Genesis Guided Rocket Senior Project

Prepared For

Professor Andreas “Baron”Wesemann

Prepared By

Kimball Goss

Submitted on April 22, 2024

Senior Flight Project: Aviation Management

TABLE OF CONTENTS

TABLE OF CONTENTS **2**
1.0 ABSTRACT **3**
2.0 INTRODUCTION AND BACKGROUND **4**
3.0 REVIEW PLAN **5**
4.0 SCHEDULE **7**
5.0 COSTS **8**
6.0 REPORT **9**
6.1 A Few Notes 9
6.2 Documentation 9
6.3 Mini Projects 9
6.3.1 Mini Project 1: Calibration 10
6.3.2 Mini Project 2: Determining Rotational Speed 11
6.3.3 Mini Project 3: Determining Orientation 11
6.3.4 Mini Project 4: PID Self-Leveling (Single axis) 11
6.3.5 Mini Project 5: SD Datalogging 12
6.3.6 Mini Project 6: Circuit Design 12
6.4 Rocket Design 12
6.4.1 Flight Profile 13
6.4.2 TVC Mount 13
6.5 Flight Computer 14
6.6 Testing 15
6.6 Launch 16
6.6 Post-Launch Analysis 17
7.0 CONCLUSIONS **25**
REFERENCES **26**
APPENDIX A: GENESIS LAUNCH COMPUTER SCHEMATIC **27**
APPENDIX B: GENESIS FLIGHT COMPUTER SCHEMATIC **28**
APPENDIX C: TIME LOG **29**
APPENDIX D: COSTS **31**

1.0 ABSTRACT

The Genesis Guided Rocket is a project done solely for the learning opportunity. There is little to no real-world application for this rocket system. This document starts at the inception of the project, and walks the reader through the creation and launch of the first version dubbed Genesis Mark I. This document is not a set of instructions or a tutorial, but more a record and explanation of the different decisions and systems of the Genesis Mark I project.

Keywords: *Thrust Vector Control, Prototype, Rocket, PID Loop, Senior Project, Scratch-build*

2.0 INTRODUCTION AND BACKGROUND

The purpose of this project is to serve as a learning platform and a resume-differentiator. The main goal being to gain a deeper understanding of rocket stability, flight control, programming, control system design, mechanical design, and multi-discipline project management. While there are plenty of already-existing boards, kits, and programs to execute a flight profile similar to this, they would not teach these subjects in as extreme depth as building it from scratch.

The idea for this project is heavily inspired by the “Scout” rockets made by Joe Barnard of Barnard Propulsion Systems (BPS). The goal of the BPS program was to make a model rocket that landed under its own power and control, very similar to the SpaceX Falcon 9 booster. This endeavor started in 2015, he succeeded in landing his first rocket in 2022. I followed his journey through YouTube for those seven years. When it came time to choose a senior flight project in late 2023, I knew I wanted to do something similar.

It took Joe 7 years to land his model rocket. I had 4 months. For this reason I chose not to take on the full ascent, motor swap, descent, and landing as it seemed too ambitious for the available timescale. Instead, I chose to take on only the first stage, ascent. Even this proved difficult with the 4 month timeframe. Joe has done a good job documenting his findings and lessons learned through YouTube, he has also built a community around BPS that is passionate about “smart” rocketry. These were resources I leaned on heavily to achieve my goal of launching a custom built guided rocket.

3.0 REVIEW PLAN

As mentioned earlier, this project is not meant to have any real world application or use case, but it is solely a project meant for me to explore all of the UAS subjects and technologies I wanted to dive deeper with. Because it is a learning opportunity, I wanted to make it easier for others to follow in my footsteps so I also tasked myself with documenting and publishing my files and results as a secondary goal.

The project had to be broken down into accomplishable sub-goals and segments due to its multidisciplinary nature. It was determined that the following skills were going to be crucial in accomplishing this project:
Mechanical design, CAD design, circuit design, circuit assembly, programming, control system design.

I transferred to Aviation Management from a Mechanical Engineering program. I've also been CAD designing and 3D printing as a hobby for years, so I already felt very confident in my mechanical design skills and my CAD skills. This meant that I would have to really work on my weak points which were circuit design, circuit assembly, programming, and control system design. The best way to do this was to make "mini-projects" that helped me learn these skills in a quick, simplified way. Once I was confident with the basics, I could then apply these newly learned skills toward the Genesis Rocket.

These essential skills are listed below, along with some details on why they were important and how it applied to the Genesis Rocket

- **Interpreting data from Sensors:** Off-the-shelf components only spit out raw data. Connecting the sensors correctly, requesting the right data, and then filtering and scaling that data correctly was vital to a successful flight.
- **Single Axis PID Auto Leveler:** Once the sensor data is reliable, the next challenge is to plug that data into a control algorithm. Using sensor data to drive actuators and measure the resulting changes is another vital process in guiding the rocket.
- **SD Card Datalogger**: This can aid in debugging and documentation. Being able to record data allows you to see and understand the systems behavior after the flight.
- **Custom Launch Computer:** The launch computer is a custom designed printed circuit board (PCB) that builds skills in the circuit design area. The launch computer is very simple, but the design strategies and tools like Autocad EAGLE were essential for designing the much more complex flight computer.

I already owned much of the vital equipment needed for this project, but I did have to acquire some extra tools and supplies. However, for transparency and the utility of other potential rocketeers, I've listed the tools needed below.

- A computer with a CAD software, 3D printer slicer program, Arduino IDE, and internet connection.
- A 3D Printer with at least a 7"x7"x7" print volume
- A soldering station
- A standard set of tools (Screwdrivers, drill, drill bits, pliers, wire cutters, etc)
- LiPo Battery Charger

I've also made a list of supplies below. Most of these items can be easily found on Amazon.com, the breakout boards can also be found on Adafruit.com, the rocket specialty supplies can be found on ApogeeRockets.com,

- Electronic Components
    - Assorted 5mm terminal blocks
    - Assorted resistors
    - Assorted electrolytic capacitors
    - Assorted 3mm LEDs
    - Assorted Buttons
    - 2x L7805CV 5V Voltage Regulators
    - 2x 30A 60V N-Channel Power Mosfets
    - 1x 4 Digit 7 Segment Digital Tube LED Display
    - 1x DaierTek Safety Cover Toggle Rocker Switch
    - 2x 30A 60V N-Channel Power Mosfets
    - 3x 9 gram Servos
    - 1x GY521 MPU 6050 IMU breakout board
    - 1x Adafruit MicroSD breakout board+
    - 1x Adafruit SPI SD flash chip breakout board
    - 1x Adafruit BMP280 breakout board (Stemma QT variant)
    - 1x Arduino Uno microcontroller
    - 1x Teensy 4.0 microcontroller
    - Male and Female battery connectors (to match your chosen LiPo batteries)
    - Spare wire (22 gauge)
    - Solder
- 3D Printer filament
- 3" LOC Body Tubes (2.92"ID, 3.02"OD, 18" Length)
- Aerotech RMS 24/40 Reloadable Motor System kit
- Aerotech F12-3J(2) 3 pack motor reloads
- "Firewire" Electric Matches (found at csrocketry.com)
- 2S 250mah-500mah LiPo battery
- Assorted 3mm screws of varying lengths.

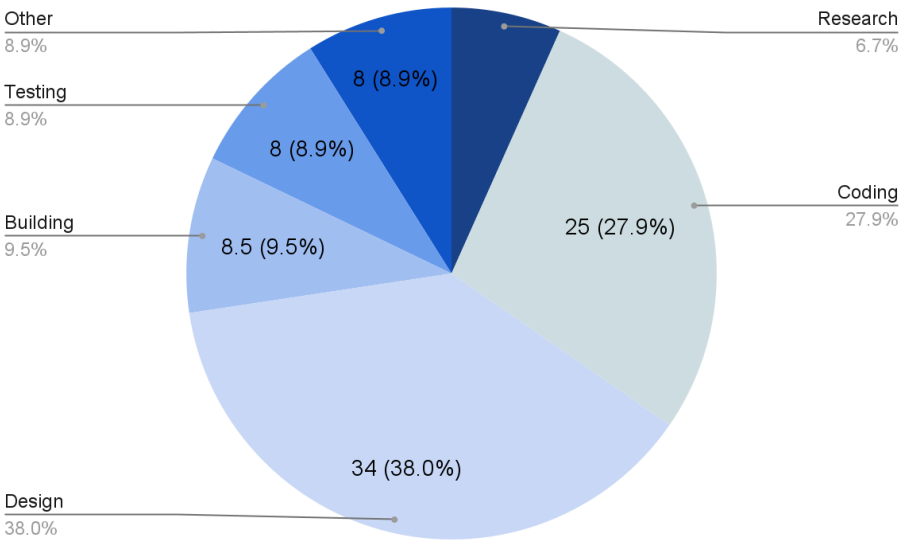
4.0 SCHEDULE

Even though a simpler “ascent only” flight profile was chosen, the schedule was still very difficult to maintain. The schedule goal and schedule execution rarely cooperated. However, if nothing else the initial schedule at least provided a roadmap of the project. There were often many things happening at once, but here is a highly simplified version of the schedule as it actually occurred:

01/01/24-01/11/24: Planning
01/12/24-02/01/24: Research, mini-projects, and preliminary design
02/02/24-03/15/24: Final design and preliminary coding
03/16/24-04/12/24: Final coding, rocket assembly, and testing
04/20/24: Launch

