# End-to-End Data Pipeline to improve a Vertical Orientation System for a Sounding Rocket


MECH5080M Team Project – Individual Report

***End-to-End Data Pipeline Architecture to improve a Sounding Rocket Stability Control***

*Author: Alexandra Posta 201318973*

*Supervisor: Dr Jongrae Kim*

*Industrial Mentor: Theo Gwynn*

*Examiner: Dr Jongrae Kim, Professor Robert Kay*

*Date: 30/04/2024*

SCHOOL OF MECHANICAL ENGINEERING

UNIVERSITY OF LEEDS

MECH5080M TEAM PROJECT 60 credits

TITLE OF PROJECT

End-to-End Data Pipeline to improve a Vertical Orientation System for a Sounding Rocket

PRESENTED BY

Alexandra Posta

OBJECTIVES OF PROJECT

The project's aim is to aid the VOS of sounding rockets through the development and integration of a software-firmware system. This system incorporates the active control of canards and advanced data management tools to support continuous improvement.

IF THE PROJECT IS INDUSTRIALLY LINKED TICK THIS BOX AND PROVIDE DETAILS BELOW ☒

COMPANY NAME AND ADDRESS:

Airbus Defence and Space
Gunnels Wood Rd, Stevenage SG1 2AS

INDUSTRIAL MENTOR:

Theo Gwynn

THIS PROJECT REPORT PRESENTS OUR OWN WORK AND DOES NOT CONTAIN ANY UNACKNOWLEDGED WORK FROM ANY OTHER SOURCES.

SIGNED

DATE 30/04/2024

## Contents

## List of Figures

## Nomenclature

| | |
|---|---|
| API | Application Programming Interface |
| COTS | Commercial-Off-The-Shelf |
| ELT | Extract Load Transform |
| GPIO | General Purpose Input/Output |
| HIL | Hardware-in-the-loop |
| HTTP | Hypertext Transfer Protocol |
| IDE | Integrated Development Environment |
| LQR | Linear–quadratic regulator |
| LURA | Leeds University Rocketry Association |
| ORM | Object-Relational Mapping |
| PCB | Printed Circuit Board |
| REST | Representational State Transfer |
| SQL | Structured Query Language |
| SWD | Serial Wire Debug |
| SPI | Serial Peripheral Interface |
| UART | Universal Asynchronous Receiver/Transmitter |
| UK | United Kingdom |
| UKRA | United Kingdom Rocketry Association |
| USB | Universal serial bus |
| VOS | Vertical Orientation Systems |

## Abstract

This thesis presents the development of a data pipeline designed to aid the active vertical stabilisation system of a sounding rocket. The primary objective was to create a robust architecture that connects firmware and software components necessary for flight control operations of Aptos, a module that contains a secondary set of fins actuated individually to stabilise trajectory.

The project involved the development of a flight firmware in bare metal C, setting up a development environment that includes the main loop routine, helper functions, and a controller initially modelled in MATLAB and Simulink. Furthermore, methods for storing and visualising flight data were established and tested to support the pipeline. The system's performance was ultimately tested during a rocket launch campaign, where hardware was mounted to a sounding rocket and operated under active control. Data was successfully collected during flight, ingested in a centralised database storage unit and visualise for further controller gain tuning.

The projects confirms that a well-integrated data pipeline is beneficial for the advancement and refinement of aerospace technologies, particularly in the development of flight controllers for sounding rockets.

# Chapter 1. Introduction

## 1.1 Introduction

Sounding rockets serve as pivotal instruments for atmospheric research and suborbital experiments. The flight trajectory of a rocket can be affected by external factors such as winds which lead to uncontrolled dispersion and lower apogees [1]. To minimise the effects of external factors and improve the flight trajectory, active vertical controllers can be used. This report presents the development of an end-to-end data pipeline meant to facilitate the active stabilisation of rockets. It focuses on the application of Vertical Orientation Systems (VOS) which computes the desired orientation of the rocket by controlling a secondary set of fins known as canards [2], [3], [4].

The end-to-end data pipeline is enabled through various coding platforms integration. It merges low-level firmware, which manages the actuation of the canards, with high-level software algorithms that process data streams, analyse flight dynamics, and execute stabilisation strategies. The following chapters outline the pipeline components: firmware development, centralised database, and data visualisation. Chapter 2 introduces the concepts, Chapters 3 to 5 detail each component, and Chapter 6 discusses system integration, followed by conclusions and future work.

The report presents a system where firmware and software are integrated elements of a single, robust architecture. This perspective is beneficial for the successful deployment and improvement of the VOS controller. Such an approach furthers the field of aerospace engineering and proposes a unified system that is not widely available or standardised in the industry.

## 1.2 Individual Project Aim

The project's aim is to aid the VOS of sounding rockets through the development and integration of a software-firmware system. This system incorporates the active control of canards and advanced data management tools to support continuous improvement.

## 1.3 Individual Project Objectives

- To complete the firmware development and convert the high level MATLAB Simulink controller into bare metal C code.
- To develop a visualisation and storage tool that aids controller refinement by allowing users to make informed decisions after analysing flight data.
- To integrate the previously defined subsystems into a coherent data pipeline that streamlines the development of the VOS flight controller.

# Chapter 2. Background and Literature Review

## 2.1 Background

The active control module, namely Aptos, utilises four independently actuated servos and fins (canards) situated in the midsection of the sounding rocket. The rocket is vertically stabilised by the fins' deflection's that generate steering moments. Now in its second year of development, the focus has shifted towards an overhaul of the firmware, software, and hardware required to operate the controller. This year's work builds upon the previous year's foundational work [5], [6], during which two launches were conducted without the control activated. This happened due to insufficient testing and hardware reliability concerns. As a result, the work presented in this report aims to streamline the development process of the controller and enhance its safety.

The concept of data pipeline, in computing, refers to a structured series of nodes, where the output of one node is the input of the next [7]. Data pipelines are designed to improve the flow of data from the source to the destination by automating the process and thereby reducing the requirement for manual involvement. Data pipelines can come in two different forms: Extract-Load-Transform (ELT) or Extract-Transform-Load (ETL) [8]. In this context, as illustrated in Figure 2.1, an ELT system was developed to use the computational resources available on the ground rather than processing data during flight. Data is extracted from the onboard computer post-flight, including atmospheric readings and controller metrics, which are then captured and stored locally on a NOT-AND (NAND) Flash memory unit. After the extraction step, data is loaded on a centralised database from where it can be visualised and postprocessed. To improve the controller further, data can be transformed in a format that is compatible with the input to the MATLAB/Simulink controller simulations. By doing this, the gain tuning can be performed using real-flight data.

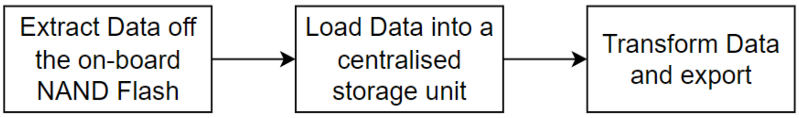



Figure 2.1 Data Pipeline Overview

## 2.2 Literature Review

In rocketry applications, there is a variety of technologies employed for data pipelines across teams and projects. An overview was conducted to analyse how individual teams have selected methodologies and components in their data architectures. This analysis creates a broader understanding of the existing solutions within the field of aerospace engineering, specifically low cost sounding rocketry.

In sounding rocket projects, Arduinos and Teensy are utilised frequently as the flight computer processing unit. A flight computer processing unit is a device that controls the aerospace vehicles, processing data from onboard sensors. These pre-made boards contain all of the circuitry needed for the processor unit and can be paired with premade breakout sensor boards. The use of these systems has been identified in various projects such as the Helen project [9] and the Gryphon I rocket launched by the Leeds University Rocketry Association (LURA) [10]. These boards are favoured for their ease of prototyping, although they often face limitations in flexibility due to predefined libraries and have high costs. Additionally, many groups, such as Ohio's University Rocketry team [11], avoid the use of their own flight hardware and rely on the readings from Commercial-Off-The-Shelf (COTS) flight computers such as the Altus Metrum Series [12], restricting their capabilities further.

For more complex applications, other rocketry teams have adopted more powerful microcontrollers such as the NXP chips, GD32 and ARM-based platforms like the STM32, such as [13] and [14], which required more advanced C programming. These alternatives provide greater flexibility at the cost of increased complexity. Despite the complexity, a lower level understanding of the system helps with debugging. For example, the launch vehicle TEXUS/MAXUS [15] integrated five different on-board experiments that had a custom built data collection system.

In the context of data storage for sounding rocketry teams, there is no standardised database system in place, nor are there centralised records of sounding rocket launches at the United Kingdom (UK) national level. The UK Rocketry Association (UKRA) is recognised as the primary information source for rocketry in the UK. Although there has been an initiative to establish a database for amateur rocketry teams [16], the necessary infrastructure is yet to be implemented. The absence of a unified system has shifted the focus of the review towards general purpose, lightweight and intuitive database platforms. Database options are detailed in Section 4.2.

In terms of rocket flight visualisation, there seems to be no publicly available dashboard technology specifically developed by university rocketry teams. However, individuals and independent groups have developed dashboards by analysing flight data from commercial aerospace companies such as SpaceX [17], [18]. These dashboards contain widgets that display general information about the launch vehicle and some telemetry information about the flight stages timings. Additionally, smaller groups have released dashboards tailored for real-time testing of the sensors on flight hardware [19]. These platforms enable users to connect physical boards directly to a device, extract sensor information and display readings via the web interface dashboard.

# Chapter 3. Firmware Development

## 3.1 Introduction

Firmware is specialised software that is embedded in the non-volatile memory of a hardware device. The hardware platform used is a custom Printed Circuit Board (PCB) that is controlled by a STM32L4R5ZI-P microcontroller (MCU). An STM32 refers to a family of 32-bit MCUs integrated circuits by STMicroelectronics. The peripherals, any external component connected to the MCU, and internals, any registers that are directly inside of the processor unit, were set manually using custom C drivers and setup files. The setup process is described in the subsequent sections and the codebase is available publicly on GitHub [20].

C has emerged as the most appropriate programming language, as it is versatile, performant and portable. A custom bare metal system was developed, where firmware operates directly on hardware without an intermediate operating system (OS). This setup allows for more control over hardware resources, which is ideal in real-time applications, such as a flight computer that runs on an STM32 embedded platform.

## 3.2 Firmware Setup

The firmware was developed inside the Visual Studio Code Integrated Development Environment (VS Code IDE), a tool that can support C code and direct interaction with hardware for debugging purposes through the inspection of memory addresses.

In the context of bare metal development, a series of configurations are needed for the compilation of the firmware on to the target MCU [21]. The high level steps include the setup of memory and registers addresses, the configuration of the interrupt vector table for error handling and the creation of startup code that initialises the memory stack. Additionally, a linker script is required to define the memory layout of the application. Internal configurations such as General Purpose Input/Output (GPIO), system ticks for timekeeping or Universal Asynchronous Receiver/Transmitter (UART) for serial communication are defined. Furthermore, to facilitate debugging and output, print statements are redirected to UART. Appendix D should be checked for a more detailed explanation of the firmware setup.

## 3.3 Flashing Methodology

The hardware setup involves powering the board either through a 7.4V battery or a Universal Serial Bus (USB) connection. A Nucleo-144 board, which incorporates an ST-LINK/V2 in-circuit debugger/programmer, is employed to upload the compiled code (flashing). Flashing involves writing the compiled code to the non-volatile memory of the MCU, which allows the program to be stored permanently, even when the device is turned off or restarted. The connection between the flight computer and the Nucleo board is established via a 4-pin Serial Wire Debug (SWD) header. Since the ST-Link interface does not support output display from the MCU, an additional serial connection is needed. The UART1 pins are exposed on the PCB and connected to a serial interface linked to the computer via USB. Data output is monitored through a PuTTY terminal session which facilitates the debugging of the programmed firmware. The hardware setup can be visualised in Figure 3.1.

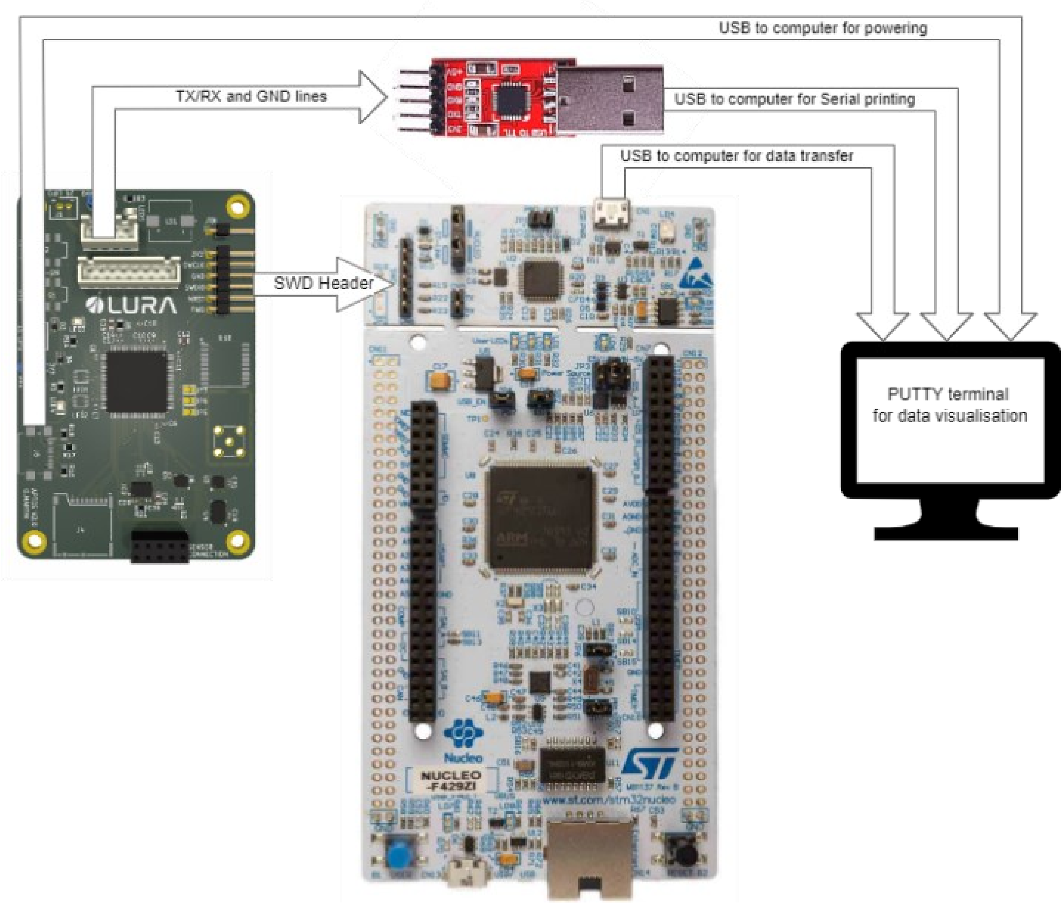



Figure 3.1 Flashing procedure for the custom Aptos PCB via a Nucleo-144

A procedure was put into place to flash code on the flight computer MCU. Firstly, the development environment was configured as described in the Appendix B. Then, the firmware was compiled into executable code by navigating to the code repository in a terminal and running the make flash command.

### 3.4 Firmware Development

The firmware development involved a collaborative effort from various team members, but the following sections cover the author's main areas of focus. The development cycle was completed through firmware implementation, debugging and testing.

A simplified version of the general firmware loop can be viewed in Figure 3.2. For a detailed view, refer to Appendix A. The code configured the STM32 MCU and initialised the communications with onboard sensors using Serial Peripheral Interface (SPI) and UART communication. This included the initialisation of drivers for the barometer sensor, accelerometer, Inertia Measurement Unit (IMU), and the NAND Flash memory.

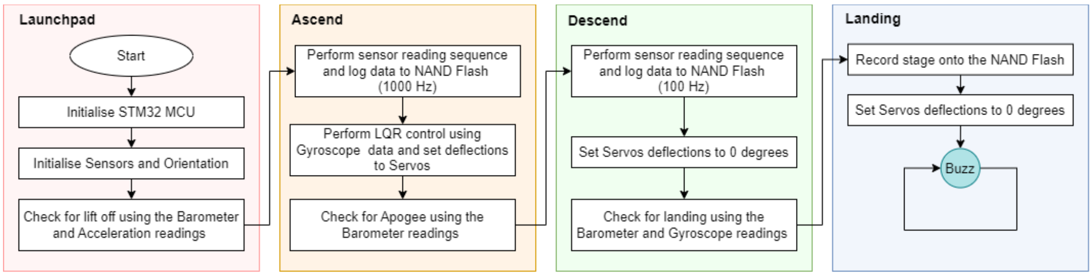



Figure 3.2 Simplified Firmware Flow Diagram (extensive diagram in Appendix A)

The flight computer captured sensor readings at frequencies that varied according to different flight phases, as listed above. During the ascend, the system recorded at a high frequency of up to 1000 Hz to ensure a comprehensive capture of the rocket's performance under maximum dynamic stress and rapid environmental changes, which are most pronounced during this phase. For the descent and landing phases, where changes are more gradual, the recording frequency was reduced to 100 Hz, optimising data storage without compromising the quality of the information gathered.

