NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DESIGN AND DEVELOPMENT OF AN ADAPTABLE FLIGHT DEMONSTRATION VEHICLE WITH MODULAR GUIDANCE AND CONTROL

by

Matthew Busta

June 2019

Thesis Advisor: Christopher M. Brophy
Co-Advisor: Vladimir N. Dobrokhodov

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
DESIGN AND DEVELOPMENT OF AN ADAPTABLE FLIGHT DEMONSTRATION VEHICLE WITH MODULAR GUIDANCE AND CONTROL

5. FUNDING NUMBERS

6. AUTHOR(S) Matthew Busta

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.

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

13. ABSTRACT (maximum 200 words)

The ability to evaluate new missile technologies and explore various flight profiles through experimental flight tests is an essential step in most technology development paths. This capability is often an expensive option and designs are frequently limited to a particular vehicle layout. An adaptable and recoverable two-stage missile flight system was designed with configurability in mind such that either ballistic flight paths or a boost-sustain configuration with a final horizontal flight profile can be evaluated. The vehicle developed included tunable guidance and control, servo actuation, structure, recovery, and avionics. The system was designed to utilize a canard control strategy for the boost phase and tail control for the sustainer phase regardless of which flight option was used. Flight testing was conducted to demonstrate structural concepts, inertial navigation sensor resolution, data collection, two stage events, and recovery

14. SUBJECT TERMS	**15. NUMBER OF PAGES**
missile design, missile control	93
	16. PRICE CODE

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

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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

DESIGN AND DEVELOPMENT OF AN ADAPTABLE FLIGHT DEMONSTRATION VEHICLE WITH MODULAR GUIDANCE AND CONTROL

Matthew Busta
Lieutenant, United States Navy
BS, Loras College, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2019

Approved by: Christopher M. Brophy
Advisor

Vladimir N. Dobrokhodov
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to evaluate new missile technologies and explore various flight profiles through experimental flight tests is an essential step in most technology development paths. This capability is often an expensive option and designs are frequently limited to a particular vehicle layout. An adaptable and recoverable two-stage missile flight system was designed with configurability in mind such that either ballistic flight paths or a boost-sustain configuration with a final horizontal flight profile can be evaluated. The vehicle developed included tunable guidance and control, servo actuation, structure, recovery, and avionics. The system was designed to utilize a canard control strategy for the boost phase and tail control for the sustainer phase regardless of which flight option was used. Flight testing was conducted to demonstrate structural concepts, inertial navigation sensor resolution, data collection, two stage events, and recovery

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- **A. OBJECTIVES....1**
- **B. PAST WORK....1**
- **C. FLIGHT SCENARIO....3**
- **D. GUIDANCE, NAVIGATION, AND CONTROL (GNC) REQUIREMENTS....6**
- **E. MODULAR CONTROL SYSTEM....7**

II. VEHICLE DESIGN....9
- **A. AERODYNAMICS....10**
  - **1. Static Stability....10**
  - **2. Canard v. Tail Control....11**
- **B. STRUCTURE....13**
  - **1. Past Coupler Testing....13**
  - **2. New Coupler Design....14**
  - **3. Electronics Bay Revisions....16**
- **C. MODULAR FIN ACTUATOR ASSEMBLY....18**
  - **1. Previous Fin Actuation Designs....18**
  - **2. Gearbox....20**
  - **3. Servos....21**
  - **4. Assembling the Pieces....22**
- **D. EVENT CONTROL....23**
- **E. RECOVERY....25**
  - **1. Streamer Side Deployment....25**
  - **2. Pressure Monitoring....26**
  - **3. Retrieval....27**
- **F. MOTOR SELECTION....28**
- **G. FUTURE WORK....28**

III. GNC....29
- **A. SENSOR COLLECTION....29**
  - **1. Hardware....29**
  - **2. Attitude Sensing....31**
  - **3. Data Filtering....32**
  - **4. Sampling Rate....37**
- **B. SYSTEM ROLL CONTROL....39**
- **C. FUTURE WORK....41**

IV. TESTING/RESULTS 43
A. CONTROL SYSTEM TESTING 43
1. Test Platform Development 43
2. Disturbance Response 45
3. Future Work 47
B. FLIGHT VEHICLE TESTING 47
1. November Two-Stage Test 47
2. February Data Recording Test 49
3. May Two-Stage Test 50

V. CONCLUSIONS 55

APPENDIX 57

LIST OF REFERENCES 73

INITIAL DISTRIBUTION LIST 75

LIST OF FIGURES

Figure 1. Vehicle Developed by Rydalch. Source: [1].....2

Figure 2. Vehicle Developed by Grohe. Source: [2].....3

Figure 3. Flight Sequence.....4

Figure 4. Flight Profile.....5

Figure 5. Major Sections and Associated Equipment.....9

Figure 6. Demonstration of Static Stability.....10

Figure 7. Canard Controlled Sustainer CG and CP.....12

Figure 8. Tail Controlled Sustainer CG and CP.....12

Figure 9. Custom Machined Aluminum Coupler. Source: [2].....14

Figure 10. Booser-Sustainer Coupler Layout.....15

Figure 11. Sustainer Layout.....17

Figure 12. Electronics Mounting Frame.....18

Figure 13. Fin Actuation System Developed by Rydalch. Adapted from [1].....19

Figure 14. Fin Actuation System Developed by Grohe. Adapted from [2].....19

Figure 15. The P20-5AR Worm Wheel Gearbox. Source: [5].....21

Figure 16. Control Assembly.....22

Figure 17. Raven 4 (Left) and StratoLoggerCF (Right). Source: [6], [7].....23

Figure 18. MissileWorks WRC+ System. Source: [8].....25

Figure 19. The MPRLS Ported Pressure Sensor Breakout. Source: [9].....26

Figure 20. Multitronix TelemertyPro Transmitter and Reciever. Source: [10].....27

Figure 21. The Microcontroller Used, the PyBoard V 1.1. Source: [11].....30

Figure 22. AHRS Used, the myAHRS+. Source: [12].....31

Figure 23. Noise and Bias in Stationary Gyroscope Readings.....33

Figure 24. EMA Filters Applied to the Bias-Corrected Stationary Data 34
Figure 25. EMA Filters Applied to February Flight Data.....35
Figure 26. Zoomed EMA Filters Applied to February Flight Data.....35
Figure 27. EMA Filters Applied to May Flight Data.....36
Figure 28. Zoomed EMA Filters Applied to May Flight Data.....36
Figure 29. Second Order System Step Responses. Source: [16].....40
Figure 30. Roll Test Platform.....44
Figure 31. Fin Control Assembly.....45
Figure 32. Test Platform Response with Proportional-Only Control.....46
Figure 33. Test Platform Response with Proportional and Velocity Control 46
Figure 34. Boken Booser-Sustainer Coupler.....48
Figure 35. Summary of Data from February Launch.....50
Figure 36. Vehicle Launched in the May Test 52
Figure 37. Broken Booster-Sustainer Coupler 53
Figure 38. Summary of Data From May Launch 54
Figure 39. Boot.py.....60
Figure 40. Main.py 61
Figure 41. Myahrs.py 64
Figure 42. Mprlslib.py.....70
Figure 43. Readdata.py.....71

LIST OF TABLES

Table 1. Gearbox Designs and Their Disadvantages 20

Table 2. Key Parameters of the P20-5AR Gearbox. Adapted from [5]. 21

Table 3. Sequence of Vehicle Events.....24

Table 4. Advantages of Microcontrollers and Single-Board Computers.....29

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AGL	Above Ground Level
AHRS	Attitude and Heading Reference System
CG	Center of Gravity
cm	centimeter
CO_2	Carbon Dioxide
COTS	Commercial-off-the-Shelf
CP	Center of Pressure
deg	degrees
EMA	Exponential Moving Average
G	Gravity
GNC	Guidance, Navigation, and Control
Hz	Hertz
I2C	Inter-Integrated Circuit
I_{xx}	Area Moment of Inertia
K	Kelvin
kPa	kilopascals
L	Length
LiPO	Lithium Polymer
m	mass
M	meter
ms	millisecond
NPS	Naval Postgraduate School
PLA	Polylacetic Acid
q	dynamic pressure
sec	seconds
SD	Secure Digital
UART	Universal Asynchronous Receiver-Transmit
UAV	Unmanned Ariel Vehicle
USB	Universal Serial Bus
V	Volt

α	EMA Weighting Factor
ζ	damping ratio
ω_n	natural frequency

ACKNOWLEDGMENTS

Thank you to my advisors, Dr. Chris Brophy and Dr. Vladimir Dobrokhodov, for their significant time and guidance on the project. Also to the NPS Rocket Lab staff of Dr. Josh Codoni, Robert Wright, Lee van Houtte, and Dave Dausen for assisting in the design, build, and launch of numerous vehicles. Thank you to Dr. Kevin Jones and the Advanced Robotic Systems Engineering Laboratory for their assistance with the test platform. Finally, thank you to Kelly and Annie for tolerating my frequent trips to the desert and my time spent at the lab and in the garage.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The ability to adapt and modify a flight system for experimental testing of new subsystems or payload delivery scenarios is important for any flight development program. Two-stage vehicles are common in high-powered amateur rocketry, but the unique flight profile, vehicle requirements, and control loads of a test platform require the use of novel construction techniques and custom fabrication of components capable of handling the high stresses placed on the system. Additionally, passive stabilization is not sufficient to achieve the wide variety of flight profiles necessary for a test platform.

A. OBJECTIVES

The primary goal of this thesis was to develop a two-stage vehicle and its guidance and control systems that are capable of accelerating a second stage to Mach 2.2, where the second stage will be designed as a test platform for new technologies. A second goal was to develop a ground based attitude control test platform to allow for control hardware and software experimentation with a physical plant. A particular focus was on incorporating a modular control system based on commercial-off-the-shelf (COTS) components that could be easily adapted to a wide variety of vehicle configurations that the test platform may take for various missions, ranging from a missile interceptor to payload delivery.

B. PAST WORK

Single-stage rocket-powered flight vehicles have been developed at Naval Postgraduate School (NPS) with limited flight control based on the Arduino platform for the ME4704 Missile Design Course. These systems have demonstrated the ability of COTS components, supplemented with innovative building techniques and select custom items, to support flight profiles beyond those normally used in high-powered rocketry. Additionally, it has been shown that COTS components can detect and cancel vehicle roll and perform pre-programmed pitch maneuvers but no attempt has been made to introduce guidance. Previous work has been geared toward a prototype vehicle for countering unmanned aerial vehicle (UAV) swarms [1], [2], which has significantly different design constraints from a two stage propulsion test platform. The test platform will have to go

faster while being capable of setting up a horizontal second stage flight profile and handle the staging to transition from boost to sustain phase.

Figure 1 shows the final vehicle developed by Ensign Fletcher Rydalch [1]. This was the first attempt by the NPS Rocket Lab at active fin control. Two opposing tail fins were each connected to a servo through linkages to demonstrate the feasibility of fin actuation with COTS components. While in-flight fin actuation was demonstrated, looseness in linkage joints resulted in play in the fins that inhibited precise control.

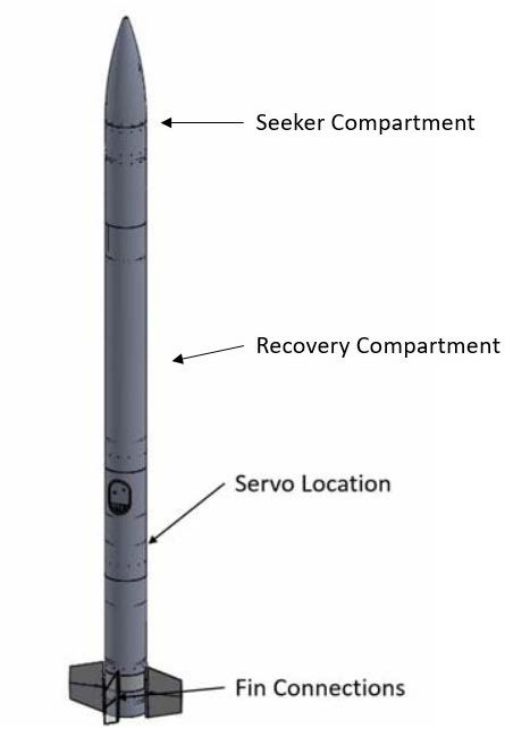


Figure 1. Vehicle Developed by Rydalch. Source: [1].

Figure 2 shows the vehicle developed by Captain Kai Grohe [2]. This vehicle sought to remove the play in the fins by replacing the linkages with direct gear to gear interface. Due to space constraints at the vehicle tail the fin actuating assembly was moved to the front of the vehicle, providing a canard controlled vehicle. Additionally, the use of aluminum coupler interfaces was tested to replace the phenolic tube couplers to reduce the flex in the vehicle at section joints.

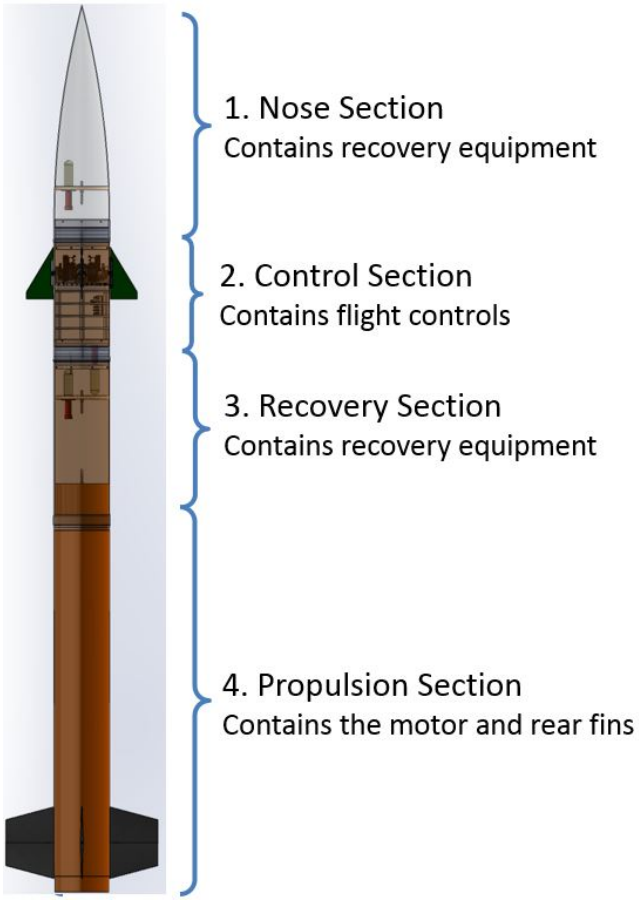


Figure 2. Vehicle Developed by Grohe. Source: [2].

C. FLIGHT SCENARIO

The major vehicle milestones for the flight test scenario are shown in Figure 3 with its profile in Figure 4. The vehicle is launched at a 45-degree (deg) angle powered by a solid rocket motor to a typical propulsion handoff speed of Mach 2.2 at 1,000 m. When these conditions are met the booster, shown in orange, separates, and is returned to the ground via parachute. Meanwhile the sustainer, shown in white, is ignited and continues its flight horizontally until the motor burns out where the sustainer pitches up into a climb to bleed speed before the recovery system is initiated, allowing it to be safely recovered.

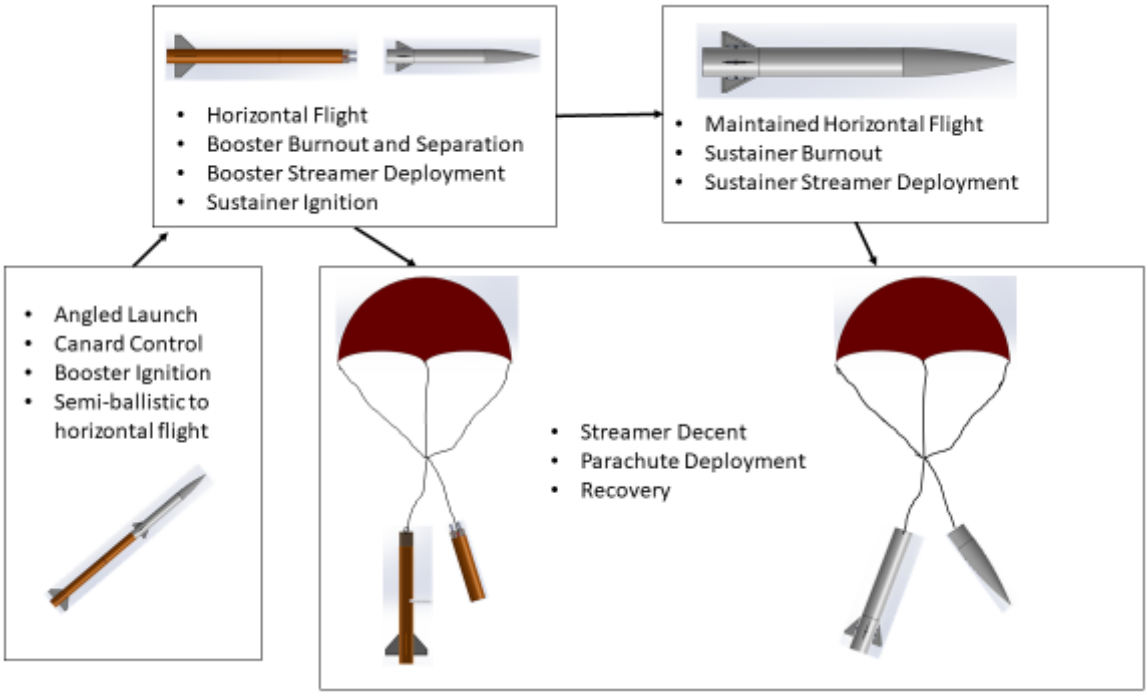
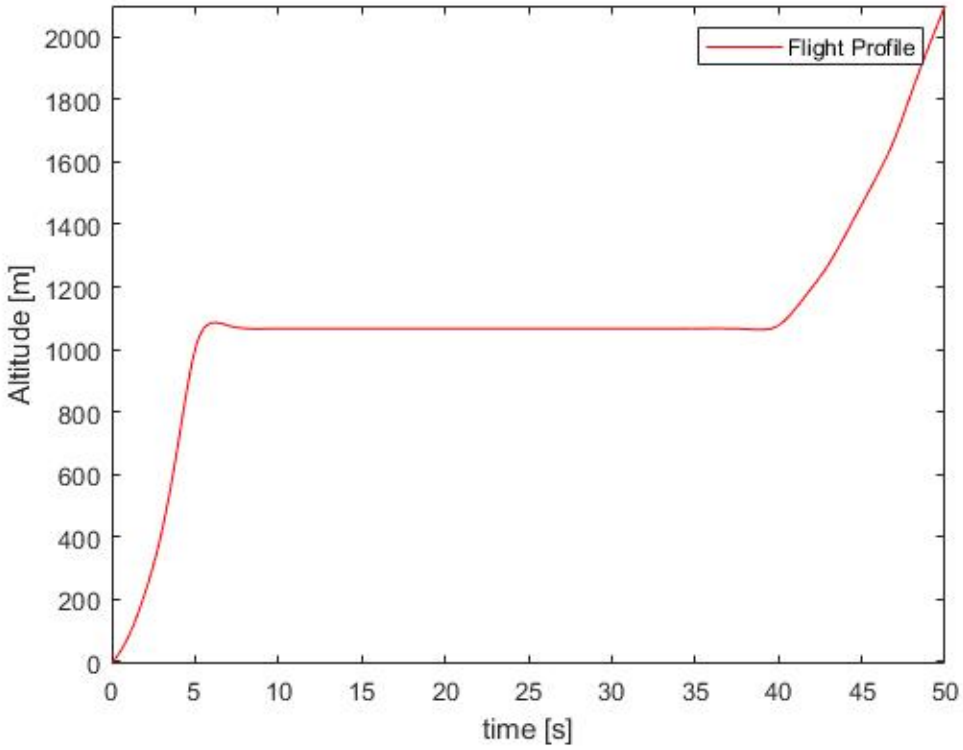


Figure 3. Flight Sequence

Based on this flight profile the expected sustained dynamic pressure at 1,000 m and Mach 2 is 251 kilopascals (kPa). This exceeds the Joint Army, Navy, National Aeronautic and Space Administration, and Air Force guidance for tactical vehicles of 72 kPa based on structural limitations of most vehicles [3]. Since this limit is being exceeded more than threefold significant attention needs to be placed on vehicle structure to ensure it is strong enough to hold up under the high forces to which it will be subjected. Additionally, with a total temperature for these conditions of 507 K, the temperature limits of exposed surfaces must be considered in material selection and protection.

