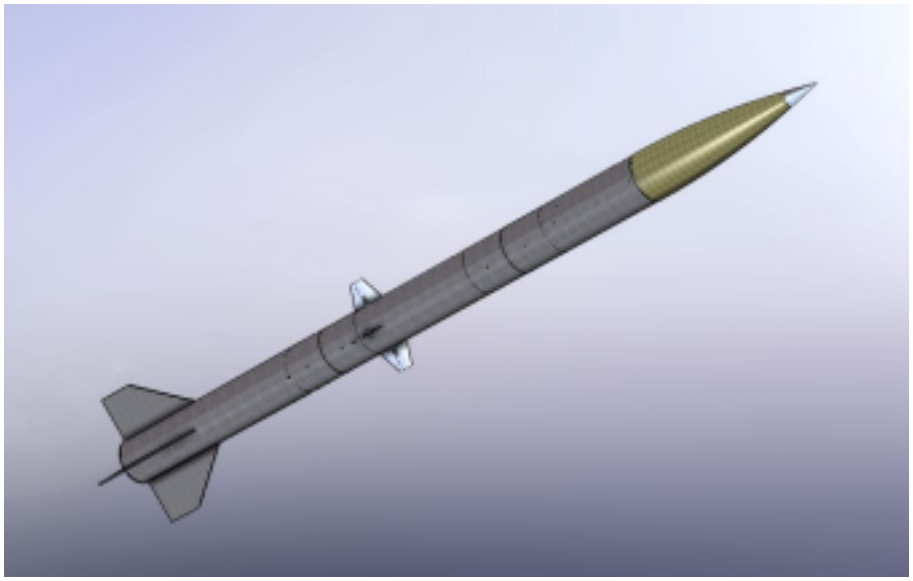
Clemson Rocket Engineering
Oculus I

Team 15 Project Technical Report to the 2022 Spaceport America Cup



W. Lane[1], A. Dempsey[2], B. Bischoff [3], T. Muennich[4], M. Conti [5], G. Sisk [6], D. Wallace [7], T. Britten [8], K. Jones [9], F. Broñola [10], D. Carlson [11], S. Paguaga [12], N. Reyner [13], J. Wiley [14]
Clemson University, Clemson, South Carolina

P. Tarle[15]
NAR Music City Missile Club

[1] Upper Leadership, President.
[2] Upper Leadership, Vice President
[3] Upper Leadership, Treasurer
[4] Officer, Structures Lead
[5] Officer, Payload Lead
[6] Officer, Active Aero Lead
[7] Officer, Flight Dynamics lead
[8] Officer, Co-Avionics lead
[9] Officer, Co-Avionics Lead
[10] General, Avionics
[11] General, Flight Dynamics
[12] General, Structures
[13] General, Structures
[14] General, Structures
[15] L3 Mentor.

Abstract

Oculus I is the Clemson Rocket Engineering (CRE) entry into the 2022 Spaceport America Cup. This rocket will be competing in the 10,000 foot, Commercial Off The Shelf (COTS) category. There are not stand out design features on this rocket, it is a standard dual deployment recovery, single stage sounding rocket, primarily as a result of club inexperience. The team went through two extremely difficult years because of COVID-19, where so much knowledge was lost that had been gained from previous competitions. This forced leadership to make the decision to develop a straightforward rocket in hopes that the next generation at CRE might develop a sound foundation to further develop the rocket team at Clemson University. The payload on board Oculus I will adhere to a 3U cubesat form factor and will be deployed from the rocket at main parachute deployment, which will occur at approximately 1000' AGL. Once deployed, the payload will control its own roll as it descends under its own parachute. The idea behind this mission being that future payload teams at CRE can take this project and eventually move it into a 2U or even 1U form factor, and then our excess space in the 3U CubeSat can be given to either a professor, a local club, or even a local k-12 school. In addition to a deploying payload, the team has a ground station that will communicate with the avionics bay on board the rocket to track live telemetry and accurately determine the final landing area of the rocket, in hopes that it will minimize recovery time. Other than these two new aspects to Oculus I, the rocket uses a solid rocket motor, commercial parachutes, and shock cord, a commercially purchased nosecone, and commercial hardware. Otherwise, the main airframe, including the fins, are fully manufactured in house.

Contents

Introduction

Mission Statement

The Clemson Rocket Engineering (CRE) team is the sole aerospace design, build, fly program on the Clemson University campus. The team is a Designated Student Organization (DSO), directly affiliated with the Clemson University Department of Mechanical Engineering, and it consists of over 100 students from all different departments and stages of school. Having competed in the Spaceport America Cup now since 2017, the team continues to grow and in experience. COVID-19 proved a difficult setback for CRE, but as has shown this year the desire for an on-campus aerospace entity at Clemson University has not faded, and only appears to be in higher demand. With professors all around campus learning of the program, and reaching out to lend a helping hand, the

team expects to continue to develop professional entry level engineers for years to come.

Work Breakdown Structure

Having gone through two extremely hectic years filled with disorganization due to the COVID-19 pandemic, the CRE Upper Leadership team made an early effort to ensure a strict organization of the team was developed and adhered to. The leadership tiers are shown in Table 1.

Table 1 Leadership Tiers

Tier No.	Tier Title	Positions Included
1	Upper Leadership	President, Vice President, Chief Engineer, Treasurer
2	Officers	Subteam Leads, IR&D Leads
3	Managers	Safety Manager, Community Outreach Manager
4	General	All other general subteam members

Table 1 provides a general idea of who is who in the club and lead to the development of the work breakdown structure shown in Table 2.

Table 2 Work Flowdown

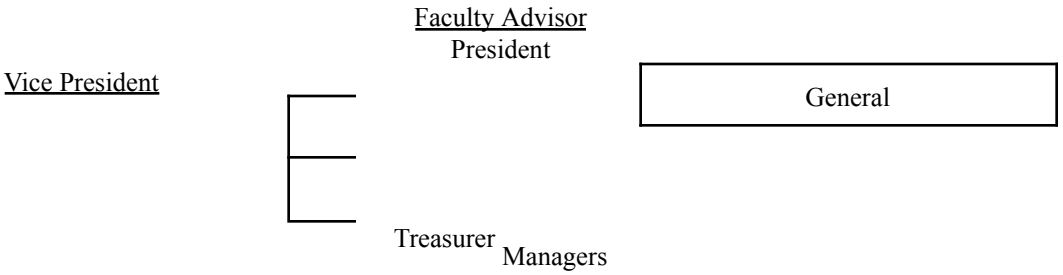


Table 2 shows how work was assigned and where the various individuals in the club were to report or direct questions throughout the year. This level of organization provided clarity to younger members who were typically unsure where to direct their questions throughout the year.

Budget Analysis

The Clemson Rocketry Engineering Team was allocated for the Fall 2021 through Summer 2022 school year $27,118 from Clemson University's Student Funding Board (SFB). The SFB is a student-led organization that allocates the Student Activity Fee to undergraduate student organizations at Clemson University. Our team the school year prior presented a budget request that shows why we need and how we will spend our funding. Prefacing this school year, the upper-leadership decided not to have dues for the general members due to this generous allocation. The pie chart in Figure 1 shows exactly how the treasurer decided to allocate the budget for general spending and to the different sub-teams.

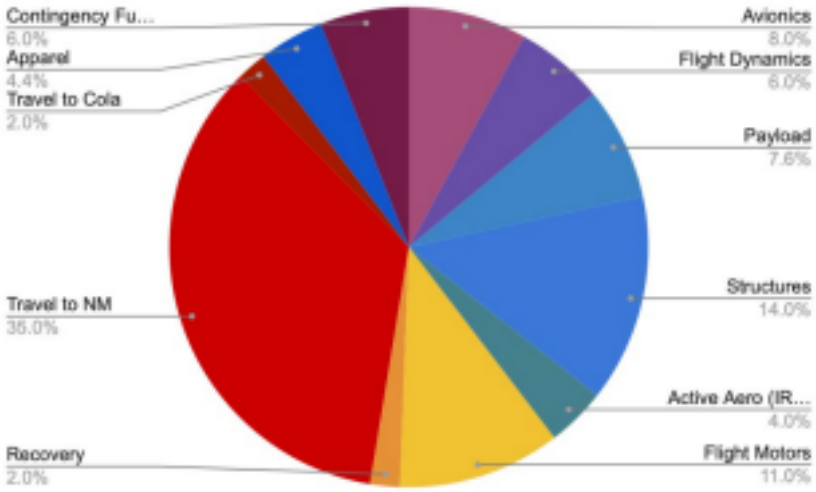


Fig. 1 Budget Pie Chart

As of April 14th, our remaining budget is $6,123.93. A large chunk of that remaining will go towards gas to New Mexico and hotel rooms in Dallas, TX for our halfway point. Out of our budget allotments; payload, active aero/R&D, and contingency funds were over budgeted. The payload subteam was basing their experiment off of the previous year (COVID IREC cancellation). So, the team had materials already to work from. Our active aero and research and development team did not spend all its allotment due to prioritizing time to focus on the main construction of the rocket. The structure's subteam was under budgeted because the team was not able to properly predict how much money was going to go into fabricating test articles of the rocket. We were able to increase the structure's budget by transferring over from active aero's and payloads remaining budget.

Mission Concept of Operations Overview

The team chose to compete in the 10K COTS category for a number of reasons. The primary reason being, when the team competed in 30K COTS in 2019, the rocket barely made it to 75% of the predicted apogee. In addition, there were only three individuals on the 2021/22 team roster that had ever participated in a high-powered rocket launch, and a majority of the team was made up of much younger individuals. This ultimately resulted in the decision made by leadership to compete in the 10K COTS category as opposed to 30K COTS in hopes that the team might perform better than previously and be able to work on the fundamentals of subsonic rocket flight with the new members of the team.

In addition to deciding what category to compete in, the team's payload would further drive the mission definition and requirements of the rocket. The payload team has been developing a cold gas thrust, roll control, mission since the 2019/20 school year, however, due to lack of club participation and inability to access the schools machine shop, the project was never finished. However, all components necessary to complete the project were already on hand. This resulted in the team's new Payload Lead deciding to continue the previous project in an effort to save money and try to ensure a higher chance of success for the team.

These two major decisions drove the development of the concept of operations shown in Figure 2.

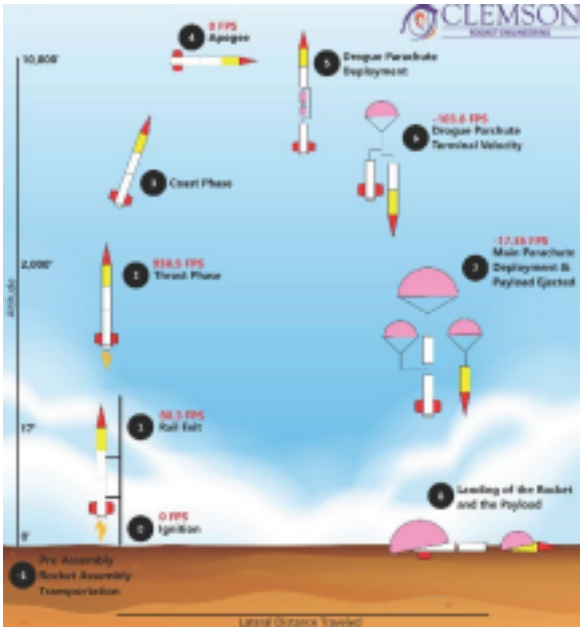


Fig. 2 Concept of Operations

Figure 2 shows ten primary stages in flight, the first labeled '-1' is the pre-flight assembly as well as the transportation of the rocket. This stage is detailed in depth throughout this report. The second stage is labeled '0' which of course is the instant the motor grains are ignited and the rocket begins to accelerate off the pad. The third stage, labeled '1' is the moment the rocket has reached a point in flight where there are no longer two points of contact on the launch rail. This means that the direction of the rocket is no longer being guided by the launch rail but rather it is reliant on the stability generated by the location of the center of pressure generate by the rocket's fins. The fourth stage, labeled '2' is the time in which the motor is burning, meaning the rocket continues to accelerate to its highest velocity. The fifth stage, labeled '3' is the time the motor spends decelerating before reaching stage six, labeled '4' being the highest point in flight, where the rocket begins its decent back to the ground. The seventh stage labeled '5' is the deployment of the drogue chute, which as shown in the eighth stage, labeled '6' brings the descent speed of the rocket to approximately 104 fps. The ninth stage, labeled '7' is when the main parachute and the payload are deployed simultaneously at approximately 1,000' AGL. From here, the rocket descends independently of the payload, both under their own parachutes, bringing each descending component to around 18 fps. The tenth and final stage of the flight, labeled '8' is when all components safely touchdown on the ground and the recovery team is sent towards their last known location to recover both components.

System Architecture Overview

This section will be used to dive deeper into the airframe and subassembly designs of Oculus I. Figure 3 shows a rendering of the rocket fully assembled. There are four clear sections of the rocket, the nosecone being the front end of the rocket has a green tint to it, and the other three sections aft of the nosecone are all black. This, however, is not representative of what material each section is made of, but rather shows what components were

made in house by the team or purchased off the shelf. In this case, the nosecone is the only visible component in Figure 3 that was purchased off the shelf, all other sections, including the fins, were constructed in house. All components besides fins are fiberglass, the fins are carbon fiber, the fiber glass tubes were dyed black for aesthetics.

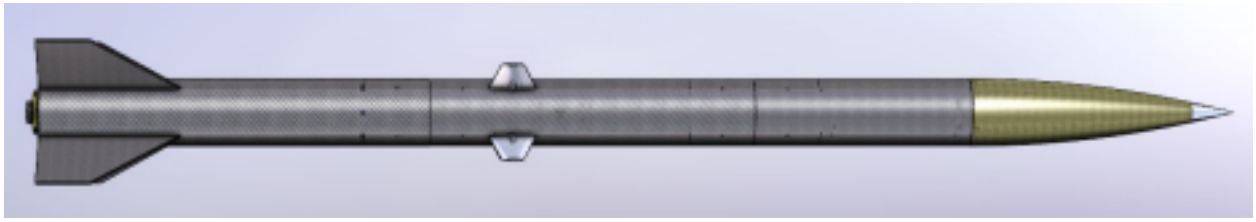


Fig. 3 Oculus I

Motor Tube

The motor tube serves two primary purposes, both of equal importance to the success of the flight of the rocket. The first is it acts as the mounting point for the motor, meaning this tube is responsible for taking virtually all of the load generated by the motor. The second purpose of the motor tube is to act as a mounting surface for the fins on the rocket, as seen in Figure 4. In addition to the motor and the fins, there are three aluminum bulkheads in the motor tube, two are a quarter inch thick and the third and upper most bulkhead is a half inch thick. This half inch thick plate has a through hole in the center, 4 equally spaced ¼-20 holes on the face and 4 equally spaced radial 10-24 holes. The through hole allows a shouldered eye bolt to pull the motor and its aft most plate into compression, putting the first of two thrust bearing plates in contact with the motor. The second thrust bearing plate, shown as the middle plate in Figure 4, is now loose between the upper shoulder of the motor and the plate where the eyebolt is seated. From here, the four equally spaced ¼-20 bolts mentioned previously are screwed into place and push the motor into compression, thus providing a second channel by which the load generated by the motor can travel during launch. There is 6 inches of empty space at the front end of the tube, as seen in Figure 4, where the coupler fits in, with holes for the shear pins.

Fig. 4 Motor Tube Side View

Avionics Tube

The original design for the avionics tube was to be a 30in long tube with a 6in inner diameter made out of fiberglass. To be as conservative as possible with the strength of the fiberglass, eight wraps were chosen to be sufficient enough to withstand the forces involved in a launch. This meant the outer diameter of the tube was 6.125in. Fiberglass was chosen specifically for its ability to be radio frequency transparent so all of the data recording instruments could transmit back to the computer module. There have been four sets of hole locations that have been put in to allow the tube to function properly, as shown in Figure 5. Each set of hole locations were measured from the bottom edge of the tube to have a consistent degree of accuracy. The first set was located three inches above the bottom edge and serves as the hole locations for the coupler. The next set of holes were 6.125in from the edge of the base. These holes serve as the mounting spot for the bottom bulkhead for the avionics subsystem. The next set of holes were 21.875in from the bottom of the tube to locate the top bulkhead for the avionics subsystem. The last set of holes were 27in from the bottom edge of the tube to locate the mounting points for another coupler. The original design had to be altered slightly to accommodate a larger section in the rocket for the parachute and recovery system. This meant the length of the tube had to be increased to 38in. The only area that this affected was the hole locations for the upper coupler to the tube instead of being 27in from the bottom of the tube it was moved to 35in from the bottom of the tube.

