VUW Rocket Avionics Package Technical Manual

Contributing Authors: David Dobbie, Jonathan Elliott, Nathaniel Fallow, Grayson Hughes, Tyaan Singh

ROCKET CONTROL SYSTEM

The rocket system has a double integral transfer function. The derivation of this transfer function is shown in “system transfer function derivation.jpg” in the control system folder in the gitlab repository. The force due to air friction Fd is ignored as negligible compared to the force applied by the motor. The rocket is modelled as freely rotating around the centre of gravity. The system was modelled in matlab and has a constant phase of 180 degrees, so is inherently unstable. A lead controller will be designed to control the rocket pitch and yaw angle.

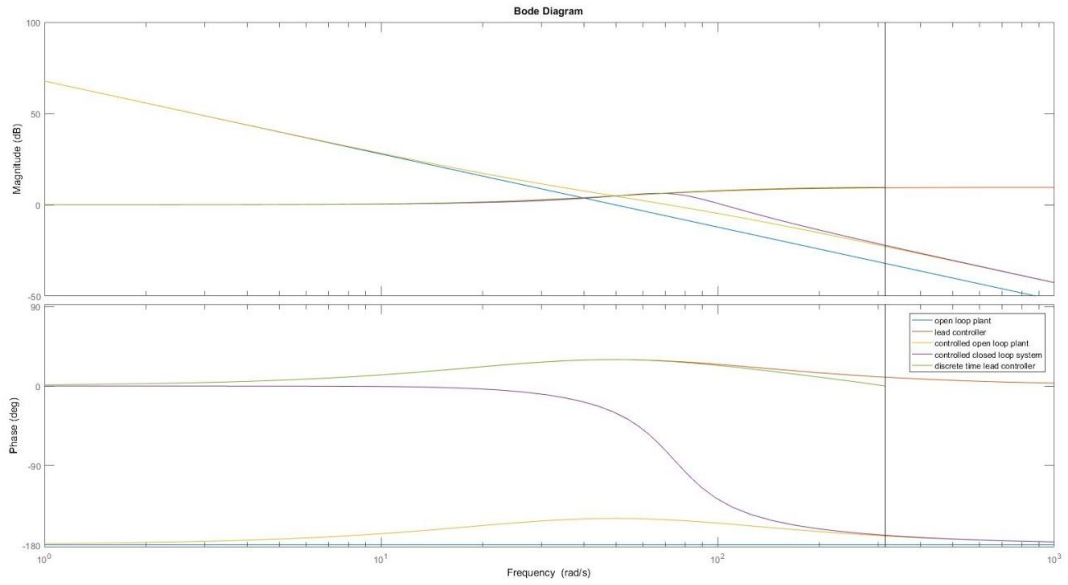
The lead controller will be designed to have a low frequency zero and a high frequency pole around the 0dB frequency of the system transfer function. A maximum phase lead of 30 degrees will be added at the 0dB frequency of the system.

The system transfer function is given by (Ft*Dm)/Is^2, where Ft is the thrust force of the motor, Dm is the distance from the motor to the centre of gravity and I is the rotational inertia. The lead controller implemented will have the form 1/alpha((s-Wb)/(s-Wb/alpha)), where frequencies Wb and Wb/alpha straddle the 0dB frequency of the system.

When the transfer function is converted to discrete time using a first order bilinear transformation, a discrete time digital filter can be created using the current values of the input, and the values of the input and output 1 sample ago.

controlSystem.m matlab code in control system folder in repository calculates the continuous time lead controller transfer function and discrete time transfer function. The inputs are the values of rotational inertia, distance to centre of gravity, motor thrust force and sampling frequency. These can be altered for other shapes and weights of rocket and matlab will output an applicable controller.

A bode plot of each of the system transfer functions is shown here.
Definitions are as follows:
* open loop plant
  - Uncontrolled system with motor angle as input and rocket angle as output
* lead controller
  - Lead controller transfer function on its own
* controlled open loop plant
  - Open loop plant multiplied with lead controller, open loop (no feedback)
* controlled closed loop system
  - Controlled open loop plant with feedback. This represents the final rocket system as a whole.
* discrete time lead controller
  - Transfer function of lead controller by itself when sampled at 100Hz, rather than continuously. The higher the sampling rate, the closer this will match the continuous controller.




```
>> controlSystem

Cd =

  2.4 z - 1.801
  -------------
   z - 0.4004

Sample time: 0.01 seconds
Discrete-time transfer function.

Warning: The closed-loop system is unstable.
> In ctrlMsgUtils.warning (line 25)
  In DynamicSystem/margin (line 65)
  In controlSystem (line 30)

PhaseMarin =

   48.4042
```

The Z transform equation that matlab outputs can be rearranged to give a discrete time controller implementable in software.