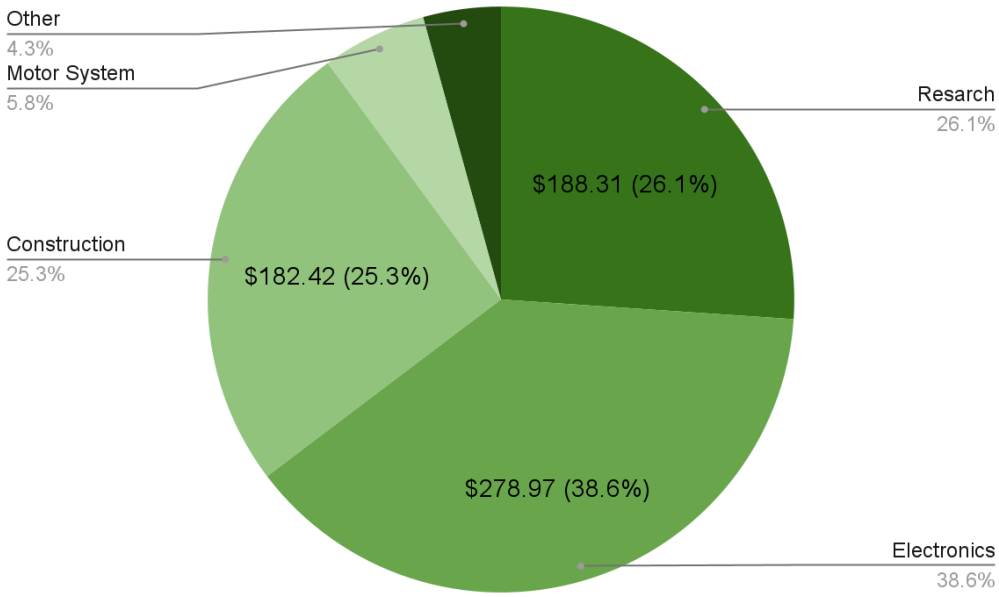
If this project were to be done again, the general order of events would remain the same. I did get slightly behind schedule in the second half of March and the beginning of April due to an increased workload from other classes. With this hindsight, if another student were to attempt this in a similar timeframe, I would suggest getting to the testing stage as fast as possible to avoid long nights and last minute runs to the hardware store. The “mini-project” architecture proved to be extremely helpful, as it was the perfect opportunity to learn new skills and allow for failure, without damaging or destroying any vital flight hardware.

As of 4/20/24 I have logged 97 hours in this project.



5.0 COSTS

The bill as of April 20, 2024 is $837.39
As is typically the case with experimental prototypes, the costs tended to keep inflating as the project went on. Even if there were no academic pressure to meet the launch date, the “sunk-cost” fallacy is easy to fall into with a project like this so I caution anyone who funds their own projects to do so with caution, good planning, and fiscal responsibility.
When the project was first conceived I wanted to aim for $500, because I wanted this project to also be accessible to other students, so it had to rely on cheap components and supplies. But I knew budgets rarely go according to plan, so in the event of budget creep I wanted to stay under $1000. Considering the current spending data I consider that goal to be a success. Below is a pie chart showing the costs of different categories of the project.



It’s important to note that these costs are not the true cost of materials for one rocket. This cost should be seen as an amalgamation of setup, tools, research, replacement parts, and finally materials. There were many times I had to buy a component two, three, even four times because I would damage it trying to learn how to use it. This most often happened with electronic components, hence the high electronics cost. Research is another area where significant money could be saved. I purchased multiple textbooks on control system design and digital signal processing. While these were helpful, I would not say they are absolutely necessary.
If I were to help another student build an identical rocket, allowing them to utilize my tools, knowledge, designs, and code, I estimate it would cost them roughly $200. Additionally, many of the electronics and parts kits come with more components than you need for just one rocket. I estimate that with that $200, you could actually build 3 or 4 identical rockets with the amount of supplies, dropping the cost per rocket to about $50 or even less depending on what supplies you buy.

6.0 REPORT

6.1 A Few Notes

This report body is not given in chronological order, but rather the order of design and execution. It's important to know this, as some details might be unclear, and some choices may seem illogical when taken out of the chronological order in which they occurred.
This report is not meant to act as a tutorial or guide on how to build this rocket, it is meant to dive into the decision making and processes behind the project.

6.2 Documentation

With the goal of being accessible, repeatable and transparent a good method of documentation was required. Originally the idea of a video diary series made a lot of sense, this was again inspired by Joe Barnard and his YouTube series. However, this goal proved to be difficult even in the beginning stages of the project. I simply did not have the resources, tools or time to work on the project, *and* document the whole thing with high quality edited video. The project was very sporadic as I was always moving from 3D design, to electronics research, to construction, etc. A video series would not have fit the project because it would have seemed very unorganized.
Early on I decided to use a simpler documentation technique that the software community pioneered. GitHub.com is geared toward software development. It allows users to upload their software files via a "commit" action. Later, when the user makes changes to the software on their local computer they then make another commit which updates the online repository to the latest version. You can also add notes to your commit actions so other users can see when and why you updated something. Each repository also shows a history of all "commits" along with their notes. This was the perfect tool for this project. The repository for this project is at the link below:

https://github.com/AlpineAce27/Senior-Project-TVC-Guidance

This is the primary tool for the documentation of this project. The repository includes photos, schematics, PCB design files, code, 3D files, and more all for free. Anyone can access this repository at any time with a full history of updates. Anytime this report refers to "the respiratory" it is referencing this tool.

6.3 Mini Projects

The first mini project to be tackled was the gathering of data and calibration of the inertial measurement unit (IMU). The IMU is responsible for measuring the orientation of the rocket. It does this with accelerometers which measure acceleration of the rocket in all three axes, and gyroscopes which measure the rotational speed. The unit used here is an MPU6050 chip mounted on a GY-521 breakout board.
When hooked up and set up correctly, the sensor spits out raw numbers for the 6 variables. (acceleration X, acceleration Y, acceleration Z, gyro X, gyro Y, gyro Z)
For example, instead of spitting out "45 degrees per second" the unit just will just spit out the number "2,988". Each axis has a unique starting point, as well as scale factor. For example the x gyroscope could have a starting value of 127 and a scale factor of 65.5 which means when it's not rotating at all (0 degrees

per second) the x gyro would read 127. Then if the unit rotates at 1 degree per second it would read 192.5 (127 + 65.5).

All three accelerometers share a scale factor, and all three gyroscopes share a scale factor. These scale factors can actually be changed, depending on how sensitive you want the unit to be. These scale factors had to be found in the datasheet (on page 12) for the MPU6050, which is in the repository under "Miniprojects/MPU 6050 Sensor/Documentation".

6.3.1 Mini Project 1: Calibration

The starting points have to be calculated every time you power the unit up. Finding these offset values is what the "calibration" code does. It does this by taking thousands of measurements from the X gyroscope, and then averaging them. If the unit is not moving, the gyroscope will settle on one value. This average is the offset value the rest of the code will compensate for when pulling data from the X gyroscope. Then it

will repeat this calibration process for the Y gyroscope, and then finally Z gyroscope. This process is done in a few seconds. The code can be found in the repository under “Miniprojects/MPU 6050 Sensor/Code/MPU_6050_01_Calibration”.
Calibration is not done for the accelerometers.

6.3.2 Mini Project 2: Determining Rotational Speed

After calibration and applying the scale factors, you can turn the data into usable measurements of “deg/sec” and “ m/s^s” by applying the correct scale factors and repeated testing. You can then graph this motion in real time. This code is in the repository under “Miniprojects/MPU 6050 Sensor/Code/MPU_6050_02_Display_X_DegreesPerSecond”.

6.3.3 Mini Project 3: Determining Orientation

Finally, To determine the current orientation of the rocket, you then have to integrate the gyroscope readings to go from deg/sec to just plain old degrees. This method is subject to drift, which means the measured orientation may slowly drift away from the true orientation. This drift is exaggerated by vibration. However, in this use case the rocket will only fly for a few seconds and the rocket motor doesn’t create much vibration, so this technique will work just fine. The orientation can also be graphed in real time. This code can be found in the repository under “Miniprojects/MPU 6050 Sensor/Code/MPU_6050_02_Display_X_Orientation”.