Data from sensors was stored in a circular buffer, designed to hold up to 50 readings, which helped to reduce noise by calculating median values and applying sensor fusion techniques for more accurate state determination. The custom-developed firmware used the buffer to record data at the moment of take-off. In contrast, most COTS [12] systems commence recording post take-off, thus missing several initial readings. The system was designed to capture the early stages of flight.

Custom functions were implemented to detect lift-off through altitude offsets and acceleration triggers, to calculate vertical velocity from pressure, and identify landing by low gyroscope standard deviation and predefined ground pressure threshold levels. The use of multiple sensor readings for a single flight stage transition ensured that the system could respond appropriately to dynamic conditions throughout the flight.

The existing LQR (Linear-Quadratic Regulator) controller and servo mechanisms were integrated to adjust the vehicle's flight controls based on processed sensor data. Data from sensors and control outputs were compiled into a structured format (FrameArray), timestamped, and logged into NAND flash storage for retrieval and analysis.

The control algorithm, originally developed in MATLAB and Simulink, was translated into C and embedded onto the firmware. The LQR sourced from the previous year controller [5] and firmware [6], were used as guidance. Further steps were taken to improve the controller's execution speed, by removing unnecessary loops, replacing memory draining variables with pass-by-reference pointers, unrolling loops to process multiple values simultaneously. The primary sensor for the LQR, the gyroscope, was initialized at various rates to determine the system's minimum operational frequency. Through trial and error, it was found out that the rates would have a stable output above 100 Hz. Detailed explanations of the controller logic can be found in Appendix C.

Data from the IMU sensor, which includes a three-axis gyroscope and accelerometer, determines the orientation of the launch vehicle. Raw gyroscope data, expressed as Euler angles (roll, pitch, and yaw), risks gimbal lock—a condition causing loss of one degree of freedom. To avoid the this, gyroscope data was converted into Quaternions, represented as four scalar values: qw (the real part) and qx, qy, qz (the imaginary part), [22]. The vehicle orientation was updated in quaternion format. The state is then converted back into Euler angles as input into controller. This conversion is needed because the controller is designed around Euler angles. Figure 3.3 was created to aid the visualisation of the canards expected deflection when motion is applied.

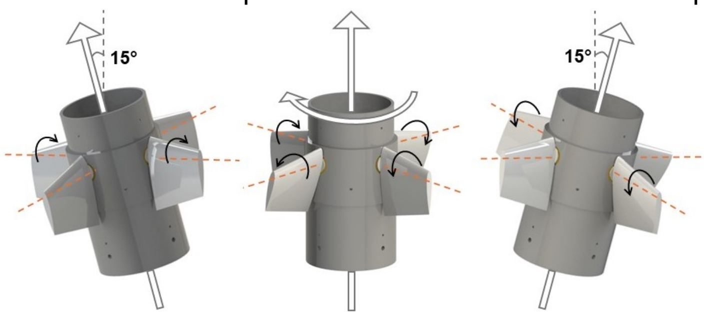



Figure 3.3 Canard Expected Deflection during yaw (left), roll (centre) and pitch (right)

To correctly determine servo deflections from the controller, the gyroscope data must be mapped to their corresponding gains. Due to an alignment discrepancy between the IMU output and the controller's expected input, an axis conversion was implemented, as outlined in Figure 3.4. The controller was configured for a left-hand coordinate system, contrasting with the right-hand coordinate data output from the IMU gyroscope. Moreover, due to the vertical orientation of the board, the roll and pitch axis were reverted.

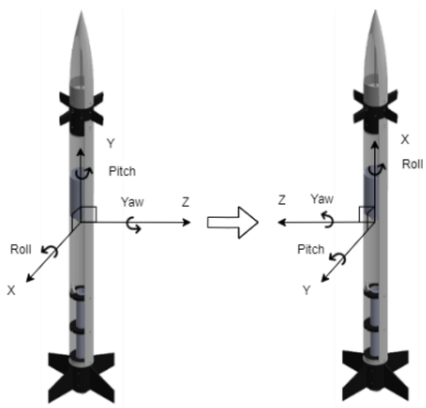



Figure 3.4 Coordinate System before correction (left) and after correction (right)

## 3.5 Firmware Testing

Each sensor custom driver functionality was evaluated through a unit testing procedure, where individual drivers were isolated to retrieve data. For more advanced drivers, such as those handling orientation, testing was conducted with a mobile phone application named Sensor Logger, which calculates the phone's position using Euler and Quaternions [23]. To validate the conversion process, the board was physically attached to the mobile phone and moved along the roll, pitch, and yaw axes, as shown in the Figure 3.5. The Quaternions calculated using the Aptos firmware closely follow the readings from the mobile app, confirming the accuracy of the orientation.

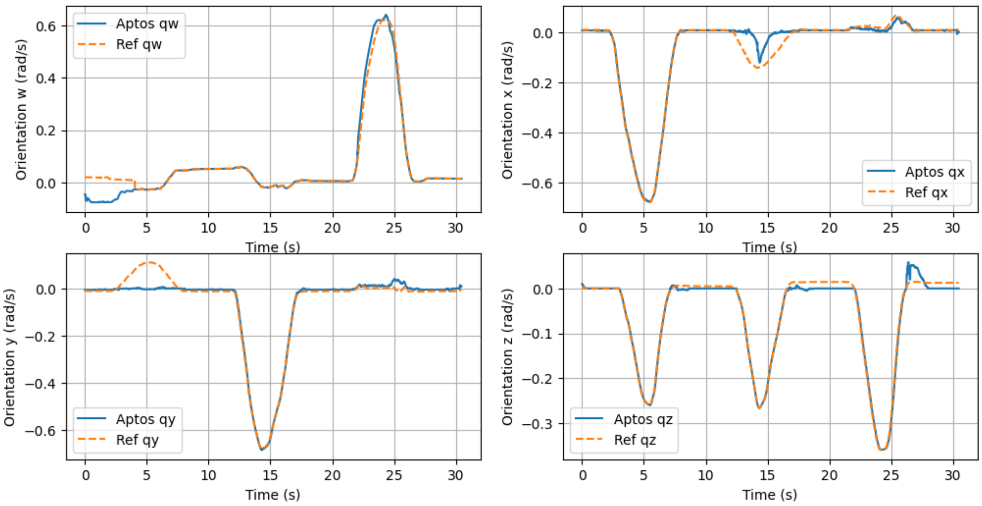



Figure 3.5 Comparison between Sensor Logger Quaternions and flight computer Quaternions

The main loop firmware testing involved placing the flight computer inside a vacuum chamber to simulate flight conditions. The chamber's air pressure was reduced to -0.6bar at the highest pump rate to emulate the atmospheric conditions encountered during flight. Despite the limited pump rates, the results confirmed that the barometer's calculations were accurate to determine the transitions between flight stages.

The flight test for the Aptos module took place on April 14, 2024, during which the system was successfully launched with active control enabled. The board correctly transitioned through the flight stages, and notable oscillations were observed, which were attributed to the control's corrective actions. However, the test revealed a flaw in the NAND flash routine, as servo four data was missing. This happened because the memory address of servo four was overwritten, by mistake, by the bits used for data correction. Additionally, while the servo outputs were intentionally limited to ±15 degrees for safety reasons, the data logged was the capped value rather than the actual angle produced by the controller.

Figure 3.6 Aptos Flight

# Chapter 4. Data Processing and Storage

## 4.1 Introduction

The subsequent phase in the pipeline evolves the storage of the collected data. A database serves as a structured platform for storing, retrieving, and managing data, enabling access and manipulation of flight information. The aim is to create a centralised flight record system that will serve as a long-term repository for flight data.

## 4.2 Data Storage Comparison

The database requirements focus on collection, storage, retrieval, accessibility, and integration [24]. The database must accommodate numerical, text, and time data types, all within a modular framework to facilitate future expansion. For data retrieval, the system requires quick search capabilities, as it is meant to manage multiple concurrent queries when flight data is requested by users. Various database platforms were evaluated such as MySQL, PostgreSQL, which offers robust security [25], MongoDB, which allows for flexible data structures, and InfluxDB, which specialises in time series data [26]. MySQL stands out for its widespread adoption, high storage capacity, and intuitive interface. The ease of setting up and managing MySQL, coupled with its familiar relational database environment, swayed the decision in its favour.

## 4.3 Data Storage Architecture

A local MySQL instance named "aptosdb" was created, along with its structure, designed to organise information into subject-based tables. The database operated on a local system, meaning it stores and manages data on the device where it is installed.

Appendix E outlines the database structure, which mirrored the master structure used in the firmware for managing data on the NAND Flash. In MySQL, a table is a structured format to store data in rows and columns, where each column holds a type and each row corresponds to a record. The database features three tables linked by a one-to-many relationship, meaning a single record from one table (primary table, "flight") can be associated with multiple records in the other tables ("flight_data" and "control_command") via a unique key. The primary table, "flight", stored general information. Meanwhile, "flight_data" included the sensor readings and "control_command" recorded controller information, specifically servo deflection angles. The two tables include timestamps and default values for all entries to avoid errors with potential undefined raw entries. MySQL provides an interactive terminal that was used to document and prepared the scripts needed in the following phase.

# Chapter 5. Data visualisation

## 5.1 Introduction

A web-based application, the user-facing component of the system, was developed to facilitate intuitive data visualisation from the databases. This tool promotes more informed decision-makings and facilitates the identification of trends and anomalies within the dataset.

For this component of the pipeline, a new web application was developed, called “LURA Dash”. The following sections detail the backend and frontend components. The backend is tasked with the application’s logic and data processing, while the frontend focuses on user interaction and visual integration.

## 5.2 Backend Framework Selection

The following requirements were selected: simplicity, to accommodate members with less web-based experience; flexibility, to keep the tool computationally lightweight without heavy dependencies; and extensibility, to allow for future features such as user authentication. A Python-based framework was selected to leverage its widespread popularity and ease of integration with MySQL databases. The ideal framework should have a solid foundation of user guides and resources to address common issues.

Flask, a Python based web framework, was chosen for its Representational State Transfer (RESTful) request handling, built-in development server, and integrated debugger that aids error correction [27]. Compared to alternative frameworks — Django's complexity, CherryPy's inadequate documentation, and Bottle's limited community [28] — Flask stands out as the most pragmatic choice. Its strong community support and comprehensive documentation ensure a smooth development process, making it an accessible and powerful tool for developers of all skill levels.

## 5.3 Framework Development

Flask served as the backbone of “LURA Dash”. It facilitated the creation and management of RESTful API (Application Program Interface) endpoints. A RESTful API is an architectural style for an API that uses Hypertext Transfer Protocol (HTTP) requests to access and use data [29]. These endpoints were defined to handle specific functionalities such as data retrieval, data storage, and dynamic content delivery. Each endpoint was mapped to a Python function, making it straightforward to implement logic that interacted directly with the backend database. The API was designed with a clear structure where each route was associated with HTTP methods that defined client interactions with the server. For instance, GET requests fetched data and POST requests submitted new data. Table 5.1 lists the endpoints that can be accessed.

Table 5.1 Web Endpoints

| Endpoint | Method | Description | Response |
|---|---|---|---|
| '/get-flights' | GET | Returns a list of all flights from the database | JSON with a list of flights |
| '/get-flight-data' | GET | Returns detailed flight data based on the flight ID | JSON with flight details and associated flight data |
| '/get-db-tables' | GET | Lists all database tables | JSON with a list of database tables |
| '/get-db-columns' | GET | Lists all columns for a specified table. | JSON with column details of a specified table. |
| '/get-db-column-data' | GET | Retrieves data for a specified column in a specified table | JSON with data from the specified column |
| '/get-db-table-data' | GET | Retrieves all or filtered data from a specified table | JSON with data from the specified table |
| '/upload' | POST | Stores uploaded flight data into the database | Confirmation message of data storage |
| '/flight-data' | GET POST | Serves the main page of the web application | HTML of the main page |
| '/database' | GET POST | Serves the database page of the web app | HTML of the database page |
| '/add-data' | GET POST | Serves the data ingestion page of the web app | HTML of the data ingestion page |
| '/export-data' | GET POST | Serves the data extraction page and handles data export | HTML of the data extraction page of exported CSV file |

To manage database interactions, SQL Alchemy was used as the Object-Relational Mapping (ORM) tool. The ORM facilitates the communication between the application and the database by using high-level entities such as classes, which mirror the tables in the database [27]. Models in SQL Alchemy defined the structure of the database, which simplified tasks like querying the database and manipulating data entries.

For the frontend, Vanilla JavaScript was used to make the application lightweight. This choice avoided the overhead associated with larger frameworks. JavaScript interfaced with the Flask backend via AJAX calls, fetching and posting data asynchronously to provide an uninterrupted user experience without the need for page reloads.

### 5.4 LURA Dash Features

"LURA Dash" offered multiple pages that enabled users to interact with data in various formats. The main tab, illustrated in Figure 5.1, allowed users to select a flight and display it on the screen. The interface featured widgets including an altitude versus time graph, vertical velocity and acceleration, and a flight path representation based on sensor fusion, along with other statistics. The "Run from Beginning" button played an entire flight. Users could stop at any point to examine a particular moment in time.


Figure 5.1 Main LURA Dash tab, data is displayed from the active controlled test flight

LURA Dash included tabs for easy handling of CSV-formatted data from the flight computer. Users could upload the flight data into the database using the tool shown on the left. Once visualised and validated, any flight data could be formatted in the appropriate form for the input of the gain tunning in MATLAB using the tool on the right.

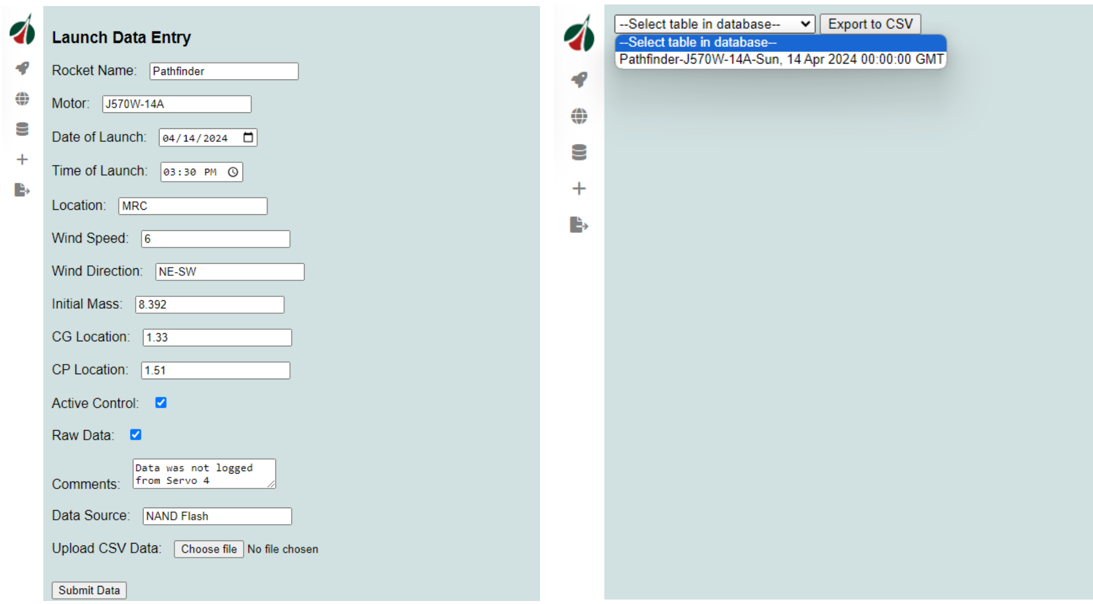



Figure 5.2 Pages on LURA Dash: the import of a new flight in the database (left) and the export of a flight into a MATLAB controller input format (right)

The raw flight data did not match the input format used for the controller gain tuning. The following parameters—altitude, vertical velocity, mass, longitudinal moment of inertia, rotational moment of inertia, centre of gravity location and Mach number—were derived from the raw values as the equations shown in Appendix F. After conversion, the data was compiled into a CSV file. This file could then be integrated into MATLAB, to enable the tuning of the controller with real-world data—a significant enhancement from the previous reliance on simulated data alone.

