2021-12

SYSTEM DESIGN AND INTEGRATION OF A RAPID RESPONSE PAYLOAD DELIVERY VEHICLE USING COMMERCIAL OFF-THE-SHELF COMPONEN

Decker, Kyle W.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SYSTEM DESIGN AND INTEGRATION OF A RAPID RESPONSE PAYLOAD DELIVERY VEHICLE USING COMMERCIAL OFF-THE-SHELF COMPONENTS

by

Kyle W. Decker

December 2021

Thesis Advisor: Joshua R. Codoni
Co-Advisor: Christopher M. Brophy

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
SYSTEM DESIGN AND INTEGRATION OF A RAPID RESPONSE PAYLOAD DELIVERY VEHICLE USING COMMERCIAL OFF-THE-SHELF COMPONENTS

5. FUNDING NUMBERS
R4Q04

6. AUTHOR(S) Kyle W. Decker

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)
ONR

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
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

This study involved the integration and implementation of multiple commercial off-the-shelf components into a rapid response payload delivery vehicle powered by a solid rocket motor. This work was motivated by the need to establish a proportional response to combat the presence of low-cost drone swarm technologies. Previous work was leveraged including flight data, system designs, and flight models to obtain the first objective of implementing a Raspberry Pi microprocessor, increasing the control loop response from 30 to 100Hz and improving the overall data acquisition attained from each flight. A proportional-derivative controller was then designed to successfully provide roll stabilization and heading during flight. Secondly, a nose-mounted camera system was implemented to serve as a lofted targeting hub to investigate the feasibility of tracking drone swarms and guiding submunitions. Multiple tests conclude that aerodynamic stabilization will be required to dampen the effects of the targeting hub oscillations during target acquisition. Lastly, this study allowed for the design, development, and evaluation of a mechanism for stable separation of rocket stages at high terminal velocities by incorporating a hybrid system of a mechanical release and carbon dioxide chamber pressurization.

14. SUBJECT TERMS
rocket, commercial off the shelf, COTS, system integration, Raspberry Pi, rapid response delivery vehicle, solid rocket motor, SRM, RPi, drone swarm, stage separation, RRPD-V, Rapid V

15. NUMBER OF PAGES
159

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.

SYSTEM DESIGN AND INTEGRATION OF A RAPID RESPONSE PAYLOAD DELIVERY VEHICLE USING COMMERCIAL OFF-THE-SHELF COMPONENTS

Kyle W. Decker
Lieutenant Commander, United States Navy
BSME, University of New Mexico, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2021

Approved by: Joshua R. Codoni
Advisor

Christopher M. Brophy
Co-Advisor

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

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This study involved the integration and implementation of multiple commercial off-the-shelf components into a rapid response payload delivery vehicle powered by a solid rocket motor. This work was motivated by the need to establish a proportional response to combat the presence of low-cost drone swarm technologies. Previous work was leveraged including flight data, system designs, and flight models to obtain the first objective of implementing a Raspberry Pi microprocessor, increasing the control loop response from 30 to 100Hz and improving the overall data acquisition attained from each flight. A proportional-derivative controller was then designed to successfully provide roll stabilization and heading during flight. Secondly, a nose-mounted camera system was implemented to serve as a lofted targeting hub to investigate the feasibility of tracking drone swarms and guiding submunitions. Multiple tests conclude that aerodynamic stabilization will be required to dampen the effects of the targeting hub oscillations during target acquisition. Lastly, this study allowed for the design, development, and evaluation of a mechanism for stable separation of rocket stages at high terminal velocities by incorporating a hybrid system of a mechanical release and carbon dioxide chamber pressurization.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. UNMANNED AERIAL VEHICLE SWARMS....1
B. CURRENT WORK....2
C. OBJECTIVES....11

II. DESIGN CHALLENGES....13
A. MODELING SYSTEM CONTROL....13
1. Roll....13
2. Control Torque....15
B. STAGE SEPARATION....17
C. TARGETING HUB....20

III. INITIAL ROCKET DEVELOPMENT....23
A. DEVELOPMENT....24
1. Control....24
2. Sled Coupling....27
3. Battlespace Imaging and UAS Tracking....30
B. RESULTS....31
1. Control....31
2. Sled Coupling....32
3. Video....33

IV. FLIGHT TESTING AND RESULTS....35
A. ROCKET 1....35
1. Development....35
2. Results....46
B. ROCKET 2....50
1. DEVELOPMENT....50
2. RESULTS....52
C. ROCKET 3....54
1. DEVELOPMENT....54
2. RESULTS....64
D. ROCKET 4....67
1. Development....67
2. Results....71
E. ROCKET 5....75
1. Development....75

2. Results 80

V. KEY DEVELOPMENTS 87
A. MODELING SYSTEM CONTROL 87
1. Roll Model 87
2. PD Controller 91
3. Model Error 93
B. SLED HOUSED ACTIVATION AND RELEASE DEVICE 95
1. Design 95
2. Control 100

VI. CONCLUSIONS 103
A. SUBSYSTEMS 103
1. Control 103
2. Sled Coupling 103
3. Video 104
B. FUTURE WORK 104
1. Control 104
2. Sled Coupling 105
3. Video 105

APPENDIX A. RASPBERRY PI SETUP 107
A. INITIAL SETUP 107
B. REMOTE CONNECTION 107
C. CONNECTING MATLAB AND SIMULINK 108
D. TESTING MATLAB INTERFACE 108
E. TESTING SIMULINK INTERFACE 109
F. DEPLOYING A STANDALONE MODEL FROM SIMULINK 110
G. CONNECTING THE BNO055 IMU 111
H. USING THE IMU 112
I. CONTROLLING A SERVO 113

APPENDIX B. BNOCAL.M 115

APPENDIX C. CALWRITE.M 121

APPENDIX D. UNCORRUPT.M 125

APPENDIX E. STOPDEPMAT.PY 127

LIST OF REFERENCES **129**

INITIAL DISTRIBUTION LIST **135**

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Raytheon Coyote Block I. Source: [12]. 4

Figure 2. Raytheon Coyote Block II. Source: [16]....4

Figure 3. Raytheon Coyote Block III Shipboard Launch. Source: [15]. 4

Figure 4. Eagle Attacking a Drone. Source: [19]. 5

Figure 5. Depiction of SRC Using EW Techniques from Mobile Sensors. Source: [20]....6

Figure 6. Depiction of Leonidas Using EMP to Counter UAS Swarms. Source: [23]. 7

Figure 7. Depiction of 40mm Grenade Canister to Deploy a Fouling Net. Source: [24]....8

Figure 8. NPS Drone Breaker Concept of Operations. Adapted from [32]. 9

Figure 9. NPS Drone Breaker. Adapted from [32]....10

Figure 10. Rocket Cross-Section....13

Figure 11. Closed Loop Transfer Function. 14

Figure 12. Control Fin Geometry. Source: [37]. 16

Figure 13. Distributed Lift Over a Control Fin. Adopted from [37]. 16

Figure 14. Parachute Bay Configuration. 17

Figure 15. Targeting Hub Under Drogue Drag. 20

Figure 16. Simply Supported Pendulum Nosecone Comparison. 21

Figure 17. Breakdown of Major RRPD-V Sections. Adapted from [32]. 23

Figure 18. Rocket 0 Assembled with the 2020 NPS Rocket Squad....24

Figure 19. Rocket 0 Avionics Bay. 25

Figure 20. Fin Servo Assembly. Adapted from [32]. 26

Figure 21. Rocket 0 in Flight....27

Figure 22. Amateur Rocketry Coupling. 28
Figure 23. Apogee Rockets CD3 High-Altitude CO2 Ejection System. Source: [42]. 29
Figure 24. PerfectFlite StratoLoggerCF. Adapted from [43]. 29
Figure 25. Raspberry Pi and V2 Camera. Adapted from [45]. 30
Figure 26. Measuring Rocket 0 Orientation with Video Data. 31
Figure 27. Reconstructed Rocket 0 Data. 32
Figure 28. Premature Separation of Sled and Nosecone. 33
Figure 29. Raspberry Pi Camera V2 EMI. 34
Figure 30. Bosch BNO055 9DOF IMU. Adapted from [49]. 36
Figure 31. Rocket 1 Simulink Model for Roll Control. 38
Figure 32. Triggered Timer. 40
Figure 33. Converting Degrees Input to PWM Signal Output 41
Figure 34. Data Logging in Flight from BNO055 42
Figure 35. Ground Testing Sled Coupling Separation with 8 Shear Pins. 43
Figure 36. Ground Testing Sled Coupling Separation with 6 Shear Pins. 43
Figure 37. Kate 2.0 Mounted for Nosecone Insertion 44
Figure 38. EMI Imposed While Testing. 45
Figure 39. V2 Camera with Ribbon Shielding and HQ Camera. 46
Figure 40. Rocket 1 Raw Heading and Demand Signal Data. 47
Figure 41. Rocket 1 Roll Response with Processed Heading and Demand Signal Data. 48
Figure 42. View From Rocket 1 Forward Camera in Flight. 49
Figure 43. Last Known Sighting of Rocket 1 Nosecone. 50
Figure 44. Rocket 2 on the Launch Rail. 51

Figure 45. Rocket 2 Airframe Failure in Flight.....52
Figure 46. Rocket 2 Control Results. 53
Figure 47. Rocket 2 Nosecone Following Impact with Ground.....54
Figure 48. Aluminum Coupling Stage Coupling. Source: [28].....55
Figure 49. BNO055 Heading Transition 55
Figure 50. Rocket 1 Control Algorithm Problematic Simulink Code.....56
Figure 51. Rocket 3 Control Algorithm with Unwrap Phase Shift. 57
Figure 52. Ramped Input with Phase Shifted Output.....57
Figure 53. Fin Cage Mock-Up Mounted on a Rotating Bearing.....58
Figure 54. Rocket 3 Control Code.....59
Figure 55. Sled Coupling Design. 60
Figure 56. Airframe Cutouts for Pin Interlock Engagement. 61
Figure 57. Initial Gear (Left), Gear with Cutout (Center), and Insert (Right) 61
Figure 58. Testing the Backup Drogue Parachute.....62
Figure 59. Caddx Lorris 4K Camera (left) and RunCam Split Mini 2 (right). Source: [53] (left) and [54] (right). 63
Figure 60. Rocket 3 Nosecone with PETG Camera Housing. 64
Figure 61. Reconstructing Rocket 3 Heading Information. 65
Figure 62. Rocket 3 Reconstructed Heading.....65
Figure 63. Rocket 3 Sled Main Parachute Opening Before a Hard Landing 66
Figure 64. Nosecone Camera View of Booster Deploying Main Parachute.....67
Figure 65. Raspberry Pi GPIO Pin Extender with Pulldown Resistor. 68
Figure 66. Damaged Sled Coupling Removed from Sled. 69
Figure 67. Polycarbonate Coupling Installed on Sled.....70
Figure 68. PETG with Carbon Nanotube Camera Housing. 71

Figure 69. Rocket 4 Response to 45 Degree Demand. 72
Figure 70. View of Nosecone Separating from Sled. 73
Figure 71. Raspberry Pi and Perfect Flight Data Compared. 74
Figure 72. Nosecone Video. 75
Figure 73. Dynamic Pressures of M3400 and N3301 for Rocket 5. 77
Figure 74. Designed Step Response with PD Controller. 78
Figure 75. Config.txt Added Code for Overclocking. Source: [58]. 78
Figure 76. Potentiometer and ADC Experimental Setup. 79
Figure 77. Testing of Drogue Containment Cap. 80
Figure 78. Rocket 5 Booster After Hard Landing. 81
Figure 79. Reconstructing Rocket 5 Heading Data. 82
Figure 80. Rocket 5 Reconstructed and Design Heading. 82
Figure 81. Rocket 5 Recalculated Response (ζ=0.99) with Heading. 83
Figure 82. Rocket 5 Sled After Impact. 84
Figure 83. Rocket 5 Nosecone After Impact. 85
Figure 84. Successive Peaks for Logarithmic Decrement. Source: [61]. 89
Figure 85. Rocket 4 Response with Points for Logarithmic Decrement. 89
Figure 86. Rocket 4 Model Compared to Recorded Data. 90
Figure 87. Roll Model with PD Controller 91
Figure 88. Simulink PID Tuner for Desired Response. 92
Figure 89. Implementing PD Controls in Deployed System. 93
Figure 90. Rocket 4 Modeled Heading Using Corrected Damping Coefficients. 94
Figure 91. Corrected Step Response Compared to Initial Design. 95
Figure 92. Separation Mechanism Exploded View 96
Figure 93. Sled Mounting Base Installed in Coupler. 97

Figure 94. Sled Coupling Base (Left) and Gear Titanium Insert (Right).....97
Figure 95. Static Load Testing of Coupling Release Mechanism. 98
Figure 96. Planetary Gears with Details (Left) and as Installed (Right). 99
Figure 97. SHARD Coupler. 99
Figure 98. SHARD Installed. 100
Figure 99. Raspberry Pi 4B with Sense HAT. 101
Figure 100. Sled Coupling Release Simulink Code. 101
Figure 101. Sense HAT Block Expanded. 102
Figure 102. Activation Block Expanded. 102
Figure 103. Release Block Expanded.....102

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Rocket 5 Gain Parameters.....92

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADC	Analog Digital Converter
AGL	Above Ground Level
APOE	Aerial Port of Embarkation
BVLOS	Beyond Visual Line of Sight
CONOPS	Concept of Operations
COTS	Commercial-off-the-shelf
ECCM	Electronic Counter-Countermeasures
EMP	Electromagnetic Pulse
EW	Electronic Warfare
FAA	Federal Aviation Administration
FWHM	Full Width at Half Maximum
GPS	Global Positioning System
HAT	Hardware Attached on Top
I2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
LOCUST	Low-Cost UAV Swarming Technology Line
LOS	of Sight
LSU	Louisiana State University
NPS	Naval Postgraduate School
OTH	Over the Horizon
PETG	Polyethylene Terephthalate Glycol Polylactic
PLA+	Acid +
PWM	Pulse Width Modulation
RPL	Rocket Propulsion Laboratory
RRPD-V	Rapid Response Payload Delivery Vehicle
SHARD	Sled Housed Activation and Release Device
UART	Universal Asynchronous Receiver
UAV	Transmitter Unmanned Aerial Vehicles
WWI	World War I

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

To begin, I would like to offer my heartfelt thanks to my thesis advisors for allowing me to entertain wild ideas on expensive systems. First to Dr. Codoni, I appreciated your unrelenting diligence in ensuring that success was realized in the ugliest of results. To Dr. Brophy, thank you for being a sounding board of reason when my ideas were too extreme or were unrealizable for the experiments at hand. Special thanks to the Department of the Navy, Office of Naval Research, Consortium for Robotics Unmanned Systems Education and Research at the Naval Postgraduate School for funding and amazing support.

I would like to thank Mr. Alexis Thoeny for the plethora of advice regarding high powered rocketry and the hours of building and wiring that ensued the discussions. To LT Allison “Shorty” Adamos, thank you for your devotion to achieving results by always rigorously testing components, often resulting in white smoke. A special thanks goes out to Vern Knowles for technical support regarding the programing of Kate and her pyrotechnics. Additionally, to the staff at Friends of Amateur Rocketry, thank you for providing the facilities to allow for all the testing we were able to perform.

Most importantly, I would like to thank my beautiful wife, Ariel, for being supportive through this study and for wrangling our four children. To Wyatt, Luke, Jack, and Virginia: I hope that this work inspires you to do great things.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Within the confines of the World, there exists foreign agents that are consistently developing cost-efficient solutions to test and counter the might of the United States' multi-billion-dollar military machine. As the technology ceiling grows, capabilities that used to belong to governments at the highest levels are now available to purchase at a local superstore. Drones with capabilities of adding attached weaponry are a household commodity to the local hobbyist and can be further developed with a handful of cheap electronics that can be assembled into jamming devices [1]. While these cheaply acquired technologies, mixed with foreign tactics and techniques, can be a hinderance to the mission of the United States; most do not deserve the response of a million-dollar kinetic kill or the deployment of additional personnel in harm's way. For this reason, the rethinking of the United States' response to certain threats is required; a proportional response is necessary.

The quickly evolving asymmetric threats that are being developed using commercial-off-the-shelf (COTS) hardware are becoming cheaper and are easier to integrate than ever. The rise of popularity in hobbyist activities and military technology advances has pushed industry to develop smaller and lighter form-factors, batteries, and computational hardware of drones, quad-copters, and other unmanned aerial systems (UAS) [2] allowing somewhat irregulated use by anyone who can obtain the technology. As an example, Louisiana State University (LSU) recently posted instructions on how to build a Global Positioning System (GPS) guided agriculture sprayer drone capable of carrying 5L of fluid for 10 minutes at a cost of under $2000 [3]. Expanding this model into a nefarious concept is not out of the impossible, nor is it unimaginable. The obtainability of an autonomous drone system to deliver an explosive device over the horizon (OTH) toward U.S. citizens or troops is now a very economical reality.

A. UNMANNED AERIAL VEHICLE SWARMS

Unmanned aerial vehicles (UAV) are generally seen as an aircraft that can fly by remote or with autonomous features utilizing UASs such as GPS, control systems, and ground architectures [4]. The first UAV was developed during World War I (WWI) by an

American team lead by Charles F. Kettering [5]. His concept was a flying torpedo that would maintain its heading using a gyro, altitude through a sensitive barometer, and distance by counting revolutions of the engine. Nicknamed the "Kettering Bug," it would fly over a target and release its wings to torpedo down to earth. WWI ended before the use of the Kettering Bug in combat but the early model of the first UAS introduced technology that would allow for expanding modern technologies to the systems we have today [5]. Modern military UASs such as the United States Air Force MQ-9 Reaper have advanced systems that can carry payloads of 17,125 N (3,850 lbs) for over 27 hours at 15,240 m (50,000 ft) [6] but have a jaw-dropping cost of over $32 million [7]. While the MQ-9 price is bound by the red tape of modern bureaucracy, private companies such as Applied Aeronautics have developed cost efficient solutions that can fly beyond visual line of sight (BVLOS) using autonomous systems with full approval of the Federal Aviation Administration (FAA). At a base price of under $2,000, this system has the ability to carry 44.5 N (10 lbs) for over 4 hours to a destination of choice by the buyer [8].

Expanding the previous example further into a system of several drones all interconnected with a goal of destruction is obtainable. In early 2018, a Russian base in Syria was attacked by several unmanned aerial vehicles of which were constructed using COTS electronics and crude flight control systems [9]. While the inventors/users of the UAVs are unknown, the use of swarm tactics by amateur users comes to play and a new threat emerges. Even more recently, Israel effectively used drone swarms in combat to target and strike militants in Gaza [10]. While the Israel military use does not strictly speak to COTS, it does provide context to the effectiveness of using small drone swarms to strike targets. The strikes that Israel conducted were so effective that they are now investing into more drones to allow for autonomous strikes OTH [10].

B. CURRENT WORK

As technology develops and UAV swarm tactics become a reality, the need to counter such capabilities arises. The threat that UAS provide was made readily apparent when a piloted gyrocopter landed on the front lawn of the White House in 2015. The Air Defense Identification Zone (ADIZ) that surrounds Washington is accompanied with

sensors from the Joint Land Attack Cruise Missile Defense Elevated Netted Sensor System (JLENS) and several anti-air and anti-missile systems capable of taking out threats with highly kinetic weapons. The flaw identified with the system is that it will not sense or attack a small vehicle flying at low altitudes, bringing the question on drones or drone swarms into reality. In the case of the U.S. Capitol, ground-based defense mechanisms consist of the AIM-120 AMRAAM, FIM-92 Stinger, and rooftop mounted .50 caliber machine guns, all of which would evaporate a drone, leaving behind a large mess, possible uncontrolled disbursement of a lethal payload, and accompanying bill to attend to afterwards [11]. The question arises with this style of defense: What happens if there are several small aerial attacks?

Several companies and governments alike have developed solutions that provide feasible mechanisms to eliminate the threat of UAS swarms with cost efficient solutions. One such system that has been selected by the United States Army for use is the Coyote UAS by Raytheon Missiles and Defense [12] as shown in Figure 1. The system methodology is a drone versus drone combat architecture where the Coyote is deployed, guided to a drone, and a small explosive annihilates the threat. While small and cost effective, at approximately $15,000 each without the warhead, the system requires its own Ku band Radar, generator, and launch device in the form of a 6x6 Army tactical truck providing eight total counter UAV drones [13]. Issues with this system are observed with the limitations in capability as this is a one-on-one kamikaze-style system that must be cued and remain within line of sight (LOS) with its organic ground station sensors. Operational capability is paused as soon as there are greater than 8 drones in the swarm, assuming no quick reload capability, or if the swarm travels behind a mountain or structure. In attempts to defeat more difficult targets, Raytheon has also developed the Coyote 2, as seen in Figure 2, which is more of a missile-on-drone technology powered by a small turbine engine [14] and the Coyote Block 3, as seen in Figure 3, for United States Navy application as part of the Low Cost UAV Swarming Technology (LOCUST) Program [15]. The same issues remain with Coyote 2; however, with added cost incurred by the propulsion system and advanced control. Coyote Block 3 is much like the initial Coyote system, built for shipboard applications [15].

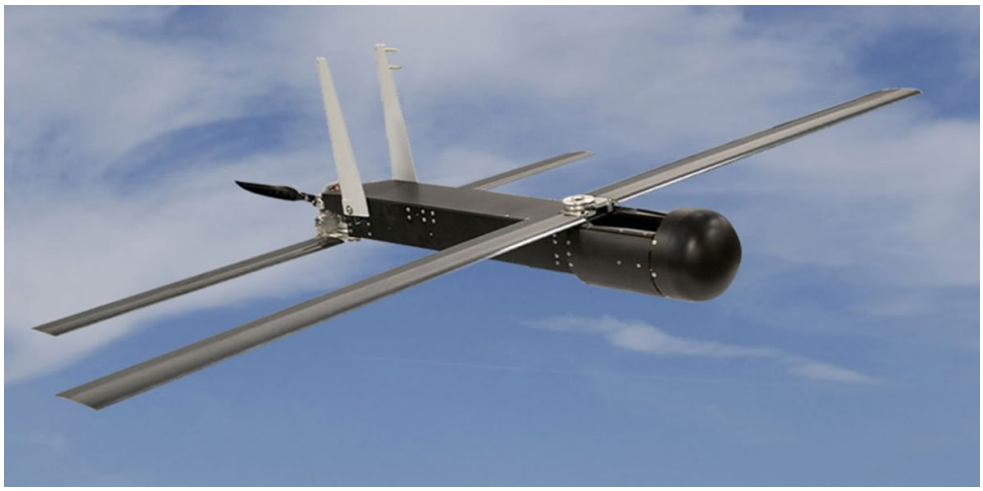

Figure 1. Raytheon Coyote Block I. Source: [12].

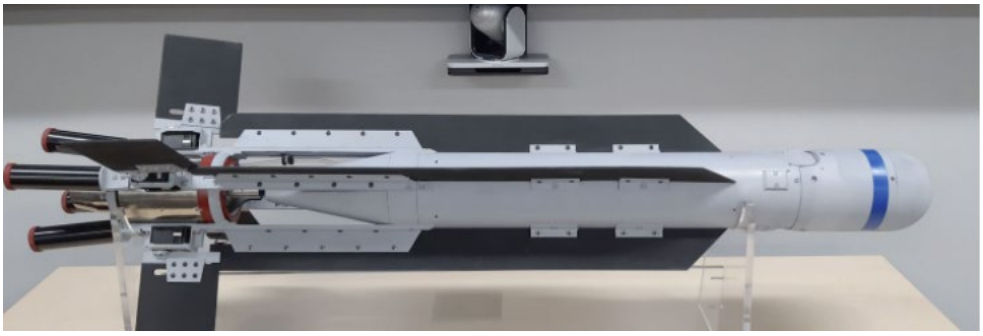

Figure 2. Raytheon Coyote Block II. Source: [16].

Figure 3. Raytheon Coyote Block III Shipboard Launch. Source: [15].

One less technical approach that has been attempted by the Dutch company Guard from Above is to use birds of prey to intercept drones. The company has built a partnership with the Dutch National Police and Military to train birds of prey to intercept and take down drones. While the concept appears cheap and environmentally friendly up front, the need for several other items such as a handler, housing environment, training areas, and food all add on cost that may not be apparent [17]. The Dutch Police stopped the program in late 2017 due to the expense and unpredictability of the birds [18]. The same concept is being entertained by the French Air Force; however, by breeding Golden Eagles for the single purpose of taking down drone swarms [19] as shown in Figure 4. Additionally, the United States Air Force is keying in on how Falcon's target their prey to help train other systems on how to approach targeting small and maneuverable targets [18]. Although these ideas are novel, they do not provide a solid defense against a motivated attacker with a swarm of drones.

Figure 4. Eagle Attacking a Drone. Source: [19].

Another company by the name of SRC has taken an approach to defeat drone swarms through the use of electronic warfare (EW). The Silent Archer system uses radars,

electro-optical, and infrared detectors to classify a drone, or a swarm of drones, and proceeds to use EW effects to render the incoming threats useless [20] as seen in Figure 5. While SRC guards the method of EW employed, one example describes disrupting a communications link with the drone and user to cause the drone to land or return to base. While the technology has been fielded by the United States Army and Airforce, this system only allows for detection within the field view of the sensors and provides only moments to react [20]. The approach to use EW to counter drone swarms is valid, but a proper electronic counter-countermeasure (ECCM) could defeat the EW effect with known parameters of the defense system[21].

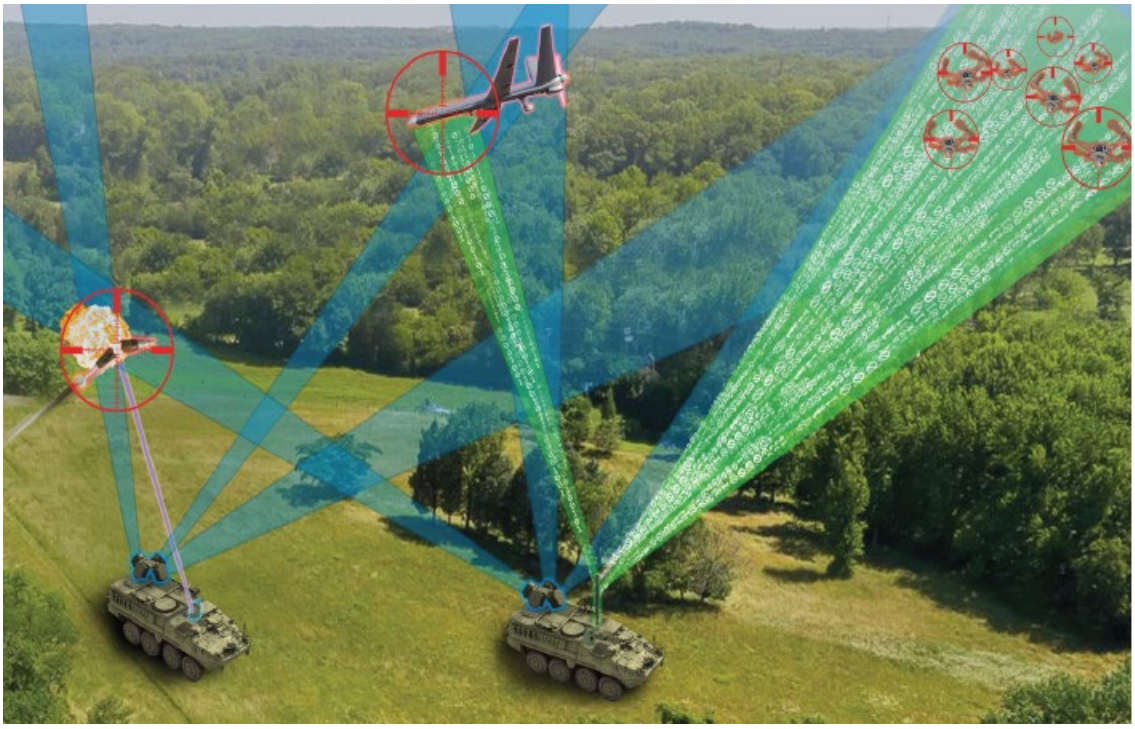

Figure 5. Depiction of SRC Using EW Techniques from Mobile Sensors. Source: [20].

Northrup Grumman has several products available for drone and drone swarm defense. One of the ventures that is notable is their partnership with Epirus for the creation of an Electromagnetic Pulse (EMP) generator using solid-state semiconductors [22] codenamed Leonidas [23] as shown in Figure 6. The semiconductors create a steerable

High-Power Microwave that target specific threats that are identified. Additionally, Leonidas integrates with the United States Army's Forward Area Air Defense Command and Control (FAAD C2) system for a layered defense approach [23]. As with the previous systems that are mentioned, Leonidas is a short-ranged defense mechanism to counter drone swarms. While the idea is fundamentally sound, questions about the safety of the device around other electronics arise; potentially leading to the need to harden all other nearby systems from EMP effects. Cost also becomes a question as the system has high-end technology in use.

Figure 6. Depiction of Leonidas Using EMP to Counter UAS Swarms. Source: [23].

The United States Army is also testing simpler techniques to counter drone swarms such as that of a net to foul the incoming UAS rotors [24]. While a net gun has a fairly short range in the realm of 6 to 15 m (20 to 50 ft) [25], this product is contained in a grenade sized container capable of being launched from a 40mm launcher. The "Scalable Effects Net Warhead," as shown in Figure 7, is capable of being launched to a maximum effective range to a point target of 150 m (500 ft), assuming the use of the M320 grenade launcher [26], greatly increasing the range of a net firing system. While this is a concept that is worth

commending, the maneuverability of drones with object avoidance could potentially avoid the incoming grenade.

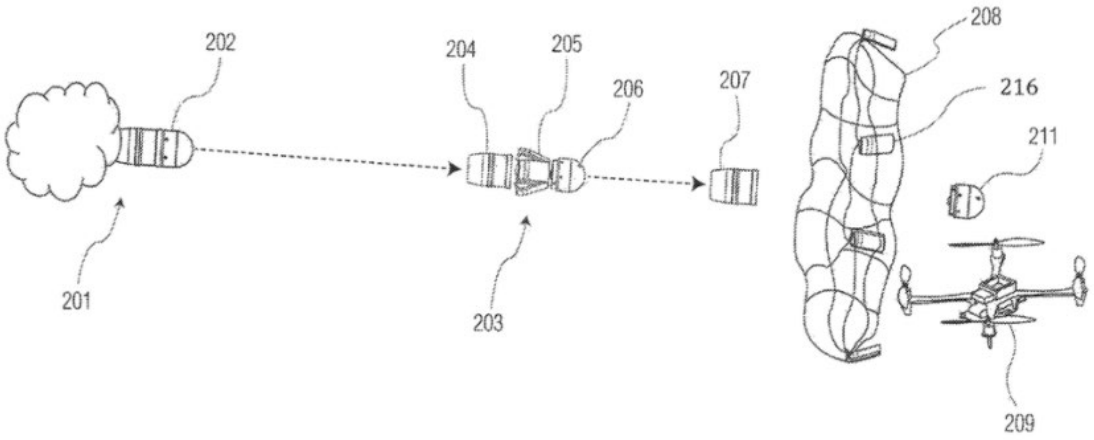

Figure 7. Depiction of 40mm Grenade Canister to Deploy a Fouling Net. Source: [24].

The previous system is worth investigating further, as a net launcher is simple and non-kinetic, leaving it as a viable option without having to worry about collateral damage associated with the countermeasure. The question arises on how to enable the systems to have greater range so that the last line of defense is not the only line of defense against swarms of UAS? Natural conclusions come to dropping a device from an airplane or launching from a forward deployed unit. Raytheon's Coyote program has already pursued the concepts of releasing drones from fighter planes, cargo planes [12], and ships[15] in what are successful ventures; however, the aforementioned vehicles for delivery require notification and do not allow for a timely insertion to truly defend against a surging drone swarm. How can a ship or ground station protect itself from an incoming drone swarm threat without relying on close distance encounters or properly pre-deployed forward stationed units?

Previous research conducted at the Naval Postgraduate School (NPS) Rocket Propulsion Laboratory (RPL) has developed a concept that uses solid rocket motor (SRM) propulsion to guide projectiles to a target with the intent on countering UAS technology as

shown in Figure 8. SRMs have benefits to their use as the technology is mature, the burn profiles are predictable, and the propellants have long shelf lives when properly stored [27]. Work performed by Kai Grohe designed a SRM delivery vehicle that focused on the feasibility of a low-cost system for a counter UAV payload [28]. Keith Lobo designed a submunition using low-cost techniques to counter UAS swarms using nets from close range [29]. Building off of the previous two designs, Matthew Busta [30] and Robert Thyberg [31] were able to employ delivery vehicle control techniques and implement the possibility of countering multiple drones with one SRM delivery vehicle. Additionally, the NPS ME4704 Final Report from the Summer of 2020 developed a complete system called the "Drone Breaker" that leverages the previously noted theses [32]. The potential to counter UAS swarms with a single munition that offers rapid response and non-kinetic effects is on the horizon.

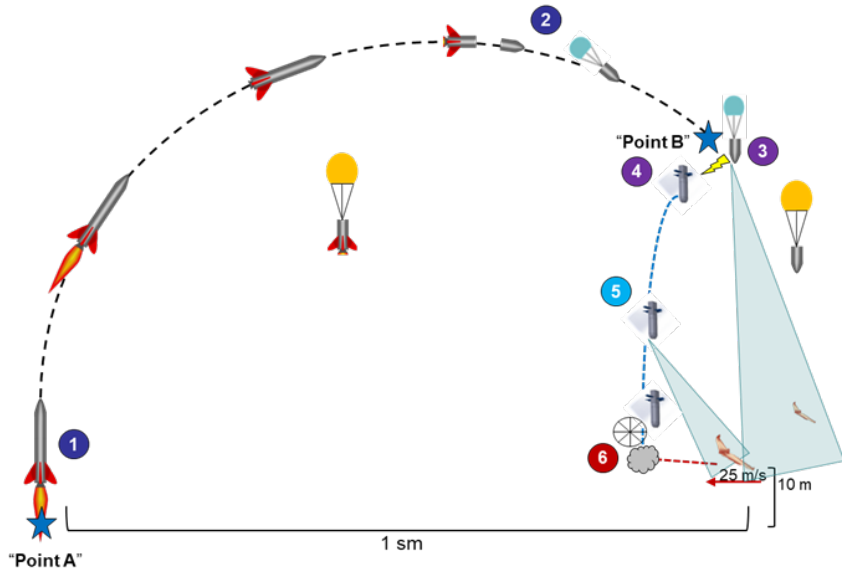

Figure 8. NPS Drone Breaker Concept of Operations. Adapted from [32].

The concept of operations (CONOPS) shown in Figure 8 utilizes the vehicle, as seen in Figure 9, to counter UAV swarms. The vehicle is given a Global Positioning System (GPS) coordinate approximately 1 statute mile in separation from the launch site. An SRM

then delivers the vehicle to the GPS position within a few seconds of notification. After apogee, the vehicle separates, and the forward section (sled) looks downward with an imaging system designed to track the UAVs with integrated neural networking algorithms. Upon detection, up to four bomblets are released and are guided by the sled's computer to their individual targets [32]. Once the targets are in range, a net and throw weight kill mechanism [29] is released to entangle and defeat all drones under the net's area of coverage [32].