Vehicle control is achieved through fins actuated by servos at the base of the sustainer, acting as canard control during the boost phase and tail control after sustainer separation. This control is necessary to assist the semi-ballistic profile during the boost phase to horizontal flight and maintain horizontal flight after sustainer ignition.



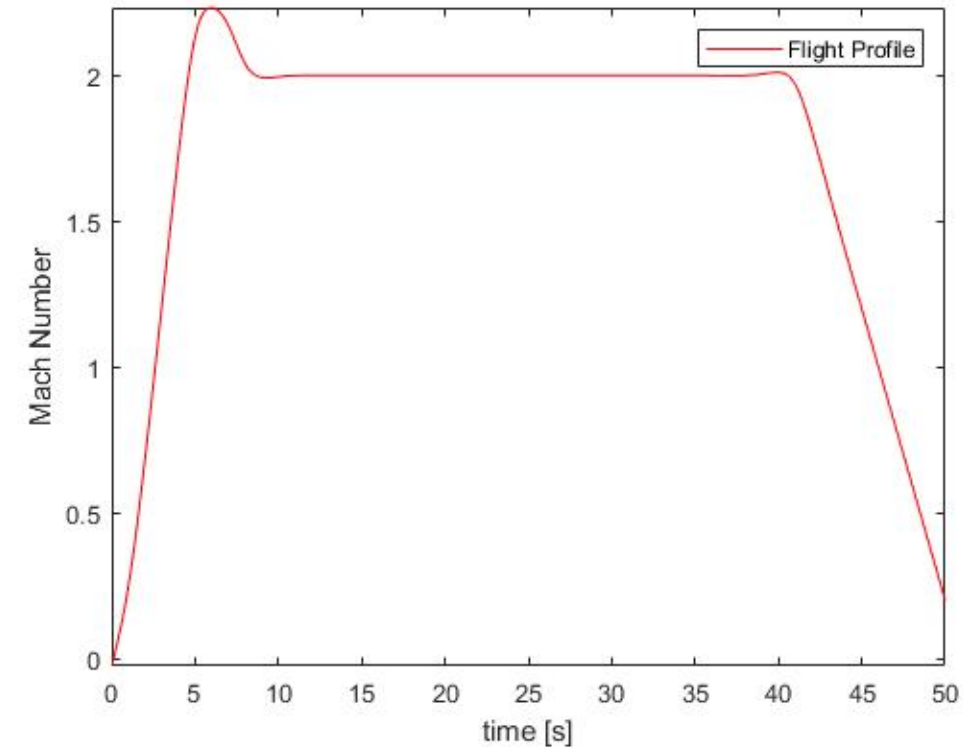


Figure 4. Flight Profile

This flight scenario differs significantly from typical amateur rocketry flights, which are launched and maintain a vertical or near vertical flight path. Active control systems are not used in amateur rocketry as they are generally unnecessary and would add weight, complexity, and expense. As a result, there are no COTS systems available for active rocket guidance or control. However, this project could also be used in other flight scenarios with a vehicle designed to reach extreme altitudes for amateur rocketry where passive stabilization becomes inadequate.

D. GUIDANCE, NAVIGATION, AND CONTROL (GNC) REQUIREMENTS

For this flight scenario the GNC system will have to:

1. Sense and estimate vehicle state
2. Reach and maintain a constant altitude control (Pitch)
3. Guide the vehicle on the desired flight bearing (Heading)
4. Maintain vehicle roll orientation
5. Conduct staging events and adapt control laws to the new vehicle configuration

Vehicle state is measured by an attitude and heading reference system (AHRS), which provides vehicle orientation determined using three-axis accelerometers, gyroscopes, and magnetometers. This output is then filtered where necessary and used to control fins through servo actuation. The entire process is managed by a microcontroller that collects and interprets readings, calculates required response, and directs servo actuation.

Controlling pitch and yaw to attain the desired altitude and flight direction is important for both controlling flight conditions as well as allowing the vehicle to be safely flown in an extended horizontal profile. Maintaining vehicle roll orientation can contribute to flight conditions for the test but is also, at least partially, a self-imposed requirement as it significantly simplifies pitch and yaw control. By using four fins in a plus configuration

and maintaining a specified roll orientation one set of fins can be used for pitch control and the other set for yaw control, eliminating the requirement for complicated signal mixing.

E. MODULAR CONTROL SYSTEM

Since the GNC architecture for this test platform is being developed before the test flight vehicle has been designed, the system must be readily adaptable by means of alternate servos, reduction gears, avionics, etc. This modularity will further enhance the utility of the system as it will not be limited to only the current vehicle iteration, but could provide the foundation for the GNC of many different flight vehicles.

THIS PAGE INTENTIONALLY LEFT BLANK

II. VEHICLE DESIGN

To minimize construction cost and accelerate the build cycle high-power amateur rocketry components were used where possible. These components are readily available and relatively cheap but were supplemented with custom fabricated components where required for strength or function. Overall vehicle layout is shown in Figure 5, with sections labeled as they will be referred to herein, along with their major components.

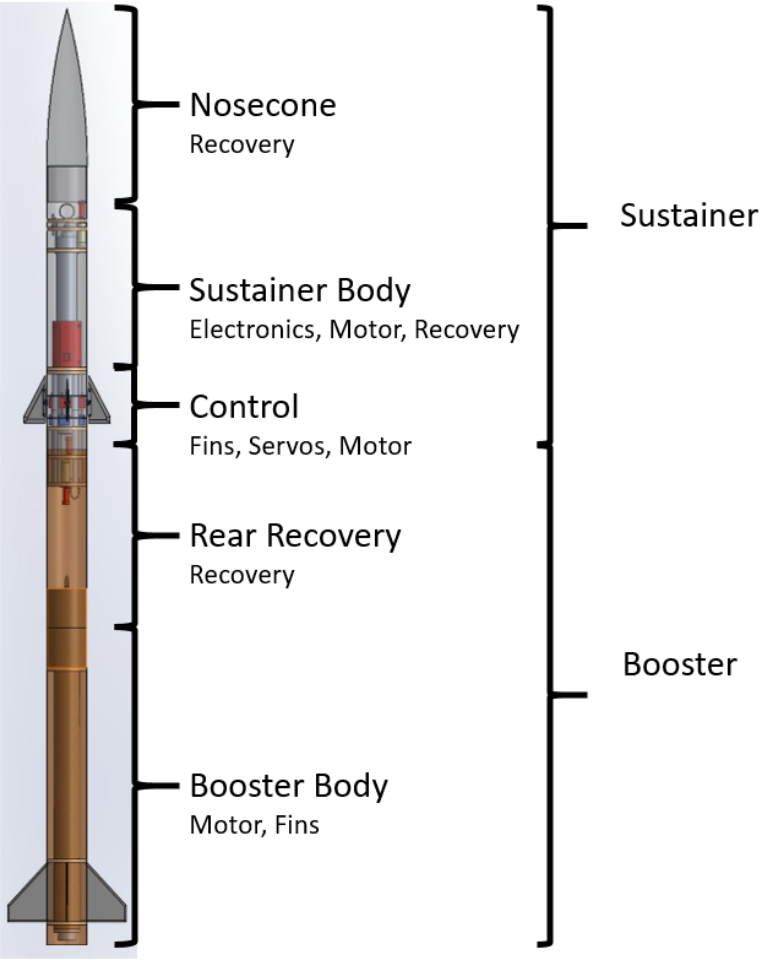


Figure 5. Major Sections and Associated Equipment

A. AERODYNAMICS

Aerodynamic analysis was conducted primarily with RockSim [4], a modeling software for high-powered rocketry.

1. Static Stability

The first priority for vehicle design was to ensure the vehicle had a positive static stability margin. A positive static stability margin ensures that as the vehicle diverges from a neutral angle of attack the aerodynamic force created provides a restoring moment to the vehicle, aiding in stable flight. This is done by ensuring that the point where the sum of the aerodynamic pressures around the body, the center of pressure (CP), is behind the point about which the body rotates, its center of gravity (CG). If this can be assured for all flight conditions the vehicle is statically stable. This condition, as well as the unstable condition, is shown in Figure 6. The CP is determined by the air pressure distribution generated by the airframe body and fins. In general, tail fins are sized and located on a rocket to shift the CP behind the CG for all flight conditions, ensuring stability.

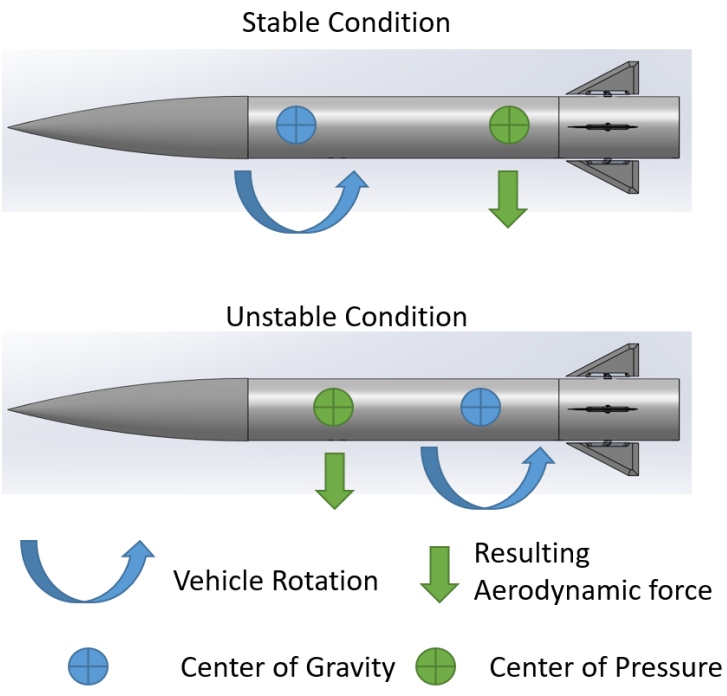


Figure 6. Demonstration of Static Stability

Drawbacks of a high static stability are a higher susceptibility to wind cocking, where the vehicle tilts into the wind, as well as the possibility for lower maneuverability. While not inherently required for actively controlled systems, a positive stability margin provides robust safety during development and is expected to provide adequate maneuverability for the test platform.

2. Canard v. Tail Control

Initial vehicle design was examined using canard control for both the booster and sustainer throughout the entirety of the flight, as the canard system developed by Grohe has demonstrated the ability to control the vehicle through precise angular control of the fins [2]. However, this vehicle layout was rejected due to the difficulty in creating a vehicle that had both an adequate stability margin for both sections and sufficient control authority.

The difficulty arose in sizing the booster tail fins, sustainer tail fins, and canard fins appropriately to maintain a positive static stability margin while also providing an adequate lever arm for the control fins to generate a moment about the CG. As shown in Figure 7 the CG is very close to the canards for the sustainer in this layout. This close proximity severely limits the moment that can be generated to control vehicle attitude. If larger canards are used to create a larger control force the CP is shifted forward and the margin to static stability is reduced. With the configuration shown already at a marginal static stability this is undesirable. Modified configurations of a canard controlled sustainer were evaluated but the heavy weight of the control assembly relative to the rest of the vehicle always kept the CG close to the canards.

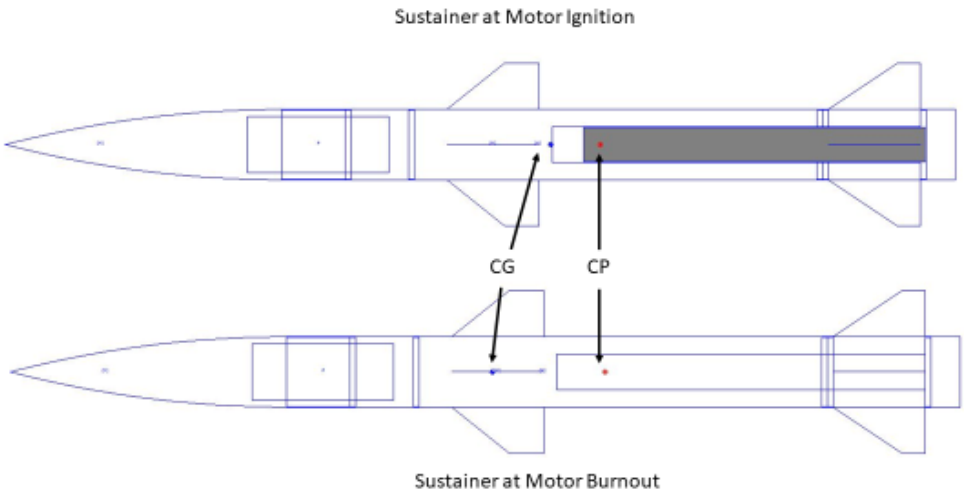


Figure 7. Canard Controlled Sustainer CG and CP

To avoid this issue tail control was examined and Figure 8 shows the improved GG, CP locations, and control lever arm relationships for the sustainer with tail control. With this layout both control authority and static stability for the sustainer can be increased through tail fins with larger area. This layout has a lower lever arm for control during the boost phase vs the configuration shown in Figure 7, but is expected to be adequate as the boost phase is designed to have lower control requirements through its semi-ballistic profile.

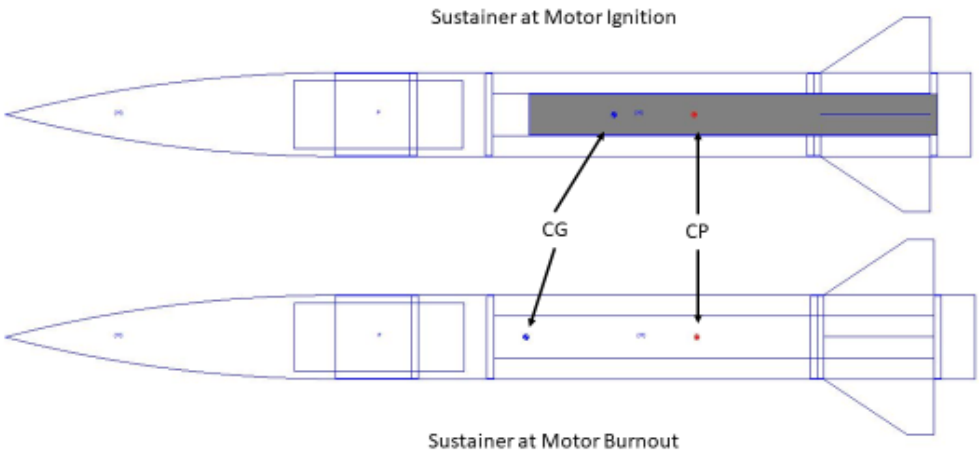


Figure 8. Tail Controlled Sustainer CG and CP

B. STRUCTURE

Most of the components used for building the test vehicles are COTS components for high-powered rocketry including structural components (phenolic tube and fiberglass wrapped phenolic tube airframes, wooden bulkheads and centering rings, plastic nosecones), staging equipment (StratoLoggerCF and Raven 4) and solid rocket motors. However, these components are designed for vertical, passively stabilized flight with lightweight payloads. To meet the additional stresses resulting from vehicle maneuvers and additional weight, novel construction techniques had to be used in certain areas to ensure structural integrity.

1. Past Coupler Testing

One of the areas where amateur rocketry construction techniques are most inadequate for this vehicle is at the interfaces between sections. A rigid connection between sections is desired to prevent vehicle flex that would introduce unwanted aerodynamic forces and structural stress. Conventional hobby construction techniques use concentric phenolic tubes where one slides over another. The general rule of thumb for this configuration is to have the length of engagement be equal to the diameter of the airframe. Shorter coupler lengths introduce more body flex, increasing body stress and aerodynamic forces while excessively long couplers add unnecessary length and weight. These techniques are adequate for the vertical flights without active control, but required modification for this project.

Each of the section interfaces of the rocket have different requirements and multiple techniques have been used to increase the rigidity of the connection and/or shorten the length of engagement required. Early steps taken by Rydalch [1] and Grohe [2] to increase the strength of the coupler were to use fiberglass-wrapped phenolic tubing for the outer tube and to double up the inner tube by epoxying in a third tube on the inner face of the male coupler to increase its thickness. An additional benefit of a fiberglass airframe is that it is less susceptible to damage during landing, making for a more reusable vehicle.

While these improvements created a stronger coupler there was also a desire to shorten the engagement length required. To this end Grohe tested tight-tolerance custom

built aluminum couplers, as shown in Figure 9. Comparison of the aluminum couplers showed slightly worse rigidity than standard length phenolic couplers, but were much better than phenolic couplers of equivalent length. However, Grohe ultimately recommended against the use of aluminum couplers because the coupler fit varied with temperature and tolerances expanded with wear, all degrading the rigidity [2].