# Chapter 6. Pipeline Integration

## 6.1 Introduction

The final phase of the project was marked by the integration of all components into a cohesive data pipeline. This process was used to validate the system's performance against the anticipated outcome from the MATLAB simulation and maintain compatibility between stages. The architecture is illustrated in Figure 6.1, which demonstrates the data flow, starting from collection and storage, followed by its conversion in various formats, which enables transition among distinct subsystems.

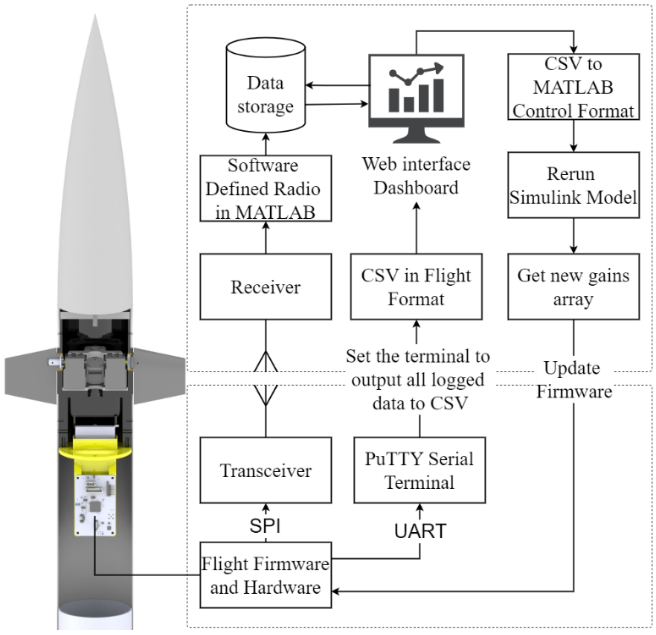



Figure 6.1 System Integration

The effectiveness of the integration was tested following a flight campaign. Data was extracted from the flight computer using PuTTY's serial terminal interface and then converted to CSV format. The dashboard required users to enter details such as the rocket's name, engine type, date, time and wind conditions. Following data visualisation, the information was then exported in a modified CSV format suitable for recalibrating the MATLAB model's gains. The pipeline eliminated the need for any custom scripts or additional steps for data conversion.

## 6.2 Pipeline Discussion and Results

### 6.2.1 Pipeline Throughput

In the post-flight evaluation, the data pipeline's throughput was quantified at approximately 0.622 MB per minute, which includes the duration of data retrieval from the flight hardware to its eventual ingestion into the database. The primary constraint was the NAND Flash's read speed, which currently outputs approximately 88 readings per second. At 100Hz, the total test flight yielded 6557 readings, which translates to 74.098 seconds dedicated solely to data extraction. An additional source in processing time is attributed to the manual transfer of the CSV file from the flight hardware. It was deemed appropriate for the following firmware iteration to have a more optimised reading routine for the NAND Flash to reduce the time footprint of the data extraction process and, by extension, the overall efficiency of the pipeline.

When the flight results were ingested into the database via the dashboard, the system required 6.227 seconds. To assess scalability, the system was subjected to a simulated data increase by a factor of ten, 65570 entries corresponding to about 12.418 hours of flight. The findings revealed a linear performance, with only a nominal increase in the database ingestion period to 58.263 seconds.

### 6.2.2 Storage Capabilities

During the test launch, the data acquisition system used 2416 Kb of storage, with the data collection process spanning 74.092 seconds. Given the small storage requirements, it is anticipated that the database can accommodate data from multiple future flights, even with substantial increases in data acquisition rates. For instance, elevating the main loop frequency from 1000Hz to 3000Hz, or extended flight durations due to factors such as wind drift or premature deployment of the main parachute, would likely not inflate the data size beyond 20 Mb for each launch.

### 6.2.3 Cloud Hosting Implications

This projection aligns with the planned transition to cloud-based storage solutions. Utilising a service such as Cloud SQL, it is estimated that the cost would remain economical at approximately $2.57 per month, as indicated by current pricing models [18]. This calculation is based on a lightweight 50 Gb database instance, operational 24 hours a day, tailored to the team’s needs that do not require constant database access. As an alternative, leveraging a custom server setup with a Raspberry Pi, another small single-board computer, could offer a cost-free solution while still fulfilling the project's data hosting requirements.

### 6.2.4 Adaptability

Additionally, the pipeline's architecture is adaptable. Modifications to the firmware, provided they maintain standard readings—barometric pressure, acceleration, IMU, temperature, and GNSS data—do not impact the database or the dashboard interface. Similarly, updates to the control system are accommodated as long as the input data derived from flight simulations are consistent. As a result, the core functionality of the architecture remains unaffected by changes in hardware or software. The pipeline is inheritably flexible and can evolve with the project’s requirements.

# Chapter 7. Conclusion and Future Work

## 7.1 Achievements

The project met all its objectives, contributing to the development of an active stabilisation system for sounding rockets. Firstly, the flight firmware that supports an active controller was developed in C, bare metal. The setup included the main routine, helper functions, and controller logic initially created in MATLAB. Methods for storing and visualising flight data were also developed and tested. These components were successfully integrated into a data pipeline that streamlines the development and refinement of a sounding rocket VOS stabilisation system.

## 7.2 Conclusion

This report details the design and implementation of a data pipeline integral to a rocket flight controller application, which bridges firmware and software components. This system handled the data demands associated with a rocket launch and multiple additional tests, achieving a throughput of approximately 0.622 MB per minute while maintaining data integrity.

A significant feature of the project was the incorporation of real-flight data into the MATLAB-based controller, which aided the analytical capabilities during post-flight analyses. This allowed for more modifications, as there was a better understanding of the dataset and a reassurance it is correct as it was real life.

The successful implementation of the data pipelines not only fulfilled the initial project goals but also laid a solid groundwork for future work in aerospace control systems. The system was designed to require minimal user intervention, thus optimising the efficiency of data flow across various components of the pipeline. This is beneficial for the improvement of the VOS control of sounding rockets equipped with canards.

## 7.3 Future Work

For future improvements, several steps are recommended to improve the pipeline:

- Data Throughput: A more efficient routine for reading NAND Flash could decrease data extraction times and increase throughput.
- Dashboard Functionality: New widgets could be added to the dashboard to show how the canards respond to the orientation of the rocket. This would allow for better control and understanding of their impact on stabilisation.
- Cloud Integration: Moving both the database and the web application to the cloud would allow team members to access data from any location, not just locally.

## References


[1] T. Noga, M. Michałów, and G. Ptasiński, “Comparison of dispersion calculation methods for sounding rockets,” *Journal of Space Safety Engineering*, vol. 8, Sep. 2021, doi: 10.1016/j.jsse.2021.08.006.

[2] F. Sève, S. Theodoulis, P. Wernert, M. Zasadzinski, and M. Boutayeb, “Flight Dynamics Modeling of Dual-Spin Guided Projectiles,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 4, pp. 1625–1641, Aug. 2017, doi: 10.1109/TAES.2017.2667820.

[3] S. Chang, Z. Wang, and T. Liu, “Analysis of Spin-Rate Property for Dual-Spin-Stabilized Projectiles with Canards,” *Journal of Spacecraft and Rockets*, vol. 51, no. 3, pp. 958–966, 2014, doi: 10.2514/1.A32830.

[4] S. Chang, D. Li, and W. Wei, “Swerve Solution for Spin-Stabilized Projectiles with Canards: A Revisit,” *Journal of Spacecraft and Rockets*, vol. 58, no. 5, pp. 1352–1360, 2021, doi: 10.2514/1.A34964.

[5] T. Youds, “Development Of An Active Control System For A Canard-Controlled Sounding Rocket,” *Leeds: University of Leeds*, vol. Masters Thesis, 2023.

[6] B. Cradock, “Design and Development of an Embedded Flight Computer for a Canard-Controlled Sounding Rocket,” *Leeds: University of Leeds*, vol. Masters Thesis, 2023.

[7] M. W. V. Alstyne, G. G. Parker, and S. P. Choudary, “Pipelines, Platforms, and the New Rules of Strategy”.

[8] A. Raj, J. Bosch, H. H. Olsson, and T. J. Wang, “Modelling Data Pipelines,” in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2020, pp. 13–20. doi: 10.1109/SEAA51224.2020.00014.

[9] V. Nair *et al.*, “Team 116 Project Technical Report for the 2019 IREC”.

[10] “Gryphon I,” LURA. Accessed: Apr. 20, 2024. [Online]. Available: https://leedsrocketry.co.uk/projects/gryphon-1/

[11] T. Moleski, T. Berger, J. Browne, A. Scott, B. Hesson, and D. Denner, “Project The Big One Team 23 Technical Report for the 2018 IREC”.

[12] “Altus Metrum.” Accessed: Apr. 20, 2024. [Online]. Available: https://altusmetrum.org/index.html

[13] N. Christopher *et al.*, “Shark of the Sky Hybrid Rocket”.

[14] “Advanced Control Team – Delft Aerospace Rocket Engineering.” Accessed: Apr. 20, 2024. [Online]. Available: https://dare.tudelft.nl/projects/act/

[15] J. Matevska, E. Noack, M. Reinhold, and E. Diekmann, *Decentralised Avionics and Software Architecture for Sounding Rocket Missions*. 2020. doi: 10.18420/SE2020_66.

[16] “Team Project Support | UKRA - United Kingdom Rocketry Association.” Accessed: Apr. 20, 2024. [Online]. Available: http://www.ukra.org.uk/tps

[17] “r-spacex/SpaceX-API.” r/SpaceX, Apr. 28, 2024. Accessed: Apr. 28, 2024. [Online]. Available: https://github.com/r-spacex/SpaceX-API

[18] “SpaceX Dashboard.” Accessed: Apr. 28, 2024. [Online]. Available: https://tdunn891.github.io/spacex-dashboard/

[19] “COSMIC AEROSPACE TOWER,” Cosmic Aerospace Technologies. Accessed: Dec. 28, 2023. [Online]. Available: https://cosmicaero.space/tower

[20] “General · AlexandraPosta/aptos,” GitHub. Accessed: Apr. 28, 2024. [Online]. Available: https://github.com/AlexandraPosta/aptos

[21] *Beginning STM32*. Accessed: Apr. 20, 2024. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-3624-6

[22] D. M. Henderson, “Shuttle Program. Euler angles, quaternions, and transformation matrices working relationships,” *Mission Planning and Analysis Division*, 1977, [Online]. Available: https://ntrs.nasa.gov/api/citations/19770024290/downloads/19770024290.pdf

[23] “Sensor Logger,” Kelvin Choi. Accessed: Apr. 28, 2024. [Online]. Available: https://www.tszheichoi.com/sensorlogger

[24] A. R. Munappy, J. Bosch, and H. H. Olsson, “Data Pipeline Management in Practice: Challenges and Opportunities,” in *Product-Focused Software Process Improvement*, M. Morisio, M. Torchiano, and A. Jedlitschka, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 168–184. doi: 10.1007/978-3-030-64148-1_11.

[25] M. Abourezq and A. Idrissi, “Database-as-a-Service for Big Data: An Overview,” *ijacsa*, vol. 7, no. 1, 2016, doi: 10.14569/IJACSA.2016.070124.

[26] A. Siddiqa, A. Karim, and A. Gani, “Big data storage technologies: a survey,” *Frontiers Inf Technol Electronic Eng*, vol. 18, no. 8, pp. 1040–1070, Aug. 2017, doi: 10.1631/FITEE.1500441.

[27] U. Patkar, P. Singh, H. Panse, S. Bhavsar, and C. Pandey, “PYTHON FOR WEB DEVELOPMENT,” *IJCSMC*, vol. 11, no. 4, pp. 36–48, Apr. 2022, doi: 10.47760/ijcsmc.2022.v11i04.006.

[28] A. L. Sayeth Saabith, M. M. M. Fareez, and T. Vinothraj, “Python Current Trend Applications - An Overview,” *IJAERD*, vol. 6, no. 10, pp. 6–12, 2019.

[29] "What is a REST API? | IBM." Accessed: Apr. 28, 2024. [Online]. Available: https://www.ibm.com/topics/rest-apis

# Appendix A – Firmware Flowchart

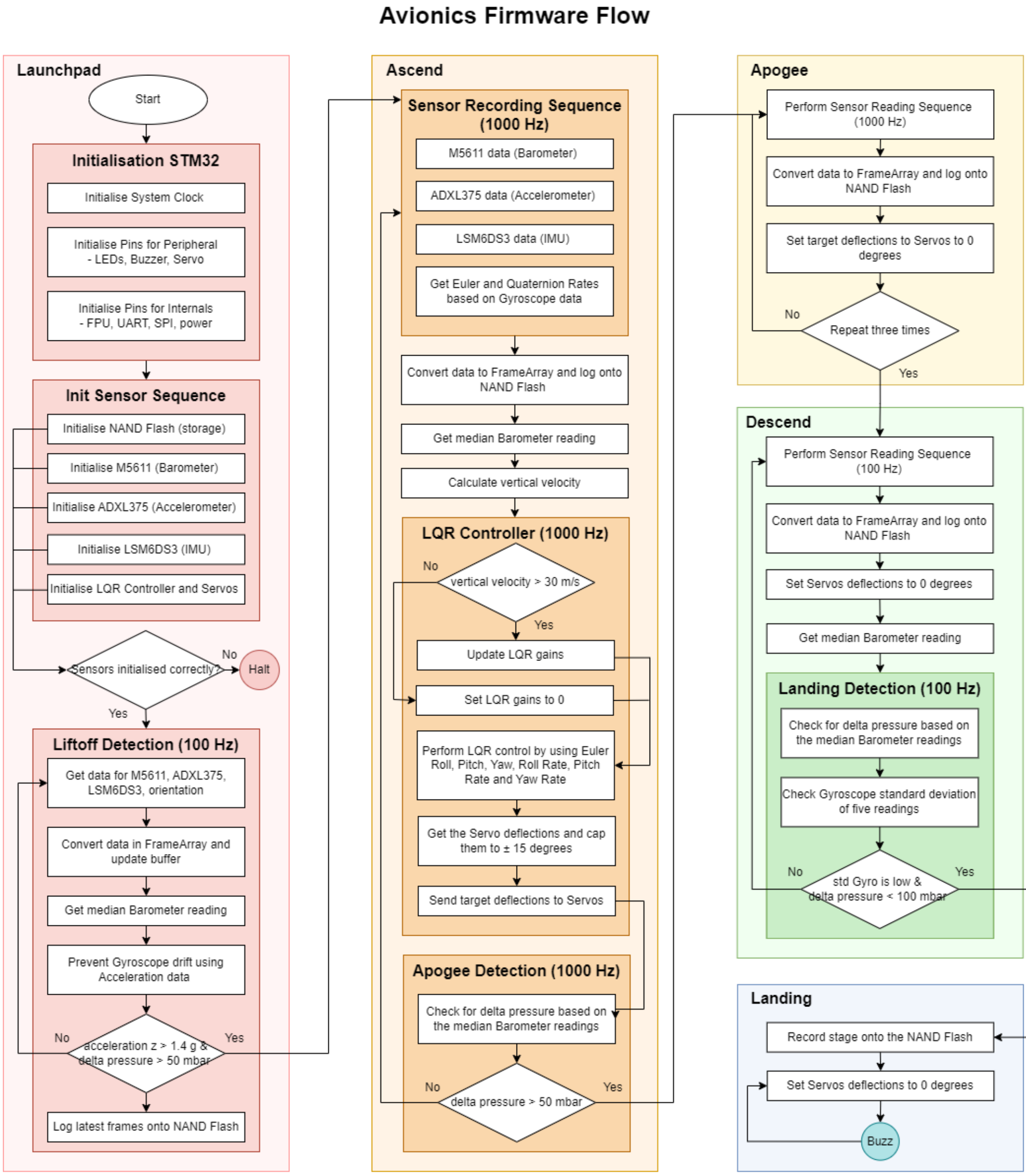



Figure A.1 Detailed Firmware Flow Diagram

## Appendix B – Dashboard GitHub README file

**Overview**

LURA Dash is a new web interface tool designed by Leeds Universiy Rocketry Association for visualisation of flight data. It offers multiple pages that enable users to interact with data in various formats.

The main page features widgets including an altitude versus time graph, vertical velocity, vertical acceleration, and a flight path representation based on sensor fusion, along with other statistics. The "Run from Beginning" button plays an entire flight. Users can stop at any point to examine a particular moment in time. LURA Dash includes tabs for easy handling of CSV-formatted data from the Aptos flight computer. Once visualised and validated, any flight data can be formatted in the appropriate form for the input of our custom controller in MATLAB.