Y(n) = 2.4*X(n) - 1.801*X(n-1) + 0.4004Y(n-1)

Y(n) is the current output. X(n) is the current input. Y(n-1) is the output one sample ago. X(n-1) is the input one sample ago.

The output of the Matlab program shows that the controlled system has a phase margin of about 48 degrees which means it is very stable and should have lots of room to move if any of the constants defined at the top change. This is actually very likely, as the motor thrust force has been modelled as a constant 11N, which in reality is not the case at all.
Ignore the error message that it is unstable, matlab will always say this when plotting a discrete time transfer function.

CONTROL SOFTWARE

The control software is written in Arduino C and implemented on a teensy 3.2 microcontroller.
The control code was written using PlatformIO, and can be easily used by importing the controlCode folder as a platformIO project. It can also be run using the arduino IDE, but the MPU9250 (the IMU) library must be copied into your arduino libraries. The library is included in the control code folder.

To set up the code to run on a teensy 3.2, the 2 servos that gimbal the motor must be set up to be plugged in to the assigned servo pins. The IMU must also be set up for I2C communication. Connect SDA and SCL to the SDA and SCL pins on the teensy.

The control software calculates the current angle of the rocket by reading values from an IMU (Inertial Measurement Unit). It uses these values to calculate the desired motor gimbal angle using the lead controller. I2C communication is used to commuicate with the IMU using the Teensy3.2 and outputs voltages to the servo motors that control the motor gimballing angle.

There are a few variables that must be altered to run this code for different rocket systems. First of all, Xcentre and Ycentre must contain the values that set the servo angles to their actual centre. Next, servoXratio and servoYratio must be altered to represent the ratio between the servo angle and the actual angle the motor moves to. In our small scale rocket system this represents the ratio between the distance from the point the motor is moved from to the pivot point, and the servo arm length. Multiplying the servo angle by this ratio gives the angle the motor sits at. Finallly servoOffsetRatio represents the inherent servo offset due to the servo actually moving a little less than the angle you set it to.

To use this code for a different control system, the lead controller values in the main loop can be altered.

CONTROL SYSTEM EVALUATION

The outcome of the control system, when used with the small scale cone rocket design was not what was expected. As soon as the motor thrust began, it seems like the motor got stuck at the max angle on one axis and the rocket spiralled. After checking the teensy after launch, it seems like the teensy turned off, causing the motor to stop gimballing.

A possible reason for this happening is that the force from the motor when not directly upwards has a horizontal force component to it that may have been greater than the servos max standing torque, pushing them past their limits. Other possible reasons include the batteries that were used pushing the centre of gravity behind the motor gimbal force and making the rocket gimbal unstable. This would explain why the control system did not work but does not explain why the teensy turned off during launch. This could possibly be due to the force of the rocket smashing into the ground after the control system failed.