6.3.4 Mini Project 4: PID Self-Leveling (Single axis)

Once you can reliably measure the orientation, you can feed it into a control algorithm. Control algorithms are complicated, so this report will not explain them in depth. However, I will provide a brief example on how the control algorithm works on Genesis.
First we assign a “desired value” which is our goal, and a P-gain value. For orientation this desired value is 0 degrees because we want the rocket to point straight up. We can arbitrarily set the P-gain to 0.6 here. We then measure the current orientation of the rocket using the IMU, in this example we can say the measurement comes out to 10 degrees. The control algorithm takes the measured value and subtracts the desired value from it to obtain an error value. In this case the error value is -10. This represents how far away we are from our goal of 0 degrees. It then multiplies this error with a P-gain (0.6) to get -6, and then sends that signal to the actuators which move the rocket -6 degrees. Now when the next loop begins, the new measurement is only 4 degrees. And just like that we have gotten closer to our desired orientation! Through this process we’ve created a math equation to steadily move toward our goal of 0 degrees. This is a Proportional Controller, because the output is directly proportional to the error. There is also a Derivative Controller, which takes into account how fast we are approaching the goal. For example, if our measured orientation is 3 degrees but we are moving at -10 degrees every second, we may want to slow down the input so we don't overshoot our goal. There is also an Integral controller which takes into account how much error we’ve built up over time. Together these make the standard PID control algorithm. This control loop algorithm along with STL files to 3D print a servo driven self leveler, can be found in the repository under “Miniprojects/MPU 6050 Sensor/Code/Self Levelling PID (Single Axis)” Unfortunately, on the rocket itself I was running into problems with actuator saturation. This can be solved with more time and effort, but with a tight schedule I elected to completely remove the Integral

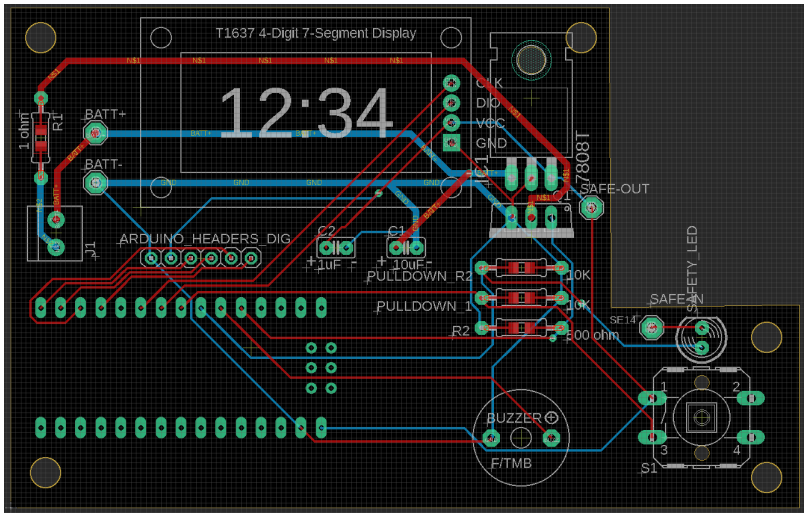
Controller from the Genesis system which negated the saturation issue. This is why the Genesis uses a "PD" control algorithm instead.

6.3.5 Mini Project 5: SD Datalogging

Having the flight computer communicate with an SD card, create files, write to those files, read from those files, and delete those files was vital to gathering any kind of data for a flight. The SD Card data logging code can be found in the repository under "Miniprojects/SD Data Logging/simple_data_logger". While this mini project was still useful in learning a new skill, I later found out it could not run effectively on the rocket. Unfortunately, the current SD card code library is very slow. The control loop on Genesis runs at 50hz which means it runs once every 20 milliseconds. The SD library was taking too long to open the file, write to the file, and then close the file. This caused the timing of the loops to exceed 20 milliseconds which was impacting the control. With more time and effort this can be fixed, but with limited time it was determined to remove data logging from the requirements for the first flight.

6.3.6 Mini Project 6: Circuit Design

I had very little experience in circuit design. I relied heavily on episode 4 of Joe Barnards "Landing Model Rocket series" in order to learn how to create a custom circuit in the Autocad program EAGLE. I was worried I would make a mistake, so I decided to design my own launch computer which is responsible for the initial countdown and lighting of the motor. This was to serve as a precursor to the more complex flight computer which had to be more compact and was responsible for the actual guidance of the rocket.



This launch computer design went well, but I did end up making a few mistakes on the first design that caused the first couple prototypes to be inoperable. These were simple wiring issues that just required some adjustments of the design and a reprint. The schematic for this circuit can be found in the appendices section. The schematic, 3D printable STL files and PCB manufacturing files can be found in the repository under "Genesis Launch Computer".

6.4 Rocket Design

The rocket design had 3 main goals:

1. It had to be light enough to be lifted by a F class 24mm rocket motor
2. It had o have enough internal space to fit a custom flight computer and a TVC mount
3. It had to be made from cheap and easily replaceable components for accessibility and easy re-builds

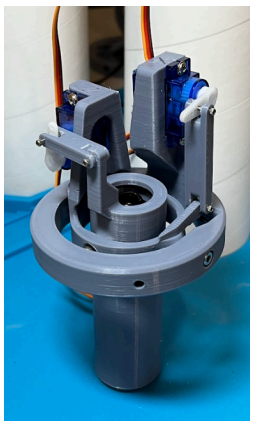
As mentioned earlier the design is based heavily on the Scout rocket series by BPS space. The Scout rockets were designed with 3” body types, a TVC mount, and had an extra weight budget for an additional landing motor (also F class) as well as deployable landing legs. An early scout rocket is shown next to the final CAD model for the Genesis, with the same general scale. The Genesis rocket is about ⅔ the height, and it is significantly lighter.

6.4.1 Flight Profile

The weight budget was the first thing to be worked out. The motor choice has an average thrust of 12 Newtons (2.7lbs). The goal is to have a trust to weight ratio of 1.5, this meant the Genesis rocket would have to weigh around 1.8 lbs. If the TVC mount works correctly, the flight would experience about 0.5G’s of acceleration for about 3 seconds. This would lead to an estimated apogee of 33 meters or 108 feet. At apogee, a black powder charge would be set off by the computer to deploy the parachute recovery system.

6.4.2 TVC Mount

The TVC mount utilized by the Scout rocket has a gimbal range of about +-5 degrees. My TVC has a range of +-10 degrees. This was done with the hope that I would never have to use a full 10 degrees of deflection. If the rocket is at an orientation that requires a 10 degree deflection, it is most likely already “tipping over” mid flight. However, any extra performance I could create would add to the likelihood of success.



The TVC mount is very important because it has to move the rocket motor across two different axes to provide vectored thrust. This vectored thrust would rotate the rocket to keep it pointed skyward. There were a few design requirements the TVC mount had to meet as well.

- It had to utilize the cheap and ubiquitous 9 gram plastic servos
- It also had to be entirely 3D printable
- It had to have a range of +-10 degrees in both axes
- It had to fit inside of a 3” round body tube

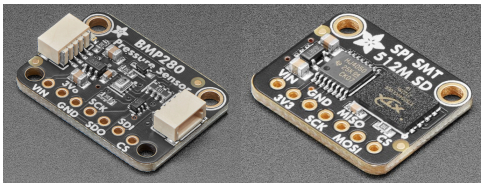
The CAD work for the TVC began on January 20th, and the design was not finalized until the beginning of March. This part went through many iterations, and the requirement to “fit inside a 3” round body tube had to be adjusted to allow for the desired travel. At certain extreme angles the control arms would collide with the inner surface of the cardboard tube. So instead of a complete TVC re-design, I elected to cut holes in the body tube to allow for the full travel of the mount.

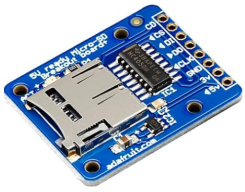
6.5 Flight Computer

The flight computer is based off of the Arduino platform as well. Most arduino microcontrollers operate around 4MHz which is not fast enough to update the TVC mount and interpret data. This was a critical piece of information I learned from the BPS space discord server from multiple users who had previously built their own guidance computers. They saved me a lot of time and headache, because I originally planned to use the arduino Nano for my controller. Per the recommendation of these community members, the Genesis Flight Computer uses a Teensy 4.0 which operates at 600MHz. This is over 15 times faster than the arduino microcontrollers. It is a more expensive board, but the high clock speed is required to have a control algorithm that runs at 50Hz. This means the measurements and actuators should update about 50 times a second. It’s important to know however that the servo’s cannot keep up with this speed of communication, especially if the assigned deflection is extreme.
Just like the GY-521 breakout board mentioned earlier, there are similar breakout boards for other sensors like SD cards and barometers. Here is a list of the breakout boards used on the Genesis Flight Computer. They are also shown in the same order.

- Teensy 4.0
- MPU-6050 GY-521 breakout board
- BMP280 Barometric Pressure & Altitude Sensor - STEMMA QT variant
- SPI Flash SD Card - XTSD 512 MB
- Adafruit Micro SD Card Breakout board+

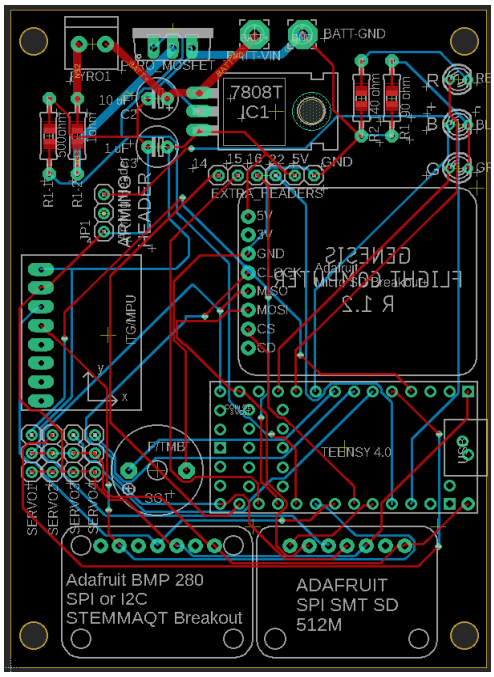






There is an N-channel Mosfet to trigger a pyro channel, this is what ignites the black powder charge to deploy the parachute. There are also some power management components like capacitors and voltage regulators, and lastly some output components like LED’s and a buzzer. With some extra programming these output devices can help a user understand what the computer is doing, and if it gets stuck or has any issues. I also use the output LEDs and buzzer to show what stage of flight the board thinks it's in, whether

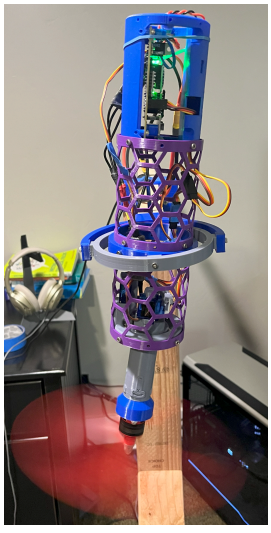
it's waiting for launch, flying, falling, or landing. This design also took two attempts to perfect, as the first design had some wiring issues that simply needed some design adjustments and a reprint. Overall it’s safe to say the precursor launch computer project was a great aid that really gave me the confidence and skill base to design the flight computer.