**Features**

- load flight off the flight computer
- visualise the final outcome of the flight
- play the entire rocket flight and pause as needed
- visualise the data straight from the database; apply filters as needed
- import CSV file with new flight
- export to CSV that is compatible for the comtroller tuning in MATLAB

**Structure**

The repository is structured as follows:

```
web_server
|──README.md
|
|──database
|   ├──commands.py          # MYSQL Database queries
|   ├──connect.py           # MYSQL Database configuration
|   ├──fakedata.py          # Fake data generator for the database
|   └──models.py            # MYSQL Database tables definition
|
|──static
|   ├──3d                   # 3D models using in the frontend
|   ├──assets               # Images using in the frontend
|   ├──css                  # The main css file
|   └──js
|     ├──add-data.js        # Contains functions used to ingest new data in the db
|     ├──custom-card.js   # Custon widgets class
|     ├──custom-data.js   # Custon flight data class
|     ├──database.js        # Database interacion from frontend
|     ├──export.js          # Export flight into csv for MATLAB input
```

```
|     ├──flight.js                  # Functions used to display flight data on the
dashboard.
|     ├──load-flight-data.js  # Code for the worker that loads the flight data.
|     └──telemetry.js             # Display telemetry data on the dashboard.
|
|──templates
|    ├──add-data.html         # HTML page that allows user to input flight data
|    ├──base.html                # HTML template for the all the rest of the pages
|    ├──database.html         # HTML page that allows user to filter the database
|    ├──export-data.html     # HTML page to export data to Simulink input
|    ├──flight-data.html          # main HTML page for flight data visualisation
|    ├──flight.html               # HTML template for the flight related pages
|    └──telemetry-data.html # HTML page for the telemetry connection
|
──app                      # Entry point for the application
```

**Requirements**

- python 3.6+
- flask
- flask mysql connector
- flask SQLAlchemy

**To set up the webserver**

- install python 3.6+
- setup virtual environment using pip install virtualenv
- create environment using virtualenv env
- activate .\env\Scripts\activate
- pip install flask
- pip install flask-cors
- pip install sqlalchemy
- pip install Flask-SQLAlchemy
- pip install mysql-connector-python

**Trobleshoot**

When debugging the flask app, you might not hit the breakpoint using Visual Studio. Make sure toset the "args" from launch.json to --no-debugger, --no-reload go to app.py and run the app with debug set to False.

## Appendix C – Controller Transition from MATLAB to C

The header file orientation_utils.h provides the necessary definitions and function prototypes to convert raw gyroscope data into quaternion and Euler angle formats. This file defines types for Euler angles and quaternions, used for orientation representation in 3D space, and includes an orientation_data structure that maintains the current and previous states of the types. It also declares functions to initialise, update, and manipulate orientation data based on inputs from the LSM6DS3 gyroscope sensor.

```
/*
  Leeds University Rocketry Organisation - LURA
  Author Name: Alexandra Posta
  Description: Header file to transform gyroscope raw data to Quateniun
and Euler
*/

#ifndef ORIENTATION_UTILS_H
#define ORIENTATION_UTILS_H

#include "drivers/LSM6DS3_driver.h"
#include <math.h>

#define M_PI_F 3.14159265358979323846f

typedef struct Euler {
    float roll;
    float pitch;
    float yaw;
} Euler;

typedef struct Quaternion {
    float w;
    float x;
    float y;
    float z;
} Quaternion;

typedef struct orientation_data {
    Quaternion current_quaternion;
    Quaternion current_rate_quaternion;
    Euler current_euler;
    Euler current_rate_euler;
    Euler previous_euler;
} orientation_data;
```

```c
/**
   @brief Convert euler angles to quaternion
   @param e Euler angles
   @param q Quaternion
*/
void orientation_quaternion_to_euler(Quaternion* q, Euler* e);

/**
  @brief Initialise the orientation data
  @note Set the orientation_data structure to 0 to initialise memory
*/
void orientation_init(orientation_data* orientation, LSM6DS3_data*
_LSM6DS3_data);

/**
  @brief Update the orientation data based on gyro readings
  @param dt Time difference in microseconds
  @param orientation Orientation data structure
  @param _LSM6DS3_data Gyroscope data
*/
void orientation_update(unsigned int dt, orientation_data* orientation,
LSM6DS3_data* _LSM6DS3_data, bool pad);

/**
  @brief Check if rocket is moving based on acceleration vector
  @param _LSM6DS3_data Gyroscope data
  @param vector Acceleration vector
  @return True if the vector is valid
*/
bool OrientationAccelerationVector(LSM6DS3_data* _LSM6DS3_data, float
vector[]);

/**
  @brief Check if stationary, to correct gyro drift, based on
acceleration vector
  @param _orientation Orientation data structure
  @param accel Acceleration vector
  @param correction Quaternion correction
*/
void OrientationAccelerationQuaternion(orientation_data* _orientation,
float accel[], Quaternion* correction);
#endif /* ORIENTATION_UTILS_H */
```

Figure C.1 Source code for orientation_utils.h

The source file orientation_utils.c, implements functions to transform gyroscope data into quaternion and Euler angle formats. The file includes essential functions for initialising orientation data, updating it based on gyroscope and accelerometer readings, and converting orientation represented by quaternions into Euler angles. Additionally, the source file handles coordinate system adjustments and gravity correction based on sensor data to maintain accurate orientation tracking despite external disturbances.

```
/*
    Leeds University Rocketry Organisation - LURA
    Author Name: Alexandra Posta
    Description: Source file to transform gyroscope data to quateniun
and euler
*/

#include "orientation_utils.h"

void orientation_quaternion_to_euler(Quaternion* q, Euler* e) {
    // XYZ order
    float qw2 = q->w * q->w;
    float qx2 = q->x * q->x;
    float qy2 = q->y * q->y;
    float qz2 = q->z * q->z;

    // Calculate direction cosine matrix
    float dcm32 = 2 * (q->y * q->z - q->x * q->w);
    float dcm33 = qw2 - qx2 - qy2 + qz2;
    float dcm31 = 2 * (q->x * q->z + q->y * q->w);
    float dcm21 = 2 * (q->x * q->y - q->z * q->w);
    float dcm11 = qw2 + qx2 - qy2 - qz2;

    // Calculate euler angles
    e->roll = (float)atan2(-dcm32, dcm33);
    e->pitch = (float)asin(dcm31);
    e->yaw = (float)atan2(-dcm21, dcm11);
}

void orientation_change_accel_coordinate_system(LSM6DS3_data*
_LSM6DS3_data) {
    int32_t temp_y = _LSM6DS3_data->y_accel;
    _LSM6DS3_data->y_accel = _LSM6DS3_data->z_accel;
    _LSM6DS3_data->z_accel = -temp_y;
}

void orientation_init(orientation_data* orientation, LSM6DS3_data*
_LSM6DS3_data) {
    float accel_vector[4];
```

```
    orientation_change_accel_coordinate_system(_LSM6DS3_data);
    if (OrientationAccelerationVector(_LSM6DS3_data, &accel_vector)) {
//try to get an acceleration vector to use as starting angle
        float pitch_angle_accel =
atan(accel_vector[1]/sqrt((accel_vector[0]*accel_vector[0])+(accel_vect
or[2]*accel_vector[2])));
        float yaw_angle_accel =
atan(accel_vector[0]/sqrt((accel_vector[1]*accel_vector[1])+(accel_vect
or[2]*accel_vector[2])));

        // Calculate initial quaternion components based on the
estimated roll and pitch angles
        float cy = cos(pitch_angle_accel * 0.5f);
        float sy = sin(pitch_angle_accel * 0.5f);
        float cp = cos(yaw_angle_accel * 0.5f);
        float sp = sin(yaw_angle_accel * 0.5f);
        orientation->current_quaternion.w = cp * cy;
        orientation->current_quaternion.x = sy * sp;
        orientation->current_quaternion.y = cp * sy;
        orientation->current_quaternion.z = sp * cy;

        orientation_quaternion_to_euler(&orientation-
>current_quaternion, &orientation->current_euler);
        // Set initial values for previous_euler
        orientation->previous_euler.roll = orientation-
>current_euler.roll;
        orientation->previous_euler.pitch = orientation-
>current_euler.pitch;
        orientation->previous_euler.yaw = orientation-
>current_euler.yaw;
    } else {  //accel wasn't close enough to 1g
        // Set initial values for current_quaternion
        orientation->current_quaternion.w = 1.0;
        orientation->current_quaternion.x = 0.0;
        orientation->current_quaternion.y = 0.0;
        orientation->current_quaternion.z = 0.0;
        // Set initial values for current_euler
        orientation->current_euler.roll = 0.0;
        orientation->current_euler.pitch = 0.0;
        orientation->current_euler.yaw = 0.0;
        // Set initial values for previous_euler
        orientation->previous_euler.roll = 0.0;
        orientation->previous_euler.pitch = 0.0;
        orientation->previous_euler.yaw = 0.0;
    }

    // Set initial values for current_rate_quaternion
    orientation->current_rate_quaternion.w = 0.0;
```

```
    orientation->current_rate_quaternion.x = 0.0;
    orientation->current_rate_quaternion.y = 0.0;
    orientation->current_rate_quaternion.z = 0.0;

    // Set initial values for current_rate_euler
    orientation->current_rate_euler.roll = 0.0;
    orientation->current_rate_euler.pitch = 0.0;
    orientation->current_rate_euler.yaw = 0.0;
}

void orientation_change_coordinate_system(LSM6DS3_data* _LSM6DS3_data)
{
    int32_t temp_x = _LSM6DS3_data->x_rate;
    _LSM6DS3_data->x_rate = _LSM6DS3_data->y_rate;
    _LSM6DS3_data->y_rate = temp_x;
    _LSM6DS3_data->z_rate *= -1;
}

// Update orientation data
// On the sensor     -> X: PITCH, Y: ROLL,  Z:  YAW (right rule)
// On the controller -> X: ROLL,  Y: PITCH, Z: -YAW (left rule)
void orientation_update(unsigned int dt, orientation_data* orientation,
LSM6DS3_data* _LSM6DS3_data, bool pad) {
    // Change orientation data to match the controller coordinate
system
    orientation_change_coordinate_system(_LSM6DS3_data);
    orientation_change_accel_coordinate_system(_LSM6DS3_data);
    float wx = ((float)_LSM6DS3_data->x_rate * M_PI_F / 180.0f) /
1000.0f; // millidegrees/second -> radians/second
    float wy = ((float)_LSM6DS3_data->y_rate * M_PI_F / 180.0f) /
1000.0f;
    float wz = ((float)_LSM6DS3_data->z_rate * M_PI_F / 180.0f) /
1000.0f;

    float qw = orientation->current_quaternion.w;
    float qx = orientation->current_quaternion.x;
    float qy = orientation->current_quaternion.y;
    float qz = orientation->current_quaternion.z;

    // Calculate the derivative of the quaternion
    orientation->current_rate_quaternion.w = 0.5f * (-wx * qx - wy * qy
- wz * qz);
    orientation->current_rate_quaternion.x = 0.5f * ( wx * qw + wz * qy
- wy * qz);
    orientation->current_rate_quaternion.y = 0.5f * ( wy * qw - wz * qx
+ wx * qz);
    orientation->current_rate_quaternion.z = 0.5f * ( wz * qw + wy * qx
- wx * qy);
```

```
    // Update quaternion using the derivative
    orientation->current_quaternion.w += orientation-
>current_rate_quaternion.w * (float)dt * 1e-6f;
    orientation->current_quaternion.x += orientation-
>current_rate_quaternion.x * (float)dt * 1e-6f;
    orientation->current_quaternion.y += orientation-
>current_rate_quaternion.y * (float)dt * 1e-6f;
    orientation->current_quaternion.z += orientation-
>current_rate_quaternion.z * (float)dt 1e-6f;

    float accel_vector[4];
    if(OrientationAccelerationVector(_LSM6DS3_data, &accel_vector) &&
pad){ //try to get an acceleration vector to use as starting angle
        float pitch_angle_accel =
atan(accel_vector[1]/sqrt((accel_vector[0]*accel_vector[0])+(accel_vect
or[2]*accel_vector[2])));
        float yaw_angle_accel =
atan(accel_vector[0]/sqrt((accel_vector[1]*accel_vector[1])+(accel_vect
or[2]*accel_vector[2])));
        // Calculate initial quaternion components based on the
estimated roll and pitch angles
        float cy = cos(pitch_angle_accel * 0.5f);
        float sy = sin(pitch_angle_accel * 0.5f);
        float cp = cos(yaw_angle_accel * 0.5f);
        float sp = sin(yaw_angle_accel * 0.5f);
        orientation->current_quaternion.w = 0.9f * orientation-
>current_quaternion.w + 0.1f * cp * cy;
        orientation->current_quaternion.x = 0.9f * orientation-
>current_quaternion.x + 0.1f * sy * sp;
        orientation->current_quaternion.y = 0.9f * orientation-
>current_quaternion.y + 0.1f * cp * sy;
        orientation->current_quaternion.z = 0.9f * orientation-
>current_quaternion.z + 0.1f * sp * cy;
    }

    // Normalise quaternions
    float norm = sqrtf(orientation->current_quaternion.w * orientation-
>current_quaternion.w +
                       orientation->current_quaternion.x * orientation-
>current_quaternion.x +
                       orientation->current_quaternion.y * orientation-
>current_quaternion.y +
                       orientation->current_quaternion.z * orientation-
>current_quaternion.z);

    // Apply normalisation
    orientation->current_quaternion.w /= norm;
```

```
    orientation->current_quaternion.x /= norm;
    orientation->current_quaternion.y /= norm;
    orientation->current_quaternion.z /= norm;

    // Convert quaternion to euler angles
    orientation->previous_euler = orientation->current_euler;
    orientation_quaternion_to_euler(&orientation->current_quaternion,
&orientation->current_euler);

    // Calculate the derivative of the euler angles
    if ((orientation->current_euler.roll < (-(M_PI_F) + 0.6f)) &&
orientation->previous_euler.roll > (M_PI_F - 0.6f)) {
        orientation->current_rate_euler.roll = (orientation-
>current_euler.roll + 2 * M_PI_F - orientation->previous_euler.roll) /
((float)dt * 1e-6f);
    } else {
        orientation->current_rate_euler.roll = (orientation-
>current_euler.roll - orientation->previous_euler.roll) / ((float)dt *
1e-6f);
    }

    orientation->current_rate_euler.pitch = (orientation-
>current_euler.pitch - orientation->previous_euler.pitch) / ((float)dt*
1e-6f);
    orientation->current_rate_euler.yaw = (orientation-
>current_euler.yaw - orientation->previous_euler.yaw) / ((float)dt *
1e-6f);
}

bool OrientationAccelerationVector(LSM6DS3_data* _LSM6DS3_data, float
vector[]){
    //convert from milli g to g
    vector[0] = _LSM6DS3_data->x_accel/1000.0;
    vector[1] = _LSM6DS3_data->y_accel/1000.0;
    vector[2] = _LSM6DS3_data->z_accel/1000.0;

    //check magnitude (in g)
    float magnitude = sqrtf(vector[0]*vector[0] + vector[1]*vector[1] +
vector[2]*vector[2]);

    //normalise the vector
    vector[0] /= magnitude;
    vector[1] /= magnitude;
    vector[2] /= magnitude;
    vector[3] = magnitude;

    if (magnitude < 0.9 || magnitude > 1.1){   //if not close to 1G
        return false;
```

```
    }
    return true;
}

void OrientationAccelerationQuaternion(orientation_data* _orientation,
float accel_vector[], Quaternion* correction){
    Quaternion q_est = _orientation->current_quaternion;

    // Estimate gravity direction in the world frame using current
orientation estimate
    float gw_x = 2 * (q_est.x * q_est.z - q_est.w * q_est.y);
    float gw_y = 2 * (q_est.w * q_est.x + q_est.y * q_est.z);
    float gw_z = q_est.w * q_est.w - q_est.x * q_est.x - q_est.y *
q_est.y + q_est.z * q_est.z;

    // Calculate error between estimated gravity direction and
accelerometer readings
    float error_x = 2 * (accel_vector[0] * gw_x + accel_vector[1] *
gw_y + accel_vector[2] * gw_z);
    float error_y = 2 * ((accel_vector[1] * gw_z - accel_vector[2] *
gw_y));
    float error_z = 2 * ((accel_vector[2] * gw_x - accel_vector[0] *
gw_z));

    // Compute feedback correction quaternion
    float alpha = 0.02f; // Correction gain
    correction->w = 1.0f;
    correction->x = alpha * error_x;
    correction->y = alpha * error_y;
    correction->z = alpha * error_z;
}
```