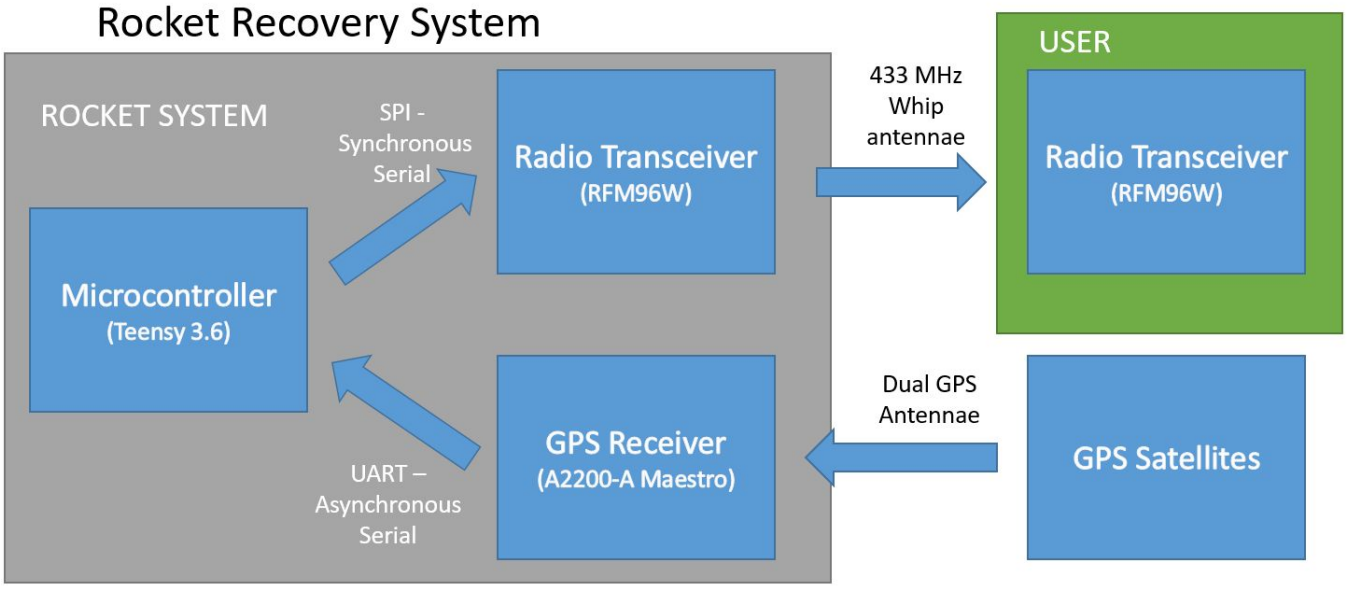
The control system has been vaguely tested empirically by rotating the rocket and watching to see the motor correcting in the correct directions. A good future pre-flight test may be to hold the rocket steady at the centre of gravity on one axis with string or something, and fire up a motor and ensure that the rocket stays pointing upwards, as in flight it will rotate around the centre of gravity.

ROCKET RADIO/GPS RECOVERY SYSTEM

This module of the rocket contains the microcontroller programming for the GPS system, the radio transmitter, and the data logging during a launch. The PCB design has provided the system required to ensure that these individual components, after construction, are operational. This document details the motivation for this package, how to implement it, any major design issues encountered, and finally any potential future work to build on.

PURPOSE

Amateur rockets get lost often. Typically, with larger scale rockets with possible ranges of the kilometres, there are serious issues with finding the package afterwards. To mitigate these long-range detection issues, a recovery system was implemented into the rocket package. This module provides convenient positional information of the rocket's whereabouts after a launch. Furthermore, it can log data into the microSD card slot provided with the Teensy 3.6 microcontroller. Additionally, the system was made more robust to handle possible failures such as final orientation of the rocket after landing, long range radio systems for obstructive terrain, and simple 'whip' antenna for minimising directionality.



FEATURES

- Long range RFM96W 433MHz radio system transmitting consistently and constantly to provide a signal to track and detect. At least 600 metres in an obstructive and involved environment (multiple concrete and metal buildings)
- Transmitted GPS coordinate data of the rocket system, as well as operational time and signal strength.
- GPS achieves lock within 10-15 minutes
- Dual antenna GPS system which is switched often to allow for maximal coverage and chance for the GPS system to get useable satellite signals.
- Micro SD card writes at every time a message is initialised, allowing for diagnostic information of the rocket to be tracked and maintained.
- Serial output receiver system that can pick up the transmitted data remotely from the rocket – includes remote diagnostics
- Passive antenna system on both the onboard_module and the home_base.

USING IT

The system can be used by uploading the appropriate code to a constructed system – refer to the PCB documentation on construction. The system used can be the Arduino IDE (https://www.arduino.cc/en/Main/Software) with Teensyduino (https://www.pjrc.com/teensy/teensyduino.html) add on. If the Arduino IDE is being used for the system, then the required libraries need to be on /User/Arduino/libraries directory.