In the future, this flight computer may become even more compact. The board is rather large because it relies so heavily on breakout boards. As shown earlier, the actual sensors on these breakout boards are very small. If the computer could be designed with the sensors directly mounted on the surface instead of the breakout boards, a great amount of space and weight could be saved.

6.6 Testing

Because so much work was put into the mini projects, very little testing was needed once the rocket was assembled. There were two tests to determine the optimal black powder charge for the parachute deployment; these tests also served to verify the functionality of the launch computer.



For some extra surety, I thought it would be a good idea to hook up a motor and propeller to the TVC mount to simulate the thrust of the motor. Ideally this would help verify the correct behavior of the code and help tune the PID values. After a couple weeks of testing with his method, it was clear the flight computer was attempting to correct the orientation exactly as planned. However, the angle measurements seemed to drift very fast. This meant it would try to stay upright for the first few seconds, but it would quickly “lose its bearing” and proceeded to aim left, right, forward and backward all while thinking it was aiming up. After some trouble shooting it was determined that the motor and propeller were adding a good deal of vibration, which caused the gyroscopes to drift much

faster than expected. After removing the motor and propeller, the rocket began behaving in a much more expected way, and the angle measurements stopped drifting. This is also the stage where I discovered the rocket struggled with actuator saturation, and as a result the “I-gain” value in the code was changed to 0, effectively eliminating the integral portion of the controller.

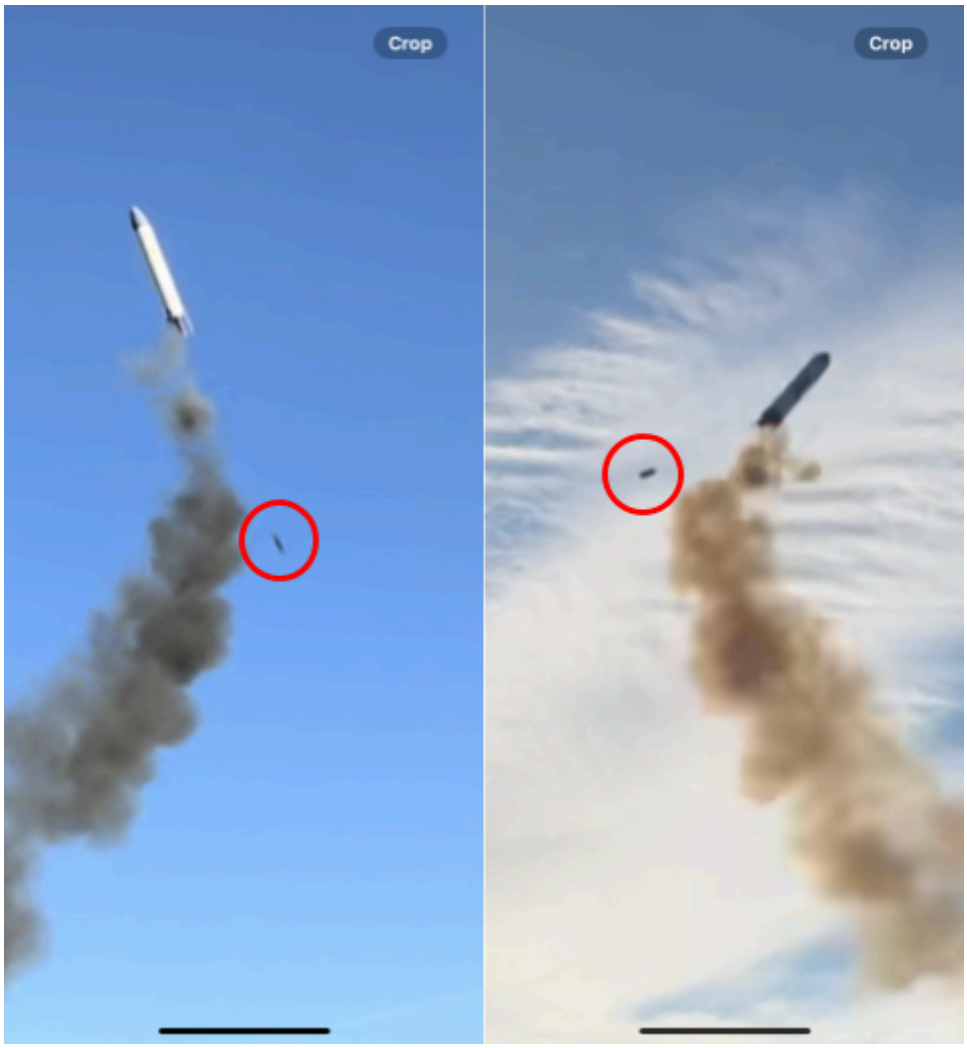
6.6 Launch

Launch occurred on April 20th 2024 at 7:30 am. This time was chosen for good weather and low winds. The launch went very well. Much of the preparation was done the day before, so the morning of the launch all we had to do was load a car with the pre-prepared supplies. We arrived at the field around 7:00 am, and proceeded to set up the launch pad (a sawhorse and some plywood). After the launch pad was assembled, the rocket was integrated. The gyro calibration was thrown off the first time, and the only way to reset the calibration is to power cycle the whole rocket. This means we had to disassemble it and then reassemble it one more time, but this ensured the gyro readings would be accurate. Once the rocket was ready for launch, the rocket was armed. After confirmation from the rocket that it was ready for flight we began the countdown.
At the end of the countdown the rocket motor ignited and the rocket left the pad leaning a bit to the left. In the video you can see the rocket actively correct itself as its orientation changes to straight up. Screenshots from that video are shown below.

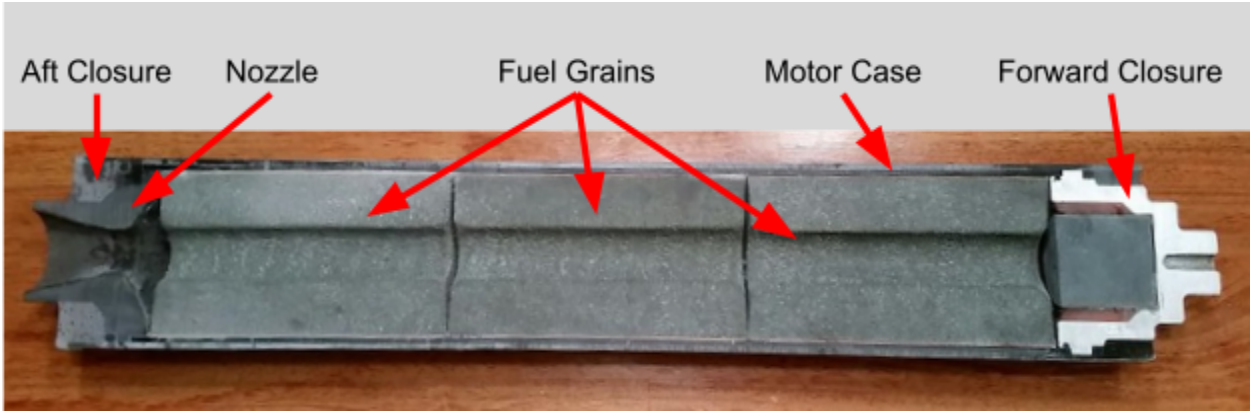
During this turn however, the rocket motor suddenly cut out and the whole assembly then fell to the grass. After stopping on the grass, the rocket detected it had landed and proceeded to go into a waiting state.

