording.
- **Limitations/Assumptions/Simplifications**: Due to limits to scope for this project the Pitch and Yaw channels are sketched but disconnected. The roll channel is implemented with an Aerospace Blockset PID controller.
- **Problems**: Pitch and Yaw channels are not completed and are disconnected.
- **Future Improvements:** Pitch and Yaw channels must be connected and tuned using an analytical method. FLIGHT TEST BUS data was selected for the ROLL ONLY version of this model. Pitch and Yaw parameters need to be added. Pitch and Yaw closed loop control lockout are not provided, however control lockout for closed loop Roll control is provided. Adding provision for closed loop Yaw and Pitch lockout is a possible upgrade. Adding a saturation limit in the roll command channel less than 100% would allow every channel to at least contribute some to the final fin position. A more complex mixing scheme prioritizing the channels based on flight condition may also be desirable.
- **Description:** Input rate command signals are compared with INU measured rates and the error is controlled to zero. Pitch and Yaw channels are limited to 75% authority through saturation blocks, roll is permitted up to 100% authority (this should probably also be limited). Roll, Pitch, Yaw command signals are then mixed. Saturation blocks ensure that a command for any fin greater than +/- *Fin_max* is not sent. Data from throughout the autopilot is added to the FLIGHT TEST BUS and outputted for recording.
- **Additional Notes / References / Resources:** Guidance for completing and tuning the lateral autopilot channels is available [22].

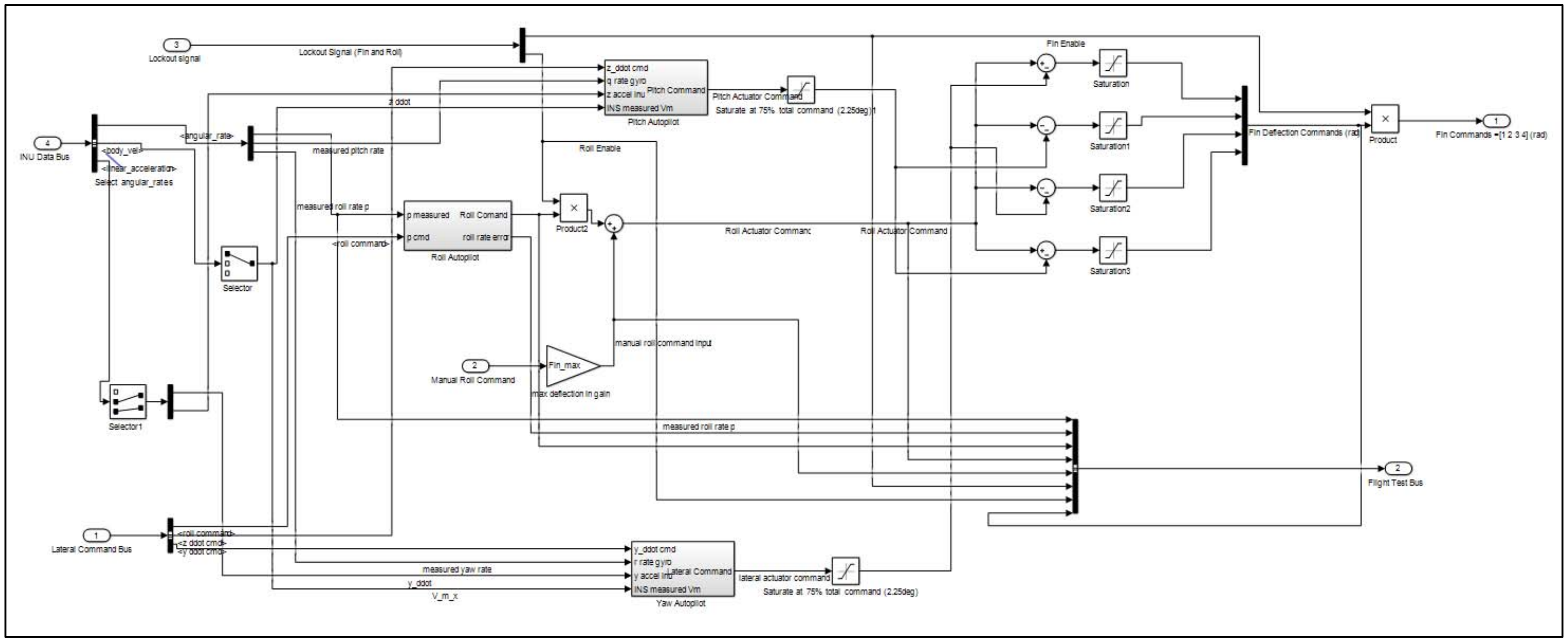


Figure 69. Autopilot

- **Block Name**: Plant Simulation, Figure 70.
- **Location**: Top Level
- **Purpose**: This block seeks to model the complete behavior of the plant to allow tuning of the Operational Flight Program (OFP) and to generate predictions of flight characteristics given initial conditions.
- **Inputs**: Fin commands (4d vector) in radians.
- **Outputs**: Simulated INS outputs. Additionally, many missile states are output to graphs for error checking and de-bugging.
- **Description:** The plant model is based around a 6 Degree of Freedom (6DOF), Euler Angle based equations of motion block from the Aerospace Blockset. The 6DOF block takes forces and moments in the body frame and converts it into motions. Models are provided for Gravity, Propulsion, Missile Aerodynamic Stability, Aerodynamic Drag, Aerodynamic Lift, INS output, and Actuator Dynamics.
- **Limitations/Assumptions/Simplifications**: There are numerous limitations to the plant model, see individual sub-block descriptions for limitations.
- **Problems**: Pitch and Yaw channels are not completed and are disconnected.
- **Future Improvements:** Additional fidelity can be added to the model by model matching with flight test, as well as adding additional Aerodynamic effects, not all of which are currently modeled.
- **Additional Notes / References / Resources:** It is recommended once the model is completed and verified that it be linearized using the MATLAB toolset, and then that linear model be used to analytically tune the autopilot. Once the tuning is complete the full plant simulation can be used to predict and then compare the missile flights for each test. When the OFP is implemented for use in the Flight Control Computer the model must be deleted out of the Simulink File prior to auto-coding.

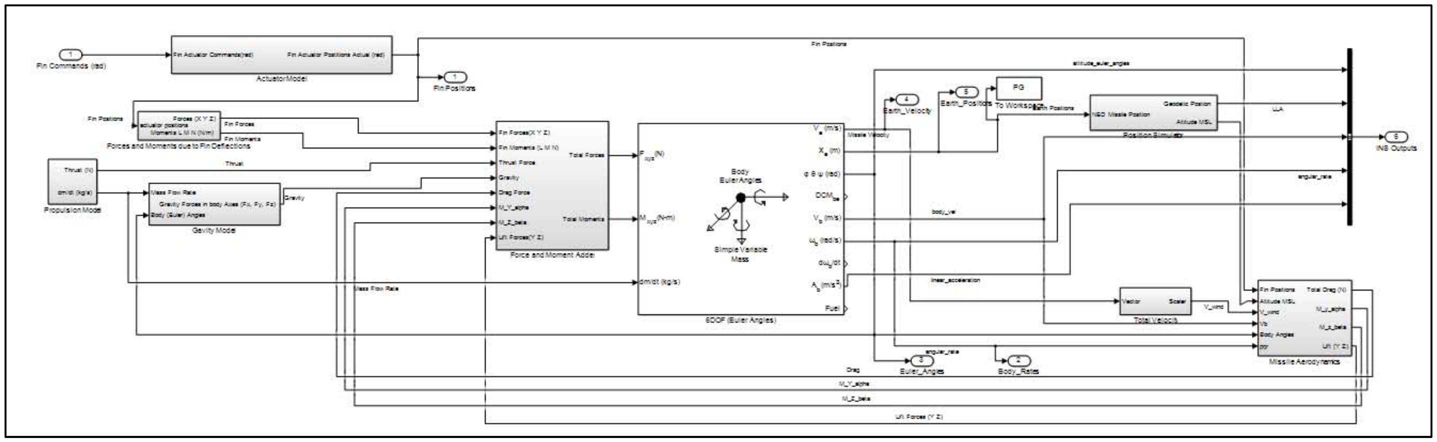


Figure 70. Plant Simulation

•

- **Block Name**: Missile Aerodynamics, Figure 71.
- **Location**: Sub-block of Plant Simulation
- **Purpose**: This block seeks to model aerodynamic characteristics of the missile by taking aerodynamic coefficients and the current missile state at each moment in time and calculates aerodynamic forces and moments in the body axis system. Wind Angles are calculated from body velocities, dynamic pressure is calculated from altitude and total velocity. From these Lift and Drag and Missile stability moments are calculated.
- **Inputs**: Fin Positions, MSL Altitudes, Total Velocity, Body Velocities, Body Angles, Body rates
- **Outputs**: Forces due to life (2D vector of Y and Z forces), Total Drag, Pitch and Yaw moments.
- **Description:** Within the missile aerodynamics block the dynamic pressure and wind angles are calculated and are used to determine the lift and drag force upon the air vehicle as a function of the aforementioned variables and fin position (see the appropriate block description for Lift and Drag). The dynamic pressure and wind angles are used with body angles and rates to calculate moments (see the appropriate block description for Stability).
- **Limitations/Assumptions/Simplifications**: The aerodynamic coefficients were provided for the GNC solution via computer modeling that used the Missile Lab program. The provided graphs plotted aerodynamic coefficients versus angle of attack (in degrees) for specific series of fin deflections. Small angles were assumed, resulting in the normal force being considered equivalent to the lift force.
- **Problems**: None identified.
- **Future Improvements:** Data from previous launches should be employed to fine tune the aerodynamic modeling and increase the fidelity of the coefficients used in the GNC model.

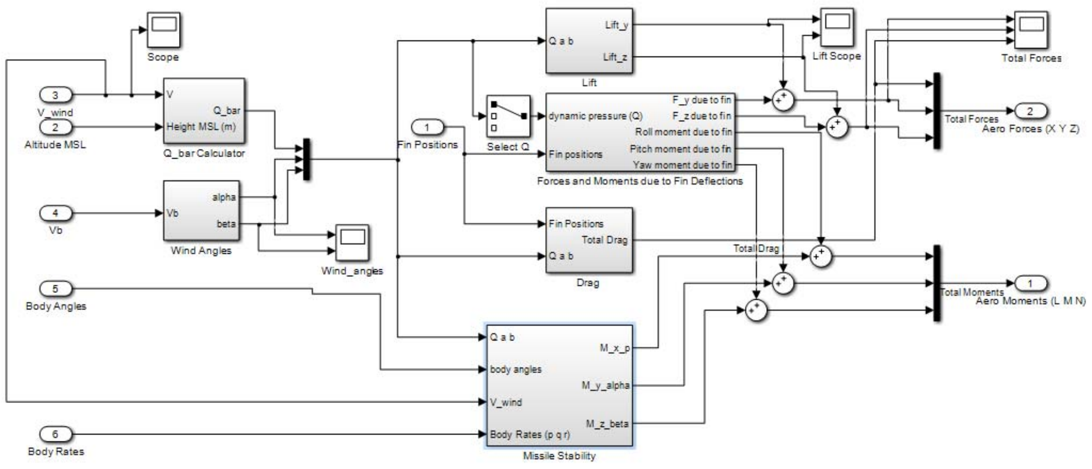


Figure 71. Missile Aerodynamics

- **Block Name**: Lift, Figure 72.
- **Location**: Sub-block of Missile Aerodynamics
- **Purpose**: This block implements a calculation of forces due to alpha and beta referred to as Lift_y and Lift_z in the diagram.
- **Inputs**: 3-d vector consisting of Dynamic Pressure (Q), alpha and beta.
- **Outputs**: Lift_z and Lift_y.
- **Limitations**/Assumptions/Simplifications: Small angles were assumed, resulting in the normal force being considered equivalent to the lift force.
- **Description**: The equations below represent the calculations.

$$F_Z = L_\alpha = \bar{Q}SC_{\ell_\alpha}$$

$$F_Y = L_\beta = \bar{Q}SC_{\ell_\beta}$$

The provided coefficient of lift due to alpha or beta is multiplied by the dynamic pressure and reference area (S) to provide the lift force in each axis. Given that the missile is axisymmetric, the no dimensional aerodynamic coefficients for each are identical. However, both are programmed via the MATLAB script (see Section III) and can be adapted if the air vehicle is changed in the future.