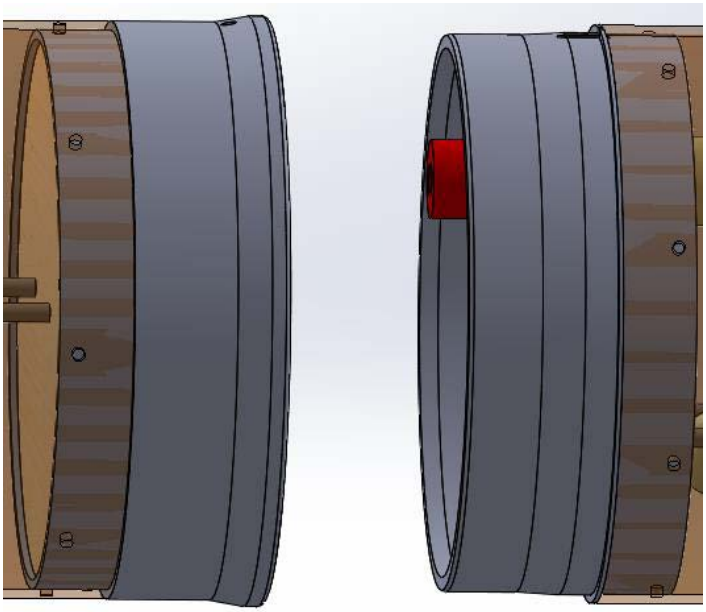


Figure 9. Custom Machined Aluminum Coupler. Source: [2].

2. New Coupler Design

One of the largest complications in going from single stage to a two-stage vehicle is coupling the sustainer to the booster. By the thumb rule the 18.3 centimeters (cm) body tube would require 18.3 cm of coupler engagement, which would leave the motor nozzle submerged approximately ten cm inside of the body tube, where typical high-powered rocketry construction has the nozzle submerged around 1.5 cm. The deeper submergence would have a detrimental effect on motor performance which could interfere with flight test results as well as subjecting a portion of the body to the hot exhaust gasses reducing vehicle reusability. Aluminum couplers could not be used because of the proximity to

exhaust gasses that would foul and distort the ring, requiring a new ring for each launch. This is not practical in a development vehicle and so alternatives were sought.

To allow for a more typical nozzle submergence of 1.5 cm while still maintaining a rigid joint the structure shown in Figure 10 was designed. It utilizes 7.62 cm of conventional coupler engagement and supplements it with four concentric carbon fiber tubes that increase the effective engagement length. The carbon fiber tubes were chosen due to their high flexural rigidity and light weight. Additionally, the tight tolerance of these tubes ensure a minimum amount of flex in the joint while the smooth finish ensures reliable separation.

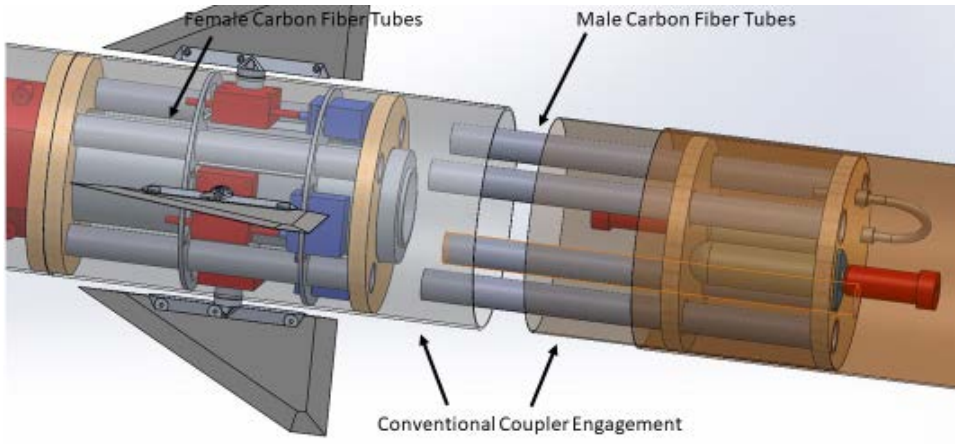


Figure 10. Booser-Sustainer Coupler Layout

The layout in Figure 10 is the second iteration of this type of coupler. The first version built and flown utilized two sets of carbon fiber tubes and after a test flight failure, discussed in IV.B.1, four sets of carbon fiber tubes were used for the final design. In addition to increasing the number of carbon fiber tubes two other significant changes were made for the second iteration. First, the length of engagement for both the carbon fiber and phenolic tubes was made the same to reduce the likelihood of partial disengagement. Second, the construction sequence was altered to ensure full contact between the female phenolic coupler tube and the sustainer aft centering ring as well as between the outer body

tubes of the two sections with the goal to ensure the maximum area to transfer the load through the vehicle and maximizing flexural strength.

3. Electronics Bay Revisions

The designs of typical amateur rockets as well as those of Grohe and Rydalch left the area around the motor mount empty with an entire section of the rocket dedicated to electronics, as shown in Figure 2, with the exception of the servo linkages for Rydalch's design. This separate section adds length and weight to the vehicle while leaving a large amount of space unused. To reduce the overall length and weight of the vehicle, this design moved the electronics to the previously unused area around the motor mount. This required modifications to previous design and build processes, as external access to the servos and electronics is required.

To achieve external access a two section sustainer body was designed around the motor mount. The lower section contains the control assembly while the upper section houses the remaining control electronics, as shown in Figure 11. The upper body tube is secured during flight to the centering rings by screws. These screws can be removed allowing the upper body tube to be slid off the front end of the motor mount, providing access to the electronics around the motor mount.

A drawback of this design is the more complicated assembly and the two section outer body tube which weakens the typical load path through the vehicle. Thrust is transmitted from the motor mount through centering rings to the outer body, which supports the vehicle. In dividing the outer body into two halves this load path is interrupted. To increase the strength of the vehicle the typical phenolic tube motor mount was replaced with a fiberglass wrapped phenolic motor mount, providing a stronger secondary load path through the vehicle. This minimizes the load under acceleration on the screws and joints of the outer body.

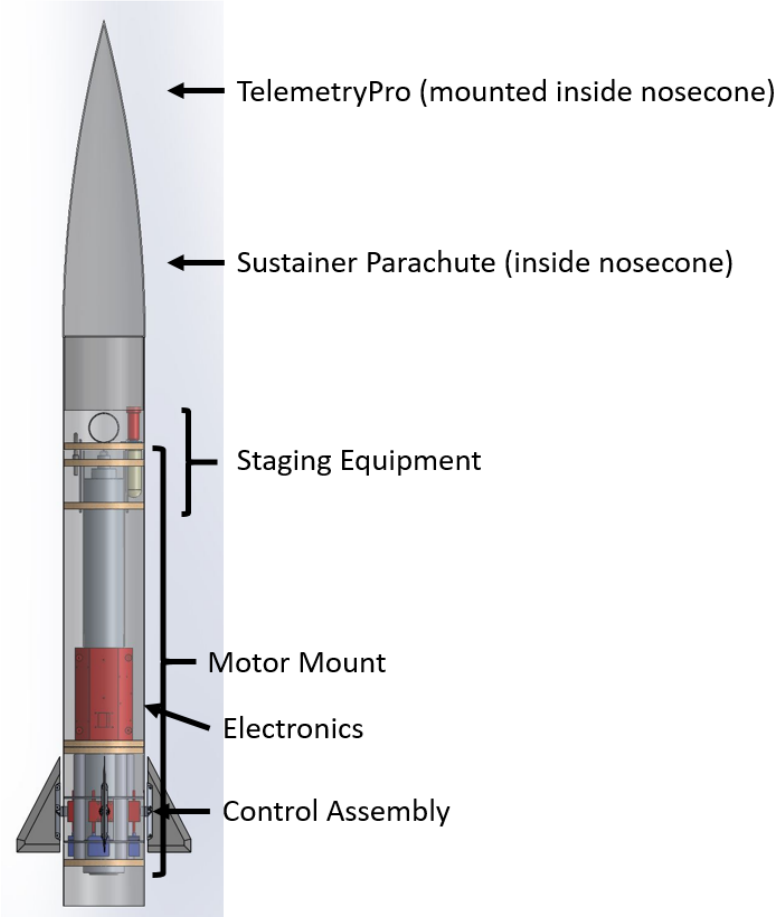


Figure 11. Sustainer Layout

The electronic hardware mounting frame is shown in Figure 12. It is 3D printed using tough polylactic acid (PLA) in two pieces and then is screwed together around the motor mount. Components are held on by machine screws into tapped holes in the tough PLA. The custom components are powered from an 850 milliamp-hour Lithium Polymer (LiPO) battery while the COTS recovery equipment is powered using 9 volt (V) batteries due to their design power limits.

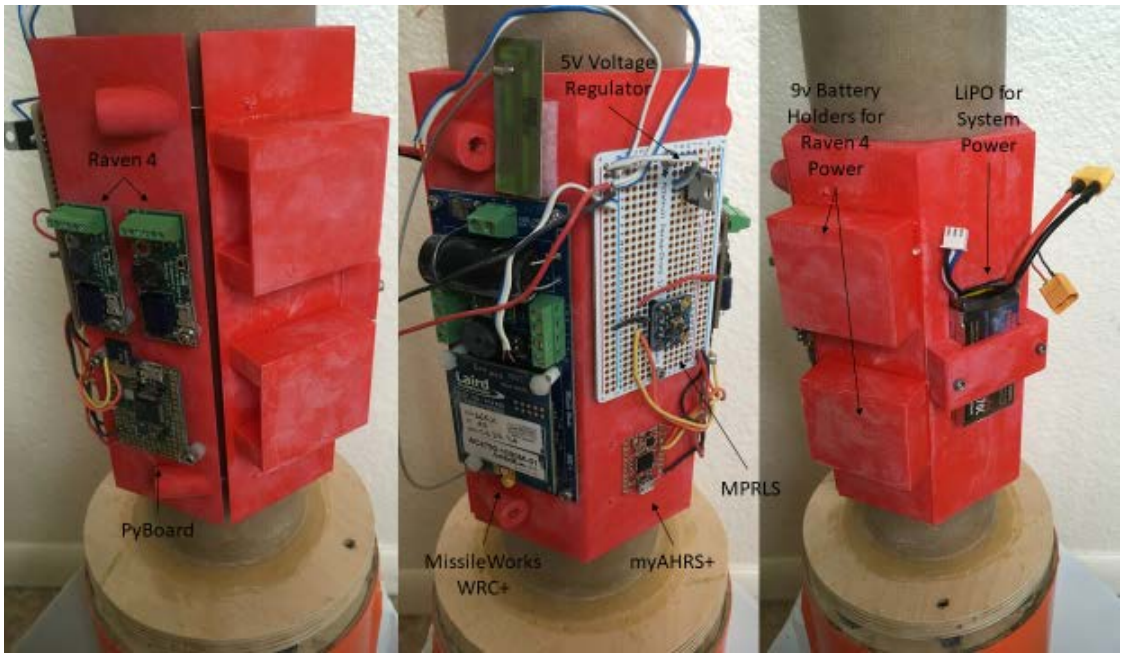


Figure 12. Electronics Mounting Frame

C. MODULAR FIN ACTUATOR ASSEMBLY

Development of a new fin actuating assembly sought to take the canard assembly developed by Grohe to a form that would allow for tail control.

1. Previous Fin Actuation Designs

The tail control system previously developed by Rydalch had the servos connected to the fins by push rod linkages that ran inside the missile body but outside the motor tube, as shown in Figure 13. The system demonstrated the ability for COTS components in a custom configuration to execute preprogrammed fin deflections capable of controlling roll and pitch maneuvers. However, the multiple connections in the push rod system resulted in a fin play that limited the ability to precisely control fin angle [1], [2].

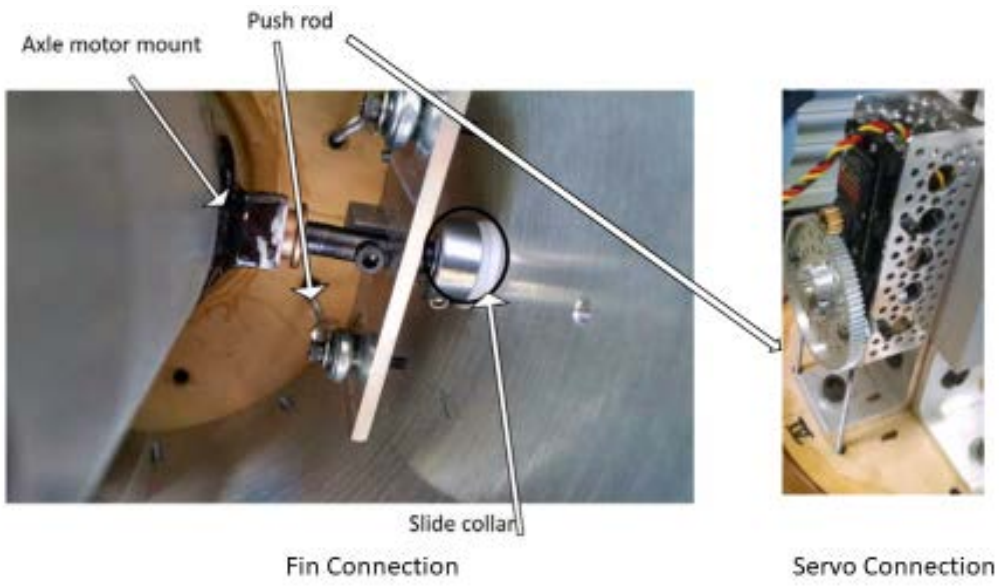


Figure 13. Fin Actuation System Developed by Rydalch.
Adapted from [1].

Grohe adjusted this system and demonstrated precise fin actuation with canard fin control by connecting the fins and servos directly via spur gears. His assembly used is shown in Figure 14. but the overall system was heavy and required its own separate section, not taking advantage of the available space along the vehicle centerline as well as the canard controlled sustainer layout not preferred based on aerodynamic stability analysis.

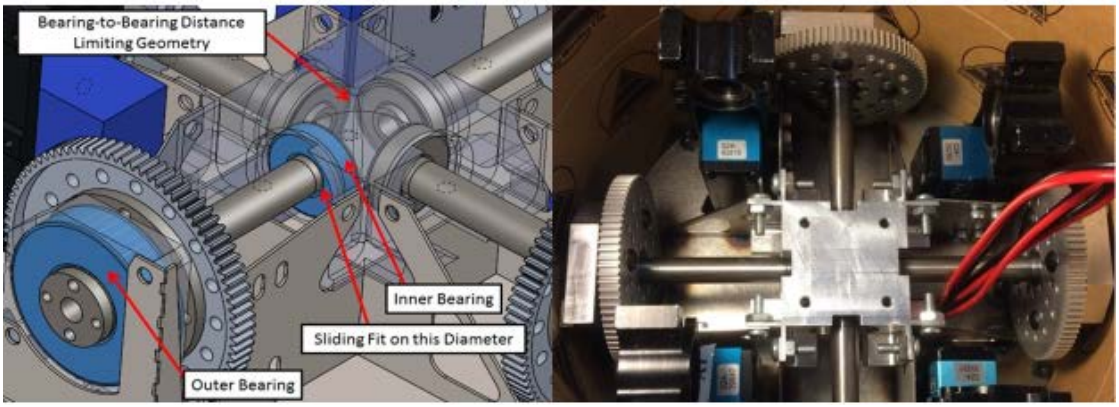


Figure 14. Fin Actuation System Developed by Grohe.
Adapted from [2].

A system using tight tolerance gear connections that can fit around the motor tube was desired that would provide the benefits of both of these systems and make for a more adaptable vehicle by allowing for a lighter, shorter base vehicle.

2. Gearbox

As the servo and fin would not be mounted on the same shaft, a mechanism for transmission of the rotational motion of the servo to the fin was necessary. The greatest challenge of the gearbox design was to fit the assembly into the narrow space around the motor mount inside the outer tube, with about 3.8 cm of space radially. Initial designs of a few configurations were conducted, with their major disadvantages in Table 1. The best option was determined to be a worm gear configuration in a very low reduction ratio. This configuration has the servo mounted with its shaft axis perpendicular to the fin shaft, allowing the use of the servos used by Grohe. Additionally, this layout maximized available torque, had direct gear to gear connections, and could still be operated at satisfactory speeds with a low reduction ratio.

Table 1. Gearbox Designs and Their Disadvantages

Design	**Disadvantages**
Spur Gears on Two Parallel Shafts	Servos that could be mounted in this manner have to be much smaller, with correspondingly lower torque ratings
Common Shaft	Same as parallel shafts, also places more stress on the servo shaft
Linkages	Previously used system had considerable play in fin motion, servos would have to be mounted above motor tube
Worm Wheel	High reduction ratios limit speeds

Both COTS and custom built worm wheel gearboxes were evaluated and ultimately a commercial gearbox, the P20-5AR shown in Figure 15, was selected due to the difficulty and time required to build a precision gearbox. Furthermore, it was determined that the

cost savings would be minimal versus a commercial system. The primary features that led to its selection are shown in Table 2.

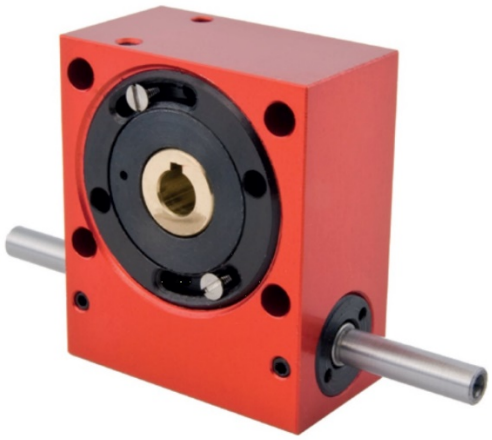


Figure 15. The P20-5AR Worm Wheel Gearbox. Source: [5].

Table 2. Key Parameters of the P20-5AR Gearbox. Adapted from [5].

Parameter	**Specification**
Reduction Ratio	5:1
Dimensions (cm)	4.7x2.2x4.0
Backlash (deg)	≤ 0.07
Mass (kilograms)	0.18

3. Servos

The servo used, a Futaba BLS-172SV, was previously used in Grohe's vehicles. Their original selection was based on their high torque, 3.63 Newton meters, and speed, 546 deg/sec, in a relatively compact size. However, their size was not compact enough to allow for mounting on the same axle as the fin, so a review of currently available servos was conducted. None were found that were compact enough to be mounted on a common shaft with the fin while also having adequate torque and speed. Furthermore, no servos of

similar dimensions were found that offered a significant advantage over the Futaba servo, so it was determined to use the current servos and utilize the worm wheel gearbox.