6.6 Post-Launch Analysis

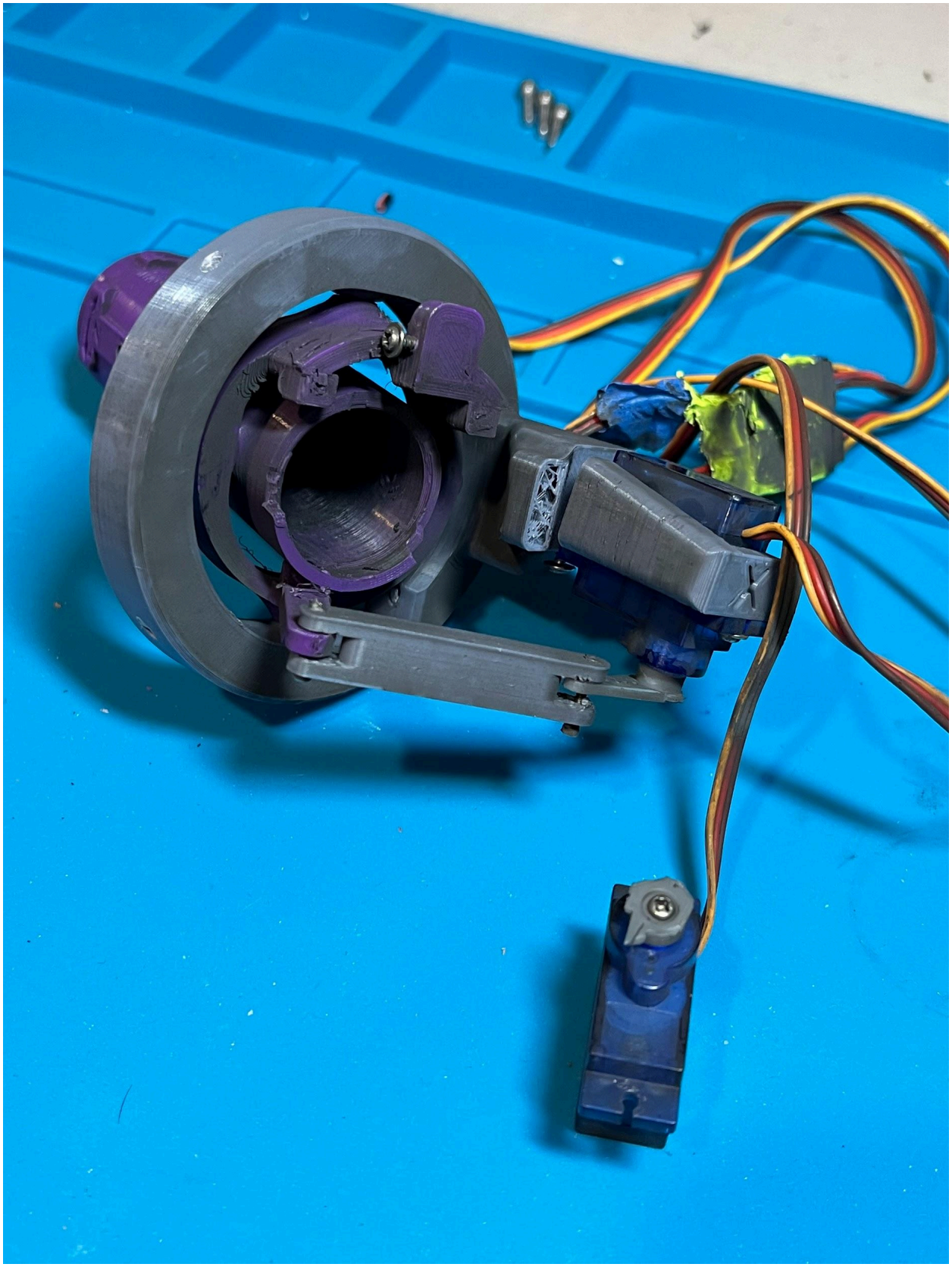
At first it was unclear what caused the motor to cut out. I initially thought the fuel grain had exploded due to air bubbles, as this is something I have seen before in my own rocket fuel formulas. Further investigation from the two camera angles showed something else though. Right as the motor cuts off, you can see something fly out of the motor mount at a very high speed.

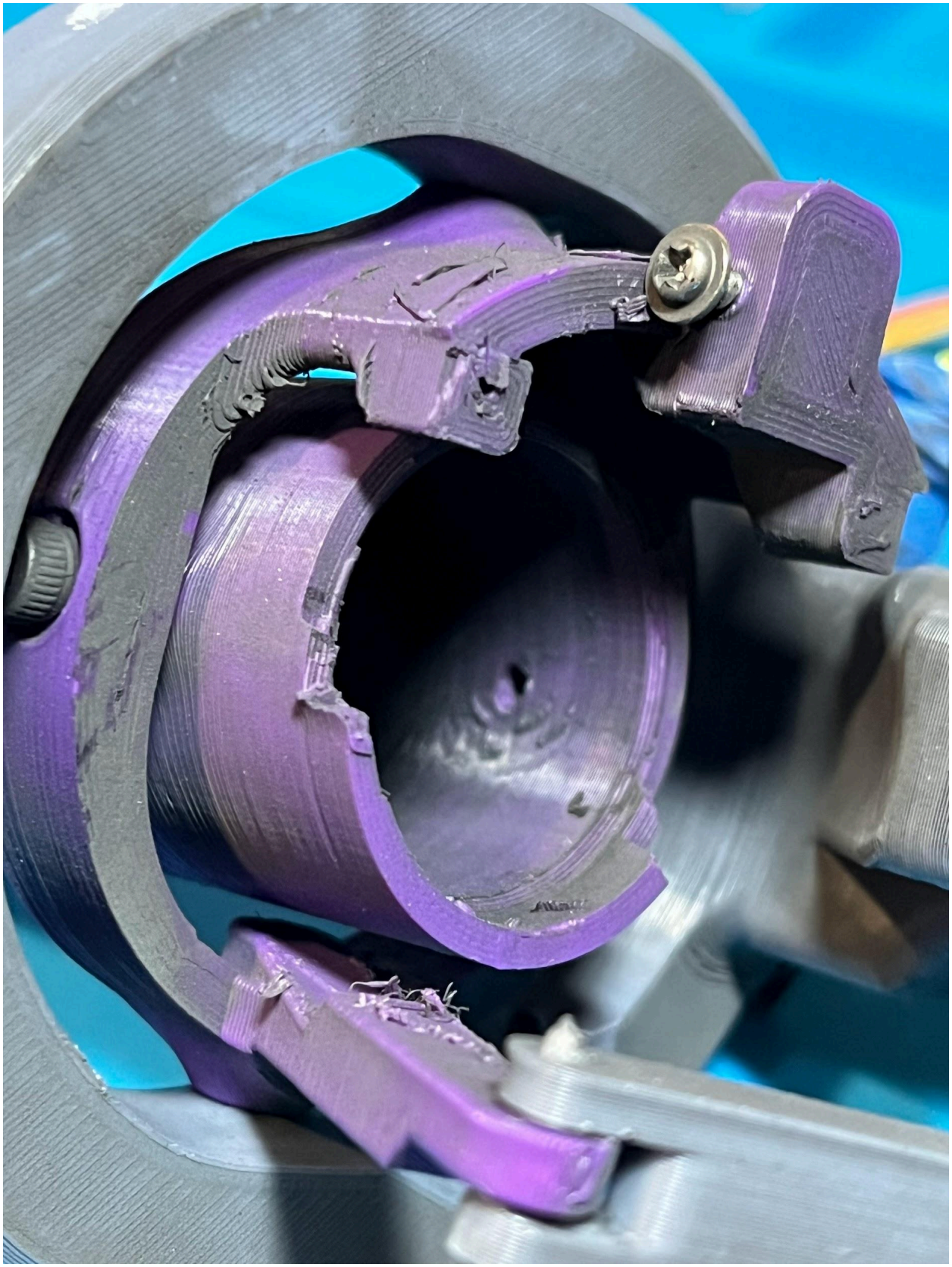


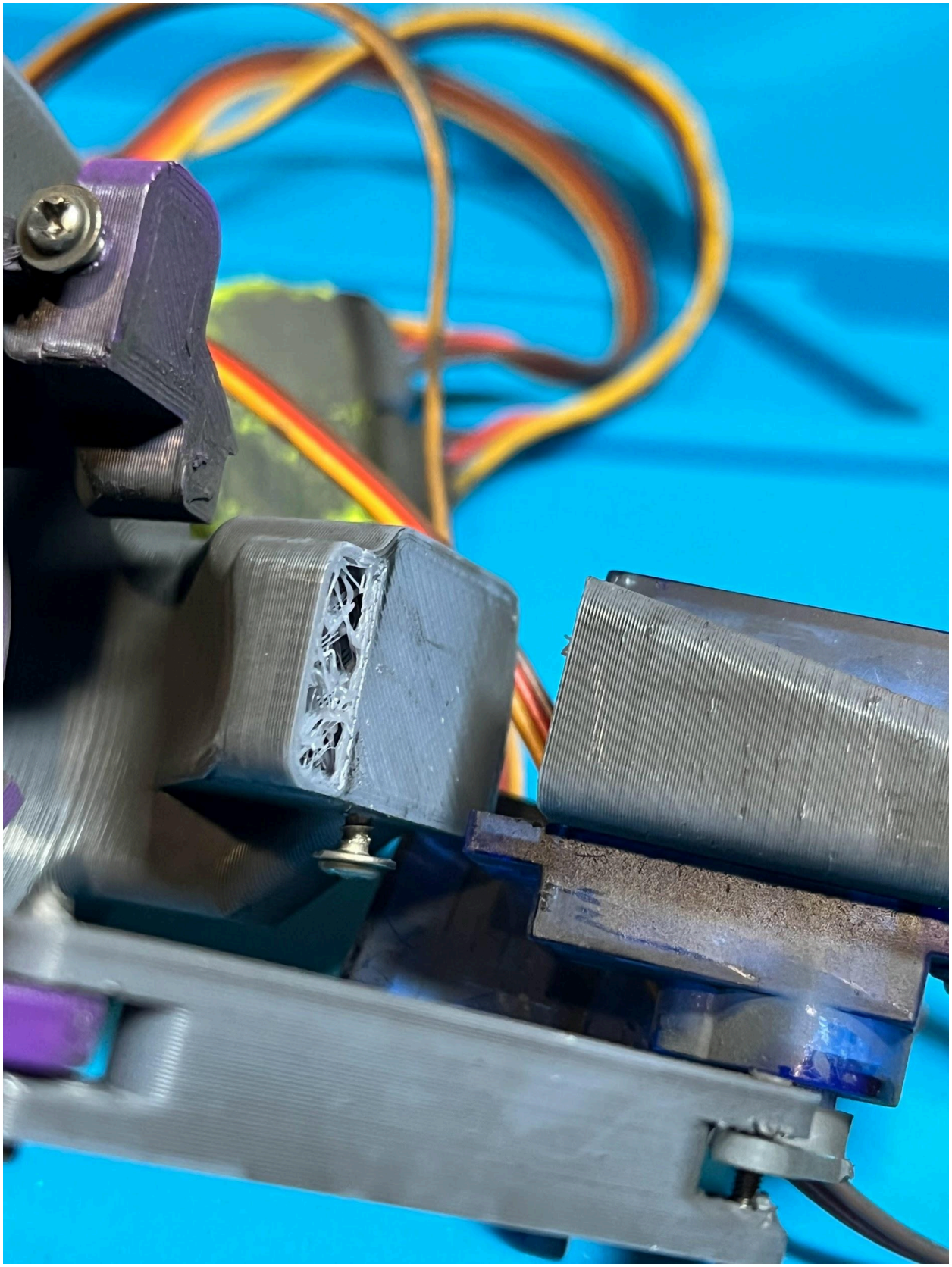
Picking up the rocket after the range was safe, it was clear the internals had not survived. It sounded like a rain stick with all the shattered plastic on the inside. After disassembling the rocket, it was clear the motor had exploded. There was soot all over, and it seemed the TVC mount had taken the brunt of the explosion. In order to understand what went wrong we first need to visualize the different parts of commercial solid rocket motors. There is a diagram below showing the different components of a commercial reloadable motor system. This is not the exact system flown on Genesis, but it is very similar.