- **Problems**: None Identified.
- **Future Improvements:**
- **Additional Notes / References / Resources:**

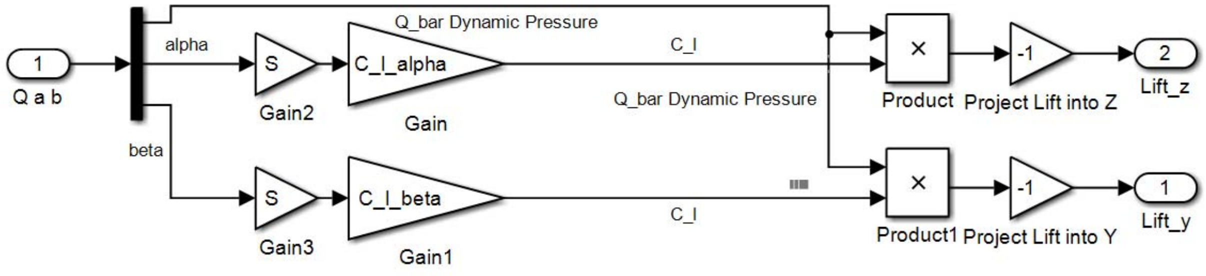


Figure 72. Lift

- **Block Name**: Rocket Stability, Figure 73.
- **Location**: Sub-block of Missile Aerodynamics
- **Purpose**: This block implements calculations of lateral stability.
- **Inputs**: 3D vector consisting of Dynamic Pressure (Q), alpha and beta. Body Angles (3 vector), pitch rate (q) and Yaw rate (r), Total velocity.
- **Outputs**: M_y_alpha, M_z_beta
- Limitations/Assumptions/Simplifications:
- **Description:** This block organizes inputs to two identical blocks, one for each of the lateral axes and outputs the moments they calculate. An arbitrarily small positive value (.1) is added to total wind velocity (V_wind) to prevent a divide by zero error at a velocity of zero.
- **Problems**: None Identified.
- **Future Improvements:**
- **Additional Notes / References / Resources**:

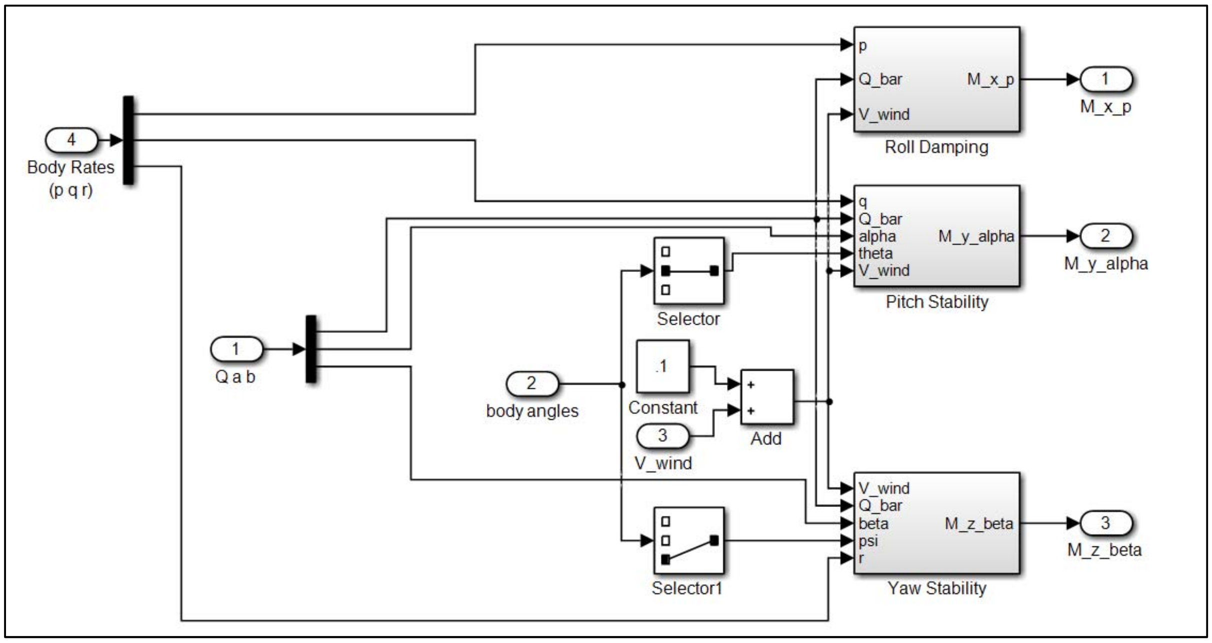


Figure 73. Rocket Stability

- **Block Name**: Pitch Stability and Yaw Stability, Figure 74.
- **Location**: Sub-block of Missile Stability
- **Purpose**: This block implements calculations of pitch stability.
- **Inputs**: Pitch rate (q), dynamic pressure (Q_bar), angle of attack (alpha), total velocity (V_wind). Theta is provided for troubleshooting and comparison.
- **Outputs**: M_y_alpha
- **Limitations/Assumptions/Simplifications**: Only the three mechanisms of pitch stability are included, however these represent the primary contributors.
- **Description:** This block implements total pitching moment through pitching moment due to angle of attack (first order pitch stability), pitching moment due to pitch rate (first order pitch damping), pitching moment due to angle of attack rate (second order pitch damping). The equation representing the implemented calculation of pitch stability is:
- $M_Y = \bar{Q}S\bar{c}\left(C_{m_\alpha} + C_{m_q}\frac{q\bar{c}}{2V} + C_{m_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2V}\right)$
- **Problems**: None Identified.
- **Future Improvements:** Additional mechanisms of pitch stability could be added. However, the largest contributors to static and dynamic stability are already included.
- **Additional Notes / References / Resources:** The Yaw Stability block implements the same calculation for the other lateral axis.

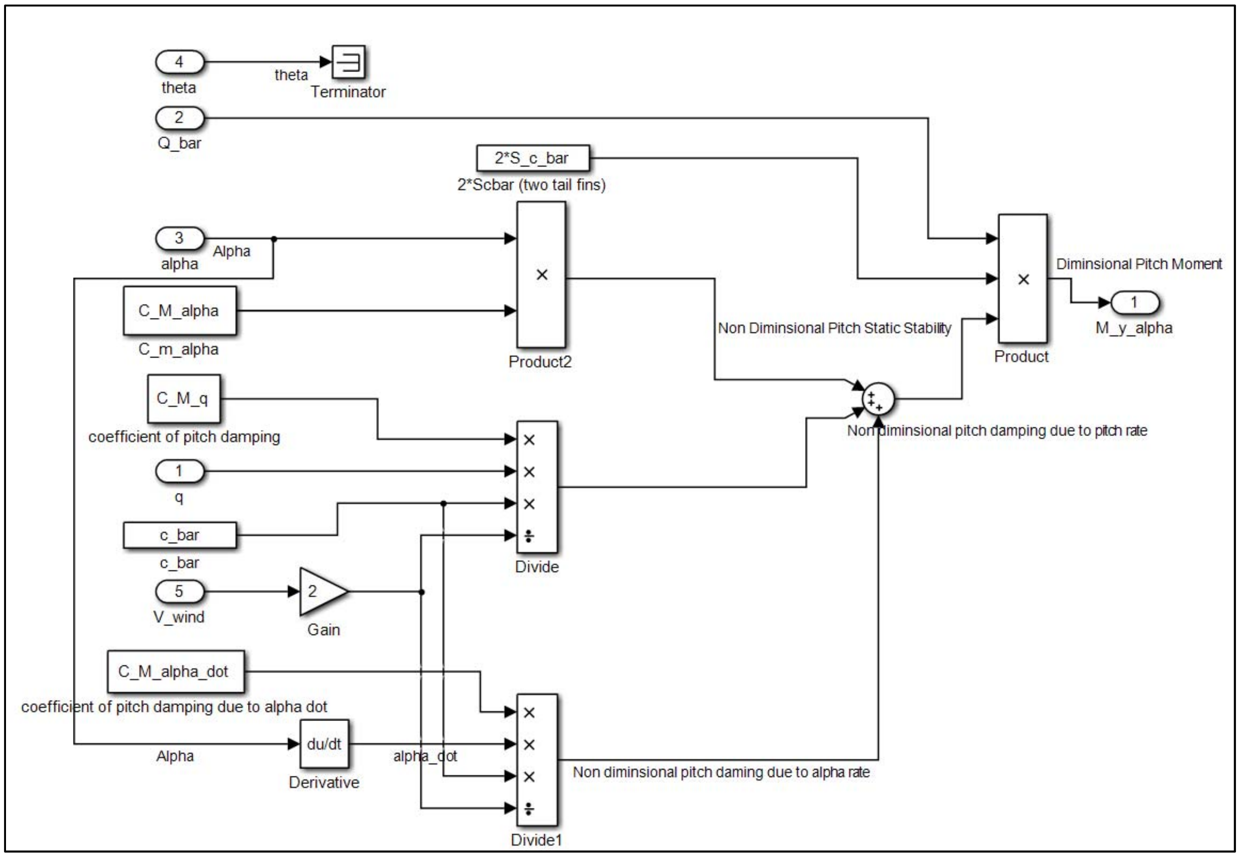


Figure 74. Pitch Stability

- **Block Name**: Roll Damping, Figure 75.
- **Location**: Sub-block of Missile Stability, which is sub-block of Missile Aerodynamics
- **Purpose**: Calculate the roll moment due to roll rate (roll damping)
- **Inputs**: Dynamic pressure, roll rate, total velocity
- **Outputs**: Roll moment
- **Limitations/Assumptions/Simplifications**: Assumes coefficient of roll damping is a total body and fin combination value.
- **Description:** Implements the below equation

$$M_X = \bar{Q}S\bar{c}\left(C_{l_p}\frac{p\bar{c}}{2V}\right)$$

- **Problems**: None noted.
- Future Improvements: None noted.
- **Additional Notes / References / Resources:** Use flight test data to validate and tune the assumed aero values to be more accurate.

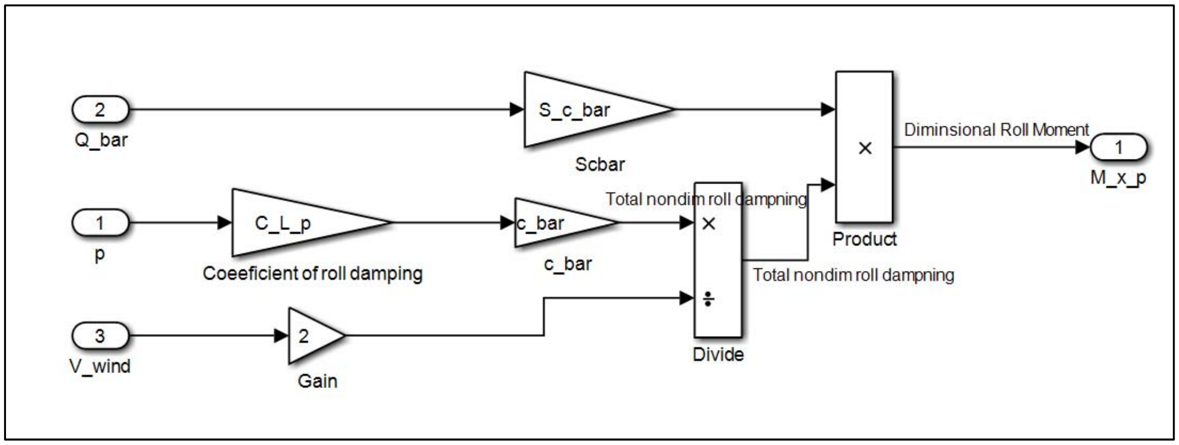


Figure 75. Roll Damping Block

- **Block Name:** Drag, Figure 76.
- **Location:** Sub-block of Missile Aerodynamics
- **Purpose:** This block implements calculations of total drag.
- **Inputs:** Dynamic pressure (Q_bar), angle of attack (alpha), angle of sideslip (beta), and fin positions (4 vector).
- **Outputs:** Total drag force (Drag)
- **Limitations/Assumptions/Simplifications:** This block assumes that the additional drag added by fin deflection is linear with fin deflection and increases C_{d_0} with a negligible effect on C_{d_α}. The drag polar was calculated from Missile Lab aero data (). The constants used can be adjusted in the setup script. No wave drag due to local transonic flow is considered.

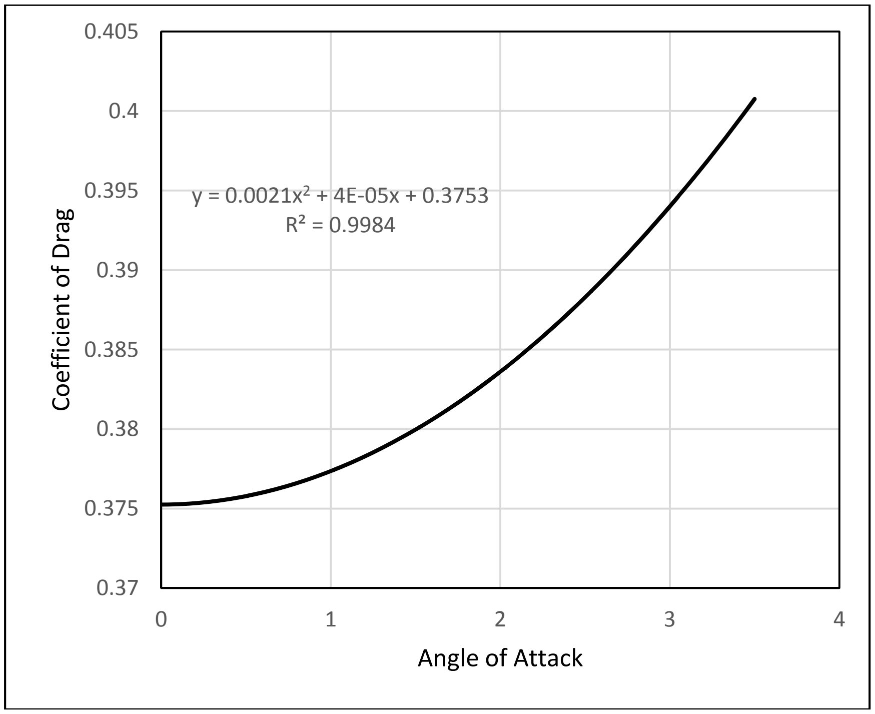


Figure 76. Missile Drag Polar (Fins at Zero Deflection) Coefficient of Drag versus Angle of Attack

- **Description:** This block calculates total coefficient of drag accounting for alpha, beta, and fin deflection and then dimensionalizes it using dynamic pressure and reference area to calculate total drag force according to the following equation:

$$F_X = -D = \bar{Q}S\left(C_{d_0} + C_{d_\alpha}\alpha^2 + C_{d_\beta}\beta^2 + C_{d_{\delta fin}}\delta fin\right)$$

The absolute value of all fin deflections are summed and taken as a ratio with the maximum fin deflection of one fin. This value is then multiplied times the numerically modeled value of the change in C_{d_0} due to one fin deflection.