4. Assembling the Pieces

With the major components selected they were put together into a functional assembly shown in Figure 16. The gearbox shaft is provided by the gearbox manufacturer and then adapted to connect to fin holders that are custom built. The base plates and vertical support would be custom machined.

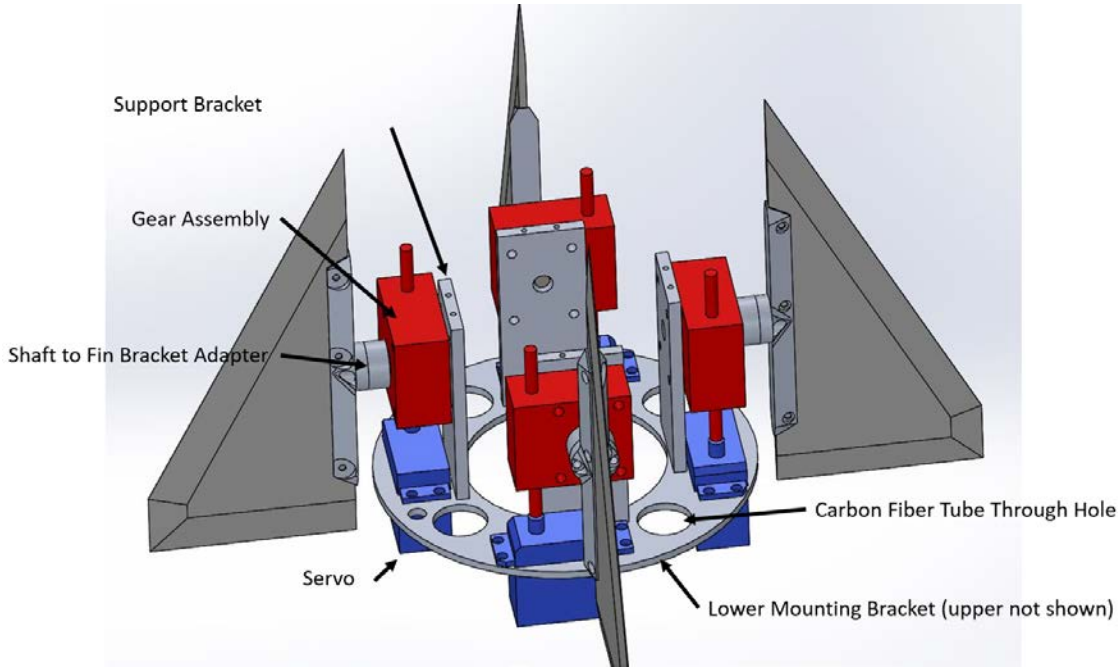


Figure 16. Control Assembly

Because of this system's large center through-hole it will be able to be rapidly implemented in a variety of vehicle configurations while maintaining the centerline available for propulsion systems. Additionally, different servos can be swapped out and the gearbox is available in a range of reduction values from 5:1 to 120:1, allowing for further customization.

D. EVENT CONTROL

In-flight staging events were controlled by either Featherweight Altimeters' Raven 4 or PerfectFlite's StratoLoggerCF, both shown in Figure 17. These are COTS components for high-powered amateur rocketry and were used with no modifications. Two Raven 4s are collocated with the sustainer electronics and two StratoLoggerCFs are in the upper end of the rear recovery section. All events are initiated by an electronic match (e-match) that sparks an ignitor for motor ignition or black powder for all other events. Table 3 shows the events that occur through the flight and the event logic.

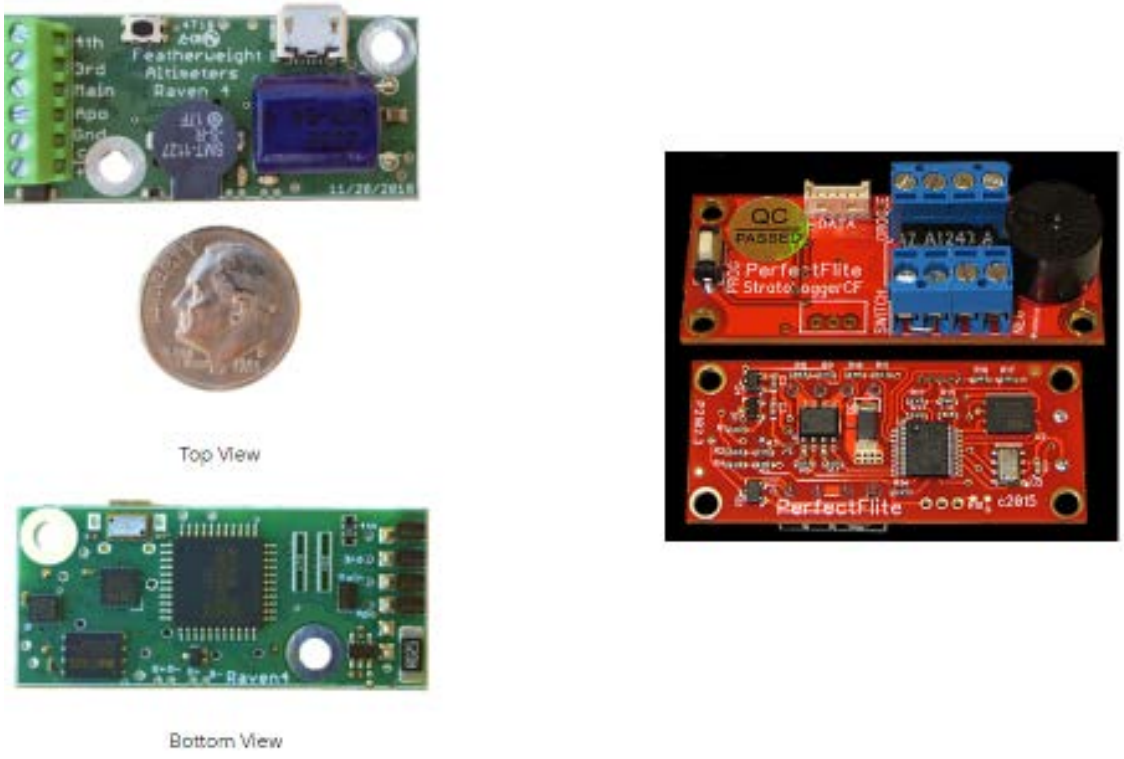


Figure 17. Raven 4 (Left) and StratoLoggerCF (Right).
Source: [6], [7].

Table 3. Sequence of Vehicle Events

Event	**Initiation Logic**
Booster ignition	Operator through control box
Booster/Sustainer Separation and Booster Streamer Deployment	Raven 4 detecting motor burnout
Sustainer Ignition	Raven 4 2 second delay after detecting motor burnout AND altitude greater than 2133 m above ground level (AGL)
Sustainer Streamer Deployment	Raven 4 detecting pressure increasing and velocity less than 213 m/s
Booster Parachute Deployment	StratoLoggerCF detecting an altitude less than 244 m AGL
Sustainer Parachute Deployment	Raven 4 detecting an altitude less than 305 Meters AGL MissileWorks WRC+ operator backup

To maximize system reliability there are two of each device (Raven 4 or StratoLoggerCF) with e-matches connected in parallel to each ignitor or charge. Finally, sustainer parachute deployment was backed up with a third e-match controlled by a MissileWorks WRC+ remote control system, shown in Figure 18, that allows for manual initiation to maximize likelihood of recovery of the most critical components.

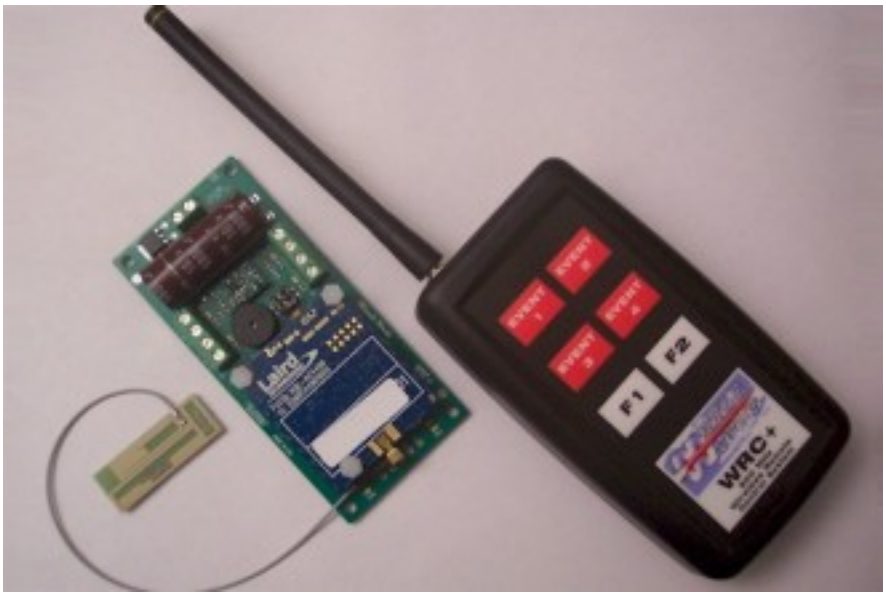


Figure 18. MissileWorks WRC+ System. Source: [8].

E. RECOVERY

The conventional recovery sequence for amateur rockets has been used in previous vehicles. In this model a small parachute or streamer is deployed at apogee by vehicle separation followed by a main parachute deployed through the separation of another section of the vehicle to slow the vehicle further for a safe landing. The streamer at apogee is used to allow for rapid but controlled descent from high altitudes until the main parachute is deployed. This strategy minimizes horizontal drift during the descent while still allowing for safe vehicle recovery. A drawback to this layout is the two sets of couplers required to permit the deployment of a streamer followed later by a parachute. This adds length, and therefore weight, while also reducing rigidity through the additional joints.

1. Streamer Side Deployment

To minimize the number of couplers required, a new streamer deployment concept was developed to push a streamer out of the side of the sustainer. This side deployment concept removes a set of couplers from the upper stage and also simplifies ejection. To separate the couplers of the rocket a COTS carbon dioxide (CO_2) deployment system is used where the e-match ignites a black powder charge that drives a spiked piston to

puncture a CO_2 canister that releases the CO_2 into the cavity to raise the pressure and separate the associated coupler. Since the side deployment system requires a much smaller volume only the black powder charge is used.

2. Pressure Monitoring

After the failure of the November Two Stage vehicle, discussed in IV.B.1, it was desired to better understand the coupler cavity pressure throughout vehicle flight and separation. To prevent couplers from separating and enclosed sections from mechanical failure on ascent vent holes are drilled into each cavity and shear pins are used to hold couplers together. The vent holes allow pressures to equalize as the external pressure lowers with altitude, but a drawback is that they allow CO_2 to escape before coupler separation. Ground tests had been conducted with pervious work but in-flight data was also desired. To this end pressure monitoring was incorporated into the GNC package using the Adafruit MPRLS Ported Pressure Sensor Breakout, shown in Figure 19. It was selected for its 0–25 PSI range and its pressure port which allow for placement of the pressure sensor with the GNC package while still monitoring coupler cavity pressure remotely through a tube.

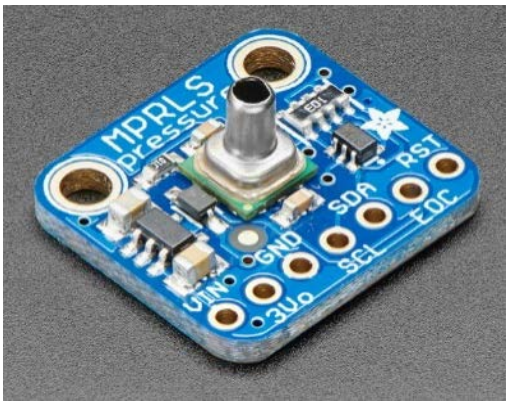


Figure 19. The MPRLS Ported Pressure Sensor Breakout. Source: [9].

3. Retrieval

After the vehicle has landed it must be located for retrieval. Previous vehicles had typically stayed below 3050 meters, and remained within a few kilometers of the launch site. Because this vehicle had a projected altitude of 7,620 meters a tracking system capable of working over longer ranges was desired. After a search of COTS systems, the Multitronix TelemetryPro Tracking System was selected. The transmitter and receiver are shown in Figure 20.

In flight, this system reports back vehicle status in real time and is capable of identifying events such as motor staging and parachute deployment. After landing the receiver displays the GPS coordinates of the vehicle while also denoting bearing and range. After vehicle recovery, further flight data can be downloaded and plotted for event reconstruction.

Figure 20. Multitronix TelemertyPro Transmitter and Reciever.
Source: [10].

F. MOTOR SELECTION

The booster motor, a Cesaroni PRO 98 N3301 solid propellant, was selected based on its high thrust, total impulse, and availability. RockSim simulations predicted that it would propel the sustainer to Mach 1.5. A larger model of motors is commercially available that is simulated to take the vehicle to Mach 2.4, the PRO 150 series, but the associated hardware can only be rented. Based on this, motor cost, as well as the large jump in performance it was decided to conduct vehicle development tests with the PRO 98 series before experimenting with the larger motors that the final vehicle will use.

G. FUTURE WORK

The control assembly was designed but unable to be built based on the eight week lead time required to receive the P20-5AR gearbox, not leaving enough time for it to be implemented into the vehicle for the final launch. Additionally, very little aerodynamic analysis has been conducted for the control fins. The control fins currently modeled were originally designed for Grohe's subsonic vehicle and further analysis for the optimal supersonic fins is required.

The program could also benefit greatly from a new method of securing couplers together. Currently three to four shear pins are used between couplers but there is uncertainty of how in-flight dynamics affect the shear pins. Also, the relationship between vent holes, shear pins and CO2 canister size for each section is not well defined. Development of a system that mechanically holds the sections together and can be rapidly actuated to release and separate the sections would remove much of the uncertainty around vehicle separation.

III. GNC

A. SENSOR COLLECTION

Based on the low sampling rates previously achieved a new sensor package was developed that was capable of measuring vehicle attitude and controlling system response.

1. Hardware

The centerpiece of the GNC hardware is the processor which connects all of the other hardware, interprets signals, performs calculations, and issues commands in response. Two broad classes of devices that could be used are microcontrollers or single board computers.

a. Microcontroller v. Single Board Computer

Common examples of a microcontroller and a single-board computer are an Arduino and a Raspberry Pi, respectively. The advantages of a microcontroller versus a single-board computer for this project are shown in Table 4. After reviewing the requirements for the project and options available it was determined that a microcontroller was the better choice, as it was believed that the additional processing power offered by a single board computer was not necessary for the project requirements.

Table 4. Advantages of Microcontrollers and Single-Board Computers

Microcontroller Advantages	**Single Board Computer Advantages**
• Smaller • Simpler to Use • Lower Power Requirements • No Operating System Overhead	• More Powerful Processors • More Built-In Capabilities

b. Microcontroller Selection

Previous work had chosen the Arduino Uno as the microcontroller for the project for its widespread usage, universal serial bus (USB) connection, and compatible hardware [1]. Development migrated to the Arduino Mega when the need for more memory and processing power was desired, but Grohe concluded at the end of his work that a new processor was required [2].

After a review of options, the PyBoard, shown in Figure 21, was selected. Its more powerful processor was not expected to bottleneck the flight package and it runs MicroPython, a lightweight version of the Python programming language designed for use on microcontrollers. Its syntax similarity to MATLAB make it a language well suited to learning for project development by future students, and Python's general popularity ensure community support will remain readily available. Furthermore, if the project eventually does require a single board computer MicroPython's similarity to Python means that all code development can be rapidly transitioned to the new platform.

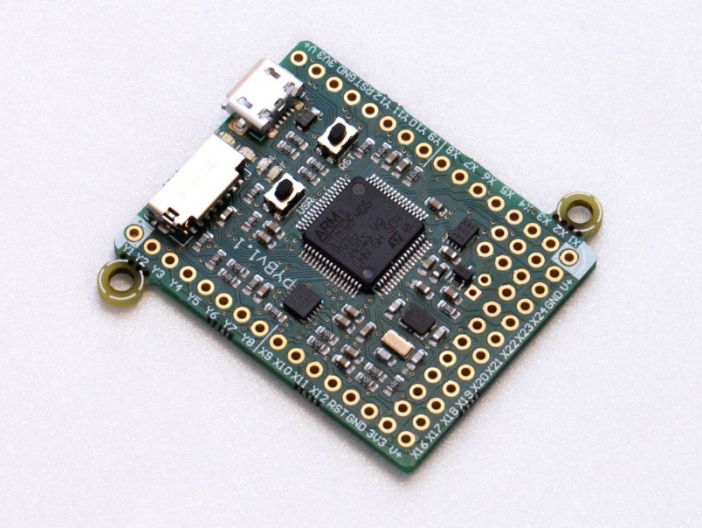


Figure 21. The Microcontroller Used, the PyBoard V 1.1.
Source: [11].

c. Attitude Sensing

For attitude sensing the myAHRS+, shown in Figure 22, was selected. Key features include:

- Triple axis gyroscope, accelerometer, and magnetometer
- Extended Kalman Filter with 100 Hertz (Hz) max output rate in Euler angles or Quaternions
- Universal Asynchronous Receiver-Transmit (UART) connectivity up to 460,800 bits per second and Inter-Integrated Circuit (I2C) up to 1 kHz

In addition to these features there was Python code available making implementation with the PyBoard straightforward. The implementation of the myAHRS+ is shown in the Appendix.

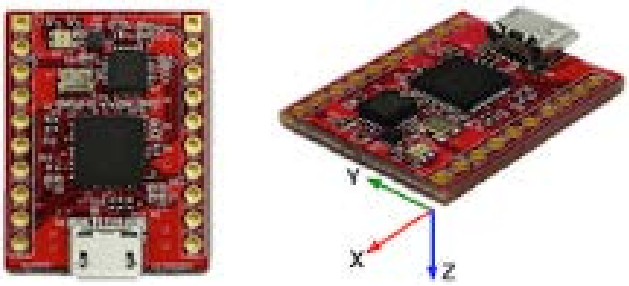


Figure 22. AHRS Used, the myAHRS+. Source: [12].

2. Attitude Sensing

Euler angles are used to identify the orientation of a body relative to a fixed coordinate frame. Their advantage is simplicity and intuitiveness, but a major drawback is revealed when pitch angle is at or very close to 90° gimbal lock. At this orientation a condition called gimbal lock occurs where a degree of freedom is lost and interpreting roll and yaw becomes impossible. This issue can be avoided at the expense of simplicity by quaternions. A quaternion is a four dimensional vector used to describe a three dimensional

attitude representation. This additional degree of freedom over defines the body, allowing for a unique representation regardless of orientation [13].