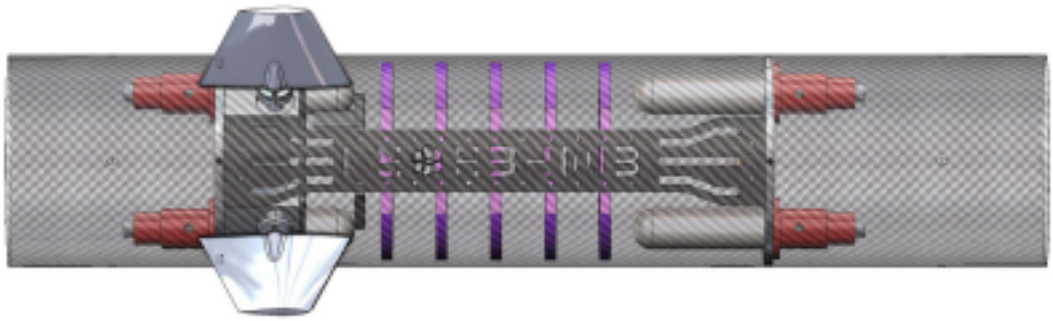


Fig. 5 Avionics Tube

Payload Tube

The original design of the payload tube was to make it 12in long with a 6in inner diameter made from carbon fiber. We decided that to be conservative with our strength estimates we used 5 wraps of our carbon fiber. The outer diameter of the tube came close to being 6.125in but was not as wide as the fiberglass tube. However, it was soon realized that we needed more space between the payload tube and avionics tube to house the recovery system of the rocket. So, a decision was made to make the payload tube 20in to accommodate the extra room needed. Unfortunately, the payload design required that its nozzles would be exposed during deployment to stabilize the payload. This meant the 20in tube had to be rethought of. After the new 38in avionics tube was constructed the old 30in fiberglass tube was cut down to be 11.25in to ensure the nozzles of the payload would be exposed during deployment. The holes already drilled into the old avionics tube were in position to accommodate the couplers needed to attach the two tubes together. In summary the tube used on this year's rocket is a 11.25in fiberglass tube with an inner diameter of 6in and an outer diameter of 6.125in. This assembly is shown in Figure 6.

Fig. 6 Payload Tube & NoseCone

Avionics

Location and overview

The avionics bay, shown in Figure 7, on the Oculus I sits between where the main and drogue parachutes are stored. The main parachute sits above the bay, while the drogue sits below. The bay is anchored above and below by an aluminum bulkhead. Each bulkhead has two holes drilled into it where the CO2 ejection charges for the main and drogue parachutes, respectively, sit in the avionics bay. The avionics system for the Oculus I has four main purposes during flight. These purposes are as follows:

- Deploying both drogue and main parachutes
- Recording sensor data locally and on the ground, providing real time data output •

Provide real time tracking of the Oculus I throughout the course of flight

- Capture video of the duration of the flight for future use

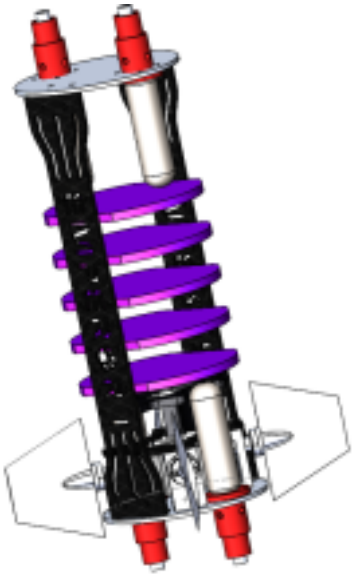


Fig. 7 CAD of Completed Avionics Bay

A carbon fiber support structure runs down two sides of the bay. Each support structure contains drilled holes, incorporating a shelving design for the components of the bay. Each shelf in the bay holds different components relating to each of the three onboard sub-systems of the overall avionics system for Oculus I. Components are screwed

down into their shelves, using a 3D printed mounting system, as seen in Figure 8, components are kept in place. Additionally, any metal screws are insulated from any component to prevent shorts or a build up of charge upon one or more components.

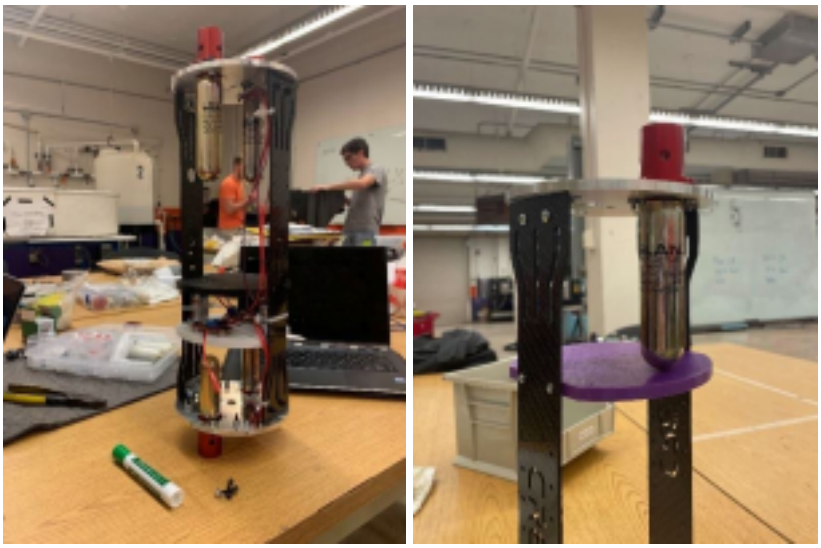


Fig. 8 Partially Complete Avionics Bay

The avionics system as a whole, not just the bay, can be broken down into four (4) main sections. The first two of these sections are the COTS (commercial off the shelf) and SRAD (student researched and developed) sub systems. These two sub-systems record and transmit the data produced by the sensors on the rocket, as well as being in charge of deploying both the drogue and main parachutes for the Oculus I.

The third sub-system present in the avionics system is the base station. Located on the ground, the base station consists of Whip and Yagi-Uda antennas, an Arduino Uno and two stepper motors and corresponding driver IC's. This sub-system is designed to receive and store data on the ground in real time during the Oculus I's flight.

Lastly, the final sub-system present in the avionics system is the Raspberry Pi 0 Camera System. Triggered by a signal sent from the base station close to launch, the camera system records footage of the entire flight, stopping when the Oculus I has landed.

COTS and SRAD systems and Wiring Diagram

The Oculus I's COTS portion of the avionics bay is controlled by two main components. The first of these is the Featherweight GPS module, which is used to assist in recovery post flight. The Featherweight is a self-contained fully functional system. This perk was part of the reason that it was chosen as our COTS GPS.

The Featherweight's provided 400mah battery lasts a listed sixteen hours on a single charge, much more than what is needed, according to the predicted setup and flight time of the Oculus I. Additionally, since the Featherweight GPS does not rely on any other components aboard the Oculus I, its self-sufficiency decreases the failure modes of the avionics bay, in the case of partial or catastrophic failure during the duration of Oculus I's flight. Due to this, the Oculus I is guaranteed to be trackable during and after flight.

The second half of the Oculus I's COTS system are the Entacore AiM altimeters (AiM). Two AiMs are present on the Oculus I's avionics bay for redundancy, in case of midflight failure of either. The AiMs, using barometric pressure, record the altitude of the rocket at a rate of ten (10) samples per second. The AiMs additionally selected for the Oculus I's avionics bay for their dual deployment system, meaning that each AiM present on the Oculus I has the ability to deploy both the drogue and main chute. Each AiM therefore, operates independently of any SRAD system aboard the rocket.

This ensures that the COTS system present on the Oculus I is not overridden in any way by an error in SRAD software or hardware, decreasing the failure modes of the avionics bay causing no parachutes to be deployed during the flight, an instance of catastrophic failure.

Additionally, the AiMs are supplied by a battery that is independent of both the SRAD system and the Featherweight GPS on the Oculus I. Shared only by the Raspberry Pi 0 (see figure 10), and the Active Aero system

(not included in figure 10), the AiMs minimize their failure modes in terms of power loss due to mistakes in both hardware and software in SRAD components elsewhere in the avionics bay.

The Oculus I's main mission is to reach its target apogee and return to the ground in a safe manner, in terms of velocity relative to the ground. While a COTS system can easily accomplish this feat on its own, the development of an SRAD detection of apogee (~10k ft.) for drogue parachute deployment and the detection of 2k ft. altitude reached on descent for the deployment of the main chute can be accomplished.

Therefore, Oculus I utilizes a Teensy 4.0 microcontroller for its SRAD system controller. In addition to the Teensy 4.0 the following sensors are used on the Oculus I's avionics bay, along with how they will be addressed in this section and the purpose of each component:

- BMP-280 (BMP): Temperature, Humidity, Altitude, and Barometer
- SAM-M8Q (SAM): GPS-breakout board present along with component
- MPU-9250 (MPU): Gyro, Magnetometer, and Accelerometer

Data on the Oculus I is transmitted to the ground station at a rate of four (4) times per second, or 4 Hz. Although the sensors aboard the Oculus I are able to provide data at a greater frequency, the Teensy 4.0's ability to do all of the tasks it needs to do, transmitting, writing, etc, was not able to be done with this high of a frequency. This began to produce lag/errors in the results being reported, and therefore the frequency was decreased.

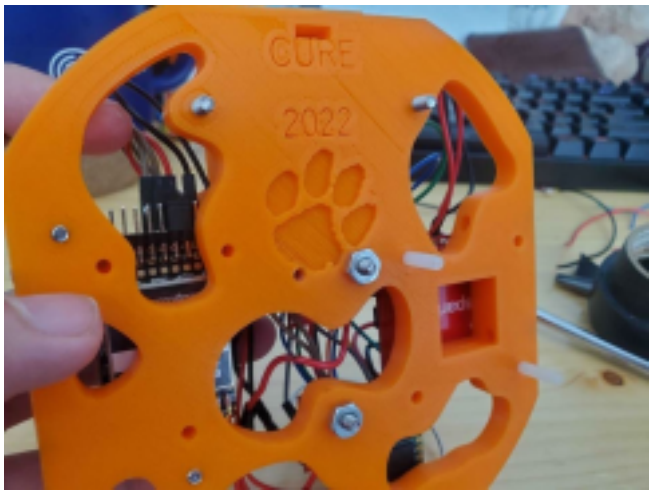


Fig. 9 SRAD system from below

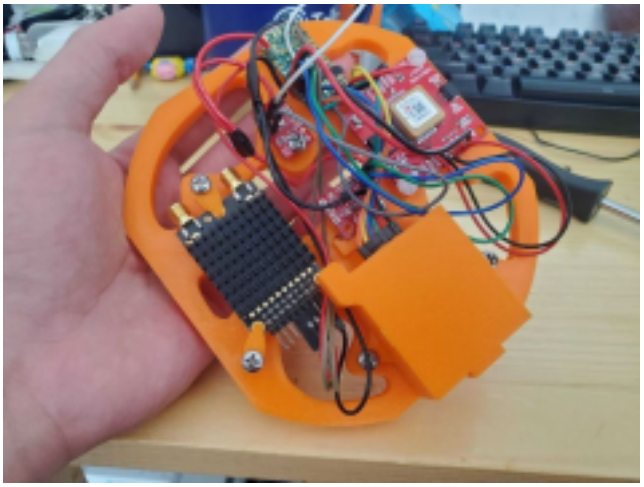


Fig. 10 SRAD System from above

Oculus I's avionics bay is turned on before it is loaded into the rocket itself. This initializes the system and enables all of the sensors in both the SRAD and COTS system but does not arm the system. To arm the system, a screw switch is used to indicate to the flight computer that launch is in the relative future.

Prior to installing the igniter, someone will approach the Oculus I and screw down the screw arming switch. Once the switch is enabled, the avionics system on the Oculus I enters a new mode where it is awaiting launch and collecting data in the meantime. Data collection begins on arming and not on detection of launch in order to preserve the data from the first moments of launch. If the flight computer waited until the launch was detected by the accelerometer, then the data from the initial portions of the launch would not be recorded and stored. This would not collect valuable data that could be evidence to anything that may go wrong during the flight.

Additionally, moments before launch, less than one minute, a team member present at the location with the base station will send a specific command to the Oculus I through the base station's antenna setup. This command will initialize the Raspberry Pi 0 Camera module. After the camera module is initialized, it will send a signal back to the SRAD avionics system of the Oculus I. Which, in turn, will then send a signal back to the base station, printing to the team member that the signal has been received and the camera has been initialized. This process is done for two reasons, the first being that upon initialization, the Raspberry Pi 0 camera module takes around 5 seconds to initialize and adjust to the lighting. Therefore, since the first few seconds of the flight are the most visually exciting, it is imperative that the cameras have time to focus before launch.

The second reason for the sending of a signal to the SRAD avionics system is to save on memory for the camera system. Even after arming, the amount of time a rocket may remain on the pad can be great. It would defeat the purpose of the camera system if the entire video consisted of the rocket sitting on the pad. Therefore, the rocket must begin recording close to the time of launch, so as the entirety, or most of the flight may be recorded.

The flight software of the Oculus I is programmed in the C programming language and was coded through the Arduino IDE. During the flight, the flight software onboard the Oculus I records the data it collects internally, along with transmitting this data to the base station. Since the Oculus I's SRAD system maintains the ability to deploy both the drogue and main parachutes, the altitude of the rocket is crucial to ensuring a successful flight.

The main concern of this comes with the problem of pressure differences. Only a relatively small number of holes may be drilled into the avionics tube in order to help with proper pressure balancing. Due to this, the pressure internal to the avionics bay may be not in line with the true environment pressure, given the altitude of the rocket at that moment in time. While drilling more holes in the avionics tube will compromise the structural integrity of the Oculus I as a whole, the problem must be solved through the use of hardware.

To solve this problem, the Oculus I's flight computer uses a time average of the measured altitude from the BMP. Since the BMP's altitude calculation is derived from the pressure inside the bay, this allows us to maneuver around the problem. The flight computer maintains a vector of five (5) pressure readings, stored as floating point values. The zeroth element of the array is filled first, followed by the first, and so on. If the average of the vector's values drops below a certain threshold near the targeted apogee for a specific amount of time, after reaching this threshold for a specific amount of time prior, the flight computer will signal to deploy the drogue parachute. The same time averaging process is implemented in regards to the main parachute at 1.5k feet.

While the Oculus I receives altitude data from the SAM GPS module, the SAM must maintain line of sight with the satellites it communicates with and has been found to be relatively sensitive to RF interference. Due to the aforementioned carbon fiber support structures, the SAM is mounted near the radial edge of the bay to minimize noise and interference. It was deemed that the SAM's failure modes were not only greater, but that the pressure difference associated with the BMP altitude sensing was an easier issue to control and counteract/prevent.

The Oculus I's most complex system in its flight computer is its ability to determine its orientation during flight. Normally, this would be done by using gyros and other sensors. Due to a significant level of gyro drift occurring within the sensors, another way needed to be built around this problem.

The solution to this problem was quaternions. Quaternions are mathematical representations that use 4D space in order to represent 3D space. Through the use of these mathematical objects, one can interpret the rotation and orientation of the rocket in real time. Additionally, this allows for the flight computer to output real time true acceleration values of the rocket, because of how quaternions work, they are not effected by outside acceleration, such as gravity.

While the Oculus I's main mission is to reach its target apogee and return to the ground in a safe manner, its secondary objective is to produce, store, and preserve data from all sensors present on the rocket. In the event of loss of communication to the Oculus I or in the instance of a catastrophic flight failure (ex: the rocket breaking in two), the preservation of data on the rocket is vital to not only partial recovery of the rocket, but also gaining an understanding of the underlying cause of the such a failure, should it happen. Due to this, the Oculus I has an onboard SD card, storing all of the data it records into a text file that may be read at a later date. This SD Card is stored in a special holder, shown in Figures 11 and 12.