- **Problems:** None Identified.
- Future Improvements:
- **Additional Notes / References / Resources:**

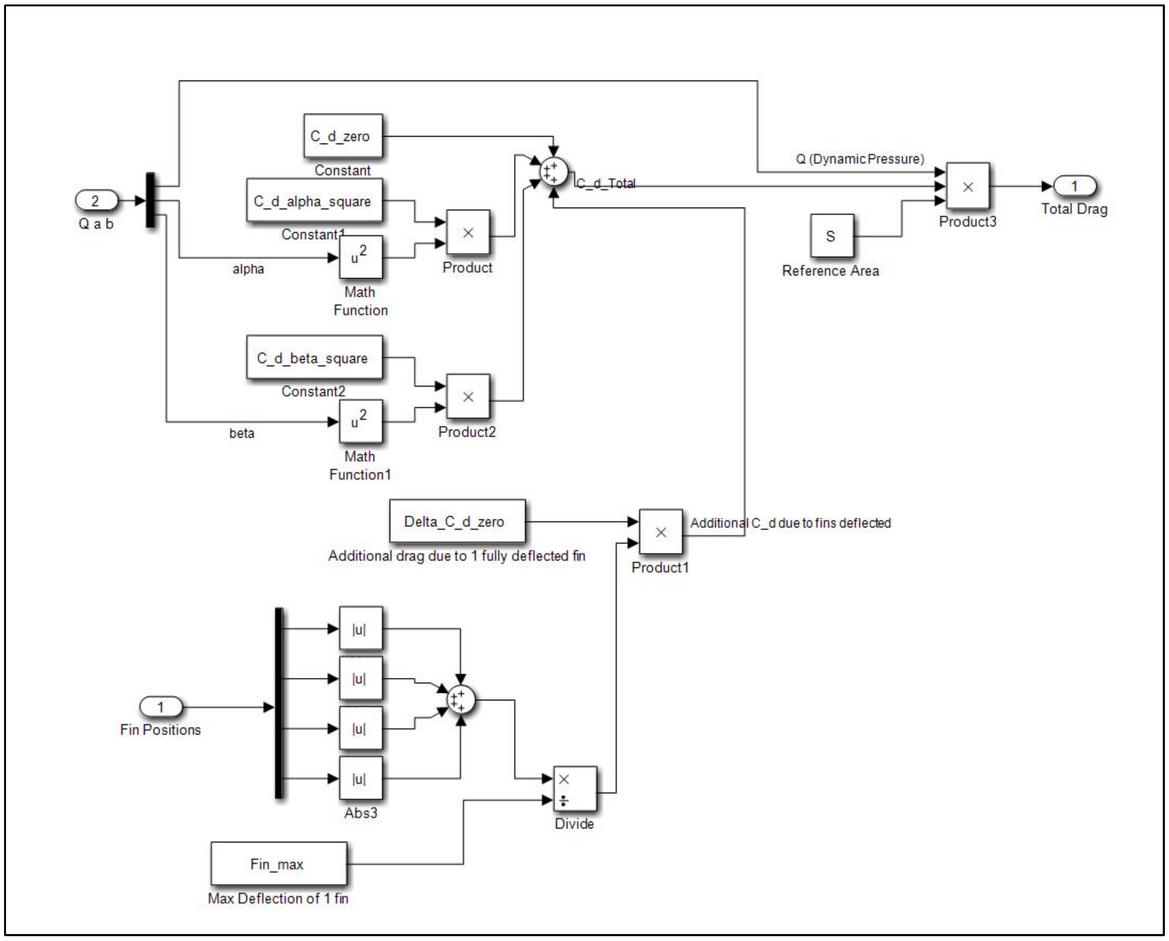


Figure 77. Drag Block

- **Block Name**: Forces and Moments due to Fin Deflections, Figure 78.
- **Location**: Sub-block of Missile Aerodynamics, which is sub-block of Plant Simulation
- **Purpose**: The purpose of this block is to calculate the forces and moments due to deflection of the fins from the fared position.
- **Inputs**: Fin positions, Dynamic pressure
- **Outputs**: Forces in y and z directions; Roll, Pitch, and Yaw moments
- **Limitations/Assumptions/Simplifications**: As small angles (<50 mrad) Normal force on the fins is considered to be parallel to the relevant body axis.
- **Description:** The product of fin deflection and Coefficient of fin normal force due to change in fin deflection ($C_{N\delta fin}$) is diminsionalized using dynamic pressure and reference area multiplied by moment arm where appropriate and summed.

$$M_{x\delta fin} = \sum_{fins} C_{N\delta fin}\delta_{fin}\bar{Q}S\bar{X}_{C.G.-fin}$$

- **Problems**: None identified.
- **Future Improvements:** None identified.
- **Additional Notes / References / Resources:** Currently implemented coefficient is a body normal force coefficient. More accurate results will be obtained by use of a coefficient relating to fin normal force only.

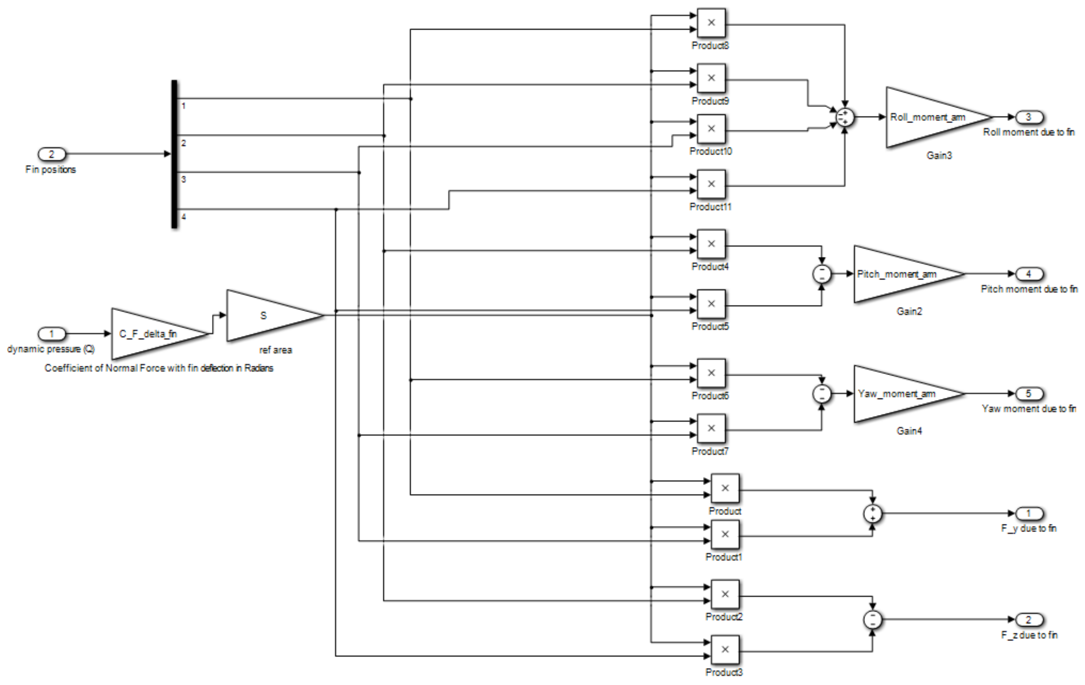


Figure 78. Forces and Moments Due to Fin Deflections Block

- **Block Name**: Actuator Model, Figure 79.
- **Location**: Sub-block of Plant Simulation
- **Purpose**: This block implements a model of the air vehicle's servo actuators.
- **Inputs**: Fin commands (radians)
- **Outputs**: Fin positions (radians)
- **Limitations/Assumptions/Simplifications**: Assumes the servos are linear second order. The damping ratio and natural frequency of the actuators was roughly calculated, with damping ratio estimated by to be 1.3 by observation of significantly overdamped behavior.
- **Description:** Linear Second Order Actuator blocks from the aerospace blockset are used to model the rockets actuators. *actuator_freq* and *actuator_damp* can be used in the setup script to change the assumed natural frequency and damping ratio of the actuators.
- **Problems**: None Identified.
- **Future Improvements:** A more exact measurement of actuator characteristics will lead to more exact frequency and damping. Further a non-linear actuator model is available in the blockset and may be more appropriate depending on what further measurement shows.
- **Additional Notes / References / Resources:** The behavior of the actuators is critical to an accurate tuning of the PID gains within the autopilot to ensure proper phase margin and for the model to accurately represent the system.

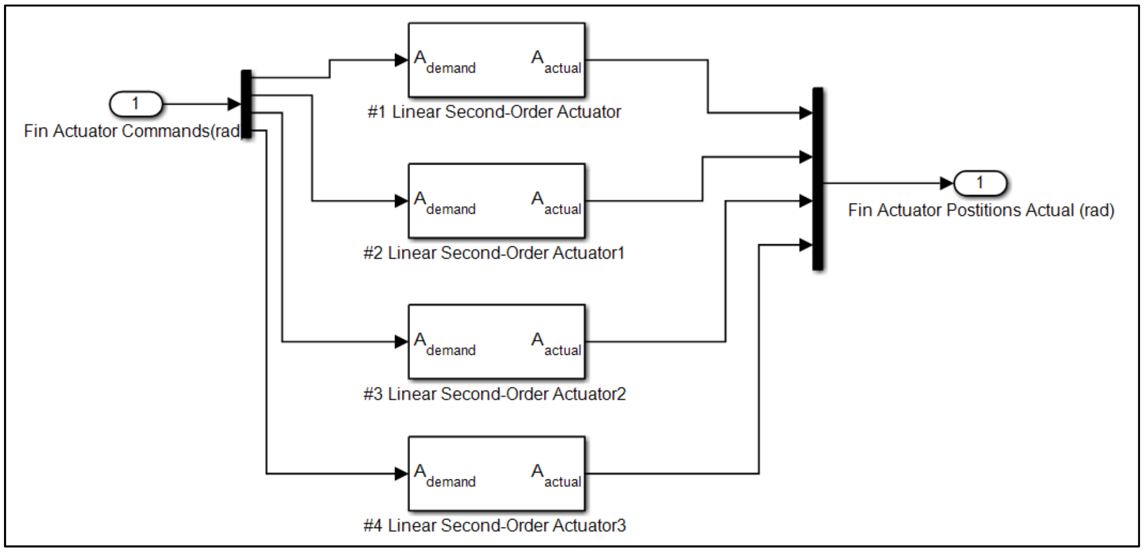

1
Fin Actuator Commands(rad
A_{demand}
A_{actual}
#1 Linear Second-Order Actuator
A_{demand}
A_{actual}
#2 Linear Second-Order Actuator1
A_{demand}
A_{actual}
#3 Linear Second-Order Actuator2
A_{demand}
A_{actual}
#4 Linear Second-Order Actuator3
1
Fin Actuator Postitions Actual (rad)

Figure 79. Actuator Model

- **Block Name**: Propulsion Model, Figure 80.
- **Location**: Sub-block of Plant Simulation
- **Purpose**: This block implements a model of the rocket engine.
- **Inputs**: None.
- **Outputs**: Thrust (N), mass flow rate (kg/s).

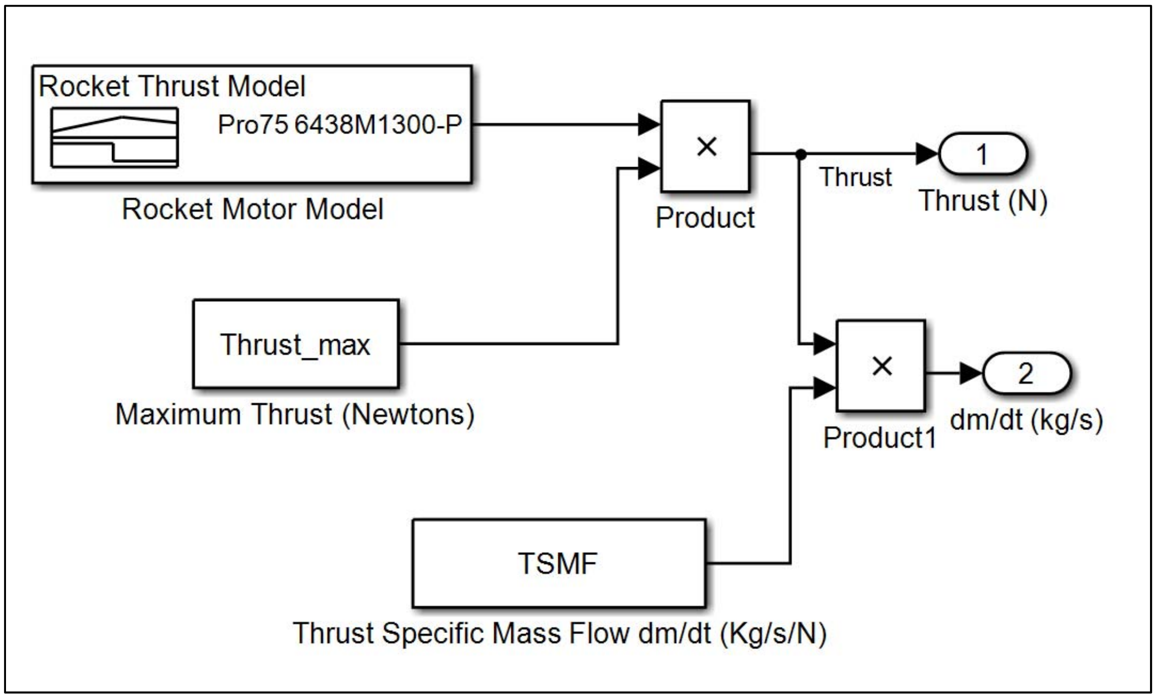


Figure 80. Propulsion Model

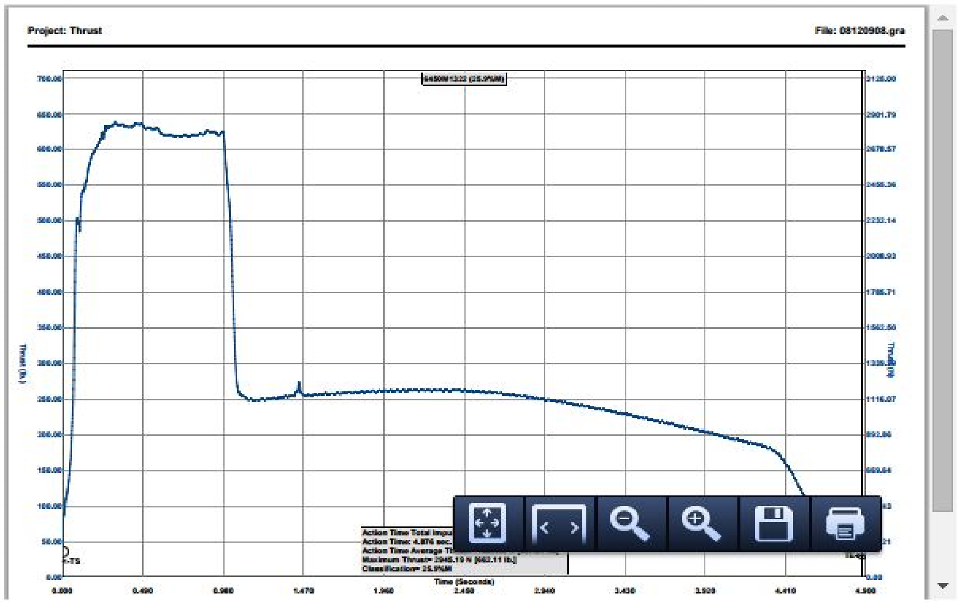
- **Limitations/Assumptions/Simplifications**: A simplified thrust model was used is assessed to have a contributed a negligible error when compared to the manufacturer provided specification for rocket thrust versus time. The thrust model is shown in Figure 81. The customized thrust for the simulations of the normalized rocket thrust is shown in Figure 82.