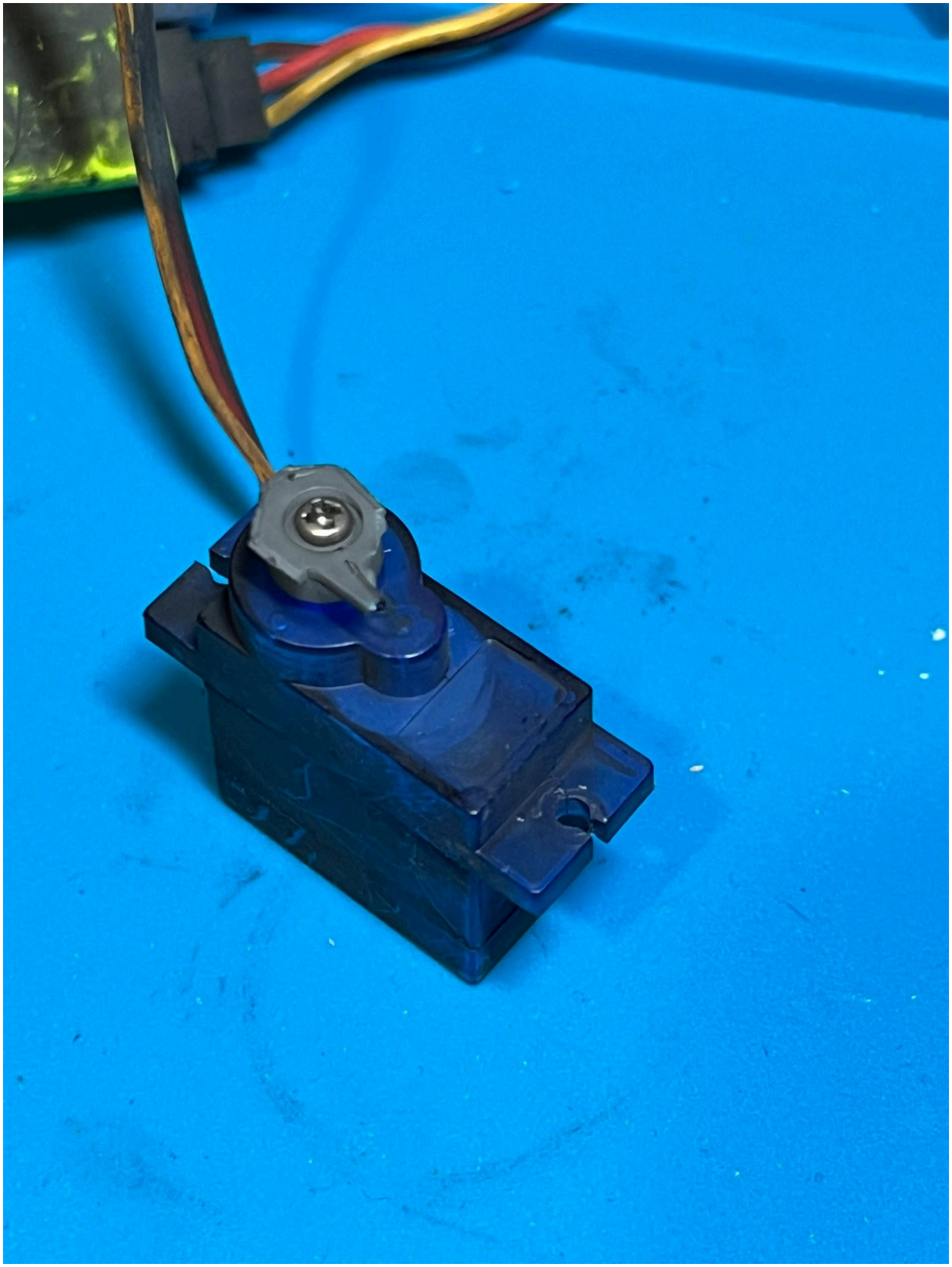


The parts you see leaving the rocket in the screenshot are the motor case, nozzle, and aft closure. The parts that stayed inside the rocket were the fuel grains and forward closure.
Because the casing was ejected with a high speed and force, that means according to Newton's 3rd law, the forward closure was shot the opposite direction with an equally high force. This is likely what destroyed the internal parts of the rocket.
With some math and video analysis, I estimated the casing to be traveling at around 30 meters per second, or 67mph. With their respective weights accounted for, this means the forward closure was traveling about 20 meters per second or 44 mph in the opposite direction, where it collided with all of the TVC components and the electronics sled. It’s important to know these are rough estimates, but it gives a clear understanding of the forces involved. The result was a catastrophic failure of the TVC mount and a fracturing of the electronics sled mount.
The following pictures were taken after the rocket was disassembled. All of these pictures have been blown up to full page size for clarity and detail.









After some searching through the grass, the casing, nozzle, and aft closure were found. The threads on both were tested as it seemed they still functioned correctly. This means the most likely cause of the motor failure is incorrect motor assembly. This would make sense, as it was my first time using a reloadable motor system. I attempted to follow the instructions closely when assembling the motor, but I think a small mistake was made when screwing in the forward closure. It is also possible that the forward closure became loose during transport, but this is unlikely. The friction of the threads and compressed O-ring make the forward closure difficult to remove after being installed, so the probability it was loosened during transport is very low. This O-ring was not in the correct position when inspecting it after the flight, but it is unclear if it was installed that way or if it was the explosion that caused it to shift.

The conclusion is almost comedic. Everything with the thrust vector control, computer, sensors, and flight profile performed as expected. The weak link in the chain was my skills with the motor system, which I put relatively little thought into. Had I flown this motor type before, it's possible I'd understand its tendencies. I put so much time and effort into the control side, I did very little research and testing with the motor!

The good news is this is easily fixable. Parts can be reprinted, and a new motor can be assembled. A second flight is possible, but it will have to be at a later date. I do plan on making a second flight happen, as it seems such a shame to stop now when I'm so close to a clean flight. I am anxious to see the full potential of the current control system, so I've already begun work on the Genesis Mk II.

7.0 CONCLUSIONS

This was an incredible project to undertake. I was very glad to have the time and resources to do something like this. Having watched others accomplish incredible feats with such simple tools, I've always wanted an excuse to try it for myself, and I'm so glad I took advantage of this opportunity. One of the goals of the project was to make an open source guided rocket that was cheap to build, cheap to repair, and easy to understand. The other goal was for me to learn more about specific technologies in UAS, and be able to replicate them on my own. While it would have been nice to see a flawless first flight, I knew I had to temper my expectations due to the sheer complexity of this project. After all is said and done, I consider this project a resounding success.

To say this has piqued my interest might be an understatement. I am constantly thinking about this project, even now after it has ended. Everyday I mentally work through new ideas, designs, and strategies to make a Genesis Mk II even better. Now I will be able to continue this project because I have the tools and expertise. With luck, I may achieve a successful flight in the future.