The following libraries were utilised for this module of the system:

- RH_RF95.h RadioHead library for Arduino radio interfacing (http://www.airspayce.com/mikem/arduino/RadioHead/classRH__RF95.html)
- SD.h provides functionality to write onto the SD card on the rocket module (https://www.arduino.cc/en/Reference/SD)
- SPI.h provides functionality on reading the radio on the system (https://www.arduino.cc/en/Reference/SPI)
- TimeLib.h for parsing the time elapsed on the system (https://github.com/PaulStoffregen/Time/blob/master/TimeLib.h)
- TinyGPS++ parses the Serial GPS data into usable data to transmit (http://arduiniana.org/libraries/tinygpsplus/)

To use the system, the recovery guide included with the repository details how to read data from the home_base system.

How to upload the receiver (home_base) system:

- Upload home_base_rev2.ino onto the revision 2 PCB or home_base _rev3.ino onto the revision 3 PCB. Only difference is the wiring of the radio module through SPI (Serial Peripheral Interface) communication.

How to upload the transmitter (onboard_module) system:

- Upload onboard_module_rev2.ino onto the revision 2 PCB or onboard_modul_rev3.ino onto the revision 3 PCB. Only difference is the wiring of the radio module through SPI (Serial Peripheral Interface) communication. Note that revision 2 does not have a GPS module onboard so it cannot give any numerical positional information.

DEVELOPMENTAL NOTES/ISSUES

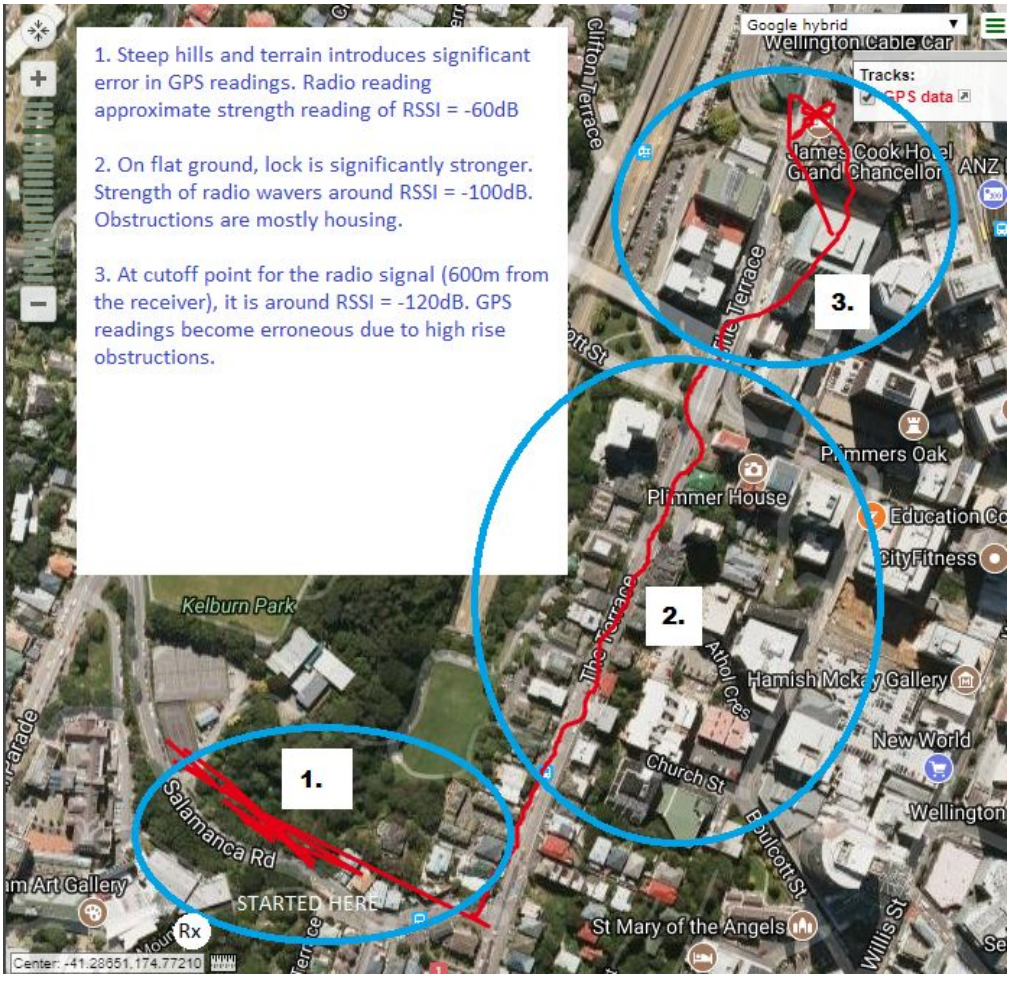
- A GPS is inherently difficult to test without a source antenna indoors to take the place of a satellite. It requires making the system work outdoors to confirm operationality
- The considerable risk of the project was that the impedance of the track towards the GPS antenna would not be functional. Therefore, further PCB development needs to be very careful with that (and should consult the A2200-A datasheet (http://www.richardsonrfpd.com/resources/RellDocuments/SYS_29/GPS_Receiver_A2200_User_Manual_V1_0.pdf).
- The antenna on the RFM96W radio is a simple quarter length whip antenna soldered to the antenna pad. This will need to be removed if anyone wants to use other antennas through the conventional SMA connector. The whip antenna is sufficient for sub 1km retrievals.
- There were missing parts to revision 2 and revision 3 of the PCB such as required pull up resistors or required wire inputs. This was especially true for the GPS.
- Be careful with reading the datasheet and PCB circuit diagrams.
- Utilising I2C for IMUs (inertial measurement units) is difficult and not typically reliable. The Teensy 3.6 will REQUIRE pullup resistors to successfully communicate on this protocol.