Pro75 6438M1300-P

Motor Data

Brandname	Pro75 6438M1300-P	**Manufacturer**	Cesaroni Technology
Man. Designation	6438M1300-P	**CAR Designation**	6438-M1300-P
Test Date	8/12/2009		
Single-Use/Reload/Hybrid	Reloadable	**Motor Dimensions mm**	75.00 x 757.00 mm (2.95 x 29.80 in)
Loaded Weight	5657 g	**Total Impulse**	6438.2 Ns (1447.4 lb-s)
Propellant Weight	3437 g	**Maximum Thrust**	2826.9 N (635.5 lb)
Burnout Weight	2062 g	**Avg Thrust**	1303.9 N (293.1 lb)
Delays Tested	plugged	**ISP**	191.0 s
Samples per second	1000	**Burntime**	4.94 s
Notes	25.7 % M		

Representative CMT Thrust Curve Click Here for Larger View of Graph



Action Time Total Impu
Action Time: 4.876 sec.
Action Time Average Th
Maximum Thrust= 2945.19 N [662.11 lb.]
Classification= 25.9%M

Figure 81. Rocket Thrust Measurement [27]

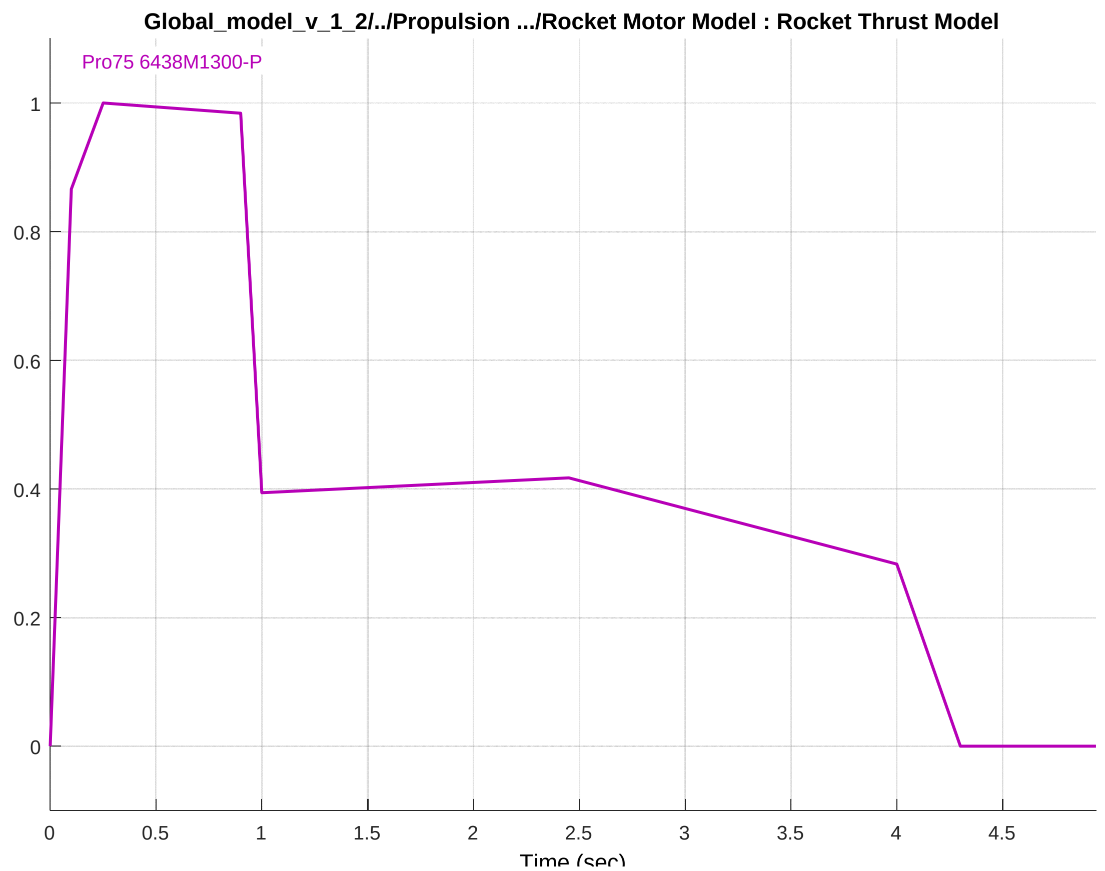


Figure 82. Custom Signal Representing Normalized Rocket Thrust

- **Description:** A normalized thrust profile was built into a custom signal block for the Pro75 6438M 1300-P rocket motor. When multiplied by *Thrust_max*, thrust over time is output. Thrust specific mass flow (*TSMF*) was calculated assuming that on average mass flow rate of the motor is proportional to thrust. Multiplied by thrust the mass flow rate is output for use elsewhere in the plant simulation.
- **Problems**: None Identified.
- **Future Improvements:** Additional motor models can easily be developed for other commercially available models to allow comparison of flight profiles with different engines.
- **Additional Notes / References / Resources:** None.

- **Block Name**: Gravity Model, Figure 83.
- **Location**: Sub-block of Plant Simulation
- **Purpose**: This block implements a model of gravity.
- **Inputs**: Mass flow rate out of the missile. Euler angles of the missile
- **Outputs**: Gravity Force Components (in body axes of missile)
- Limitations/Assumptions/Simplifications:
- **Description:** Gravity in the flat earth reference system is transformed into three dimensional forces acting on the missile body via trigonometric calculations and additions that account for the changing mass of the missile over time (given the characteristics of the rocket motor) and the changing orientation of the boy axis of the missile within the NED frame (pitch and roll angle). Of note, yaw angle has no effect.
- **Problems**: Gravity will always remain one integrator behind because the Euler angles are calculated earlier in the global model 6-DOF block. Thus, at high roll rates this lag allows enough time for forces to be generated and alpha and beta excursions to result (which at high missile roll rates can become unbounded and cause the missile to appear unstable).
- **Future Improvements:** The model could be more responsive and the errors minimized if the overall step size is reduced. This would be done at the expense of increased computation power and time required to run the simulation.
- **Additional Notes / References / Resources:**

The below matrix represents a three dimensional transformation from the body frame to the navigational (NED) frame.

$$R_y^x = \left[R_x^y\right]^T = \begin{bmatrix} c\theta c\psi & -c\theta s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

The transpose was used to go instead from the NED to the body frame:

$$R_x^y = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\theta s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix}$$

This matrix, when multiplied by the gravity vector in the NED frame provides the gravity contribution in the body axes of the missile. This is depicted below, with negligible or zero components ultimately removed:

$$\begin{bmatrix}\ddot{x}_b\\ \ddot{y}_b\\ \ddot{z}_b\end{bmatrix}=\begin{bmatrix}c\theta c\psi & c\theta s\psi & -s\theta\\ -c\theta s\psi+s\phi s\theta c\psi & c\phi c\psi+s\phi s\theta s\psi & s\phi c\theta\\ s\phi s\psi+c\phi s\theta c\psi & -s\phi c\psi+c\phi s\theta s\psi & c\phi c\theta\end{bmatrix}\begin{bmatrix}\ddot{x}_n\\ \ddot{y}_n\\ \ddot{z}_n\end{bmatrix}$$

$$=\begin{bmatrix}c\theta c\psi & c\theta s\psi & -s\theta\\ -c\theta s\psi+s\phi s\theta c\psi & c\phi c\psi+s\phi s\theta s\psi & s\phi c\theta\\ s\phi s\psi+c\phi s\theta c\psi & -s\phi c\psi+c\phi s\theta s\psi & c\phi c\theta\end{bmatrix}\begin{bmatrix}0\\ 0\\ 9.8\end{bmatrix}$$

$$\begin{bmatrix}\ddot{\boldsymbol{x}}_{\boldsymbol{b}}\\ \ddot{\boldsymbol{y}}_{\boldsymbol{b}}\\ \ddot{\boldsymbol{z}}_{\boldsymbol{b}}\end{bmatrix}=\mathbf{9.8}\begin{bmatrix}-\boldsymbol{s\theta}\\ \boldsymbol{s\phi c\theta}\\ \boldsymbol{c\phi c\theta}\end{bmatrix}$$

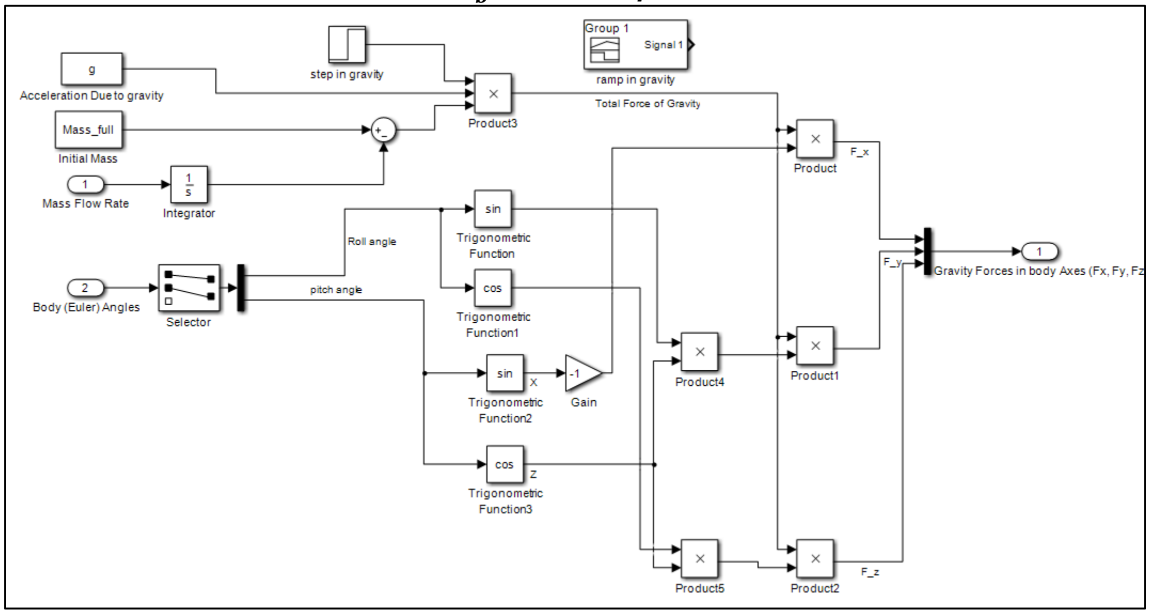


Figure 83. Gravity Model

- **Block Name**: 6 Degrees of Freedom (DOF) Euler Angles, Figure 84.
- **Location**: Sub-block of Plant Simulation
- **Purpose**: This block computes the Euler angles of a mass when subjected to forces and moments.
- **Inputs**: Total Forces (F_{xyz}) and Total Moments (M_{xyz}) acting upon the missile, in Newtonsand Newton-meters, respectively.
- **Outputs**: Missile velocity (V_e, V_b), Missile position (X_e), Euler angles (Φ, Θ, Ψ), missile acceleration (A_b)
- **Limitations/Assumptions/Simplifications**: The block assumes that the following equation is true:

$$\frac{M_{FULL}}{M_{EMPTY}} I_{EMPTY} = I_{FULL}$$

- **Description:** This is a standard SIMULINK block found in the Aerospace Block-Set.
- **Problems**: None noted.
- **Future Improvements**:
- **Additional Notes / References / Resources:** The following initial conditions are set using the global setup script: Initial Euler orientation *Euler_initial*, Initial mass *Mass_full*, Empty Mass *Mass_empty*, Full mass *Mass_full*, Empty inertia matrix *I_empty*, Full inertia matrix *I_full*.

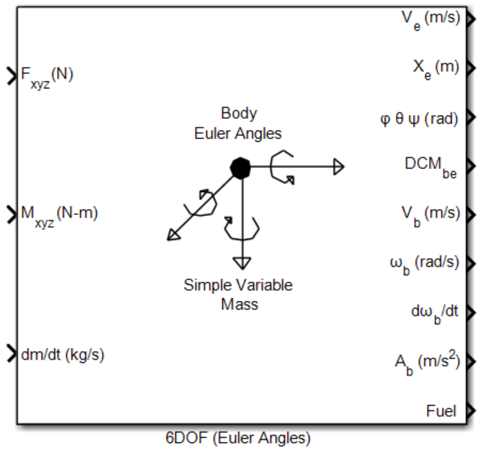


Figure 84. 6DOF Block from Simulink Aerospace Blockset

- **Block Name**: Q Bar Calculator, Figure 85.
- **Location**: Sub-block of Missile Aerodynamics, which is sub-block of Plant Simulation
- **Purpose**: Calculates the dynamic pressure.
- **Inputs**: Velocity, Height (MSL)
- **Outputs**: Dynamic Pressure
- **Limitations/Assumptions/Simplifications**: Assumes missile is below 11,000m (i.e., within the troposphere).
- **Description:** The block computes the dynamic pressure the missile is subjected to, adjusted for the change in air density as the missile changes altitude according to the below equations:

$$\rho = \rho_0 * \left(\frac{T_0}{T_0 + L_b * h}\right)^{1+\left(g*\frac{.0289644}{8.31432*-.0065}\right)}$$

$$\bar{Q} = \frac{\rho V^2}{2}$$

- **Problems**: None noted.