This 3D printed structure allows for the ensured recovery of any and all flight data gathered by the Oculus I during its flight.

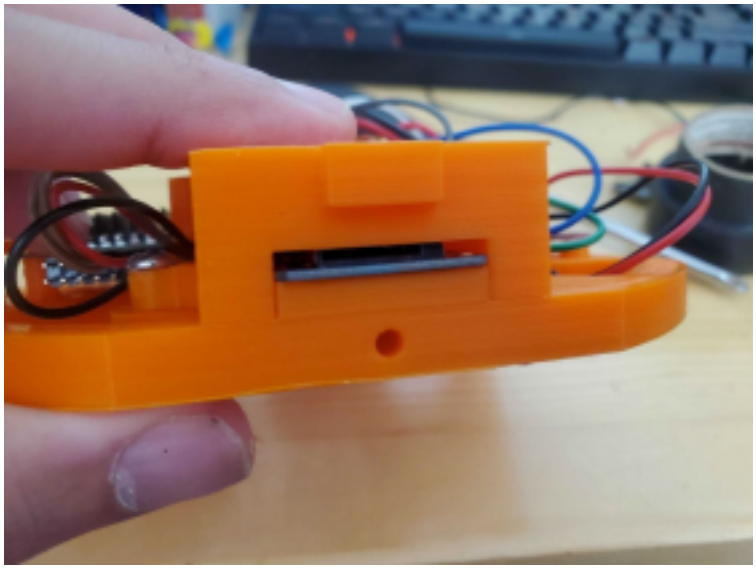


Fig. 11 SRAD SD Card Holder, Open

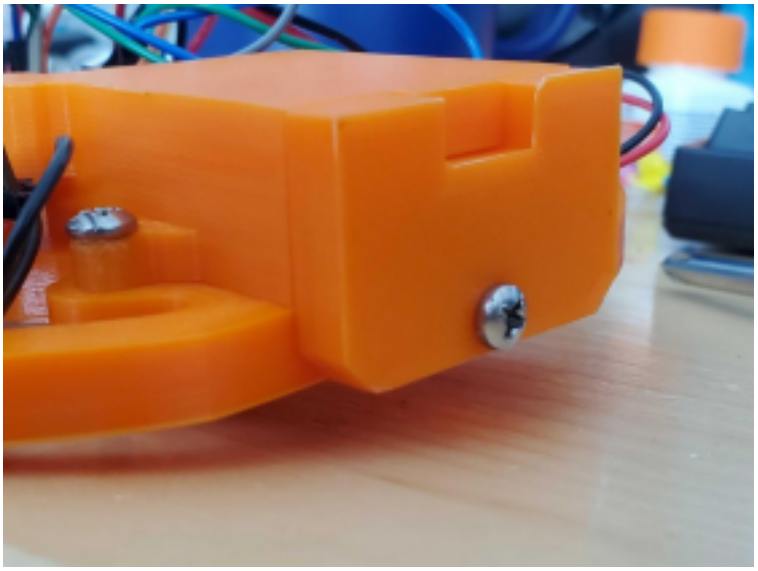


Fig. 12 SRAD SD Card Holder, Closed

Figure 13 shows the wiring diagram of the SRAD, COTS, and Raspberry Pi 0 systems present on the Oculus I. As mentioned before, a Teensy 4.0 is the main control unit of the SRAD system, while two AiM's (pictured as one single component since their purpose is redundant) control the COTS system present.

The Teensy interfaces with the BMP, SAM, and the MPU in real time, collecting and recording the data from the sensors. As mentioned before, and seen on the diagram, the power source for the Teensy and other SRAD components and the COTS AiMs are independent of each other. The "V_in" and "V_in2" pins designate different battery sources, as one can also see as denoted by table 4. Table 4 is a guide to the wiring diagram, showing each label name, its corresponding number in the KiCad teensy 4.0 footprint, as well as the actual pin number on the 4.0. Additionally, table 4 shows the source and destination of said label, as well as the pin number at the destination (where destination is assumed to not be the Teensy in the case of transceiving), and the purpose of each label.

Due to the limited use of components, such as the Raspberry Pi 0, as well as the lack of KiCad footprints provided by the manufacturer online, custom footprints were made and were simplified to using only the pins necessary. For these components, the footprints shown are not representative of the component's capabilities and its full utilization.

Additionally, each ejection relay is tied to an “Ematch,” which when brought high by the incoming signal by either the SRAD or COTS system outputs a large amount of current. This lights the Ematch, lighting the black powder charge inside the raptor CO2 ejector, breaking the pressurized CO2 canister and ejecting the parachute, whether that be the drogue or the main.

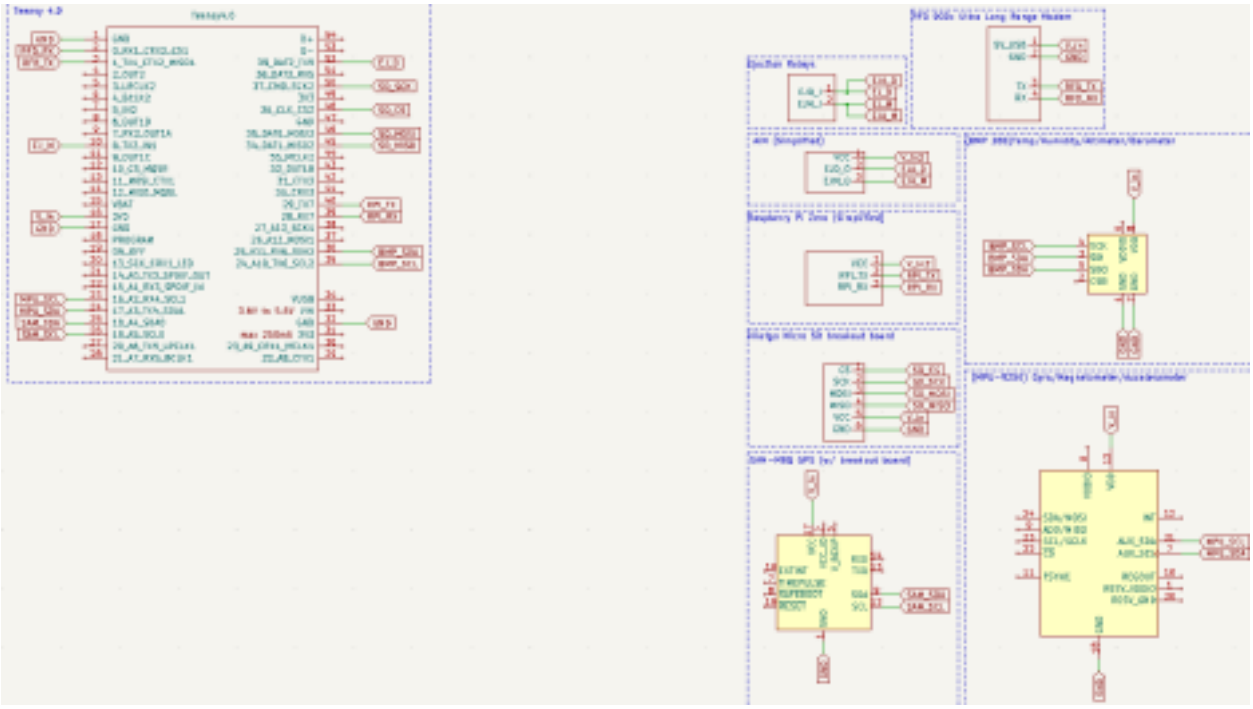


Fig. 13 Oculus I KiCad Wiring Diagram

Table 3 Key For Table 4

Key for Table 4

Power/GND
Raspberry Pi
MPU
SAM
BMP
AiM
Ejection Relay
Micro SD
RFD Modem

Table 4 Wiring Diagram Guide

Label Name:	Teensy Footprint Pin #:	Teensy Pin #:	Source/Destination:	Destination Pin #:	Purpose:
GND	1	N/A	GND	N/A	Grounding
RFD_RX	2	0	RFD Modem	4	RFD modem receiving
RFD_TX	3	1	RFD Modem	3	RFD modem transmitting
EJ_M	10	8	Teensy/Relay	2	Main Ejection Chute Signal from Teensy
V_in	16	N/A	Battery (1)	N/A	Power Supply
GND	17	N/A	GND	N/A	Grounding
MPU_SCL	23	16	Teensy/MPU	7	Slave Clock Timing
MPU_SDA	24	17	MPU or Teensy	21	Slave Data Transceiving
SAM_SDA	25	18	SAM or Teensy	9	Slave Data Transceiving
SAM_SCL	26	19	Teensy/SAM	12	Slave Clock Timing
GND	32	N/A	GND	N/A	Grounding
BMP_SCL	35	24	Teensy/BMP	4	Slave Clock Timing
BMP_SDA	36	25	BMP or Teensy	3 & 5	Slave Data Transceiving
RPI_RX	39	28	Raspberry Pi/Teensy	3	Receiving Signals from the Pi
RPI_TX	40	29	Teensy/Raspberry Pi	2	Transmitting Signals to the Pi

SD_MISO	45	34	SD Card/Teensy	4	SD Card Slave Data Sending Line
SD_MOSI	46	35	Teensy/SD Card	3	Teensy Master Data Sending Line
SD_CS	48	36	Teensy/SD Card	1	Chip Select Line
SD_SCK	50	37	Teensy/SD Card	2	Serial Clock Line
EJ_D	52	39	Teensy/Relay	Relay	Drogue Ejection Chute Signal from Teensy
V_in2	N/A	N/A	Battery (2)	N/A	Second Battery Source
GND	N/A	N/A	GND	N/A	Grounding
EJA_D	N/A	N/A	AiM/Relay	2	Drogue Ejection Input from AiM
EJA_M	N/A	N/A	AiM/Relay	3	Main Ejection Input from AiM

Base Station

The final section of the Oculus I's avionics system is the bay's corresponding base station on the ground. The base station consists of a Arduino Uno connected to two DRV8825 stepper motor drivers, which are attached to two stepper motors. The first of these stepper motors controls horizontal movement, rotating the Yagi antenna radially. The second stepper motor controls the vertical movement, rotating the Yagi antenna up or down, depending on whether the rocket is ascending or descending. The base station has three main purposes. The first of these is to record the sensor data from the components on-board the Oculus I. By transmitting and storing this data on the ground, as well as on the on-board SD card, Oculus I's avionics system gives a greater guarantee of recovering useful data, even in the event of catastrophic failure.

The second purpose of the base station for the Oculus I's avionics system is tracking of the rocket. The base station consists of a whip and Yagi-Uda antenna. A 3D MATLAB rendering of the Yagi antenna can be found in figure 14. Due to the Yagi's nature of not being an omni-directional antenna, per figure 15, a MATLAB 3D rendering of the antenna's gain, the vector of the Yagi's direction must be approximate to the location of the rocket in 3D space, in real time. Without such a system, the Yagi will not maintain a strong connection with the rocket.

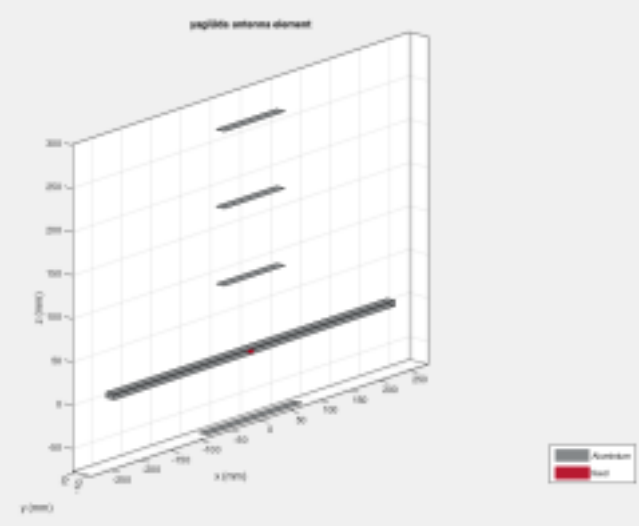


Fig. 14 3D rendering of the Yagi antenna's reflectors

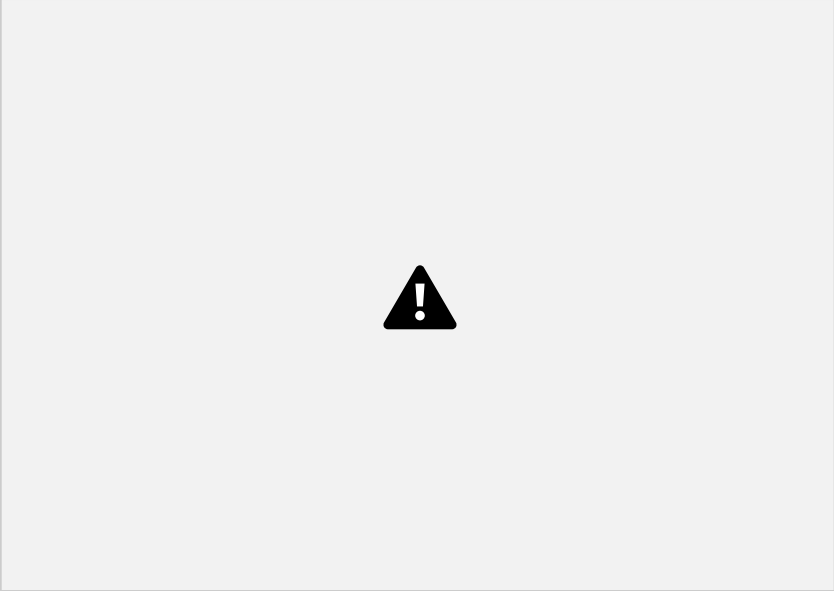


Fig. 15 Yagi 3D Gain plot

The final purpose of the base station is to enable the Raspberry Pi 0 camera system aboard the rocket. Within thirty seconds of launch, a user will enter a command into the base station, which when executed will be transmitted to the rocket and begin the recording of the flight. More info on the Raspberry Pi 0 Camera System can be found in the next section.

To communicate with the ground, Oculus I uses two radially mounted patch antennas operating in the 900 MHz bandwidth. These patch antennas are mounted so that a line between the two is perpendicular to a line between the two carbon fiber support structures of the avionics bay. This is to reduce/eliminate the RF interference from the support structures, due to them being made of carbon fiber. The presence of two antennas on-board the rocket allowed for full duplex communication with the base station.

On the ground, the base station uses a Whip antenna in conjunction with a Yagi antenna to maintain connection with the rocket throughout its flight. As discussed prior, while the Whip antenna emits a signal omnidirectionally, the Yagi antenna operates most effectively along a single direction vector. That is, the signal strength of the device communicating back and forth with the Yagi is maximized when the location to which the antenna points to is in line with the device it is communicating with.

Due to this need, the Yagi must implicitly track the rocket in 3D space throughout its flight. In order to facilitate this movement, the base station can rotate both radially and vertically in the dimension of spherical coordinates. Controlled by two OSM technology stepper motors, which are driven each by a DRV8825 stepper motor driver IC. Although the base station rotates in a spherical coordinate system, with and being the two coordinates of interest, it receives data in terms of rectangular coordinates, latitude, longitude, and altitude. Therefore, the base station must perform a series of trigonometry calculations in order to convert between coordinate systems.

The Oculus I's base station uses a predefined location for its own "home" coordinates, while taking in GPS data in order to determine the rocket's current horizontal location, relative to its start. As the rocket climbs in altitude and descends, it will drift, and therefore, without adjustment, the Yagi signal strength will decrease as the angle increases. To resolve this, the base station must calculate the difference in the angle with which is currently sits and where the rocket currently is.

The base station software is preloaded with the coordinates of where it will sit, defined as the red triangle in figure 16. The orientation of the base station will also be known, where latitudinal lines run perpendicular with the base station, while longitudinal lines run parallel. Since the launch pad we will be using is defined by when we launch, hardcoding a GPS value for the launch pad is not possible. The software will know the coordinates of the launchpad, defined as the GPS coordinates at time, t, equals zero (0), defined as the green circle in figure 16. Additionally, at any time, t, the base station will know the coordinates of the rocket, Rt, which is defined as the blue star in figure 16.