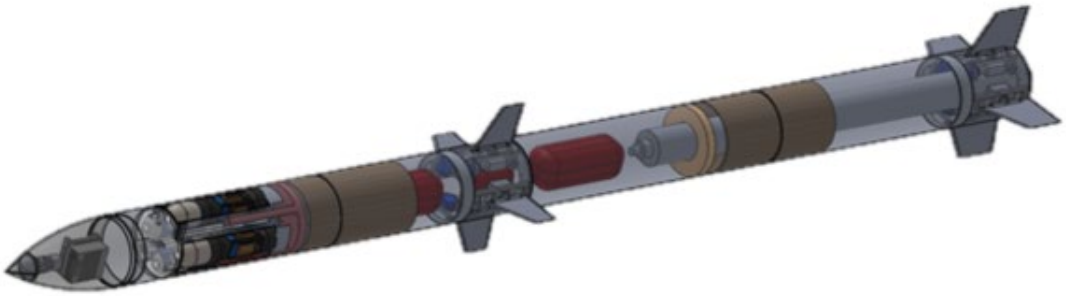

Figure 9. NPS Drone Breaker. Adapted from [32].

While the Drone Breaker is a novel concept, the integration of the technology required to achieve the goal of placing submunitions on their targets requires improvement on the rockets flown by Busta and Thyberg. Busta's rocket was limited by sensor capacity at achieving 30Hz refresh rates from the inertial measurement unit (IMU) sensor data. This limitation proved to be detrimental in being able to properly control the flight of the rocket. Additionally, the stage coupling mechanisms used are unreliable as the system relies on Nylon shear-pins to hold the rocket together and then separate when commanded by the release of CO_2. The lack of quality control in shear pins and uncertainty of the effect of flight dynamics on the shear pins is believed to have led to previous rocket failure [30]. Furthermore, the feasibility of using a nose-fixed camera system is questionable as previous launches have indicated that nosecone stability and image quality may not be sufficient to guide the submunitions to their targets [31].

C. OBJECTIVES

Building off previous work performed at the RPL, a Rapid Response Payload Delivery Vehicle (RRPD-V) with SRM propulsion will be created with the intent of delivering a payload to counter UAS swarms. In order to build the RRPD-V to meet industry standards, COTS components will be utilized to keep cost as low as possible while still maximizing performance. The objectives of this work are as follows:

1. Integrate COTS components to a guided SRM system and provide adequate computational bandwidth and control speeds/rates for sub-sonic guided flight architecture.
2. Determine the feasibility of utilizing a nose mounted video camera system to detect and track UAS swarms.
3. Develop and integrate a reliable stage separation mechanism using COTS components.

THIS PAGE INTENTIONALLY LEFT BLANK

II. DESIGN CHALLENGES

A. MODELING SYSTEM CONTROL

1. Roll

To understand the dynamics of a rocket's system in flight, the equations of motion must first be understood. As observed in Figure 10, a cross-sectional view is required to develop the equations of motion about a cylinder on its primary axis. The roll of a rocket about its z-axis can be modeled with a clockwise orientation of positive. The forces acting on the roll of the rocket must then be accounted for [33]. Only two external forces yield any force on the rocket's body with respect to roll; the force of drag from the fixed fins resisting the movement and the desired force from the control fins to create the movement.

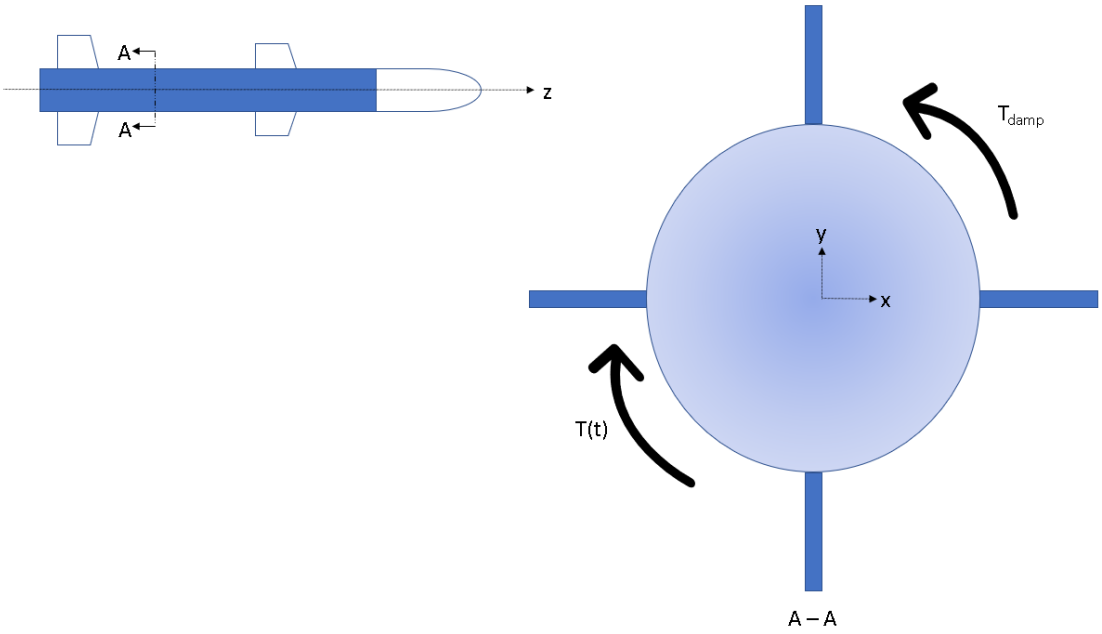

Figure 10. Rocket Cross-Section

Summing the torques of the system yields:

$$\sum T = I_{zz}\ddot{\theta} = T(t) - T_{damp} \tag{1}$$

Where I_{zz} is the moment of inertia about the primary axis of a cylinder, $\ddot{\theta}$ is the angular acceleration of the rocket, $T(t)$ is the torque demanded from the control fins, and the torque due to counter-torque of the fixed control fins is T_{damp}. The torque due to drag can be written in terms of d, dampening coefficient, and $\dot{\theta}$, the angular velocity, are shown as follows:

$$T_{damp} = d\dot{\theta} \tag{2}$$

Substituting Equation 2 into Equation 1 yields the following:

$$I_{zz}\ddot{\theta} + d\dot{\theta} = T(t) \tag{3}$$

Taking the Laplace Transform of Equation 3, with all initial conditions at zero, yields the following transfer function in the frequency domain:

$$\frac{\theta(s)}{T(s)} = \frac{1}{I_{zz}s^2 + ds} \tag{4}$$

Equation 4 represents the open loop transfer function that characterizes the roll of the rocket model. In order to obtain useful information out of the model and to simulate the feedback received by sensors, the system needs to be closed by providing a feedback loop [33] as shown in Figure 11.

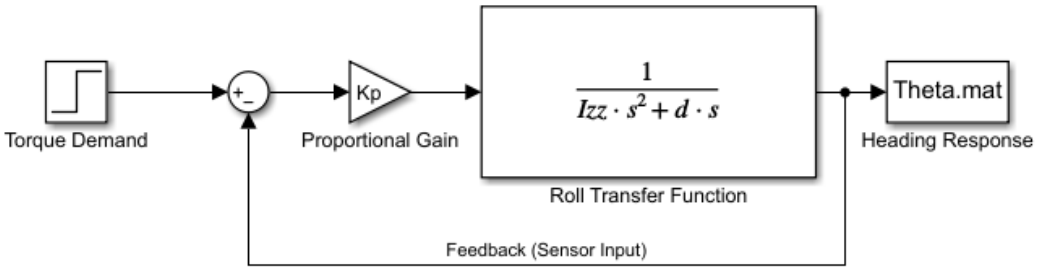

Figure 11. Closed Loop Transfer Function.

With the feedback loop inserted, the transfer function transforms into the following:

$$\frac{\theta(s)}{T(s)} = \frac{K_p}{I_{zz}s^2 + ds + K_p} \tag{5}$$

Equation 5 represents the rocket's roll transfer function in the form of a generalized 2nd order system whose parameters can be designed based on the desire of the controller.

2. Control Torque

Torque of the fins on the body of the rocket change as a function of rocket speed, air density, and angle of attack of the control fin. To properly define what the input to the roll transfer function is, the factors that make up the torque signal must be analyzed. The torque due to a single fin is shown as in Equation (6) where L is the Force due to lift on the control fin and c is the length of the effective moment arm on the control fin about the center of the rocket body.

$$T = Lc \tag{6}$$

The lift of the fin is described in Equation (7), where C_L is the coefficient of lift, q is dynamic pressure, and S is the surface area of the control fin [34].

$$L = C_L qS \tag{7}$$

The coefficient of lift can be calculated using flat plate theory with small angle approximations as shown in Equation (8) where α is the angle of attack in radians [35].

$$C_L = 2\pi\alpha \tag{8}$$

Dynamic pressure can be calculated as shown in Equation (9), where ρ is the atmospheric density, and V is the velocity of the rocket [34].

$$q = \frac{1}{2}\rho V^2 \tag{9}$$

The length of the moment arm from the fin to the rocket centerline will be calculated using relationships of distributed loads on static structural members [36]. The fin design, as shown in Figure 12, has two main sections: a rectangular section and a triangular section whose lift moment arm can be described as a function of the distributed lift on the control fin as shown in Figure 13.

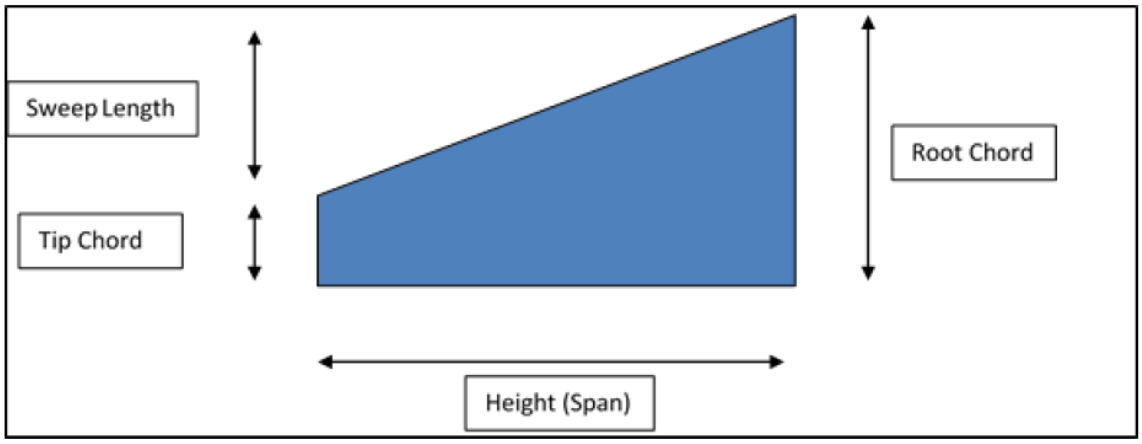

Figure 12. Control Fin Geometry. Source: [37].

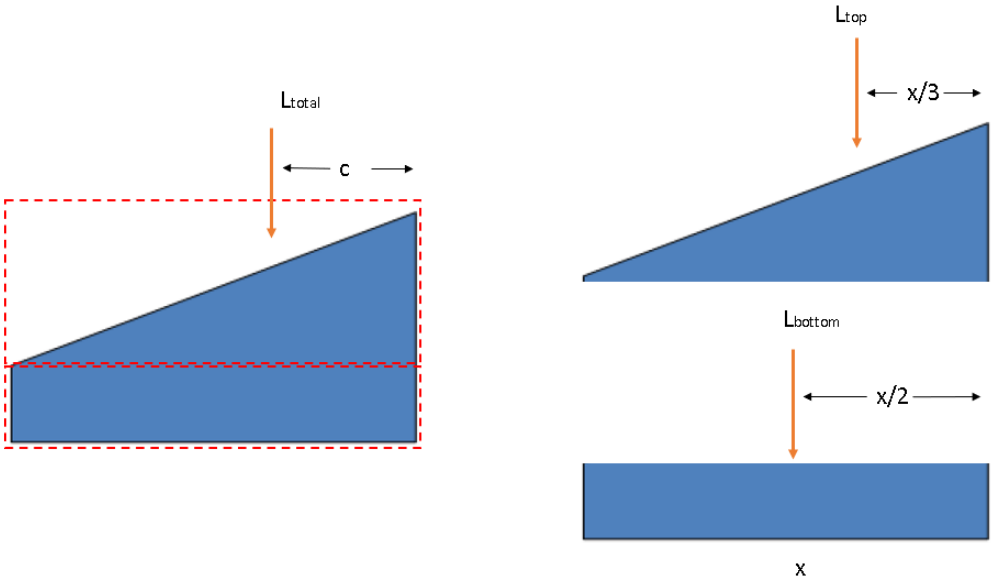

Figure 13. Distributed Lift Over a Control Fin. Adapted from [37].

As shown in Equation (7), the force of lift is directly proportional to the surface area of the control fin. After summing the moments, an effective moment arm of which the torque can be calculated can be determined using the geometry measured from the control fin [36].

$$L_{total}c = L_{top}\frac{x}{3} + L_{bottop}\frac{x}{2} \tag{10}$$

The length of the effective moment arm about the fin is then added to the radius of the rocket airframe to yield the total moment arm for a fin applying a force on the rocket as 16.637 cm (6.55 in).

Keeping in mind that there are two fins providing additive torque to the rocket frame, the torque can be calculated at any given velocity and altitude with Equation (11).

$$T = 2\pi\alpha\rho v^2 sc \tag{11}$$

The derived relationships will be used to determine the how a control fin's deflection will affect the flight path of the rocket at a given altitude and velocity.

B. STAGE SEPARATION

A key feature of a rocket's capability is that of stage separation, parachute release and recovery. Currently, nylon shear pins with a nominal shear force of 110 N (25lbf) each are used for stage separation; however, lack of quality assurance makes this method unreliable. The pins are sheared when a CO_2 canister releases its contents into the enclosed volume of a bay allowing for rapid pressurization within the bay. An example of the setup is shown in Figure 14. Further details on the operation of this parachute bay separation are offered in Section III.2.

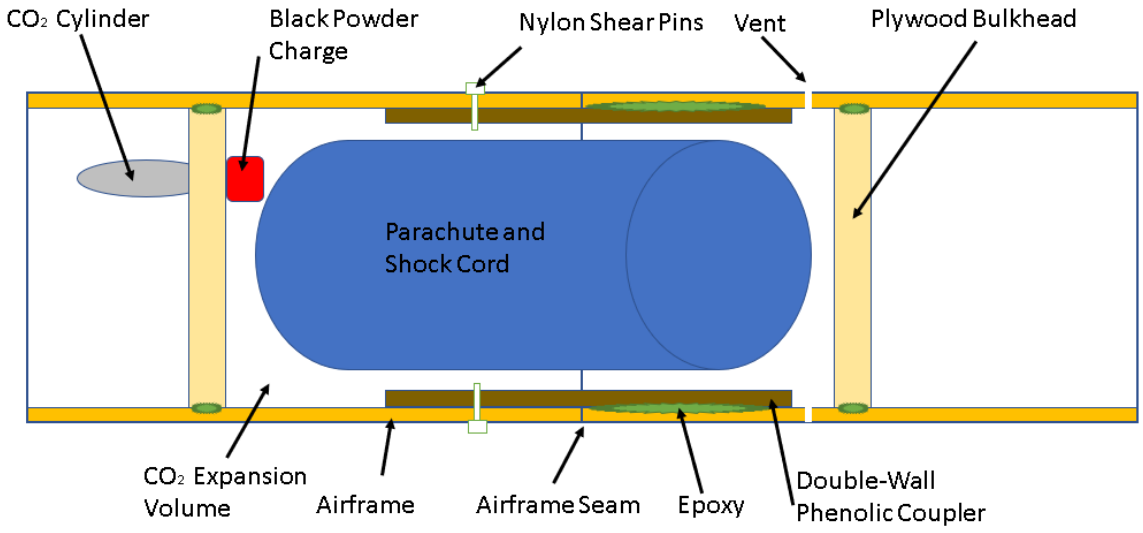

Figure 14. Parachute Bay Configuration.

The challenges involved with this type of separation coupling is that vent holes are required to allow for pressure to equalize as the rocket climbs out in altitude. Without the vent holes, the rapid increase in altitude will cause a differential pressure across the airframe with a higher pressure inside the parachute bay and a lower pressure outside the airframe. For example, a flight to 1000 m (3280 ft) will result in a differential pressure of 11450 Pa (1.66 psia); which, when applied across the surface area of a 17.8 cm (7.5 in) bulkhead yields a force of 327.7 N (73.67 lbf) acting on the shear pins. The differential pressure at higher altitudes can easily cause a premature release of the parachute.

While vent holes seem like the logical answer, the issue becomes that the vent holes need to be sized to allow for equalizing pressure for altitude but small enough to allow for pressurization from the CO_2 expansion. Additional vent holes are generally added as the parachute inside the expansion volume can easily block the vent holes leading to over pressurization. Additionally, sufficient shear pins must be inserted into the airframe to allow for enough strength to hold the rocket together during flight and drogue deployment but too many shear pins will not allow separation when commanded from the CO_2 cylinder release. Rules of thumb exist in the amateur rocketry community that claim a 0.635 cm (1/4 in) vent hole is required for every 1639 cm^3 (100 in^3) of volume in the bay [38]. While the rules of thumb are good for amateur applications they only speak to the proper venting of the bay. Adding shear pins to the system add additional error as they are affected by manufacturer quality, temperature, age, and rigidity of the rocket.

The ideal gas law can be used to establish a baseline of the force that will be provided against a bulkhead to allow for separation [31] where P is Pressure, V is volume, n is the number of moles, R is the universal gas constant, and T is the temperature.

$$PV = nRT \tag{12}$$

Assuming the right side of the equation remains constant and solving for ratios between two volumes can yield a relationship to understand the pressure inside the parachute bay.

$$P_{Bay} = P_{CO_2} \frac{V_{CO_2}}{V_{Bay}} \tag{13}$$

Using the volume and pressure of a 38g CO_2 cylinder [39] and the volume of a 19.05 cm (7.5 in) diameter airframe with 22.86 cm (9 in) in depth yields a pressure in the bay of 49.5kPa (7.18psia) acting on the surface area of the bulkhead, assuming instantaneous ejection of 38g of CO_2 prior to venting. With the surface area of the bulkhead having a diameter equal to the airframe diameter minus the coupler material, the force can be calculated where F is the force on the bulkhead, P is the Pressure in the volume, and A is the surface area of the bulkhead.

$$F = PA \tag{14}$$

With 1335 N (300 lbf) of force on the bulkhead the use of 6 shear pins yields a 100% margin of safety to allow for separation. The margin is necessary to take into account the effect of pressure being released from the vent holes which is generally believed to account for half the force that would otherwise be pushing on the bulkhead [40]. While the calculations require theoretical results, ground testing is always required to ensure the shear pins will shear when required.

The other issue not discussed is the force transmitted on the shear pins when the drogue parachute deploys. As seen from data obtained in previous launches, the acceleration observed from a 40kg nosecone when the drogue parachute deploys is approximately 5G's. The force can then be calculated where m is the mass and a is the acceleration.

$$F = ma \tag{15}$$

With the conditions as stated, the Force felt on the nosecone due to the acceleration of the drogue is 890N (200 lbf), which would undoubtedly cause the 6 shear pins to fail. An option exists to use smaller vent holes, excepting some of the differential pressure seen due to altitude, while adding more shear pins and doubling up CO_2 charges to create a larger separation; however, the integrity of the airframe would need to be further examined prior to implementation. The careful balancing of having enough shear pins to allow for the shock of the drogue parachute and few enough to allow for separation when desired describes why a new solution is required to provide reliable separation when commanded.

C. TARGETING HUB

Obtaining stable video feed from the Nosecone of the rocket is necessary when trying to image UAS swarms and provide guidance data to a submunition. The conceptual design allows for the Nosecone to look downwards at the battlespace and image everything within the video field of view. Previous work done by Thyberg [31] provided information concerning the viewability of a bomblet, but the stability of the system is of greater concern as it was not well understood if a Nosecone descending under a drogue was stable enough to allow for the imagery to be useful. Thyberg concluded that the video quality from a GoPro could provide the resolution necessary for observing a bomblet at approximately 125 m (410 ft). This line of effort to understand the stability of the system will coincide with simultaneous attempts to increase the resolution of the video recording devices. The conceptual operation is shown in Figure 15. Initial concerns arise that the system will act much like a simply supported pendulum, as shown in Figure 16, and disturbances will cause the nosecone to sway in a randomized harmonic motion. Best efforts will be utilized to understand if the nosecone, under the light drag of a drogue parachute, is sufficient to eliminate the harmonic effects of a suspended mass in flight.

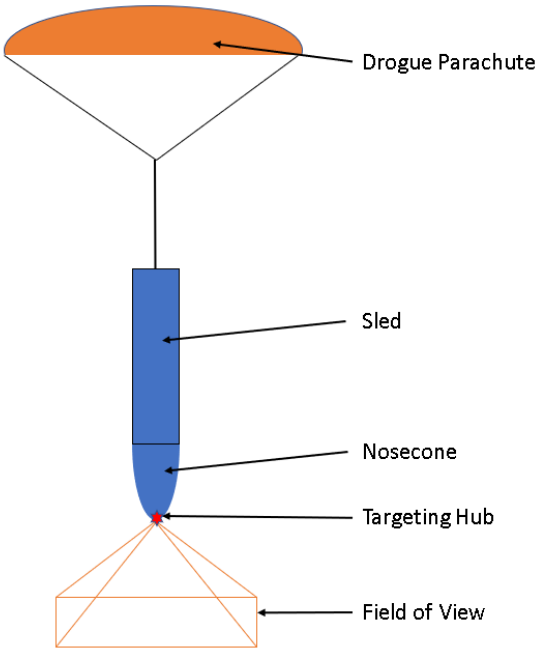

Figure 15. Targeting Hub Under Drogue Drag.

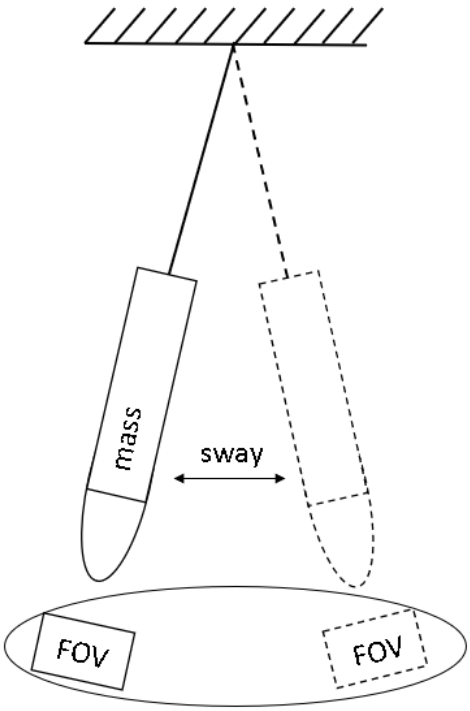

Figure 16. Simply Supported Pendulum Nosecone Comparison.

THIS PAGE INTENTIONALLY LEFT BLANK

III. INITIAL ROCKET DEVELOPMENT

Development of the RRPD-V was done with intermittent testing performed through six launches over the course of a year: Rocket 0 being the first. As shown in Figure 17, the major sections of the rocket fall into two major systems: the Sled and the Booster. The Sled contains the nosecone, a main parachute, a drogue parachute, camera systems, a downlink transmitter, and the Sled Coupling. The Booster contained the SRM, avionics bay, a main parachute, a drogue parachute, and the control fins.

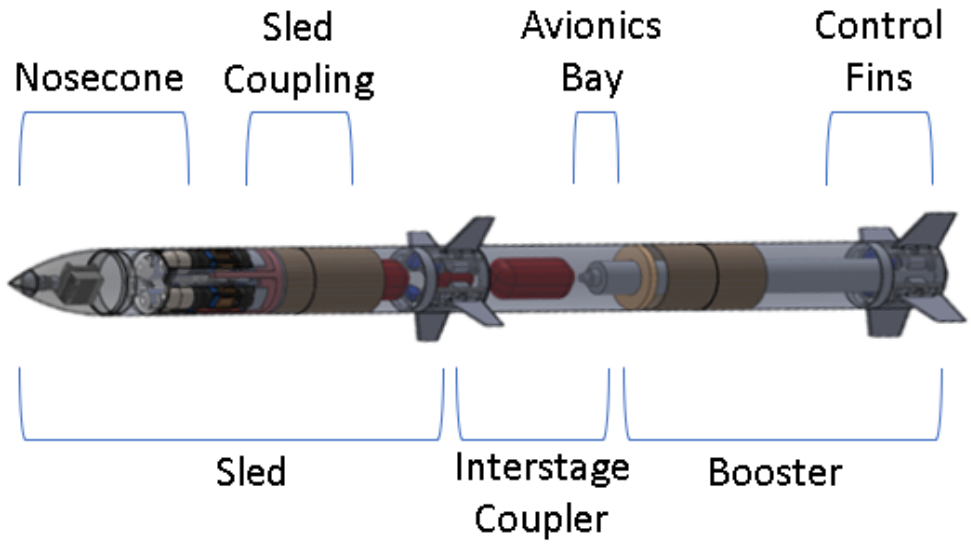

Figure 17. Breakdown of Major RRPD-V Sections. Adapted from [32].

The inaugural rocket served as an introduction to high powered rocketry and was designed following the recommendations from the ME4704 Tactical Missile Design report from 2020 [32] as shown in Figure 17. Deviations in the system design occurred to allow for rapid prototyping of the rocket as it was a complete redesign of previous rocket systems. While the dimensions of the system and rough weight was maintained, the steerable canards and bomblet payload were not included as the goals were to gain camera footage from the nosecone during freefall and to gather data from a gyro-based roll control system

from previous rocket designs. Final assembly is shown in Figure 18. Details on subsystems of interest follow.

Figure 18. Rocket 0 Assembled with the 2020 NPS Rocket Squad

A. DEVELOPMENT

1. Control

The control system for this rocket was inherited from a previous project and served as a starting point to understand how to properly control the RRPD-V in flight. The Avionics Bay contained the Bosch BNO055 IMU and the Arduino Uno microcontroller as shown in Figure 19.

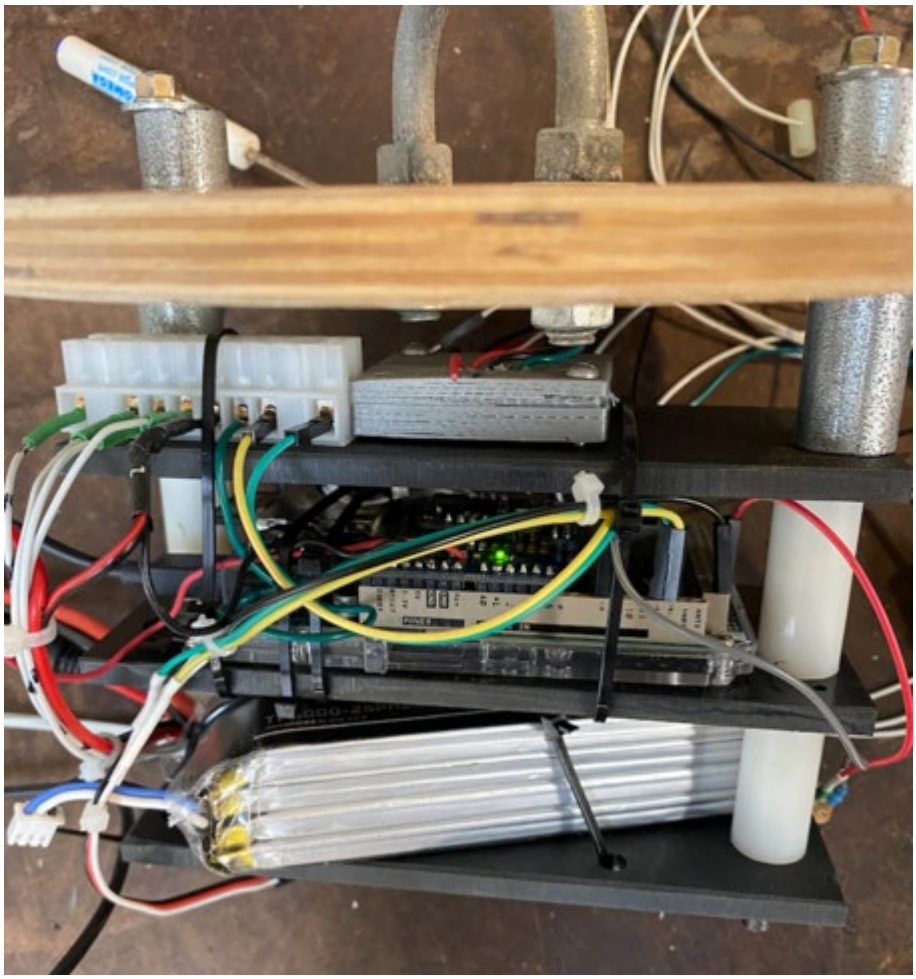

Figure 19. Rocket 0 Avionics Bay.

The IMU is used to sense the angular velocity of the rocket via the onboard gyro and the signal is used in an algorithm to control roll on the Arduino. The algorithm logic was setup as a simple control to counter any change in angular velocity with an opposing +/-3-degree deflection of four control fins. The battery and wiring provide for power and Pulse Width Modulation (PWM) signal distribution to the tail mounted fin servo assembly as modeled in Figure 20.

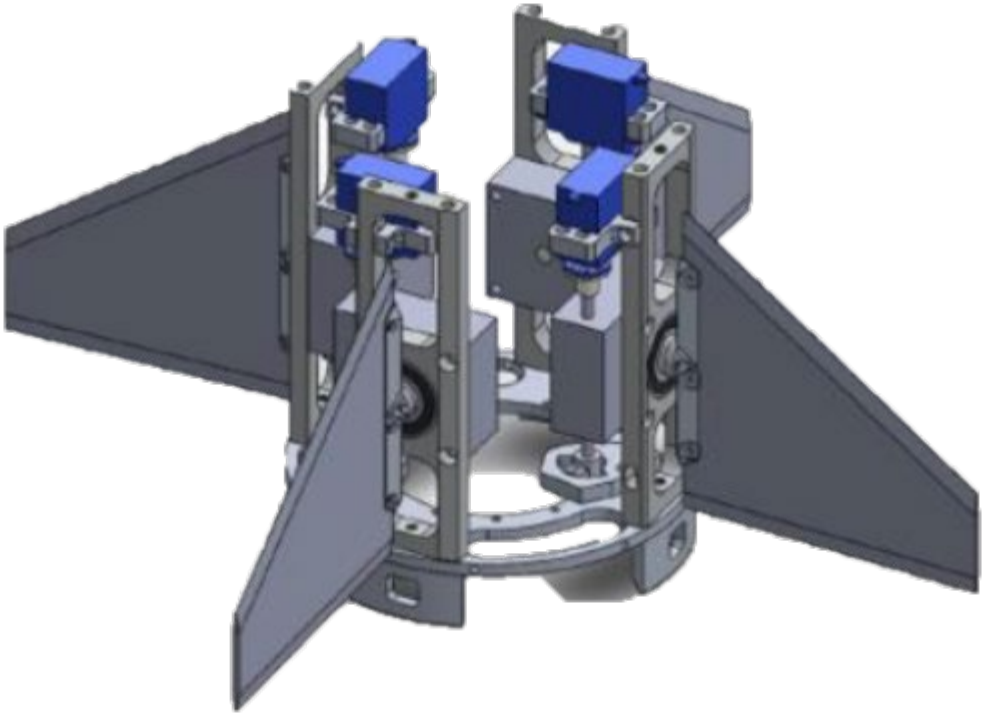

Figure 20. Fin Servo Assembly. Adapted from [32].

The Futaba HPS-H700 servos are used in this system due to their control rates of 0.07 seconds for 60 degree rotation and acceptable torque of 3.61 N-m [41]. The servos drive the 10:1 Ondrives gearboxes which then control the angle of the control fins. The control fins shown are adapted from the Alpha Fin 5 designed in the 2019 ME4704 design report [37] and modified in the 2020 ME4704 design report due to previous failures at high speed [32] . While the servos used for this flight are different from the design report, as they only have 3.61 N-m of torque as compared to the 7.768 N-m design parameter, limiting their use to 3 degrees Angle of Attack maintained a positive torque margin with the expected dynamic pressures [37].

Following the flight of Rocket 0 on November 21, 2020, as shown in Figure 21, the Fin Servo Assembly successfully augmented the rocket's roll but produced large oscillations that caused concern over the stability of the system with induced disturbances. Additionally, using a single gyro to measure the state of the rocket's orientation was largely unsuccessful as it was susceptible to heading drift. Further details will be discussed in Chapter 4.

Figure 21. Rocket 0 in Flight.

2. Sled Coupling

The method of coupling the rocket's seams and releasing them when commanded utilized technology readily available in the amateur rocketry world. A combination of 8-inch rocket airframe and a double wall coupler provides the separation boundary as depicted in Figure 22. Four 111 N (25 lbf) Nylon shear pins are inserted in small holes going through both the rocket airframe and coupling tubing. The aim of this system is to hold the rocket together under normal flight conditions and then to release when commanded.

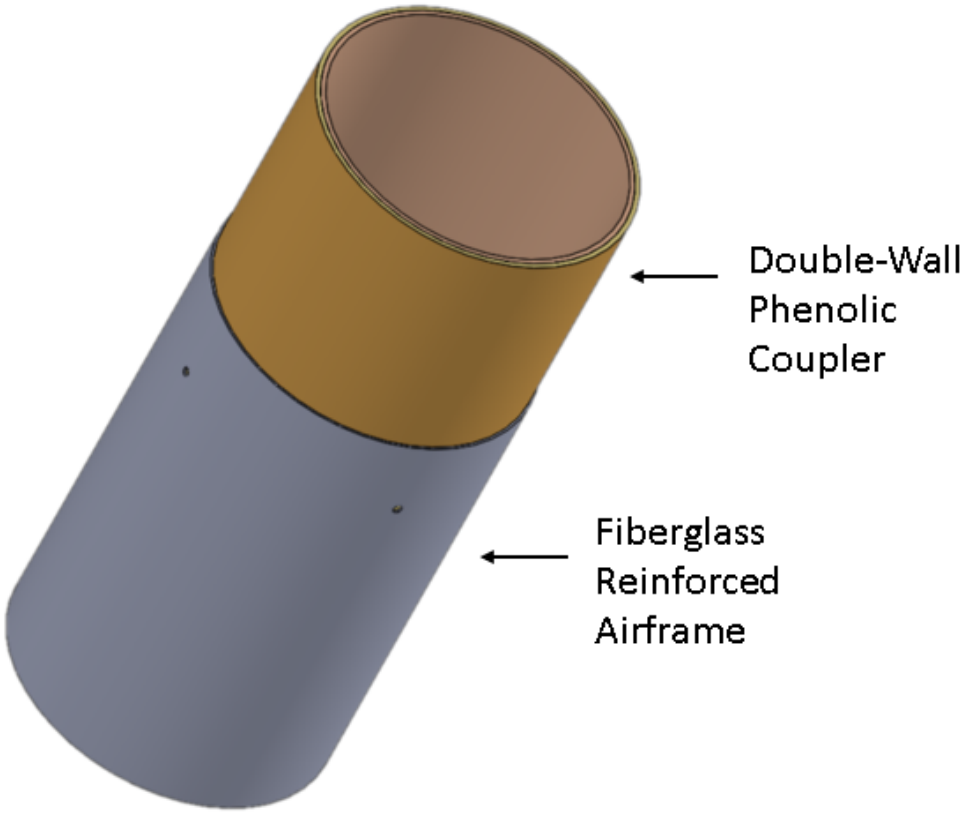

Figure 22. Amateur Rocketry Coupling.