- The Arduino IDE with Teensyduino is reliable after getting the appropriate libraries installed and sorted out.

SYSTEM EVALUATION

The system was given a small scale test by moving the transmitter around a several hundred metre distance while the receiver was in a fixed location (lower left corner). The map was generated from the stream of serial data on the ground base and then sent into a web gps visualiser (http://www.gpsvisualizer.com/). The involved environment of Wellington Central stress tested the effectiveness of the recovery system. There was minimal to no line of sight between the receiver base and the transmitter. This was done to recreate similar constraints in finding the rocket. The 17.3cm whip antenna on both the receiver and the transmitter was the worst case scenario of the system. This gives a significant indicator of the robustness of such a simple antenna system.

This system should not need to be reconfigured and rebuilt if the RFM96W is kept as the radio to use. It should be able to be implemented directly as a beacon on a rocket in its current state.



POTENTIAL FUTURE WORK

- Add IMU readings to change the antenna being used according to the orientation of the rocket. Will lead to more consistency than continual switching.
- The radio system can have an addressing system to make it more compatible with diverse types of 433MHz radios. Only using the RFM96W is a feature limitation in the development of the system.
- An external directional antenna would increase the range finding on the system significantly. Also, an active antenna can possibly lead to a more advantageous range.
- Fully encasing the radio system would make it more robust in outdoor conditions

ROCKET PRINTED CIRCUIT BOARD

DESIGN

The printed circuit board (PCB) of the rocket was designed using KiCad, an open source PCB design software.Many of the components used did not have premade schematic symbols nor layout footprints, and so were custom made based on dimensions given in datasheets.

FEATURES

- Teensy 3.6 microcontroller
- A2200A GPS
    - includes precise impedance-matched traces to the antennae (50ohm). Traces between the GPS unit and the two antennae were kept 1.44mm wide where possible for this reason, and a thin antenna ground trace was run around them to provide additional shielding.
    - A GPS typically has a single antenna, however this board implements two antennae facing in opposite directions, as they are highly directional. This setup allows the microcontroller to switch between the active antenna using an RF switch to maximise the chance of successful satellite lock.
- - RFM96W Radio
    - Connected to the teensy via SPI.
- MPU9250 IMU
    - Connected to the teensy via I2C. The footprint used is outdated, as the IMU breakout it was designed for was found to be unreliable, and was replaced with another more reliable breakout.
- Two servo footprints
    - Each has a central ground pin, a 5V supply pin, and a control pin. These pads were not tested.
- Piezoelectric buzzer
    - Connections for a simple piezo buzzer to produce loud noise to aid recovery. Untested.
- Igniter
    - A simple FET-controlled channel between the terminals of the power supply. Untested, considered legacy and should be replaced with a better system.
- Regulated power supply
    - Includes a bypass electrolytic capacitor, diode to prevent damage from accidental reverse polarity in the power supply, and 5V regulator to ensure input voltage is within the safe range of the teensy’s onboard regulator.