The base station must determine the angle between where it currently points and where it must point in order to point to the coordinates of Rt. In order to solve this problem, the base station must first calculate the change in latitude and longitude between the base station and the launch pad, since this will not change, this is only done once. Second, the base station must calculate the beta prime, β', variable to determine the angle between the longitudinal leg and the hypotenuse of the triangle. After that, for every t, the base station must calculate the distance from the launchpad to the rocket (distlr(t)) and the distance between the base station and the rocket (distbr(t)). With the values of the triangle relating the rocket and the launchpad, the angle lambda prime (λ') can be calculated to indicate the

For time, t, equals zero (0) only:

$$\text{[illegible]}_{\text{[illegible]}} = |\text{[illegible]}_{\text{[illegible]}} - \text{[illegible]}_{\text{[illegible]}}|$$

$$\text{[illegible]}_{\text{[illegible]}} = |\text{[illegible]}_{\text{[illegible]}} - \text{[illegible]}_{\text{[illegible]}}|$$

$$\text{[illegible]}' = \text{[illegible]}$$

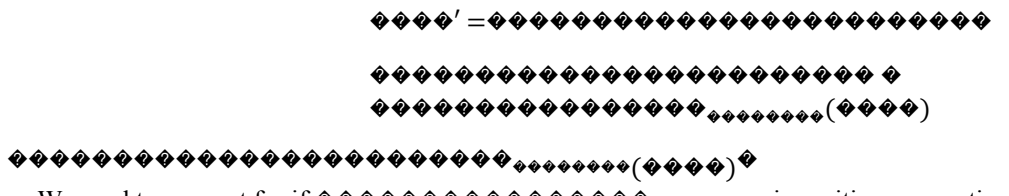
Each time, t, where $\text{[illegible]} \in \text{[illegible]}$:

$$\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) = |\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) - \text{[illegible]}_{\text{[illegible]}}| \;\&\; \text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) = |\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) - \text{[illegible]}_{\text{[illegible]}}|$$

$$\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) = (\text{[illegible]}_{\text{[illegible]}}^{2} + \text{[illegible]}_{\text{[illegible]}}^{2})^{1}{}_{2}$$

$$\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) = |\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) - \text{[illegible]}_{\text{[illegible]}}| \;\&\; \text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) = |\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) - \text{[illegible]}_{\text{[illegible]}}|$$

$$\text{[illegible]}_{\text{[illegible]}}(\text{[illegible]}) = (\text{[illegible]}_{\text{[illegible]}}^{2} + \text{[illegible]}_{\text{[illegible]}}^{2})^{1}{}_{2}$$

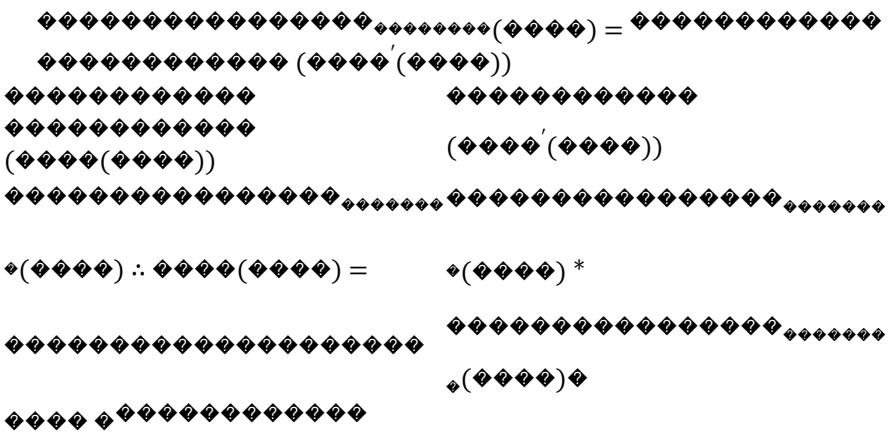


We need to account for if  is positive or negative:

Positive: $\text{﷽﷽﷽﷽}'(\text{﷽﷽﷽﷽}) = 180° - \text{﷽﷽﷽﷽}' - \text{﷽﷽﷽﷽}'$

Negative: $(\text{﷽﷽﷽﷽}'(\text{﷽﷽﷽﷽}) = 180° - \text{﷽﷽﷽﷽}' + \text{﷽﷽﷽﷽}'$

Law of sines:



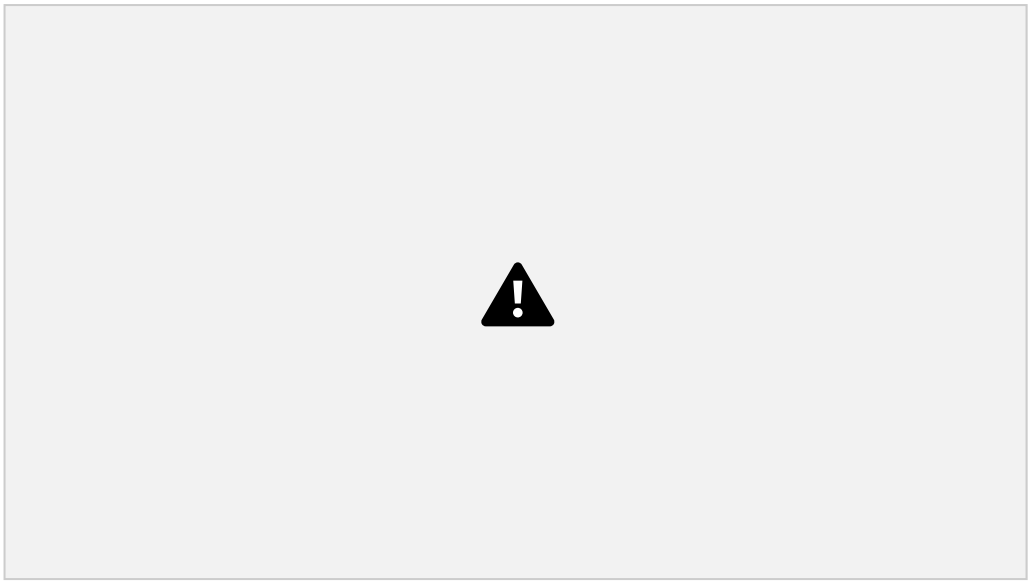


Fig. 16 Visualization of horizontal component of 3D tracking

This algorithm provides the angle, , between the base station relative to the launch pad and the base station relative to the rocket. While this piece of data is useful, it does not give us the mathematical info to implement any sort of tracking. To do so, we must compare the theta at time t and t-1:

$$|\square\square\square\square\square\square\square\square\square| = |\square\square\square\square(\square\square\square\square) - \square\square\square\square(\square\square\square\square - 1)|$$

If the absolute value of the change in the angle values is greater than 1.8 degrees, which is the smallest degree value that the OSM stepper motors can rotate, then the motor must turn in either direction. In order to learn whether to turn left or right, a simple IF statement is implemented in software:

```
if(theta > theta_old)
{
    /* Drive motor so that the station rotates left */
}
else if(theta < theta_old)
{
    /* Drive motor so that the station rotates right */
}
else
{
    /* do nothing */
}
```

To track the rocket throughout the flight, the vertical component of 3D space is vital to maintaining a strong and reliable signal with the Yagi antenna. Therefore, station must also take advantage of the altitude data that sent with the GPS data. Using the distance from the base station to the rocket, distbr which we found in our horizontal calculation, and the altitude of the rocket, we can find the angle, φ'. This angle represents the angle between the ground and the direction vector which would point the base station at the rocket and is a simple arctan function:

$$\square\square\square\square' = \square\square\square\square\square\square\square\square\

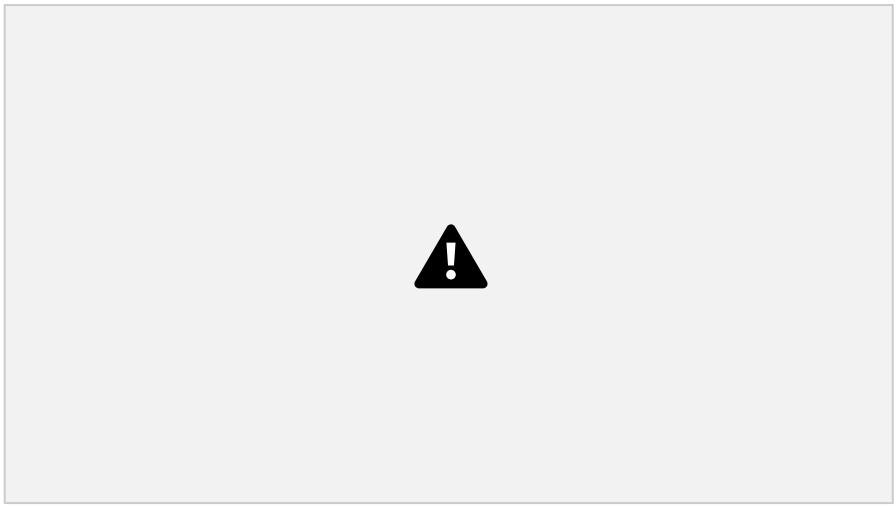


Fig. 17 Visualization of the vertical component of the 3D tracking

The same algorithm stated above allows us to indicate whether the station must rotate the Yagi antenna upwards, on ascend, or downwards, on descent:

```
if(phi > phi_old)
{
    /* Drive motor so that the station rotates upward*/
}
else if(phi < phi_old)
{
    /* Drive motor so that the station rotates downward */
}
else
{
    /* do nothing */
}
```

When calculating for the angle, φ', between the ground and the rocket at time, t, the curvature of the Earth was discussed as a possible variable that could cause significant error upon the system's ability to function correctly. It was found that, even at apogee, 10k feet, the effect of the curvature of the Earth upon the calculation of phi prime would only be approximately five feet of difference, an insignificant amount. Therefore, the curvature of the Earth was ignored in our calculations.

These two sets of calculations allows the base station to drive the stepper motors in the correct direction to rotate the Yagi antenna to the path of the rocket, thereby maintaining a strong connection. The Yagi, in conjunction with the Whip antenna, produce a system that is reliable through the Whip's omnidirectional ability, and is innovative through the Yagi's ability to implicitly track the rocket.

While the rocket will be transmitting data back to the ground at a rate of four times per second, due to the hardware limitations of the Arduino Uno and the mathematical functions of used in solving for θ and φ', especially, the base station will only adjust twice per second. While this may seem to be an inhibitor to the usefulness of the system, since the horizontal distance between the rocket/launch pad and the base station is large and the off the rail speed is relatively small, compared, in value only, to the aforementioned distance, the change in theta does not exceed any rate that would inhibit the system's ability to correctly track the rocket at a "refresh" rate of two times

per second.

Raspberry Pi 0 Camera System

A new system being attempted with the rocket this year includes a camera system to record the rocket's flight. The main component controlling the camera is a Raspberry Pi 0 that controls when the camera starts and stops recording. The camera itself is a Raspberry Pi camera v2 recording at an HD resolution at approximately 24 frames per second. Shortly before the launch, a signal will be sent from the main avionics system to the Raspberry Pi to tell it to start recording. The main avionics system is connected to the Raspberry Pi 0 via an input/output wire, which allows basic communication between the two components. The camera will record the entire flight until shortly after the rocket touches down. When the accelerometer detects the rocket has stopped moving, the main avionics system

sends a signal to the Pi 0 to stop recording. All video data will be saved on a 16 Gb micro SD card and analyzed after the flight.

Payload

Overview of Payload Narrative

The payload for the 2021-2022 Clemson Rocket Engineering Club was designed to be a 3U CubeSat that can stabilize its rotation in the roll axis after ejecting at 2000 feet during the descent of the main rocket. Following ejection, the payload, coupled with the nose cone, will descend under its own parachute at a velocity that is less than 20 fps. The payload and nose cone are hard mounted together in order to avoid potential failure modes. Due to its separation from the main rocket, the payload will have a Featherweight GPS tracker fixed to the outside of the housing in order to provide live tracking of its location during descent and at touchdown. The deployment of the payload is initiated by the main rocket avionics, but a barometric sensor within the payload will be used to activate the roll control system once it verifies the payload has been ejected and is falling under stable descent. During the descent, an onboard microcontroller will gather live angular velocity readings from an IMU that will be used to control the firing of two solenoid valves which will release pressurized air through two sets of tapered nozzles in order to create a coupled moment controlling the roll axis rotation of the payload during descent. The idea behind this year's payload was to create a way to guarantee roll axis stability for future payloads which may have a scientific experiment onboard. A 3U CubeSat form factor was chosen in order to meet the CubeSat physical standard with hopes that a 1U module could be left empty to prove a scientific experiment can be integrated into the payload in future years. The subsequent sections will go into more detail about the housing, the mission systems, and the avionics of the payload as well as the testing that was performed to verify different parameters of the payload.

Mission Systems

The mission system of the payload is made up of the cold gas thruster components used to control the roll of the CubeSat while it is descending under its parachute after ejection at main deployment. The propellant of the system is pressurized air stored in a 3000 psi, 13 cubic-inch, Ninja paintball tank. When the ball valve of the tank is depressed the air flows through a built-in regulator to go from 3000 psi to 750 psi. Following the exit of the tank, the pressurized air will go through ⅛" copper tubing into a ball valve which controls the flow of the propellant through the rest of the cold gas thruster system. The ball valve must be closed for the pressurized tank to be screwed into its adapter in order for it to not expel pressurized air upon full integration. Once fully installed, the ball valve can be opened to allow for the propellant to pass through a subsequent pressure regulator in order for it to be depressurized to 100 psi so that the normally-closed solenoids can function properly. A duel-exit pressure regulator was chosen so the singular propellant flow could be split into two in order to allow for control of the two separate coupled moments used to counter the CubeSat's rotation in both directions. Following the pressure regulator, the two separate flows will follow identical paths; each going through a solenoid which splits its respective flow into two more where each goes to its own nozzle. A propellant flow path visualization diagram, picture of the true mission system setup, and a chronological parts list are shown in Figure 18 and Table 5 respectively.

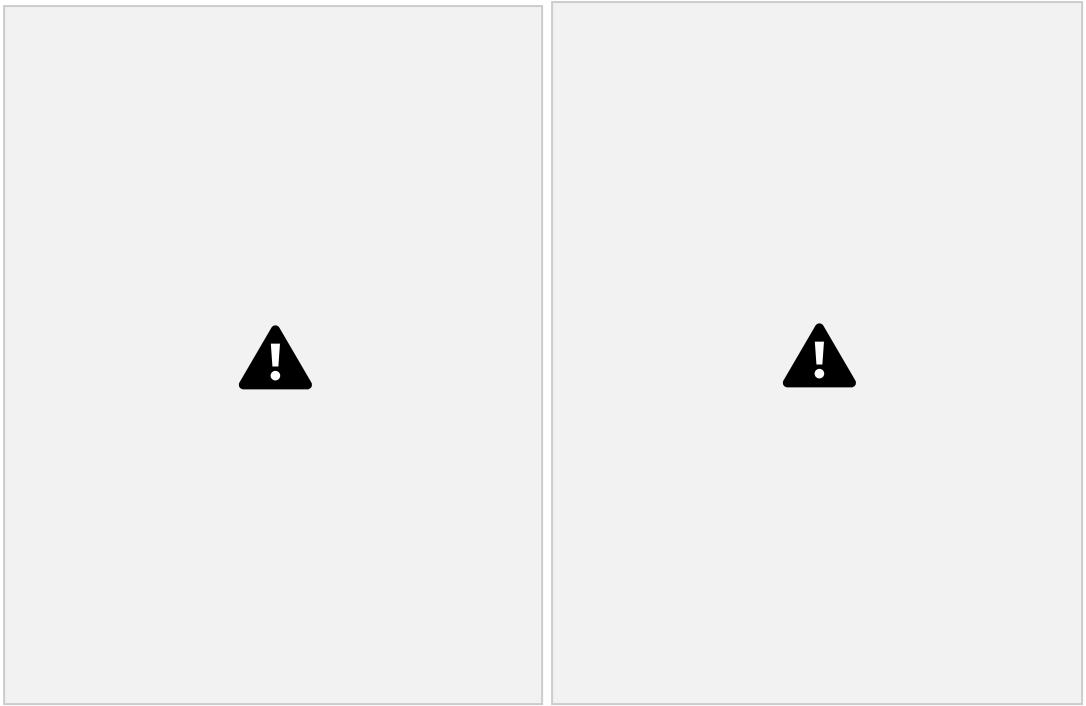


Fig. 18 Flow diagram of propellant through the mission system components (left); Actual payload mission system setup minus two nozzles and their tubing (right)

Table 5 Chronological parts list, in terms of propellant flow path, of the Payload's mission system

Item Part Number Item Description Cost/qty Quantity Propellant Tank ANS Gear: NinjaTank-13 Propellant vessel $69.95 1 Tank Valve Adapter AIH: 63-4224 0.825-14 NGO to ¼" NPT $7.34 1 90° Elbow Adapter Swagelok: SS-200-8-4 ¼" NPT-F to ⅛" compression $25.90 1 Ball Valve Swagelok: B-41S2 ⅛ "

compression to ⅛” compression $51.70 1 Straight Adapter Swagelok: SS-200-7-2 ⅛” compression to ⅛” NPT-F $11.00 1 Pressure Regulator Palmer Pursuit: PPSP017 750 psi to 100 psi $93.50 1 90° Elbow Adapter McMaster: 50785K411 ⅛” NPT-M to ⅛” NPT-F $3.35 2 Straight Adapter Swagelok: B-200-1-2 ⅛” NPT-M to ⅛” compression $3.70 4

Solenoid US Solid: USS-PSV00040 Engage/Disengage thrusting $14.99 2

Union Tee Swagelok: B-200-3TMT ⅛” M-NPT to 2x ⅛” compression $12.10 2 Nozzle n/a SLA Printed, tapered design n/a 4 Copper Tubing McMaster: 8967K86 ⅛” OD, 3ft length $4.35 2

Static copper tubing and compression fittings were used rather than flexible nylon tubing and quick-connect fittings to ensure the tubing would not disconnect during flight as well as to meet certain pressure requirements of the mission system. The two solenoids and four nozzles would bolt directly into the housing of the CubeSat, but the Ninja tank would be fixed into a 3D printed mount which would then be bolted to the CubeSat housing. Components such as the ball valve and pressure regulator would be secure without a permanent fixing method as a result of the rigidity of the static copper tubing.