Separation of the airframe from the coupler tubing occurs when an electronic match sets off a small black powder charge inside Apogee Rocket's CD3 High-Altitude CO_2 Ejection system as shown in Figure 23. The black powder ignites, sending a small piston with a firing pin to puncture the CO_2 canister. The CO_2 is released inside the volume of the section of airframe intended and provides a force for separation against the shear pins.

Figure 23. Apogee Rockets CD3 High-Altitude CO2 Ejection System. Source: [42].

The signal to separate comes from a PerfectFlite StratoLoggerCF as shown in Figure 24 . The system has a pressure sensing altimeter that detects apogee and proceeds to provide the current necessary to ignite the electronic match. The event settings of the system can be initiated in a variety of ways. On this launch, the PerfectFlite was set up to deploy the drogue parachute at apogee and to deploy the main parachute at 458m (1500 ft) above ground level (AGL).

Figure 24. PerfectFlite StratoLoggerCF. Adapted from [43].

During the flight, the main parachute prematurely deployed due to the shear pins failing mid-flight. The impulse of the drogue parachute on the sled airframe was sufficient force to allow for the main parachute to deploy. While it is better to have premature separation of the Sled's main parachute bay than none, it did not allow for the recording of video from the nosecone while suspended from a drogue as intended.

3. Battlespace Imaging and UAS Tracking

Capturing video from the nosecone of the rocket deviated from the 2020 ME4704 design report in that the Nvidia Jetson Xavier AGX processor and Allied Vision Alvium 1800U camera [32] were not considered due to a cost-risk analysis. Firstly, confidence was moderate that the rocket would fly as performed as it was a new build. Additionally, the Nvidia and Alvium camera combination was over $2000 and it was a challenge to get the system operational within the launch time frame. With these factors in mind, the decision to use a Raspberry Pi 4B and Raspberry Pi Camera Module V2 , as shown in Figure 25 was made as the cost is less than $100 and could provide 1080p Resolution at 30Hz [44].

Figure 25. Raspberry Pi and V2 Camera. Adapted from [45].

Utilizing Python and a single line of code, the system was able to record video for the life of the battery supplying it and available memory on the SD card. As tested, the battery could maintain the Raspberry Pi for over 2 hours and could record well over 2 hours of video data on a 32GB SD card. While the system performed optimally during flight, unanticipated electromagnetic interference (EMI) from a downlink transmitter disrupted the camera signal and distorted the video.

B. RESULTS

The inaugural launch of the RRPD-V (Rocket 0) was an observation flight to gather experience with high powered rocketry on November 21, 2020.

1. Control

Unfortunately, the forces on the vehicle due to the extreme angular accelerations felt during the flight caused the SD card to unseat, losing flight data from the IMU that was to be obtained. This launch still provided valuable data, however, as the installation of a forward and aft looking camera external to the vehicle allowed for estimation of the flight roll and roll rates by overlaying a transparent protractor over the aft looking video as shown in Figure 26. The video was converted to black and white to increase the contrast and a line was overlayed along a stretch of road to mark a reference point. The data was obtained by moving through each frame and noting the time stamp and maximum values of each oscillation. The corresponding data is displayed in Figure 27.

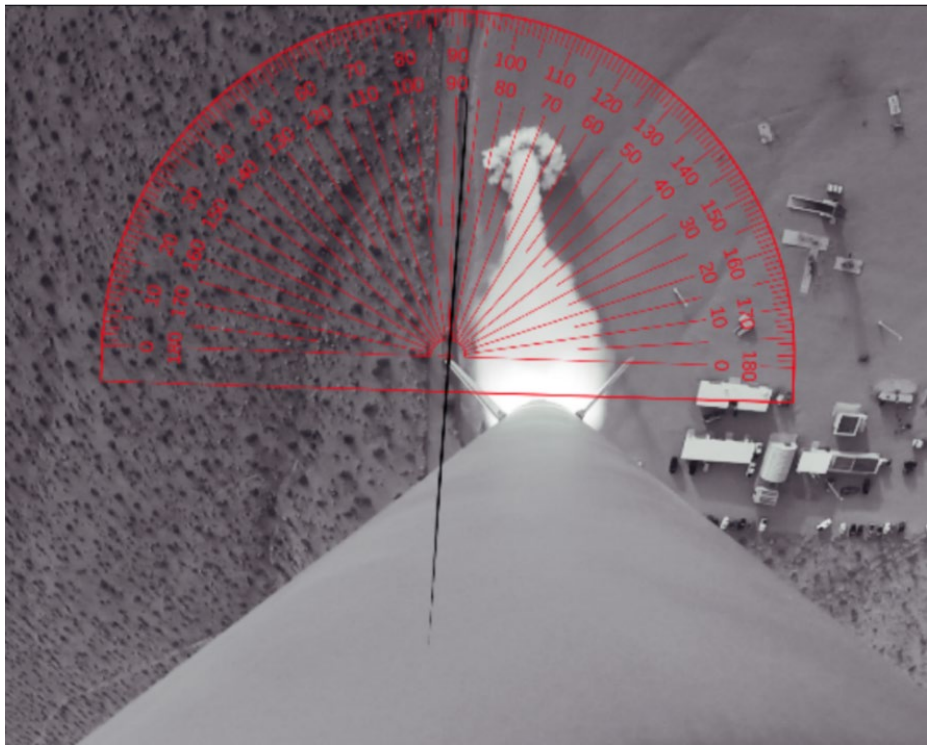

Figure 26. Measuring Rocket 0 Orientation with Video Data.

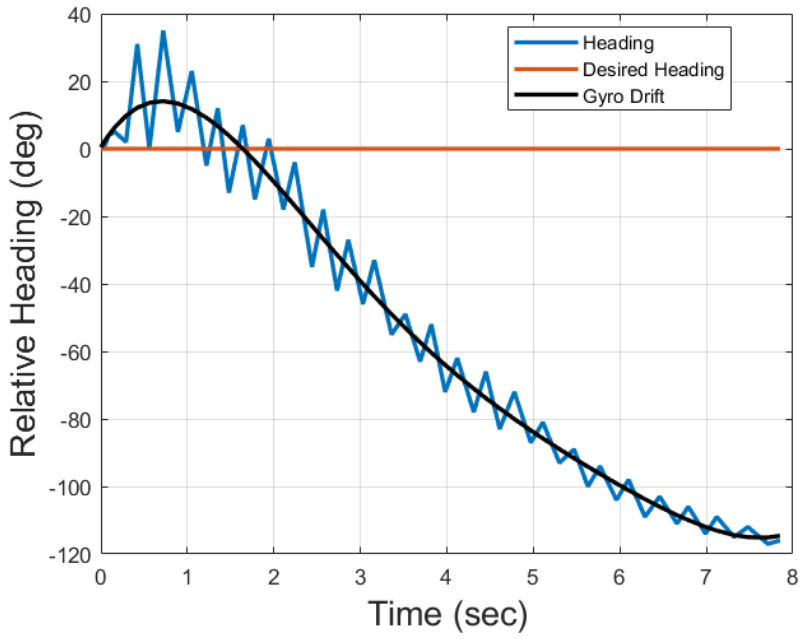

Figure 27. Reconstructed Rocket 0 Data.

The raw heading data shown in Figure 27 has some features to it that are undesirable in many ways. The legacy control algorithm that was used was relying on controlling angular velocity. Unfortunately, this gyro rate sensor was prone to gyro drift where initial relative heading became uncertain immediately after launch. Accordingly, a trend line is plotted with the data to show the drift in what the vehicle believes the heading to be. While this data was reconstructed from crude mechanisms and the accuracy of the heading values cannot be verified, overall trend in the data did exist and allowed for observing the heading slip. Additionally, the amplitude of the oscillations can be approximated to be roughly 10 degrees with a frequency of 3Hz. The system as shown has no damping applied as the amplitude and frequency of the oscillations is constant.

2. Sled Coupling

The Sled coupling failed to hold together after apogee as it prematurely released the main parachute due to the force of the drogue parachute being rapidly inflated. As caught in the forward-looking external camera in Figure 28, the Sled is releasing from the

nosecone immediately following drogue deployment. In theory, the shear pins should have held until the PerfectFlite signaled release at 457 m (1500 ft) AGL.

Figure 28. Premature Separation of Sled and Nosecone.

While the premature separation is unfortunate, it did establish that 4 shear pins on this system was too few. Ground tests will be used to determine the appropriate number of shear pins to allow for the system to work properly.

3. Video

1080p video data was achieved by the Raspberry Pi and Raspberry Pi Camera V2 at 30Hz from the flight. While the video quality was acceptable during ground tests with all equipment operational, the data showed signs of EMI after the SRM ignited as shown in Figure 29. The ribbon cable connecting the Raspberry Pi and its camera was speculated to be the cause of the interference as the cable passed by a 250mW downlink data transmitter.

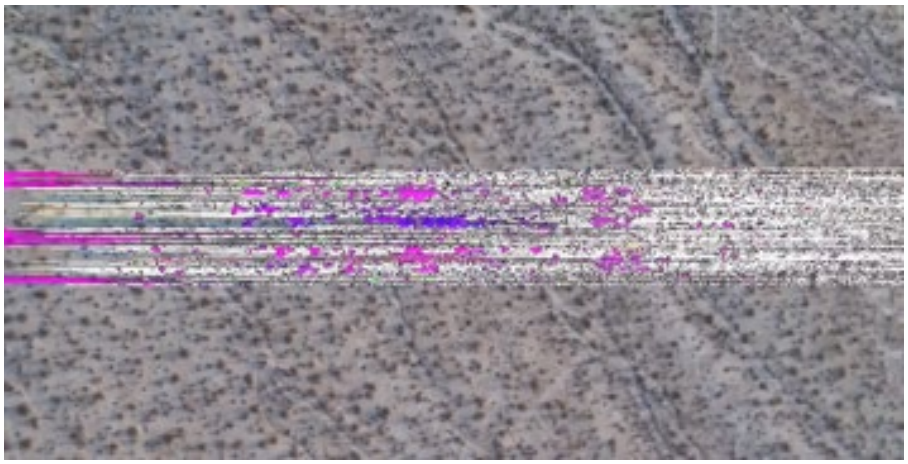

Figure 29. Raspberry Pi Camera V2 EMI.

Additionally, due to premature separation of the Sled, the majority of the video was not taken from the perspective of a free-falling Sled on a drogue parachute but instead from the large drag of the main parachute. This allowed the nosecone to swing as if it were on a pendulum instead of holding steady. While this flight was not satisfactory for video collection, it did provide several datapoints from which improvements could be made.

IV. FLIGHT TESTING AND RESULTS

A. ROCKET 1

1. Development

Building on the results of Rocket 0 detailed in Chapter IV(A), the major aim of Rocket 1 was to employ new technology and improve on lessons learned. This rocket was designed to use the Cesaroni M3400 SRM to provide an average thrust of 3421.1 N (770 lbf) over a time period of 2.9s [46].

a. Control

Vehicle control was seen as a large hurdle to Rocket 0 as the refresh rate from the BNO055 IMU to the Arduino Uno was limited to 30Hz based on the Arduino hardware bottlenecks. This limitation was theorized to have a large impact to the ability of the rocket vehicle to stabilize because the rocket was moving very quickly and has an extreme roll response to the smallest disturbance. Accordingly, the Raspberry Pi 4B with 8GB RAM was available with 1.5GHz processing capability [47] and the BNO055 IMU offered filtered Euler angles or quaternions with both relative and true bearing at 100Hz [48]. Accessing the 100Hz information would increase the rates of data threefold and was deemed a worthwhile investment. Additionally, the Raspberry Pi 4B is compatible with MATLAB and Simulink with compiler support to run programs autonomously. This allows for more tools to be considered outside of the box of Python or C coding that natively run on the Raspberry Pi. A detailed guide of how-to setup a Raspberry Pi to communicate with MATLAB and the sensors employed is provided in Appendix A.

(1) BNO055

The BNO055, as shown in Figure 30, is a complex IMU that allows for multiple modes of operation to obtain the required data. Rocket 0 only used the BNO055 for sensing angular acceleration through its gyroscope but is capable of much more. The IMU contains an accelerometer, a gyroscope, and a magnetometer that can be used on their own or as Kalman filtered, fused data output in the form of Euler Angles or Quaternions [48]. While

Quaternions are generally preferred in the aerospace industry due to singularities that exist when calculating Euler angles at π/2 radians [34]. Despite the standard, Euler angles were chosen for this work because the IMU accounts for the singularities natively with internal calculations [48]. Additionally, Euler angles allow for intuitive prototyping as the direct output matches the orientation of the sensor. To get the Euler angle output, the IMU must be set to combine at least two of the three sensors to provide an orientation. The decision was made to not use the magnetometer as it would likely be subject to interference as it is installed in an electronics suite. Thus, the operational mode is set to 'IMU' allowing for the fusing of the gyro and accelerometer to output Euler Angles [48]. This approach will only provide relative orientation of the rocket body to its calibrated orientation. The magnetometer would add in a third leg to provide a true bearing. Fortunately, the relative bearing can be accounted for using coordinate transformations, when necessary.

Figure 30. Bosch BNO055 9DOF IMU. Adapted from [49].

Connecting the BNO055 to the Raspberry Pi is possible using two methods, either through Universal Asynchronous Receiver Transmitter (UART) or through Inter-Integrated Circuit (I2C) Protocol. While UART offers a simple connection without the need for a clock to synchronize signals, an I2C bus can host up to 128 sensors at much faster speeds than UART can offer, up to 3.4GHz [50]. Unfortunately, the BNO055 requires the I2C connection speed to be slowed to 10kHz to ensure proper operation, limiting the benefits of using the I2C communications protocol [48]. While the differences in speed of the connection between the two protocols offer no benefit on how to connect

the BNO055 to the Raspberry Pi, the capability of hosting multiple sensors is an investment worth developing. For this reason, I2C was the protocol chosen to provide connection to the Raspberry Pi.

To operate in the IMU mode, the BNO055 must undergo a calibration prior to use. The calibration must be initiated after selecting the operating mode by holding the sensor still to calibrate the gyroscope and by maneuvering and holding the BNO055 in 6 different planes of 45-degree changes to calibrate the accelerometer, as detailed in Appendix A. A script has been written, BNOCAL.m, to allow for the connection of MATLAB and the BNO055 to the Raspberry Pi in Appendix B. Additionally, the code allows for the saving of the calibration data into a mat file so that the calibration data can be re-written without going through the calibration procedure in the event the sensor loses power. To write the calibration, the script CalWrite.m has been created and is in Appendix C.

(2) Simulink

While MATLAB is necessary to initially connect with the Raspberry Pi and calibrate the BNO055, Simulink offers an intuitive and visual based software for creating flight control algorithms. Additionally, the deployment of Simulink code to the Raspberry Pi has proved to be more seamless than that of writing raw code in MATLAB. With these items in mind, Simulink was chosen as the platform to create the control algorithms.

Using Simulink allowed for the use of several block-sets in its library that are specifically made for the Raspberry Pi. To create the first roll control system, as seen in Figure 31, a starting point of collecting the data from the BNO055 IMU was required. The capability was achieved in the form of the 'I2C Master Read' blocks that allowed for the selection of the sensor in hexadecimal notation and its associated register where the information is stored. Once the data is collected, it can then be converted to a double precision floating point number for Simulink computations, and corrected with a conversion factor to provide the expected output data type [48]. This is done for the data registers containing the information for the Euler Angle of Roll and for Angular Velocity, Roll Rate, about the primary axis of the rocket.

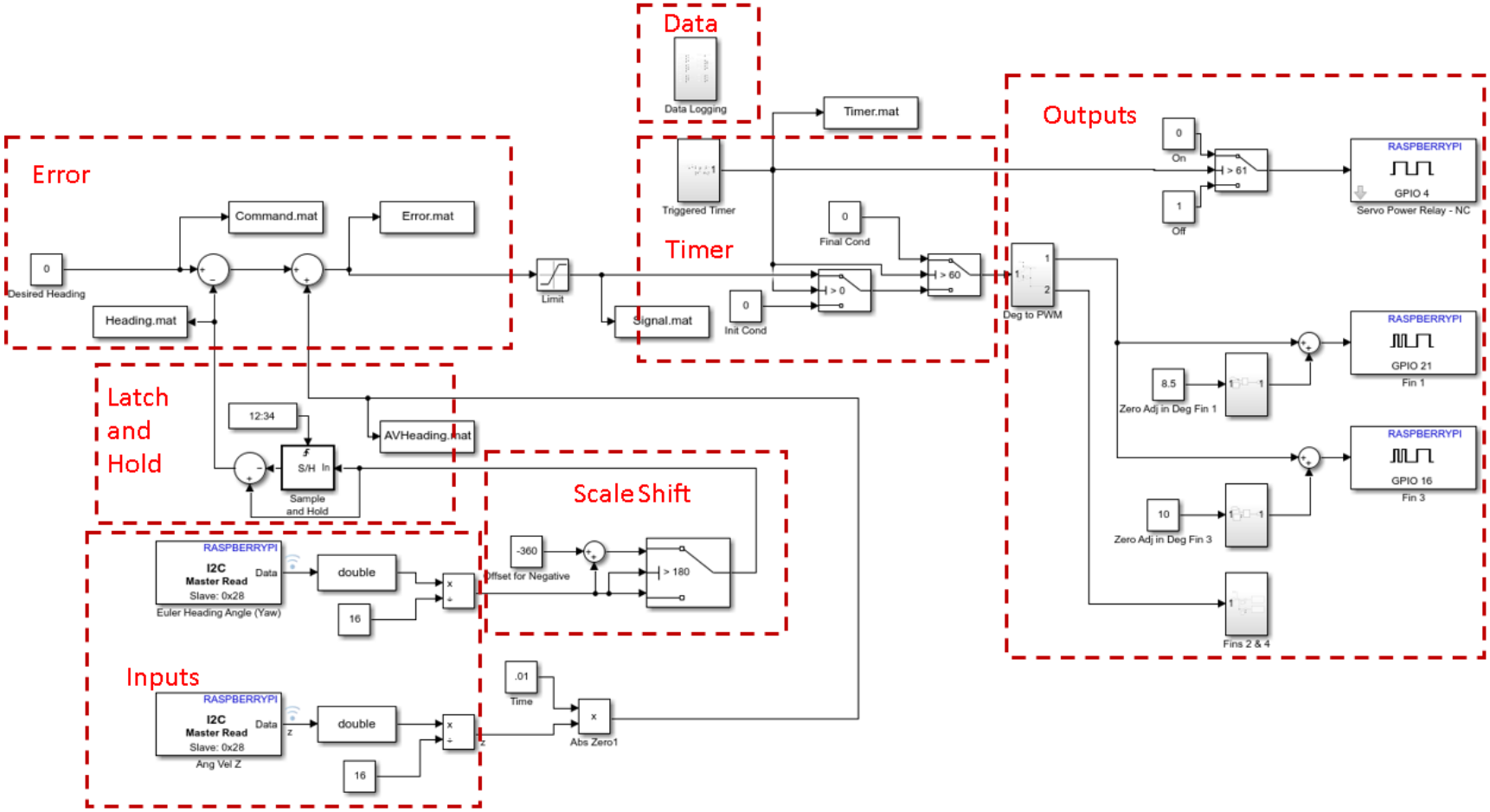

Figure 31. Rocket 1 Simulink Model for Roll Control.

After the data for the heading and angular velocity are in proper formats, they are transformed into useful ranges of data for control. The Roll sensor has a range of 0–360 degrees, resulting in issues when the sensor is near 0 or 360. To counter this, an attempt was made to shift the range of the sensor to ± 180degrees as it was not expected for the rocket to obtain a roll that is carried beyond 180 degrees. The heading signal is then sent to a 'Latch and Hold' block so that any heading that is initially output will effectively tare and yield a relative zero. The initial bearing off the launch rail would be the heading about which the roll control is based on. Additionally, the angular velocity signal is multiplied by the sample time step to provide an additional signal into the calculation to follow. Although the methodology is flawed in this setup, the process of needing to account for sensor discontinuities and incorporate derivative controls is being developed.

The heading is further subtracted from the desired heading and added to the angular velocity to yield an error signal. The error signal is sent through a 'saturation' block where upper and lower bounds of the signal are established. The upper and lower limit was set at ± 10, respectively, to allow for a 1-degree deflection of the control fins following a 10:1 reduction through the reduction gears. This signal is then only used after a triggered timer, as shown in Figure 32, is initiated and becomes greater than zero. The triggered timer is initiated when the linear acceleration on the z-axis becomes greater than 2g. This logic loop is required because testing revealed that a simple latch and hold concept would cause the timer to reset after the linear acceleration bounced below and above the threshold set. After 60 seconds, the fins would return to a zero position and at 61 seconds a relay would trip to remove power from the servos to prepare for landing without over torquing.

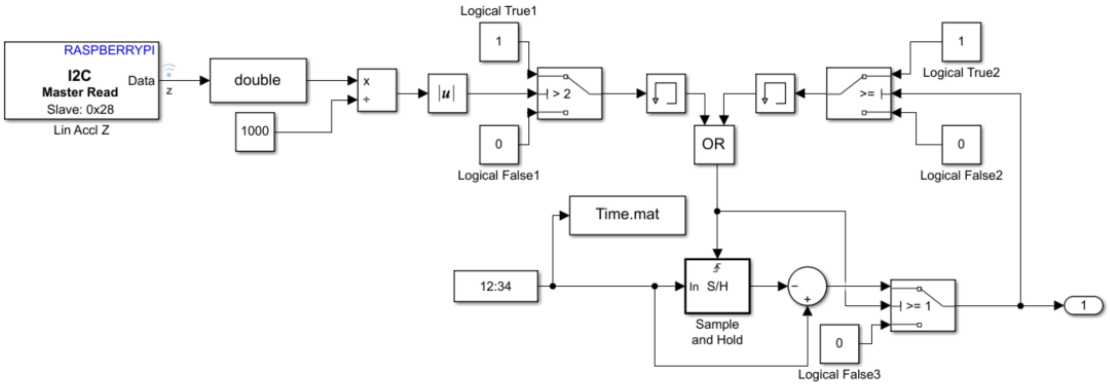

Figure 32. Triggered Timer.

The signal is then pushed through a block that converts the degrees demanded into a PWM signal required to drive the servos, as shown in Figure 33. While Simulink offers a block in their library to convert degrees demanded to drive servos, the blocks are very computationally heavy and caused an unacceptable lag while testing the program. The PWM signal then needs to be adjusted prior to entering each servo's individual General Purpose Input Output (GPIO) pin on the Raspberry Pi. The adjustment allows for zeroing the control fins prior to actual flight as the servo zero mark may not align with the control fins zero as aligned with the rocket airframe.

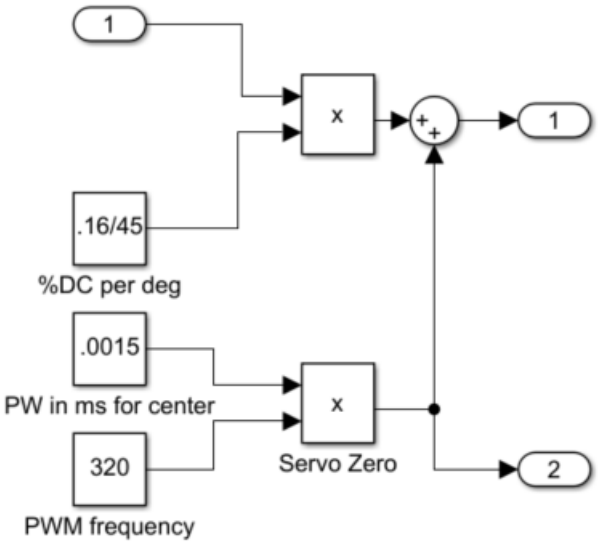

Figure 33. Converting Degrees Input to PWM Signal Output

Once the triggered timer activates, control signals are sent to the servos for 60 seconds of flight, as this was deemed to be twice as long as the signal necessary in most cases. After 60 seconds, the signal returns to zero and a relay controlling power to the servos is activated to open the circuit. The relay control was added to the control system to attempt to save the servos upon a rough landing of the booster section of rocket post flight. This implementation is in response to Rocket 0's flight where the booster was found with the servos making loud noises postflight. The rough landing caused an impulse through the reduction gears to the servo causing damage to the worm gears.

Lastly, a data logging block was implemented to store all data possible from the BNO055 for post flight analysis or future analysis as required. As seen in Figure 34, the items logged are linear acceleration, angular velocity, raw acceleration, temperature, quaternions, and gravity vectors. While all of the data might not be readily necessary to analyze the flight, establishing a historical record of the data is desired and accomplished.

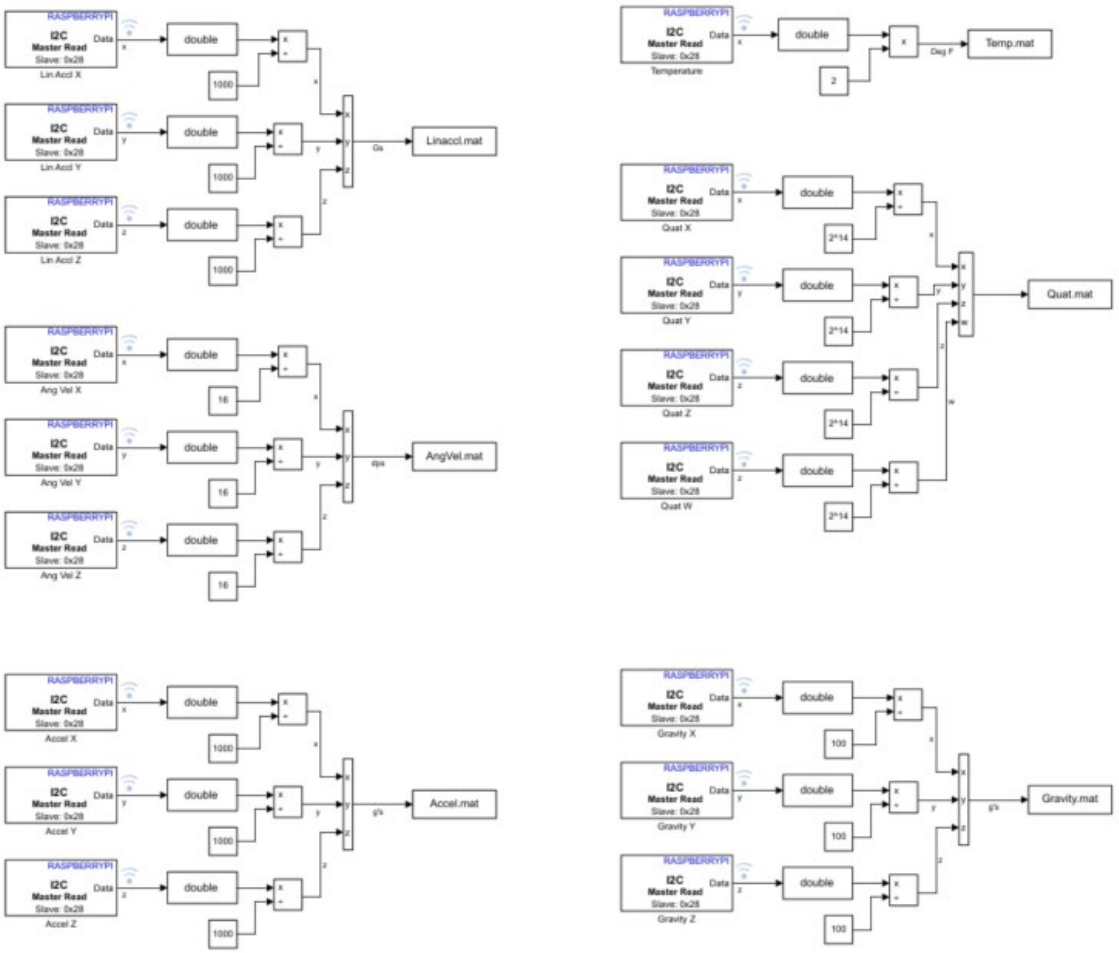

Figure 34. Data Logging in Flight from BNO055.

b. Sled Coupling

Following the results of Rocket 0, ground tests were performed to determine the maximum amount of shear pins that could be inserted into the mating coupler pieces and still allow for separation upon initiation of the CO_2 charge. An initial test was performed with 8 shear pins. The experimental setup was performed using the Sled, parachute, and the forward coupling that would connect to the nosecone as shown in Figure 35. The electronic match was detonated by a 9V battery manually at a distance. The resulting test did not allow for separation of the rocket.

Figure 35. Ground Testing Sled Coupling Separation with 8 Shear Pins.

Following the failed test, the CO_2 cartridge and Black Powder were reloaded, and 6 shear pins were inserted. Much like the previous test, the electronic match was ignited with a 9V battery yielding successful results as shown by separation and a cloud of CO_2 being released in Figure 36. The decision was made to utilize 6 shear pins as the CO_2 and electronic match stock was running low. The resulting flight would reveal another failure.

Figure 36. Ground Testing Sled Coupling Separation with 6 Shear Pins.

c. Video

While the video received from Rocket 0 was partially successful, there were several improvements that needed to be made. Areas of improvement were focused on eliminating EMI and in obtaining higher quality resolution. Additionally, the Sled Coupling required improvement to obtain stable video.

(1) Electromagnetic Interference

The source of EMI was readily known as a downlink transmitter from Kate 2.0, a flight data recorder and GPS locator as shown in Figure 37, provided a 250mW transmission [51]. While the camera system was previously tested on the ground with all electronics in operation, no EMI was experienced. Again, the camera was put into operation with all flight electronics in operation and no EMI was noted. As the Kate 2.0 transmitter has the capability to transmit at both 0.5W and 1W for high altitude rocket flight transmission, these modes were tested with no sign of interference.

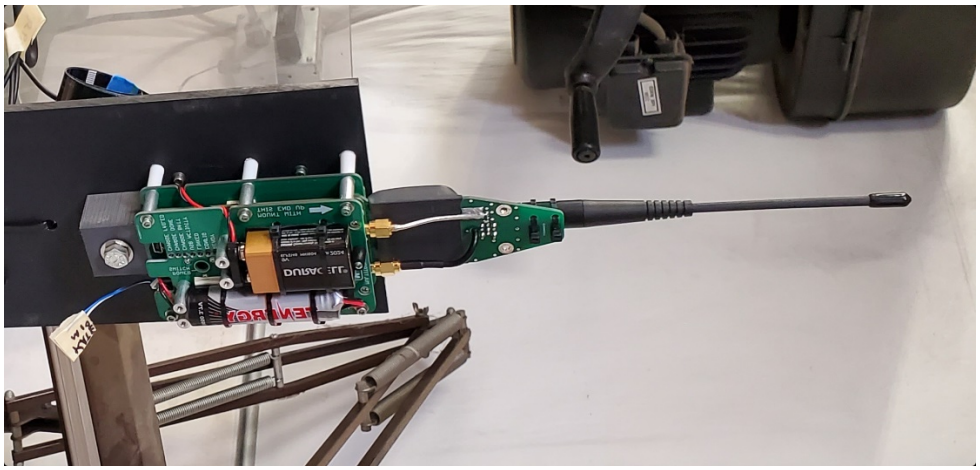

Figure 37. Kate 2.0 Mounted for Nosecone Insertion

The manufacturer of Kate 2.0 was contacted to help understand the issue at hand. Vern Knowles with Multitronix LLC was able to aid and mentioned to test while moving the Raspberry Pi ribbon cable around the transmitting antenna. The follow-on testing

revealed that the ribbon intermittently touching the transmit antenna would reproduce the EMI that was experienced during flight as seen in Figure 38.

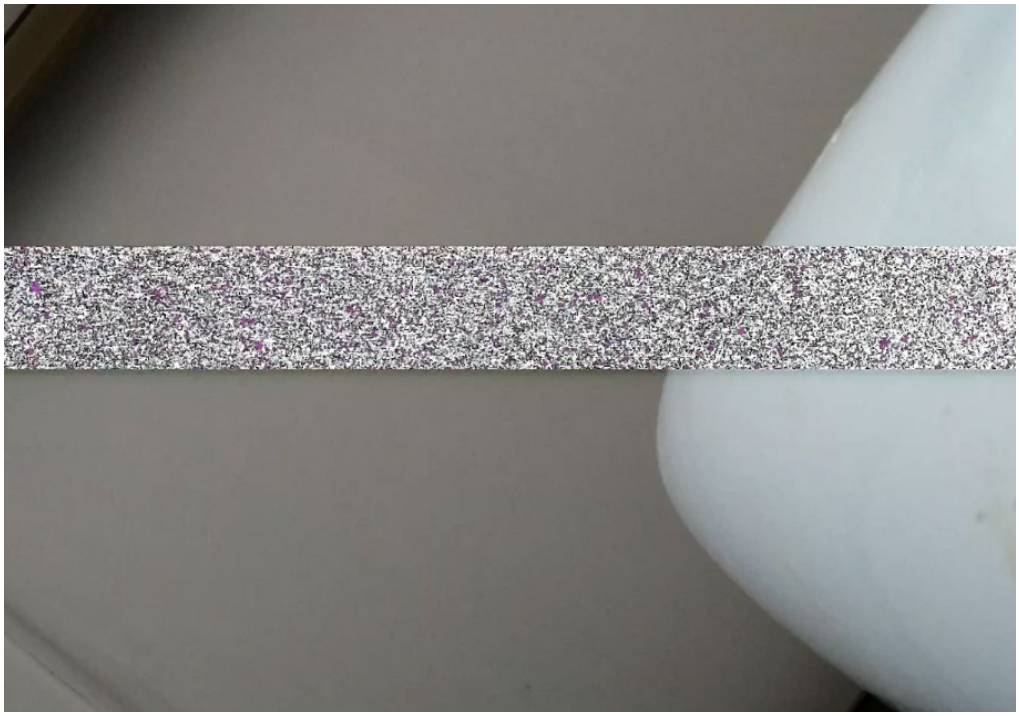

Figure 38. EMI Imposed While Testing.

To mitigate the EMI, aluminum foil was wrapped around the Raspberry Pi ribbon cable to be used as shielding, as seen in Figure 39, to test whether the ribbon cable could effectively be shielded or determine if a new video system was required. The foil was wrapped around the cable and tested by moving the ribbon cable transmission occurred. The test was successfully repeated while eliminating the EMI. Additionally, a second test was performed where the transmit power was increased to 1W; the test was successfully repeated with no observed EMI. The aluminum foil was replaced with braided stainless steel wiring harness shielding for flight due to quality concerns with aluminum foil.