REFERENCES

Barnard, J. (2019, January 11). *Schematic Design - Landing Model Rockets EP. 4*. YouTube. https://www.youtube.com/watch?v=23gJY8a8rHw&list=PLyV774-3p8348Fl5V6ciIBh0ZA446q93I&index=5

Barnard, J. (n.d.). *Gallery*. BPS.Space. https://bps.space/pages/gallery

Chris' Rocket Supplies. (n.d.). Aerotech F12-3J Black Jack Rocket Motor. https://www.csrocketry.com/rocket-motors/aerotech-rocketry/motors/24mm/24/40-reloads/aerotech-f12-3j-black-jack-rocket-motor.html

GY-521 3 axis gyroscope + accelerometer module MPU-6050. All Top Notch. (n.d.).https://alltopnotch.co.uk/product/gy-521-3-axis-gyroscope-accelerometer-module-mpu-6050/

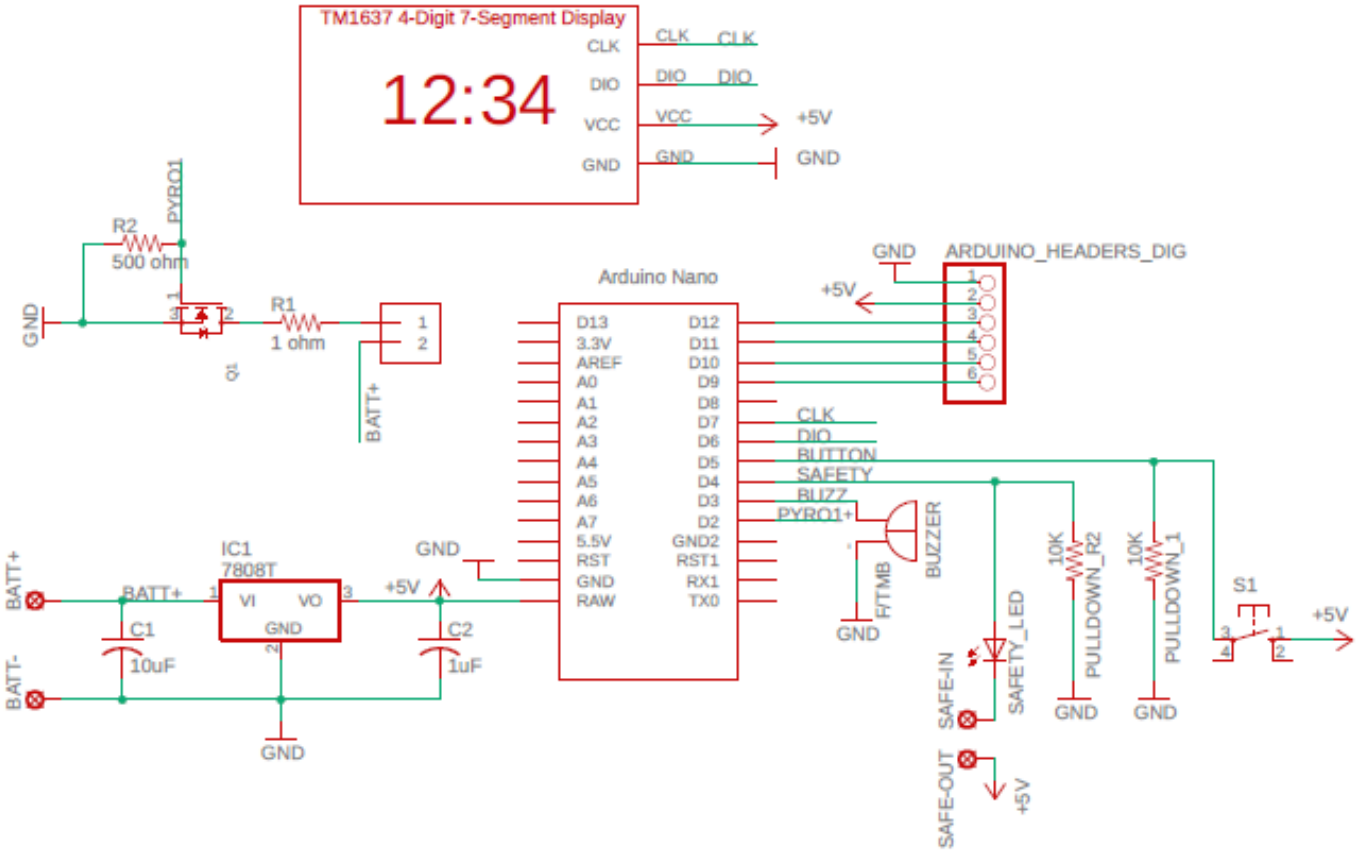
Adafruit BMP280 I2C or SPI Barometric Pressure & Altitude Sensor. Adafruit industries blog RSS. https://www.adafruit.com/product/2651

Adafruit SPI Flash SD Card - XTSD 512 MB. adafruit industries blog RSS. https://www.adafruit.com/product/4899

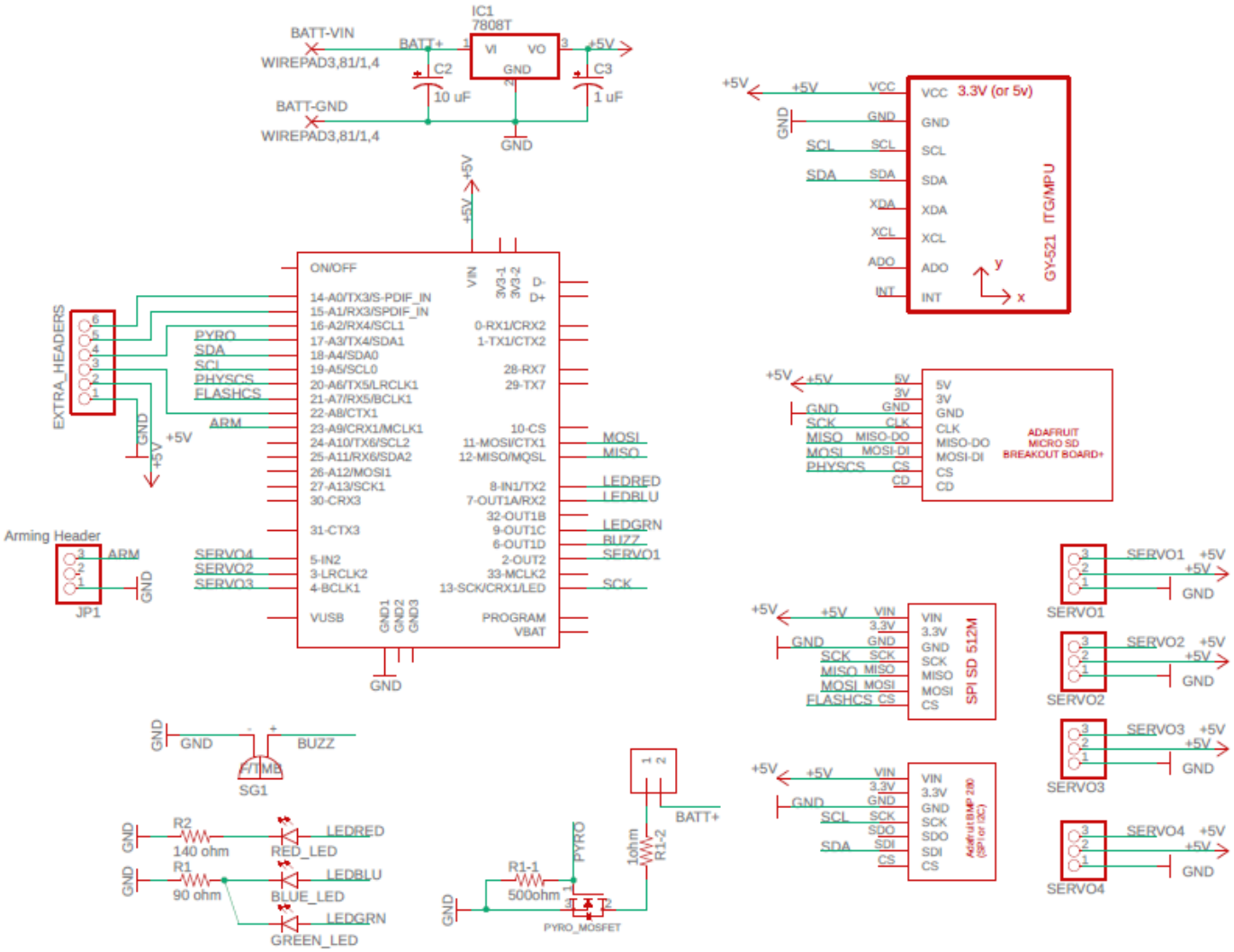
MicroSD card Breakout Board+. adafruit industries blog RSS.https://www.adafruit.com/product/254

reichelt elektronik GmbH Internet Team webmaster@reichelt.de. (n.d.). *TEENSY 4.0*. www.reichelt.at.https://www.reichelt.at/at/de/teensy-4-0-usb-ohne-header-teensy-4-0-p269006.html