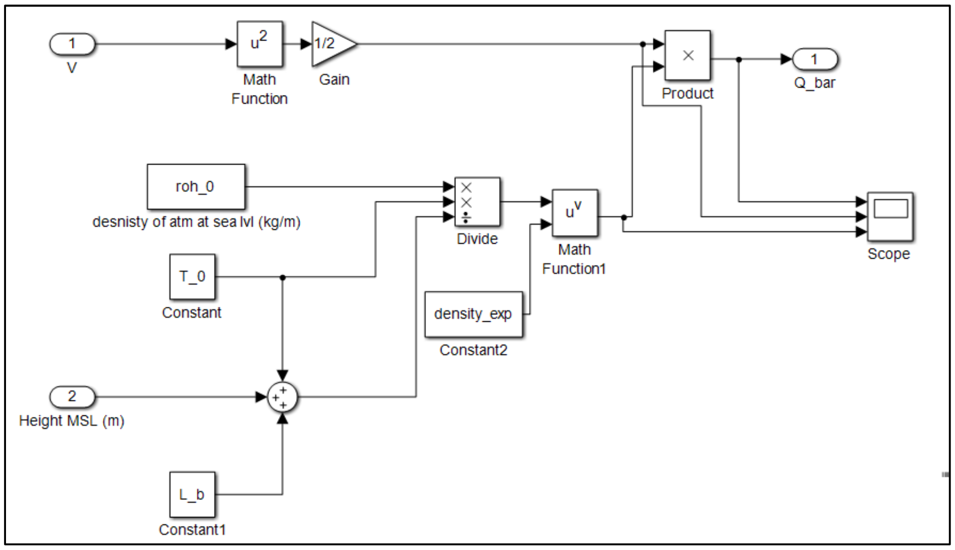


Figure 85. Q Bar Calculator

- **Block Name**: Wind Angles, Figure 86.
- **Location**: Sub-block of Missile Aerodynamics, which is sub-block of Plant Simulation
- **Purpose**: Compute the angle of attack (alpha) and angle of sideslip (beta) of the air vehicle.
- **Inputs**: Missile velocity (body axes)
- **Outputs**: Alpha (radians), Beta (radians)
- Limitations/Assumptions/Simplifications: None.
- **Description:** The body velocity of the missile is broken into components and trigonometry is used to compute the required angles.
- **Problems**: None noted.
- **Future Improvements:** None noted.
- **Additional Notes / References / Resources**:

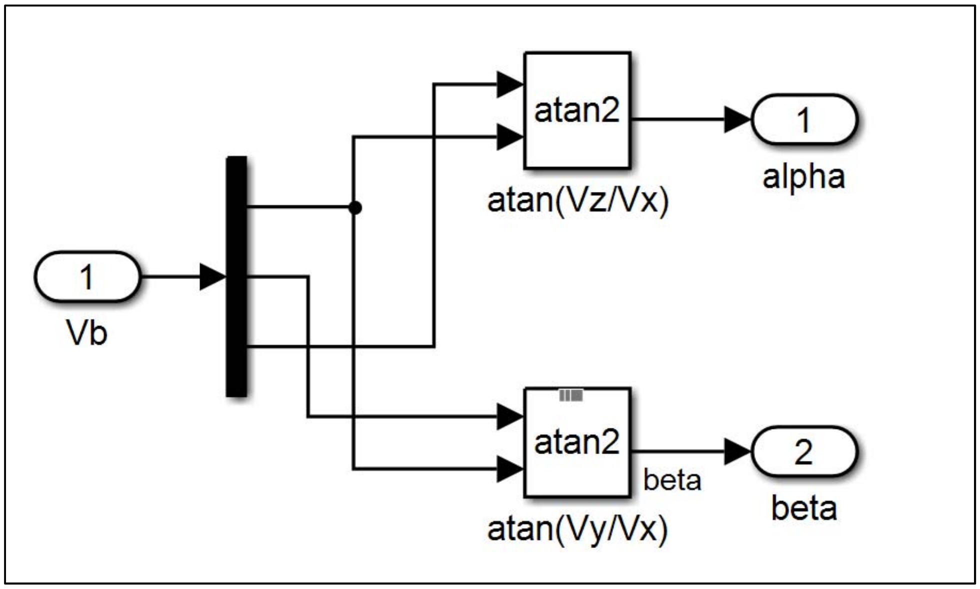


Figure 86. Wind Angles Block

- **Block Name**: Position Simulator, Figure 87.
- **Location**: Sub-block of Plant Simulation
- **Purpose**: Simulate the output of the actual GPS hardware by converting the position output by the 6DOF model in local tangent plane (NED) into geodetic (Lat Long Altitude or LLA).
- **Inputs**: NED position
- **Outputs**: Geodetic Position (LLA) and Altitude MSL (used in static pressure calculation)
- **Limitations/Assumptions/Simplifications**: Launch Position is specified in the setup script variable *Launch_Position.*
- **Description:** Flat Earth to LLA block from the Aerospace blockset is used to convert the output of the 6DOF into earth coordinates. This allows the output of the plant simulation block to emulate the input of the GPS in the real-world missile. Additionally, MSL altitude is output for use in calculating static pressure and thus dynamic pressure.
- **Problems**: None noted.
- **Future Improvements:** None noted.
- **Additional Notes / References / Resources:**

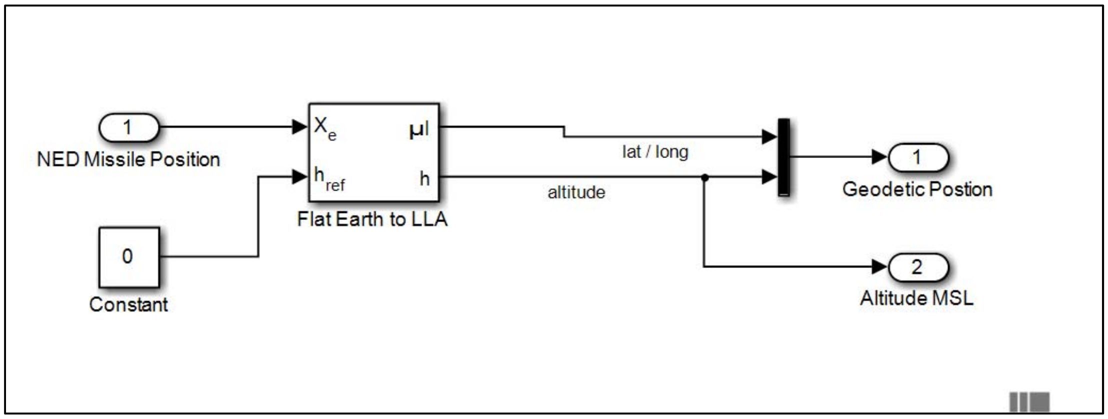
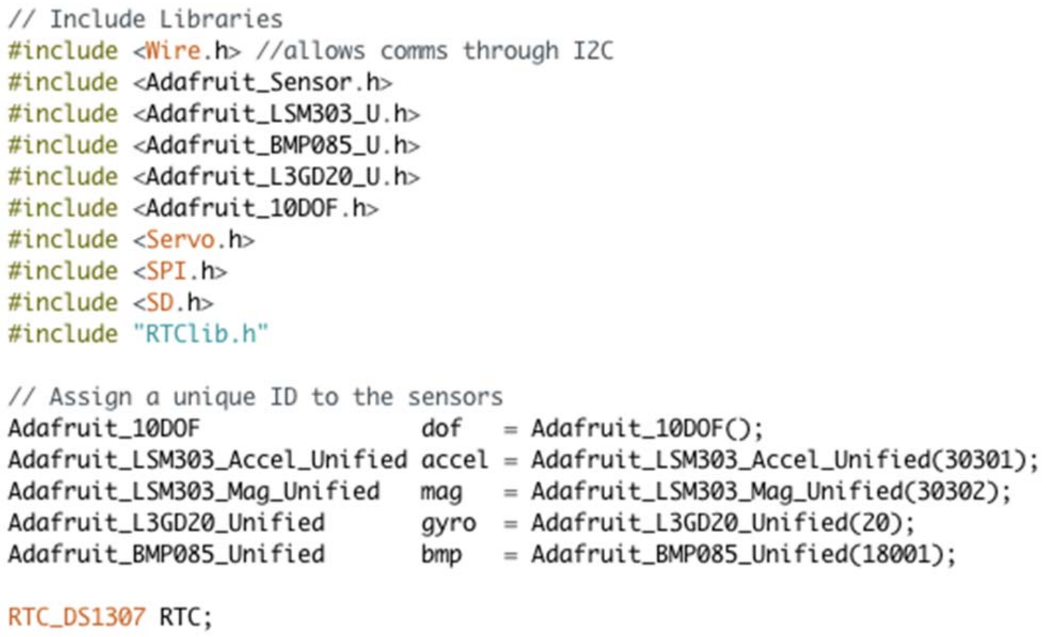
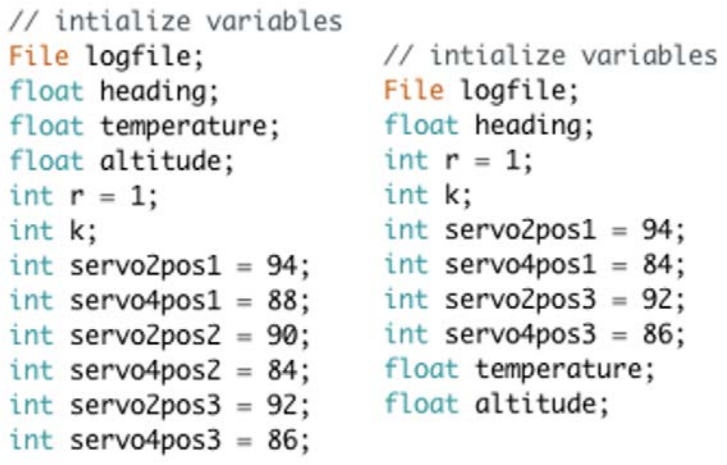


Figure 87. Position Simulator

APPENDIX B. ARDUINO LAUNCH CODE

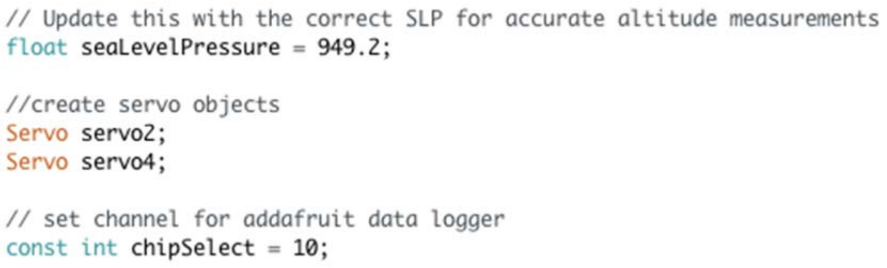


```
// Include Libraries
#include <Wire.h> //allows comms through I2C
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Adafruit_BMP085_U.h>
#include <Adafruit_L3GD20_U.h>
#include <Adafruit_10DOF.h>
#include <Servo.h>
#include <SPI.h>
#include <SD.h>
#include "RTClib.h"

// Assign a unique ID to the sensors
Adafruit_10DOF                dof   = Adafruit_10DOF();
Adafruit_LSM303_Accel_Unified accel = Adafruit_LSM303_Accel_Unified(30301);
Adafruit_LSM303_Mag_Unified   mag   = Adafruit_LSM303_Mag_Unified(30302);
Adafruit_L3GD20_Unified       gyro  = Adafruit_L3GD20_Unified(20);
Adafruit_BMP085_Unified       bmp   = Adafruit_BMP085_Unified(18001);

RTC_DS1307 RTC;
```



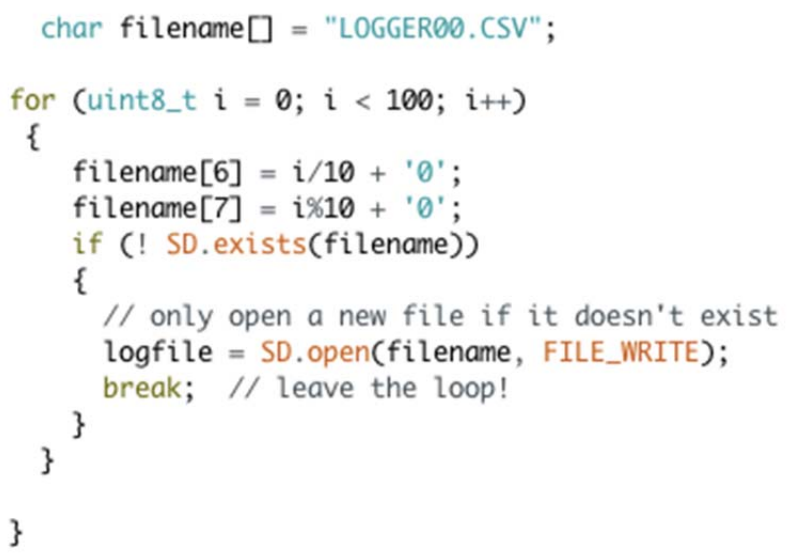
```
// intialize variables
File logfile;
float heading;
float temperature;
float altitude;
int r = 1;
int k;
int servo2pos1 = 94;
int servo4pos1 = 88;
int servo2pos2 = 90;
int servo4pos2 = 84;
int servo2pos3 = 92;
int servo4pos3 = 86;
```

```
// intialize variables
File logfile;
float heading;
int r = 1;
int k;
int servo2pos1 = 94;
int servo4pos1 = 84;
int servo2pos3 = 92;
int servo4pos3 = 86;
float temperature;
float altitude;
```

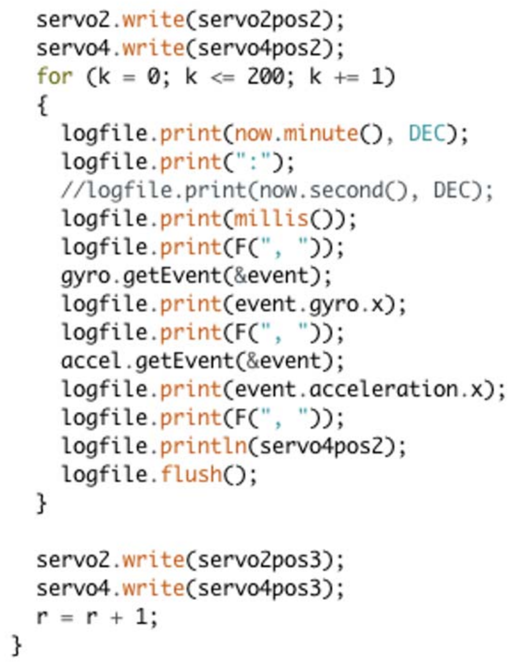


```
// Update this with the correct SLP for accurate altitude measurements
float seaLevelPressure = 949.2;

//create servo objects
Servo servo2;
Servo servo4;

// set channel for addafruit data logger
const int chipSelect = 10;
```