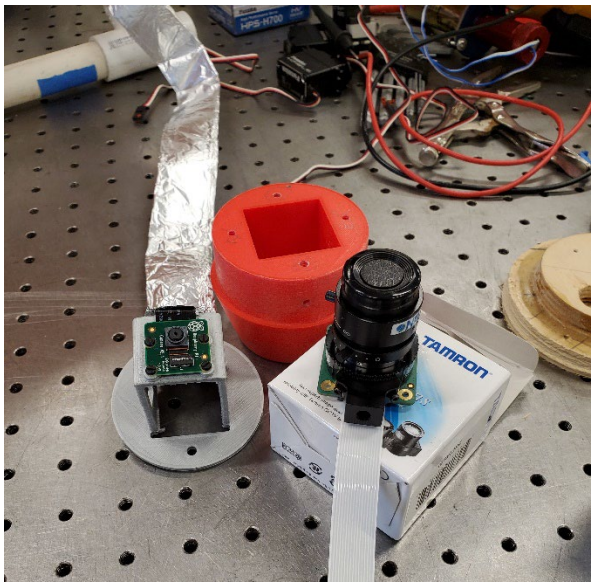

Figure 39. V2 Camera with Ribbon Shielding and HQ Camera.

(2) Camera Selection

The Raspberry Pi V2 Camera, while capable, did not provide the resolution desired to eventually track targets. Because of this, the Raspberry Pi High-Quality Camera was selected as it was capable of recording 4K video at 10Hz and has interchangeable lenses allowing for changing the field of view [44] as shown in Figure 39. Unfortunately, the Raspberry Pi 4B does not allow recording video of resolutions higher than 1080p as the h.265 codec is for video streaming only. The h.264 codec can only record 1080p resolution at 30Hz which provides no better quality than with the V2 Camera [45]. While the video resolution could not be increased, the ability to change the lenses offered a potential magnification of the area and a narrowed field of view. The High-Quality Camera would be chosen for the next flight.

2. Results

After implementing the Raspberry Pi microprocessor to replace the Arduino microcontroller and ensuring locks were in place to hold down the SD cards, two RRPD-V designs (Rocket 1 and Rocket 2) were flown on April 17, 2021.

a. Control

Rocket 1 data was expected to return results to show how a purely proportional control system could provide feedback to allow for the roll control of the RRPD-V. As shown in Figure 40, the results were confused and jostled. The black solid line indicates the heading of the IMU and rocket, the red solid line is the demanded signal to the servos, and the black dotted line is the commanded heading. At first glance, the data appeared corrupt as the heading bounces about perceived asymptotes and the demanded signal looks nothing like a curve but instead resembles a square waveform. The peaks and valleys of the heading signal resembled what was expected, a sinusoidal pattern that was bouncing back between -180 and 180 degrees instead of around zero. Looking back at the BNO055, the range of the sensor measuring roll is from 0 to 360 degrees, indicating that the system had hit the limits of the sensor and had wrapped from -180 to 180 degrees due a self-imposed scale shift of 180 degrees, attempting to provide an 180-degree buffer that the rocket would be allowed to stay within. Additionally, the data set appears to have a bias that is introduced by attempting to create an initial heading of zero when the rocket was on the rail waiting to be fired. Using MATLAB code, the bias, 180-degree flops, and noise are removed to produce Figure 41.

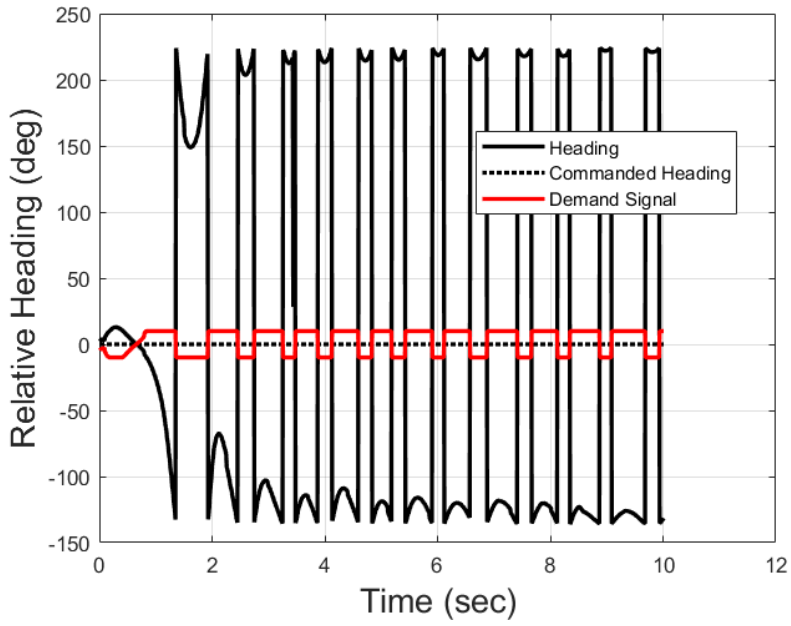

Figure 40. Rocket 1 Raw Heading and Demand Signal Data.

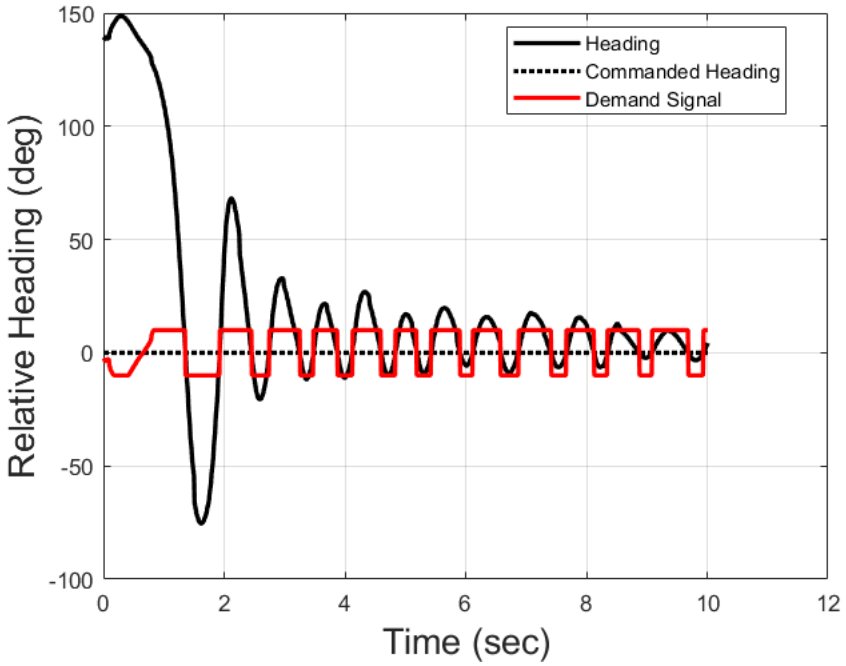

Figure 41. Rocket 1 Roll Response with Processed Heading and Demand Signal Data.

Analyzing Figure 41, the first second of data shows a desired response. The control fins were given a signal to counter the rotation of the rocket and the roll indicated a change of direction. However, after the first second the demand shifts from resisting the rotation to a signal that is encouraging rotation, yet the rocket continued to oscillate about a center point. Upon reviewing the Simulink code used for this flight, it became obvious that the attempt to shift the scale to -180 to 180 degrees instead of 0 to 360 degrees was largely flawed. Given that the initial heading on the rail was likely close to the sensors 0 to 360 degree transition point, the code worked appropriately until the transition point was hit following the first rotation. After the transition point, the signal given was opposite of the desired input and allowed for an approximate 150 degree roll of the airframe where the signal then bounced back and forth between the imposed $\pm$180 degrees from the scale shift. This response pointed out that the method was invalid and that a new way to handle sensor thresholds was required.

b. Sled Coupling and Video

The Sled Coupling did not hold up on this flight as premature separation of the nosecone was once again observed. The force from drogue deployment was too great for 6 shear pins. As shown in Figure 42, the Sled main parachute can be observed in the distance only three seconds after separation. As the shear pins again failed, the need for a new method of coupling separation is desired. With this flight, however, the nosecone was not recovered due to the Sled's shock cord snapping. Because of the high winds, the nosecone was allowed to be taken across the desert faster than the recovery team could catch up to it. The last sighting was a picture taken by Peter Thoeny while on recovery of the booster section in Figure 43. Consequently, the video data was never recovered, and the nosecone was deemed a total loss.

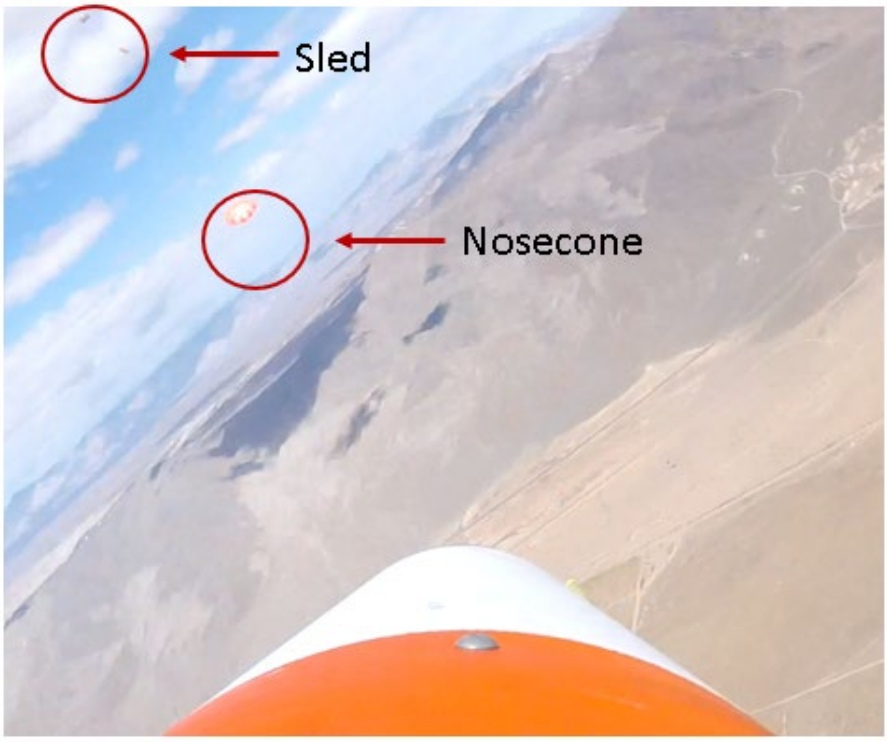

Figure 42. View From Rocket 1 Forward Camera in Flight.

Figure 43. Last Known Sighting of Rocket 1 Nosecone.

B. ROCKET 2

1. DEVELOPMENT

The vehicle development on Rocket 2 was very similar to Rocket 1 with the exception that it did not have a nose mounted camera and that it had a second stage motor in the sled to attempt to achieve higher altitudes and flight data at higher dynamic pressures. In addition, the SRM on the booster stage was the Cesaroni N3301 which was expected to yield an average thrust of 3294.3 N (740 lbf) over 5.86s [52] compared to the M3400's 3421.1 N (770 lbf) average thrust over a time period of 2.9s [46] . The second stage would hold a Cesaroni M3100. Due to the larger motors and shifting of the center of gravity, 31 N (7 lbf) of weight was required to be added to the nosecone to ensure stability. Rocket 2 can be seen in Figure 44 from a picture taken by John Gayler as it is being raised on the launch rail following the flight of Rocket 1.

Figure 44. Rocket 2 on the Launch Rail.

a. Control

The control algorithm used was the exact same as Rocket 1 and provided an attempt to understand how the controls would act at the higher expected dynamic pressures. The data had not been analyzed from Rocket 1 at this point, as the launch was on the same day. The overall trends would tend to be the same.

b. Sled Coupling

The Sled Coupling had not been altered at this point as the overall objective of this rocket was to ensure that a second stage flight could be achieved. As such, the coupling design was the same as seen in Figure 22.

c. Video

Nosecone video was not used on this rocket as it was theorized that the higher intended dynamic pressures required a more aerodynamic nosecone.

2. RESULTS

The flight of Rocket 2 was largely unsuccessful as the rocket airframe failed only seconds into flight. The spine of the rocket snapped due to a structural weak point in the mating of the booster and interstage coupler. The additional weight added to the nosecone only increased the bending torque experienced by the rocket. A screen capture from cell phone video can be seen in Figure 45.

Figure 45. Rocket 2 Airframe Failure in Flight.

a. Control

While the control algorithm is the same as was seen in Rocket 1, the results are somewhat different as the roll of the system had intermediate oscillations. Since the system had a hard landing, the data was corrupted and only a small amount of initial data was able to be captured. With the help of Dr. Hyatt Moore IV, the data was able to be pulled from the corrupt files using a MATLAB function, as available in Appendix D, to recover data with a known time array. As shown in Figure 46 following correction for bias and 180-degree flips, the rocket had some proportional control for the first 0.25 seconds until the BNO055

sensor reached its internal 0 to 360 boundary. After this point, the system tended to roll towards the commanded heading but had oscillations in between due to the excessive winds during the launch. After the system reached zero, it is commanded to move about the commanded heading until the rocket became unstable.

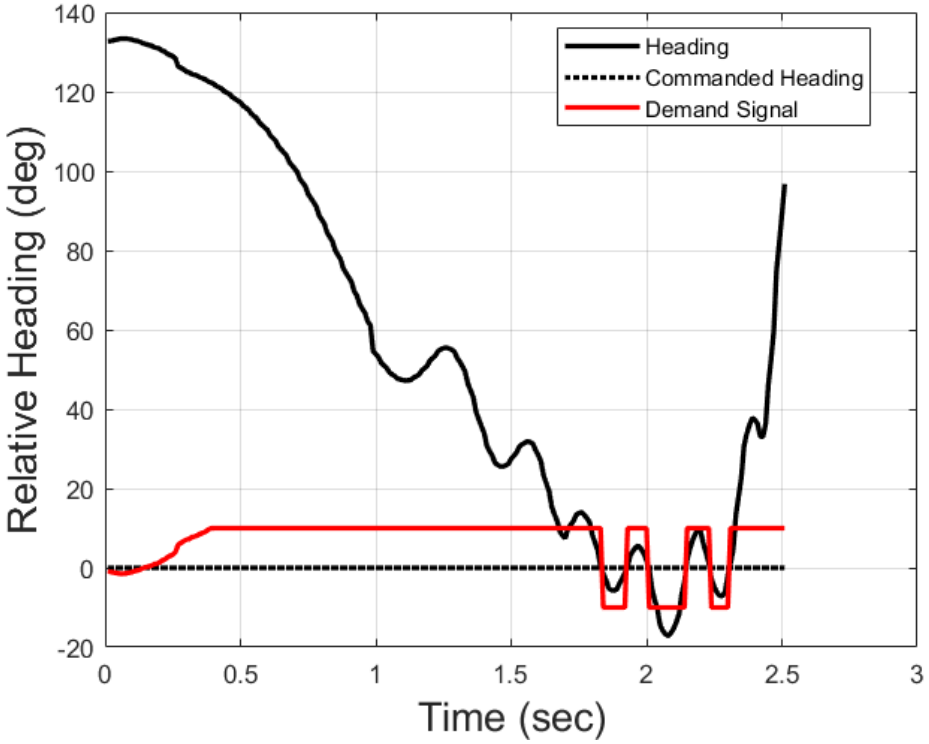

Figure 46. Rocket 2 Control Results.

b. Sled Coupling

The sled coupling design was not analyzed in this rocket as the force from the airframe failure caused premature separation and the shock cord attaching the nosecone snapped. As shown in Figure 47, the nosecone freely fell to the ground impacting at 160 m/s. The nosecone and its associated electronics were deemed a complete loss.

Figure 47. Rocket 2 Nosecone Following Impact with Ground.

c. Video

A nosecone camera system was not installed on this system to preserve the aerodynamic integrity of the nosecone.

C. ROCKET 3

1. DEVELOPMENT

Building on the experiences of Rocket 1 and Rocket 2, several improvements were required to ensure the flight would be able to meet the objectives in Chapter 1. This rocket was built using the booster from Rocket 1, due to the crash of Rocket 2, and the Sled from Rocket 2 as it added more volume for parachute packing and provided the use of the aluminum coupler designed by Kai Grohe for stage separation between the booster and Sled as seen in Figure 48.

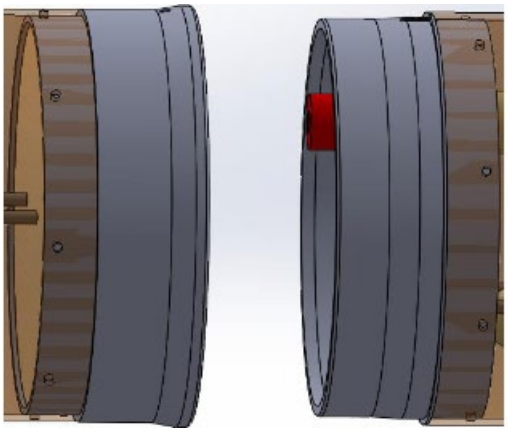

Figure 48. Aluminum Coupling Stage Coupling. Source: [28].

a. Control

The control algorithm from Rocket 1 and Rocket 2 allowed for some control over the system, but the 0 to 360 degree heading transition of the BNO055 IMUs proved to be problematic as displayed in Figure 49. The issue during flight stemmed from trying to shift the sensor from a scale of 0 to 360 to -180 to 180 degrees, as shown in Figure 50. The attempted corrective function, boxed in red, allowed for wrapping the scale from -180 to 180 but still had a transition problem when rotating past the IMUs heading boundary.

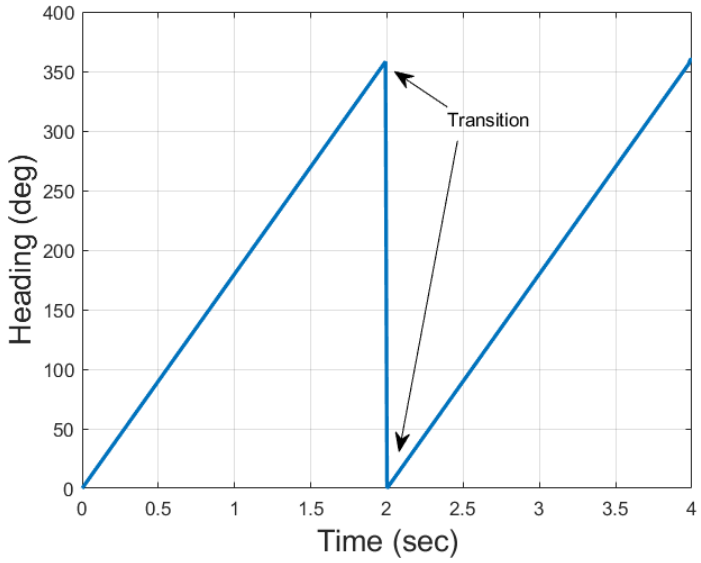

Figure 49. BNO055 Heading Transition

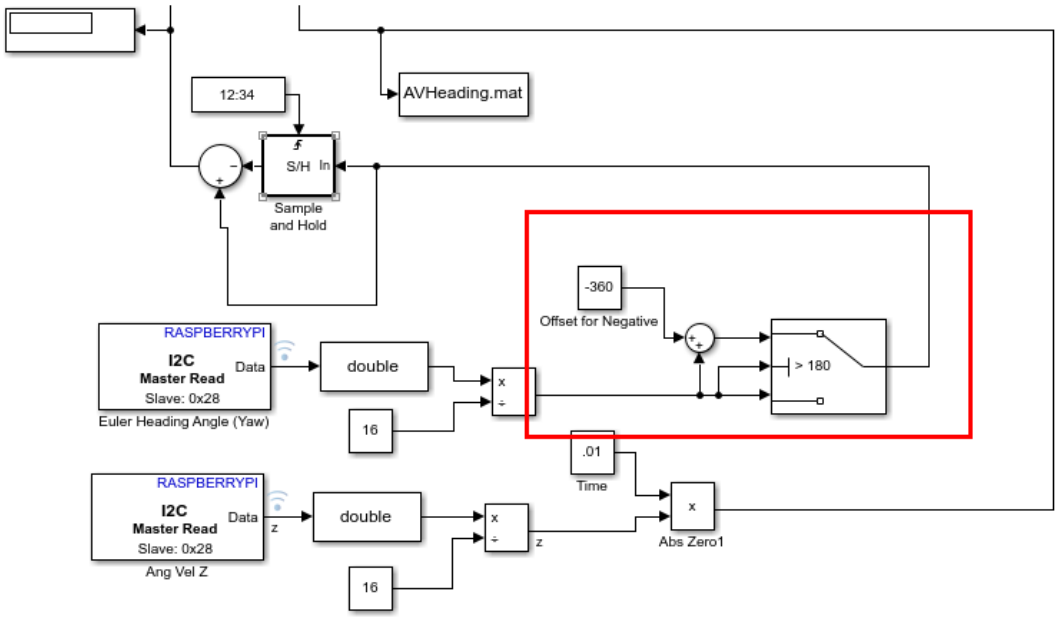

Figure 50. Rocket 1 Control Algorithm Problematic Simulink Code.

Several attempts were made to develop an algorithm that would allow for the use of the IMUs full range of inputs and to wrap the transitions so that once the 360 boundary was met, the next value would be 361 degrees instead of 0. Similarly, if the 0 boundary was reached, the next heading would be -1 instead of 360. This issue was able to be resolved using a DSP Toolbox block, “unwrap,” in Simulink designed to shift phase of a signal during signal processing. The “Unwrap” block was implemented by easily transitioning the signal to radians, providing a phase shift if necessary, and then shifting the data back to degrees. The implementation is shown in Figure 51.

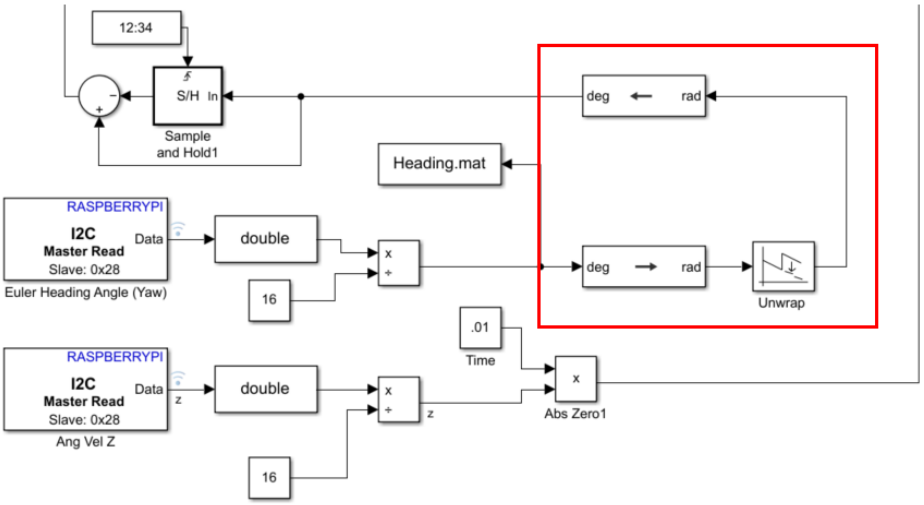

Figure 51. Rocket 3 Control Algorithm with Unwrap Phase Shift.

Testing of the 'Unwrap' tool yielded results as shown in Figure 52 with a 0 to 360 ramp input and a phase shifted output. The output of the function is exactly as is required and allows for the crossing of the 0 to 360 IMU boundary using a phase shift.

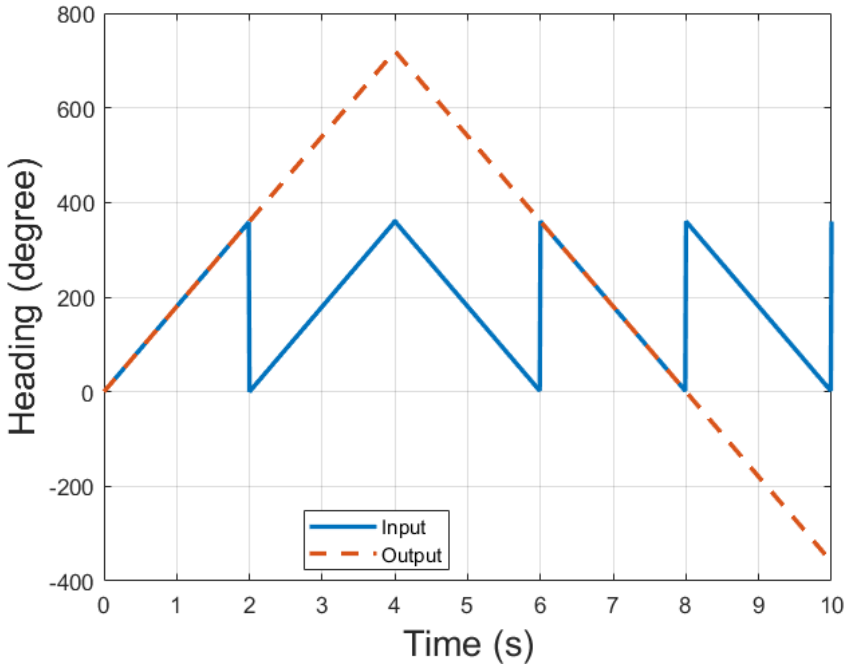

Figure 52. Ramped Input with Phase Shifted Output.

To further test the implementation of the ‘unwrap’ function, a full-scale fin cage assembly was built from spare parts with the direct connection of a Raspberry Pi, BNO055 IMU, and servo assembly with reduction gears as shown in Figure 53. The results of running this Simulink function would be the exact same as tested with a ramp input. This mock-up provides positive assurance that the developed code responds as intended.

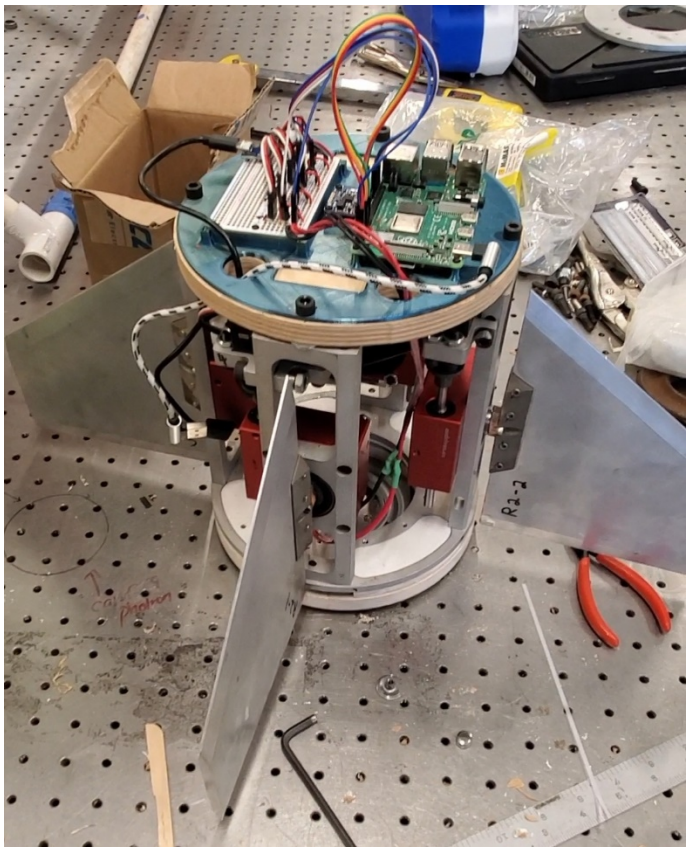

Figure 53. Fin Cage Mock-Up Mounted on a Rotating Bearing.

The final Simulink code would also implement two different maneuvers as shown in Figure 54 with the red box to the left. The first maneuver would implement a 22.5-degree roll at 1 second followed by a 90-degree roll at 3.25 seconds. Additionally, as shown in the red box on the right of Figure 54, the signal is required to be the negative value of the calculated as the reorientation of the servos and reduction gears caused a reverse in fin direction. Multiplying by -1 addressed the issue.

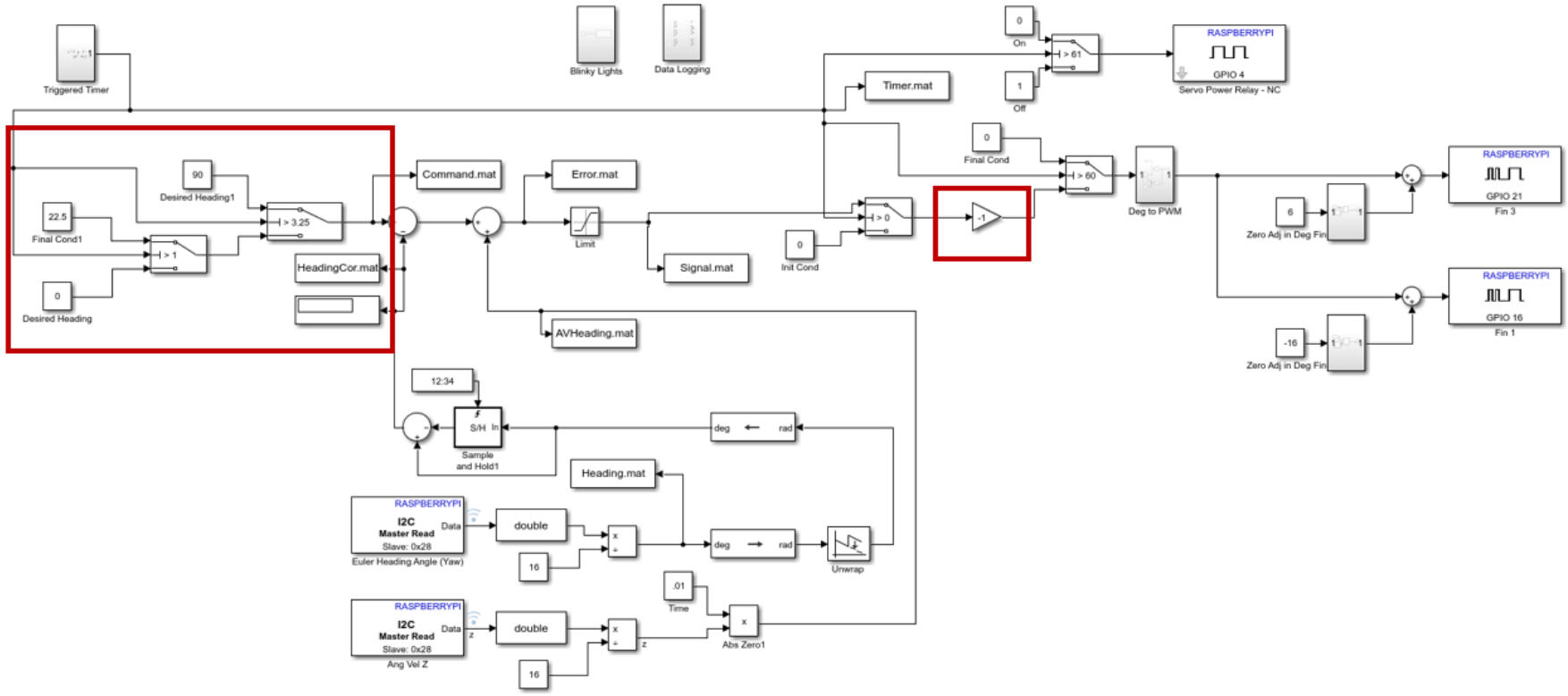

Figure 54. Rocket 3 Control Code.

b. Sled Coupling

Observing the premature Sled release occurring in both Rocket 0 and Rocket 1, the need to address the coupling became readily apparent as it also effected the results for recording video data. Several ideas were entertained from hinged latches to spring loaded barrel bolts, but the final design came down to a rotating planetary gear that would maneuver 4 steel pins through an interlock. Details are further discussed in Section V.B as the coupling is a key development over the span of the flight campaigns. The interim design is shown in Figure 55 from a top view on left and a rotated bottom view on right.

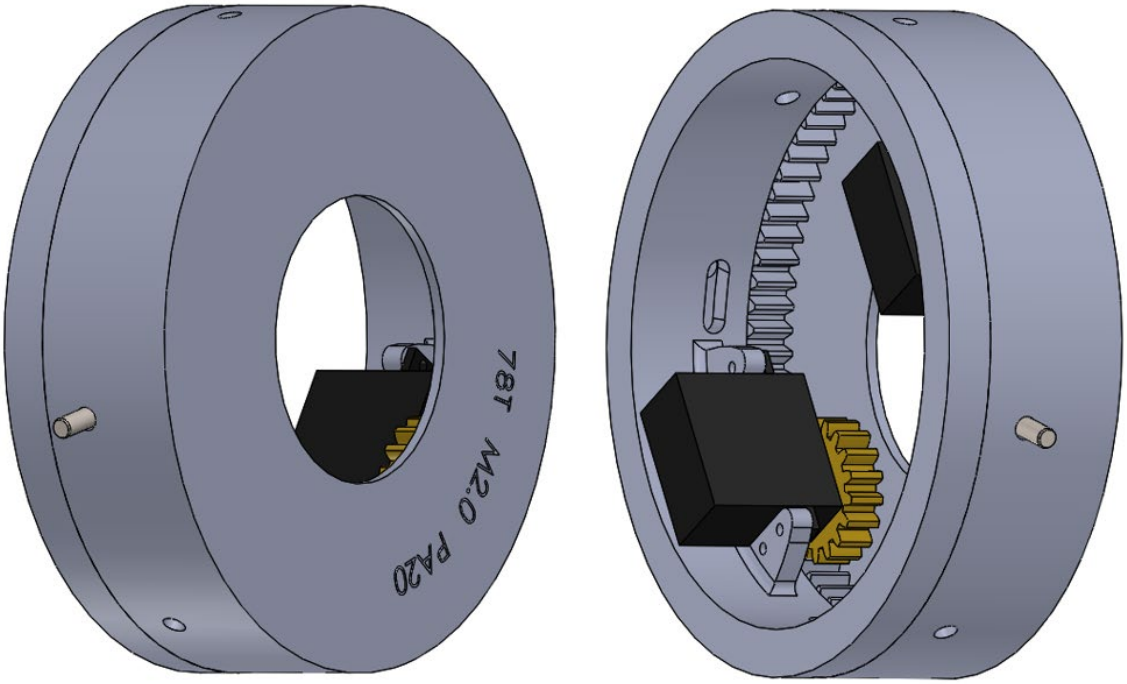

Figure 55. Sled Coupling Design.

The 4 steel pins interface with the rocket airframe as shown in Figure 56. While the airframe cuts were made by hand, the design is proven through the static load and release testing performed prior to implementation. The testing provided assurance that the airframe and mechanical features could withstand the expected loads experienced during drogue

deployment. The system was tested and failed with a suspended load of 1023 N (230 lbf). The weakest members were identified as the splines of the pinion gears connecting to the servos with details shown in Figure 57 (Left). The splines were made into an insert that was 3D Printed out of Titanium to eliminate the weak component as shown in Figure 57 (Center and Right).