The roll control of the CubeSat comes from nozzles in diagonally opposite corners that are paired to create the coupled moment used to counter the CubeSat’s rotation. Each pair of nozzles is driven by a solenoid which has a response time of less than 50 ms and is engaged and disengaged using a microcontroller by way of live angular velocity readings from an IMU and a relay. Based on the IMU readings, the microcontroller will send voltage to the appropriate relay to open or close the desired solenoid based on which way the CubeSat is rotating. More on the payload avionics is discussed in the following section.

Software (Avionics)

The main function of Payload’s avionics was to have a microcontroller (Arduino Uno) read live roll-axis angular velocity from an IMU (Adafruit BNO055) and use it to control the firing of solenoids based on the rotation speed and direction of the payload and nose cone assembly. Another key function of the avionics was to utilize a barometric sensor (Adafruit MPL3115A2) to verify the payload is ejected from the main rocket and is falling at a stable descent rate i.e. the parachute is fully deployed. Without this verification, the cold gas thruster system could activate while in the rocket which could cause the entire launch to be a failure. The arduino uno was powered by a rechargeable 9V 1000mAh LiPo battery. The solenoids require 12V each so a 6S LiPo was chosen to power them. The controls were actually quite simple and no control algorithms were needed. If the CubeSat was rotating counter clockwise (CCW), the nozzles that created a coupled moment forcing clockwise (CW) rotation would fire until CCW had ended. If there was overshoot, the other set of nozzles would fire. This would continue until the rotation of the CubeSat was less than 1 RPM in either direction. Figure 19 shows the payloads avionics wiring diagram and Table 6 outlines the process by which the Arduino will be activated and will begin performing the payloads mission.

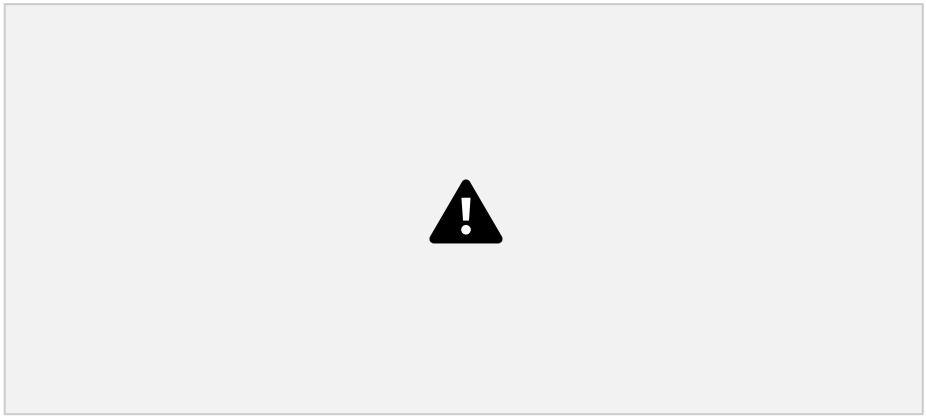


Fig. 19 Payload Avionics wiring diagram minus the 12V solenoids' power supply

Table 6 Microcontroller (Arduino Uno) Process

Stage	Description
Waiting to launch	Arduino is off
Rocket Assembly	The Arduino will be switched to on, at this point, the arduino should be in standby, waiting to begin receiving data from the 9DOF IMU. After this stage, there should be no more actions necessary to arm the payload bay.
Rocket Delivery	The Arduino must remain in standby, it will experience various movements and extreme temperatures, none of which should trigger the arduino to come out of standby mode.
Rocket Launch	The Rocket will launch, creating a large vertical acceleration, this is the action we will use in order to decide to bring the arduino out of standby mode.
Rocket Flight	At this point, the arduino should be on and recording acceleration data from the 9DOF IMU to the SD card, however, the solenoid valves should not be actuating at all.
Apogee	The arduino should continue recording acceleration data to the SD card
Drogue Deployment	The arduino should continue recording acceleration data to the SD card
Main Deployment	The arduino should continue recording acceleration data to the SD card, but should recognize this event through pressure changes caused by decreasing altitude and allow a 5 second run off after this event before beginning to actuate the solenoid valves.
Payload Descent	The arduino should continue recording acceleration data to the SD card, as well as controlling the solenoid valves to reduce all rotation.
Payload Burnout	The arduino should continue recording acceleration data to the SD card, however when all gas

is expended, the payload will no longer be able to correct its rotation, this event does not need to be recorded in any way in flight, and the valves should be allowed to continue to actuate.

Payload Landing When the payload touches down, all accelerations should turn to 0 (or close to) according to the 9DOF, and after the arduino recognizes that there is no acceleration data for over 2 minutes, it is required to shut off.

Because the payload is ejected from the main rocket body, it is required to have a GPS tracking system. It was decided that a Featherweight GPS tracker would be fixed to the outside of the CubeSat housing. During the descent, live coordinates will be read from the tracker and displayed by an iPhone application that is linked to said tracker. Even if the payload and nose cone assembly drift out of sight during descent, it will still be possible to find them using the GPS and iPhone application.

Payload Housing Design

The housing of the payload was designed to ensure that it would meet the standard dimensions of a 3U CubeSat; meaning it is 10cm x 10cm x 30cm in size and at least 4kg in total mass. The housing was designed to be manufactured from 6061-T6 aluminum plates and assembled using steel bolts. The 6061-T6 aluminum plates were chosen to be 0.25” thick in order to allow for them to be directly bolted to each other using 4-40 bolts rather than requiring the use of joining brackets. Any plate thickness less than 0.25” would not allow for this assembly method because the walls would be too thin to safely accept the 4-40 bolts. Because the CubeSat housing mimicked a hollow rectangular prism, and was not similar to a skeleton chassis where there are only corner beams with transverse support bracing, weight was a concern. Examples of the two, for context, can be seen in Figure 20.

Fig. 20 Skeleton Chassis CubeSat Housing (Left) [1], Solid Wall CubeSat Housing (Right) [2]

It was decided that weight reduction could be performed, if necessary, by milling material away from the housing walls where deemed appropriate, so this was determined to be a viable option. The solid wall CubeSat was chosen over the skeleton chassis CubeSat due to the ease of internal component integration that it would provide as well as the need for less total components for complete assembly. Also, manufacturing would be much simpler due

to less components to manufacture as well as the components having simpler geometries.

The swivel itself was designed by payload subteam members but also purchased as a COTS component. The payload team felt that a COTS swivel would not provide the ideal frictionless rotation environment for the CubeSat to showcase its roll control abilities so one was designed as well. By having a swivel with a bearing in it rather than a barrel swivel, rotation of the CubeSat would continue much longer before friction could slow it down. Both a Rosco 1000lb swivel and the SRAD swivel would be tested and compared to ensure the most viable option was chosen. As for the SRAD swivel designed by this year's payload subteam, an angular ball bearing was chosen in order to ensure axial loads would be suitable for the swivel. The bearing housing was custom designed to mount to the top of the payload using four 6-32 bolts. Due to limitations in angular contact bearing choices, a custom eyebolt shaft was designed and manufactured to fit the 10mm inner bore of the bearing while also being compatible with the limited eye bolt nuts available. Though the shaft was designed to be interference fit with the bearing's inner bore, grooves were strategically placed on the shaft to allow for the utilization of retaining rings to ensure the shaft does not slip out of the bearing. The only concern with this was that the snatch force of the parachute deployment would be too high for the retaining rings, but after performing the calculations seen below, the team was confident in the safety of the design. In the Equation 1 below, F is the snatch force of the parachute opening, ρ is the density of air at 1500 ft AGL, V is the velocity of the rocket at payload ejection, C_d is the coefficient of drag of the parachute, and A_m is the projected area of the parachute. In Equation 2, P is the thrust load capacity of the retaining rings, D is the diameter of the retaining ring, T is the thickness of the retaining ring, S_s is the shear strength of the retaining ring, and K is the factor of safety of the retaining ring which, for this application, is recommended to be 3.

$$F = \frac{1}{2}\rho V^2 C_d A_m = \frac{1}{2}(1.986 * 10^{-3}\ \text{slugs/ft}^3)(103.8\ \text{ft/s})^2(1.5 * 12.0637\ \text{ft}^2) = 193.69\ \text{lbf} \quad [1]$$

$$P = (0.0328\ \text{in})(.00328\ \text{in})(18148669.62\ \text{lbf/in}^2)P$$

$$P = P * P * P_{ys} * P$$

$$3 = 2044.66\ \text{lbf}_{ys} \quad [2]$$

The calculated thrust load capacity of the retaining ring is much larger than the provided value, therefore it was decided to use the provided value of 890 lbs for further calculations. With that known, the factor of safety of the retaining ring, which will be doubled up, is 9.2. Another concern was that the bolts used for assembling the swivel housing together and the bolts used for fixing the swivel housing to the payload could fail due to the snatch force. Preliminary calculations were completed and showed that with four bolts at each connection point, the factor of safety would be greater than 21 while using a minimum bolt size of 4-40.

$$P = \frac{P}{4}(P_{ys} - 0.9743\ P_{ys})^2, 4-40: P_{ys} = 0.112, P_{ys} = 40, 6-32: P_{ys} = 0.138, P_{ys} = 32 \quad [3]$$

$$P_{ys} = P_{ys}: P_{4-40} = \frac{193.69}{P_{ys}} = \frac{193.69}{4(0.006)} = 8026.53\ \text{psi}, P_{6-32} = \frac{193.69}{} \quad [4]$$

	$\frac{\text{[illegible]}}{4(0.009)} = 5329.80$ [illegible]	
	[illegible]: [illegible] $= 170000$ [illegible] 8026.53 [illegible] $= 21.2$, [illegible] $= 130000$ [illegible] 5329.80 [illegible] $= 24.4$	[5]

As mentioned previously, the parachute choice for the payload and nose cone assembly was a 48" classic elliptical chute from fruity chutes. This parachute will allow for a stable descent of approximately 25.96 ft/s^2 with the assembly weighing 14.5 lbs. To verify the fruity chutes descent rate calculator was accurate, the falling speed, V_d, of the payload and nose cone assembly was calculated as shown below. Further, the impact energy, KE, of the assembly was calculated as well.

	[illegible] $= \sqrt{2}$ [illegible] [illegible] $= \sqrt{2 * 14.5}$ [illegible] $* 32.17$ [illegible] / [illegible]^2 0.0765 [illegible] / [illegible]$^3 * 1.5 *$ 12.0637 [illegible]$^2 = 25.96$ [illegible] / [illegible]	[6]
	[illegible] $= \frac{\text{[illegible]}}{\text{[illegible]}} *$ [illegible]^2 $2 = (14.5$ [illegible] $/ 32.17$ [illegible] / [illegible]$^2) *$ $(25.96$ [illegible] / [illegible]$)^2$ $2 = 151.87$ [illegible] $-$ [illegible]	[7]

By verifying the values determined by fruity chutes for descent speed and landing energy, the payload subteam was very confident in the recovery design for the payload and nose cone assembly at IREC.

Recovery

The payload and nose cone assembly will be descending under their own separate 48" classic elliptical parachute with nylon lines after ejection from the rocket at 2000 feet AGL. Since the goal of the payload is to control its rotation about the roll axis, the connection point between its eyebolt and the parachute should be as frictionless as possible such that the payload can spin freely. If there is resistance in the swivel connection, the shock cord could wind up which would cause a torque, which could be avoidable, to be felt by the payload. This torque could cause issues when the payload is trying to adjust its rotational speed in the roll axis. Also, by allowing the payload to spin freely about its connection to the shock cord, the risk of the eyebolt unthreading itself becomes greatly reduced if not avoided completely.

Updated Payload Narrative

Due to the lack of feasibility of the cold gas thruster design, especially for future years' payloads, a new roll control system was developed and will continue to be developed throughout the summer leading up to IREC. The new design consists of the same payload housing with a new mission system and avionics. The mission system will be comprised of three reaction wheels which will be driven by brushless motors, similar to those on drones and

UAVs. The reaction wheels, which will be made of copper because of its high density, will use their rotation to output a torque opposite the rotation direction of the payload in order to counter its rotation. This works based on the conservation of angular momentum. As the rotation speed of the reaction wheel is changed, an equal but opposite change in the CubeSat's rotation will occur. This is evident in Equation 8, where CS is the change in the CubeSat's angular velocity, RW is the change in the reaction wheel's angular velocity, and IRW and ICS are the reaction wheels' and CubeSat's moments of inertia, respectively.

$$\Delta\omega_{CS} = -\frac{I_{RW}}{I_{CS}}\Delta\omega_{RW} \quad [8]$$

The reaction wheel parallel to the top and bottom of the CubeSat will control its rotation about the roll axis, and the reactions wheels in the yaw and pitch axes, which are perpendicular to the top and bottom faces of the CubeSat will be used to quickly mitigate the payload's swing under the parachute after it ejects at main deployment.

The mission system of the payload was greatly reduced in size and mass by switching roll control methods. Rather than needing a pressure vessel, copper tubing, various fittings, solenoids, etc., only three reaction wheels and three motors are required. This allowed the payload subteam to free up 1U of space in the CubeSat which could be utilized in further years for a scientific experiment. The reaction wheels will have a tapped hole which will thread onto the motor shaft and then be fully secured by screwing a locknut on top of it. The motor itself will be bolted to the housing. All three motors will be powered by a single 6S LiPo battery which produces 22.2V and 2600 mAh. The motors are 1800 kV, meaning for each volt they draw they can produce 1800 rpm. For a 1800 kV motor drawing 7.4V, it could produce 13,320 rpm. Based on preliminary calculations, this should prove sufficient to detumble the CubeSat

from its terminal angular velocity and then make adjustments to its roll-axis orientation. Further testing and simulations will be completed to establish more design credibility.

The reaction wheels' avionics functions very similarly to the cold gas thruster's avionics. The Arduino Uno reads live angular velocity values from the IMU, but this time the velocity for all three axes is required in order to control all three reaction wheels. The barometric sensor is used in the same manner as it was with the cold gas thruster system. Rather than sending outputs to one of two relays, as with the cold gas thruster system, a 4-in-1 electronic speed controller was chosen to control all three motors. A control algorithm is still being worked on to take the IMU data and certain reaction wheel, motor, and CubeSat design parameters and use it to create a stable system.