Figure C.2 Source code for orientation_utils.c

The header file LQR_controller_driver.h outlines the interface and structure for implementing an LQR controller. This file declares the LQR_controller struct, which contains arrays for handling different gain sets based on the rocket's velocity and orientation state.

```
/*
  Leeds University Rocketry Organisation - LURA
  Author Name: Alexandra Posta
  Description: Include LQR Controller header file
*/
#ifndef LQR_CONTROLLER_DRIVER_H
#define LQR_CONTROLLER_DRIVER_H

#include "orientation_utils.h"

#define STATE_SPACE_DIM     6       // Euler 3xangle 3xrates
#define NUM_GAINS           50
#define NUM_SERVOS          4
#define MAX_VELOCITY        120     // ms-1
#define MIN_VELOCITY        30      // ms-1
#define CANANDS_THRESHOLD   1500    // milidegree*1000

typedef struct LQR_controller {
    float* current_gain;
    float current_gain_index;
    float gain[NUM_GAINS * STATE_SPACE_DIM * NUM_SERVOS];
    float available_gains[NUM_GAINS * NUM_SERVOS * STATE_SPACE_DIM];
    float avg_gains[NUM_GAINS][NUM_SERVOS][STATE_SPACE_DIM];
    float zero_gains[NUM_SERVOS * STATE_SPACE_DIM];
} LQR_controller;

/**
  @brief Initialise the LQR controller
  @param lqr LQR controller structure
*/
void LQR_init(LQR_controller* lqr);

/**
  @brief Update the gains of the LQR controller
  @param lqr LQR controller structure
  @param velocity Current velocity of the rocket in m/s
  @note the gains are set to zero if the velocity is below or above a
threshold
*/
void LQR_update_gain(LQR_controller* lqr, int velocity);

/**
  @brief Perform the LQR control
```

```
  @param lqr LQR controller structure
  @param orientation Current orientation data
  @param servo_defs Servo deflections angles
*/
void LQR_perform_control(LQR_controller* lqr, orientation_data
orientation, ServoDeflections* servo_defs);

#endif /* LQR_CONTROLLER_DRIVER_H */
```

Figure C.3 Source code for lqr_controller.h

The source file lqr_controller.c LQR controller designed for managing rocket orientation and stability. It includes several functions: LQR_init initialises the controller by setting up initial gain values across arrays. The LQR_update_gain function dynamically adjusts the controller's gains based on the rocket's velocity, applying zero gains if the velocity falls outside predefined safe operational ranges, thus maintaining control stability. Additionally, LQR_perform_control calculates necessary servo deflections based on current orientation and selected gains, incorporating safety thresholds to prevent exceeding mechanical limits.

```
/*
  Leeds University Rocketry Organisation - LURA
  Author Name: Alexandra Posta
  Description: Include LQR Controller source file
*/

#include "lqr_controller.h"

int _ravel_index_2d(int i, int j)
{
    return i * STATE_SPACE_DIM + j;
}

int _ravel_index_3d(int i, int j, int k) {
    return i * STATE_SPACE_DIM * NUM_SERVOS + j * STATE_SPACE_DIM + k;
}

void LQR_init(LQR_controller* lqr) {
    // Set the zero gain array to zero
    for (uint8_t i = 0; i < sizeof(lqr->zero_gains); i++) {
        lqr->zero_gains[i] = 0;
    }

    // Initialise the current gain and index to zero
    lqr->current_gain = &lqr->zero_gains[0];
    lqr->current_gain_index = 0.0f;

    // Initialise average gains
    double _avg_gains[NUM_GAINS][NUM_SERVOS][STATE_SPACE_DIM] = {
        {
            {5.9761e-05, -0.37796, -1.1106e-15, 0.26723, -0.38847, -
2.2002e-16},
            {5.9761e-05, 0.37796, 1.1899e-15, 0.26723, 0.38847,
2.4892e-16},
            {5.9761e-05, 1.3538e-15, -0.37796, 0.26723, 6.8727e-16, -
0.38847},
```

```
            {5.9761e-05, -1.1147e-15, 0.37796, 0.26723, -5.3388e-16,
0.38847},
        },
        {
            {5.9761e-05, -0.37796, -1.1945e-15, 0.26723, -0.3698, -
6.5912e-16},
            {5.9761e-05, 0.37796, 1.1945e-15, 0.26723, 0.3698, 8.5519e-
16},
            {5.9761e-05, 1.1922e-15, -0.37796, 0.26723, -5.9471e-17, -
0.3698},
            {5.9761e-05, -9.2426e-16, 0.37796, 0.26723, 4.3445e-16,
0.3698},
        },
        {
            {5.9761e-05, -0.37796, -2.541e-16, 0.26723, -0.35334,
3.2436e-16},
            {5.9761e-05, 0.37796, 1.0805e-16, 0.26723, 0.35334, -
3.8215e-16},
            {5.9761e-05, 5.845e-16, -0.37796, 0.26723, 4.866e-16, -
0.35334},
            {5.9761e-05, -7.2657e-16, 0.37796, 0.26723, -5.5427e-16,
0.35334},
        },
        //    REST OF THE CONTROLLER GAINS ARE NOT INCLUDED TO AID
READABILITY
    };

    // Include available gains
    for (int i = 0; i < NUM_GAINS; i++) {
        for (int row = 0; row < NUM_SERVOS; row++) {
            for (int col = 0; col < STATE_SPACE_DIM; col++) {
                lqr->avg_gains[i][row][col] =
(float)_avg_gains[i][row][col];
                lqr->available_gains[_ravel_index_3d(i, row, col)] =
(float)_avg_gains[i][row][col];
            }
        }
    }
    // Set the current gain
    lqr->current_gain = &lqr->available_gains[0];
}

void LQR_update_gain(LQR_controller* lqr, int velocity) {
    // Update gains based on speed
    if (velocity < MIN_VELOCITY) {  // Stop controller if speed to high
or low
        lqr->current_gain = &lqr->zero_gains[0];
    } else if (velocity > MAX_VELOCITY) {
```

```
        lqr->current_gain_index = 49;
        lqr->current_gain = &lqr-
>available_gains[_ravel_index_3d((int)lqr->current_gain_index, 0, 0)];
    } else {
        lqr->current_gain_index = ((float)NUM_GAINS - 1) *
(float)(velocity - MIN_VELOCITY) / (float)(MAX_VELOCITY -
MIN_VELOCITY);
        lqr->current_gain = &lqr-
>available_gains[_ravel_index_3d((int)lqr->current_gain_index, 0, 0)];
    }
}

void LQR_perform_control(LQR_controller* lqr, orientation_data
orientation, ServoDeflections* servo_defs) {
    // Extract Euler angles and Rates
    float _orientation[STATE_SPACE_DIM] =
{orientation.current_euler.roll,
 orientation.current_euler.pitch,
 orientation.current_euler.yaw,
 orientation.current_rate_euler.roll,
 orientation.current_rate_euler.pitch,
 orientation.current_rate_euler.yaw};

    // Perform control
    servo_defs->servo_deflection_1 = 0;
    servo_defs->servo_deflection_2 = 0;
    servo_defs->servo_deflection_3 = 0;
    servo_defs->servo_deflection_4 = 0;

    for (int col = 0; col < STATE_SPACE_DIM; col++) {
        servo_defs->servo_deflection_1 += lqr-
>current_gain[_ravel_index_2d(1, col)] * _orientation[col] * 100.0f *
180.0f / M_PI_F; //store in degrees * 100
        servo_defs->servo_deflection_2 += lqr-
>current_gain[_ravel_index_2d(2, col)] * _orientation[col] * 100.0f *
180.0f / M_PI_F;
        servo_defs->servo_deflection_3 += lqr-
>current_gain[_ravel_index_2d(3, col)] * _orientation[col] * 100.0f *
180.0f / M_PI_F;
        servo_defs->servo_deflection_4 += lqr-
>current_gain[_ravel_index_2d(4, col)] * _orientation[col] * 100.0f *
180.0f / M_PI_F;
    }

    if (servo_defs->servo_deflection_1 > CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_1 = CANANDS_THRESHOLD;
    } else if (servo_defs->servo_deflection_1 < -CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_1 = -CANANDS_THRESHOLD;
```

```
    }

    if (servo_defs->servo_deflection_2 > CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_2 = CANANDS_THRESHOLD;
    } else if (servo_defs->servo_deflection_2 < -CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_2 = -CANANDS_THRESHOLD;
    }

    if (servo_defs->servo_deflection_3 > CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_3 = CANANDS_THRESHOLD;
    } else if (servo_defs->servo_deflection_3 < -CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_3 = -CANANDS_THRESHOLD;
    }

    if (servo_defs->servo_deflection_4 > CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_4 = CANANDS_THRESHOLD;
    } else if (servo_defs->servo_deflection_4 < -CANANDS_THRESHOLD) {
        servo_defs->servo_deflection_4 = -CANANDS_THRESHOLD;
    }
}
```

Figure C.4 Source code for orientation_utils.c

# Appendix D – Firmware Setup

The startup file, displayed below, prepares the environment for the execution of a firmware application. It is executed immediately after the system is powered up or reset.

```
/*
  Leeds University Rocketry Organisation - LURA
  Author Name: Alexandra Posta
  Description: Startup file for the firmware; suitable for STM32L4R5
*/

// Startup code
__attribute__((naked, noreturn)) void _reset(void) {
  // Initialise memory
  extern long _sbss, _ebss, _sdata, _edata, _sidata;
  for (long *src = &_sbss; src < &_ebss; src++) *src = 0;
  for (long *src = &_sdata, *dst = &_sidata; src < &_edata;) *src++ =
*dst++;

  // Call main()
  extern void main(void);
  main();
  for (;;) (void) 0;  // Infinite loop
}

extern void SysTick_Handler(void);  // Defined in main.c
extern void _estack(void);          // Defined in link.ld

// 16 standard and 95 STM32-specific handlers
__attribute__((section(".vectors"))) void (*tab[16 + 95])(void) = {
    _estack, _reset, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
SysTick_Handler};
```

Figure D.1 Startup file

The code snippet below provides a set of system call implementations for newlib, a C standard library. These system calls handle operations like memory management with _sbrk, file manipulation routines such as _open, _close, and _unlink, and basic process controls including _exit and _kill. For instance, _write is redirected to send data serially over USART1, showing an adaptation to the embedded context where standard input/output interfaces might not be directly available.

```c
/*
  Leeds University Rocketry Organisation - LURA
  Author Name: Alexandra Posta
  Description: System calls for newlib
*/

#include "mcu.h"
#include <inttypes.h>
#include <stdbool.h>
#include <stdlib.h>

int _fstat(int fd, struct stat *st) {
  if (fd < 0) return -1;
  st->st_mode = S_IFCHR;
  return 0;
}

void *_sbrk(int incr) {
  extern char _end;
  static unsigned char *heap = NULL;
  unsigned char *prev_heap;
  if (heap == NULL) heap = (unsigned char *) &_end;
  prev_heap = heap;
  heap += incr;
  return prev_heap;
}

int _open(const char *path) {
  (void) path;
  return -1;
}

int _close(int fd) {
  (void) fd;
  return -1;
}

int _isatty(int fd) {
  (void) fd;
```

```c
  return 1;
}

void _exit(int status) {
  (void) status;
  for (;;) asm volatile("BKPT #0");
}

void _kill(int pid, int sig) {
  (void) pid, (void) sig;
}

int _getpid(void) {
  return -1;
}

int _read(int fd, char *ptr, int len) {
  (void) fd, (void) ptr, (void) len;
  return -1;
}

int _link(const char *a, const char *b) {
  (void) a, (void) b;
  return -1;
}

int _unlink(const char *a) {
  (void) a;
  return -1;
}

int _stat(const char *path, struct stat *st) {
  (void) path, (void) st;
  return -1;
}

int mkdir(const char *path, mode_t mode) {
  (void) path, (void) mode;
  return -1;
}

int _write(int fd, char *data, int len) {
  (void) fd, (void) data, (void) len;
  if (fd == 1) uart_write_buf(USART1, data, (size_t) len);
  return -1;
}
```

Figure D.2 System Calls

A linker script dictates how the compiler should place the program's sections into the memory of the target device.

```
/*
  Leeds University Rocketry Organisation - LURA
  Author Name: Alexandra Posta
  Description: linker script for the HFC firmware; suitable for STM32
*/

ENTRY(_reset);
MEMORY {
  flash(rx)  : ORIGIN = 0x08000000, LENGTH = 2048k
  sram(rwx) : ORIGIN = 0x20000000, LENGTH = 192k  /* remaining 64k in a
separate address space */
}
_estack     = ORIGIN(sram) + LENGTH(sram);    /* stack points to end of
SRAM */

SECTIONS {
  .vectors  : { KEEP(*(.vectors)) }   > flash
  .text     : { *(.text*) }           > flash
  .rodata   : { *(.rodata*) }         > flash

  .data : {
    _sdata = .;   /* .data section start */
    *(.first_data)
    *(.data SORT(.data.*))
    _edata = .;  /* .data section end */
  } > sram AT > flash
  _sidata = LOADADDR(.data);

  .bss : {
    _sbss = .;              /* .bss section start */
    *(.bss SORT(.bss.*) COMMON)
    _ebss = .;              /* .bss section end */
  } > sram

  . = ALIGN(8);
  _end = .;     /* for cmsis_gcc.h  */
}
```

Figure D.3 Linker File

A Makefile is a configuration file used with the make utility, a tool that automates the building of executable programs from source code. By defining specific "targets" and the rules to build these targets, a Makefile is used to automate the process of uploading or "flashing" the compiled firmware onto a specific hardware device, such as a STM32. The target executes a series of commands that transfer the binary file to the device's memory, enabling it to run the new code directly.

```
CFLAGS  ?=  -W -Wall -Wextra -Wundef -Wshadow -Wdouble-promotion \
            -Wformat-truncation -fno-common -Wconversion -Wno-unknown-
pragmas \
            -g3 -Os -ffunction-sections -fdata-sections -I. -Iinclude \
            -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16
$(EXTRA_CFLAGS) \
            -lm
LDFLAGS ?= -Tlink.ld -nostartfiles -nostdlib --specs nano.specs -lc -
lgcc -Wl,--gc-sections -Wl,-Map=$@.map
SOURCES = main.c startup.c syscalls.c STM32_init.c
drivers/MS5611_driver.c drivers/BME280_driver.c  \
          drivers/ADXL375_driver.c drivers/LSM6DS3_driver.c
test_routines.c data_buffer.c filters.c \
          orientation_utils.c lqr_controller.c drivers/SERVO_driver.c
kalman_filter.c

build: firmware.bin

firmware.elf: $(SOURCES)
    arm-none-eabi-gcc $(SOURCES) $(CFLAGS) $(LDFLAGS) -o $@

firmware.bin: firmware.elf
    arm-none-eabi-objcopy -O binary $< $@

flash: firmware.bin
    st-flash --reset write $< 0x8000000

dfu: firmware.bin
    STM32_Programmer_CLI -c port=usb1 --download firmware.bin 0x8000000

clean:
    del -rf firmware.*

debug:
    openocd -f ./openocd/scripts/board/st_nucleo_l4.cfg
```

Figure D.4 Makefile

# Appendix E – Database structure

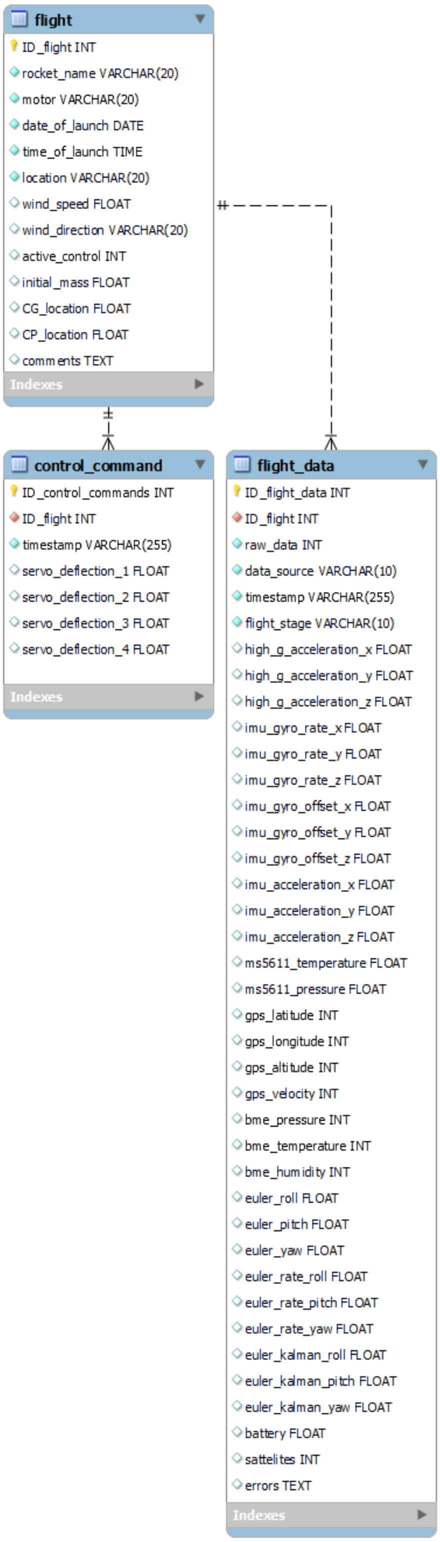



Figure E. 1 Database structure

# Appendix F – MATLAB Input Format Equations

Equation (1) estimates the altitude based on the atmospheric pressure measured at a given height compared to the sea level pressure. 0.19 approximates the change in pressure with altitude under a standard atmosphere.