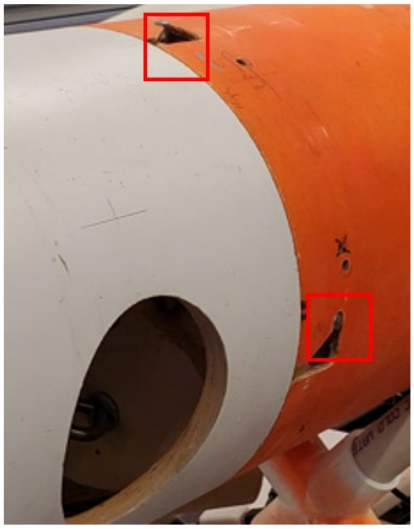

Figure 56. Airframe Cutouts for Pin Interlock Engagement.

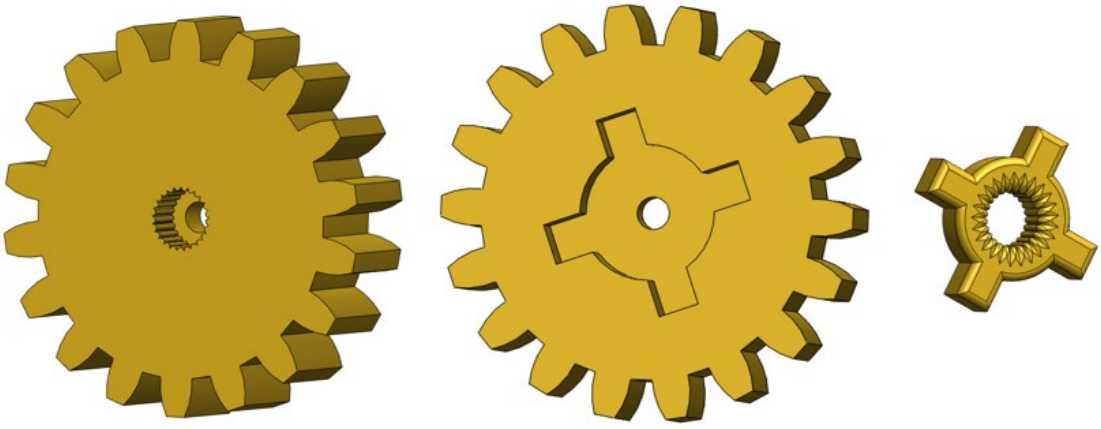

Figure 57. Initial Gear (Left), Gear with Cutout (Center), and Insert (Right)

The large hole in Figure 56 was created to insert a backup drogue parachute to release upon command. The backup parachute was deemed necessary as the experimental nature of the release mechanism had no method of manually releasing the main parachute. As such, the backup drogue parachute could be released, and the two drogues would have sufficient drag to ensure a safe landing of the Sled. The release of the backup parachute would be triggered from the auxiliary pyrotechnic channel on Kate 2.0 which would be initiated by the associated receiver. The current flows from the auxiliary channel to ignite an electronic match in a small canister of black powder. The pressure created from the explosion releases a cap held in place with shear pins as shown being tested in Figure 58.

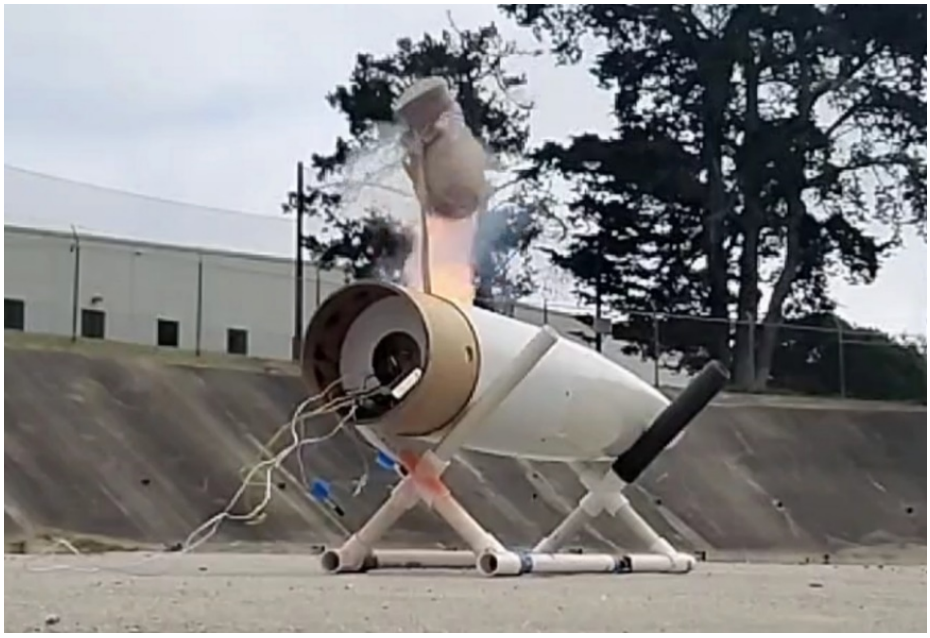

Figure 58. Testing the Backup Drogue Parachute.

c. Video

The video system was changed in Rocket 3 as the need for higher resolution video became apparent from Rocket 0 and Rocket 1. A decision was made to procure the Caddx Loris 4k Camera System, as shown in Figure 59, due to its ability to provide 4k resolution at 60Hz [53]. To preempt the design of a camera housing, the RunCam Split Mini 2 HD Camera was used due to its availability to use as a prototype with a similar form factor as

the Caddx Lorris. Unfortunately, the procurement failed due to backorder and the camera system defaulted to the RunCam which only allowed for 1080p resolution at 60Hz [54]. The benefit with using the RunCam is that it was able to be connected to an RF transmitter, enabling real-time capture of nose cone video feed.

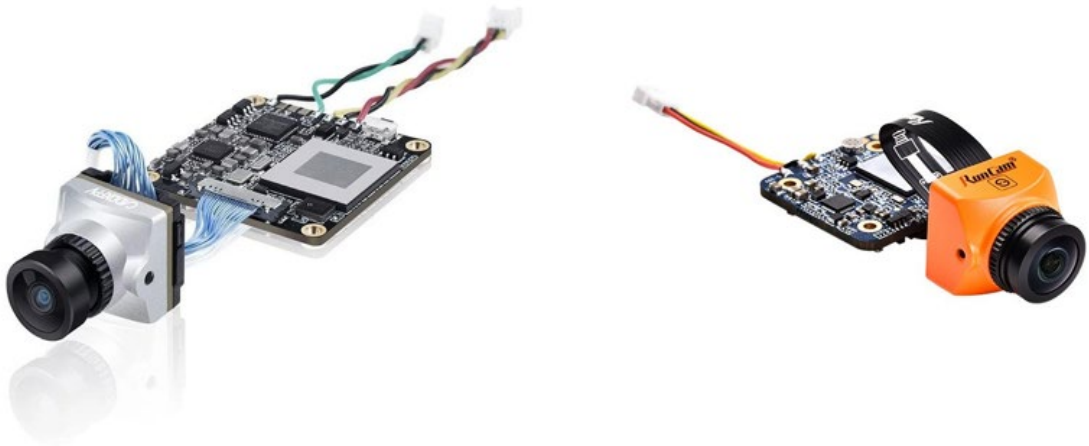

Figure 59. Caddx Lorris 4K Camera (left) and RunCam Split Mini 2 (right). Source: [53] (left) and [54] (right).

Seeing as though the launch for Rocket 3 was in the Mojave Desert during July, the camera housing was 3D printed out of Polyethylene Terephthalate Glycol (PETG) instead of Polylactic Acid + (PLA+), as had previously been done, due to its resistance to deformation at increased temperatures. While PLA begins deforming at 55 C (131 F) the PETG deformation is not seen until 75 C (167 F) [55]. Even though temperatures were not expected to be above 44.3 C (110 F), the effects of heat on PLA+ had already been realized at the launch of Rocket 2 in April where temperatures were in the 21–27 C (70–80 F) range. The resulting camera housing is shown in Figure 60, with color chosen to match the color scheme of the rocket.

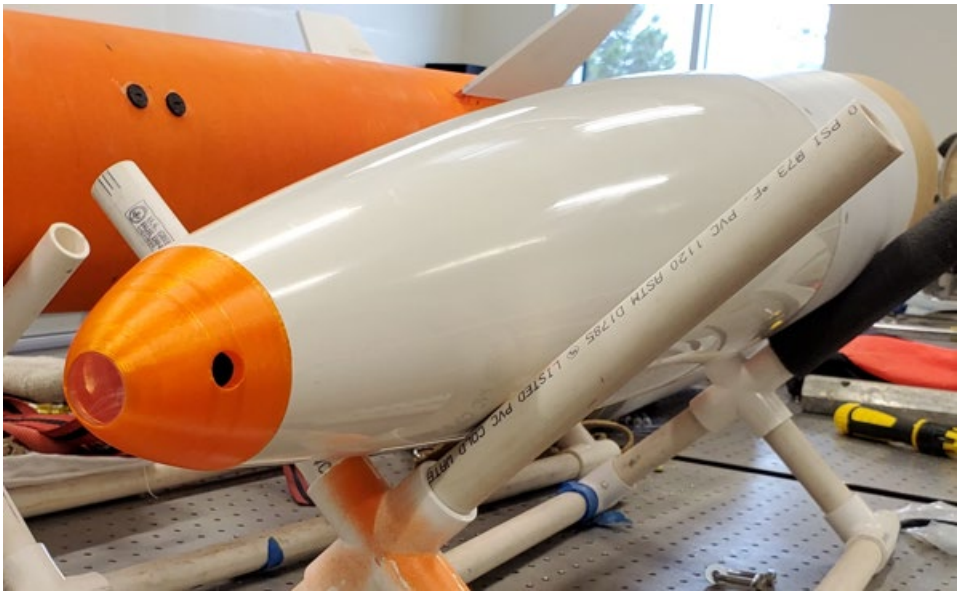

Figure 60. Rocket 3 Nosecone with PETG Camera Housing.

2. RESULTS

The third RRPD-V attempt was flown on July 17, 2021. While the flight was less than optimal, several lessons were learned.

a. Control

Rocket 3 did not have a successful launch with regards to controls. The SRM used for this rocket was the Cesaroni N2900, which was the only SRM available because shipping was delayed. While the N2900 was capable of providing a decent flight, the motor did not produce the same level of initial thrust as the M3400 [56]. Because of this, gravity played a large effect in pulling the nose of the rocket towards horizontal early in the flight. The motor was planned to reach approximately 2438 m (8000 ft), but instead only hit approximately 610 m (2000 ft). Additionally, the Simulink program was not properly deployed to record data, leading to the need to reconstruct using video feed, as shown in Figure 61, by using an overlay of a protractor as was required with Rocket 0.

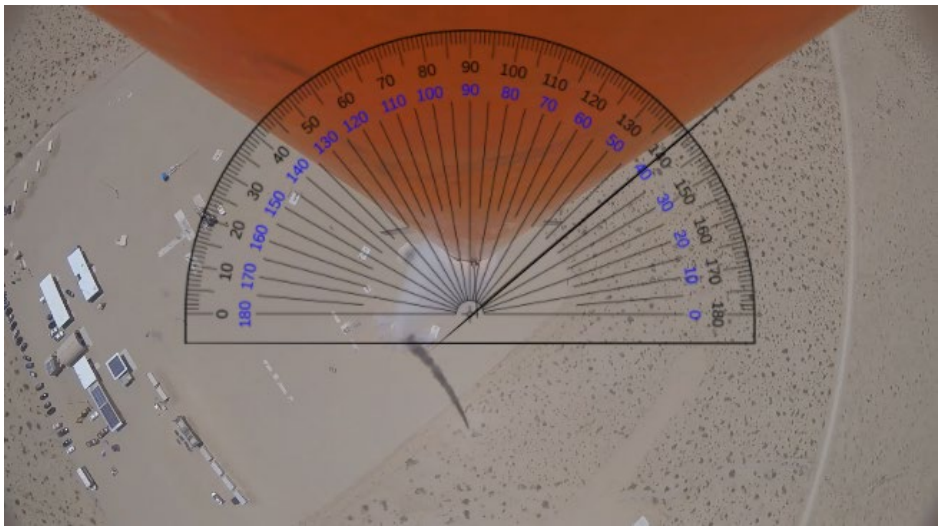

Figure 61. Reconstructing Rocket 3 Heading Information.

The data obtained from the reconstruction is poor as the flight did not stay stable for long. As shown in Figure 62, the rocket attempted to make a 22.5 degree turn at 1 second into the flight but was disrupted shortly afterwards due to the lack of stability within the flight. This is largely due to using a different motor than planned upon as the flight was intended for the Cesaroni M3400.

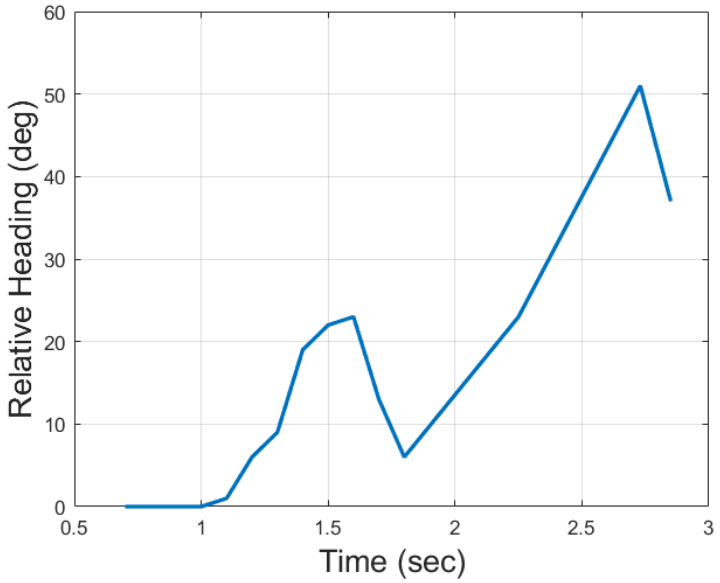

Figure 62. Rocket 3 Reconstructed Heading.

b. Sled Coupling

While the control went unstable, there was still success in that the rocket was able to stay together to allow for the newly designed Sled coupling mechanism to react. Similar to the deployed control algorithm on the Booster's Raspberry Pi, the Raspberry Pi in the nosecone experienced the same error and did not yield any data from the flight. Audio can be heard in a video camera placed inside the separation mechanism of the servos opening and CO2 being released, opening the Sled main parachute bay. Unfortunately, the drogue did not deploy properly, and the system was in a free fall due to the lack of a pulling force by the drogue. As can be seen in Figure 63, an amateur photographer was attempting to capture the team's reaction to failure as the Sled main parachute opened last minute, saving Sled and nosecone assemblies.

An additional problem with the system was that one of the steel pins became stuck on a CO2 canister as the release command was given. While this did not hinder the opening of the system, it did cause the servos to burn out. An easy method to circumvent this is to install a relay to turn off the servos 2 seconds after the command to release is given. This will be implemented in Rocket 4.

Figure 63. Rocket 3 Sled Main Parachute Opening Before a Hard Landing

c. Video

The nosecone mounted video camera in Rocket 3 did succeed in capturing video data at 1080p while in freefall. However, due to the failure of the drogue to deploy, the Sled fell sideways after the Booster and Sustainer separated. Additionally, as shown in Figure 64, the Orange PETG used for the camera housing was translucent, allowing an orange halo to degrade the footage. Drogue deployment would prove to be vital in Sled operation.

The video receiver also failed to operate properly during launch as the video was unable to be recorded on the handheld receiver. The transmitter was able to be received but only while on the ground. As soon as the rocket was in the air, the video feed diminished. Possible interference from the 250mW transmitter on the Kate 2.0 transmitter is thought to be a cause of the attenuation but the test will be performed again in Rocket 4 to verify the results.

Figure 64. Nosecone Camera View of Booster Deploying Main Parachute.

D. ROCKET 4

1. Development

The multiple failures seen with Rocket 3 prompted a quick turnaround of the system to be flown only a month later in August. Several changes were made to the internals of the CO2 systems to ensure vehicle flight success could be obtained.

a. Control

The control algorithm that was used in Rocket 3 was considered likely to have produced proper results if the Cesaroni M3400 was used vice the N2900 due to the M3400's higher initial thrust and lower expected dynamic pressures, as discussed in Section 4.C.2.a. With a new M3400 delivered at the launch site, the code was altered slightly to remove the second maneuver from Rocket 3; only providing a roll to 45 degrees at 1 second into flight to simulate a step input so that the response could be used to model a proper controller.

Additionally, the freefall of Rocket 3 identified the need to be able to secure the running programs while in flight, as the sudden removal of power could cause data corruption. While Simulink contains a 'Stop Simulation' function, it only works while simulating code on a laptop and not while deployed. As a workaround, two GPIO pins on the Raspberry Pi were used to accomplish the task. The first GPIO pin demands a high signal (1) while the control algorithm is running via the deployed Simulink model. When the controller is to be shut down, the GPIO pin is forced low (0) to signal the transition. A separate GPIO is sensing the first via a direct connection in a script running in Python, as shown in Appendix E, on the Raspberry Pi. When the second GPIO senses the first GPIO go low, the Python script forces the running control process to shutdown properly and secure the data. As shown in Figure 65, a 100kΩ resistor is required in parallel with the GPIO and a ground source to ensure the GPIO pin's setting does not drift from low to high as during benchtop tests. This setup is also replicated on the Raspberry Pi for Sled Coupling control.

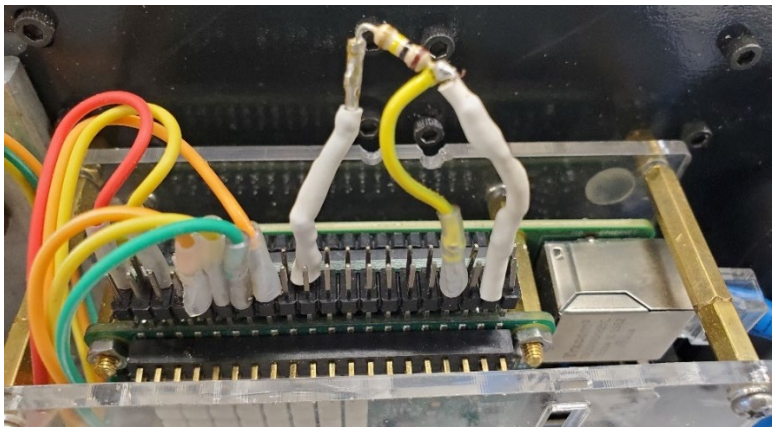

Figure 65. Raspberry Pi GPIO Pin Extender with Pulldown Resistor.

b. Sled Coupling

Several lessons learned were derived from Rocket 3 with the deployment of the Sled's main parachute. First, the Sled's drogue parachute is required to provide drag to enable the release of the main parachute after the CO_2 has been released. The Sled's drogue parachute did not deploy properly during Rocket 3's flight. The failure to deploy was determined to be that the drogue parachute did not have any method of forcing the parachute out of the Sled. The drogue parachute was corrected by placing an additional CO_2 behind it to allow a force to push the small parachute away from the Sled. Second, the CO2 that provides separation of the Sled Coupling was releasing and pushing the main parachute back into the airframe. To address this, the CO2 system from the Sled coupling was relocated to push the parachute out of the Sled's main parachute bay.

While the mechanics of the sled coupling assembly worked flawlessly during the flight, some damage was observed on the Sled airframe as shown in Figure 66. This was caused by the steel pin's rapid release and required replacing. The damaged portion of the coupling was able to be cut off with the intention of making a coupler that would add the extra length required for the parachute bay and provide more strength for the pin activation. The quick turnaround of the rocket did not allow for the machining of an aluminum part or the building of a new Sled; therefore, 3D printing was utilized to create a new mating surface for the coupling.

Figure 66. Damaged Sled Coupling Removed from Sled.

The need for the coupling to have strength is apparent and led to the selection of material to print the coupling out of. PLA+ was considered as its toughness is impressive,

but the temperature in the Mojave Desert was still expected to be around 38C (100F) and the material would have deformed in the heat. Additionally, PETG was considered for its ability to withstand deformation, but testing did not allow for it to withstand shear stress between its printed layers. The final selection was made to go with Polycarbonate filament as the material is able to withstand high temperatures and has higher strength and toughness than PLA+ [57]. As shown in Figure 67, the Polycarbonate sleeve slides over the airframe outer diameter and has the same inner diameter allowing for the extension of the airframe to make up for the lost length due to damage.

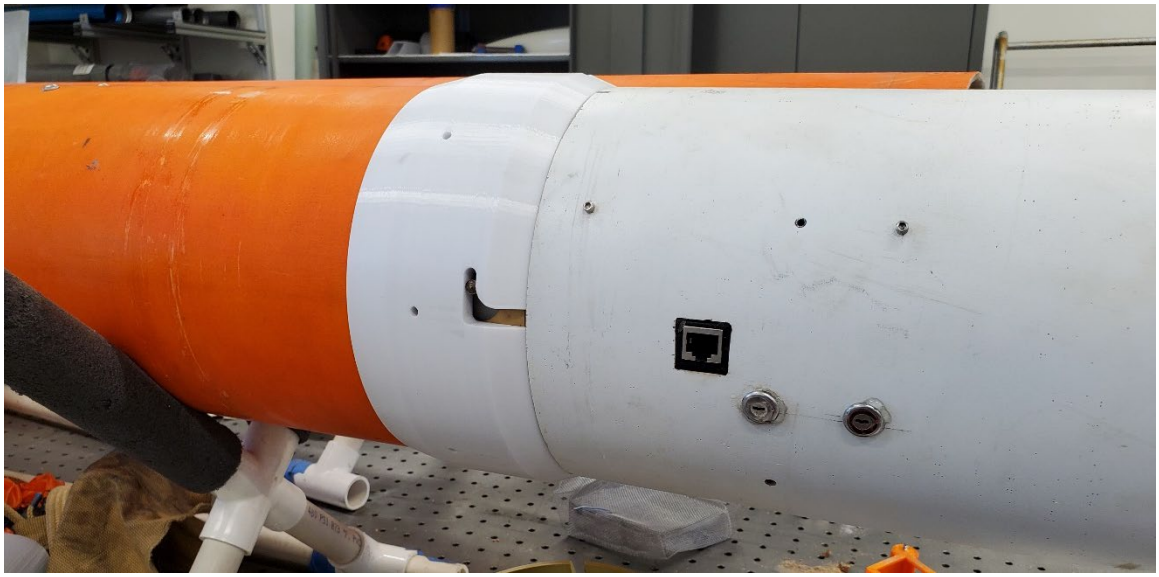

Figure 67. Polycarbonate Coupling Installed on Sled.

c. Video

The video imagery was not satisfactory from the flight of Rocket 3 due largely to the drogue not deploying, resulting in the Sled and nosecone freefalling horizontally with respect to the ground. Additionally, an orange haze could be seen in the video from the translucent orange PETG material used to create the housing. The drogue issue was already corrected as discussed previously, but the camera housing needed reprinting. The white Polycarbonate used for the coupling was too reflective to be used for the housing; therefore, black PETG with embedded Carbon Nanotubes was chosen as it also provides electrostatic

discharge protection to further aid in eliminating any EMI to the video. The final product can be seen in Figure 68 as installed on the nosecone.

Figure 68. PETG with Carbon Nanotube Camera Housing.

2. Results

Rocket 4 was a large success as all pieces of the system performed as expected except for the Booster main parachute. Upon release, the main parachute tangled with the drogue leading to a hard landing for the Booster. While the landing was less than optimal, there was minimal damage to the Booster, allowing for another flight to occur. The damage that did occur was in the avionics bay as the Polycarbonate structures holding the electronics shattered under the force of impact. This was a minor setback as all data was able to be retrieved from the Raspberry Pi. The flight occurred on August 21, 2021.

a. Control

A Cesaroni M3400 SRM was used to propel Rocket 4. The M3400 was chosen because it previously provided enough thrust to allow for steady performance, maintaining a subsonic rocket speed. While the power was prematurely removed from the Raspberry Pi due to the Booster impacting the ground at a high rate of speed, the corrupted data was again able to be preserved using the MATLAB function described in Appendix D. The

preserved heading data was able to describe the response of a proportional controller to the flight system after failures in previous launches. As seen in Figure 69, the system has a response that is underdamped with the exception of perturbations caused by wind gusts. Additionally, the Full Width at Half Maximum (FWHM) becomes more pronounced in the flight while the dynamic pressure is decreasing. The data will prove vital in being able to determine the unknown constants in the system model, allowing for the design of a PD controller on Rocket 5.

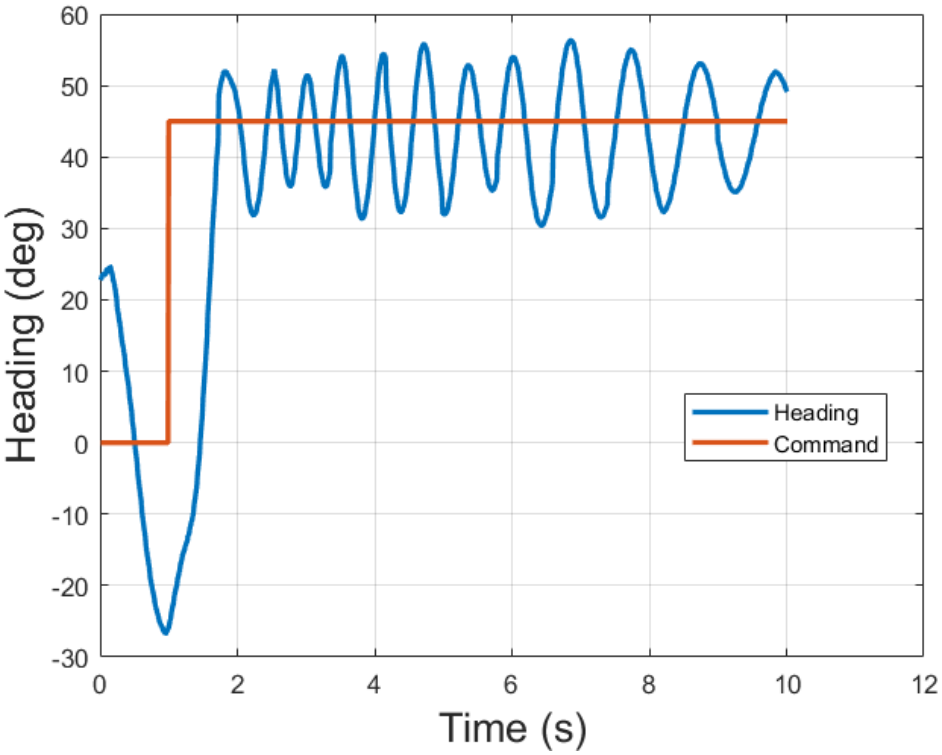

Figure 69. Rocket 4 Response to 45 Degree Demand.

b. Sled Coupling

The sled coupling worked flawlessly with its new Polycarbonate 3D printed sleeve. With all components flight proven, the Sled Housed Activation and Release Device (SHARD) can now be readily reproduced as all mechanical pieces are 3D printed with Polycarbonate and Titanium. As shown in Figure 70, the SHARD allowed for releasing of

the nosecone from the Sled and the expected quick releasing of the Sled main parachute. The relocation of the CO2 system allowed for the separation to occur while simultaneously pushing the parachute out of the Sleds parachute bay. Additionally, the CO2 cylinder that provided force to the drogue parachute to release when required, providing the drag necessary to allow for the Sled coupling to separate and for the obtaining of nosecone video.

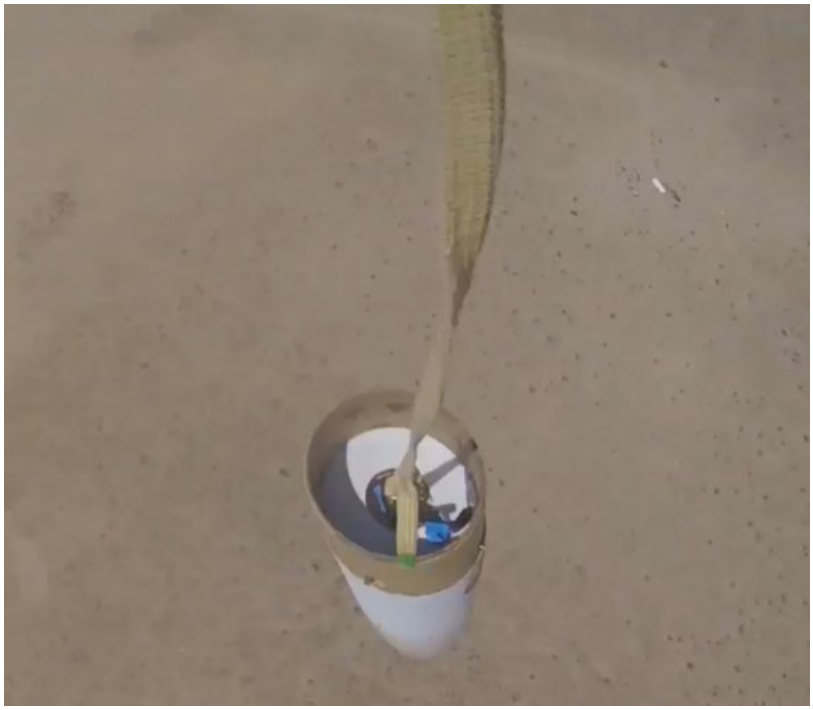

Figure 70. View of Nosecone Separating from Sled.

Additionally, data from the Raspberry Pi Sense HAT and the SHARD are compared to the Perfect Flight data to identify how the systems are working in tandem. As can be seen in Figure 71, the systems are within 15 m (50 ft) of each other at maximum separation, providing a benchmark in accuracy of the sensors for programing purposes.

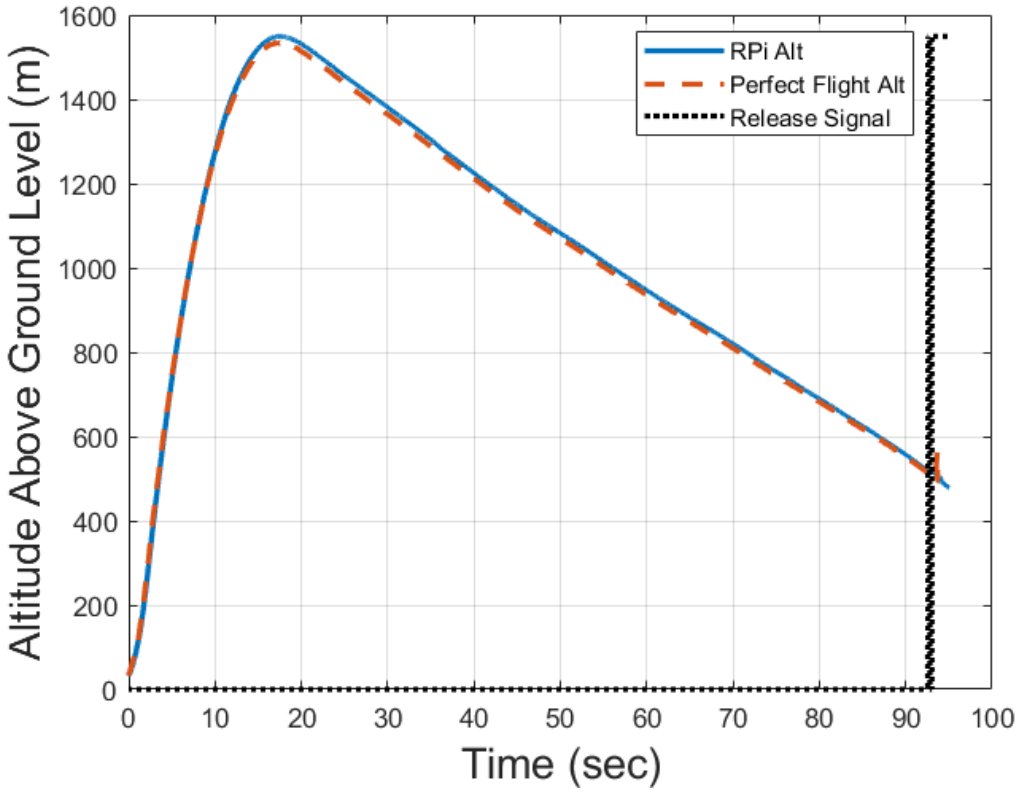

Figure 71. Raspberry Pi and Perfect Flight Data Compared.

c. Video

Rocket 4 was able to obtain successful video footage for the first time due to the advancements in the Sled Coupling and in the forced deployment of the drogue. As shown in Figure 72 from 1220m (4000 ft), the video system was able to obtain clear footage of the desert floor while suspended from a drogue parachute. Even though the video is not 4k resolution, major improvements can be seen from Rocket 0s video where the nosecone hung from the main parachute. First, there is no visible EMI due to the shielding over the camera system and its wires. Second, the drogue parachute allows for higher velocities and stabilizes the system as it is descending. In comparison, Rocket 0 had several rotations per second and Rocket 4 showed an average of one rotation every five seconds. Rocket 4s angular velocity is much lower than that of Rocket 0, showing marked improvements on the system.

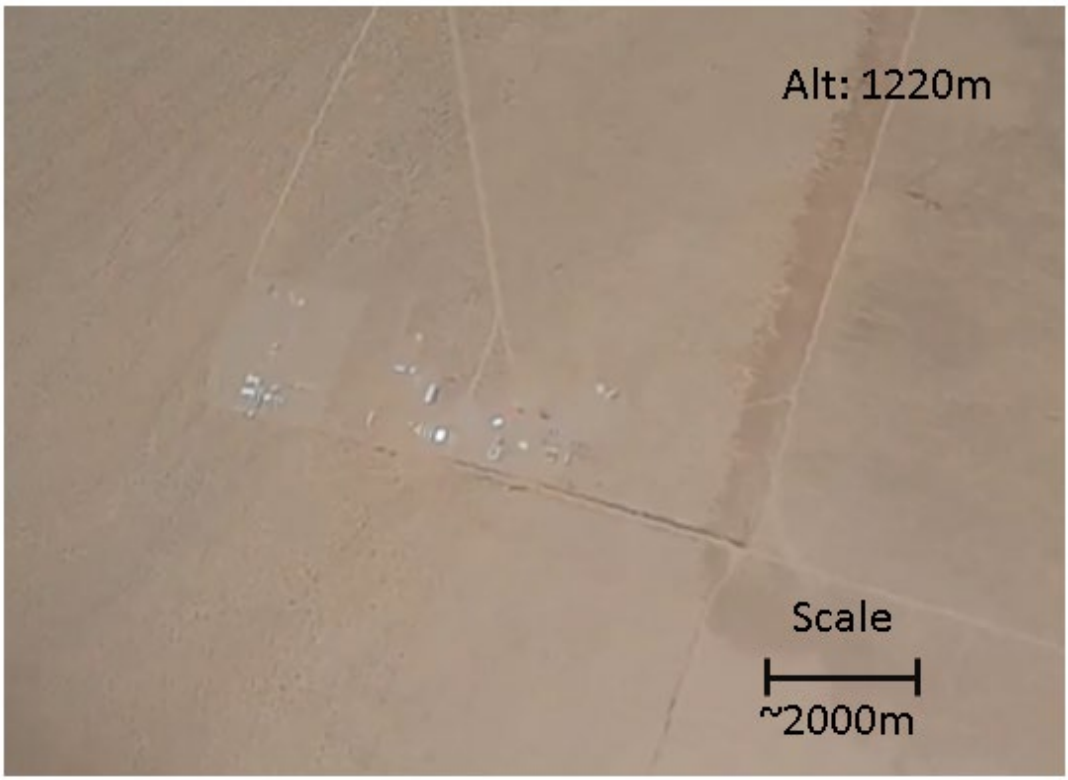

Figure 72. Nosecone Video.