APPENDIX A: GENESIS LAUNCH COMPUTER SCHEMATIC



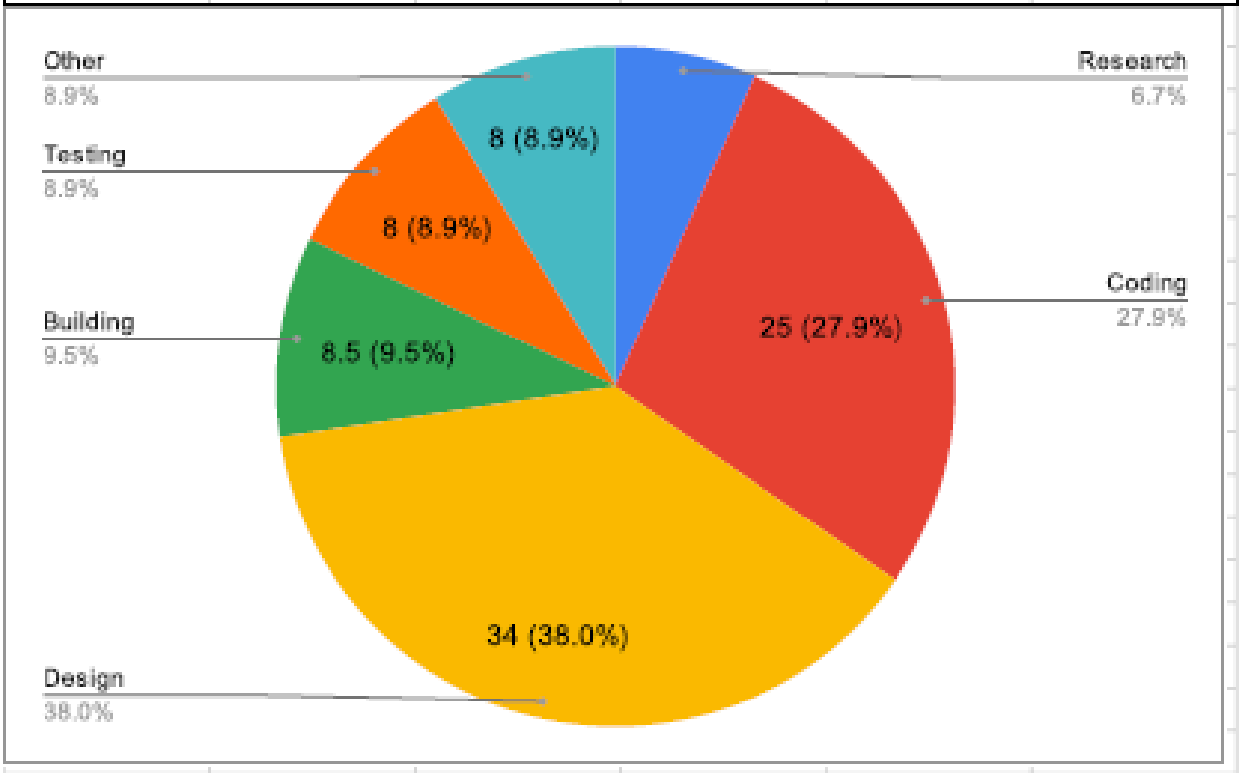
APPENDIX B: GENESIS FLIGHT COMPUTER SCHEMATIC



APPENDIX C: TIME LOG

Date	Hours	Activity	Activity Category	Total Hours:	97
01/09/24	1	UROC and NAR Coordination	Planning		
01/11/24	2	Schedule, Monday Board, Project Outline	Planning		
01/17/24	3	"Calibration" Mini Project Coding	Coding		
01/17/24	1	Signal Filtering Research	Research		
01/18/24	1	Recursive Averaging Filter Coding	Coding		
01/19/24	0.5	Low Pass and Kalman Filter Research	Research		
01/19/24	1	Integrating GitHub into Workflow	Documentation		
01/20/24	1.5	TVC Mount CAD Design	Design		
01/22/24	2.5	Coding IMU Noise Filters	Coding		
01/22/24	2	DSP Textbook Reading	Research		
01/23/24	2	"Display Angluar Velocity" Mini Project Coding	Coding		
01/24/24	1.5	TVC Mount CAD Design	Design		
01/25/24	4	TVC Mount CAD Design	Design		
01/25/24	0.5	STL Slicing and File Preparation for 3D Printing	Building		
01/26/24	1	TVC Mount Cad Design (Adjustments)	Design		
01/26/24	2	Launch Computer Design	Design		
01/27/24	1.5	Launch Computer Schematic Design	Design		
01/29/24	1	TVC Mount Cad Design (Adjustments)	Design		
01/29/24	3	Launch Computer PCB Design	Design		
01/31/24	1	Launch Mount CAD Design	Design		
02/05/24	2	Kalman Filter Research	Research		
02/05/24	2	Launch Computer PCB Assembly	Building		
02/09/24	1	"Display Orientation" Mini Project Cad Design	Design		
02/12/24	1	"Display Orientation" Mini Project Coding	Coding		
02/12/24	0.5	Test Stand CAD	Design		
02/12/24	4	"Display Orientation" Mini Project Coding and Debu	Coding		
02/14/24	2	"Single Axis PID" Mini Project Coding	Coding		
02/14/24	1.5	Launch Computer PCB Re-assembly	Building		
02/16/24	0.5	PID Loop Research	Research		
02/16/24	0.5	"Single Axis PID" Mini Project Coding	Coding		
02/21/24	2	Flight Computer Design	Design		
02/21/24	3	Flight Computer Design	Design		
02/22/24	2	Flight Computer Design	Design		
02/23/24	1	Flight and Launch Computer Adjustments	Design		
02/26/24	2	Flight and Launch Computer Adjustments	Design		
03/03/24	2	Deployable Nosecone CAD Design	Design		
03/05/24	1	Coding Flight Software	Coding		
03/06/24	3	TVC Mount Cad Design (Adjustments)	Design		
03/06/24	2	Coding and Debugging	Coding		
03/07/24	2	Flight Computer Adjustments	Design		
03/10/24	1	Soldering and PCB assembly	Construction		
03/12/24	1	Soldering and PCB Assembly, minor Debugging	Construction		
03/13/24	1	Electronics Sled Mount CAD	Design		
03/15/24	1	First integration of Genesis Rocket	Costruction		
03/15/24	3	TVC PID Coding and Testing	Testing		
03/18/24	1	Deployable Nosecone CAD Design Adjustments	Design		
03/18/24	1	Testing Motor Mount Assembly	Construction		
03/23/24	2	Test Stand Construction	Construction		
03/26/24	1	Test Stand Parts CAD	Design		
03/28/24	1	TCV PID initialization coding	Coding		
04/02/24	3	SD and Flash Data Logging Mini Project	Coding		
04/11/24	2	SD Data Logging Debugging	Coding		
04/12/24	1	TVC PID Coding and Testing	Testing		
04/12/24	2	Flight Code	Coding		
04/16/24	1	Black Powder Charge testing	Testing		
04/17/24	3	Flight Code State Machine testing	Testing		
04/19/24	2	Launch Preparation	Other		
04/20/24	2	Launch	Other		

Research	Coding	Design	Building	Testing	Other
6	25	34	8.5	8	8
1	3	1.5	0.5	3	1
0.5	1	1.5	2	1	2
2	2.5	4	1	1	1
2	2	1	1	3	2
0.5	1	2	1		2
	4	1.5	1		
	2	1	2		
	0.5	3			
	1	1			
	1	1			
	3	0.5			
	2	2			
	2	3			
		2			
		1			
		2			
		2			
		2			
		1			
		1			



APPENDIX D: COSTS

Date	Cost	Item	Category
01/10/24	$10.00	UROC Yearly Membership	Research
01/12/24	$30.00	NAR Yearly Membership	Research
01/12/24	$5.00	BPS Monthly Membership	Research
01/14/24	$143.31	Texbooks on Signal Processing and Control	Research
01/15/24	$13.00	MPU6050 Sensor 5 Pack	Electronics
01/16/24	$14.93	Adafruit BMP280 Breakout Board	Electronics
01/17/24	$19.95	Adjustable Action Camera Tripod	Documentation
01/19/24	$11.97	Flash Chips (SPI and I2C)	Flight Computer
01/20/24	$41.72	F12 Motor Reloads	Motor
01/20/24	$25.76	74mm x 18" Body Tube (4 pack)	Construction
01/20/24	$89.26	Reusable Aerotech RMS-24/40 Hardware Set	Motor
01/24/24	$41.40	Electronics Components	Electronics
01/24/24	$8.00	Electronics Components	Electronics
01/25/24	$27.90	Tools	Construction
01/25/24	$25.75	Assorted Fasteners	Construction
01/29/24	$21.80	Launch Computer PCB Manufacturing	Electronics
01/31/24	$66.56	Drill Bits, Soldering tools, SD cards	Construction
02/18/24	$42.92	PLA + 3D printer Filament, Parachutes	Construction
02/21/24	$29.00	Teensy 4.0 (for flight computer)	Electronics
02/21/24	$21.00	Electronics Components	Electronics
02/22/24	$9.00	Electronics Components	Electronics
02/26/24	$15.38	Discounted Launch and Flight Computer PCB Man	Electronics
03/03/24	$11.00	Linear Actuator 5V For nosecone	Actuators
03/04/24	$9.99	Assorted Resistors kit for electronics	Electronics
03/06/24	$42.00	SD Card Breakouts and SPI Flash Breakout	Electronics
03/07/24	$14.00	assorted female 2.54mm header pins	Electronics
03/07/24	$20.00	Flight Computer PCB revision 1-2	Electronics
03/20/24	$7.50	Micro USB Adapters	Electronics
04/02/24	$19.29	Parachute	Construction
Total Cost:	$837.39		

Resarch	Electronics	Construction	Motor System	Other
$188.31	$278.97	$182.42	$41.72	$30.95
$10.00	$13.00	$27.90	$41.72	$19.95
$30.00	$14.93	$25.75		$11.00
$5.00	$11.97	$66.56		
$143.31	$41.40	$42.92		
	$8.00	$19.29		
	$21.80			
	$29.00			
	$21.00			
	$9.00			
	$15.38			
	$9.99			
	$42.00			
	$14.00			
	$20.00			
	$7.50