Manufacturing Methods

Composites

Tubes

Our airframe is composed of precisely cut and machined composite tubes built and refined entirely in house. The tubes are sized in a manner that allows precise placement of rocket subsystems within the airframe. An array of materials can be used to make up the tubes, however, carbon fiber and fiberglass were used because they are relatively cheap while having high strength to weight ratios. We developed a manufacturing process that was iterated throughout the past year, with the process used on the final body tubes being as follows.

First, the fiber was cut out to a width and length for eight complete wraps of fiberglass or four full wraps of carbon fiber and then wrapped around a spool to easily transition it around a cardboard blue tube mandrel in one piece. Then, the mandrel was wrapped in one layer of mylar sheet and secured with tape to allow easy tube removal from the mandrel. Once this was completed, the Aeropoxy, an epoxy resin and hardener, were weighed, respectively, to obtain the correct amount for a 100:27 resin to hardener ratio. The epoxy was thoroughly mixed with hardener, and one coat was painted onto the mylar on the mandrel. After this, the wrapping process began by slowly and straightly unwrapping the fiber from the spool and around the mandrel, being sure to fully wet the fiber with epoxy using a paint brush before continuing to wrap. This process was continued until the fiber was fully wrapped and wetted onto the mandrel, so a cylindrical shape was formed. A layer of peel-ply sheet was wrapped around the completed tube and secured with tape to ensure consistent surface finish and to negate contamination during curing. While curing, the mandrel was placed vertically in an undisturbed environment for at least 48 hours,

shown in Figure 21.

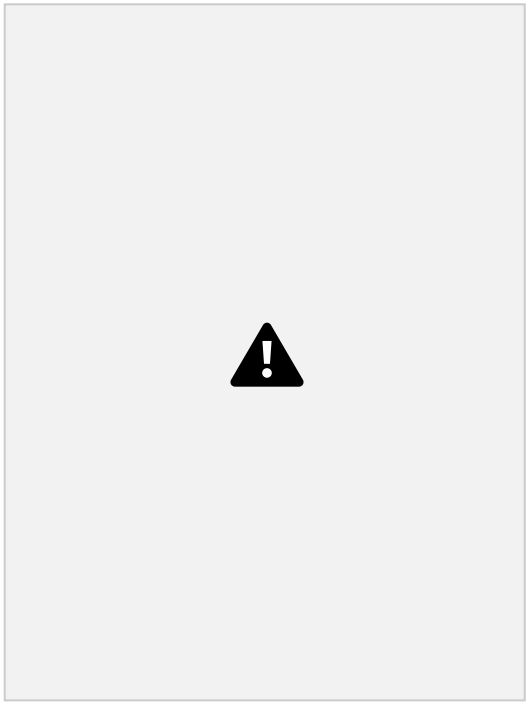


Fig. 21 Carbon Fiber Composite Tube After Curing and Removing Peel-Ply & Mylar

One of the biggest issues encountered during the manufacturing process was when we were unable to separate the body tube from the mandrel. There were many reasons for this occurring including wrapping the initial mylar sheet too tightly around the mandrel, wrapping the mylar sheet too loosely around the mandrel, and wrapping the fiber too tightly around the mandrel. If the mylar sheet was too tight around the mandrel and was thus unable to slide off the mandrel, then the only way to remove the body tube was to break the adhesion of the epoxy and the mylar by force which generally proved difficult due to the significant surface area connecting the two. Alternatively, if the mylar sheet was too loose, a wrinkle could form which the composite would then conform to, creating a sort of key, making it impossible to remove the body tube from the mylar, and the wrinkle may take up all of the slack in the mylar, once again making it so the mylar could not slide off the mandrel. Finally, even if the mylar was correctly wrapped around the tube, if the fiber was wrapped very tightly around the mandrel it would be extremely difficult or impossible to slide the body tube off the mylar or the mylar off the mandrel. These issues were resolved by ensuring that the mylar sheet was able to slide on the mandrel, but without an extra slack in the sheet, and by making sure that only the weight of the fiber itself was providing wrapping pressure, instead of creating tension in the fiber sheet while wrapping.

Another issue encountered during the manufacturing process was that the body tube rarely cured in a perfectly circle shape, sometimes by as much as an eighth of an inch too wide or short. This deformation was caused by allowing the body tube to cure on the mandrel horizontally, so the mandrel would deform slightly under its own weight causing an elliptical cross section. This issue was fixed by curing the body tube on the mandrel vertically, allowing the mandrel to maintain its circular shape.

A problem encountered after manufacturing a tube was the inside of the tube would not be adequately wetted out. This is because during the layup of the carbon fiber a conservative amount of epoxy resin would be used. A dried out tube runs the risk of being more likely to break or being torn through. This can be an issue in a situation like motor startup where the tubes tear and then there is a loose motor. This issue was fixed by being more generous with the epoxy resin application and making sure the fibers are well saturated. This, however, led to more issues of being able to get the tube off of the mandrel. This is because the extra epoxy coating made the wraps heavier and tighter to the tube, which left little room for the tube to be pulled off. Along with this, the peel-ply sheet

would not peel off of the outside of the tube correctly. The peel-ply sticking resulted from a combination of a tight wrap around the tube and pushing the peel-ply down by hand to remove air pockets across the wrap.

After the tube was wrapped it would be finished off by cutting down, sanding, and drilled for assembly and subsystem incorporation. First the tube would be measured and marked to the length it needs to be according to the design and cut down on a horizontal band saw. The cut edges would then be evened and smoothed with a dremel. Following this, the tube was then sanded down to a smooth finish to remove cured epoxy pulled through the perforated peel-ply on the outer surface and to prepare for painting. Holes then had to be made for securing the thrust plates which at first was a challenge to find a way to evenly and consistently make the holes. A solution for this was designing and 3D printing a jig for multiple hole patterns that can be placed on the tube so making holes would be even around the tube and at the correct spots. The jig would be placed a certain distance from the edge and the holes would be using a drill. To ensure the holes were correct the assigned thrust plate would be inserted and lined up and tiny drill bits would be stuck in to see if the holes were lined up. The issue with the method of not using the jig would be that holes often wouldn't line up and they would have to be redone. To fix the incorrect holes, they would be filled with epoxy resin.

Fins

The fins are entirely attached to the exterior of the motor tube, meaning a solid attachment to the tube is required. The fin core consisting of a sheet of ⅛ inch carbon fiber was first attached to the body tube using 5-minute epoxy and a 3d printed jig to ensure the fins were attached in equal intervals around the rocket and with as little fin cant as possible. Once the fins were attached, g5000 "Rocketpoxy" was used to complete the filet of the fin to the motor tube. The edge of the filet area was taped off to remove excess epoxy easily. Epoxy was then mixed and allowed to sit, resulting in a thicker consistency. The epoxy was then added to the filet in blobs and covered in a layer of mylar. PVC pipe was then clamped to form the desired shape of the filet. The PVC and tape were removed after allowing the epoxy to cure partially. After the filets, seven carbon fiber layers are used to complete a tip-to-tip lay-up. The tip-to tip lay-up was completed similar to the method described in Jim Jarvis' "The Jarvis Illustrated Guide to Carbon Fiber Construction." Because of the numerous layers of carbon used in the lay-up, each layer of carbon grew in size, getting closer to and eventually reaching over the end of the fins. The extra material was cut off and then sanded to have a smooth finish with the edge of the fin.

Avionics: Main Frame

The avionics bay consists of 3 major components; the bulkheads, the towers, and the shelves. The bulkheads were constructed as described in (the bulkhead section). To complete the carbon fiber towers, sheets of ⅛ inch carbon

fiber plates were cut on a 2.5d CNC machine. The carbon fiber plates are attached to the bulkheads with commercially available brackets. The shelves are 3d printed and are designed to fit the components required for the avionics bay. **Aluminum**

Bulkheads

The main airframe of the rocket consisted of the tubes previously discussed and the internal bulkheads, which are hard mounted points in the rocket, which allow for the precise and rigid placement of internal components such as payload, avionics, motor and ejection charges. These plates were all machined from 6061-T6 aluminum. The thickness of the plates were driven by the loads the plates would experience, where the failure modes of the plates were at the bolts securing the bulkheads to the airframe. Throughout the design, the minimum number and size of bolts used were 4 x 6-32 bolts and the maximum were 6 x 10-24. Having determined the necessary thickness for the radial bolt patterns, the face holes were then designed such that whatever components would be mounted to the bulkheads were in the proper location.

With the design of the bulkhead finalized, the team then laid out the appropriate steps that would be used to machine the bulkheads. All bulkheads were made by the team, in house, on a manual mill and/or lathe. The manual lathe was used to get the stock aluminum down to the appropriate radius which varied based on the location of the bulkhead (inside coupler or not), and then the plates were faced to get the bulkheads to the appropriate thickness. From here, the plates were taken to the manual mill where radial holes were drilled using a spindexer for accurate radial hole locations. Now having the appropriate radial holes, thickness and diameter, the plate was rotated in the manual mill vice and face holes were drilled having referenced the vice and the sides of the plate to get an accurate center location of the plate. From here, if necessary, the plates were placed back on the lathe for boring. The final step for any bulkhead was to then carefully hand tap any radial holes or face holes that required the operation.

Payload: CubeSat

The housing consists of 4 side walls (two different designs) and a top and bottom plate. For our first iteration, the housing parts were created using a stock 6061-T6 aluminum plate of 0.25" thick. After using a horizontal bandsaw to roughly cut out the shape of each part, the edges were then faced with a manual mill. Once the edges were faced, the part was truly straight, and detail could be added to it. The detail as depicted in drawings found in Appendix F
was then added using the same manual milling machine and drill presses. Going forward, a CNC milling machine or water jet will be used to add further detail, if needed to reduce mass.

Fig. 22 CRE Members Using Manual Mills

All internal components are COTS and will be attached to the housing walls, except for a few 3D printed parts used for attaching components together and aligning parts. The full CubeSat rendering is shown in Figure 23.

Fig. 23 Payload Housing Solidworks with High Detail

Payload: Swivel

First, a piece of 6061-T6 aluminum stock was turned down to the specified diameter using a manual lathe. Next, the center through-hole was bored out using the same manual. The lathe was then used to create the tenon and afterward the mill with an indexer was used to drill out the hole. These holes were then tapped to fit 6-32 bolts.

Fig. 24 Payload Swivel CAD Rendering

Conclusion

Clemson Rocket Engineering is a relatively new program at Clemson University and at the Spaceport America Cup. The school does not have an aerospace program, let alone any facilities to safely develop a rocket motor of any kind. All that being said, the team is incredibly proud to share Oculus I with the international rocketry community at the 2022 Spaceport America Cup. In addition to having the teams first ever fully operational payload, the team has also developed a ground station to record and view live telemetry, as well as developed the groundwork for an active drag system that although will be inert on Oculus I, it will further enhance the teams ability to compete at a high level in the coming years. This year has shown that there is a continued increase in the desire for an aerospace program of any kind to be brought to the university and through the continued success of the program at the competition each summer, the program only expects to grow more. One issue that is already being addressed is the lack of sponsors and community outreach, where the leadership team this year failed to acquire sponsors or conduct thorough community outreach. That, however, was a result of the team coming out of two years of disorganization and confusion, and the elected officials for the coming year have a very clear plan as to how CRE will grow its footprint in the Southeast by collaborating with schools, programs, and companies in the area. The team looks forward to competing at the Spaceport America Cup in June of 2022, and hopes to show the international sounding rocket

community that Clemson University does not just have rocket enthusiasts, but also has extremely talented young engineers who are driven to produce the best possible rocket year in and year out.

Acknowledgements

CRE would like to thank Dr. Garrett Pataky, Jessica Lang, Nick Stancil, Clemson University Machining and Technical Services, Peter Tarle, The Citadel Rocketry Team, the Clemson University Student Government, and the Clemson University Department of Mechanical Engineering for their continued support of our program. Without the help of these individuals, departments, and programs, we would not be able to participate in the Spaceport America Cup, and many students would lose out of the only opportunity at Clemson to develop any form of aerospace experience for their resumes.

Appendix

Appendix A. System Performance Analysis

Firstly, to analyze the stability of the rocket the stability margin is graphed with respect to

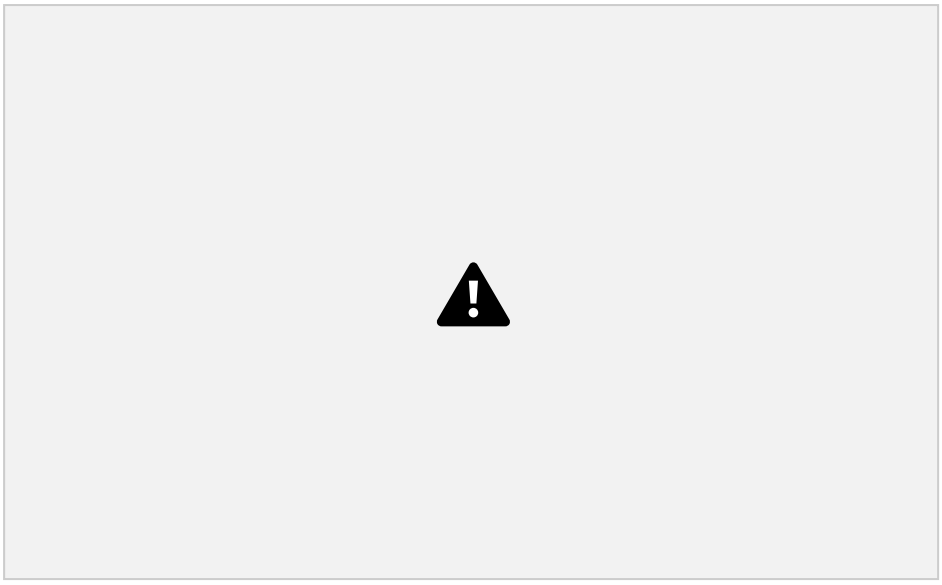
time. 

Fig. 25 Stability Margin vs. Time

As can be seen in Figure 25, the stability margin is initially 1.0, and increases to 1.5 in less than a second after launch. Additionally, the rocket's stability margin continues to increase as time goes on due to the motor ejecting fuel which shifts the center of gravity further away from the center of pressure. This suggests that the rocket is stable and will be stable for the duration of flight.

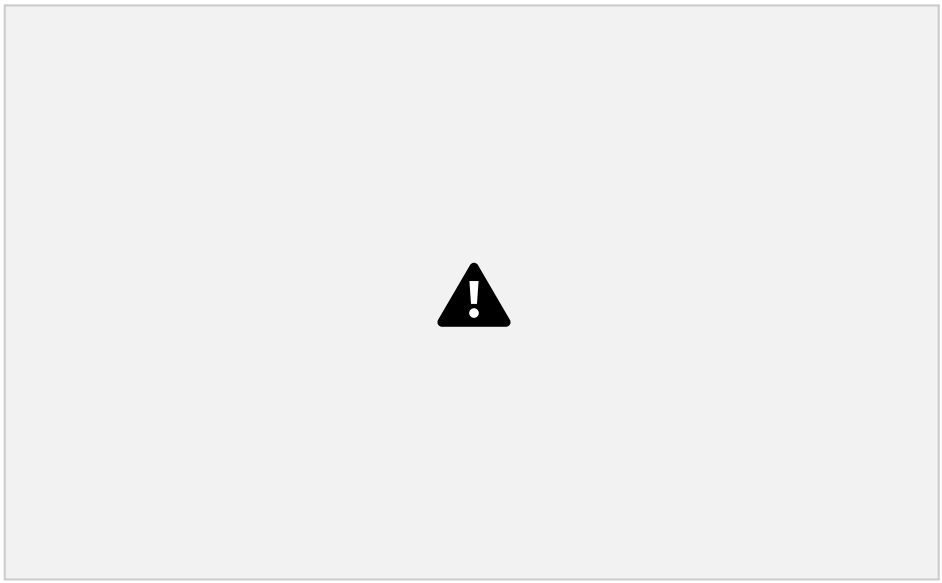


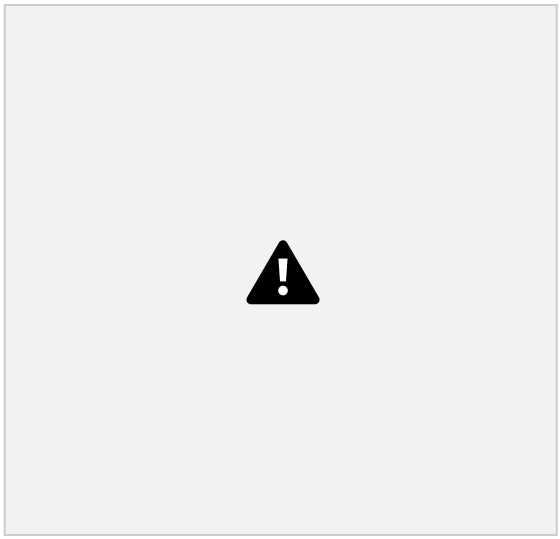
Fig. 26 Altitude vs. Time

From Figure 26, it can be seen that the flight of the rocket follows a normal trajectory, and that parachute deployment will allow the rocket to make a steady descent back to the ground. This further suggests that the rocket is stable for flight. Lastly, the off-the-rail velocity of the rocket is found to be less than 100 ft/s, so further analysis is needed to ensure that the rocket will be stable despite any flight conditions that may occur at launch. To address this, historical wind speed data was gathered from 2018-2021 and was used in conjunction with Openrocket software, and

the multi-level wind plugin to determine the stability of the rocket. The process is as follows. Firstly, historical wind speed data was gathered from 2018-2021 from June 21-25, which corresponds to the available launch dates for IREC

this year. Refer to Table 7 for an example of this collected data.