E. ROCKET 5

1. Development

The development of the fifth and final RRPD-V implements a different design for the Interstage Coupler main parachute release bay. An additional coupler, designed by Grohe, was used to provide additional structure stiffness [28]. The coupling design used in rockets 1 through 4 was seen as a source of deflection in the rocket body during flight, resulting in structural failure for Rockets 2 and 3 and likely non-vertical climbouts from an induced lift with the bending moment. This weak point prompted a search for a reliable design. The hope was that the coupling replacement would remove undesirable bending moments, leaving more chance for successful implementation of controls, proper release of the Sled Coupling, and to allow for attaining video feed from the ballistic flight of the nosecone. While overall success would not be realized, the individual successes of each subsystem will be identified.

a. Control

The successful launch and retrieval of data from Rocket 4 allowed for several breakthroughs with the control algorithm. Rocket 4 was able to deliver data showing the proper development of a proportional control system. With the data available, its characteristics were analyzed to define a model of the rocket's roll response and develop a proportional derivative (PD) controller. Furthermore, the Raspberry Pi was overclocked to increase the speed of the system as previously deployed programs have lagged. Additionally, a potentiometer will be used to directly sense a fin's position during flight, allowing for the characterization of the lag in the fin position from the commanded position.

The scheduling of controls required consideration for this flight as the SRM for this rocket is the Cesaroni N3301 with much higher dynamic pressures than the M3400 used previously. Because of the higher expected dynamic pressures, the control algorithm would only be implemented while the rocket was in the M3400 dynamic pressure range and set to zero the fins during the transient. Using data obtained by Alexis Thoeny from simulations run in OpenRocket, a rocket flight simulation software, the time frames correlated to only using the control algorithm between 0 and 3 seconds and from 12 to 20 seconds as shown in Figure 73.

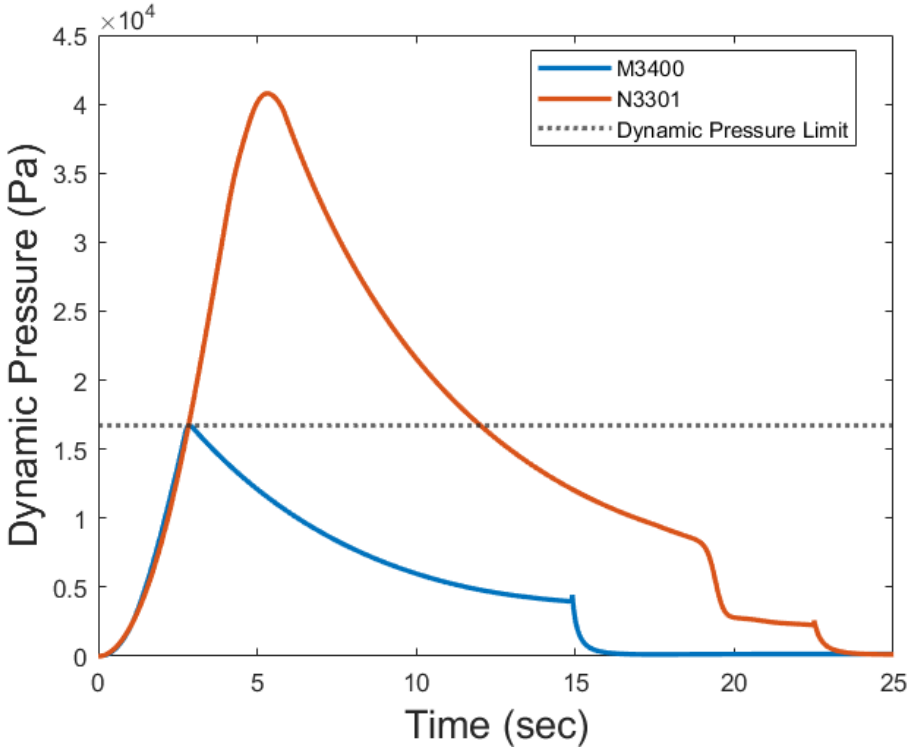

Figure 73. Dynamic Pressures of M3400 and N3301 for Rocket 5.

(1) PD Control

The results of defining a PD controller are developed in Sections V.A.1 and 2, as they are key developments in the RRPD-V system. Results of the designed PD controller offer an expected step response as shown in Figure 74. While the response is aggressive, the rocket moves so fast that microseconds must be accounted for, resulting in a desire for a fast response.

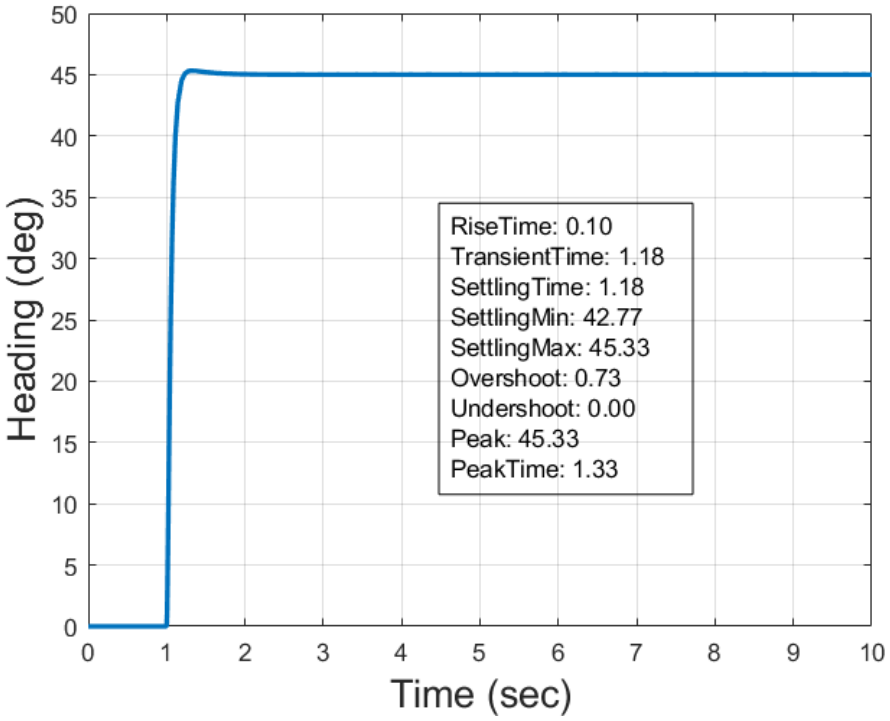

Figure 74. Designed Step Response with PD Controller.

(2) Overclocking

On previous rockets, the deployed control algorithms had pushed the Raspberry Pi to its limits, causing slight delays in the system clock. This was mostly realized by watching video of the system running and events that should take place at 10 seconds would occur at 11 or 12 seconds. While this is mostly negligible, the attempt was made to increase CPU speed and provide more CPU voltage by overclocking the Raspberry Pi. To achieve this, two lines of code were added to the Raspberry Pi config.txt file as shown in Figure 75. The first line increases the voltage to the CPU to the maximum level allowed without voiding warranty and the second line sets the CPU to 2Ghz vice the standard 1.5GHz [58].

```
over_voltage=6
arm_freq=2000
```

Figure 75. Config.txt Added Code for Overclocking. Source: [58].

(3) Fin Position

An effort was made to characterize the response of the rocket's fin demand to its actual position. Such data would result in characterization of phase-lag for hardware in the loop. This was accomplished through the coupling of a potentiometer to the direct output of the servo. As shown in Figure 76, a mount was designed to align the potentiometer in the correct position on a test rig to mimic the mount on the rocket. The changes in voltage caused by the rotations of the potentiometer would be read by the ADS1115 Analog Digital Converter (ADC) and saved to a file on the Raspberry Pi. The signal would then be post processed to allow for the direct comparison of demanded to actual fin position.

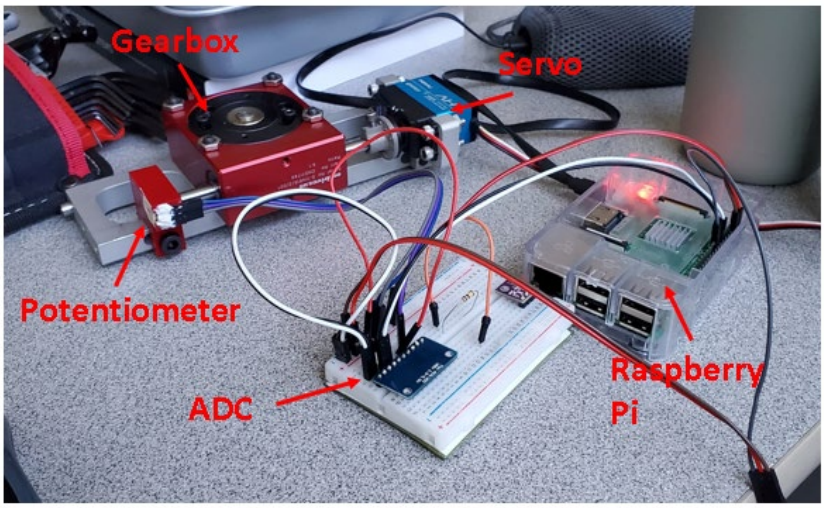

Figure 76. Potentiometer and ADC Experimental Setup.

b. Sled Coupling

Sled Coupling Design was not altered in Rocket 5 from the design in Rocket 4. The SHARD was inspected and cleaned post-flight from Rocket 4 and the system proved to be fully operational. As such, no changes were made and a third attempt to prove the system's effectiveness would be attempted.

c. Video

The video camera would not be changed with Rocket 5 as the 4k resolution cameras were not received with enough time from the manufacturer. Furthermore, the video RF transmitter was removed from the system as it provided little value to the test objectives and instead created more heat than could be removed. The major change that would be made would be delaying the drogue parachute in the Sled from deploying until 5 seconds after apogee to gather video during ballistic flight of the Sled assembly. This required the testing and development of a cap system to contain the drogue until deployment was desired. As seen in Figure 77, carboard from a moving box was proven capable to hold and release the drogue via a CO2 charge activated at 5 seconds following apogee.

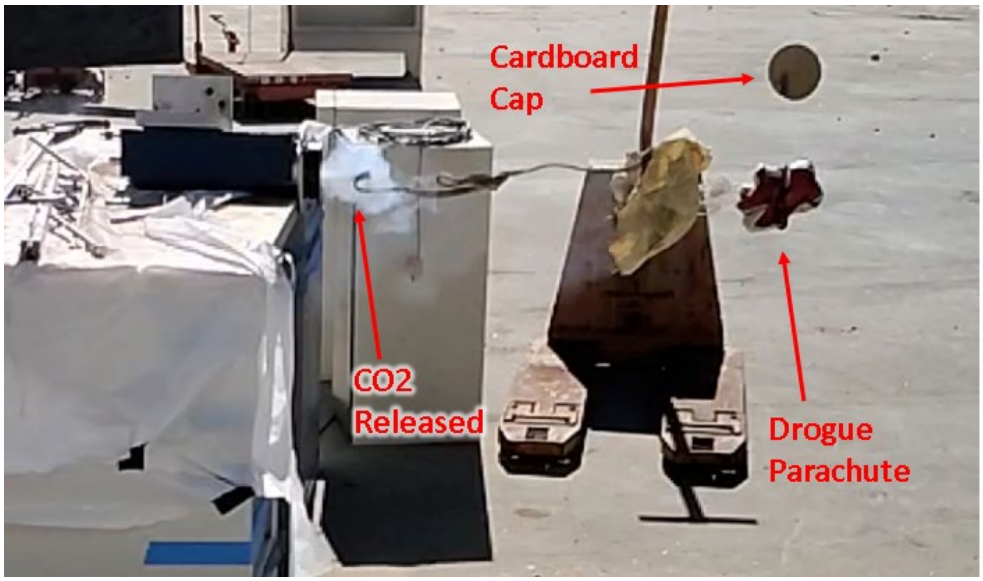

Figure 77. Testing of Drogue Containment Cap.

2. Results

The final flight of the RRPD-V was a total loss. Approximately 5 seconds into flight, the system became unstable and suffered from a failure at the section coupling the Interstage Coupler and the Booster sections of rocket. The Booster main parachute attempted to deploy at over 183 m/s (600 ft/s) and was immediately ripped free. The rocket underwent a freefall from approximately 1220 m (4000 ft) and impacted the ground with

zero mercy, as shown in Figure 78. Post analysis would show that the aluminum couplers bound when the rocket ripped apart, making separation near impossible for the Sled drogue parachute to deploy. All electronics were destroyed, and all data was lost except for altimeter data on four of five PerfectFlite systems. Kate 2.0 downlink transmitted data was preserved but the lack of resolution does not allow for any useful analysis.

Figure 78. Rocket 5 Booster After Hard Landing.

a. Control

The aftermath of the hard landing led to an ejection of the Raspberry Pi's SD card to an undisclosed location on the desert floor. To attempt to reconstruct any flight data, the externally mounted GoPro video camera footage would be used with a protractor overlay, as shown in Figure 79. Because the camera is fixed to the rocket body, the fixed edges of launch pads are used as references and the data is reconstructed as shown in Figure 80.

Figure 79. Reconstructing Rocket 5 Heading Data.

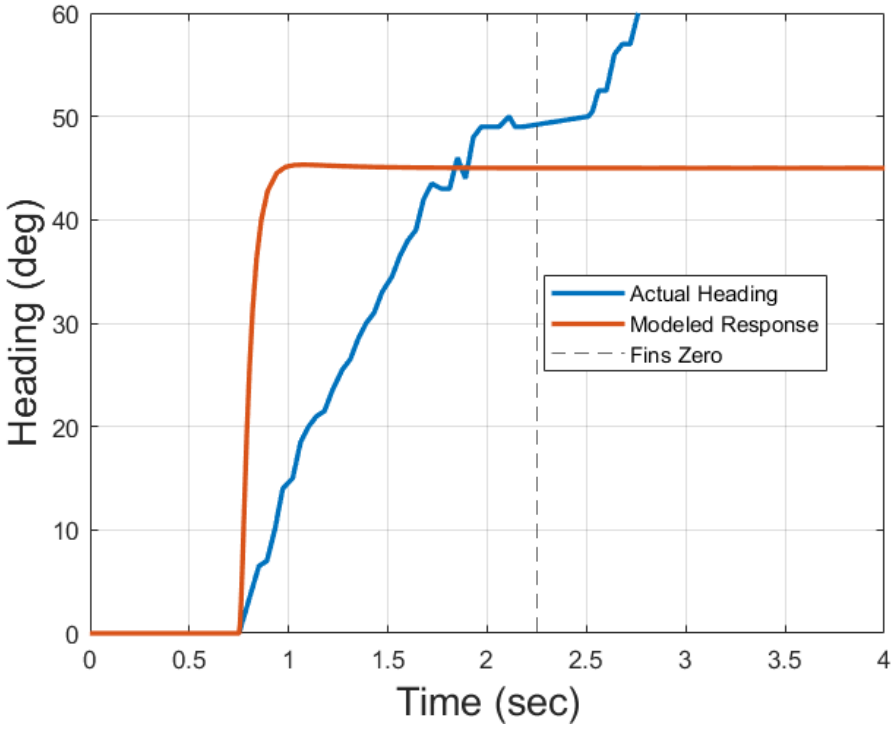

Figure 80. Rocket 5 Reconstructed and Design Heading.

The heading can be seen to settle briefly on the commanded 45 degree heading and then starts to drift off before the heading becomes unstable around 3 seconds. One issue to

note is that the fins turned off at roughly 2.25 seconds into flight and turned back on approximately 7.5 seconds into flight instead of 3 seconds and 10 seconds as stated in the Simulink coding. The discrepancy correlates to the ratio of overclocking the CPU from 1500 to 2000 MHz as it turns out that the default timing for the Raspberry Pi is established by counting CPU cycles for timed events in embedded codes [59], thus the increased CPU speed would inadvertently speed up the clock.

Additionally, upon post processing of the data, an error was discovered in the design of controller in that the damping ratio was calculated incorrectly. The specifics are discussed in Section V.A.3, but the results do not yield a major change in the system response. The new response can then be seen with the data reconstructed, as shown in Figure 81, to prove that the actual response is closer to the newly calculated response. Had the fins been allowed to continue the maneuver, the rocket's response should have mirrored the expected step response.

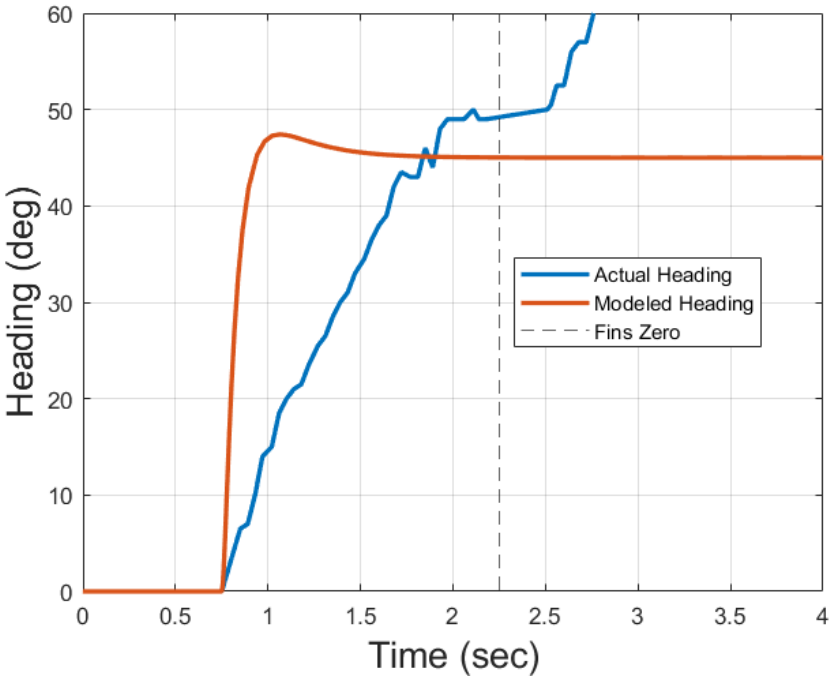

Figure 81. Rocket 5 Recalculated Response (ζ=0.99) with Heading.

b. Sled Coupling

The SHARD was completely destroyed on impact as shown in Figure 82. While post flight inspection shows that all CO2 charges were properly blown, the mechanism failed to release the Sled main parachute as intended. The servos were still in fairly good condition, so their positions were preserved until they could be identified at the RPL. The servos were reinstalled on a Raspberry Pi and given the command to open with very little movement in response. The servos were then cycled to assure they were operational. Due to the servos being in the open position, it can be speculated that the release latches opened, but it was not enough to cause separation. A difference in this flight compared to others is that the parachute inside the Sled's main parachute bay was much smaller than the parachute previously flown. The smaller parachute left more open volume in the bay which then absorbed the CO2 that was meant to cause a differential in pressure. The CO2 release not being as strong as intended coupled with the failed release of the drogue parachute did not provide enough axial force for the separation of the assemblies.

Figure 82. Rocket 5 Sled After Impact.

The installed backup drogue parachute was deployed as commanded by the user on the ground in attempt to save the system. The force of the drogue parachute deployment

while falling near 48.8 m/s (160 ft/s) caused the nosecone to shear and break in half at approximately 152 m (500 ft) AGL. The wooden bulkhead shown in Figure 82 was the mounting point for the backup drogue parachute.

c. Video

The video camera in the nosecone was also thoroughly destroyed, as shown in Figure 83. The failure of the Sled drogue and main parachute left the system with no chance of survival. The SD card was found installed in the camera; however, it was cracked and was not able to provide any data.

Figure 83. Rocket 5 Nosecone After Impact.

THIS PAGE INTENTIONALLY LEFT BLANK

V. KEY DEVELOPMENTS

Due to the multiple flights over the course of a year and plethora of data available from each, the key components that have been developed are displayed throughout this chapter. As such, the development of rocket control models and the design of the SHARD are described.

A. MODELING SYSTEM CONTROL

1. Roll Model

To properly understand the roll of the rocket during flight, the constants of the transfer function need to be identified to complete a model. As developed in Chapter 2, the open loop transfer function is shown in Equation (4) from Section 2.A.1.

$$\frac{\Theta(s)}{T(s)} = \frac{1}{I_{zz}s^2 + ds} \tag{4}$$

Closing the loop on the transfer function and providing a proportional term updates the transfer function to represent the control that was provided in Rocket 4. Graphically, the system can be seen in the Simulink model as shown in Figure 11and mathematically as shown in Equation (5) from Section 2.A.1 where K_p is the proportional gain, I_{zz} is the Moment of Inertia along the z-axis, and d is the damping coefficient.

$$\frac{\theta(s)}{T(s)} = \frac{\frac{K_p}{I_{zz}}}{s^2 + \frac{d}{I_{zz}}s + \frac{K_p}{I_{zz}}} \tag{5}$$

With the closed loop transfer function known, the constants can then be compared to a traditional second order system for further analysis as displayed in Equation (16) where ω_n is the undamped natural frequency and ζ is the damping ratio.

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n^2 s + \omega_n^2} \tag{16}$$

The following relationships can be identified by comparing Equations (5) and (16) , then solving for the parameters of interest:

$$\omega_n = \sqrt{\frac{K_p}{I_{zz}}} \tag{17}$$

$$d = 2I_{zz}\zeta\omega_n \tag{18}$$

Since the reduction gears from the servo to the fin were 10:1, the proportional gain will be set to 0.1. The rocket shape will be modeled as a solid cylinder whose moment of inertia about the primary axis is described as shown in Equation (19) where m is the mass and r is the radius of the rocket [60].

$$I_{zz} = \frac{mr^2}{2} \tag{19}$$

With the mass, radius, and proportional gain known, the undamped natural frequency can be solved for as follows:

$$\omega_n = \sqrt{\frac{2K_p}{mr^2}} = \sqrt{\frac{2(0.1)}{(45.36kg)(0.0984m)^2}} = 0.6746\frac{rad}{s}$$

To identify the damping ratio, an attempt was made to use the response of Rocket 4 with the definition of logarithmic decrement, δ [61], as shown in Equation (20) and illustrated in Figure 84, where X is the peak of the underdamped response and T_d is the time between peaks.

$$\delta = \ln\frac{X_1}{X_2} = \zeta\omega_n T_d \tag{20}$$

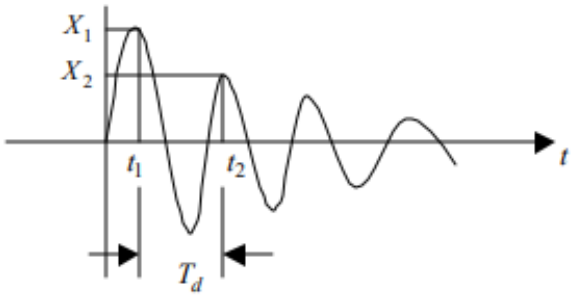

Figure 84. Successive Peaks for Logarithmic Decrement. Source: [61].

Solving for the damping ratio yields the following relationship:

$$\zeta = \frac{\ln \frac{X_1}{X_2}}{\omega_n T_d} \tag{21}$$

Observing the response from Rocket 4 as seen in Figure 85, the two successive points used for this analysis are shown as they are the points that correspond to the motor burnout and the maximum dynamic pressure.

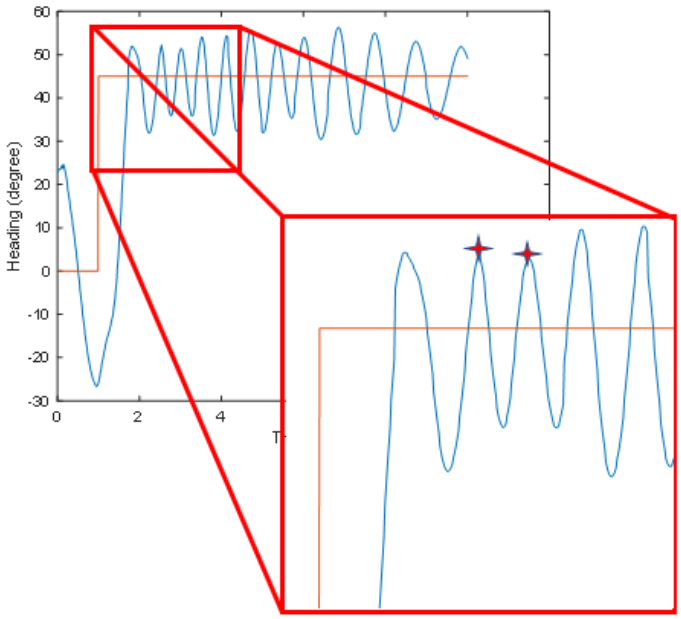

Figure 85. Rocket 4 Response with Points for Logarithmic Decrement.

The damping ratio can then be calculated and indicates that the system is actually overdamped, though it is not apparent from Rocket 4's response as the wind was gusting over 20kts as indicated by a weather sock.

$$\zeta = 2.42$$

With all values identified, the modeled response of the system can be observed and compared to the actual vehicle response from Rocket 4. To implement the modeled system torque must be calculated from the commands given to the servos from the actual flight. Torque can be calculated by using Equation (11), using flight data from Kate 2.0 to obtain air density and vehicle velocity. where α is the angle of the control fin, ρ is the density of air at the vehicle's altitude, v is the vehicle's velocity, s is the surface area of the fin, and c is the moment arm of the fin to the center of the rocket.

$$T = 2\pi\alpha\rho v^2 sc \tag{11}$$

Following the tuning of the system to implement the proper moment arm length, the comparison of the modeled system and observed system are shown in Figure 86.

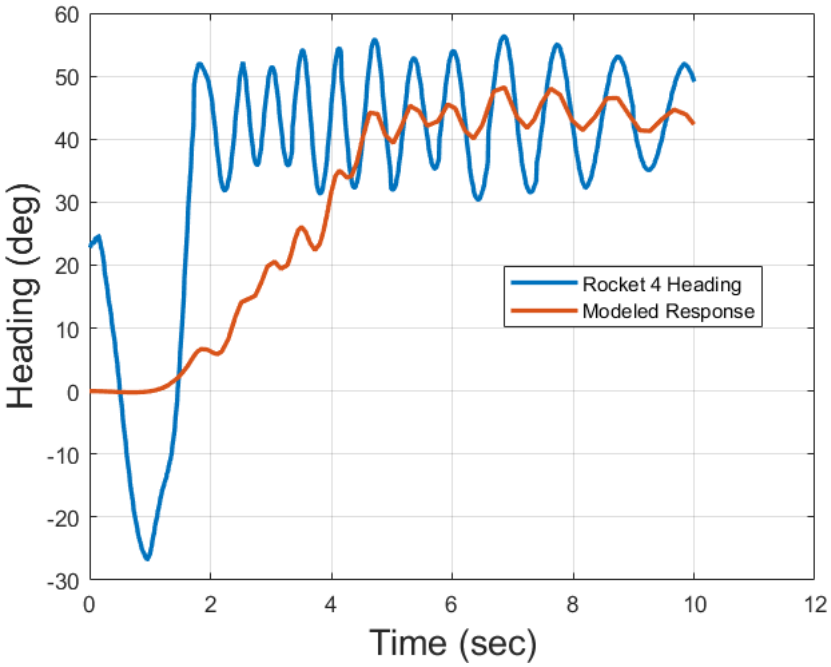

Figure 86. Rocket 4 Model Compared to Recorded Data.

As can be observed, the peak magnitude of the response does not appear to be the same as the actual flight heading, but the observed trends are similar in nature as the FWHM are the same. While not perfect, the model is sufficient to prepare a controller for Rocket 5.

2. PD Controller

To implement derivative control into the system, Simulink's PID Controller will be used to properly tune the system. As shown in Figure 87, the PD Controller block replaces the proportional gain block, and a step signal replaces the input.

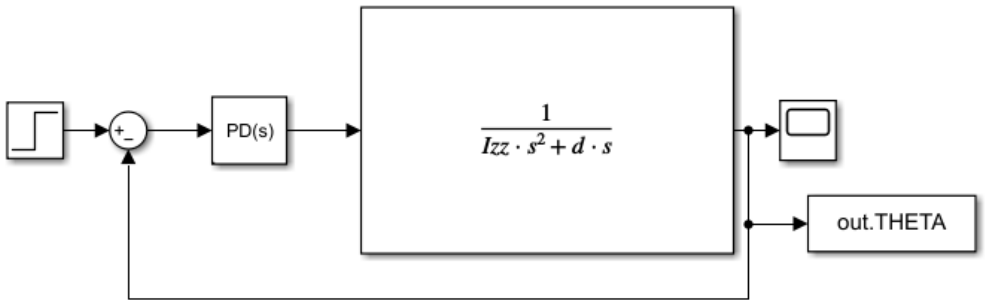

Figure 87. Roll Model with PD Controller

The PID Tuner is then used to create the appropriate response. As shown in Figure 88, the model is tuned to be close to critically damped to minimize oscillations during actual flight. The gains that will be selected for use are shown in Table 1 with an included filter coefficient that will be used in a lowpass filter. The lowpass filter is used to remove high frequency noise from the signal [62]. As there is not much noise expected from the BNO055 due to its Kalman filtered output, the value is set low.

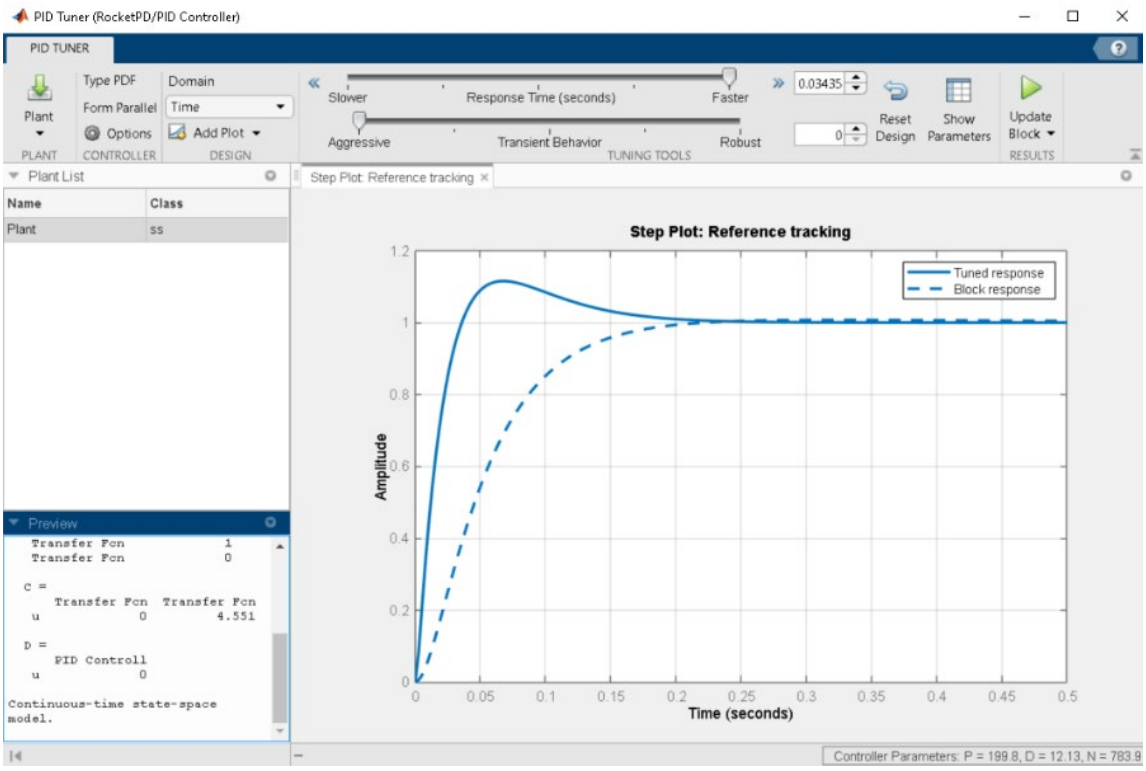

Figure 88. Simulink PID Tuner for Desired Response.

Table 1. Rocket 5 Gain Parameters.

Parameter	Value
Proportional Gain	13.39485
Derivative Gain	3.619978
Filter Coefficient	100

The gains must now be implemented into the Simulink model that will be deployed to the Raspberry Pi. To implement the gains, an attempt was made to utilize the PD Tuner Block in the deployed system; however, the attempt failed as the system would not run. Implementation of the gains [63] are done, as shown in Figure 89, by using the error signal as the sole source. The major change to the system is the removal of the angular velocity term from the input of the system, as it was realized that it is better to integrate or derivate the noise associated with one sensor than to have the additive effects of noise from two sensors: a lesson from Dr. Mark Karpenko in AE3830.

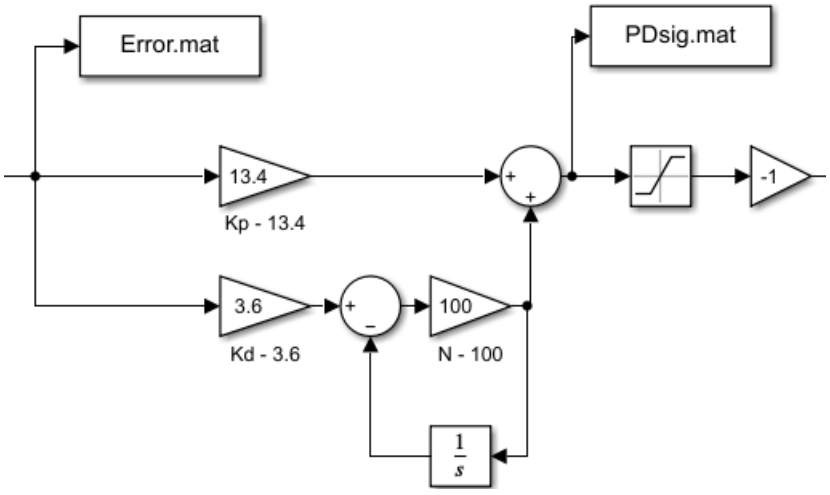

Figure 89. Implementing PD Controls in Deployed System.

3. Model Error

Upon post processing of the data from Rocket 5, an error was discovered in the design of controller in that the damping ratio was calculated incorrectly. While the error was large, it was not caught ahead of the flight. The damping ratio was calculated using the logarithmic decrement, of which the points declared in Figure 85 were not used as intended. Instead of using two successive peaks, a peak and a trough were used in error, which resulted in an improper damping ratio calculation. Unfortunately, when attempting to recalculate the damping ratio using the intended points, the results were unrealistic and it is theorized that the high winds on the day of flight caused large deflections in the Rocket's heading. Being able to identify where wind took effect is not realizable and a different method to determine the dampening ratio must be used. The time to peak can be shown in Equation (22), and the damping ratio can be solved as shown in Equation (23) [64].