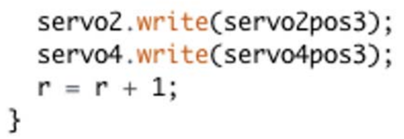
```
//Initialises the sensors for the 10DOFSensor
void initSensors()
{
  if(!accel.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    Serial.println(F("Ooops, no LSM303 detected ... Check your wiring!"));
    while(1);
  }
    if(!mag.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }
  if(!bmp.begin())
  {
    /* There was a problem detecting the BMP180 ... check your connections */
    Serial.println("Ooops, no BMP180 detected ... Check your wiring!");
    while(1);
  }
  if(!gyro.begin())
  {
    /* There was a problem detecting the L3GD20 ... check your connections */
    Serial.print("Ooops, no L3GD20 detected ... Check your wiring or I2C ADDR!");
    while(1);
  }
}


void setup(void)
{
  Serial.begin(115200);
  Wire.begin();
  RTC.begin();
  DateTime now;
  servo2.attach(9);
  servo4.attach(3);

  //See if the card is present and can be initialized:
  if (!SD.begin(chipSelect))
  {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println("card initialized.");

  //Initialise the sensors
  initSensors();
```



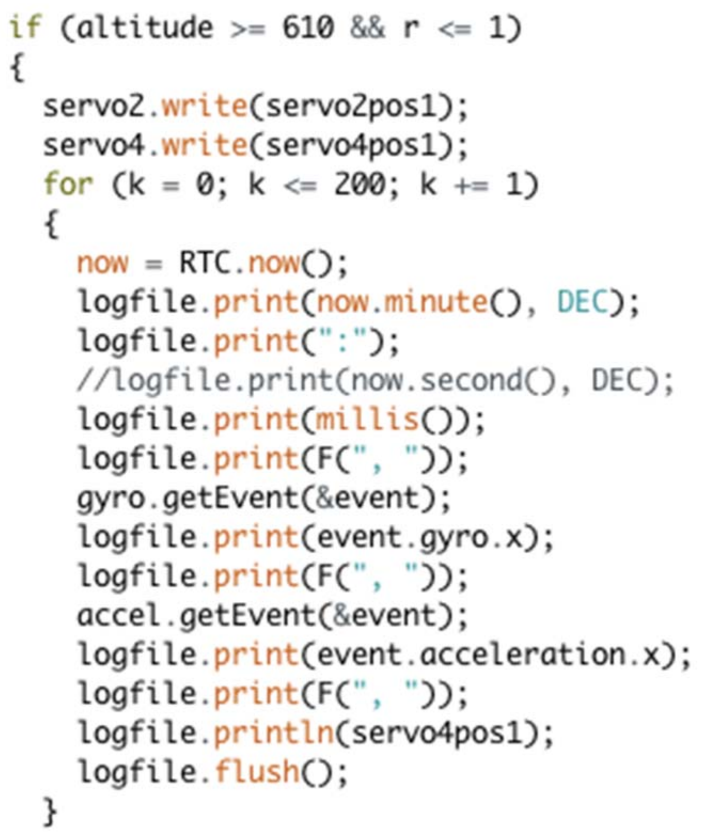
```
  char filename[] = "LOGGER00.CSV";

for (uint8_t i = 0; i < 100; i++)
 {
    filename[6] = i/10 + '0';
    filename[7] = i%10 + '0';
    if (! SD.exists(filename))
    {
      // only open a new file if it doesn't exist
      logfile = SD.open(filename, FILE_WRITE);
      break;  // leave the loop!
    }
  }

}
```



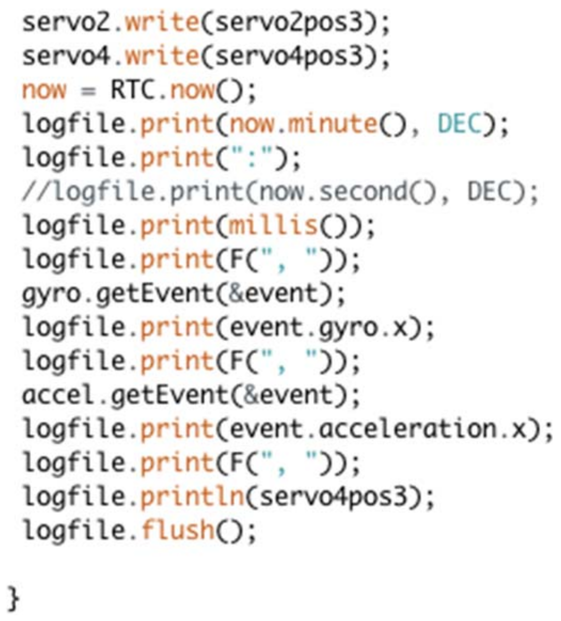
```
    servo2.write(servo2pos2);
    servo4.write(servo4pos2);
    for (k = 0; k <= 200; k += 1)
    {
      logfile.print(now.minute(), DEC);
      logfile.print(":");
      //logfile.print(now.second(), DEC);
      logfile.print(millis());
      logfile.print(F(", "));
      gyro.getEvent(&event);
      logfile.print(event.gyro.x);
      logfile.print(F(", "));
      accel.getEvent(&event);
      logfile.print(event.acceleration.x);
      logfile.print(F(", "));
      logfile.println(servo4pos2);
      logfile.flush();
    }

    servo2.write(servo2pos3);
    servo4.write(servo4pos3);
    r = r + 1;
  }
```



```
    servo2.write(servo2pos3);
    servo4.write(servo4pos3);
    r = r + 1;
  }
```

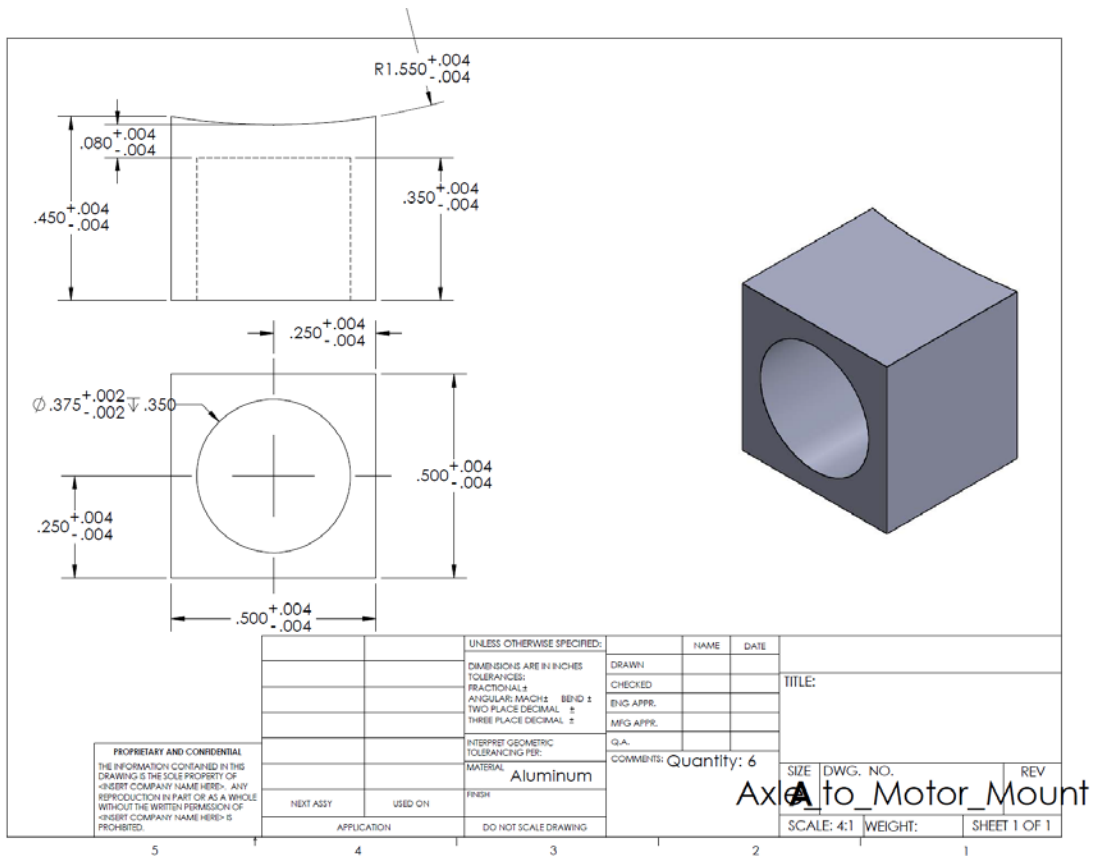


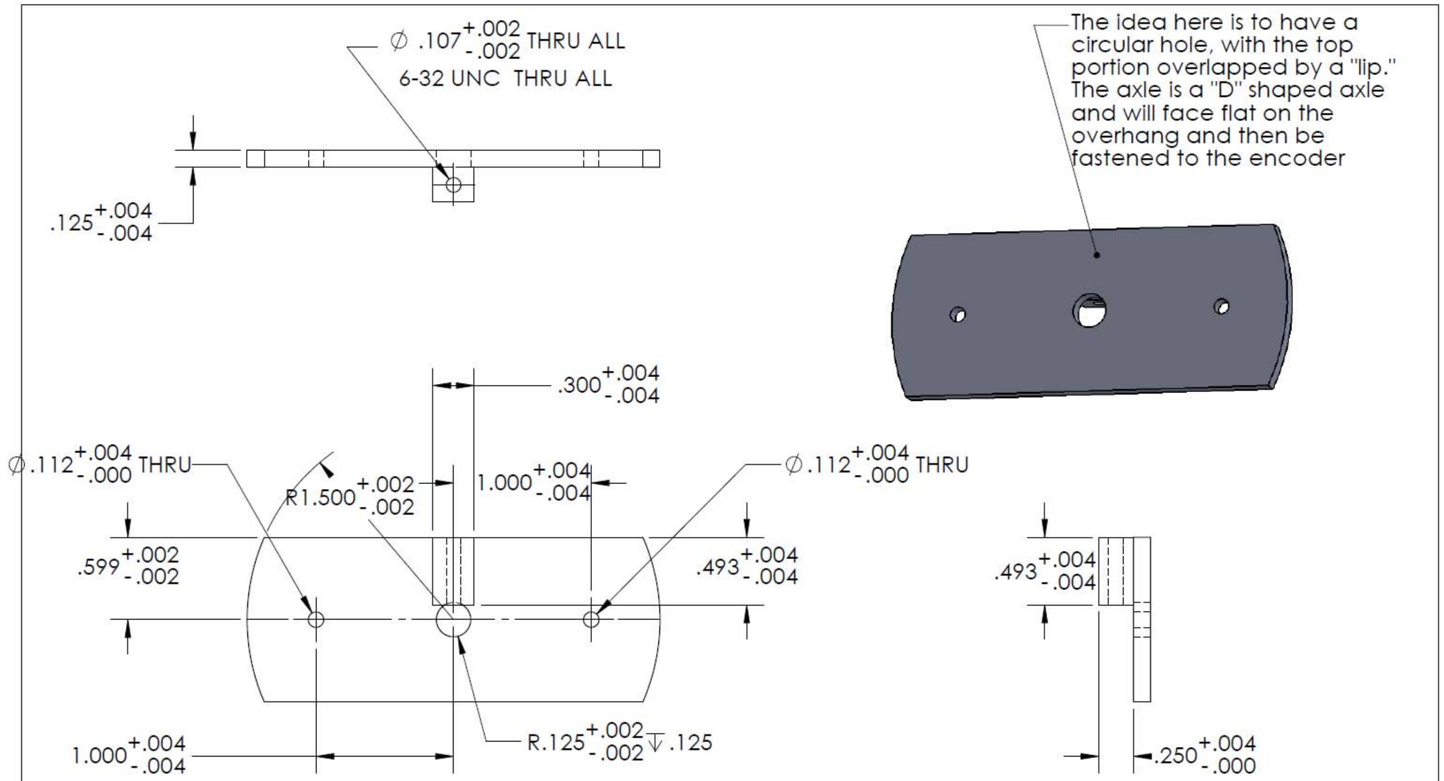
```
if (altitude >= 610 && r <= 1)
{
  servo2.write(servo2pos1);
  servo4.write(servo4pos1);
  for (k = 0; k <= 200; k += 1)
  {
    now = RTC.now();
    logfile.print(now.minute(), DEC);
    logfile.print(":");
    //logfile.print(now.second(), DEC);
    logfile.print(millis());
    logfile.print(F(", "));
    gyro.getEvent(&event);
    logfile.print(event.gyro.x);
    logfile.print(F(", "));
    accel.getEvent(&event);
    logfile.print(event.acceleration.x);
    logfile.print(F(", "));
    logfile.println(servo4pos1);
    logfile.flush();
  }
```



```
 servo2.write(servo2pos3);
 servo4.write(servo4pos3);
 now = RTC.now();
 logfile.print(now.minute(), DEC);
 logfile.print(":");
 //logfile.print(now.second(), DEC);
 logfile.print(millis());
 logfile.print(F(", "));
 gyro.getEvent(&event);
 logfile.print(event.gyro.x);
 logfile.print(F(", "));
 accel.getEvent(&event);
 logfile.print(event.acceleration.x);
 logfile.print(F(", "));
 logfile.println(servo4pos3);
 logfile.flush();

}
```

APPENDIX C. ENGINEERING DRAWINGS FOR MANUFACTURED PIECES





The idea here is to have a circular hole, with the top portion overlapped by a "lip." The axle is a "D" shaped axle and will face flat on the overhang and then be fastened to the encoder