Because of the high likelihood of the vehicle being at or near a pitch angle of 90° the myAHRS+ attitude determination module was set to output quaternions. However, for testing and simplicity of implementation the quaternions were then converted to Euler angles for control logic, although this conversion may be removed at later development stages. The conversion implemented was taken from ME4703 class notes [14] and is shown in the Appendix.

3. Data Filtering

Figure 23 shows the measurements of one gyroscope channel over fifty seconds (sec) with the sensor stationary, with other channels showing similar performance. Two sources of error are readily apparent in the raw data. First, despite the sensor being held stationary the signal is oscillating around a mean value of 1.4364 deg due to a constant bias present in the sensor. The second source of variation seen is the high frequency jitter in the signal that is white noise due to thermo-mechanical events and is effectively a random error signal added to each reading. [15] The bias error in the figure is corrected by taking the mean of the data set and subtracting this value off. This method works well for post processing data but is not feasible for real time data adjustment. One option for correction of data on the vehicle is to take a calibration reading using the mean of a stationary data set upon sensor startup and store this value as a correction factor for use during flight.

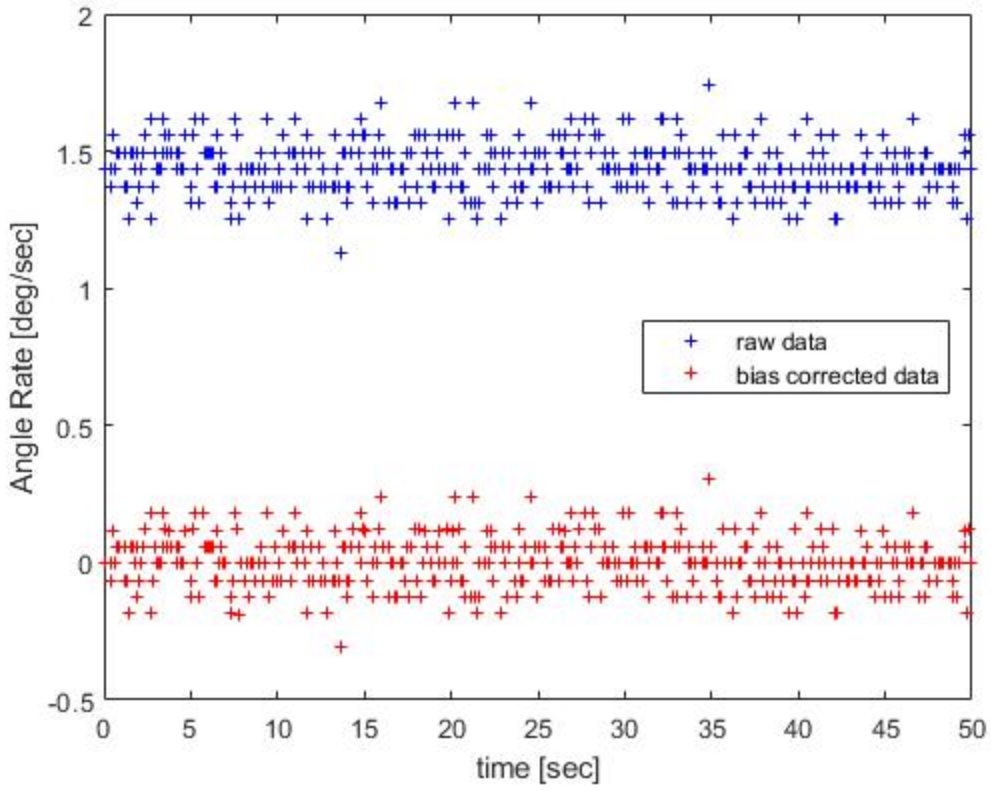


Figure 23. Noise and Bias in Stationary Gyroscope Readings

As vehicle motion is regulated by vehicle angular rates it was desired to minimize the impact of the white noise through a low pass filter. Both an exponential moving average (EMA) filter and filters generated with the MATLAB filter graphical user interface were analyzed, with the EMA filter selected for the project based on the simplicity of the EMA in both computation and implementation versus the very slight improvement of the MATLAB generated filter.

The EMA was implemented using Equation (1) where α is the weighting factor and x is the most recent data point. This filter design stems from the classical design of a linear discreet Kalman filter. Selection of α dictates the performance of the filter. If $\alpha = 1$ is used the output will follow the input exactly, with no consideration for previous values. Conversely, as $\alpha \to 0$ the output is less sensitive to input and will stick more closely to the previous values. The former case will provide a filter that follows the measurement

without suppressing noise while the latter case suppresses noise at the expense of response time. Analysis of stationary and flight data was conducted to evaluate the most appropriate selection for α.

$$\mathrm{EMA}(i)=\alpha x(i)+(1-\alpha)\mathrm{EMA}(i-1) \tag{1}$$

Figure 24 shows the bias-corrected stationary gyroscope readings overlaid with the EMA filter using $\alpha = 0.1$, 0.3, and 0.5. As expected, $\alpha = 0.5$ display the least rejection of noise while $\alpha = 0.1$ shows the most. Figure 25 and Figure 26 shows x axis gyroscope data output by the myAHRS+ from the February flight compared to EMA filtered data for various values of α while Figure 27 and Figure 28 show the same for May flight data. These figures show that $\alpha = 0.1$ exhibits a significantly greater lag than the other values of α. Based on these graphs it was determined that an alpha of 0.3 provided the best balance of rejecting white noise while still responding quickly enough to changes in the roll rate.

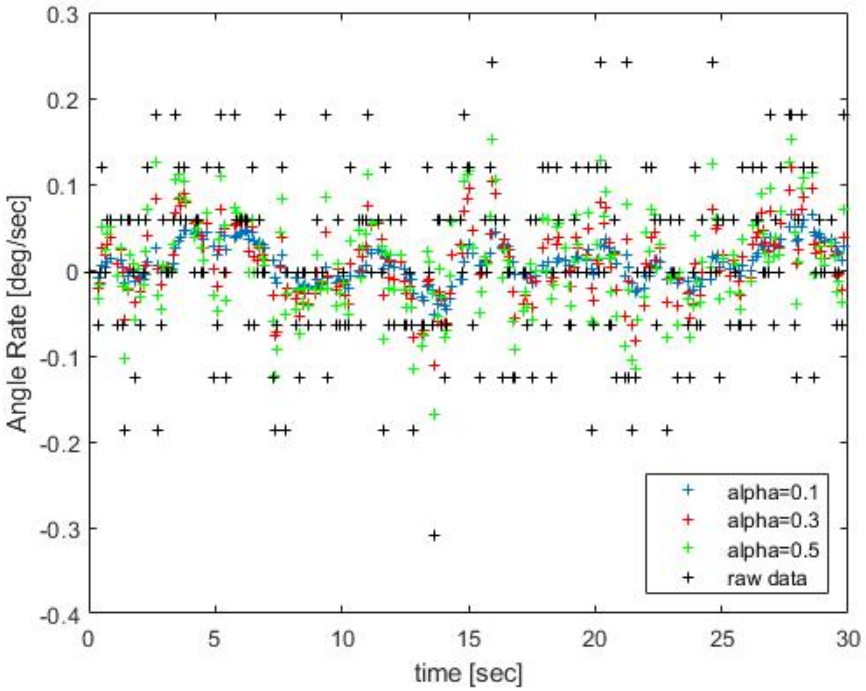


Figure 24. EMA Filters Applied to the Bias-Corrected Stationary Data

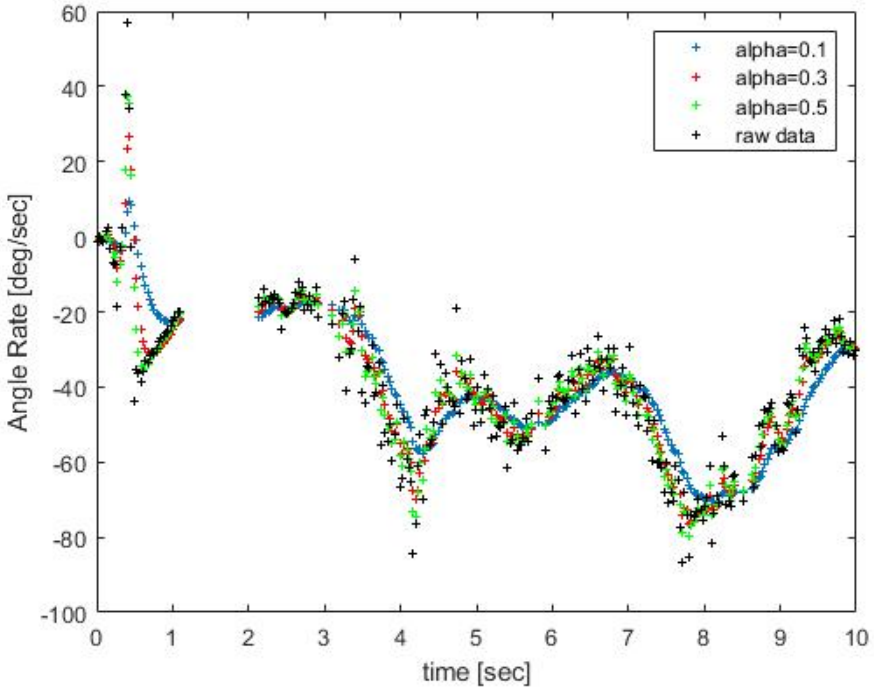


Figure 25. EMA Filters Applied to February Flight Data

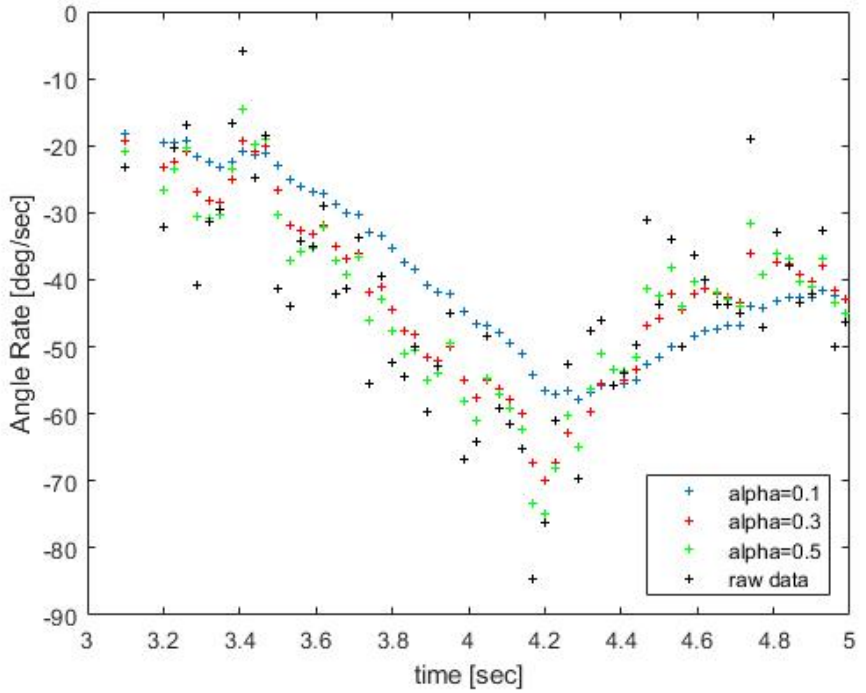


Figure 26. Zoomed EMA Filters Applied to February Flight Data

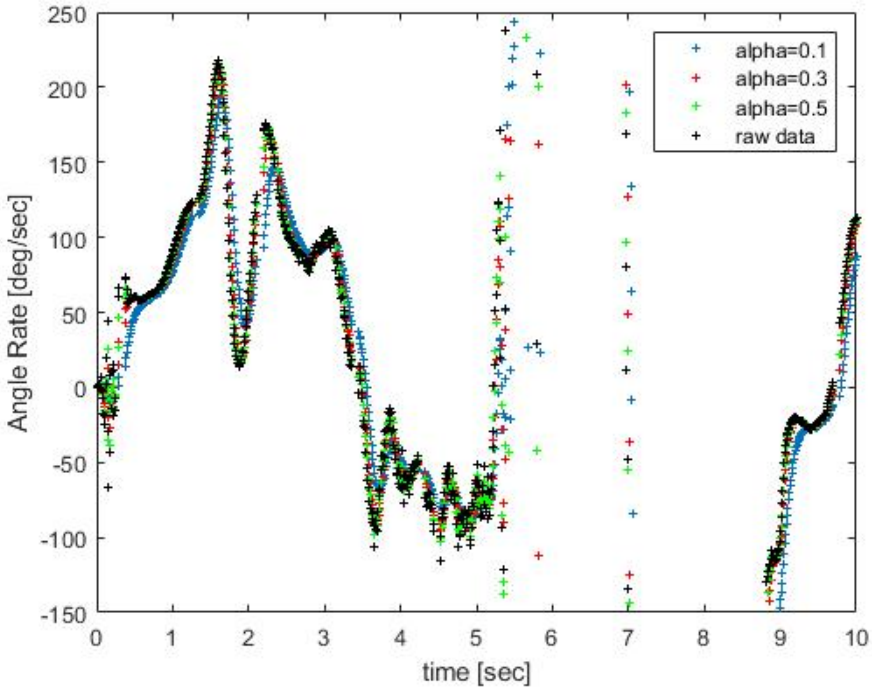


Figure 27. EMA Filters Applied to May Flight Data

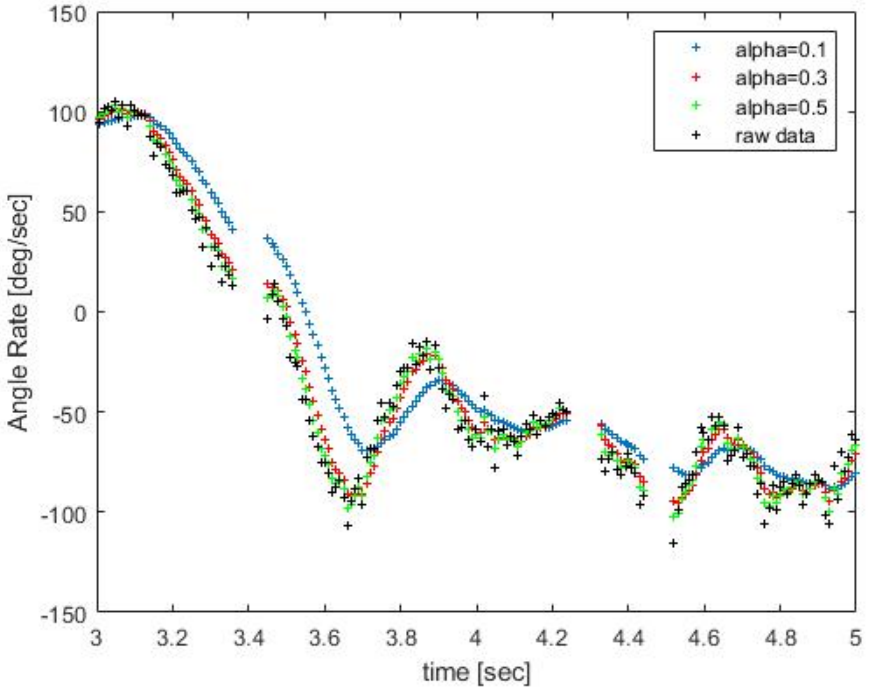


Figure 28. Zoomed EMA Filters Applied to May Flight Data

4. Sampling Rate

A significant motivation for going away from the Arduino based control system was the low sampling rates, with around 30 Hertz (Hz) achieved. The Nyquist sampling theorem dictates that a continuous signal should be sampled at twice its highest frequency to prevent data loss while in practice, due to noise in measurements, it is generally desired to sample at five to ten times the highest frequency. Therefore, to determine a target sampling for attitude control the frequency of oscillation in the three principal axes must be known. Based on vehicle geometry and mass distribution the oscillation about the vehicles longitudinal axis (roll) is expected to have the highest frequency and by ensuring that the sampling rate can adequately reconstruct this motion it can be assured that all vehicle oscillations are sufficiently sampled.

To estimate an upper limit on the roll frequency it was approximated as a pendulum about the center of the body and the frequency of oscillation was calculated using Equation (2) as 20.5 Hz for both the booster and sustainer. The area moment of inertia, I_{xx}, and mass, m, were measured from the SolidWorks model that was correctly updated for material properties and length, L, was half of the vehicle diameter.

$$\omega = \sqrt{\frac{mGL}{I_{xx}}} \tag{2}$$

This determination of the upper limit is corroborated with actual vehicle launches where vehicle motion begins to be lost when filters with a cutoff frequency of 12 Hz are applied to the data, confirming that a sampling rate of 30 Hz is not sufficient.

The hard limit on achievable sampling rate for the hardware was determined by the myAHRS+. It has a peak output rate of 100 Hz and it was believed that the PyBoard should be able to operate at or very near this frequency, but the initial program was only achieving 30 Hz, with this sampling rate seen in Figure 26. Testing revealed that the program loop rate was being bottlenecked by writing data to the Secure Digital (SD) card. For the first launch data was logged to the SD card as part of every program loop. The first update to the program was to store the data to a buffer and only write to the SD card when the buffer is filled. This extended SD card writes to every 40 program loops before the PyBoard was

at risk of running out of available buffer memory. Initial testing showed significant improvement, with sampling rate more than doubling to 68 Hz, but this was somewhat misleading. The program would run at 93 Hz while filling up the buffer, but the SD card write would then take 300 milliseconds (ms) to write the data to the SD card, resulting in unacceptable dead time for control response during data write.

Through further testing and research, it was found that the SD write bottleneck could be completely removed by changing the write mode to the SD card. It was taking 300 ms to write text data to the SD card in bunches, while a binary SD card write took slightly over one ms. By shifting to binary data writing, data was being written to the SD card in every program loop and the program bottleneck was now the reading to of data from the myAHRS+. After implementing this change loop rates of 98 Hz were achieved.

While this rate was could have been used in flight, it was slowed to ensure logging reliability. When achieving 98 Hz, the data file is opened at the start of the program and is closed at the very end. However, the file must be properly closed or the data will be lost. If power is momentarily lost in flight or the SD card becomes unseated all data since the most recent file close is lost. If the file close and open (cycle) actions were conducted in every loop the data rate was 78 Hz. As a compromise to ensure a significant portion of data would not be lost while still maximizing data sampling rate a counter was implemented to cycle the file once every second, giving an effective sampling rate of 90 Hz, shown in Figure 28, and minimizing the amount of data that would be lost in the event of system failure.