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \tag{22}$$

$$\zeta = \sqrt{1-\left(\frac{t_p \omega_n}{\pi}\right)^2} = \sqrt{1-\left(\frac{0.29s * 1.0667 rad/s}{\pi}\right)^2} = 0.9951 \tag{23}$$

Inserting all experimentally determined values back into Equation (18), the damping coefficient is determined.

$$d = 0.4665$$

The new damping coefficient is inserted back into Rocket 4's theoretical model to observe the response. Through an iterative process the new proportional gain is derived to be 0.3 using the calculated moment arm from the force of lift on the fin to the center of the rocket as 6.55in. As shown in Figure 90, the modeled response of Rocket 4 now appears to have oscillations that peaks are within the same magnitude but are off on the actual heading. Additionally, the FWHM remains similar between the data. Due to the heavy winds on the day of flight, it is theorized that the deflections in heading that are modeled would have been countering wind gusts.

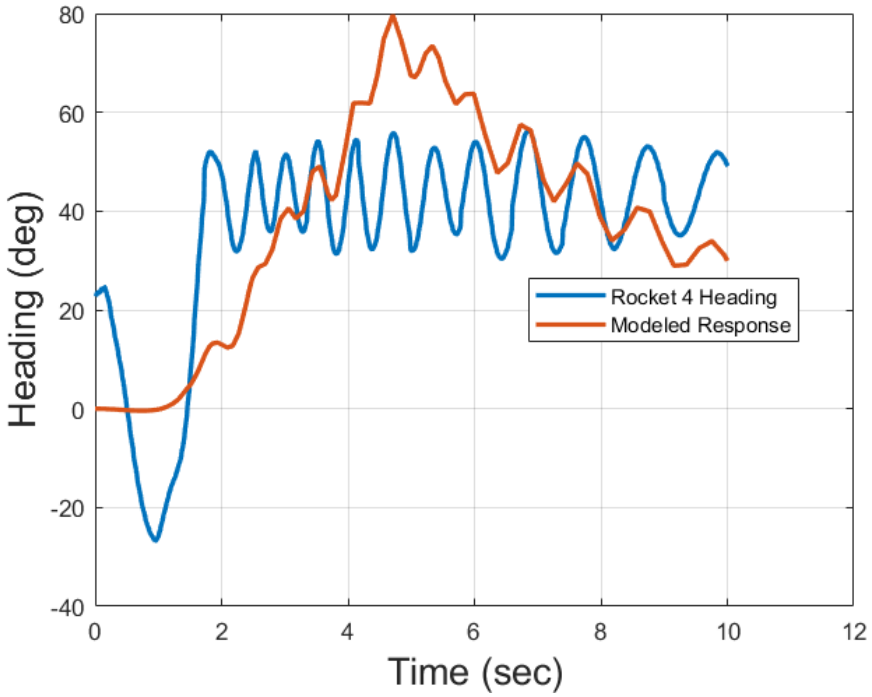

Figure 90. Rocket 4 Modeled Heading Using Corrected Damping Coefficients.

With the model proving to be more accurate, the same controller that was used in Rocket 5's actual flight was re-simulated to identify the change in the system response that should have been seen due to the errors made. As seen in Figure 91, the gains in Table 1 prove to show a larger overshoot and a slightly longer settling time but provide a strong response regardless of the damping ratio used in the system.

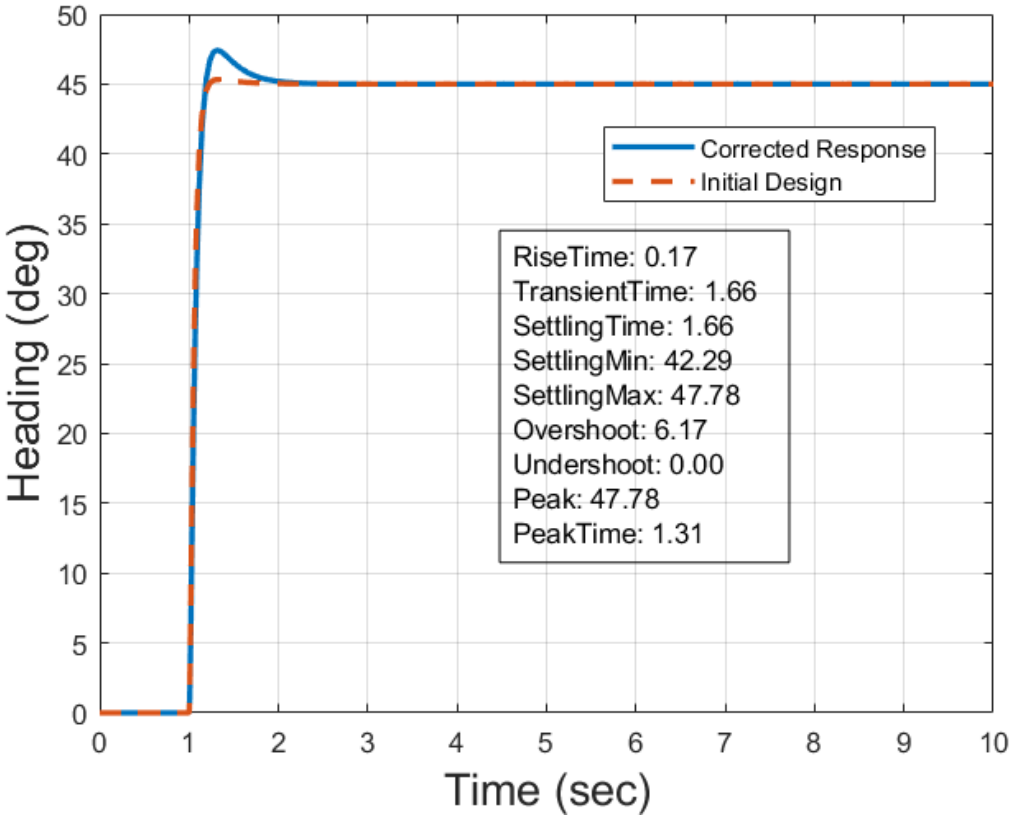

Figure 91. Corrected Step Response Compared to Initial Design.

B. SLED HOUSED ACTIVATION AND RELEASE DEVICE

1. Design

Premature separation between the nosecone and Sled plagued the first two launches as shear pins did not prove to be a viable way to control the weights required in the nosecone for both stability and payload delivery. As shown in the exploded view, the

mechanism consists of three main parts: the Planetary Gear Assembly, Mounting Base, and Coupler.

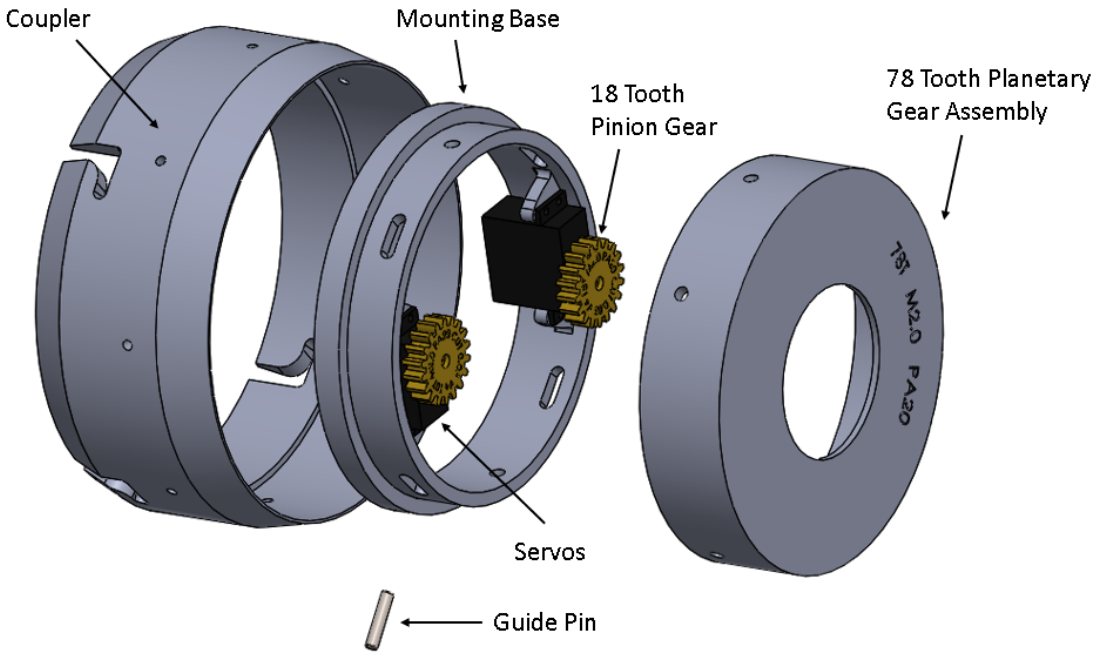

Figure 92. Separation Mechanism Exploded View

The Mounting Base is fixed in place using epoxy on a plywood bulkhead inside the cardboard coupler material. Four 2.54 cm (1 in) holes are then cut in the coupler material to allow for the Guide Pins to pass through as shown in Figure 93. The Planetary Gear Assembly aligns with the Pinion Gears and is held in place by the Guide Pins. The Planetary Gear Assembly is rotated by the control of 2 servos that simultaneously provide torque via pinion gears fit on the splines of the servos. As shown in Figure 94, the assembly holding the servos and the pinion gears are 3D printed from Polycarbonate.

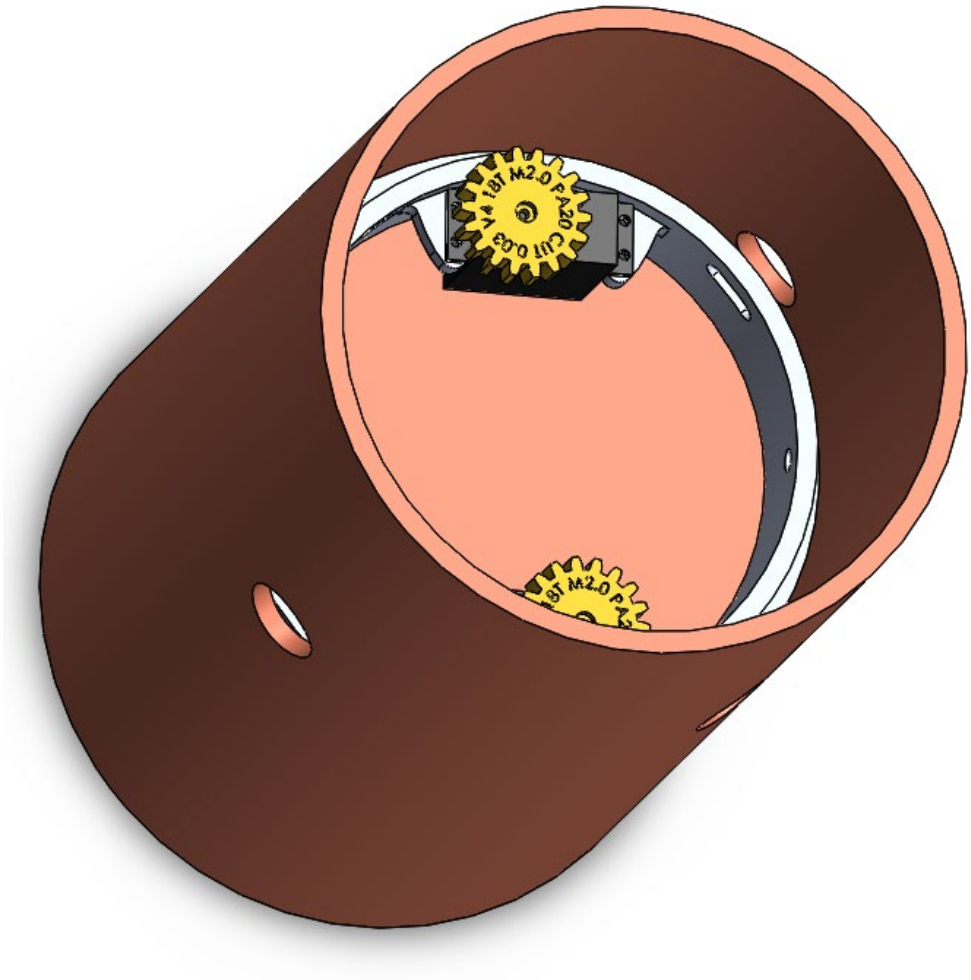

Figure 93. Sled Mounting Base Installed in Coupler.

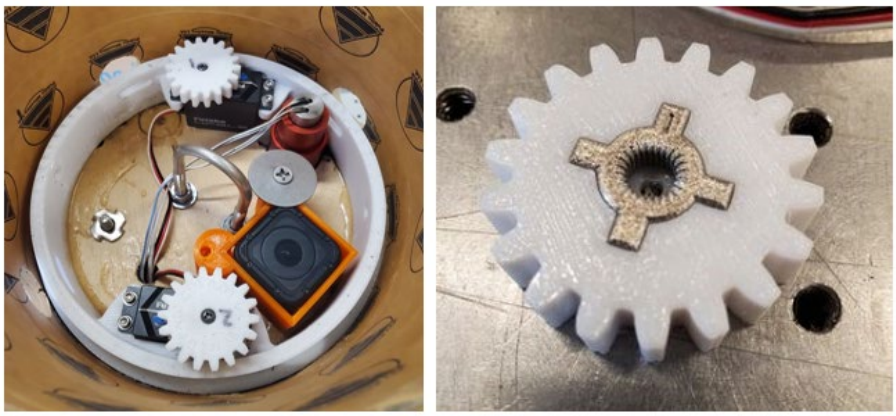

Figure 94. Sled Coupling Base (Left) and Gear Titanium Insert (Right).

The insert shown in the right of Figure 94 is 3D printed from Titanium following several rounds of testing revealed that the pinion gear's splines were the weakest part of the system after failing with a suspended load of 1020 N (230 lbf). The experimental test rig is shown in Figure 95 with a Raspberry Pi 4B controlling the servos powered by a 7.4V Lithium Polymer battery. Remote connection via a Wi-Fi signal allows for the wireless control of the Raspberry Pi.

Figure 95. Static Load Testing of Coupling Release Mechanism.

The servos used in this application are the Futaba A700 models as they each provide 7.26 N-m (5.35 ft-lb) of Torque [65]. The resulting torque on the gears is amplified by 4.3:1 ratio due to the planetary gear of the Rotating top assembly as shown in Figure 96. Total torque provided to the 4 steel pins is then 62.4 N-m (46 ft-lb) which is sufficient to release a 200 N (45 lbf) Sled suspended from a drogue.

Figure 96. Planetary Gears with Details (Left) and as Installed (Right).

The Coupler is also 3D printed out of Polycarbonate and is affixed to the airframe using eight screws. As shown in Figure 97, the Coupler fits firmly around the airframe but maintains the inner diameter. The final product is shown in Figure 98 as assembled.

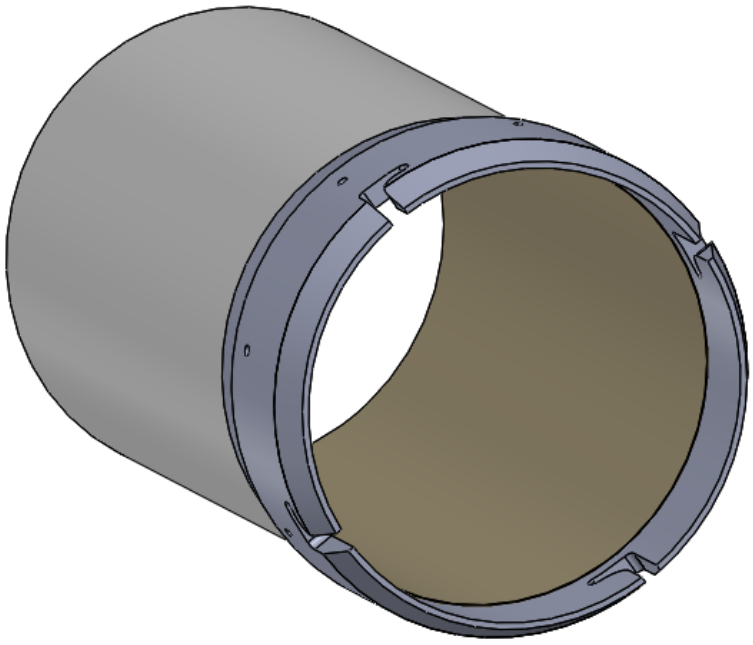

Figure 97. SHARD Coupler.

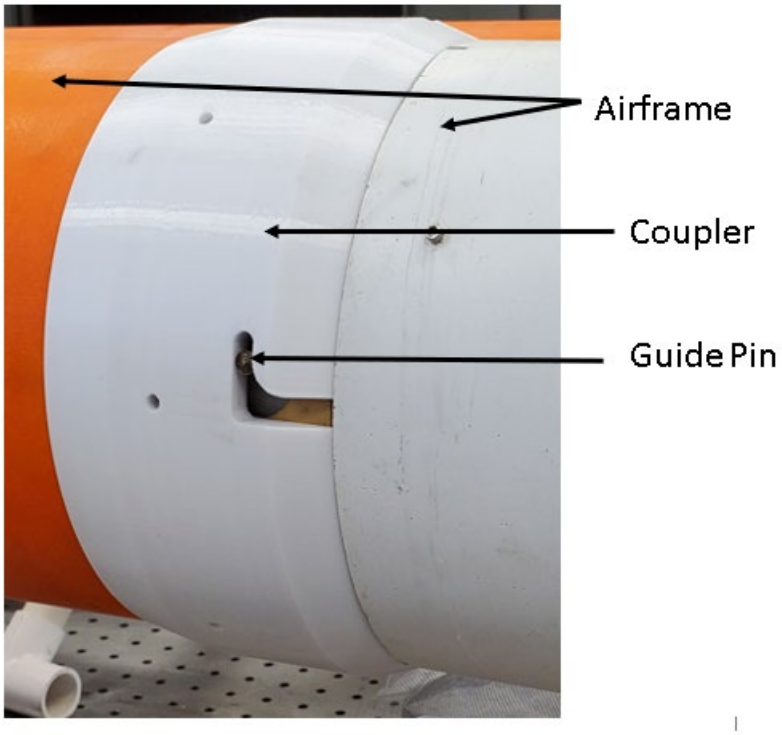

Figure 98. SHARD Installed.

2. Control

Controlling the system to release when intended is the next hurdle to undertake with this system. While it is easy enough to manually maneuver the servos from the ground, the system requires activation at predetermined conditions. The method chosen involves using a Raspberry Pi 4B with a Sense Hardware Attached on Top (HAT) designed for use on the International Space Station [66] as it has an onboard Pressure sensor that can be used to calculate altitude. As seen in Figure 99, the Raspberry Pi is connected to the servos via GPIO pins and is providing control using PWM signals. The code is shown in Figure 100.

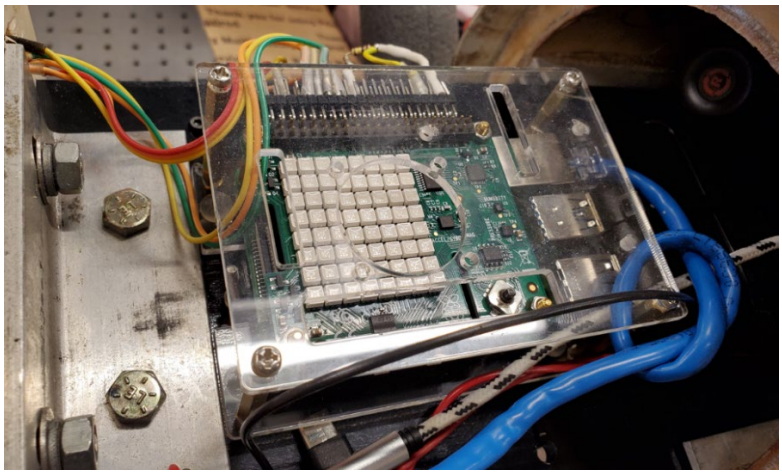

Figure 99. Raspberry Pi 4B with Sense HAT.

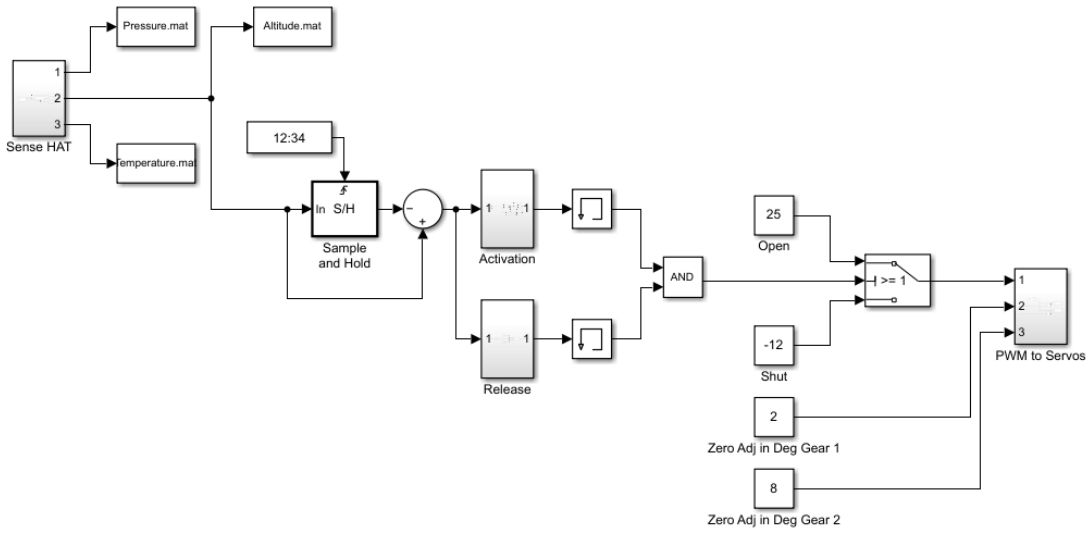

Figure 100. Sled Coupling Release Simulink Code.

The Sense HAT block is expanded as shown in Figure 101 to allow for unit conversion and then to correlate pressure to altitude. It is then subtracted from itself to provide a measurement of altitude above its current ground level. The Activation block as expanded in Figure 102, allows for the system to be activated when the altitude is greater than 610 m (2000 ft). The Release block, as expanded in Figure 103, allows for a release signal at 450 m (1500 ft). When the Activation and Release signals hold true, the signal is sent to rotate the servos to the open position.

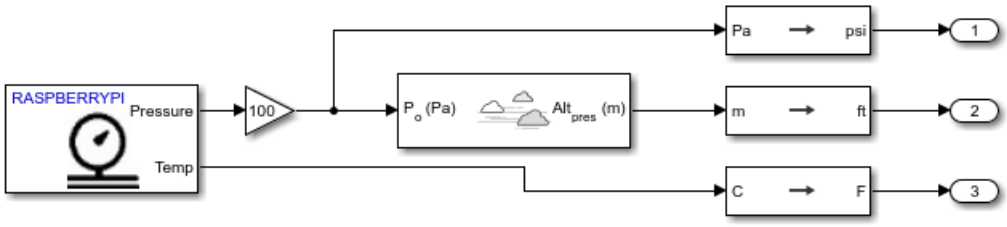

Figure 101. Sense HAT Block Expanded.

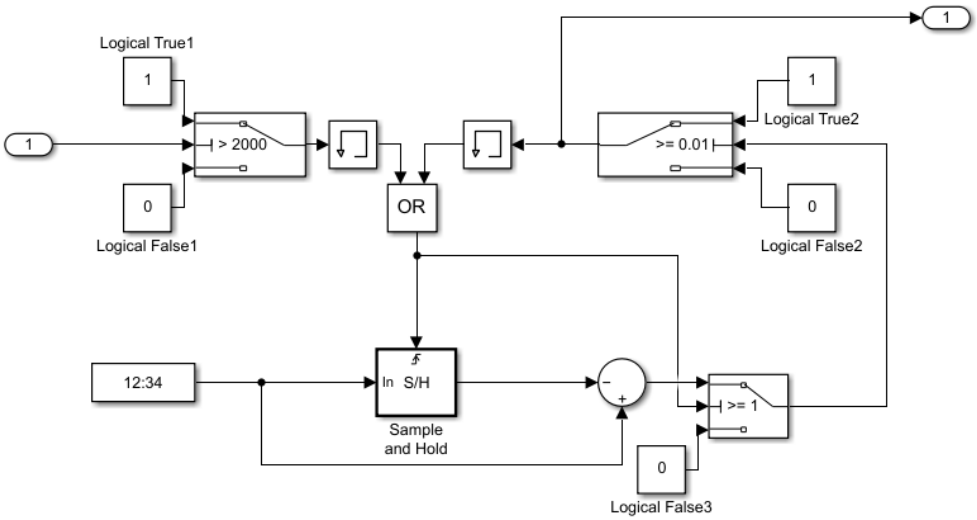

Figure 102. Activation Block Expanded.

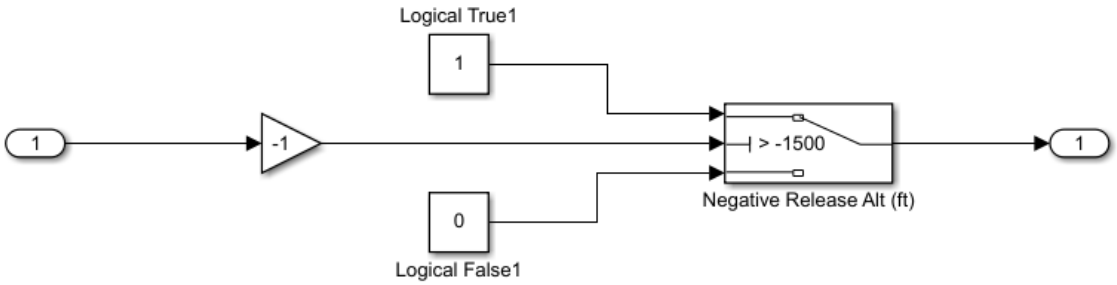

Figure 103. Release Block Expanded.

VI. CONCLUSIONS

Several conclusions can be drawn from the six different test campaigns to sum what has been observed. While all flights were not perfect, data was obtained from each flight whether in the form of visual imagery and/or sensor logs created during flight. Additionally, many lessons learned were developed and implemented in the follow-on flights. The following is a summary comparing the success of the subsystems to the objectives initially stated in Chapter 1 and a compilation of the future work that can be accomplished to contribute to the success of the RRPD-V.

A. SUBSYSTEMS

1. Control

Although it took six flights to fully develop use of the Raspberry Pi 4B as a control platform, the overall objective was met. The Raspberry Pi 4B is a COTS component that was utilized to control the flight of the RRPD-V at rates that were capable of controlling a sub-sonic rocket. Coupled with the BNO055 IMU, the Raspberry Pi 4B was able to successfully utilize the full sensor data rate of 100Hz through the use of overclocking the Raspberry Pi CPU. The signals obtained from the BNO055 were then able to be controlled using Simulink programming deployed in C+ code to the Raspberry Pi for execution. The deployed control algorithms then autonomously provided PWM signals to the servos that were able to successfully control the flight of the rocket about its main, z-axis.

2. Sled Coupling

Following the first two flights, it became obvious that the Sled coupling needed a redesign to allow for a reliable release of the Sled main parachute. After thorough testing, failure, and retesting, the SHARD was developed as a mechanical system to allow for accomplishing the objective. The system had two successful flights on Rocket 3 and 4 but failed to release the parachute on Rocket 5. As stated in Section 4.E.2.b, the failure of the parachute to release on Rocket 5 does not appear to be the fault of the SHARD as it was

tested and showed that it activated as required. With the final testing, the design is seen as a success and should be considered for use on future rocket designs.

3. Video

The feasibility of using a nose mounted video camera system to detect and track UAS swarms was tested in every rocket, except Rocket 2, with Rocket 4 being the only one to successfully capture the video data while under the drogue parachutes drag. While the 1080p resolution is insufficient to provide the quality required to target UAS swarms, the concept is still proven feasible. The video obtained during Rocket 4 was generally in view of the majority of the area of interest, but the simple approach of the Sled hanging from a drogue parachute still allowed for the Sled to rotate which is undesirable for continuous UAS tracking.

B. FUTURE WORK

1. Control

Building the control algorithms necessary to control a rocket are now realizable due to the increased data rates of 100Hz using the Raspberry Pi microprocessor versus the previously used Arduinos that allowed for the maximum ability of 30Hz. With certainty, the roll control of the RRPD-V has been developed, but the pitch and yaw planes have yet to be understood for the system. Developing an understanding of the controls required to provide a pitch and yaw control coupled with roll control are an obvious next step. Understanding these factors will allow for the ability to place the rocket in a location at a desired time. Possible next campaigns could include steering the rocket about a vector of gravity that is acting normal to the earth. This would allow an understanding of the response of the RRPD-V prior to attempting a roll and pitch maneuver to put the vehicle in the proper position. Additionally, GPS should be introduced into the control algorithm to introduce another frame of reference for the flight of the system. Lastly, a consideration should be made to place the control fins as canards instead of tail control. While, they will have less control on the pitch and yaw planes during flight with the booster attached, it will leave the Sled with control after separation. The booster control fins should also be replaced with larger fins to enable a larger stability margin by shifting the CP far back as possible.

2. Sled Coupling

While the design is successful, there is still some work that could be further investigated to allow for more assured separation. First off, the Raspberry Pi and its associated Sense HAT should have its own method of activating the CO2 systems black powder charge. This would ensure that the pressure observed by the Sense HAT would be the one fully controlling the system instead of having a hybrid of the Raspberry Pi and PerfectFlite systems for CO2 activation. The danger of having the PerfectFlite in the system is that it could light off the CO2 prior to the Raspberry Pi giving the release command, which could cause the failure of separation as shown in Rocket 5. Additionally, further refining of the coupler design could be done to allow for a way to provide backup for separation if the Raspberry Pi or servos fail to function.

3. Video

The need to stabilize the Sled during its decent has become obvious as the rotations and oscillations seen during the flight of Rocket 4 were still unacceptable. Several methods can be implemented that would allow for the stabilization of the video camera such as active canards during decent, the employment of grid fins for breaking and stabilization, or even the use of an active gimbal on the video camera to counter the rotation of the Sled. While there are additional options available, these offer the most promise as providing real-time video stabilization software are CPU intense. Additionally, other methods should be investigated to track the UAS swarms from the Sled nose such as Infra-Red or RADAR.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. RASPBERRY PI SETUP

A. INITIAL SETUP

1. Install Raspberry Pi Imager on local computer by installing from https://www.raspberrypi.org/software/ .
2. Write the Raspbian Image to an SD card using the Raspberry Pi Imager.
3. Install the SD Card into the Raspberry Pi. Do not power on.
4. Hook up a keyboard, mouse, and monitor to the Raspberry Pi.
5. Apply power to the Raspberry Pi and follow prompts for initial use. Specifically, enable "i2c," "ssh," "VNC," and "camera" when prompted.
6. When the system is connected to the internet, note its IP address by clicking the VNC logo in the right hand corner.
7. Make note of the Raspberry Pi login credentials. Default username is "pi" and password is "raspberry."

The IP address will change with each Wi-Fi network. Best practice is to connect to a Wi-Fi hotspot on a cell phone or on a computer so that the IP address can be readily observed if a connection has failed. On a cell phone, the settings for the hotspot can be opened and the connected IP addresses can be observed. On a Windows based computer, the mobile hotspot networking options can be opened to see the connected devices. This is important because VNC and MATLAB will only connect automatically to IP addresses that are known.

B. REMOTE CONNECTION

1. Download the VNC Viewer on a local computer from https://www.realvnc.com/en/connect/download/viewer/
2. Ensure Raspberry Pi and local computer are on same network.
3. Open VNC viewer and enter the IP address of the Raspberry Pi followed by login credentials.

4. Remote connection is now established to the Raspberry Pi.

C. CONNECTING MATLAB AND SIMULINK

1. Open MATLAB.
2. Under add-ons, search for and download the following:

i) MATLAB Support Package for Raspberry Pi Hardware

ii) Simulink Support Package for Raspberry Pi Hardware

iii) MATLAB Coder

iv) HDL Coder

3. Under the "APPS" tab download and install the Raspberry Pi Resource Manager.

If a prompt did not appear to install the MATLAB and Simulink support packages on the Raspberry Pi, an initialization of the setup is required. This can be done by clicking the "Add-Ons" button, again, and then select "Manage Add-Ons." Scroll to the MATLAB Support Package and select the three dots on the right. Proceed down the list to select "Setup." A pop-up screen will now guide through the process of installing packages on the Raspberry Pi. There is an option to install MATLAB's Operating System or to install the packages on the current operating system. While both appear operational, the MATLAB operating system does not allow for the full functionality needed to change i2c busses as will be described later. Select the option to allow for installing the packages on the current operating system.

Once the installation is finished, the Raspberry Pi is ready to start interfacing with the host computer, MATLAB, and Simulink.

D. TESTING MATLAB INTERFACE

Within the MATLAB Command Window, type the following to establish a connection to the Raspberry Pi and a prompt should appear showing details of the connection:

mypi=raspi

If the command issues an error saying that the IP address is not available, go into the Raspberry Pi Hardware Resource Manager and select the large yellow "plus" sign to open the drop down and then select the option for configuring a new device. Enter the IP address, username, and password to add the new device.

To determine control of the Raspberry Pi, command the green LED on the Raspberry Pi board to turn on and off by typing the following:

On: writeLED(mypi, 'led0',1)

Off: writeLED(mypi, 'led0',0)

If the light reacts as commanded, MATLAB is connected to the Raspberry Pi. If not, troubleshoot.

E. TESTING SIMULINK INTERFACE

Simulink offers a series of block sets that allow for easy and efficient interaction with the Raspberry Pi. To test this, a simple program to allow the Raspberry Pi's onboard green LED to flash can be initiated.

1. Open the "Library Browser" button within the "Simulation" tab of the Simulink window.
2. Scroll down to "Simulink Support Package for Raspberry Pi Hardware" and select "Basic."
3. Within this library, drag the block labeled "LED" to the blank workspace and then double click on the block to select the appropriate Board. Apply the changes.
4. The previous block is looking for a signal source of highs and lows to illuminate the LED, to do this again search in the library for a "Pulse Generator" using the search box in the top left. Drag and drop the Pulse Generator to the workspace and connect it to the LED.

5. Double clicking on the Pulse Generator will give options on how to setup the pulse.

If this is the first time using Simulink to deploy a script to a Raspberry Pi or if the Raspberry Pi has never been connected to Simulink, the script will need to be setup to work with the Raspberry Pi. To do this, select the "Modeling" tab and select the "Model Settings" icon to open the "Configuration Parameters" screen. Select "Hardware Implementation" and under "Hardware Board" Select "Raspberry Pi." If the Raspberry Pi has not been connected previously, select "Target hardware resources" and enter the IP address, Username, and Password. This can also be used to verify that the appropriate device is selected. Apply the changes and a new tab named "Hardware" is available.