$$h = 44330 * \left(1 - \left(\frac{p}{1013.25}\right)^{0.19}\right) \tag{1}$$

Equation (2) updates the vertical velocity of the rocket by adding the change in velocity due to acceleration over a small time interval, $\Delta t$. The constant 0.00980655 converts acceleration from the standard gravitational unit $g$ to $m/s^2$, aligning with the standard unit of velocity in meters per second.

$$v = v + a * 0.00980655 * \Delta t \tag{2}$$

Equation (3) calculates the mass decrease of a rocket over time as it burns propellant. The initial and propellant mass are divided by the burnt time, $t$.

$$m = m_{initial} - \frac{m_{propellant}}{t} \tag{3}$$

The longitudinal moment of inertia, $I$, of the rocket can be calculated using the Equation (4), where $I_{propellant}$ is the moment of inertia of the remaining propellant and $I_{structure}$ is the moment of inertia of the structural mass (excluding propellant).

$$I_{longitudinal} = I_{propellant} + I_{structure} \tag{4}$$

The moment of inertia for cylindrical bodies, typical rocket shapes, about their longitudinal axis can be calculated using Equation (5). In here, $m$ is the mass of the cylinder (propellant or structure), $r$ is the radius of the cylinder and $h$ is the height.

$$I_{cylinder} = \frac{1}{12} * m * (3 * r^2 + h^2) \tag{5}$$

Equation (6) calculates the rotational moment of inertia for a body, assuming a simplified cylindrical distribution of mass. The radius, $r$, indicates how far the mass, $m$, is spread from the rotational axis, and the 0.5 is a coefficient that changes based on the geometry of the body.

$$I_{rotational} = 0.5 * m * r^2 \tag{6}$$

The centre of gravity (CG) for the rocket is calculated based on the amount of propellant consumed, with an assumption that the CG shift is linearly dependent on the propellant mass consumed. The change in CG location is given by the Equation

(7), where $CG_{initial}$ is the initial centre of gravity location and $\Delta CG$ is the shift in the centre of gravity due to propellant consumption.

$$CG_{new} = CG_{initial} - \Delta CG \tag{7}$$

The shift in the centre of gravity ($\Delta CG$) can be calculated as Equation (8):

$$\Delta CG = \frac{m_{consumed}}{m_{initial}} * \frac{CG_{initial}}{2} \tag{8}$$

The Mach number is the ratio of the object's velocity to the speed of sound in the surrounding medium. $\gamma$ represents the heat capacity ratio of the air, $R$ is the specific gas constant for air, and $T$ is the ambient temperature. This equation is used to determine how supersonic the object's movement is relative to the speed of sound at a given temperature and atmospheric condition.

$$mach = \frac{v}{\sqrt{\gamma * R * T}} \tag{9}$$

# Appendix G – CPP

**IMPROVING AN ACTIVE STABILITY SYSTEM OF A SOUNDING ROCKET BY ADDING DATA MONITORING AND INTERPRETATION METHODOLOGIES**

**MECH5080M Contract Performance Plan**

MECH5080M Project Title*: **Improving an active stability system of a sounding rocket by adding data monitoring and interpretation methodologies***

***Students**:*

| | |
|---|---|
| *Alexandra Posta* | *201318973* |
| *Alexandre Monk* | *201299981* |
| *Antoine Durollet* | *201439724* |
| *Oliver Martin* | *201327297* |
| *Sam Bruton* | *201317656* |

*Supervisor: Dr Jongrae Kim*

*Industrial Mentor: Theo Gwynn*

*Date: 08/11/2023*

## Contents

## 1. Introduction

### 1.1 Background

The Leeds University Rocketry Association (LURA) is a student rocketry team, founded in 2021. In a short span, LURA has launched multiple rockets and set a new standard for United Kingdom (UK) teams at the Spaceport America Cup. The team is also on track to break the UK amateur altitude record, targeting an ascent to 13 kilometres [1]. All of the team's efforts are pointed towards the overarching long term goal of reaching the Karman line, the boundary between Earth's atmosphere and outer space, which no UK student team has reached. To support this goal, the Aptos Project has been created to develop a working active vertical control system that will allow future LURA rockets to maintain a vertical flight path and reach higher altitudes.

External factors have a significant impact on a rocket's trajectory. Typically, two main systems are employed to mitigate the trajectory. The first is a passive system, that is achieved by controlling the centre of pressure and gravity of the rocket [2]. The rule for stability is that the centre of pressure should be located at least one rocket diameter's length behind the centre of gravity [3]. However, the passive control system is not enough as the rocket will always weather cock due to cross winds, hence the addition of an active control system. [4] The second option is to use control surfaces. These surfaces come in various forms: they can be similar to the elevators on commercial aircraft, which adjust the passive fins' trailing edges, or they can be entire fins that rotate, akin to the rudders on fighter jets known as rolling tails [5]. The previous Aptos group suggested the use of canard fins mounted at the front of the rocket as the active control system. Their design was inspired by other rocketry teams, such as TU Delft, who successfully created a control system module to control roll [6]. The previous group built Pathfinder, a rocket capable of doing active control, and launched it at the Fairlie Moore Rocketry Site in Scotland.

However, the launches done last year lacked active stabilisation due to issues on the electronics and software systems. The current Aptos team plans to refine the existing work by optimising the code, redesigning the telemetry and electronics, as well as conducting at least one launch with the active control system enabled. If successful, this project would then be incorporated into future LURA rockets to reach higher altitudes and potentially set a UK precedent and aid other teams in their own development efforts.

### 1.2 Aim

The aim is to improve the active vertical stabilisation system of a sounding rocket, by using data monitoring, transmission, and interpretation techniques. This will allow refinement of the control system to correct the rocket's orientation with greater precision, in favour of a higher apogee.

### 1.3 Objectives

1. Create a control algorithm and simulation using a high-level development tool.
2. Create an electrical system & custom flight computer to provide all the required functionality to enable active control, telemetry, and data monitoring systems.
3. Improve the design of the canards system to achieve a more robust design and the ability to feedback the position to the control algorithm.
4. Establish air to ground telemetry communication with the rocket.
5. To perform data filtering, analysis, and visualisation to further improve the control loops.

### 1.4 Deliverables

*Table 2.1 Deliverables for each objective*

| | |
|---|---|
| 1 | Control algorithm producing output control data given rocket input sensor data.<br>Simulation of the rocket flight path and canard orientation given random and systematic interference.<br>Storage of all output data produced from the control algorithm for study and use in offline simulations. |
| 2 | Schematics of the custom flight computer & electrical wiring.<br>PCB Gerber files for the custom flight computer.<br>Manufactured custom flight computer & electrical system. |
| 3 | CFD analysis to determine canards shape.<br>Design an actuation system for the canards.<br>Manufacturing of the canards and actuation system. |
| 4 | Radio PCB that can interface with the flight computer to broadcast telemetry data.<br>Onboard and ground antenna designs and hardware capable of reliably transmitting data beyond apogee. |
| 5 | An easy to access and manipulatable database that contains flight data.<br>A web-based application to visualise data in dashboard format.<br>A script that feeds data from the database into the control algorithm. |
| Extra | A group report and five individual reports that outline the work completed.<br>A PowerPoint presentation to present the findings.<br>An ethical report that provides ethical considerations.<br>A GitHub repository that contains the control algorithm, flight computer firmware and data display related software. |

## 2. Project Outline

### 2.1 Tasks, milestones and timeline

The tasks, milestones, and timeline are laid out in Figure 2.1. While blue is the default colour, red highlights crucial tasks that are essential to the project's development. Although the team aims to finish all tasks within the given timeframe, some may require additional days, as shown by the floating lines.

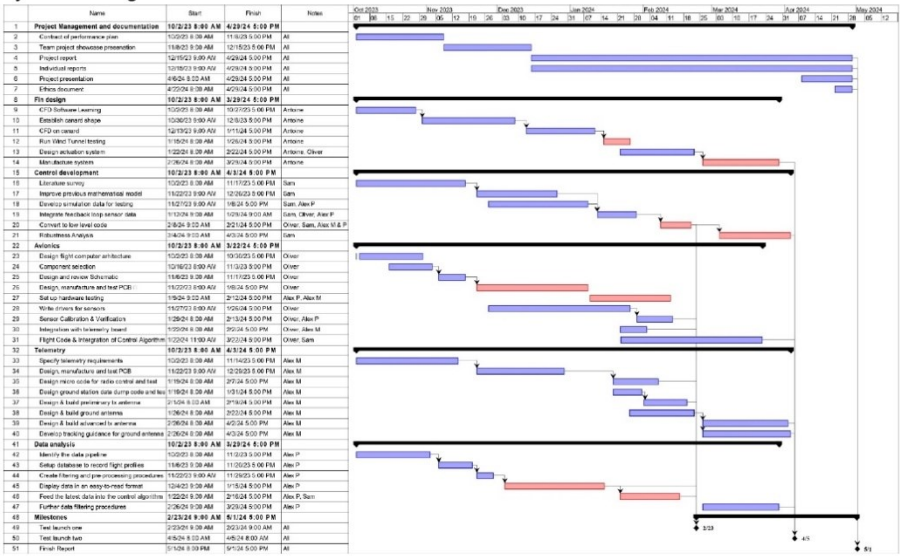


*Figure 2.1 Project Aptos Gantt Chart*

Three milestones were identified: the First Launch, the Second Launch and the Project Report deadline, all of which must be met by the 5th of May 2024. The initial milestone, scheduled for February 2024, tests early-stage systems on a lower-altitude launch without canards. In preparation for the launch, the control algorithm will be tested in a simulated environment, hardware-in-the-loop testing will be applied to the custom flight computer and ground telemetry testing will be performed. The second milestone is the launch of Pathfinder with an activated control system. The interim period focuses on refining electronics, software, and integrating mechanical systems. The final milestone corresponds to the deadline of the project report. Approximately one month has been allocated in April for report writing and final data analysis. The last task left for the team is to generate an ethics document related to the project.

## 2.2 Team structure

2.2.1 Software Engineer - Alexandra Posta

Alexandra is a fifth year Mechatronics & Robotics student with a placement completed at Scuderia AlphaTauri Formula One as a Software Engineer. As a Software engineer, Alexandra has developed data pipelines from the Wind Tunnel sessions, custom web applications for competitor analysis and simulation tools for pre-tunnel pressure testing. This experience makes Alexandra a candidate to filter, store and display flight data. From a rocketry perspective, Alexandra is leading the Avionics pocket from the Leeds University Rocketry Association (LURA), putting her in a good position to lead the group and organise launch days.

2.2.2 Electronics & Telemetry Engineer - Alexandre Monk

Alex is a Mechatronics & Robotics student who has completed a 14-month internship at Renishaw, focusing on FPGA bus design and Flash integration, building good experience in communications. Additional PCB design work completed on the placement will also aid the board design for the onboard telemetry. He also has extensive experience with automated C code generation from MATLAB, which should alleviate workload during this project when transferring the control algorithms developed for simulation onto hardware. Previous work on APRS and amateur radio tracking systems for weather balloons has provided the understanding necessary to design of all parts of the telemetry system. Past Formula Student electronics work and electric powertrain projects have given Alex a good electronics foundation and the practical experience necessary for reliable PCB design in high vibration applications.

2.2.3 Aerodynamics Engineer - Antoine Durollet

Antoine is a mechanical engineering student who has been a part of the Aerostructures team at the Leeds University Rocketry Association for a year. During this year he has gained valued experience in ensuring the integrity of the structure of rockets, as well as using the different flight simulation software to optimize the shape of rockets. All of these skills can be reapplied to design a canard actuation system. He has been learning about Fluids Dynamics for the last 5 years and knows how to use different CFD software, which will help him make decisions based on aerodynamics constraints. All those experiences give him the knowledge to fulfil the role of aerodynamics engineer.

2.2.4 Electronics Engineer - Oliver Martin

Oliver is a Mechatronics & Robotics student who has completed a 13-month internship at Red Bull Advanced Technologies, as an electrical design engineer. While on placement he gained experience defining electrical systems and their requirements, and then taking the appropriate steps to develop the system in an industry environment. He also has experience using microcontrollers, designing circuits, and programming in other projects, including working in the Avionics team at LURA. Therefore, he is well suited to the role of Electronics Engineer leading the development of the Avionics system.

2.2.5 Control Engineer - Sam Bruton

In the role of Control Systems Engineer, Sam is a final year Mechatronics and Robotics student with industry experience designing, prototyping, testing and commissioning factory operations equipment and machinery for Siemens on a 14-month placement. As a Robotics and Automation Engineer, he was responsible for system design and integration and has the ability to communicate and liaise with team members with different backgrounds, to successfully implement a system with exemplary control. He is also a member of the LURA Avionics team working on the control system for their latest rocket and has experience in simulation and modelling. Taking all this into consideration, he is best suited for this role.

**2.3 Resources**

2.3.1 Software Resources

The Avionics circuit schematics and PCB Gerber files will be produced using KiCAD, an open-source, free-to-use software. In conjunction with KiCAD, Library Loader from SamacSys will be used to add the necessary components into KiCAD, also free to use. For the control system design & simulation, MATLAB/SIMULINK will be used. Any CAD models for physical components will be designed in SolidWorks and CFD analysis will be carried out using Ansys. These three software packages have licences provided by the university.

2.3.2 Monetary Resources

In addition to software, there is a requirement for capital expenditure to purchase components facilitate launching the rocket. As the rocket structure has already been built and is reusable, the Bill of Materials is reduced from that required to build a complete rocket. Only components that are being re-engineered or are single use are included. Table 2.1 outlines the top-level project budget, a more detailed breakdown of costs can be found in Appendix A.

*Table 2.1 - Budget Outline*

| Item | Cost |
|---|---|
| Wind Tunnel Jig | £15.49 |
| Canards & Actuation | £29.65 |
| Avionics | £297.54 |
| Telemetry | £70 |
| Launch costs | £513 |
| **Sub Total** | **£933.18** |
| Contingency 20% | £186.64 |
| **Total** | **£1,119.82** |

## 3. Project Considerations

### 3.1 Risk analysis

Several risks were identified that could prevent the completion of the project, and steps were taken accordingly to mitigate the possibility and effects of any obstructing risks. An approach analysing risk probability and severity was taken in order to identify the most influential risks and introduce additional mitigation measures accordingly.