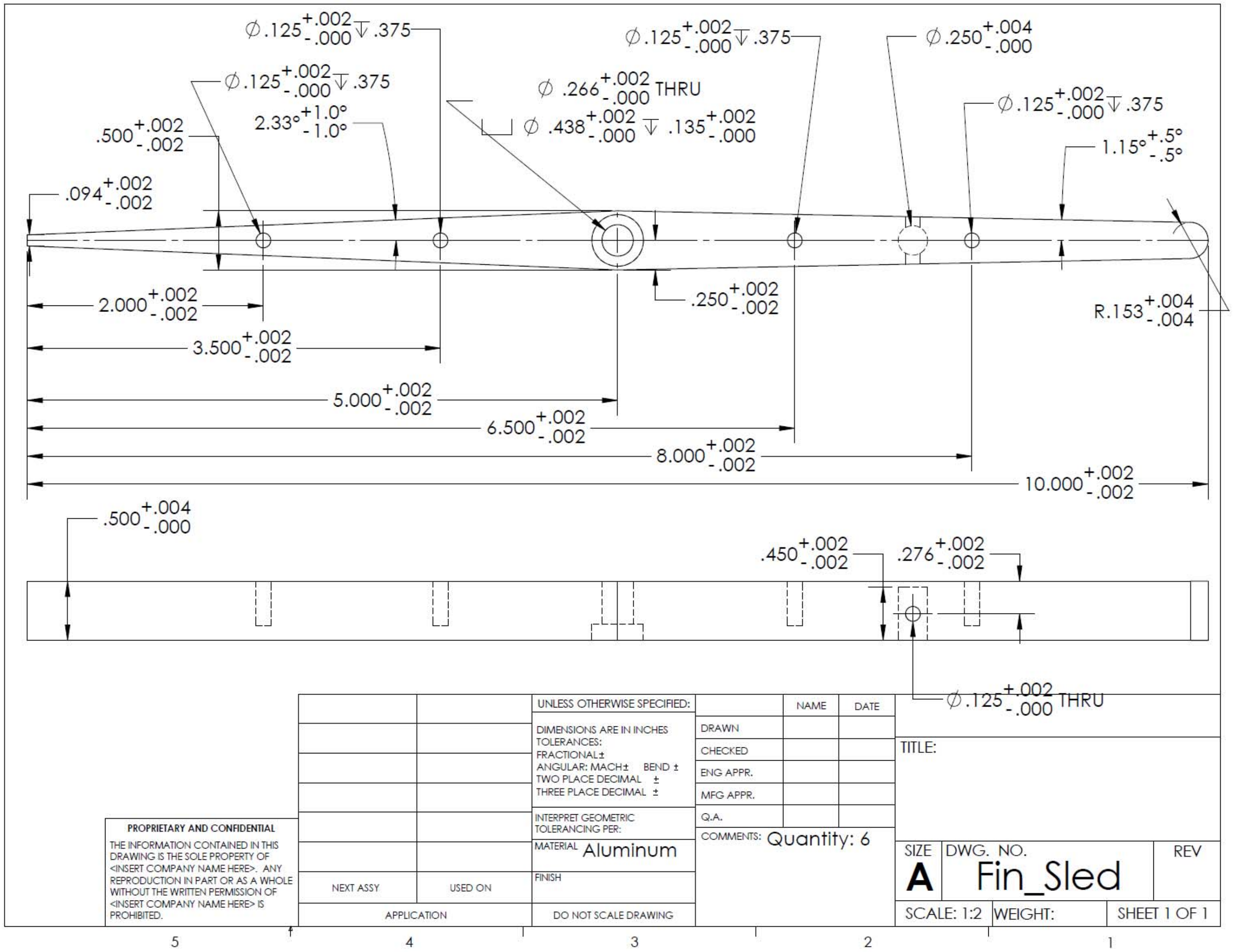
PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

NEXT ASSY	USED ON	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL± ANGULAR: MACH± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±		NAME	DATE
		INTERPRET GEOMETRIC TOLERANCING PER:	DRAWN		
		MATERIAL Aluminum	CHECKED		
		FINISH	ENG APPR.		
APPLICATION		DO NOT SCALE DRAWING	MFG APPR.		
			Q.A.		
			COMMENTS: Quantity: 6		

TITLE:		
SIZE A	DWG. NO. encoder1	REV
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1



⌀.125 +.002 -.000 ↧ .375
⌀.125 +.002 -.000 ↧ .375
⌀.125 +.002 -.000 ↧ .375
⌀.125 +.002 -.000 ↧ .375
⌀.250 +.004 -.000
⌀ .266 +.002 -.000 THRU
⌴ ⌀ .438 +.002 -.000 ↧ .135 +.002 -.000
2.33° +1.0° -1.0°
1.15° +.5° -.5°
.500 +.002 -.002
.094 +.002 -.002
.250 +.002 -.002
R.153 +.004 -.004
2.000 +.002 -.002
3.500 +.002 -.002
5.000 +.002 -.002
6.500 +.002 -.002
8.000 +.002 -.002
10.000 +.002 -.002
.500 +.004 -.000
.450 +.002 -.002
.276 +.002 -.002
⌀.125 +.002 -.000 THRU
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL±
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL Aluminum
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS: Quantity: 6
TITLE:
SIZE A
DWG. NO. Fin_Sled
REV
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
5
4
3
2
1

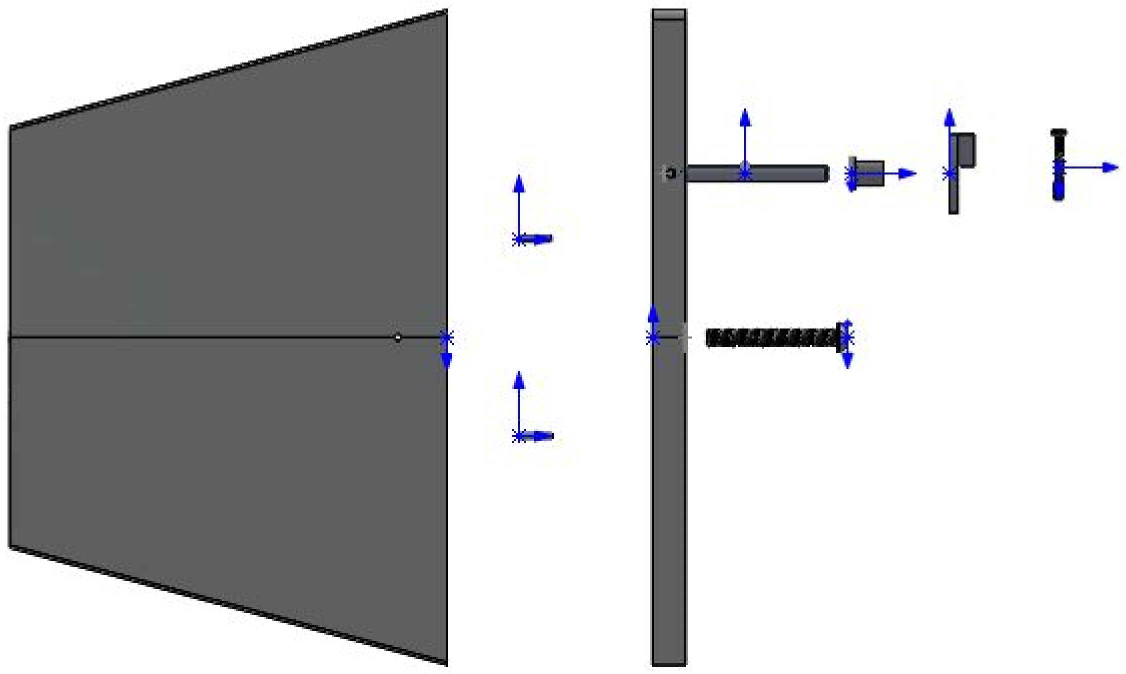
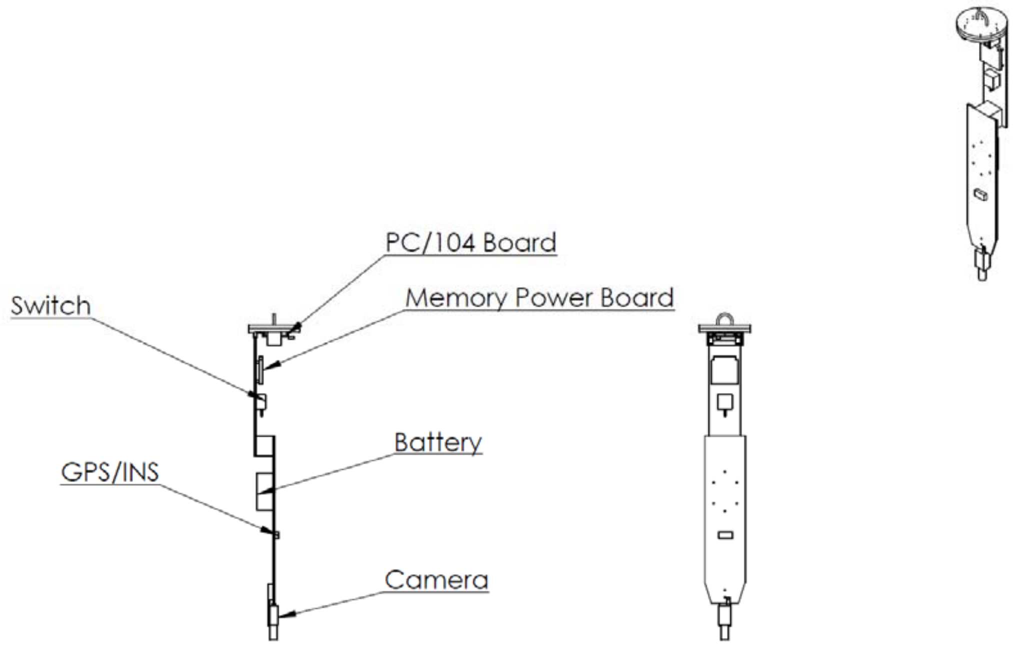


Figure 88. Fins to Servo Control Horn Hardware

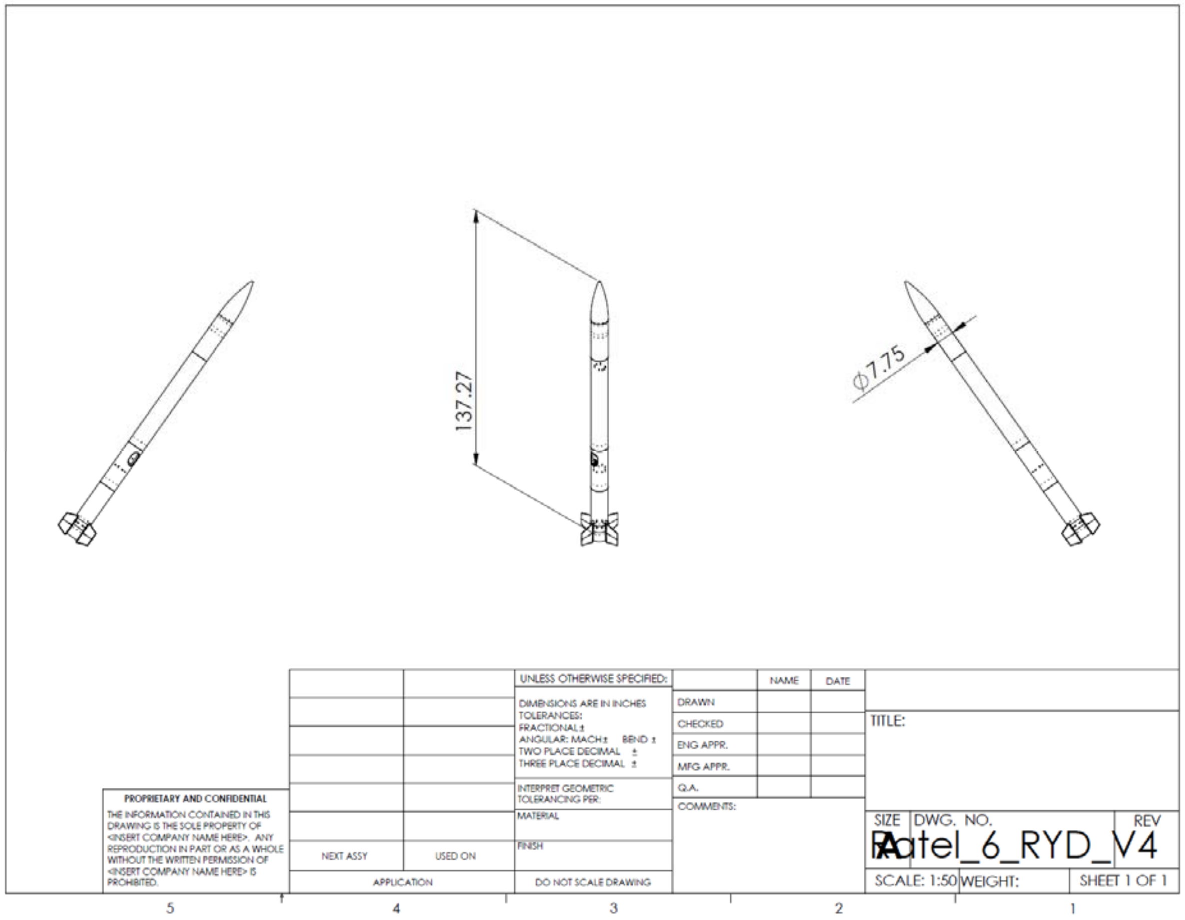


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE	
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN			
			CHECKED			TITLE:
			ENG APPR.			
			MFG APPR.			
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.			
		MATERIAL	COMMENTS:			SIZE A DWG. NO. Nosecone REV
NEXT ASSY	USED ON	FINISH				
APPLICATION		DO NOT SCALE DRAWING				SCALE: 1:20 WEIGHT: SHEET 1 OF 1

PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

5 4 3 2 1


137.27
ϕ7.75
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE A
DWG. NO. Ratel_6_RYD_V4
REV
SCALE: 1:50
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
5
4
3
2
1