In addition to the file cycling limit, sampling rate is also limited by the UART baud rate of 460,800 bps, which is the method of communication between the myAHRS+ and PyBoard in between file cycles. It is believed that the remaining 2 Hz of sampling available to the myAHRS+ (neglecting file cycle) could be achieved by implementing an I2C communication protocol. Another possible improvement to the sampling rate could involve the use of multithreading or timers, which could allow for file cycling with sampling at the maximum rate of the myAHRS+.

B. SYSTEM ROLL CONTROL

For vehicle control a lateral autopilot architecture was selected that utilizes the rate and angle measurements in feedback to produce a command that regulates any attitude disturbances to zero. This classical two-loop architecture was selected because it conveniently allows for intuitive experimentation and tuning of the rate and the positional loops with the ability to independently control the damping ratio and the natural frequency of the resulting controller. The analytical model of this system is shown in Equation (3). The left hand side represents the control logic while the right hand side is the vehicle response. This equation is then manipulated to achieve the system model, shown in Equation (4).

System roll control was implemented using the equation:

$$\left(\dot{\phi}_{ref} - \dot{\phi}\right) K_d + \left(\phi_{ref} - \phi\right) K_p = I_{xx}\ddot{\phi} \tag{3}$$

and if

$$\phi_{ref} = \dot{\phi}_{ref} = 0$$

then

$$I_{xx}\ddot{\phi} + \dot{\phi}K_d + \phi K_p = 0$$

and taking the Laplace Transform gives the model:

$$\phi(s) = \frac{\frac{K_d\phi(0)}{I_{xx}} + s\phi(0)}{s^2 + \frac{K_d}{I_{xx}}s + \frac{K_p}{I_{xx}}}. \tag{4}$$

If $k_d = 0$ in Equation (3) then the transfer function simplifies to Equation (5), which shows why angle only cannot be used to control vehicle attitude, as the system poles lie on the imaginary axis. This represents an undamped system that will exhibit a sinusoidal oscillation, shown in Figure 29, that continues indefinitely. Thus by introducing rate control damping is added to the system, making it possible to achieve a stable attitude, shown in Figure 29 as under, over, and critically damped.

$$\phi(s) = \frac{s\phi(0)}{s^2 + \frac{K_p}{I_{xx}}} \tag{5}$$

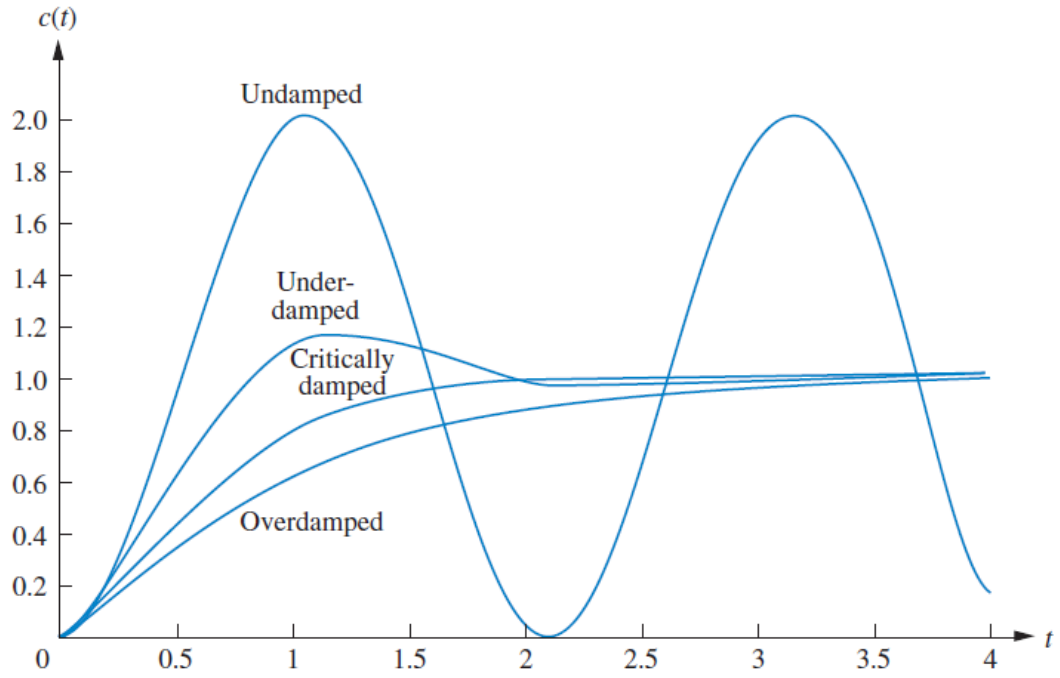


Figure 29. Second Order System Step Responses. Source: [16].

The denominator of Equation (4) can be matched to the general form of the second order system shown in Equation (6). The parameters ζ, damping ratio, and ω_n, natural frequency, can be related to the desired specifications for a second order system to calculate the required values of the control gains, K_p and K_d, to adjust the control response. These specifications are rise time, peak time, percent overshoot, and settling time. Any two of these specifications can be defined for the problem and the necessary control gains calculated.

$$s^2 + 2\zeta\omega_n s + \omega_n{}^2 \tag{6}$$

However, the optimum control gains for the vehicle will change throughout the flight in two significant ways. The first is the step change in system configuration when

the booster and sustainer separate, requiring a change in control gains. The second is the changing control authority of the fins as the dynamic pressure (q) changes with flight speed and altitude. One relatively simple way to adjust for this is to use a gain scheduling technique to make the control gain a function of the dynamic pressure as shown in Equation (7). Applied to the sustainer, the gain (K_{cruise}) can be optimized for the cruising dynamic pressure and then adjusted for the current dynamic pressure by a simple ratio.

$$K_{current} = \frac{K_{cruise} * q_{current}}{q_{cruise}} \tag{7}$$

C. FUTURE WORK

System control is very early in its implementation, leaving much room for future work. Only roll control was discussed but the same ideas will need to be extended to pitch and yaw control as well to achieve the specified flight profile. The 2018 ME4704 class demonstrated the ability to take the missile model developed in ME4703 and update the model parameters using Missile DATCOM, a semi-empirical modeling database, to obtain estimates of aerodynamic coefficients and the modeling software SolidWorks [17] for estimates of vehicle mass properties [14]. Implementing this vehicle model in Simulink would provide a much better model for optimizing control gains though flight stages.

Efforts were made unsuccessfully to implement the TinySine WiFi Skin [18] with the PyBoard to allow for wireless communications during testing and while on the launch rail. The system was able to connect to a Wi-Fi network but failed whenever data transmissions were attempted. A new version of the PyBoard, the PyBoard D-series has on board Bluetooth and Wi-Fi connectivity and is recommended for implementation to provide these capabilities with better built in support.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TESTING/RESULTS

Testing was conducted both on the vehicle through launches and also on the GNC hardware and software through bench tests.

A. CONTROL SYSTEM TESTING

A platform for qualitative testing of control architecture was desired to verify system operability and provide a cheap, rapid, and readily observable check on the control system performance prior to full scale flight testing. However, its utility is somewhat limited as specific parameters derived from its use cannot be used interchangeably in the flight vehicle due to their lack of sufficient similitude.

1. Test Platform Development

Based on the goal of having a simple platform to test system dynamics only roll was incorporated into the test platform design. Figure 30 shows the assembled platform. It consists of a simulated missile body that has four fins driven by servos. The servos are Hitec HS-5245MG servos provided by the Advanced Robotic Systems Engineering Laboratory and are controlled by the PyBoard using myAHRS+ for attitude sensing. More compact servos than those of the actual flight vehicle were used to allow for a smaller test platform, but the actual servos will need to be characterized to account for their effect on the system. This missile body is slid onto a shaft supported by bearings that allows the missile body to roll. This assembly is then placed in an airstream and control response tested.

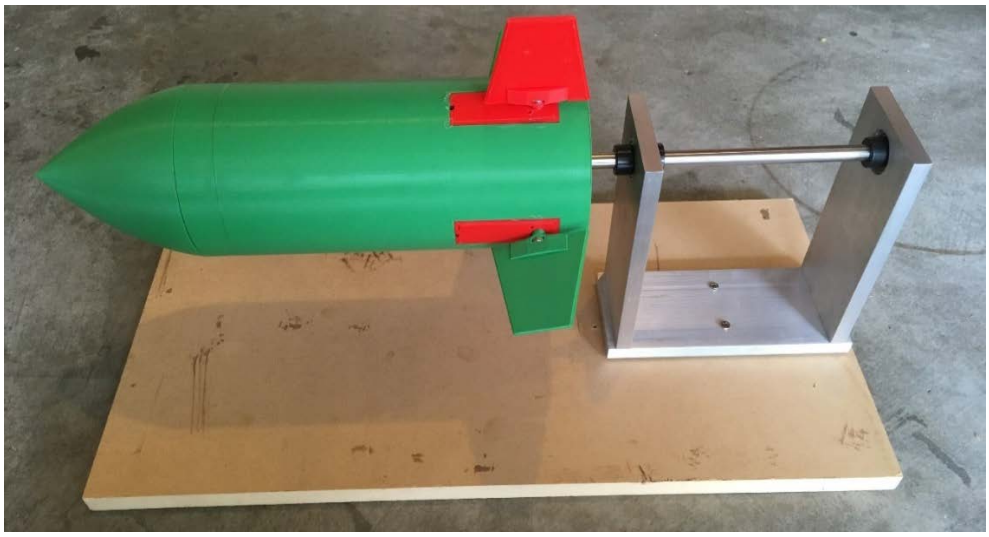


Figure 30. Roll Test Platform

The body was designed to hold four servos that each drive a fin in the plus configuration. Initially the design included mounting fins directly to the servo arm but a discussion with Professor Kevin Jones believed that this setup would provide poor angular accuracy and transmit the aerodynamic fin forces directly to the servo shaft and bearings which were not designed to support these forces [19]. To alleviate these issues, the assembly shown in Figure 31 was built with a sleeve bearing to support the fin forces and a four bar system to transmit the rotation of the servo horn to the fin control arm. This acts as a reduction system, increasing servo torque and control precision. Servo rotation required for a given fin deflection is calculated in real time using the solution to the four bar linkage problem developed by Constans et al. [20].

After assembly a suitable air stream was sought for testing. Based on its dimensions no wind tunnel on campus could fit the platform in it test area so other options were tested. A large pedestal fan was tried but did not create enough air flow focused on the body to create rotation with maximum fin deflection, but a blower rated for 2670 cubic meters per minute generated enough airflow across the body to permit the fins to adequately control vehicle roll.

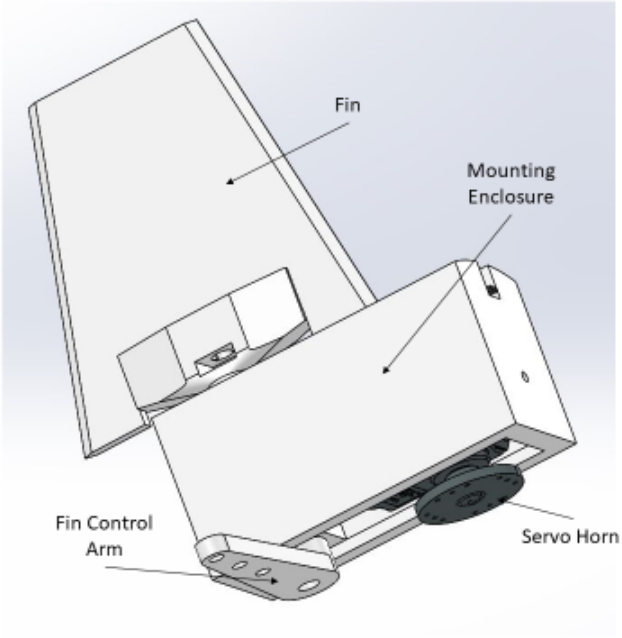


Figure 31. Fin Control Assembly

2. Disturbance Response

With the test platform operational the first test was to develop a simple controller capable of regulating the vehicle roll angle. Figure 32 shows data logged by the onboard control system with a proportional only controller. This result matches the neutrally stable behavior predicted by Equation (5), and shown in Figure 29, for an undamped system where the body oscillates indefinitely. At 3 seconds an external disturbance is imparted on the vehicle and the vehicle oscillates in a repeatable pattern demonstrating a neutrally stable second order system; this result is expected for the double integrator dynamics with proportional only feedback. To stabilize this system velocity feedback was added and roll control was implemented as was developed in Equation (4), with K_p=0.5 and K_d=0.1. The results of this trial are shown in Figure 33 where the system can now be stabilized by the controller, exhibiting characteristics of an underdamped system where vehicle angle repeatedly overshoots the desired angle.

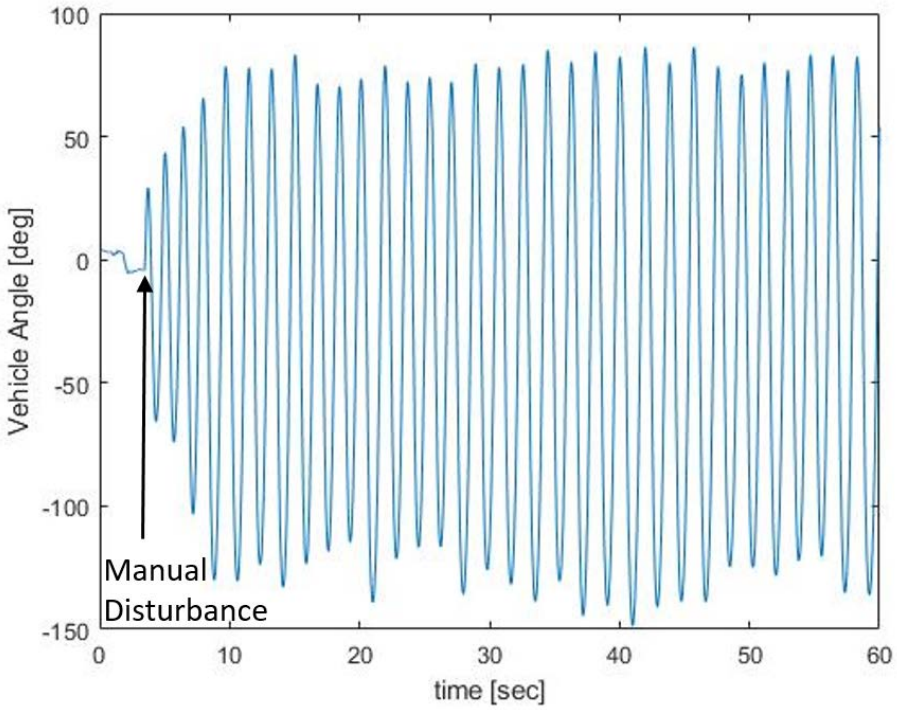


Figure 32. Test Platform Response with Proportional-Only Control

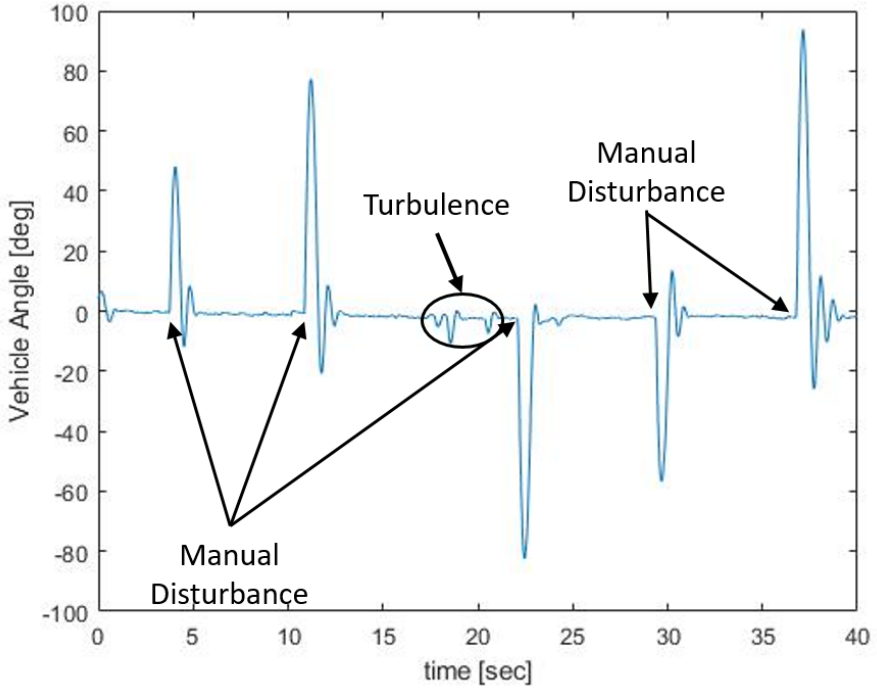


Figure 33. Test Platform Response with Proportional and Velocity Control

3. Future Work

Through the assembly and use of the test assembly multiple areas for improvement have been found. Figure 32 shows that the body is not well balanced, as the midpoint of the oscillations does not occur at a roll angle of zero. This interferes with system response and could be rectified by a built in balancing system. It can also be seen in Figure 33 that the system response is somewhat inconsistent, where the larger disturbance at 23 seconds has a smaller overshoot than the smaller disturbance at 30 seconds. This is possibly a result of the turbulent flow generated by the blower that can also be seen in the fluctuations between 17 and 21 seconds where no outside disturbance was introduced into the system.

Additionally, the fins selected were chosen to maximize surface area, not performance, and an analysis on optimal fin shape could enhance the system response. This will also be dependent on the airflow over the vehicle and the control gains used so these items must all be looked at in parallel Advances to the system that allow for pitch and/or yaw could better illustrate the dynamics of flight and the coupling of vehicle motion.

B. FLIGHT VEHICLE TESTING

Flight tests were conducted at the Friends of Amateur Rocketry test site in Randsburg, California.

1. November Two-Stage Test

A flight test was conducted on November 17, 2018. The vehicle tested is similar to the vehicle discussed in Chapter II, with the notable difference of no active control, as the sustainer fin assembly was replaced with fixed fins. Its goals were to collect data from the myAHRS+ and PyBoard for further development as well as demonstrate use of the booster-sustainer coupler, Raven 4 performance, TelemetryPro Tracking System, second stage ignition, and the streamer side deployment. Most of these goals were not tested as the vehicle broke apart at the booster-sustainer coupler in mid-flight during booster burnout. At the time of breakup, the vehicle was traveling Mach 1.149 at an altitude of 850 meters 4.2 seconds into the launch based on the TelemetryPro system. The sustainer parachute deployed at this point and was ripped free from the bulkhead, which resulted in a loss of

nearly all of the sustainer. The booster, rear recovery section, and the nosecone were recovered.