Each table contains risks associated with the project. The overall risk was calculated by multiplying the probability factor by severity and may go up to 25. The overall risk was recalculated after the mitigation factors were applied.

| **Risk:** Underestimation of actual duration to complete the planned functionality. | | | | | |
|---|---|---|---|---|---|
| **Probability** | 4 | **Severity** | 3 | **Overall Risk** | 12 |
| **Mitigation:** | Increase parallelism and reduce dependencies to allow more flexibility in individual task completion time. Factor of safety added to task time estimates to reduce likelihood of overrunning. | | | | |
| **Probability** | 3 | **Severity** | 1 | **Overall Risk** | 3 |

| **Risk:** Difficulties obtaining / manufacturing components. | | | | | |
|---|---|---|---|---|---|
| **Probability** | 2 | **Severity** | 3 | **Overall Risk** | 6 |
| **Mitigation:** | Widely available components with viable alternatives selected. Manufacturing methods that are widely available for low cost are used to mitigate risk of a single suppliers or component types being unavailable. | | | | |
| **Probability** | 1 | **Severity** | 2 | **Overall Risk** | 2 |

| **Risk:** Budget shortage. | | | | | |
|---|---|---|---|---|---|
| **Probability** | 3 | **Severity** | 4 | **Overall Risk** | 12 |
| **Mitigation:** | A core implementation budget and additional launch budget are specified. If a certain part of the budget is exceeded, the team can reduce the scope of testing without compromising on the deliverables. If necessary, external companies can be approached for sponsorship in order to bring in sufficient funds to complete the project. A contingency of 20% has been added within the budget estimate to mitigate risk of overspending. | | | | |
| **Probability** | 3 | **Severity** | 2 | **Overall Risk** | 6 |

| **Risk:** Expertise lacking. | | | | | |
|---|---|---|---|---|---|
| **Probability** | 3 | **Severity** | 3 | **Overall Risk** | 9 |
| **Mitigation:** | The project is designed to make use of existing knowledge on the team, without significant additional learning. Software packages are used that most members are familiar with. Simulation allows the team to experiment with novel solutions without significant risk to project completion. Academic and industrial supervisors can provide guidance if unexpected problems occur. | | | | |
| **Probability** | 2 | **Severity** | 2 | **Overall Risk** | 4 |

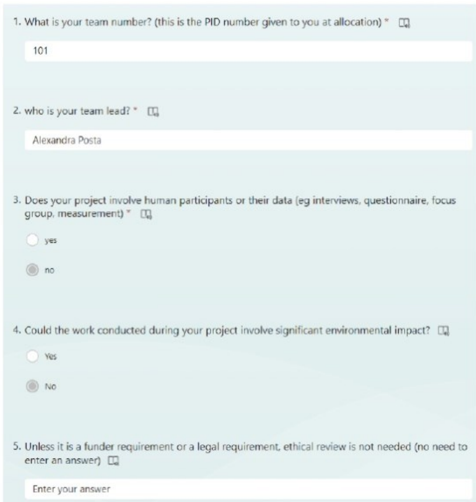
| **Risk:** Unable to launch a sounding rocket. | | | | | |
|---|---|---|---|---|---|
| **Probability** | 2 | **Severity** | 4 | **Overall Risk** | 8 |
| **Mitigation:** | Some risks associated with launch permissions and logistics are difficult to predict, however any planning or forms required for these will be completed as far ahead of time as possible to enable alternative or revised plans to be implemented. Simulation, and bottle rocket launches that do not require travel or licensing can be completed which would neglect only the physical canard implementation. Multiple launches are also planned to reduce the impact of a single launch cancellation. The possibility of downgrading to a smaller solid propellant rocket that is easier to license also exists. | | | | |
| **Probability** | 2 | **Severity** | 2 | **Overall Risk** | 4 |

| | | | | | |
|---|---|---|---|---|---|
| **Risk:** Catastrophic rocket failure or failure of a project part. | | | | | |
| **Probability** | 1 | **Severity** | 5 | **Overall Risk** | 5 |
| **Mitigation:** | The case of destruction of hardware has been planned for by reducing the value of actual components as much as possible to enable remanufacturing if necessary. An extensively tested and reliable rocket is used to reduce the risk of new problems or catastrophic failure occurring. Parts are individually tested for robustness and take-off forces are considered during the design phase. | | | | |
| **Probability** | 1 | **Severity** | 4 | **Overall Risk** | 4 |

| | | | | | |
|---|---|---|---|---|---|
| **Risk:** Loss of expected man hours. | | | | | |
| **Probability** | 4 | **Severity** | 2 | **Overall Risk** | 8 |
| **Mitigation:** | A margin of error has been added in task time estimates. The project Gannt chart will be routinely updated and regular meetings and schedule revisions ensure the project can continue on track as much as possible. | | | | |
| **Probability** | 4 | **Severity** | 1 | **Overall Risk** | 4 |

### 3.2 Ethical considerations

This project includes no human participants or their data and thus considerations to this effect do not have to be made. This has been pointed out in the ethical approval form, on the right. The subject is merely active control and telemetry for a rocket. Whilst these technologies can be implemented on weapons systems, for example in missile design, which can be considered immoral [7], the research conducted here will not intentionally contribute to the defence sector. Its application and scientific value for a student team outweighs any potential use by the defence sector.


1. What is your team number? (this is the PID number given to you at allocation) *

101

2. who is your team lead? *

Alexandra Posta

3. Does your project involve human participants or their data (eg interviews, questionnaire, focus group, measurement) *

yes

no

4. Could the work conducted during your project involve significant environmental impact?

Yes

No

5. Unless it is a funder requirement or a legal requirement, ethical review is not needed (no need to enter an answer)

Enter your answer

*Figure 3.1 Ethical Approval Form*

A consideration should be made as to the environmental impact of launching a rocket, as the sounding rocket launches planned for the project do emit greenhouse emissions from the black powder and solid propellant burned onboard. However the quantities of these are small and thus have an insignificant impact, meaning ethical approval is not required.

Risk of a rocket rocket failure event is incredibly low. A launch with the control system switched off will be conducted first to enable control system behaviour. This is to be evaluated and deemed safe before any controlled launch occurs. The rocket will also be launched at a designated launch site, clear of people or property, to low altitudes, where even a catastrophic failure or incorrect guidance would not result in destruction or harm to any life or property.

### 3.3 Project Stakeholders

Three stakeholders are interested in the success of this project. Firstly, the Engineering Department from the University of Leeds would benefit from the project. A vertical control algorithm would not only enhance the university's reputation in rocketry but also draw positive attention as it allows students to undertake innovative projects. Secondly, Theo Gwynn from Airbus is integral to the project, as he can offer vital industrial expertise alongside his Airbus colleagues to push the project forward. Finally, LURA anticipates considerable advantages from a successful outcome, as it would enable the integration of active control stabilization systems in its future rockets — essential for setting new altitude records in the UK.

## 4. Conclusion

In conclusion, project Aptos aims to integrate a more robust data pipeline into the control algorithm of the vertical active stabilisation system of a sounding rocket. Ultimately, this would enable the greater team, LURA, to launch rockets at higher altitudes. The team working on the project has a diverse range of skills and background knowledge to build upon the system developed in the previous year. This year, the team's primary focus will be on refining the control algorithm. To achieve this, innovative methodologies will be incorporated to improve the way we acquire, transmit, process, and utilize data.

## 5. References

[1] UKRA. 2019. *UK Rocketry Altitude Records.* [Online]. [Accessed on 1st November 2023]. Available from: http://www.ukra.org.uk/records/allclass

[2] Benson T. NASA. 2021. *Conditions for Rocket Stability*. [Online]. [Accessed on 1st November 2023]. Available from: https://www.grc.nasa.gov/WWW/k-12/rocket/rktstabc.html

[3] Mandell G., Caporaso G., Bengen P. B. 1973. *Topics in Advanced Model Rocketry*.

[4] Benson T. NASA. 2021. *Weather Cocking*. [Online]. [Accessed on 1st November 2023]. Available from: https://www.grc.nasa.gov/www/k12/rocket/rktcock.html#:~:text=Rocket%20Weather%20Cocking&text=Following%20the%20liftoff%20of%20a,the%20top%20of%20the%20figure

[5] Pike J. 2016. *F-14 Tomcat*. [Online]. [Accessed on 1st November 2023]. Available from: https://www.globalsecurity.org/military/systems/aircraft/f-14-design.htm

[6] DARE n.d. 2014. *Advanced Control Team – Delft Aerospace Rocket Engineering*. [Online]. [Accessed 5th November 2023]. Available from: https://dare.tudelft.nl/projects/act/

[7] Forge J. *The Morality of Weapons Research*. Science and Engineering Ethics, vol. 10, pp. 531-542, 2004.

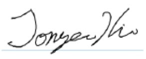
**Appendix A. Budget**

**Project Aptos Module CPP Budget**

| Cost Item | Item Name | Quantity | | Price per unit | URL | Total Price(inc. VAT) |
|---|---|---|---|---|---|---|
| **Wind Tunnel Jig** | | | | | | **£ 15.49** |
| Aluminium Plate | 8mm Aluminium Plate Sheet - Grade 5083 (100mm x 150mm) | 1 | | £ 9.99 | https://www.amazon.co.uk/8mm-Alumini | £ 9.99 |
| Rods to fix wing | Aluminium Round Bar/Rod 4mm - 16mm : Grade 6082 T6 (4mm x 100mm) | 2 | | £ 2.75 | https://www.amazon.co.uk/Aluminium-Ro | £ 5.50 |
| | | | | | | |
| **Canards and Actuation** | | | | | | **£ 29.65** |
| Axle | M3 Stainless Steel circular rod | 1 | | £ 3.07 | https://www.fixandfast.co.uk/stainless-ste | **£ 3.07** |
| Ball Bearing | ID M3 Ball Bearing | 4 | | £ 2.55 | https://www.accu.co.uk/rotary-bearings/6 | **£ 10.20** |
| Canard and Gear Filament | RS PRO 1.75mm Random Colour PLA 3D Printer Filament | 3 | | £ 5.46 | https://uk.rs-online.com/web/p/3d-printin | **£ 16.38** |
| | | | | | | |
| **Avionics** | | Per board | Spare | | | **£ 297.54** |
| MCU | STM32L4R5VIT6 | 1 | 1 | £ 13.37 | https://uk.rs-online.com/web/p/arduino/7 | £ 26.74 |
| IMU 6 axis | LSM6DS3TR-C | 2 | 1 | £ 5.85 | https://www.mouser.co.uk/ProductDetail/ | £ 17.55 |
| Barometer | MS5611-01BA | 2 | 1 | £ 6.06 | https://uk.rs-online.com/web/p/pressure- | £ 18.18 |
| High G 3 axis Accelerometer | ADXL375 | 2 | 1 | £ 12.75 | s/ADXL375BCCZ?qs=sGAEpiMZZMu6vlrqIF | £ 38.25 |
| GNSS | MAX-M10S-00B | 1 | 1 | £ 16.80 | https://www.mouser.co.uk/ProductDetail/ | £ 33.60 |
| EEPROM | 25LC512 | 1 | 1 | £ 1.82 | https://www.mouser.co.uk/ProductDetail/ | £ 3.64 |
| Voltage Regulator | TLV76733DGNR | 2 | 1 | £ 0.70 | https://www.mouser.co.uk/ProductDetail/ | £ 2.10 |
| Current limiter | ST890CDR | 1 | 1 | £ 2.37 | https://www.mouser.co.uk/ProductDetail/ | £ 4.74 |
| RGB LEDs | SMD LEDs | 2 | 2 | £ 0.55 | https://www.mouser.co.uk/ProductDetail/ | £ 2.20 |
| RED LED | SMD LEDs 0805 | 2 | 2 | £ 0.47 | https://www.mouser.co.uk/ProductDetail/ | £ 1.88 |
| BLUE LED | SMD LEDs 0805 | 1 | 1 | £ 0.42 | https://www.mouser.co.uk/ProductDetail/ | £ 0.84 |
| Buzzer | Piezo SMD Buzzer | 1 | 1 | £ 2.33 | https://www.mouser.co.uk/ProductDetail/ | £ 4.66 |
| Resistors | 0805 size SMD resistors | 20 | 20 | £ 0.20 | | £ 8.00 |
| Capacitors | 0805 size SMD capacitors | 20 | 20 | £ 0.30 | | £ 12.00 |
| Diodes | Schotky diodes 0805 | 5 | 3 | £ 0.40 | | £ 3.20 |
| OR Gates | 74AHC1G32GW-Q100,1 | 5 | 2 | £ 0.20 | https://www.mouser.co.uk/ProductDetail/ | £ 1.40 |
| Mosfet | N Channel Mosfet | 1 | 1 | £ 0.30 | https://www.mouser.co.uk/ProductDetail/ | £ 0.60 |
| Connectors | Pin headers, Battery connectors, & other PCB mounted connectors | 5 | 3 | £ 1.00 | | £ 8.00 |
| PCBs | Custom PCBs from JLC PCB | 2 | - | £ 25.00 | | £ 50.00 |
| Arming Switch | | 1 | 0 | £ 3.00 | | £ 3.00 |
| LiPos | 850MAH 7.4V 2S 35C SUPERSPORT PRO LIPO BATTERY (With XT30 Connector) | 2 | 2 | £ 10.49 | https://www.overlander.co.uk/lipo-batteri | £ 41.96 |
| STM32 Nucleo Board | For programming and testing | 1 | | £ 15.00 | | £ 15.00 |
| **Telemetry Electronics** | | | | | | **£ 70.00** |
| Radio Receiver Transmitter | | 1 | | £ 10.00 | | £ 10.00 |
| PCB | | 2 | | £ 10.00 | | £ 20.00 |
| Rocket Antenna | | 1 | | £ 15.00 | | £ 15.00 |
| RTL-SDR | | 1 | | £ 25.00 | | £ 25.00 |
| Ground Antenna | | 1 | | £ 15.00 | | £ 15.00 |
| | | | | | | |
| **Launch Costs** | | | | | | **£ 513.00** |
| 1st launch Motors | H123W-14A 38/240 RMS-PLUS | 2 | | £ 42.50 | https://wizardrockets.co.uk/product/h123 | £ 85.00 |
| 2nd launch Motors | J570W-14A 38/1080 RMS-PLUS Motor | 2 | | £ 114.00 | https://wizardrockets.co.uk/?product=j57 | £ 228.00 |
| Fuel | Fuel for journey to/from SARA and MRC (based off previous launches) | 2 | | £ 100.00 | | £ 200.00 |
| | | | | | | |
| **Sub-Total** | | | | | | £ 933.18 |
| **Contingency** | | 20% | | | | £ 186.64 |
| **Total** | | | | | | **£ 1,119.82** |

# Appendix H – Meeting logs

- MECH5080M Team Project - Supervision Meeting Log

| **Meeting number**: 1 | **Date**: 02/08/23 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine, Dr Jongrae Kim |
|---|---|---|
| **Agenda** | | |
| • Introduction to the group project | | |
| **Progress since last meeting** | | |
| • Generated the brief | | |
| **Key notes** | | |
| • We need to write the initial flight simulations in Python/MATLAB and only after translating to C. The results need to be compared<br>• Minimum of two people need to check any software developments<br>• Antoine is in charge of assembly<br>• Think about the IMU/navigation system that will be used on the rocket | | |
| **Actions for next meeting** | | |
| Decide:<br>• What to achieve for the first launch<br>• Clear task distribution (create a block diagram for it as well and use it as a tracking system)<br>• Draft for budget<br>• Decide who is leading | | |
| **Supervisor signature** | | |
|  | | |

| **Meeting number**: 2 | **Date**: 17/10/23 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine |
|---|---|---|

**Agenda**

- Progress Update
- Decision on leader and submission of ethics form
- Go briefly through the previous work
- Task allocation and create rocket system diagram
- Discuss what would we like to achieve for the first launch
- Check the budget

**Progress since last meeting**

- Get the information from the previous Aptos group
- Alex Monk did research:
  - We're limited to a 10mW transmitter. We could get an extension to 400mW, but that's not a lot of data. At apogee we can get about 50bytes/s of data

**Key notes**

Team leader is… Alex Posta. The module leader is required to be informed of this decision by email

Specification:

- Talked about how the general tasks were split, and what to do for the next meeting
- Check document here: https://leeds365-my.sharepoint.com/:w:/r/personal/mn20a2d_leeds_ac_uk/_layouts/15/Doc.aspx?sourcedoc=%7B4A9DAA4B-9BEC-4609-ABE8-D0C9B6BAA0F7%7D&file=Specification.docx&action=default&mobileredirect=true

First launch:

- Flying depends on the weather. Launch sites open back in February, but the weather is pretty unstable then. We should aim for the system to be ready by mid-February but expected to fly in March. (19th February)
- For the launch:
  - Minimum viable avionics system: get the first iteration for PCB and get data

Budget:

- Try to get 2 launches in: one small MRC (mid Feb) and one big one in SARA
- Rough estimate £1500

**Actions for next meeting**

Task Allocation:

- Fill out the specification document individually. This will be reviewed by the end of the week.
- Estimate the length of each task for addition to the Gantt chart.

- Have a look at the previous CPP and come back to talk about what needs to be changed.
- Talk to Dr Kim,
  - organize a meeting with Airbus (Theo Gwynn)
  - clarify whether we can use last year's data.
- Alex Monk: system diagram and what is needed for the first launch.
- Ollie: think what electronics are needed and include them into budget.
- Sam: check budget for control + redesign control for canards.
- Antoine: check WT and prior simulations for CFD.
- Alex Posta:
  - Submit the ethics form to the module leader (Wassim Taleb).
  - Check the general cost of launches and what is needed for the first launch.