Table 7 Historical Wind Speed Data June 21, 2018 at 11 am



This collection of data is done for the following times: 5 am, 11 am, 2 pm, and 5 pm. We believe that these time periods are the most likely for launch, and each of these corresponds to important criteria. If launch is available at sunrise, this would be the optimal choice for launch due to wind speeds being the lowest at this time. To account for the possibility of this option not being available, wind speed data is also collected for the times of 11 am and 2 pm. Launch conditions are most likely to be the least optimal in the afternoon due to surface wind speeds naturally increasing as the day progresses until evening. To account for this, wind speed data was also gathered for the time of 5 pm. This data is then put into OpenRocket's multi-level wind plugin to determine the attack angle over time. Refer to the Figure 27 for the resulting graph.

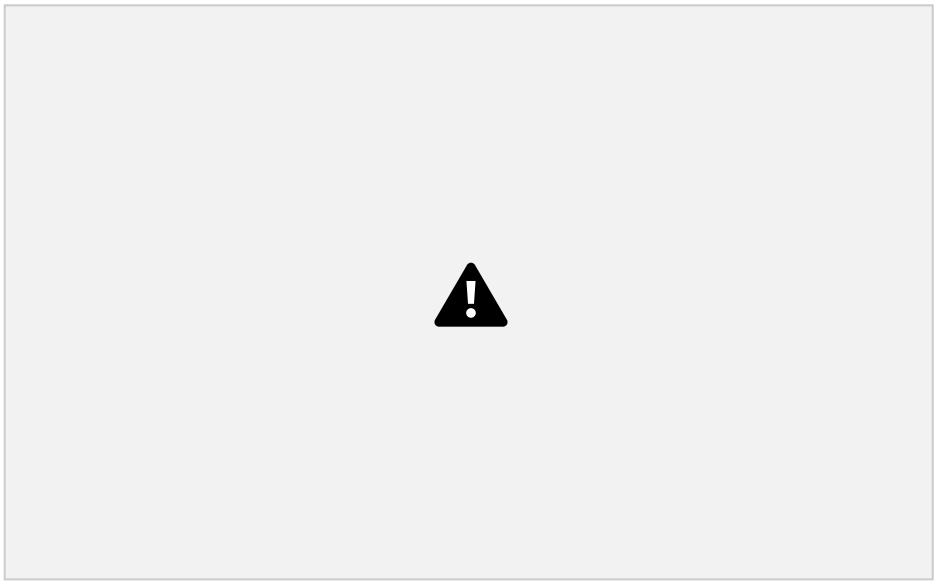


Fig. 27 Attack Angle vs. Time

After this graph is generated, the logarithmic decrement method is used to obtain the damping ratio of the rocket to

determine its stability given specific wind conditions. The equation below is the expression for the logarithmic decrement method.

$$\delta = \frac{1}{n} \ln \frac{x(t)}{x(t+nT)}$$

$$\zeta = \frac{\delta}{\sqrt{(2\pi)^2 + \delta^2}}$$

Where n is the number of the last peak analyzed, which in this case is the third peak, x(t) is the overshoot of the first peak, and x(t+nT) is the overshoot of the last peak analyzed. After obtaining , the damping ratio, , can then be found. The equation below is the expression for the damping ratio.

$$\zeta = \frac{1}{\sqrt{1 + \left(\frac{2\pi}{\delta}\right)^2}}$$

In this case, is 0.138, which means that the rocket is stable for these wind conditions, as it is below 0.3, and above 0.05. This process was done for all of the historical wind speed data gathered. Below, Tables xx - xx contain the minimum and maximum values of for each year and time.

Table 8 Damping Ratio of Variable Wind Speed at 5 am for 2018-2021

Table 9 Damping Ratio of Variable Wind Speed at 11 am for 2018-2021

Table 10 Damping Ratio of Variable Wind Speed at 2 pm for 2018-2021

Table 11 Damping Ratio of Variable Wind Speed at 5 pm for 2018-2021

Referencing the figures above, it can be determined that the damping ratios are within the acceptable range of values which suggests that the flight will be stable despite any wind conditions on launch day and time

Appendix B. Project Test Reports

Ejection Testing

The ejection test for the main parachute is in progress. Two 33-gram Raptor CO_2 Ejection Systems will be located at the top and bottom of the avionics bay (four Raptors in total). The front two will deploy the main parachute and payload, and the bottom two will deploy the drogue parachute. Once fired, the Raptors will pressurize their respective sealed chamber. The force on the bulkhead due to pressure in the chamber will be enough to fracture shear pins, separating the body tubes and deploying the parachutes. Only one Raptor is needed for the pressurization of the chamber, however, opting for two will ensure a high chance of success in case one of them fails.

In the testing of the ejection, the Raptors will be fired via a 9-volt battery. This test will not be using any simulated data to test altitude deployment; this test is to ensure that the pressurization of the chamber is enough to fracture the shear pins and thus ensure a successful deployment of the main parachute/payload. Currently, one of the bulkheads on the avionics bay needs to be turned down via lathe to fully fit inside the avionics' body tube. Once this is completed, the avionics and payload body tubes can be fully assembled. If the payload is not completed in time for the ejection test, an object of similar volume will be used to simulate it (as the main concern for pressurization is volume, not mass). Each of the bulkheads will be sealed off with silicone-based putty.

For a successful ejection test, it is necessary that each of the shear pins fractures, and that the main parachute deploys out of the avionics body tube. The test will be conducted horizontally and on stands for simplicity. The test will also be performed over grass, which will allow components to land on a soft cushion. The expected completion date of this test is May 15th.

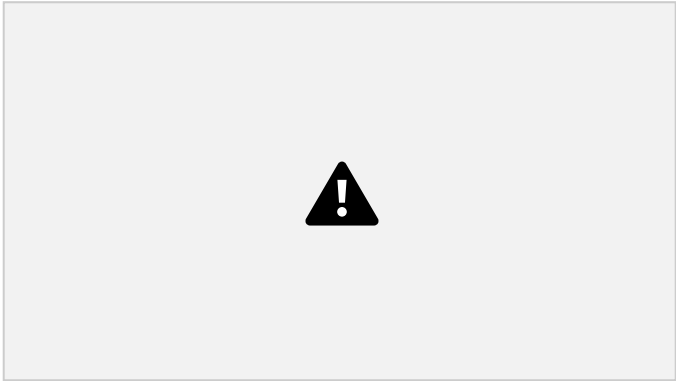


Fig. 28 CRE Member Preparing for Ejection Test (05/09)

Systems Integration Testing

The plan for integration testing is as follows. The first step was to secure the nose cone to the payload tube. The payload was then secured to the payload bulkhead. The avionics bay was then slid into the avionics tube to test the fit of the bulkheads that cap the avionics bay. One of the bulkheads did not fit well so the bay was disassembled, and the bulkhead turned down on a lathe to ensure a proper fit. Once the payload and avionics bay were in position the parachutes were packed and loaded in the space between avionics and payload. The motor was then slid into the motor tube and the bulkheads and thrust plate were then bolted to the rocket. Before the top half of the rocket and the motor tube were joined together the main drogue parachute was placed on the top of the motor. Then the two halves were joined together to form our completed rocket.

Range Testing

The modem being used on the rocket, a RFD 900x, has an approximate range of 40km, far more than the rocket will need to be able to communicate. The furthest possible distance the rocket should reach, a likely combination of the apogee and its horizontal component, is likely to be approximately 4 km from the base station it's communicating with. This is about 10x the estimated distance the rocket should be able to communicate. While complex tests of the limits of the range have not been fully tested, basic tests have concluded that the rocket should be able to communication during the entire duration of its flight. As a redundancy, the avionics system will be storing all data gathered onto the hardware in the event a communication loss happens. The featherweight GPS, which is used for the recovery of the rocket, does not need to be connected to the modem like the rest of the avionics bay. This ensures a constant connection to a GPS system to guarantee a recovery of the rocket.

GPS Testing

The testing of GPS components are essential to an ensured recovery of the Oculus I. By being able to track with coordinates the nearly exact location of the Oculus before and after the flight, recovery is expediated and guaranteed. Due to this, testing of the SRAD GPS, SAM-M8Q, consisted of multiple tests, each increasing in complexity and scale.

The first test of the SAM GPS was a standard connection and printing procedure where the SAM printed its location, along with its other ancillary values. In this test, the GPS remained in one location and was not transported anywhere. The goal of this test was to gain a greater understanding of the SAM and its operation in a closed and controlled environment, minimizing the risk of error or other hinderances. This test was at first not successful, but was retried and was completed, as shown in figure 29.

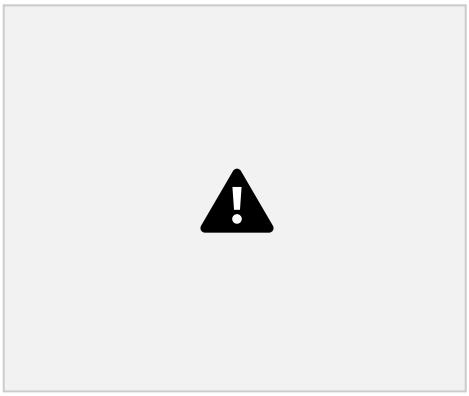


Fig. 29 Stationary GPS Output Test

The second test of the SAM GPS was a stationary test in a non-controlled environment. This consisted of getting in a car and traveling to a parking lot, and initializing the GPS, seeing if the output was consistent. It was expected that this test would be passed with ease, and it was, as the only difference between this test and the first test was the location.

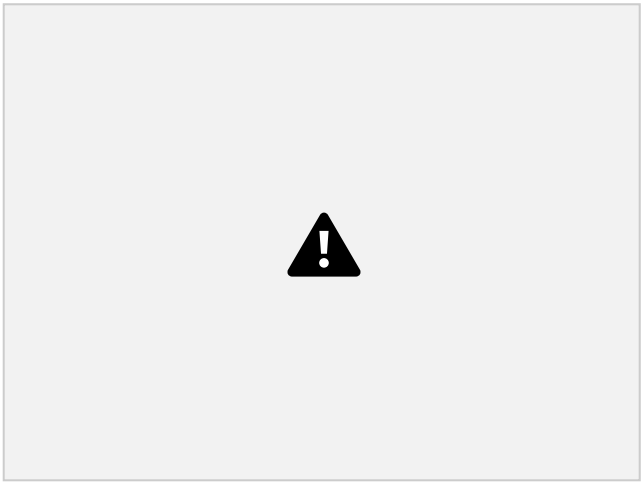


Fig. 30 Stationary Car GPS test

The final test of the SAM GPS module was a moving GPS test connectivity test. This test was to determine the accuracy and connectivity of the SAM GPS during movement. This test was conducted by initializing the SAM GPS and then driving around to various locations at various speeds. The SAM GPS completed this test, showing that

it was a reliable choice for the Oculus I's SRAD GPS. A photo of the output can be found in figure 31.

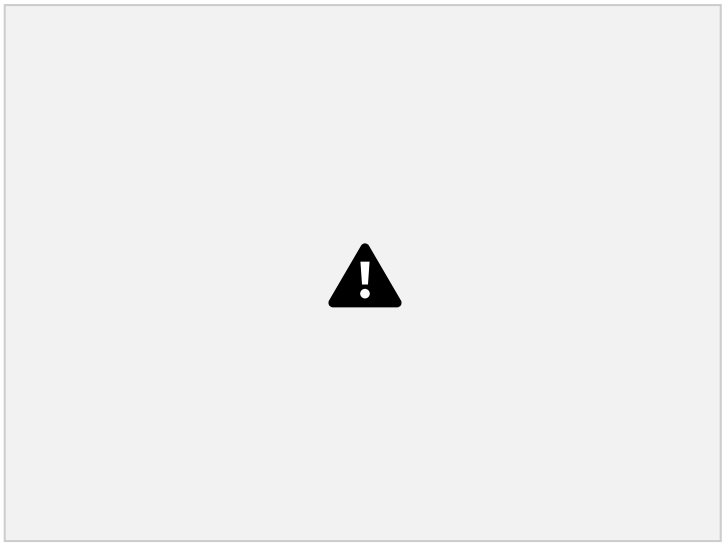


Fig. 31 SAM GPS Moving Test Output

Since the Featherweight GPS module is a COTS system, a basic connectivity test was conducted to ensure that the component received was not defective. After that, the Featherweight was deemed ready for flight aboard the Oculus I, due to the accreditation of its manufacturer.

Payload Testing Discussion

For the 2021-2022 payload roll control design, six different tests were developed to verify the validity of different performance parameters of the design ranging from swivel load durability and internal component temperature resistance to the actual roll control of the CubeSat. The "Nozzle Thrust Test" (Test 1) and the "Pressurized Tank Duration Test" (Test 2) were completed simultaneously using a strain gauge which was outputting readings to an arduino uno, see the wiring diagram in Figure 32. The results of the Nozzle Thrust Test are in Table 12.

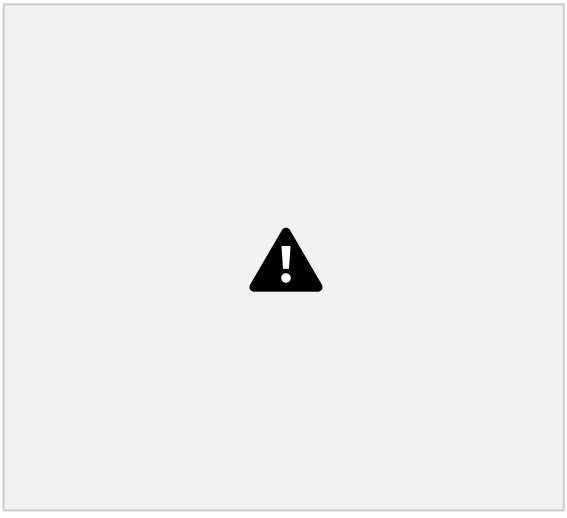


Fig. 32 Wiring diagram for the strain gauge used in Test 1 and Test 2

Table 12 Results of the Nozzle Thrust Test

Trial #	Starting Force [g]	Avg Measured Force [g]	Force of Nozzle [g]
Trial 1	-40	-18.70	21.30
Trial 2	-30	-8.13	21.87
Trial 3	-26.5	-4.69	21.81

The starting pressure in the tank was 2750 psi which was only 250 psi short of the maximum pressure rating for the tank. Each trial was run for only 2 seconds and the measured force values were averaged from the readings collected during that time. After three trials the tank was depleted. From that, it was determined that the pressurized tank could only provide optimal pressure for one nozzle for 6 seconds. This meant that for all 4 nozzles, the tank would only have enough air for approximately 3 seconds of continuous use. The test setup is shown in Figure 33.