6. Save the created model with a unique name.

7. In the "Hardware" tab select "Monitor & Tune" button after setting a time period to run the model. The light should now be blinking with a flash as setup for the duration of your established time period.

F. DEPLOYING A STANDALONE MODEL FROM SIMULINK

The process of deploying a standalone model on the Raspberry Pi is somewhat straightforward after the devices are connected and functioning. Only the method of deploying models to the Raspberry Pi via Simulink will be discussed here as the MATLAB deployment methods are not as refined as Simulink's capabilities. To deploy the previous model to the Raspberry Pi to run without the host computer connected, select the "Build, Deploy, and Start" button under the "Hardware" tab. The program will pause for several seconds while initializing generating the code required execute the model on the Raspberry Pi. To stop the model, use the Raspberry Pi Resource Manager app previously installed in MATLAB. To ensure proper operation, the following should be noted:

1. "Monitor and Tune" does not deploy the model fully for standalone use. It is a tool for debugging and testing. "Build, Deploy, and Start" is required to fully deploy the standalone model to Simulink. When "B0uild, Deploy,

and Start" is used, use MATLAB's Hardware Resource Monitor to stop by selecting "Stop Process."

2. When building and testing standalone models, change the model rebuild options to "Rebuild: Always" by selecting "Model Settings" icon to open the "Configuration Parameters" window and selecting "Model Referencing." Although this seems like a waste of time, debugging multiple issues has proven this to be effective in minimizing errors between rebuilds.

3. Never use the "save as" feature to rename your model or to use it for another purpose as Simulink will not associate all of the features correctly when trying to deploy the model. As a workaround, select the features in the workspace that are needed, right click on a block, and copy the portions. Open a blank model, paste the features, and save under a new name.

4. To enable file saving, select the "Model Settings" icon and search for "mat-file" in the search bar. Enable by checking the box and applying the settings.

G. CONNECTING THE BNO055 IMU

To hookup the BNO055 to the Raspberry Pi, the proper connections must be wired.

1. Raspberry Pi 3V3 to BNO VIN
2. Raspberry Pi Ground to BNO GND
3. Raspberry Pi GPIO 2 (SDA) to BNO SDA
4. Raspberry Pi GPIO 3 (SCL) to BNO SCL

To allow for use of the BNO055 IMU, the Raspberry Pi's I2C clock speed must be stretched to match that of the BNO055 (10kHz). While there are plenty of guides on how to do this, none of them work as described. The only method found that works is described below.

On the Raspberry Pi's command line type the following to open the config file:

sudo nano /boot/config.txt

scroll down to the information that shows "dtparam=12c_arm=on" and add the following code beneath it to establish a software I2C bus vice the hardware implemented bus:

dtparam=i2c_arm_baudrate=10000

To exit the configuration file, press "Ctrl X" followed by "y" followed by "Enter." Reboot the Raspberry Pi for changes to take effect. The only bus that MATLAB or Simulink will recognize is bus 1, even though there are possibilities for many more. For more information on the different overlays, the following command can be entered in the raspberry pi command line:

sudo nano /boot/overlays/README

H. USING THE IMU

To use the BNO055 IMU, it must first be calibrated. The calibration can be performed by using Appendix B. Once the Calibration is done, a file with the calibration data is saved and can be rewritten without having to repeat the calibration by using Appendix C.

Now that good data is flowing between the Raspberry Pi and the BNO055, a quest for accessing the data arises. Referring to the Datasheet for the BNO055, the complexity of the sensor begins to come to light. The data fields seem endless, so one field will be analyzed to gain an understanding of how to access it. For the case here, the Euler Heading Register will be accessed as will offer an intuitive sense of heading following calibration.

Looking into the data sheet, the Register Name "EUL_Heading_LSB" will provide 8 bits of data in the Least Significant Bit format with uint-16 precision under Register Address "1A." Finding the data is scattered in several places in the datasheet, so patience must be exercised.

To establish the connection between the raspberry pi and the BNO055 the following command must be entered with the default BNO055 address being "0x28." Keeping in mind that the addresses are written in hexadecimal, the following must be used to establish the connection:

BNO=i2cdev(mypi,'i2c-1','0x28')

To read the data, the register must be accessed and then converted into the format expected by multiplying by a conversion established in the datasheet. Enter the following to read a value of the heading:

Heading=readRegister(BNO,hex2dec('1A'),'uint16')/16

I. CONTROLLING A SERVO

While SIMULINK has a few options to control servos, only one has been found to allow for servo actuation without adding latency to the deployed system. Using the "Standard Servo Write" Block in Simulink a signal can be sent to the servos to command position via PWM. PWM signals are established based on width of the duty cycle of the pulse. As such, the most common pulse used to set zero on the servo is 1.5 ms. The Futaba servos operate on a 320Hz signal and through experimentation, the relationship between the change in pulse width per degree is 0.16/45. The PWM signal is then written as follows:

PWM=(Heading*0.16/45) +(0.0015*320)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. BNOCAL.M

```
%% BNO Calibration and Test
% Kyle Decker

%Developed to connect to Raspberry Pi and Sensors connected to the I2C bus.
%This code connects to the Raspberry Pi, BNO055 IMU, ADS1115 ADC and Servo.
%Calibration of the BNO055 is performed and settings are saved for use in
%CalWrite.m.
%The ADC is initialized and set to operate per the datasheet.

%clear all; close all; clc;

%Pick Operational Mode
%8 - Fusion with Acc and Gyro only - Relative
%12 - Fusion with Acc, Gyro, and Mag - Absolute

Opmode=8;

%Choose whether ADC and Servo connected
%1 - ADC and Servo Connected
%0 - ADC and Servo not used

ADC=0;

%% Connect To RPi and Create Connection to Sensors using i2c and clock
%stretching
if exist('i2csensor')==0

   %%%% Change config.txt to update i2c clock speed
   %%%% Only bus 1 can be seen by Matlab to allow clock stretching
   %%%% sudo nano /boot/config.txt
   %%%% add "dtparam=i2c_arm_baudrate=10000"
   %%%% sudo nano /boot/overlays/README for more information

   mypi=raspi                  %observe baudrate

   %%%%Only works with one i2c bus active%%%%%%%%%%
   bus=string(mypi.AvailableI2CBuses)      %get i2c bus
   address=string(scanI2CBus(mypi,bus));       %get addresses available
   i2csensor=i2cdev(mypi,bus,'0x28')           %initiate the BNO055 '0x28'
       if ADC==1;
       adc=i2cdev(mypi,bus(1),'0x48')          %initiate the ADC '0x48'
       end
end

%% BNO Calibration

% Ensure on Page 0 and initiate Calibration

Page=0;
writeRegister(i2csensor,hex2dec('7'),Page)                  %Change Page #0
```

```
PageID=readRegister(i2csensor,hex2dec('7'),'int8')          %Verify
writeRegister(i2csensor,hex2dec('3F'),bin2dec('100000'))    %Sensor Reset
pause(1)                                                    %for effect

%% Set Desired Units
Units=double(bin2dec('1 00 1 0 0 0 1'));                    %Per Datasheet
writeRegister(i2csensor,hex2dec('3B'),Units,'uint8');       %Write to BNO
writeRegister(i2csensor,hex2dec('3D'),Opmode)               %Set to Fusion

%For other Fusion modes or raw data check the Data sheet
OpMode=double(readRegister(i2csensor,hex2dec('3D'),'int8')) %Check OpMode set
                                                           %correctly

%% Absolute Heading Calibration
% Reads calibration information while performing calibration
if OpMode ==12

GyrCal=0;

disp('Do your cal thing! 3s are GOOD')
pause(3)
while GyrCal<3
   disp('Set sensor flat to cal Gyro')
   CalStat=double(readRegister(i2csensor, hex2dec('35'),'uint8'));
   binstr5=string(bitget(CalStat,5));
   binstr6=string(bitget(CalStat,6));
   GyrCal=double(bin2dec(binstr6+binstr5))
   pause(2)
   clc
end
disp('Gyro Cal Success!')
pause(2)

MagCal=0;

   while MagCal <3
       disp('Rotate sensor in fig 8 for mag cal')
       CalStat=double(readRegister(i2csensor, hex2dec('35'),'uint8'));
       binstr1=string(bitget(CalStat,1));
       binstr2=string(bitget(CalStat,2));
       MagCal=bin2dec(binstr2+binstr1)
       pause(1)
       clc
   end
   disp('Mag Cal Success!!')
pause(2)

   AccCal=0;

   while AccCal<3
       disp('Move sensor in 45deg inc on all axis to cal acc')
       CalStat=double(readRegister(i2csensor, hex2dec('35'),'uint8'));
       binstr3=string(bitget(CalStat,3));
       binstr4=string(bitget(CalStat,4));
```

```
        AccCal=bin2dec(binstr4+binstr3)
        pause(1)
        clc
    end
    disp('Acc Cal Success!!!')
pause(2)

    SysCal=0;
    while SysCal<3
        CalStat=double(readRegister(i2csensor, hex2dec('35'),'uint8'));
        binstr7=string(bitget(CalStat,7));
        binstr8=string(bitget(CalStat,8));
        SysCal=(bin2dec(binstr8+binstr7)+GyrCal+AccCal+MagCal)/4
        pause(1)
        clc
    end
    disp("System Fully Calibrated")

% Relative Heading Calibration. Does not use magnetometer.
elseif OpMode==8

    disp('Do your cal thing! 3s are GOOD')
    pause(3)
    GyrCal=0;
    while GyrCal<3
        disp('Set sensor flat to cal Gyro')
        CalStat=double(readRegister(i2csensor, hex2dec('35'),'uint8'));
        binstr5=string(bitget(CalStat,5));
        binstr6=string(bitget(CalStat,6));
        GyrCal=double(bin2dec(binstr6+binstr5))
        pause(2)
        clc
    end
    disp('Gyro Cal Success!')
    pause(2)

    AccCal=0;
        while AccCal<3
        disp('Move sensor in 45deg inc on all axis to cal acc')
        CalStat=double(readRegister(i2csensor, hex2dec('35'),'uint8'));
        binstr3=string(bitget(CalStat,3));
        binstr4=string(bitget(CalStat,4));
        AccCal=bin2dec(binstr4+binstr3)
        pause(1)
        clc
    end
    disp('Acc Cal Success!!!')
    pause(2)

    SysCal=0;
    while SysCal<3
        SysCal=(GyrCal+AccCal)/2
        pause(1)
        clc
```

```
    end
    disp("System Fully Calibrated")

    %Save Cal Data

    writeRegister(i2csensor,hex2dec('3D'),0) %Go to Calibration Mode

    %Gyro Offset
    Goff.zm=double(readRegister(i2csensor, hex2dec('66'),'int16'))
    Goff.zl=double(readRegister(i2csensor, hex2dec('65'),'int16'))
    Goff.ym=double(readRegister(i2csensor, hex2dec('64'),'int16'))
    Goff.yl=double(readRegister(i2csensor, hex2dec('63'),'int16'))
    Goff.xm=double(readRegister(i2csensor, hex2dec('62'),'int16'))
    Goff.xl=double(readRegister(i2csensor, hex2dec('61'),'int16'))

    % MAG Offset
    Moff.rm=double(readRegister(i2csensor, hex2dec('6A'),'int16'))
    Moff.rl=double(readRegister(i2csensor, hex2dec('69'),'int16'))
    Moff.zm=double(readRegister(i2csensor, hex2dec('60'),'int16'))
    Moff.zl=double(readRegister(i2csensor, hex2dec('5F'),'int16'))
    Moff.ym=double(readRegister(i2csensor, hex2dec('5E'),'int16'))
    Moff.yl=double(readRegister(i2csensor, hex2dec('5D'),'int16'))
    Moff.xm=double(readRegister(i2csensor, hex2dec('5C'),'int16'))
    Moff.xl=double(readRegister(i2csensor, hex2dec('5B'),'int16'))

    % Acc Offset
    Aoff.rm=double(readRegister(i2csensor, hex2dec('68'),'int16'))
    Aoff.rl=double(readRegister(i2csensor, hex2dec('67'),'int16'))
    Aoff.zm=double(readRegister(i2csensor, hex2dec('5A'),'int16'))
    Aoff.zl=double(readRegister(i2csensor, hex2dec('59'),'int16'))
    Aoff.ym=double(readRegister(i2csensor, hex2dec('58'),'int16'))
    Aoff.yl=double(readRegister(i2csensor, hex2dec('57'),'int16'))
    Aoff.xm=double(readRegister(i2csensor, hex2dec('56'),'int16'))
    Aoff.xl=double(readRegister(i2csensor, hex2dec('55'),'int16'))


    %Go back to Fusion Mode
    writeRegister(i2csensor,hex2dec('3D'),Opmode)

    % Save Cal Variables to File
    save('CalData.mat','Goff','Moff','Aoff')
end

pause(2)

%% Initiate ADC and servo
if ADC==1
binary=swapbytes(uint16(bin2dec('1 011 000 0 101 0 0 0 11')));
%Settings per Datasheet
writeRegister(adc,1,binary,'uint16');                                    %Write
Settings to ADC
serv=servo(mypi,16,'MinPulseDuration',.00149,'MaxPulseDuration',.0025);
%Initiate Servo on GPIO 16
end
```

```
%% BNO Test Graph
writeDigitalPin(mypi,4,1)                %Turns on Relay if connected

disp("Test me out! Move me! (~15sec)")
pause(1)
disp("Ready?")
pause(1)
disp("3")
pause(1)
disp("2")
pause(1)
disp("1")
pause(1)
disp("Go!")
clear pitch; clear roll; clear heading;

if ADC==1
tic
   heading(1)=0;                                          %initial condition
   writePosition(serv,60+heading(1));                     %zeros servo

   for i=1:100
   pitch(i)=double(readRegister(i2csensor, hex2dec('1E'),'int16'))/16;
   roll(i)=double(readRegister(i2csensor, hex2dec('1C'),'int16'))/16;
   heading(i)=double(readRegister(i2csensor, hex2dec('1A'),'int16'))/16;
   writePosition(serv,heading(i)-heading(1)+60);           %drives servo

   %Read ADC
   data = double(swapbytes(uint16(readRegister(adc,0,'uint16'))));
                                          %2's Compliment data type
   voltage(i)=data*6.144/(2^15);          %Set for 6.144V Scale

   end
else
tic
   for i=1:100
   pitch(i)=double(readRegister(i2csensor, hex2dec('1E'),'int16'))/16;
   roll(i)=double(readRegister(i2csensor, hex2dec('1C'),'int16'))/16;
   heading(i)=double(readRegister(i2csensor, hex2dec('1A'),'int16'))/16;
   end
end
time=toc
Hz=i/time
writeDigitalPin(mypi,4,0)                    %Turns off Relay if connected

%% Figures
figure(1)
plot(pitch)
hold on
title('BNO055 Calibration Euler Angle Test Plot')
plot(roll)
plot(heading)
legend('Pitch','Roll','Yaw','Location','best')
```

```
hold off

%ADC Plots
if ADC==1
Time=linspace(0,time,i);

figure(2)
plot(Time,(filloutliers(voltage,'linear')))    %Remove outliers and normalize
hold on
xlabel('Time (s)')
ylabel('ADC Voltage (V)')
title('ADC Voltage vs Time')
hold off

figure(3)
p=filloutliers(normalize(voltage),'linear');
plot(Time,normalize(p))                        %plot smoothed data
hold on
plot(Time,normalize(roll))
legend('ADC Raw','Normalized Sine Wave Input','Normalized
ADC','Location','best')
title('Normalized Servo Input and ADC Output vs Time')
hold off
end
```

APPENDIX C. CALWRITE.M

```
%% Writing BNO Calibration when Calibration File Exists from BNOCAL.m
% Kyle Decker

%close all; clc; clear all          %Uncomment for troubleshooting

%% Setup

%Pick Operational Mode
%8 - Fusion with Acc and Gyro only - Relative Ref Frame
%12 - Fusion with Acc, Gyro, and Mag - Absolute Ref Frame

Opmode=8;

%Choose whether ADC and Servo connected
%1 - ADC and Servo Connected
%0 - ADC and Servo not used

ADC=0;

%%
if exist('mypi')==0

%%%% Change config.txt to update i2c clock speed
%%%% Only bus 1 can be seen by Matlab to allow clock stretching
%%%% sudo nano /boot/config.txt
%%%% add "dtparam=i2c_arm_baudrate=10000"
%%%% sudo nano /boot/overlays/README for more info

mypi=raspi                  %observe baudrate

%%%%Only works with one i2c bus active%%%%
bus=string(mypi.AvailableI2CBuses)          %get i2c bus
address=string(scanI2CBus(mypi,bus(1)))     %get bno address
i2csensor=i2cdev(mypi,bus(1),'0x28')        %initiate the sensor (BNO)
   if ADC==1
   adc=i2cdev(mypi,bus(1),'0x48')          %initiate the ADC '0x48'
   end
end

load('CalData.mat')                          %load calibration data

%% BNO055 Init
writeRegister(i2csensor,hex2dec('3F'),bin2dec('100000'))    %BNO055 Sensor
                                                           %Reset
pause(1)                                                    %Pause for effect
writeRegister(i2csensor,hex2dec('3D'),0)                    %Set to Config
                                                           %Mode
pause(2)

%% Initiate ADC
```

```
if ADC==1
%Initiate ADC with Binary settings from datasheet
binary=swapbytes(uint16(bin2dec('1 011 000 0 101 0 0 0 11')));
writeRegister(adc,1,binary,'uint16');           %Write Settings to ADC
end

%% Gyro Offset
writeRegister(i2csensor, hex2dec('66'),Goff.zm)
writeRegister(i2csensor, hex2dec('65'),Goff.zl)
writeRegister(i2csensor, hex2dec('64'),Goff.ym)
writeRegister(i2csensor, hex2dec('63'),Goff.yl)
writeRegister(i2csensor, hex2dec('62'),Goff.xm)
writeRegister(i2csensor, hex2dec('61'),Goff.xl)

%% MAG Offset
writeRegister(i2csensor, hex2dec('6A'),Moff.rm)
writeRegister(i2csensor, hex2dec('69'),Moff.rl)
writeRegister(i2csensor, hex2dec('60'),Moff.zm)
writeRegister(i2csensor, hex2dec('5F'),Moff.zl)
writeRegister(i2csensor, hex2dec('5E'),Moff.ym)
writeRegister(i2csensor, hex2dec('5D'),Moff.yl)
writeRegister(i2csensor, hex2dec('5C'),Moff.xm)
writeRegister(i2csensor, hex2dec('5B'),Moff.xl)


%% Acc Offset
writeRegister(i2csensor, hex2dec('68'),Aoff.rm)
writeRegister(i2csensor, hex2dec('67'),Aoff.rl)
writeRegister(i2csensor, hex2dec('5A'),Aoff.zm)
writeRegister(i2csensor, hex2dec('59'),Aoff.zl)
writeRegister(i2csensor, hex2dec('58'),Aoff.ym)
writeRegister(i2csensor, hex2dec('57'),Aoff.yl)
writeRegister(i2csensor, hex2dec('56'),Aoff.xm)
writeRegister(i2csensor, hex2dec('55'),Aoff.xl)

%% Set Desired Units
Units=double(bin2dec('1 0 0 1 0 0 0 1'));               %Per Datasheet
writeRegister(i2csensor,hex2dec('3B'),Units,'uint8');   %Write Settings

%% Set To Fusion Mode
pause(1)
writeRegister(i2csensor,hex2dec('3D'),Opmode) % Set Operation Mode
pause(2)

%% BNO Test Graph
disp("Test me out! Move me! (~15sec)")
pause(1)
disp("Ready?")
pause(1)
disp("3")
pause(1)
disp("2")
pause(1)
disp("1")
```

```
pause(1)
disp("Go!")
clear pitch; clear roll; clear heading;

tic
for i=1:100
pitch(i)=double(readRegister(i2csensor, hex2dec('1E'),'int16'))/16;
roll(i)=double(readRegister(i2csensor, hex2dec('1C'),'int16'))/16;
heading(i)=double(readRegister(i2csensor, hex2dec('1A'),'int16'))/16;
end
time=toc
Hz=i/time

%% Plots for verification
plot(pitch)
hold on
title('BNO055 Calibration Test Plot')
plot(roll)
plot(heading)
legend('Pitch','Roll','Yaw','Location','best')
hold off

Temp=double(readRegister(i2csensor,hex2dec('34'),'int8'))*2
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. UNCORRUPT.M

```
%Recover Corrupt Files
%This function shifts the bits to account for a corrupt file header.
%Works for files that contain a time array for the first column.
%data_in is the .mat file.
%columns is the number of columns the data contains.
%timestep is the size of the time step in the time array
%data_out is the name of the variable to assign the data
%Ex: good_file=uncorrupt('corrupt_file.mat',4)

%Kyle Decker
%8/29/2021

%Special thanks to Dr. Hyatt Moore IV for providing guidance towards
%identifying how to save the data in corrupt files from deployed Simulink
%code on a Raspberry Pi.

function data_out=uncorrupt(data_in,columns,timestep)

 for i=0:100
     a = fopen(data_in,'r');                %open the corrupt file in read
                                             only mode

     b = fseek(a,i,'bof');                  %shift the bits by 'i' amount
                                            from the beginning of file

     c = fread(a, [columns,inf], 'double')'; %read the information starting
                                             at bit 'i' with # of columns
                                             identified in the input

     frewind(a);                            %shift bits back to the
                                             beginning of file

     fclose(a);                             %close file

     if mean(diff(c(:,1)))==timestep        %check for the time array in
                                             first column

        data_out=c;                         %if time array is present,
                                             output the saved data
     else
         continue                       %If false, continue loop
     end
 end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. STOPDEPMAT.PY

```
#Stopping A Deployed MATLAB/SIMULINK File using a GPIO trigger

#Kyle Decker

#7/23/21

#Use simulink to trigger pin 26 low to indicate the end of the program. Pin 26
#and 13 are connected.

#With a pulldown resistor to ground. Pin 26 is an output and 13 is an input

#USE cmd ‘top’ to determine actual name of running process


import RPi.GPIO as GPIO          #Library allowing use of the GPIO pins

import os                        #Library to type cmd line commands


GPIO.setmode(GPIO.BCM)           #Sets the GPIO’s pin number type

GPIO.setwarnings(False)          #Stops warnings. - SIMULINK is using the
GPIOs

GPIO.setup(13, GPIO.IN)          #Sets GPIO as a sensing port


a=1                              #Initial Condition

while a==1:

 b=GPIO.input(13)                #Sense GPIO condition

 a=b

 print(b)

if a==0:                         #Simulink will trigger the pin to zero

  os.system(“sudo killall R4SledMainRel.e”)   #Sends cmd line to kill the
                                              #program

 GPIO.cleanup()                  #Resets GPIO pins
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Electronics Hub, 2018, "Simple Mobile Jammer Circuit |How Cell Phone Jammer Works?" [Online]. Available: https://www.electronicshub.org/mobile-jammer-circuit/

[2] Pledger, T., 2021, "The Role of Drones in Future Terrorist Attacks," AUSA [Online]. Available: https://www.ausa.org/publications/role-drones-future-terrorist-attacks

[3] Price, R., 2019, "Build Your Own Sprayer Drone," LSU AgCenter [Online]. Available: https://www.lsuagcenter.com/articles/page1531429896515

[4] Pilot Institute, 2020, "What's the Difference Between Drones, UAV, and UAS? Definitions and Terms" [Online]. Available: https://pilotinstitute.com/drones-vs-uav-vs-uas/

[5] Hunt, D., 2017, "World War 1 History: The Kettering Bug—World's First Drone," Owlcation [Online]. Available: https://owlcation.com/humanities/World-War-1-History-The-Kettering-Bug-Worlds-First-Flying-Bomb

[6] Airforce-Technology, 2021, "Predator RQ-1 / MQ-1 / MQ-9 Reaper UAV, United States of America" [Online]. Available: https://www.airforce-technology.com/projects/predator-uav.

[7] Hambling, D., 2020, "News: Why The Air Force Needs A Cheaper Reaper," Navmar Appl. Sci. Corp. [Online]. Available: https://www.nasc.com/why-the-air-force-needs-a-cheaper-reaper/

[8] Applied Aeronautics, 2021, "Albatross UAV : BVLOS Drone" [Online]. Available: https://www.appliedaeronautics.com

[9] Rempfer, K., 2018, "Did U.S. Drones Swarm a Russian Base? Probably Not, but That Capability Isn't Far off.," Mil. Times [Online]. Available: https://www.militarytimes.com/news/2018/10/29/did-us-drones-swarm-a-russian-base-probably-not-but-that-capability-isnt-far-off/

[10] Hambling, D., 2021, "Israel's Combat-Proven Drone Swarm May Be Start Of A New Kind Of Warfare," Forbes [Online]. Available: https://www.forbes.com/sites/davidhambling/2021/07/21/israels-combat-proven-drone-swarm-is-more-than-just-a-drone-swarm/

[11] Rogoway, T., 2015, "Gyrocopter Exposes Weaknesses In D.C.'s Elaborate Air Defense System," Jalopnik [Online]. Available: https://jalopnik.com/gyrocopter-exposes-weaknesses-in-d-c-s-elaborate-air-de-1698170601

[12] Raytheon Missiles & Defense, 2021, “Coyote UAS” [Online]. Available: https://www.raytheonmissilesanddefense.com/capabilities/products/coyote

[13] Trevithick, J., 2018, “Army Buys Small Suicide Drones To Break Up Hostile Swarms And Potentially More,” The Drive [Online]. Available: https://www.thedrive.com/the-war-zone/22223/army-buys-small-suicide-drones-to-break-up-hostile-swarms-and-potentially-more

[14] Hambling, D., 2020, “See Raytheon’s Jet-Powered Interceptor Drone In Action,” Forbes [Online]. Available: https://www.forbes.com/sites/davidhambling/2020/05/07/raytheon-coyote-drone-jet-powered-interceptor/

[15] Trevithick, J., 2021, “The Navy Plans To Launch Swarms Of Aerial Drones From Unmanned Submarines And Ships,” The Drive [Online]. Available: https://www.thedrive.com/the-war-zone/39535/navy-contract-exposes-plans-to-launch-swarms-of-drones-from-unmanned-boats-and-submarines

[16] Host, P., 2020, “US Army Approves Raytheon’s Coyote Block 2 C-UAS for Foreign Sales,” Janes [Online]. Available: https://www.janes.com/defence-news/news-detail/us-army-approves-raytheons-coyote-block-2-c-uas-for-foreign-sales

[17] Guard From Above, 2019, “Intercepting Hostile Drones Using Birds of Prey” [Online]. Available: https://guardfromabove.com/

[18] Ong, T., 2017, “Dutch Police Will Stop Using Drone-Hunting Eagles since They Weren’t Doing What They’re Told,” The Verge [Online]. Available: https://www.theverge.com/2017/12/12/16767000/police-netherlands-eagles-rogue-drones

[19] Essary, T., 2017, “These Drone-Hunting Eagles Aren’t Messing Around,” Time [Online]. Available: https://time.com/4675164/drone-hunting-eagles/

[20] SRC, Inc., 2021, “Silent Archer Counter-UAS Technology” [Online]. Available: https://www.srcinc.com/products/counter-uas/silent-archer-counter-uas.html

[21] Federation of American Scientists, “Chapter 11 Countermeasures,” Fundam. Nav. Weapons Syst. [Online]. Available: https://man.fas.org/dod-101/navy/docs/fun/part11.htm. [Accessed: 13-Sep-2021]

[22] Northrop Grumman, 2021, “Counter Unmanned Aerial Systems (C-UAS)” [Online]. Available: https://www.northropgrumman.com/what-we-do/land/counter-unmanned-aerial-systems-c-uas

[23] O'Hara, M., 2021, "Leonidas," Northrop Grumman [Online]. Available: https://www.northropgrumman.com/wp-content/uploads/Northrop-Grumman-Epirus-High-Power-Microwave-Datasheet_210158.pdf

[24] Atherton, K., 2019, "Army Tests Using a Grenade to Stop Drones with a Net," Army Times [Online]. Available: https://www.armytimes.com/news/your-army/2019/02/11/army-tests-using-a-grenade-to-stop-drones-with-a-net/

[25] The Net Gun Store, 2018, "Net Gun Firing Distance." https://thenetgunstore.com/blogs/blog/net-gun-firing-distance

[26] United States Army Combined Arms Center, 2019, "Grenadier Guide" [Online]. Available: https://usacac.army.mil/sites/default/files/publications/17965.pdf

[27] Shekhar, H., 2011, "Prediction and Comparison of Shelf Life of Solid Rocket Propellants Using Arrhenius and Berthelot Equations," Propellants Explos. Pyrotech., **36**(4), pp. 356–359.

[28] Grohe, K., 2017, "Design and Development of a Counter-Swarm Prototype Air Vehicle," Master's Thesis, Naval Postgraduate School, Monterey, CA.

[29] Lobo, K., 2018, "Submunition Design for a Low-Cost Small UAS Counter-Swarm Missile," Master's Thesis, Naval Postgraduate School, Monterey, CA.

[30] Busta, M., 2019, "Design and Development of an Adaptable Flight Demonstration Vehicle with Modular Guidance and Control," Master's Thesis, Naval Postgraduate School, Monterey, CA.

[31] Thyberg, R., 2019, "Design and Testing of a Multi-Unit Payload Delivery and Tracking System for Guided Munitions to Combat Uav Swarm Threats," Master's Thesis, Naval Postgraduate School, Monterey, CA.

[32] Brophy, C., Class Design Report, 2020, .

[33] Bingham, B., Class Notes, 2019, .

[34] Siouris, G. M., 2004, *Missile Guidance and Control Systems*, Springer, New York.

[35] Brophy, C., Class Notes, June 21, .

[36] Philpot, T. A., 2013, *Mechanics of Materials: An Integrated Learning System*, Wiley, Hoboken, N.J.

[37] Brophy, C., Class Design Report, 2019, .

[38] Knowles, V., 2007, “Altimeter Port Sizing” [Online]. Available: https://www.vernk.com/AltimeterPortSizing.htm

[39] Co2cartridge, “CO2 38G threaded cartridge” [Online]. Available: http://co2-cartridge.co.uk/product/co2-38g-threaded-cartridge/. [Accessed: 12-Nov-2021]

[40] Leederville, 2017, “Shear Pin Calculations | Rocket to the Edge of Space” [Online]. Available: https://leederville.net/rocket/index.php/2017/10/21/shear-pin-calculations/

[41] Futaba, 2021, “HPS-H700” [Online]. Available: https://futabausa.com/product/hps-h700/

[42] Apogee Components, 2021, “CD3 Adventurer Kit” [Online]. Available: https://www.apogeerockets.com/Ejection-Systems/CO2-Ejection-Systems/CD3-Adventurer-Kit-12g-16g

[43] PerfectFlite, 2017, “Altimeters” [Online]. Available: http://www.perfectflite.com/altimeters.html

[44] King, P., ed., 2020, *The Official Raspberry Pi Camera Guide*, Raspberry Pi (Trading) Ltd, Cambridge.

[45] Mouser Electronics, 2021, “Adafruit Raspberry Pi Camera Board v2 - 8 Megapixels,” Newest Prod. [Online]. Available: https://www.mouser.com/new/adafruit/adafruit-pi-camera-board-v2/

[46] ThrustCurve, 2021, “Cesaroni 9994M3400-P” [Online]. Available: https://www.thrustcurve.org/motors/Cesaroni/9994M3400-P/

[47] Raspberry Pi, 2019, “Raspberry-Pi-4-Product-Brief.Pdf” [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf

[48] Bosch Sensortech, 2014, “BNO055 Intelligent 9-Axis Absolute Orientation Sensor,” Data Sheet [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf

[49] Adafruit, “Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055” [Online]. Available: https://www.adafruit.com/product/2472. [Accessed: 10-Oct-2021]

[50] Hopkins, J., 2020, “Understanding the Differences Between UART and I2C,” Total Phase Blog. https://www.totalphase.com/blog/2020/12/differences-between-uart-i2c/

[51] Multitronix LLC, 2019, “TelemetryPro Kate 2.0 Transmitter User Manual” [Online]. Available: https://www.multitronix.com/uploads/3/1/1/5/31157709/mx150_usermanual_rev1_1.pdf

[52] Chris’ Rocket Supplies, LLC, 2021, “Cesaroni N3301 White Rocket Motor” [Online]. Available: https://www.csrocketry.com/rocket-motors/cesaroni/motors/pro-98/6gxl-reloads/cesaroni-n3301-white-rocket-motor.html

[53] GetFPV, 2021, “Caddx Loris 4k Recording and FPV Camera System” [Online]. Available: https://www.getfpv.com/caddx-loris-4k-recording-and-fpv-camera-system.html

[54] GetFPV, 2021, “RunCam Split Mini 2 FPV / HD Camera” [Online]. Available: https://www.getfpv.com/runcam-split-mini-2-fpv-hd-camera.html

[55] Kambria, 2017, “3D Printing Filament: Effects of Temperature on PLA and PETG” [Online]. Available: https://kambria.io/blog/effects-of-temperature-on-pla-and-petg-3d-printing-filament/

[56] ThrustCurve, 2021, “Cesaroni 17613N2900-P” [Online]. Available: https://www.thrustcurve.org/motors/Cesaroni/17613N2900-P/

[57] Simplify3D, 2021, “Ultimate Materials Guide - 3D Printing with Polycarbonate” [Online]. Available: https://www.simplify3d.com/support/materials-guide/polycarbonate/

[58] Shawn, 2020, “How to Safely Overclock Your Raspberry Pi 4 to 2.147GHz.” https://www.seeedstudio.com/blog/2020/02/12/how-to-safely-overclock-your-raspberry-pi-4-to-2-147ghz/

[59] Philipp, M., 2017, “Using the Raspberry Pi Timer for Embedded Environments.” https://blog.studica.com/raspberry-pi-timer-embedded-environments

[60] Nave, R., 2016, “Moment of Inertia,” HyperPhysics [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html.

[61] Griffin, J., 2008, “Damping Ratio Estimation from Time Domain Response - Log Decrement,” Dyn. Syst. Controls Handouts [Online]. Available: https://www.andrew.cmu.edu/course/24-352/

[62] Keim, R., 2018, “What Is a Low Pass Filter? A Tutorial on the Basics of Passive RC Filters,” Circuits [Online]. Available: https://www.allaboutcircuits.com/technical-articles/low-pass-filter-tutorial-basics-passive-RC-filter/

[63] Bhandare, D. S., Shaikh, H. M., and Kulkarni, N., 2016, “Design and Implementation of Self-Tuning Fuzzy-PID Controller for Process Liquid Level Control,” Semantic Sch. [Online]. Available: https://www.semanticscholar.org/paper/Design-and-Implementation-of-Self-Tuning-Fuzzy-PID-Bhandare-Shaikh/10e2bc1450e48971ba6def98a5ec5d1f5b44f9b4

[64] Electrical4U, 2021, “Time Response of Second Order Control System” [Online]. Available: https://www.electrical4u.com/time-response-of-second-order-control-system/

[65] Futaba, “HPS-A700” [Online]. Available: https://futabausa.com/product/hps-a700/

[66] Ander, J., 2017, “The Possibilities of the Sense HAT,” Raspberry Pi [Online]. Available: https://www.raspberrypi.com/news/sense-hat-projects/.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California