PCB EVALUATION

The radio and GPS systems on the PCB have been validated experimentally, as discussed in the Radio/GPS Recovery System Evaluation notes. The GPS was the sub system requiring the most precision, so having confirmation that it works is immediately a great asset. Some additional connections were needed to ensure good operation - a simple wire from the A2200A enable pin to a digital pin on the teensy, as well as pull up resistors from the I2C pins to Vout, all on the A2200A.

The PCB is lacking in power buffering of components which potentially makes it more susceptible to errors due to transient fluctuations in the power supply. While this was not an issue in testing of the radio

and GPS systems, it could drastically reduce the effectiveness of the electronics in flight if components are not properly buffered.

The igniter subsystem is not a safe design, as it essentially creates a short between the power supply terminals. This will very rapidly drain any battery flown with the system and almost certainly cause a brownout in the power supply to the other components on the board including the teensy.

Finally, the board was too large to fly in the small scale (C-class) cone rockets used for flight testing. This meant that the board and its sub systems could not be validated under full flight conditions. This will not be an issue for future flights using the larger G-class rocket design, as there will be much more room to mount the board.

POTENTIAL FUTURE WORK

- Replace ad hoc green wires on the GPS with proper traces
    - GPS enable pin to a digital pin on teensy
    - Pull up resistors between I2C and Vout pins - SMD resistors and traces could be run on underside of the board.
- Isolated high voltage power supply for igniter to prevent it browning out the rest of the system.
- Replace through hole components with SMD equivalents (especially the 7805 regulator) to save on space and weight.
- Update IMU breakout footprint.
- Add buffer / bypass capacitors to better ensure stable power supply to all components.
- Cut down size where possible.

MOTOR GIMBAL SYSTEM

PURPOSE

Amateur rockets are typically flown with only passive control elements. Passive elements will normally provide an approximately straight flight path but can be prone to disturbances such as winds or thermals. These disturbances can often cause changes to a rocket's flight course, especially low altitude winds near the launch site. Additionally the passive control elements do not provide control at low speeds which causes the rockets to rely on a launch rail when initially accelerating. The Motor gimbal system will provide an active control element so that the rocket can react to any disturbances. This gimbal will also remove the rockets requirement for the launch rail meaning that the rocket can be launched from various different areas including airborne platforms.

OPERATION

The gimbal system is built upon the gyroscope concept. This concept uses two rings that have a pivoting axis for either x rotational movement or y rotational movement. Using this, the gimbal has the ability to freely rotate the rocket motor through the required degrees of movement. This motor movement will then give the rocket the ability to make any course corrections that the control system instructs.

CONSTRUCTION

Note: the MK3.3 gimbal is the only iteration to be launch tested therefore meaning any launch issues with the MK4.2 gimbal have not been discovered.

MK3.3

The MK3.3 is the gimbal system designed to be used in the Athena MK2.1 (Cone) Rocket. The rotational movement is driven by two HKM-282A servos, one servo per axis. The voltage input of these servos are to be connected to the 3.3V lines of the Teensy microcontroller, the signal wires of the X direction servo and Y direction servo is to be connected to pin 14 and pin 23 respectively finally the servo grounds can be connected to any of the Teensy’s ground lines. The gimbal rings are constructed using a PLA 3D printer. The gimbal rings are assembled using two M3x5mm screws (inner ring and motor mount) and two M3x8mm screws (outer ring and inner ring) where the smaller rings are placed inside the larger rings. The inner ring is orientated so that its control arm is parallel to the servo placed in the outer rings servo housing. The motor mount ring is then oriented so that the majority of its mass is on inner rings servo housing side. The servos are mounted using the hardware provide with the servos. The rings are then connected to their adjacent servos using control rod. This rod is bent and threaded through the control rod holes in both the servo control arms and ring control arms.

MK4.2

The MK3.3 is the gimbal system designed to be used in an up to G class motor rocket that was currently in development. The rotational movement is driven by two Hc2422T servos, one per axis. These servos are to be connected to ports I2 and J3 on the Athena MK1 PCB, at this point of development the orientation of the servos (X and Y) is undetermined. The gimbal rings are constructed using a PLA 3D printer. The gimbal rings are assembled using four M3x10mm screws where the smaller rings are placed inside the larger rings. The inner ring is orientated so that its control arm is parallel to the servo placed in the outer rings servo housing. The motor mount ring is then oriented so that the majority of its mass is on inner rings servo housing side. The servos are mounted using a collection of M3 screws, their lengths include 2x 10mm, 4x 20mm and 2x 35mm. The rings are then connected to their adjacent servos using

control rod. This rod is bent and threaded through the control rod holes in both the servo control arms and ring control arms.