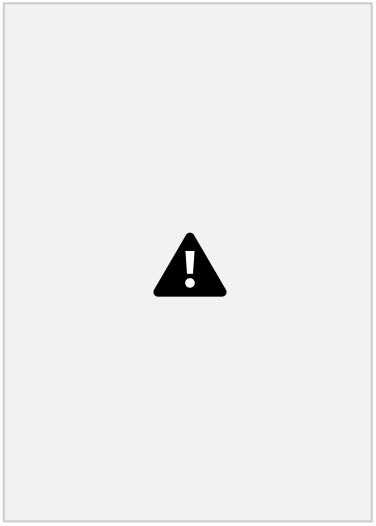


Fig. 33 Payload's Nozzle Thrust test and Pressurized Tank Duration test physical setup

After the completion of this test, the payload subteam decided that the current method for roll control would not be viable. The nozzles did not create enough thrust to efficiently control the CubeSat's rotation and the pressurized tank did not have a large enough volume to allow for roll control for the entire descent. Also, it was highly unlikely that the battery to power the solenoids, and the arduino and its power supply would be able to fit into the CubeSat with the lack of space that remained after integrating all the mission systems components and tubing into it. It was decided that a new roll control method would have to be designed in order for rotation stabilization of the cubesat to be successful while also allowing space for a scientific experiment in 1U of the CubeSat in the future. The details of the new payload roll control method are discussed in the following section. Because an entirely new payload design was required, and still in progress, tests three through six have not been completed yet, but will be performed before IREC.

Payload Testing Documentation

Keep in mind we are performing these tests (in Clemson, SC) at ~725' above sea-level (99KPa), but the ejection will take place (near Las Cruces, NM) at ~6,600' (80KPa)

TEST 1: Nozzle Thrust

What: Nozzle Thrust

Why: Ensure the most optimal nozzle shape is used to guarantee Payload's limited supply of pressurized gas is utilized as efficiently as possible during descent

How: We will 3D print a tapered nozzle design, mount the nozzle to a small load cell or kitchen scale configuration, and use the data collected from the load cell to determine nozzle thrust force and volumetric flow rate **Step by step**:

1. Design tapered nozzle
2. 3D print the nozzle prototype (SLA preferred)
3. Set up the cold gas thruster system
4. Mount a nozzle to the strain gauge load cell assembly
5. Strain gauge fixed to the 3D printed test stand
6. Nozzle mounting bracket fixed to the free end of the strain gauge
7. Strain gauge wiring connected to the Arduino
8. Attach the nozzle to the tubing exiting the solenoid
9. Send pressurized air through the nozzle for 2 seconds
10. Record Data
11. Analyze the data and determine different relevant parameters of the tapered nozzle

Verification Criteria: Nozzle thrust force is in line with what was theoretically calculated for the coupled moment needed to adjust CubeSat roll

TEST 2: Pressurized Tank Duration

What: Pressurized Tank Duration

Why: Determine a realistic time frame for which the cold gas thruster system can operate based on volume of pressurized air (at 3000 psi) in the pressurized tank

How: The pressurized tank will be set up to the cold gas thruster system and a load cell will be connected to a singular nozzle. The solenoids will open, and the pressurized tank will expel pressurized air until empty **Step by step**:

1. Setup the cold gas thruster system
2. Pressurized tank
3. Ball valve
4. Pressure regulator
5. Both solenoids
6. Four nozzles
7. Load cell connected to one of the nozzles
8. Open both solenoids
9. Open the ball valve completely, allowing the flow of pressurized air, and start a timer simultaneously
10. Begin reading load cell data
11. Stop the timer when the nozzle is no longer producing optimal thrust
12. Review the data to determine the optimal run time of the pressurized tank

Verification Criteria: The pressurized tank successfully burned for 90 seconds (estimated descent time) *TEST 3: Swivel Load Test*

What: Swivel Load Test

Why: Ensure the swivel eyebolt assembly can withstand the snatch force of the parachute at payload ejection (mounting bolts and shaft retaining rings). Potential failures include: shear between eyebolt and shaft, retaining ring failure, mounting bolt failure, and assembly bolt failure.

How: Perform theoretical calculations to find the snatch force the swivel will experience upon payload deployment This is done using the snatch force equation: 12V2CdAm

Where ρ is the density of air at the altitude ASL, V is the velocity at deployment (terminal velocity under drogue), C_d is the coefficient of drag of the parachute, A_m is the area of the parachute, and F_s is the factor of safety. This equation was found from wiki.rit.edu.

Snatch Force equation value: 193.69 lbf
Using the snatch force equation value, the following testing process was developed:
Step by step:
Design the testing assembly (utilize a winch or load cell, if available)
Procure / 3D print all necessary components for testing of the swivel
Construct the testing assembly
Perform the test multiple times to ensure the swivel can withstand the theoretically calculated snatch force (193.69 lb f)
Also perform an FEA analysis for further verification
Verification Criteria: Potential failures listed in the "Why" section do not occur during testing *TEST 4: Temperature Test*
What: Temperature Test

Why: Ensure all components of the payload can withstand up to 155°F to replicate the potential launch day environment. Will run the CG system to ensure the high temperature from sitting on the launch pad in the desert does not disrupt the system's performance. Also need to verify the avionics and the avionics' power supply is not impacted by the high temperature to the point of poor (or no) performance. Due to time constraints, only the avionics, GPS and power supply will be put through this test prior to the test launch on April 2. The complete payload, once finished, will go through the test after the test launch and prior to SAC.
How: Activate all components and place into an oven until temperatures of 155 degrees Fahrenheit are reached. Ensure the GPS and Microcontroller/sensors still work and determine the life of the battery. Compare to battery life and sensor/microcontroller performance at normal temperatures
Step by step:

1. Preheat oven to 150-175°F
2. Activate avionics (Arduino, barometric sensor, IMU, GPS, and power supplies) such that voltage from batteries is being drawn
3. Once the oven is preheated, place the avionics into the oven on an aluminum plate to replicate the CubeSat housing wall it will be on
4. Once the goal temperature is reached, remove the avionics and aluminum plate from the oven 5. Read data collected by the microcontroller at high temperatures (barometric and IMU data), measure voltage output from the batteries, and read data from the GPS

Verification Criteria: The microcontroller successfully reads accurate data from the barometric sensor and IMU up to 155°F. The GPS still functions up to 155°F. The batteries still output the required voltage and have enough life to last the duration of the wait on the launch pad, the rocket flight, and the payload descent. *TEST 5: Avionics Test (Roll Control)*
What: Avionics Test (Roll Control)
Why: To ensure the roll control system of the payload is able to counter the terminal angular velocity that the CubeSat and nose cone assembly could experience after deployment and can subsequently counter all minor rotations imparted on the assembly to ensure it does not rotate about its roll axis during descent
How: Hanging the payload nose cone assembly by its swivel from a string will mimic its orientation during descent under its parachute
Step by step:

1. Power on the microcontroller and the roll control system power supply
2. Hang the payload and nose cone assembly by a string
3. Power on the digital laser tachometer
4. Spin the assembly up to the terminal angular velocity (shown by the digital laser tachometer) that was calculated beforehand
5. Engage the roll control system by altering the state of the microcontroller to detumble the assembly
6. After detumbling occurs, poke the assembly multiple times to create small rotations to be countered
7. Repeat steps 4-6 five times

Verification Criteria: The payload and nose cone assembly is able to successfully detumble in a timely manner and, further, can keep its same orientation while experiencing smaller forces causing rotation about its axis. *TEST 6: Avionics Test (Roll Control)*

What: Avionics Test (Ejection Recognition)
Why: Ensure the microcontroller can use the barometric sensor to determine if the payload has ejected from the rocket and is falling under steady descent. Potential failures include: pre-ejection payload activation (payload still in rocket), activation of roll control system before parachute has fully deployed, no activation of the roll control system for the entirety of the descent.
How: Take the avionics of the payload into a 2 passenger plane (with a member of the Clemson Flying Club) that will fly slightly above the altitude ASL where deployment will occur at IREC and then descend past the ejection point. Reading data from the microcontroller, ensure the avionics acts appropriately.
Step by step:

1. Enter the 2 seater airplane with a microcontroller that is powered and on
2. Connect the microcontroller to a laptop and begin reading data
3. Fly to 7000' ASL
4. Descend to 6000' ASL
5. Check that the microcontroller has recognized the ejection altitude of 6600' ASL and is sending outputs to drive the roll control system
6. Repeat steps 2-5 three times

Verification Criteria: The microcontroller successfully recognizes the altitude where ejection will take place at IREC and switches states to drive the roll control system.

Appendix C. Hazard Analysis

INTENTIONALLY LEFT BLANK: DOES NOT APPLY

Appendix D. Risk Assessment

The risk assessment matrix shown in Figure 34 is used to evaluate the various risks listed in Table 14 to determine the severity of the risk in order to determine the level of mitigation required.

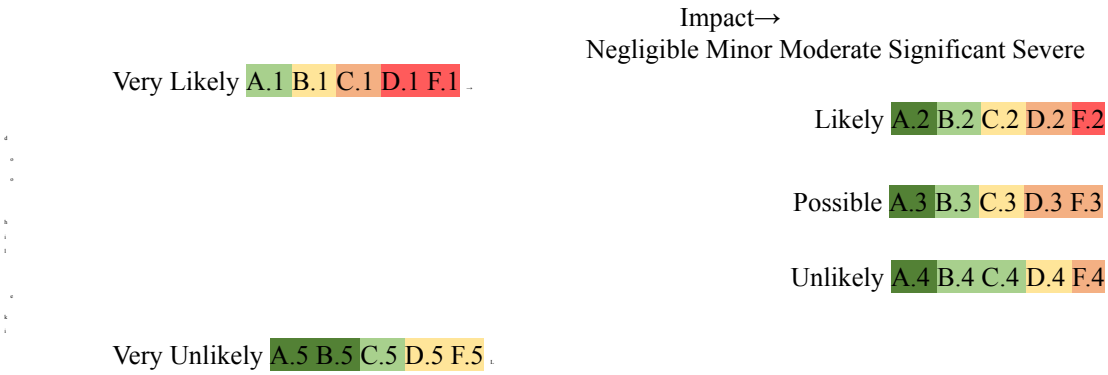


Fig. 34 Risk Assessment Matrix

Table 13 is used to further understand the evaluation of each individual risk in Table 14.

Table 13 Risk Assessment Matrix Reference Table

No.	Color Description
1	Risk is not of concern, and therefore will not need to be addressed.
2	Risk is of minimal concern, will be addressed if simple solution is found.
3	Risk is of concern; a safeguard will be put in place to mitigate the risk
4	Risk is considerable, either a safeguard will be put in place or the system will undergo a redesign to mitigate the risk.
5	Risk is serious, the project can not move forward if the system does not undergo a complete redesign.

Table 14 lists all risks associated with Oculus I that could inflict harm upon an individual in attendance at the Spaceport America Cup and evaluates how the mitigation approach will lessen the risk either in likelihood or impact.

Table 14 Risks Associated with Oculus I

Hazard	Possible Causes	Likelihood/Impact	Mitigation	Likelihood/Impact Post Mitigation
Rocket Falls off launch rail.	Rail button tear out	C.5	Have the launch ops team wear hard hats and safety glasses.	B.5
	Rail collapse	D.5	Have the launch ops team wear hard hats and safety glasses.	C.5
Recovery deploys during prelaunch operations.	Accelerometer Failure	C.4	Test the accelerometer readings prior to installing the avionics bay into the rocket.	C.5
	Software Bug	C.3	Test the avionics software extensively through pressurized ejection testing.	C.5
Motor explodes	Inconsistencies in propellant grains.	F.4	Visually inspect propellant grains prior to installing into the motor. Visually inspect nozzle	C.5

	Clogged Nozzle.	D.3	C.5
when ignited.	prior to motor being moved to vertical on the pad.		
	failure.	F.4	Visually inspect motor

Rocket strays from

Misaligned

Motor casing

Fins. C.3

casing for manufacturing imperfections.

During fabrication, a precise 3D printed jig is used to ensure fin alignment.

D.5

C.5

rocket off course. D.3 Use stability analysis to determine maximum allowable wind for flight.

projected course.

Wind gusts push B.5

Fin tear out. C.2 Fin flutter analysis and fin deflection testing. C.5

Dead Battery

C.3

Check battery voltages

prior to installing avionics. C.5

Recovery Fails to Deploy.

Bad Accelerometer Data C.3 Verify accelerometer readings prior to installing avionics. C.5

Bad Barometric Data C.3 Verify barometer readings prior to installing avionics. C.5

Appendix E. Assembly, Preflight & Launch Checklists

Table E. 1 Assembly Checklist

Number Item Initial 1 Wire ejection charges to safe avionics systems.
2 Load main and guide chutes.
3 Load payload main.
4 Connect payload tube to avionics tube with shear pins.
5 Slide motor into bottom motor tube.
6 Bolt in motor bulkheads and thrust plate to secure the motor inside the tube. 7 Load drogue.
8 Slide avionics bay into the avionics tube.
9 Arm all systems.
10 Connect the avionics tube to motor tube with shear pins.
11 Secure igniters to the grain.

Table E. 2 Payload Preflight Checklist

Number Item Initial 1 Assemble the Front-Back1 wall, Side1 wall, and the Bottom plate together using 8X ⅜” 4-40 bolts
2 Bolt the Payload bulkhead to the Bottom plate using 4X ½” 6-32 bolts 3 Bolt the arduino uno to Front-Back1 using 4X ⅜” 4-40 bolts
4 Place the Featherweight GPS tracker into its housing and bolt it to the Front-Back1 wall
5 Place the 6S LiPo into its 3D printed housing and bolt the housing to the Front-Back2 wall
6 Secure the three reaction wheels to the three motors by threading their tapped center hole onto the threaded motor shaft and tightening a locknut down on top of them 7 Secure one motor-reaction wheel assembly to the Mid2 plate, one to the Side1 wall, and one to the Front-Back2 wall; all using 4X 10mm M3 bolts
8 Secure the Front-Back2 wall to the Side1 wall using 4X ⅜” 4-40 bolts 9 Secure the Mid2 plate to the Side1 wall and Front-Back1 wall using 4X ⅜” 4-40 bolts 10 Ensure the ESC is in the bottom module of the CubeSat and the motor wires are fed through their respective slots before bolting
11 Secure the Mid3 plate to the Side1 wall and Front-Back1 wall using 4X ⅜” 4-40 bolts 12 This plate should separate the middle and top modules of the CubeSat 13 Bolt the ballast onto

the top of the Mid3 plate using 4X ½” 6-32 bolts 14 Mount the ESC to the Side1 wall
15 All motors wires should already be soldered to the ESC before assembly 16 Connect all Avionics together - ESC and 6S LiPo to microcontroller assembly 17 Turn the microcontroller on and into standby mode
18 Bolt Side2 to the Front-Back1 wall, Mid2 plate, and Mid3 plate using 8X ⅜” 4-40 bolts
19 Bolt the Front-Back2 wall to the Side1 and Side2 walls using 8X ⅜” 4-40 bolts 20 Mount the swivel to the Top plate using 4X ⅜” 6-32 CS bolts
21 The swivel should already be pre-assembled and ready for mounting
22 Bolt the Top plate into the Front-Back1 and Front-Back2 walls
23 Turn the Featherweight GPS on
24 Payload Assembly is complete and it is ready to be integrated into the rocket

Table E. 3 Avionics Preflight Checklist

Number Item Initial 1 All components are secured tightly with their specific sockets, and placements 2 Test Software to confirm if there are no compile errors or reading in wrong data. 3 Retest the second bullet
4 Turn on recording cameras and confirm they are recording
5 Clean off debris from Avionics Bay
6 Arm the Avionics system with Screw Key Switch
7 Hand the bay to the Assembly Team

Table E. 4 Launch Checklist

Number Item Initial 1 Initiate a test, read sensor values, and provide a confirmation of reasonable readings via the ground control interface
2 Send the arm command to the main computer
3 Send the ignite state signal to the main computer
4 Start the motor within 60 seconds of giving the ignite state signal
5 Continue monitoring readings from ground control through the recovery phase 6 Use the GPS and radio signal to recover the rocket
7 Turn off the Jeti switch

Appendix F. Engineering Drawings

Appendix F contains all major technical drawings for the Oculus I airframe. Drawing 1/11 labels said major bulkheads and tubes, the corresponding part drawings are found throughout Appendix A.