Video footage from the rear recovery section is inconclusive but it seems that the rocket began to separate prematurely at the booster/sustainer coupler for unclear reasons. Possibilities include drag separation or inadequate venting resulting in a pressure buildup inside the coupler. The sustainer then began to cant, which could have resulted in coupler failure, shown in Figure 34, due to the increased aerodynamic forces.

Changes to the vehicle from this test for the final launch include increasing the number of carbon fiber tubes from two to four and using a fiberglass wrapped phenolic tube around the fins rather than a plain phenolic tube to increase the strength of the coupler. Pressure sensors were incorporated onto future designs to allow for the monitoring of coupler internal pressure and allow for better determination of vent hole sizing. Finally, additional cameras were added to the vehicle to allow for better determination of inflight events.

Figure 34. Boken Booser-Sustainer Coupler

One goal for this launch that was tested was the use of the TelemetryPro system for vehicle recovery. The system worked exactly as advertised and allowed for a simple retrieval of the nosecone while also permitting review of the flight based on both GPS and accelerometer data.

2. February Data Recording Test

The primary goal of the second launch was to collect data using the myAHRS+ and PyBoard. To ensure this goal was achieved the scope of other tests was scaled back significantly. This vehicle was a single stage vehicle with total impulse and a thrust to weight ratio similar to the expected final vehicle. This vehicle was successfully launched and recovered with the TelemetryPro system at a range of 8 km from the launch site.

Figure 35 shows some of the data collected by the sensor package. Significant noise is seen in pitch rate from 3 seconds to 9 seconds, demonstrating the need for a filter on the data if the signal is going to be used for attitude control. It is apparent from the plots that there was a dropout in logging from approximately one second to two seconds. It is believed this is due to motion of the SD card, as the PyBoard was oriented where acceleration would tend to disengage the SD card. For future builds the PyBoard was oriented so that acceleration would help seat the SD card and a dab of hot glue was applied as well. The angle plot reinforces the need for roll control in the vehicle had a tendency to slowly roll.

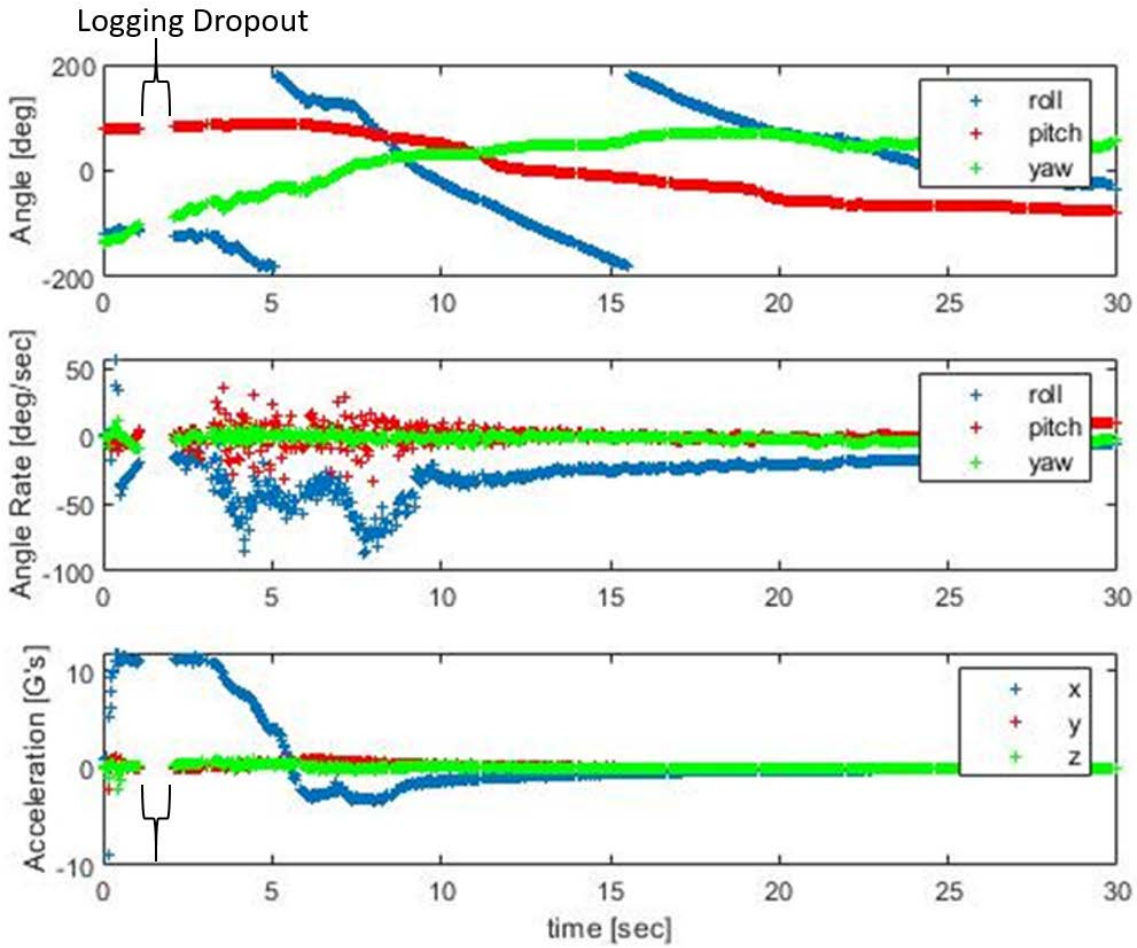


Figure 35. Summary of Data from February Launch

A problem encountered at this launch was that both the drogue and main parachutes deployed at apogee, resulting in a slow descent where the vehicle drifted over 8 km before landing. This reinforced the need to rework the recovery deployment mechanisms to prevent their premature deployment and the side streamer deployment previously discussed was implemented again for trial in the final vehicle.

3. May Two-Stage Test

A flight test was conducted in May 2019. The vehicle tested is shown in Figure 36 and is built as designed in Figure 5, with the exception of having fixed fins in place of the fins in the control section. Its goals were to demonstrate use of the booster-sustainer coupler, Raven 4 performance, collect data from the sensor package, demonstrate second

stage ignition, and the streamer side deployment. The vehicle broke apart mid-flight at 1,365 meters, Mach 1.26, 5.2 seconds into the launch. Based on ground observation both parachutes deployed at this time and were ripped free from the vehicle. The sustainer's descent was slowed by a streamer so while it was significantly damaged all of the electronics were recovered. Additionally, the TelemetryPro tracking system was successfully recovered undamaged despite the nosecones' free fall impact. All other components were severely damaged or lost.

Flight data from the external camera was unable to be recovered, making it particularly difficult to determine the sequence of events that led to failure. Video from the camera monitoring the booster-sustainer coupler inner region was retrieved but did not show any abnormal behavior that would lead to vehicle separation. The streamer and nosecone separation charges were fired, likely during sustainer decent, indicating operation of the Raven 4 event control, but it is unclear where in the flight these were triggered. Figure 37 shows what was recovered from the male portion of the booster-sustainer coupler. The phenolic tube was shattered in flight but only one of the carbon fiber tubes is broken. It seems plausible then that the booster and sustainer may have cleanly separated for an instant but the booster motor continued to provide thrust, driving into the sustainer where further vehicle breakup occurred.

When the vehicle broke apart 5.2 seconds into the launch the Cesaroni PRO98 N3301 should have still been providing around 800 Newtons of thrust, making it unlikely that base drag was the cause of vehicle separation. This is consistent with the x axis acceleration data, shown in Figure 38, that shows a small positive acceleration. It is possible that pressure buildup in a section was responsible for the separation but it is difficult to postulate a cause based on the data. The z axis acceleration was at its 16 gravity (G) limit from 5.29 to 6.11 seconds, likely when the parachutes deployed. The data shows acceleration in the y and z directions increasing starting at 3.3 seconds along with additional noise in the x direction, corresponding to when the vehicle reached Mach 1. The y and z accelerations continued to grow in magnitude until vehicle breakup and are likely the result of an aerodynamic force. Possible causes of this unknown aerodynamic force

include the camera housing that protrudes from the body, early streamer deployment, or body flex due to inadequate or partially separated couplers.

Figure 36. Vehicle Launched in the May Test

Figure 37. Broken Booster-Sustainer Coupler

Figure 38 shows some of the data collected from the sensor package. In addition to the higher sampling rate achieved over the February launch there were no dropouts in logging from the SD card coming unseated, despite similar forces to the previous launch. However, the file cycle process can be seen occurring at one second intervals throughout the flight that results in about a tenth of a second delay in cycle execution. One exception to this is seen in the two delays that occurred between 4 and 5 seconds, with the cause unknown.

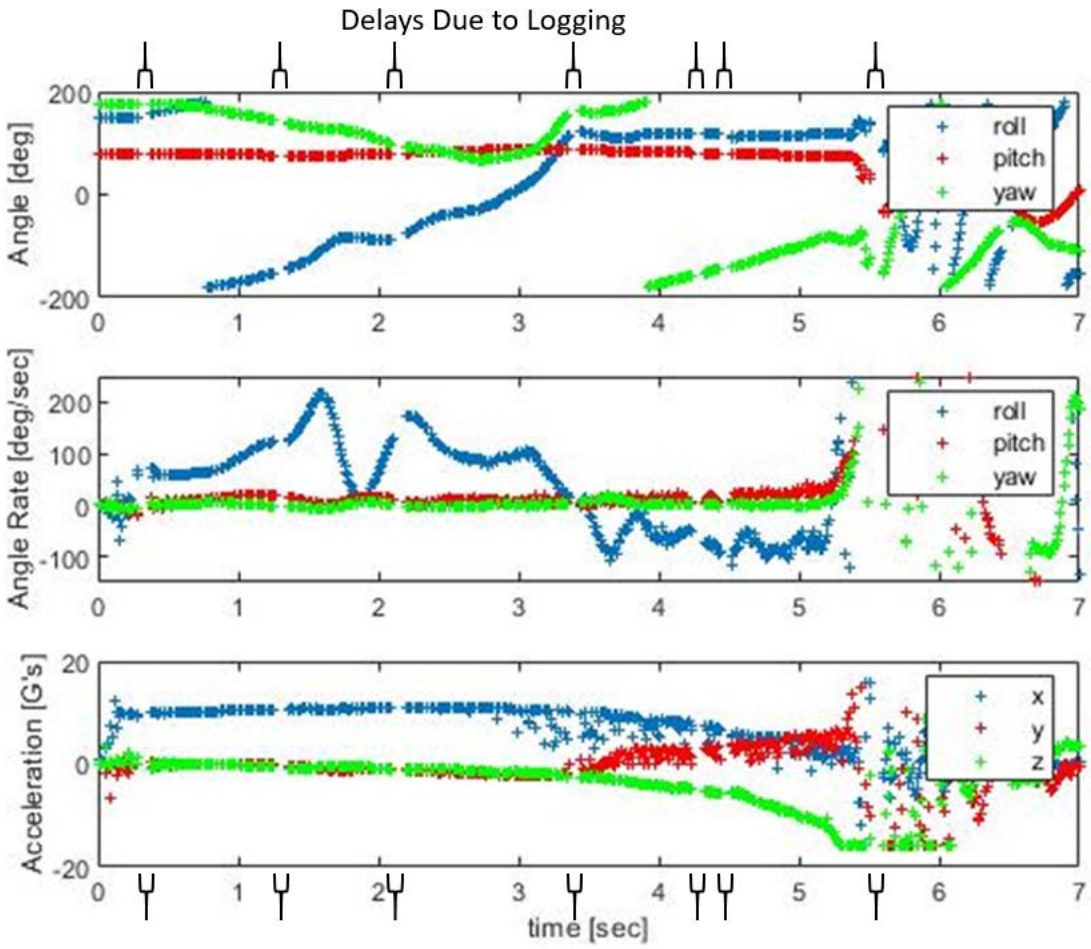


Figure 38. Summary of Data From May Launch

V. CONCLUSIONS

A two-stage rocket-powered vehicle on which to implement a tailorable GNC system was designed and built. The design for an adaptable, modular control system for a missile demonstrator as well as a two-stage test vehicle was developed. Two attempts were made to launch and recover a two-stage vehicle that resulted in vehicle break up at transitional flight conditions where staging was expected to occur. The cause of break up was not clear but has identified several areas to focus future efforts on, particularly controlling vehicle stage separation.

A guidance and control system for the vehicle along with a ground based attitude control test platform was designed and constructed. The sensor package was tested on two flights and demonstrated the most complete sensor package fielded by the NPS Rocket Lab based on the parameters measured and their sampling rates. A Kalman Filter Weighting Factor, α, of 0.3 was found to be sufficient at removing sensor noise in angle rate measurements while minimizing phase lag.

A stationary test platform was designed to qualitatively assess system suitability, and does indeed appear to be a suitable platform upon which to base future flight control efforts. This system demonstrated the ability for the sensor package to control test platform fins using an easily tunable proportional and derivative controller to counteract outside disturbances, maintaining a specified roll angle.

The structural failures during two separate launch attempts demonstrated the need for additional focus on the structural design of a two stage vehicle. The structural failure of the vehicle during the two stage launches precluded testing of some of the efforts in this thesis. The primary complication is based around the stage separation and it is recommended to develop a system that rigidly secures two sections together mechanically until separation is desired where the system releases and allows sudden, complete separation instead of a more gradual event based on uncertain pressure differentials and motor burn characteristics.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

This appendix is intended to assist future students in implementing the flight code developed for this project. To begin, the files boot.py, main.py, myahrs.py, and mprlslib.py are copied into the PyBoard file structure via USB. A serial monitor is recommended for easy monitoring of program output and exceptions, such as PuTTY for Windows or Screen for Mac or Linux. In Screen (although this should apply to any serial monitor), once the files are uploaded and the PyBoard is connected to the computer via USB CTRL+C raises a keyboard interrupt to exit any running program and CTRL+D performs a soft reset, maintaining the connection between the computer and PyBoard but restarting the PyBoard allowing for monitoring the complete execution of the code. The serial monitor can be used as an interactive interpreter after the code has finished running or by pressing CTRL+C.

Figure 39 shows boot.py, the file that is run first every time the PyBoard is powered up or reset. This file allows the user to select if the PyBoard will connect to the computer in its normal mode or USB storage mode. If the user switch on the PyBoard is pressed within two seconds of startup the PyBoard will go to USB storage mode and run the file ‘test’, declared in line 23. The ‘test’ file does not exist; it is simply a placeholder to exit boot.py. If the user switch is not pressed the PyBoard will go to its normal mode and run the file declared by line 28, ‘main.py’.

Figure 40 shows main.py, the program that directs the actions during flight. The first 29 lines initialize variables and set up timers. Line 24 sets the specifications for pulse width modulation control of servos. Current values are for the roll test platform servos. Lines 32 to 40 create the folder ‘log’ on the SD card inserted into the PyBoard if it does not exist, and then creates the file named logX, where X is the number of files in the log directory plus one in append binary mode. Append mode is used to prevent overwrite of any data previously saved to the file name and binary mode allows for much faster write times but requires unpacking to view data, managed by readdata.py.

Lines 44 to 53 control connecting to the myAHRS+ on bus 6 of the PyBoard. The Default baud rate for the myAHRS+ is 115,200 and it must be initialized at this rate if it

has not been powered up and changed to another baud rate recently. Lines 51–53 can be uncommented to shift the PyBoard to a different baud rate as the myAHRS+ will save this baud rate even if unpowered for a short period of time and the connection would need to be reestablished at the baud rate last used. For future work it is recommended to set up the program to establish the highest baud rate regardless of previous settings or shift to an I2C communication protocol.

Immediately after changing myAHRS+ settings the myAHRS+ sends confirmation messages that are cleared from the queue by lines 57 and 58.

While the timer started on line 61 is less than the time specified in line 29 the primary loop, lines 65 to 101 execute repeatedly. Line 70 reads a line from the myAHRS+ and lines 71–74 attempt to parse the line. If an exception is raised the program execution point is moved back to line 65 and the loop restarts.

Lines 83-86 control the servos, with line 85 being used for the roll test platform to calculate servo rotation required for a desired fin angle. Lines 89-93 read the MPRLS sensor while ensuring it will not interfere with the program loop if no pressure is available.

Line 96 and 97 write the data to the SD card and lines 98–101 close and then open the log to ensure data is saved at one second intervals to minimize data loss in the event of failure. If lines 98-101 are not included the program executes more quickly but if an event occurs that does not properly close the file all data is lost that was written after the last file close. These events could include an unhandled exception, power loss, or the SD card being disconnected. Other options that may have advantages over the current implementation include opening the file in unbuffered mode or using the built-in function *flush*().

The library of most of the functions for main.py is shown in Figure 41, myahrs.py. *myahrsinit*() controls initialization of communications between the PyBoard and myAHRS+ and will need to be modified if the automatic baud rate selection previously discussed is implemented. The *send_command*(), *read_myahrs*(), and *parse_data*() functions are heavily based on those outlined by the tutorial for the myAHRS+ [12].

The function *quat2eul*() uses the calculation from ME4703 class notes [14] to convert quaternions to Euler angles. Lines 104-110 catch an exception raised when the

pitch is very near 90 degrees (approximately within 0.5 degrees) where the input to arcsin is greater than 1, which may be the result of a rounding error. If this occurs the pitch value is set to 90 degrees

The function *eul2quat*() has not been verified for correct operation but is based on ME4703 class notes [14] to convert Euler angles to quaternions and may be useful further in the development process where most of the calculations could be handled in quaternions rather than Euler angles.

Two functions, *gravitycorrrpy*() and *gravitycorrquat*(), have been implemented to remove the acceleration due to gravity from the raw acceleration output. This may be useful later in development in establishing and maintaining horizontal flight. The quaternion version is from the FreeIMU project [21]. Both of these functions assume the acceleration of gravity is one and a possible improvement would be to detect the local gravity on startup and correct for this value.

Figure 42, mprlslib.py, is based on the Adafruit library for CircuitPython [9] but most of the expanded functionality was not transitioned over. Also values returned do no match actual atmospheric pressures and it is believed the conversion equation in lines 17–19 needs to be adjusted for better calibration parameters.

Figure 43 is the file to unpack the data file saved by main.py to a format that can be read by Excel and MATLAB. The parameters used to pack the data must be used in unpacking. Line 7 is the file and its location that is to be unpacked, while line 15 is the location the unpacked file will be saved.