POTENTIAL FUTURE WORK

- Estimated required angle possibly too small for smaller rockets i.e. Athena MK2.1 (Cone Rocket), therefore the range of motion for the MK3.3 gimbal could be increased for improved control during initial take-off.
- The servos on the MK3.3 gimbal appeared to have a slow response likely due to them being underpowered, therefore the servos should be replaced with higher torque and rotation speed.

ROCKET DESIGN

ROCKET MODELS

The different rocket designs were modelled using OpenRocket software which could simulate the performance when launched. However this only simulated the passive behaviour of each rocket. The components were made with OnShape, a cloud based CAD program.

CONSTRUCTION

The components for the rocket were made with a 3D printer provided by the client. The were made of PLA filament which did melt under high heat so the could not be used in areas directly adjacent to the motors nozzle or ejection charge.

ATHENA MK 2.0

The Mk 2.0 is the passively stable, single stage, C-motor rocket.
The rocket is aerodynamically stable due to the large “cone” that greatly increases the drag force that the rocket experiences. This was done as it reduced the overall rocket size as well as the maximum height the rocket would reach. Limiting the rocket apogee height with aerodynamic drag was chosen over limiting by increasing the rocket’s weight because it also improved the rocket’s aerodynamic stability.
The motor mount is incorporated directly into the cone and a small hole is in the cone directly above the motor to vent the ejection charge. A second (square) hole was placed adjacent to the motor mount to allow the Athena Mk 2.0 to be placed on the launch rail. This did reduce the aerodynamic stability of the rocket but due to the low apogee point this was deemed to be an acceptable drawback.

ATHENA MK 2.1

The Mk 2.1 is the single stage, C-motor rocket with active stabilization in the form of the gimbal.
The motor mount was removed from the rocket body, now being a component of the gimbal. The opening in the cone from the launch rail was removed as the active stabilization system allowed the rocket to be launched without a launch rail.
Approximately 41 mm from the base of the cone, six evenly distributed structural supports extend from the cone to a ring in the center to which the Gimbal is mounted. Sections of the ring were removed, between two of the support structures on opposite sides, so that three support beams were connected together on each side of the cone. This was done to allow the gimbal to be placed within easily and to create space for the electronics.
The electronics placed within the Athena Mk 2.1 was a Teensy 3.2 and an MPU 9250 IMU. A mount for the Teensy 3.2 is placed on the inner wall of the cone.

ATHENA MK 3.0

The Mk 3.0 is the two stage, actively stable G-motor (with an I-motor second stage) rocket.
The first stage can be used with smaller motors, such as a C or D class, by incorporating the “booster seat” which is a motor mount that can be placed within the gimbal.
The first stage is actively stable with the use of the MK4.2 Gimbal with its outer ring and support structures now being directly connected with the rocket body. This removes the need to screw the gimbals outer ring in reduces the risk of the gimbal disconnecting. It contains a mount for the PCB (revision 3) and a 7.2V 2-cell LiPo battery.

The top of the first stage body is coupled to the bottom of the second stage with a friction fit. The first stage is designed to be much wider than the next stage, also incorporating a nose-cone, to allow for the rocket to drag separate. The connecting section includes shunts directly beneath the nose-cone so the larger motor will not damage the lower stage when it ignites.
The second stage is a more standard rocket which is passively stable. It contains electronics for radio and GPS communication. The PCB used in both sections are designed to be of equal size, however the first stage can use the Teensy 3.2 instead of the larger Teensy 3.6 to drive the Gimbal.

POTENTIAL FUTURE WORK

- More appropriate mounts could be added for the IMU, and batteries used when testing the Athena Mk 2.1
- The motor mount of the gimbal could have been extended to increase the moment arm (and torque) applied by the servos. This would have also reduced the heat experienced by the components
- Adding a shunt system for the first stage ejection charge to the Athena Mk 3.0 so that it would no longer be required to remove the motor's ejection charge pre-launch.
- Sectioning the components into multiple parts due to possible 3D printing restrictions
- Implement an electronic separation system between the stages and adding wiring bypasses (small holes) to the bodies of both stages so the electronics can communicate between stages