APPENDIX D. RAW DATA FROM FLIGHT

A. RAW DATA FROM LAUNCH ONE

GPS/INS Data in Nose Cone

Roll (rad)	Pitch (rad)	Yaw (rad)	X AngRate (rad/s)	Y AngRate (rad/s)	Z AngRate (rad/s)	Xaccl (m/s^2)	Yaccl (m/s^2)	Zaccl (m/s^2)	Lat (deg)	Long (deg)	Ht (m)	Time (sec)
-2.507	1.549	-1.367	0.014	-0.002	0.000	0.498	-0.029	0.723	35.348	-117.809	603.614	0.000
-2.509	1.547	-1.370	0.229	0.265	-0.139	121.302	7.510	15.173	35.348	-117.809	603.621	0.040
-2.689	1.537	-1.551	-0.195	0.176	-0.033	78.660	1.688	19.447	35.348	-117.809	603.732	0.080
-2.779	1.526	-1.644	-0.243	0.117	0.017	79.736	1.865	7.615	35.348	-117.809	603.970	0.120
1.471	1.507	0.510	-3.097	0.016	-0.021	-11.066	0.615	0.170	35.348	-117.809	1267.044	6.762
1.355	1.508	0.517	-3.074	0.018	-0.024	-11.025	0.618	0.085	35.348	-117.809	1270.313	6.802
-2.274	1.527	-0.168	-4.404	0.043	-0.029	-10.834	-0.038	0.050	35.348	-117.809	1346.030	7.762
-2.462	1.525	-0.172	-4.821	0.032	-0.028	-10.793	-0.145	0.130	35.348	-117.809	1348.882	7.802
-2.655	1.523	-0.164	-5.272	0.029	-0.036	-10.829	-0.223	-0.037	35.348	-117.809	1351.717	7.842
-1.235	0.849	2.747	-8.490	-0.075	-0.067	-7.477	-6.069	3.775	35.348	-117.809	1591.521	12.564
-1.572	0.846	2.750	-8.490	-0.073	-0.073	-7.439	-6.483	1.495	35.348	-117.809	1592.566	12.604
-1.908	0.843	2.755	-8.489	-0.070	-0.079	-7.438	-6.118	-0.738	35.348	-117.809	1593.594	12.644
-0.786	0.947	-2.053	-8.489	-0.060	-0.092	-8.051	-3.983	5.415	35.348	-117.809	1594.976	12.684
-1.126	0.942	-2.053	-8.488	-0.054	-0.099	-8.049	-5.142	3.752	35.348	-117.809	1595.963	12.724
-2.668	1.135	-2.779	-3.458	0.053	0.020	-29.782	-2.412	1.965	35.348	-117.809	1632.482	16.088
-2.815	1.142	-2.785	-3.531	0.094	0.047	22.589	-4.882	-34.047	35.348	-117.809	1632.234	16.128

Arduino Data Launch 1

Time		AngRate (rad/s)		ServoPos
0.000058		0.92		86
0.000116		1.08		86
0.000174		1.27		86
0.000232		1.51		86
0.001856		4.85		86
0.001914		4.81		86
0.001972		4.8		86
0.00203		4.79		86
0.006612		2.77		88
0.00667		2.77		88
0.006728		2.81		88
0.006786		2.83		88
0.006844		2.89		88
0.007656		4.82		88
0.007714		4.99		88
0.007772		5		88
0.026854		5		84
0.026912		5		84
0.02697		4.99		84
0.053708		-3.02		86
0.053766		-3.03		86
0.053824		-3.03		86
0.072384		-0.01		86
0.072442		0		86
0.0725		0.01		86
0.075864		0		86

B. RAW DATA FROM LAUNCH TWO

GPS/INS Data in Nose Cone

Roll (rad)	Pitch (rad)	Yaw (rad)	X AngRa (rad/s)	Y AngRa (rad/s)	Z AngRa (rad/s)	Xaccl (m/s^2)	Yaccl (m/s^2)	Zaccl (m/s^2)	Lat (deg)	Long (deg)	Ht (m)	Time (sec)
-2.766	1.539	-1.553	-0.087	-0.084	0.026	37.720	-0.187	-3.322	35.348	-117.809	602.100	0
-2.789	1.540	-1.567	-0.181	0.010	0.020	37.394	0.233	-1.327	35.348	-117.809	602.446	0.04
-2.774	1.542	-1.546	-0.194	-0.130	0.008	36.741	0.157	0.099	35.348	-117.809	602.852	0.08
-2.037	1.480	2.156	-1.714	-0.018	-0.074	9.453	-0.139	-0.458	35.348	-117.809	881.822	3.884
-2.085	1.478	2.177	-1.681	-0.041	-0.039	7.884	-0.210	-2.117	35.348	-117.809	887.094	3.924
2.252	1.261	2.440	-8.471	1.289	-0.524	-14.551	-0.267	-3.478	35.348	-117.809	1274.064	7.286
2.077	1.248	2.612	-8.478	1.334	-0.410	-14.239	1.124	-3.400	35.348	-117.809	1277.885	7.326
1.894	1.238	2.778	-8.487	1.329	-0.248	-14.115	2.248	-1.637	35.348	-117.809	1282.142	7.366
1.706	1.234	2.938	-8.493	1.286	-0.132	-14.289	1.626	0.775	35.348	-117.809	1285.918	7.406
0.964	1.354	-0.903	-2.960	1.460	0.823	-15.764	3.977	1.931	35.348	-117.808	1326.305	7.848
1.162	1.356	-0.593	-2.226	1.548	0.685	-15.655	5.054	0.373	35.348	-117.808	1330.416	7.888
1.392	1.352	-0.280	-1.590	1.626	0.483	-15.131	4.772	2.581	35.348	-117.808	1333.881	7.928
1.605	1.343	0.012	-1.911	1.606	0.243	-14.997	7.699	0.735	35.348	-117.808	1337.323	7.968
-1.999	-1.215	0.205	-8.528	0.818	0.518	8.937	-3.877	0.748	35.348	-117.808	1554.954	13.138
-2.240	-1.206	0.098	-8.545	0.499	0.849	9.166	-3.376	-0.090	35.348	-117.808	1555.282	13.178
Roll (rad)	Pitch (rad)	Yaw (rad)	X spin (rad/s)	Y spin (rad/s)	Z spin (rad/s)	Xaccl (m/s^2)	Yaccl (m/s^2)	Zaccl (m/s^2)	Lat (deg)	Long (deg)	Ht (m)	Time (sec)

Arduino Data Launch 2

AngRate (rad/s)		ServoPos		Time
0		86		0
-0.03		86		0.000058
0.07		86		0.000116
4.88		84		0.008584
4.99		84		0.008642
5		84		0.0087
1.62		84		0.013282
1.79		84		0.01334
4.81		84		0.017342
5		84		0.0174
5		84		0.017458
5		86		0.024012
4.94		86		0.02407
4.65		86		0.024998
4.54		86		0.025056
-0.58		86		0.031262
-0.6		86		0.03132
-0.52		86		0.037352
-0.51		86		0.03741
-1.25		86		0.049764
-1.24		86		0.049822

LIST OF REFERENCES

[1] U. Gaerther, “UAV swarm tactics: An agent-based simulation and Markov process analysis,” Naval Postgraduate School, Monterey, California.

[2] P. Scharre, “Counter-swarm: A guide to defeating robotic swarms,” 31 March 2015. [Online]. Available: http://warontherocks.com/2015/03/counter-swarm-a-guide-to-defeating-robotic-swarms/. [Accessed 2 March 2016].

[3] L.Pham, D. Balbuena, M, Casserly, B. Dickerson, S. Graves, V. Maldonado, B. Pandya, and J, Sanders, “UAV swarm attack: Protection system alternatives,” Naval Postgraduate School, Monterey, 2012.

[4] D. Hambling, “U.S. Navy plans to fly first drone swarm this summer,” 4 January 2016. [Online]. Available: http://www.defensetech.org/2016/01/04/u-s-navy-plans-to-fly-first-drone-swarm-this-summer/. [Accessed 16 February 2016].

[5] K. Osborn, “Air Force developing swarms of mini-drones,” 27 May 2015. [Online]. Available: http://www.defensetech.org/2015/05/27/air-force-developing-swarms-of-mini-drones/. [Accessed 29 February 2016].

[6] J. Koebler, “Chinese drone ‘swarms’ designed to attack American aircraft carriers,” *U.S. News and World Report*, 2013.

[7] U.S. Navy, “A Cooperative Strategy for 21st Century Seapower,” March 2015. [Online]. Available: http://www.navy.mil/local/maritime/150227-CS21R-Final.pdf. [Accessed 7 January 2016].

[8] V. Clark, “Sea Power 21,” 2002. [Online]. Available: http://www.navy.mil/navydata/cno/proceedings.html. [Accessed 17 May 2016].

[9] N857, O., “Initial Capabilities Document for Maritime Expeditionary Security,” 2012. [Online]. Available: http://www.marines.mil/Portals/59/Med%20Res%20Maritime%20Security%20Cooperation_An%20Integrated_USN-USMC-USCG_Approach%20w%20PCN.pdf.

[10] Department of the Army, “Request for information: counter unmanned aerial system capability,” 20 February 2014. [Online]. Available: https://www.fbo.gov/index?s=opportunity&mode=form&id=94d4624458cac9978a69abc1ff6ccbd3&tab=core&_cview=0. [Accessed 29 February 2016].

[11] A. Grimm, “TR-8 Model Rocket Technical Report,” 2012. [Online]. Available: http://www2.estesrockets.com/pdf/1948MRST.pdf. [Accessed 26 April 2016].

[12] W. Colburn, “Where Did Model Rocketry Really Start?,” 2012. [Online]. Available: http://www.apogeerockets.com/education/downloads/ Newsletter314_large.pdf. [Accessed 26 April 2016].

[13] Christopher Lovelace, Jon Silverberg, Paul Wright, Collier Crouch, Amy Gabriel, Tan, Choon Ming, Martin, Teo Huifen, Serene, Yang, Kangjie, Roy, “ME4704 Missile Design,” Monterey, CA, 2014.

[14] Rydalch, Fletcher D.; Aguilar, Alejandro; Aldridge, Brent; Dirk, John; Hartman, Travis; Lim, Wee Yoew, “ME 4704 Missile Design,” Naval Postgraduate School, Monterey CA, 2016.

[15] Servocity, “How do servos work?” Robotzone, LLC., 1999–2015. [Online]. Available: https://www.servocity.com/html/ how_do_servos_work_.html#.V0fPi5ErKCg. [Accessed 26 May 2016].

[16] “Adafruit Arduino,” Adafruit [Online]. Available: https://www.adafruit.com/ category/17. [Accessed 8 June 2016].

[17] LORD MicroStrain, “LORD MicroStrain Sensing Systems,” 2015. [Online]. Available: http://files.microstrain.com/3DM-GX4-15_User_Manual_(8500-0046).pdf. [Accessed 16 May 2016].

[18] J. Lucas, “livescience.com,” 20 September 2014. [Online]. Available: http://www.livescience.com/47930-what-is-aerodynamics.html. [Accessed 7 March 2016].

[19] G. Siourus, “Aerodyanamic forces and coefficients,” in *Missile Guidance and Control Systems*, New York, Springer-Verlag, 2004, p. 681.

[20] L. Brown, M. Auman and K. Kirby, “Technical Report RDMR-SS-12-08,” in *MissileLab User’s Guide*, Huntsville, AL. USA., U.S ARMY, 2012, p. 129.

[21] C. Yang Quan and D. Xue, *System Simulation Techniques with MATLAB and Simulink*, John Wiley & Sons, Ltd., 2014.

[22] G. M. Siouris, *Missile Guidance and Control Systems*, New York: Springer-Verlag, 2004.

[23] Featherweight Altimeters,” Featherweight Altimeters LLC, 24 April 2013. [Online]. Available: http://www.featherweightaltimeters.com/uploads/ Raven_Users_Manual_13April24b.pdf. [Accessed 11 April 2016].

[24] Cesaroni Technology, “Pro75 high power rocket motor reload kits,” Cesaroni Technology Incorporated, 2012. [Online]. Available: http://www.pro38.com/ products/pro75/motor/MotorData.php?prodid=6118M3100-P. [Accessed 19 April 2016].

[25] Cesaroni Technology Incorporated, “Pro75 high power rocket motor reload kits,” Cesaroni, 2012. [Online]. Available: http://www.pro38.com/products/pro75/motor/MotorData.php?prodid=6819M1540-P. [Accessed 19 April 2016].

[26] V. Dobrokhodov, Missile flight and control, Carmel, CA, 2015.

[27] Cesaroni Technology, “Pro75 6438M1300-P Representative CMT Thrust Curve,” Cesaroni Technology, 2009.

[28] N. Goyer, “aircraft market place,” 20 August 2009. [Online]. Available: http://www.acmp.com/blog/why-all-the-tails-incidence-or-angle-of-attack.html. [Accessed 12 March 2016].

[29] M. F. Platzer, Missile Aerodynamics, Monterey, CA. NPS class notes, 2015.

[30] NASA, “NASA,” 05 May 2015. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/drag1.html. [Accessed 07 March 2016].

[31] J. Scott, “aerospaceweb.org,” 19 September 2004. [Online]. Available: http://www.aerospaceweb.org/question/aerodynamics/q0194.shtml. [Accessed 2016 March 10].

[32] NASA, “NASA,” 22 October 2014. [Online]. Available: https://spaceflightsystems.grc.nasa.gov/education/rocket/rktstab.html. [Accessed 09 March 2016].

[33] P. B. Jackson, “Overview of Missile Flight and Control Systems,” *John Hopkins APL Technical Digest*, pp. 9–24, 2010.

[34] U.S. Naval Test Pilot School, *Flight Test Manual 103: Fixed Wing Stability and Control*, Patuxent River, Maryland: Naval Air Warfare Center, Aircraft Division, 1997.

[35] A. Noureldin, T. B. Karamat and J. Georgy*, Fundamentals of Inertial Navigation, Satellite Based Positioning and Their Integration*, New York: Springer, 2013.

[36] E. Bone, “Unmanned Aerial Vehicles: Background and Issues for Congress,” 2003. [Online]. Available: http://fas.org/irp/crs/RL31872.pdf.

[37] T. V. Milligan, “Questions Answered About Spinning Rockets,” Apogee Components Inc., Colorado Springs, CO, 2009.

[38] V. Dobrokhodov, “Kinematics and dynamics of fixed wing UAVs,” in Handbook of Unmanned Aerial Vehicles, New York, Springer, 2014, pp. 243–277.

[39] W. L. Boon, DTS5705 Guided weapons (seekers). Lecture 3 - EO seekers., class notes, 2015.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California