File - E:\NPS\Thesis\Thesis Writing\programs\boot.py

```
 1 # boot.py -- runs on boot-up
 2 # Let's you choose which script to run.
 3 # For USB access to SD card:
 4 #       * press reset and do nothing else
 5 #       * press user switch and hold until orange LED goes
    out
 6 # To run 'myahrslogger.py':
 7 #       * press reset
 8
 9 # This is to prevent issues with multiple devices
   accessing the SD card.
10
11 import pyb
12
13 pyb.LED(3).on()
14 pyb.delay(2000)
15 switch_value = pyb.Switch()()
16 pyb.LED(3).off()
17
18 pyb.LED(4).on()
19
20 # if switch wasn't pressed run this in USB mode - does not
    have to be a valid file
21 if switch_value:
22     pyb.usb_mode('VCP+MSC')
23     pyb.main('test')
24
25 # is switch was pressed run this
26 else:
27
28     pyb.usb_mode('VCP+HID')
29     pyb.main('main.py')
30
```

Page 1 of 1

Figure 39. Boot.py

File - E:\NPS\Thesis\Thesis Writing\programs\main.py

```
 1 import struct
 2 import pyb
 3 import myahrs
 4 import os
 5 import mprlslib
 6
 7 # control constants
 8 kroll = 1
 9 kdroll = 1
10 servomax = 20 #(degrees)
11
12 # sets the time between cycling the file to ensure
   previous data is not lost
13 # in the event of a failure
14 tlog = 1000 #(milliseconds)
15
16 # sets time to run the program
17 ttot = 60 # time to run in minutes
18 ttot *= 60000 # converted to milliseconds
19
20 # pin setup and servo calibration
21 blue = pyb.LED(4)
22 red = pyb.LED(1)
23 #servo1 = pyb.Servo(1)
24 #servo1.calibration(1000, 2000, 1500, 2000, 2200)
25
26 # initialize variables
27 euler = [0,0,0]
28 filtgyro = [0,0,0]
29 trun = 0
30
31 # log setup
32 if 'log' not in os.listdir('/sd/'):
33     os.mkdir('/sd/log')
34     print('created log dir')
35 else:
36     print('log dir already there')
37
38 filename = '/sd/log/log'+str(len(os.listdir('log'))+1)+'.
   bin'
39 print('opening'+filename)
40 log = open(filename, 'ab')
41
42 # myAHRS setup
43 # opens connection at the default baud rate
```


Figure 40. Main.py

File - E:\NPS\Thesis\Thesis Writing\programs\main.py

```
44 myahrsbus = 6
45 baudrate = 115200
46 uart = pyb.UART(myahrsbus, baudrate)
47 myahrs.myahrsinit(uart, baudrate)
48
49 # higher baud rate slightly increases logging speed but
   makes startup more
50 # difficult
51 #baudrate = 460800
52 #uart = pyb.UART(myahrsbus, baudrate)
53 #myahrs.myahrsinit(uart, baudrate)
54
55 # wait one second to prevent interference of myAHRS
   settings responses with
56 # readings, and then clear the line
57 pyb.delay(1000)
58 kill = myahrs.read_myahrs(uart)
59
60 # start the timers
61 tstart = pyb.millis()
62 logtimer = pyb.millis()
63 blue.on()
64
65 while ttot > trun:
66
67     trun = pyb.elapsed_millis(tstart)
68
69     # read myAHRS, reattempting if the message is unable
   to be parsed
70     line = myahrs.read_myahrs(uart)
71     try:
72         no, q, accel, gyro, mag, T = myahrs.parse_data(
   line)
73     except:
74         continue
75
76     # calculate euler angles
77     euler = myahrs.quat2eul(q)
78
79     # filter gyro data
80     filtgyro[0] = myahrs.filter(gyro[0], filtgyro[0], 0.3)
81
82     # fin command
83     rollerr = kroll*euler[0] + kdroll*filtgyro[0]
84     rollcmd = myahrs.servorange(rollerr,servomax)
```


Figure 40. (con't) Main.py

File - E:\NPS\Thesis\Thesis Writing\programs\main.py

```
 85      #rollcmd = myahrs.barangle(rollcmd)
 86      #servo1.angle(rollcmd)
 87
 88      # read pressure sensor, loop is unaffected if read
    fails
 89      try:
 90          press = mprlslib.readpres(1)
 91      except:
 92          press = 0
 93          pass
 94
 95      # log data and save every tlog interval
 96      data = [no]+q+accel+gyro+mag+rpy+[filtgyro[0]]+[
    rollcmd]+[press]+[trun]
 97      log.write(struct.pack('%df' % len(data), *data))
 98      if tlog < pyb.elapsed_millis(logtimer):
 99          log.close()
100          log = open(filename, 'ab')
101          logtimer = pyb.millis()
102
103 log.close()
104 blue.off()
105 print('\nclosed '+filename)
106
107
```

Page 3 of 3

Figure 40. (con't) Main.py

File - E:\NPS\Thesis\Thesis Writing\programs\myahrs.py

```
1  """myAHRS+ functions for MicroPython based off of python
   example from github for myAHRS+"""
2  import pyb
3  import math
4
5  def myahrsinit(uart):
6      """Initializes PyBoard connection to myAHRS on uart
   and sets up myAHRS
7      settings"""
8
9      uart.init(115200, bits=8, parity=None, stop=1)
10
11     send_command(uart, 'mode,AC')
12
13     send_command(uart, 'asc_out,QUATIMU')
14
15     send_command(uart, 'divider,1')
16
17     # can change the baud rate
18     #send_command(uart, 'baudrate,460800')
19
20
21 def send_command(serial_port, cmd_msg):
22     """Sends command to myAHRS via serial_port. Message
   options:
23         A = ASCII output, T = trigger mode
24         B = Binary output, C = continuous
25
26         RPYIMU has Euler attitude and compensated IMU
27         QUATIMU for quaternions and compensated IMU"""
28
29     cmd_msg = '@' + cmd_msg.strip()
30     crc = 0
31     for c in cmd_msg:
32         crc = crc ^ ord(c)
33     command = cmd_msg + '*%02X' % crc + '\r\n'
34     serial_port.write(command.encode())
35
36     if cmd_msg != '@trig':
37         while True:
38             line = serial_port.readline().strip()
39             return line
40
41
42 def read_myahrs(serial_port):
```

Page 1 of 6

Figure 41. Myahrs.py

File - E:\NPS\Thesis\Thesis Writing\programs\myahrs.py

```
43      """Attempts to read message from myAHRS until
    successful"""
44
45      while True:
46          # read data message
47
48          line = serial_port.readline()
49
50          # attempt to avoid error when no message is
    received
51          try:
52              return line.decode()
53          except:
54              print('msg not received')
55              continue
56
57
58  def parse_data(line):
59      """parses QUATIMU or RPYIMU data message into
    components"""
60
61      line = (line.split('*')[0]).strip()  # discard crc
    field
62      fields = [x.strip() for x in line.split(',')]
63
64      if fields[0] == '$QUATIMU':
65          # myahrs quaternion sequence is x,y,z,w, this
    outputs in w,x,y,z
66          sequence_number, q1, q2, q3, q0, ax, ay, az, gyrox
    , gyroy, gyroz, magx, magy, magz, T = \
67              (float(x) for x in fields[1:])
68          q = [q0, q1, q2, q3]
69          a = [ax, ay, az]
70          gyro = [gyrox, gyroy, gyroz]
71          mag = [magx, magy, magz]
72
73          return int(sequence_number), q, a, gyro, mag, T
74
75      elif fields[0] == '$RPYIMU':
76          sequence_number, phi, theta, psi, ax, ay, az,
    gyrox, gyroy, gyroz, magx, magy, magz, T = \
77              (float(x) for x in fields[1:])
78          rpy = [phi, theta, psi]
79          a = [ax, ay, az]
80          gyro = [gyrox, gyroy, gyroz]
```

Page 2 of 6

Figure 41. (con't) Myahrs.py

File - E:\NPS\Thesis\Thesis Writing\programs\myahrs.py

```
 81          mag = [magx, magy, magz]
 82
 83          return int(sequence_number), rpy, a, gyro, mag, T
 84
 85      else:
 86          return -1
 87
 88
 89  def quat2eul(q):
 90      """Converts quaternions to euler angles. Try/Except
    to catch arcsin of value greater
 91      than 1"""
 92
 93      q0 = q[0]; q1 = q[1]; q2 = q[2]; q3 = q[3]
 94
 95      q22 = q2 * q2
 96
 97      r2d = 180/math.pi
 98
 99  #   Original
100      # Calculate roll
101      roll = r2d * math.atan2(2 * (q2 * q3 + q0 * q1), 1 -
    2 * (q1 * q1 + q22))
102      # Calculate yaw
103      yaw = r2d * math.atan2(2 * (q1 * q2 + q0 * q3), 1 - 2
     * (q22 + q3 * q3))
104      try:
105          # Calculate pitch
106          pitch = -r2d * math.asin(2 * (q1 * q3 - q0 * q2))
107          return [roll, pitch, yaw]
108      except:
109          print('pitch error')
110          return [roll, 90, yaw]
111
112
113  def eul2quat(angles):
114      """Converts euler angles to quaternions
115      NOT VERIFIED"""
116
117      c1 = math.cos(angles[2])
118      c2 = math.cos(angles[1])
119      c3 = math.cos(angles[0])
120      s1 = math.sin(angles[2])
121      s2 = math.sin(angles[1])
122      s3 = math.sin(angles[0])
```

Page 3 of 6

Figure 41. (con't) Myahrs.py

File - E:\NPS\Thesis\Thesis Writing\programs\myahrs.py

```
123
124     w = c1 * c2 * c3 - s1 * s2 * s3
125     x = s1 * s2 * c3 + c1 * c2 * s3
126     y = s1 * c2 * c3 + c1 * s2 * s3
127     z = c1 * s2 * c3 - s1 * c2 * s3
128
129     return w, x, y, z
130
131
132 def gravitycorrrpy(angles, araw):
133     """coordinate transformation to remove gravity
    acceleration using euler angles
134     Uses DCM from ME4703 notes"""
135
136     r2d = 180 / math.pi
137     phi = angles[0]/r2d; theta = angles[1]/r2d
138
139     sphi = math.sin(phi); cphi = math.cos(phi)
140     stheta = math.sin(theta); ctheta = math.cos(theta)
141
142     #coordinate transformation from ME4703 notes
143     axc = stheta
144     ayc = -sphi*ctheta
145     azc = -cphi*ctheta
146
147     ax = araw[0]-axc
148     ay = araw[1]-ayc
149     az = araw[2]-azci
150
151     return ax, ay, az
152
153
154 def gravitycorrquat(q, araw):
155     """coordinate transformation to remove gravity
    acceleration using quaternions
156     equation from http://www.varesano.net/blog/fabio/
    simple-gravity-compensation-9-dom-imus"""
157
158     # quat version seems to do a little better than euler
     angle
159
160     axc = 2 * (q[1] * q[3] - q[0] * q[2])
161     ayc = 2 * (q[0] * q[1] + q[2] * q[3])
162     azc = q[0] * q[0] - q[1] * q[1] - q[2] * q[2] + q[3]
    * q[3]
```

Page 4 of 6

Figure 41. (con't) Myahrs.py

File - E:\NPS\Thesis\Thesis Writing\programs\myahrs.py

```
163
164     ax = araw[0]+axc
165     ay = araw[1]+ayc
166     az = araw[2]+azc
167
168     return ax, ay, az
169
170
171 def servorange(cmd, maxrange):
172     """returns the value closest to zero to limit maximum
     requested servo
173     angle"""
174
175     if abs(cmd) == cmd:
176         return min(cmd, maxrange)
177     else:
178         return max(cmd, -maxrange)
179
180
181 def filter(current, previous, a= 0.3):
182     """implements EMA filter"""
183
184     avg = a * current + previous * (1 - a)
185
186     return avg
187
188
189 def barangle(theta2):
190     """Determine angle based on the bar system """
191     # https://www.researchgate.net/publication/
    290533574_An_efficient_position_solution_for_the_fourbar_
    linkage
192
193     # Define bar geometry
194     a = 3
195     b = 4.3
196     c = 2
197     d =3.835
198
199     # calculations
200     C = 1/(2*b*c)
201     D = C*(b*b + c*c)
202     F = b/c
203
204     theta2 = math.radians(90-theta2)
```

Page 5 of 6

Figure 41. (con't) Myahrs.py

File - E:\NPS\Thesis\Thesis Writing\programs\myahrs.py

```
205
206     r = d-a*math.cos(theta2)
207     s = a*math.sin(theta2)
208     E = D-C*(r*r + s*s)
209     if E > 1:
210         E = 1
211     if E < -1:
212         E = -1
213     delta = math.acos(E)
214     A = (F-E)/math.sin(delta)
215     B = s/r
216     theta3 = math.atan2(1-A*B,A+B)
217     theta4 = theta3 + delta
218
219     return 90-math.degrees(theta4)
220
```

Page 6 of 6

Figure 41. (con't) Myahrs.py

File - E:\NPS\Thesis\Thesis Writing\programs\mprlslib.py

```
""" Reads the adafruit MPRLS sensor connected to 'bus',
returns the pressure in PSI. Sensor address is cannot be
changed.
Adapted from adafruit code for circuit python and
Honeywell MPR data sheet."""

from pyb import I2C

def readpres(bus):
    """reads MPRLS sensor, returns calculated pressure"""
    i2c = I2C(bus, I2C.MASTER, addr=0x18, baudrate=115200)

    data = bytearray(5)
    cmd = bytearray(3)
    cmd[0] = 0xAA

    i2c.send(cmd,addr=0x18)
    i2c.recv(data, addr=0x18)

    rawpsi = data[1]<<16 | data[2]<<8 | data[3]
    psi = (rawpsi-0x19999A)*25
    psi /= 0xE66666 - 0x19999A

    return psi
```

Page 1 of 1

Figure 42. Mprlslib.py

File - E:\NPS\Thesis\Thesis Writing\programs\readdata.py

```
 1 import struct
 2
 3 # number of data fields
 4 numval = 21
 5
 6 # open and read binary file
 7 with open('/home/mbusta/Documents/Gitlab/missilecontrol/
   rolltest/log4.bin','rb') as binary_file:
 8     data = binary_file.read()
 9
10 # unpack the data
11 num = len(data)/4
12 data = struct.unpack('%df' % num , data)
13
14 # create file to write to
15 file = open('/home/mbusta/Documents/Gitlab/missilecontrol/
   rolltest/point5-1.csv', 'w+')
16
17 # title fields
18 file.write('no, q0, q1, q2, q3, ax, ay, az, gx, gy, gz, mx
   , my, mz, roll, pitch, yaw, filtroll, rollcmd, press, time
   \n')
19
20 # write data to file
21 n = int(len(data)/numval)
22 for i in range(1,n):
23     sample = data[((i-1)*numval):(i*numval)]
24     file.write('%s\n' % str(sample).strip('()'))
25 file.close()
26
27
28
```

Page 1 of 1

Figure 43. Readdata.py

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] F. Rydalch, “Missile demonstrator for counter UAV applications,” M.S. thesis, Dept. of Mechanical and Aerospace Engineering, NPS, Monterey, CA, USA, 2016.

[2] K. Grohe, “Design and development of a counter-swarm prototype air vehicle,” M.S. thesis, Dept. of Mechanical and Aerospace Engineering, NPS, Monterey, CA, USA, 2018.

[3] C. Brophy, interview, Mar. 2019.

[4] Apogee Components, “RockSim.” Accessed 5 May 2019. [Online]. Available: http://www.apogeerockets.com/Rocket_Software/RockSim

[5] onedrives.us, “Worm wheel gearbox - 5:1 ratio.” Accessed 5 May 2019. [Online]. Available: http://www.ondrivesus.com/worm-wheel-gearbox-51-ratio_p-14390.html

[6] Featherweight Altimeters, “Raven Altimeter.” Accessed 5 May 2019. [Online] Available: http://www.featherweightaltimeters.com/raven-altimeter.html

[7] PerfectFlite, “StratoLoggerCF Altimeter.” Accessed 5 May 2019. [Online]. Available: http://www.perfectflite.com/SLCF.html

[8] MissileWorks, “WRC+.” Accessed 5 May 2019. [Online]. Available: http://www.missileworks.com/wrc/

[9] Adafruit, “Adafruit MPRLS Ported Pressure Sensor Breakout - 0 to 25 PSI.” Accessed 5 May 2019. [Online]. Available: http://www.adafruit.com/product/3965

[10] Multitronix LLC, “TelemetryPro Tracking System.” Accessed 5 May 2019. [Online]. Available: http://www.multitronix.com

[11] MicroPython, “MicroPython pyboard v1.1.” Accessed 5 May 2019. [Online]. Available: http://store.micropython.org/product/PYBv1.1

[12] withrobot, “myAHRS_plus.” Accessed 5 May 2019. [Online]. Available: https://github.com/withrobot/myAHRS_plus

[13] VectorNav Technologies, “Quaternion Math.” Accessed 5 May 2019. [Online]. Available: http://www.vectornav.com/docs/default-source/documentation/vn-100-documentation/AN002.pdf?sfvrsn=19ee6b9_13

[14] “Dynamics of Motion,” class notes for ME4703: Missile Flight and Control, Dept of Mechanical and Aerospace Engineering, NPS, Monterey, CA, USA, winter 2018.

[15] VectorNav Technologies, “Gyroscope.” Accessed 5 May 2019. [Online]. Available: http://www.vectornav.com/support/library/gyroscope

[16] N.S. Nise, *Control Systems Engineering*, 6th ed. Delhi, IN: Wiley & Sons, 2011, pp. 172.

[17] Dassault Systems, “SolidWorks.” Accessed: 5 May 2019. [Online]. Available: http://www.solidworks.com

[18] TinySine, “WiFi Skin for PyBoard.” Accessed: 5 May 2019. [Online]. Available: http://www.tinyosshop.com/wifi-skin-for-pyboard

[19] K. Jones, interview, Jan. 2019.

[20] E. Constans, T. R. Chandrupatla, and H. Zhang, “An efficient position solution for the fourbar linkage,” *Int. J. Mechanism and Robotic Systems*, vol. 2, no. 3/4, pp. 365–373, Jan. 2015. [Online]. doi: 10.1504/IJMRS.2015.074122

[21] F. Varesano, “FreeIMU,” Varesano.net, Accessed: 5 May 2019. [Online]. Available: http://www.varesano.net/projects/hardware/FreeIMU

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California