**Supervisor signature**

| Meeting number: 3 | Date: 20/10/23 | Attendance: Alex Monk, Alex Posta, Sam, Oliver, Antoine, Dr Jongrae Kim |
|---|---|---|

**Agenda**

- Go through actions taken since last meeting
- Check initial:
  - task allocations
  - specifications
  - budget
- Talk through launch schedule
- Any other questions

**Progress since last meeting**

All:

- Initial task allocation and specification put together
- Initial budget sheet created
- Team lead decided and ethical form submitted
- Each group member did some research on their respective topics
- Chapters put together for the CPP

**Key notes**

- Bi-weekly meeting starting 3rd November, 12pm Fridays.
- Decision to use STM32 as flight computer.
- For first rocket launch, where canards will not be active, feed target orientation data into the control system, instead of the actual orientation, to check response given an optimal flight. Consider control system response when orientation error is large.
- Add mechanical end stops to the canards.

**Actions for next meeting**

- Create a Gantt Chart for tasks and send it BEFORE the next meeting. Make sure to structure it to clearly demonstrate tasks dependencies, i.e. which tasks can only be started after another is finished.
- Add hardware & software simulation tasks to task list.
- Create a technical specification / requirements document, and include metrics/methods that demonstrate successful testing/completion.
- Create a design interfaces document showing connections and comms protocols between each section / PCB / chip that will be utilised. Include a complete system diagram.
- The document should also contain the data pipeline, showing data format from sensors, to flight computer, to storage & telemetry.
- Have CPP draft complete by next meeting.
- Create a mass budget estimate and select a target altitude.

**Supervisor signature**

| **Meeting number**: 4 | **Date**: 24/10/23 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine |
|---|---|---|

**Agenda**

- Progress of Gantt chart
- System Diagram
- CPP section allocation
- Avionics:
  - Obtaining Arduino & Servos for testing
  - Component redundancy & failsafe procedure
  - Arming/initialisation
- Discussion of Spin Can

**Progress since last meeting**

Antoine:
- Software Learning for CFD Analysis

Oliver:
- Avionics Specification, draw.io diagram and KiCad progression

Alex Monk:
- Telemetry Specification progression

Alex Posta:
- Outline of cpp and gantt chart

Sam:
- Research

**Key notes**

- Looked through the Gantt Chart and assigned people to all tasks.
- Decision to have two different power sources for avionics/canards.
- Add a relay for the canard? To cut the power. Yes, decided it should be investigated.
- Split the CPP tasks.

**Actions for next meeting**

- All to go through the Gantt chart and check if all section topics and deadlines are correct
  - Alex Monk – Tuesday
  - Antoine – Wednesday
  - Oliver – Wednesday
  - Alex Posta – Thursday
  - Sam - Thursday

**Supervisor signature**

| **Meeting number**: 5 | **Date**: 31/10/2023 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine |
|---|---|---|

**Agenda**

- Progress since last
- Gantt Chart
- CPP
- Friday meeting

**Progress since last meeting**

Alex Monk:

- Picked a frequency for telemetry, leaning towards a SMD transceiver chip

Oliver:

- Schematics are at about 50%, Resources section of CPP mostly completed, more budgeting

Antoine

- Research into different aerofoils, understanding better last year's aerofoil and why it was chosen

Alex Posta:

- Mainly looked at CPP and Gantt chart

Sam:

- CPP, research into control

**Key notes**

- CPP due at 12 on 8/11/23
- Do a review for the avionics schematics in two weeks
- Alex Monk – preference for C language for ground station interface, putting data into database.
- Gantt chart timing for telemetry doesn't work for the first launch
- Current title of the project doesn't fit with the work we need to do
- Going over objectives and deliverables – each person should do the objectives and deliverables for their section of the project.
- For resources, have small table for costs, if there is a page free then add the full table
- Meeting with airbus
  - Small PowerPoint about LURA and Aptos
  - What can airbus provide knowledge wise?
    - How they process data, data filtering, pipelines, data correction
    - How do they suggest we test the system before flight
    - Suggestions about mounting the canards and linkage to the servos
  - Financial support

**Actions for next meeting**

- Review the budget by end of Thursday 2/11/23 - ALL
- Move schematics & all files onto the shared OneDrive – ALL
- Get hold of Theo's individual report – Alex Posta
- Review gantt chart timings – Alex Monk
- Have draft of all CPP sections by Friday 3/11/23 – ALL
- Create a more accurate project title that reflects the project
- PowerPoint for airbus meeting – Antoine

**Supervisor signature**

| **Meeting number**: 6 | **Date**: 03/11/2023 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine, Dr Jongrae Kim, Theo Gwynn |
|---|---|---|

**Agenda**

- Small PowerPoint about LURA and Aptos
- Ask Airbus
  - If they need anything from us?
  - What can airbus provide knowledge wise?
    - How they process data, data filtering, pipelines, data correction
    - How do they suggest we test the system before flight
    - Suggestions about mounting the canards and linkage to the servos
  - Financial support
- Ask for feedback on CPP and Gantt Chart

**Progress since last meeting**

- All – work on the CPP

**Key notes**

- Objectives:
  - Get started on the wind tunnel early as it takes a long time to have it available
  - Need to define success parameters for avionics
  - Need to think of a backup plan in case of things don't work

- Airbus Q&A:

1) What does Airbus need from us?
   a) Airbus just wants to develop relationship with university and students. Airbus will never be able to use the data. So, we define what we want to do
2) How they process data, data filtering, pipelines, data correction
   a) Theo will contact someone for that
3) How do they suggest we test the system before flight
   a) Theo will contact someone for that
4) Suggestions about mounting the canards and linkage to the servos
   a) Mount the canards to a stronger body. Want to shift and separate the load
5) Can we be added to the presentations, with the CubeSat project?
   a) Theo will see what he can do
6) Can we get money?
   a) Probable not directly from Airbus. However, Airbus works with companies that support uni teams. So, Theo will reach out to them
7) Any advice for LURA in the future?
   a) LURA has done well, but it will be interesting how it goes seeing that Theo Y. is gone

**Actions for next meeting**

Drop a message to Theo with questions.

**Supervisor signature**

| **Meeting number**: 7 | **Date**: 21/11/2023 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine |
|---|---|---|

**Agenda**

- Progress Update
- Budget Discussion
- Schematics Review
- Presentation Review
- Task List for next meeting

**Progress since last meeting**

Antoine:
- Has selected the fin shapes, but has been busy with LURA

Alex Monk:
- Theoretical schematic for the PCB. Implementation in KiCad required

Sam:
- Literature and project research undertaken, but has been busy with LURA

Oliver:
- Didn't do much. Worked on LEDs, buzzers that we will need. Did some current predictions.

Alex Posta:
- Looked at data bases. Seem to be going for InfluxDB
- Got feedback from Theo

**Key notes**

- When is the presentation poster deadline?
- Hardware testing should start in beginning of January
- CRC checking was recommended by Theo to check the data. Need to create a protocol that includes CRC checking. He talked about SpaceWire protocol.
- Dave asked about data filtering.
- Theo didn't answer the question about actuators.

- Budget must be submitted fast. Uni can be slow to approve budget and order stuff for us. Hopefully it's not the SIPR method.
- Budget:
- Draft PCB to be included

- Look at the schematics on our own, but Ollie gives us an overview of what he has drawn. Will have a review later this week. Has been decided to be on Friday
  Maybe use diodes to prevent reverse current? Drop a message to Arthur about it.
- Need to decide how the internal structure is.

- Need to prepare a ppt for the presentation showcase.

**Actions for next meeting**

- Antoine:
  - Need to run CFD sims
  - Alex Monk:
  - Need to do draft PCB on KiCAD
  - Create a chat with Theo Gwynn
  - Ollie:
- Want to get schematics done this week
- Alex Posta:
- Ask Dr Kim on Wednesday how to order stuff

**Supervisor signature**

| **Meeting number**: 8 | **Date**: 29/11/2023 | **Attendance**: Alex Monk, Alex Posta, Sam, Oliver, Antoine |
|---|---|---|

**Agenda**

- Updates from everyone
- PowerPoint Presentation
- What to do for next week

**Progress since last meeting**

Antoine:
- CFD simulations are ready, Ansys took quite some time to run some simulations
- Got the model in a steady state

Alex Monk:
- None

Sam:
- MATLAB legacy, looking into LQR => improve with steady state error

Oliver:
- Completed the schematic, schematic review and started the PCB design

Alex Posta:
- Install MySQL and InfluxDB locally and test them

**Key notes**

- Go over the PowerPoint for the December showcase

**Actions for next meeting**

Oliver:
- Finish the PCB and complete BoM
- Generate new PDF with PCB

Alex Monk:
- Have schematics ready and reviewed
- Try to start PCB layout

Sam:
- Fix the file for MATLAB
- Develop the equations for the new canard + steady state error

Antoine:
- Finish CFD simulations for next week

Alex Posta:
- Select a database (MySQL) and start on the server (Flask)

All:
- Work on the PowerPoint

**Supervisor signature**

| **Meeting number**: 9 | **Date**: 06/12/2023 | **Attendance**: Alex Monk, Alex Posta, Oliver, Antoine, Dr Jongrae Kim |
|---|---|---|

**Agenda**

- Order PCB components
- Check the PowerPoint for the presentation

**Progress since last meeting**

All:

- Have worked on providing information for the PowerPoint

**Key notes**

- Model mathematical model of the rocket to feedback into the control when testing
    - How do we model the force relative to speed on the canards
    - Complexity comes from speed => assume air density is constant
    - Velocity changes => torque generated by fin is difficult (this needs implementation in simulator)
    - Then convert the response into fake sensor data

- Review of PowerPoint
    - Too many figures per each slide
    - Too much text on slides

**Actions for next meeting**

- Everyone to change their sections of the PowerPoint to account for the feedback

**Supervisor signature**

| **Meeting number**: 10 | **Date**: 11/12/2023 | **Attendance**: Alex Monk, Alex Posta, Oliver, Antoine, Sam |
|---|---|---|

**Agenda**

- PowerPoint Project Showcase
- Work during winter
- When we are back

**Progress since last meeting**

- Work on the Project Showcase PowerPoint

**Key notes**

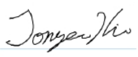
- Split presentation:
  - Introduction: Alex Posta
  - Aims + Objectives: Sam
  - Risk assessment: Alex Monk
  - Previous work: Ollie
  - Design/Update sections: Each one of us should talk about ours
  - Future work and conclusion: Alex Posta

- January:
  - Exams: 17$^{th}$ and 19$^{th}$ of January
  - PCB: After the 19$^{th}$ of January start to assemble
  - Wind Tunnel Testing: after the 22nd of January

**Actions for next meeting**

- Over Christmas (up until the 27$^{th}$ Dec):
  - Oliver:
    - Flowchart,
    - Look at drivers/logic,
    - Pseudocode for logic
  - Sam:
    - Improve MATLAB controller
  - Antoine:
    - CAD for testing jig
    - start actuators CAD
  - Alex Monk:
    - Initial design for the hardware-in-the-loop testing
    - Basic antenna design
  - Alex Posta:
    - Webserver + UI
    - Look into drivers for firmware

**Supervisor signature**



| **Meeting number**: 11 | **Date**: 12/01/2024 | **Attendance**: Alex Monk, Alex Posta, Oliver, Antoine, Sam |
|---|---|---|

**Agenda**

- Updates over Christmas
- Estimate arrival time for boards and actions
- Define tasks to do over January
- Launch Ops

**Progress since last meeting**

Antoine:

- Finalized shape and planform of canard

Alex Monk:

- Designed antennas, and ordered driven element planar patch PCBs

Ollie:

- Made a first software flowchart, but is pretty basic

Sam:

- No news, has been working for his exams

Alex Posta:

- Made a webserver for flight data

**Key notes**

- Parts are still waiting for approval and haven't been ordered yet
- Might want to focus on other tasks as the part arrival date is a big unknown. Lots of software and design to do
- Report is due on 1st May. Ideally, we will fly in the 1st week of April, but if we fly in the 2nd we can start writing it before flying

- Launch Ops:
    - Need to do a lot of testing to have the green light from UKRA. Especially if we want to go to MRC instead of SARA
    - Have flight computer running and have hardware testing done by end of February

**Actions for next meeting**

- Alex Monk to help Alex Posta for boards
- Ollie to do firmware if the boards do not arrive, and look at MATLAB translation into C with Sam. There's a library in Simulink but need to check it works properly
- Antoine needs to write the wind tunnel procedures and have them ready for after exams. And start doing a draft design of the actuation system
- Check actual data against the simulated data
- Having next meeting on the 26th at 12pm

**Supervisor signature**

| **Meeting number**: 12 | **Date**: 26/01/2024 | **Attendance**: Alex Posta, Oliver, Antoine |
|---|---|---|

**Agenda**

- Updates
- Workshop situation
- General testing procedures
- Wassim project updates
- Revise actions for next week

**Progress since last meeting**

Antoine:

- Canards are almost ready for testing. Will go to the workshop today to finalize the design and start printing.

Alex Monk:

- Received PCBs and ordered and receive filaments for custom antenna design

Oliver:

- Received PCBs
- Stencil is here! Just need to cut it at G68
- Look into servo drivers

Sam:

- No updates this week due to other commitments

Alex Posta:

- Improve on the web server, drivers for accelerometer/IMU, research hardware in the loop testing

**Key notes**

- Don't solder in the new workshop. Should be done either in electronics lab or Ollie's house.
- For WT testing, ask Antoine.

For Wassim

- we received 2 examples papers on Minerva
- talk about final report, next Tuesday at 1PM over Teams, anyone can join on the link: https://teams.microsoft.com/l/meetup-join/19%3ameeting_Yjc5NGFlZGUtZmE0MS00NzhiLTk3YTItN2YzNzViMmZkYTIw%40thread.v2/0?context=%7b%22Tid%22%3a%22bdeaeda8-c81d-45ce-863e-5232a535b7cb%22%2c%22Oid%22%3a%22f746f915-85b4-4cee-8456-4848428704d1%22%7d

**Actions for next meeting**

- Have common work sessions. Can go to the West Teaching Lab. Idea is to work together and talk, not each on their own. Next working session should be after the meeting with Dr Kim Jongrae.
- Have the weekly meetings at 10am/11am on Fridays.

**Supervisor signature**

| **Meeting number**: 13 | **Date**: 02/02/2024 | **Attendance**: Alex Posta, Oliver, Sam |
|---|---|---|

**Agenda**

- Updates
- Airbus conference
- Launch situation
- Revise actions for next week

**Progress since last meeting**

- Sam:
  - Go through the control and install adds-on
- Oliver:
  - Accelerometer, temp and IMU drivers' updates
  - Servo driver
- Alex Posta:
  - Try to run the MATLAB/Simulink simulation to get the controller into C

**Key notes**

- Try to run the control on MATLAB and realise there is not documentation in terms of what to install, which file to start

**Actions for next meeting**

Sam:

- Create a list of Adds-On and documentation (instructions, flow-chart) for the control
- Make sure the algorithm runs as last year

Oliver and Alex Posta:

- Continue working on firmware

**Supervisor signature**

| **Meeting number**: 14 | **Date**: 09/02/2024 | **Attendance**: Alex Posta, Alex Monk, Antoine Oliver, Sam |
|---|---|---|

**Agenda**

- Updates
- Look at part lists
- Launch Plan

**Progress since last meeting**

Sam
- Control: Simulation running, but only with certain OpenRocket data. Filtering and project organisation ongoing

Alex Posta:
- Firmware: Debugging on the flight computer:
    - SPI test and get it to work
    - Read barometer data

Oliver:
- Flight Computer: Board almost completely soldered, no obvious shorts so far

Antoine:
- Mechanical: Transmission design proposed

**Key notes**

- Servo transmission needs a chamfer, servo needs more secure attachment. Mounting system shouldn't protrude outside rocket body.
    - Suggested larger bearing / bearing removal and having the canard break on impact instead.

- For launch on 10th March:
    - Barometer, accelerometer, IMU data recorded
    - Initial control loop running with no direct output
    - Data saving to NAND Flash
    - Launch with simple antenna design.

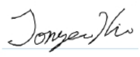
**Actions for next meeting**

- Do a mouser order for missing components

- Alex Posta wants to get accelerometer data reading out on flight computer, will move onto hardware in the loop testing of Ollie's board once MATLAB running

- Need to design a mounting system for flight computer and telemetry board
    - Alex Monk and Ollie need to send Antoine CAD models for boards

**Supervisor signature**



| **Meeting number**: 15 | **Date**: 16/02/2024 | **Attendance**: Alex Posta, Antoine, Oliver, Sam |
|---|---|---|

**Agenda**

- Updates
- Deadlines
- Launch Operations
- Purchasing

**Progress since last meeting**

- Control: Get the MATLAB script running, implement Kalman filter on the Barometer (input)
- Firmware: None
- Flight Computer:
  - Solder last parts (create soldering procedures) + create updates for future versions
  - Create secondary part order
- Mechanical: Start designing the PCB support for the launch
- Telemetry: None
- Structure for individual report - Ollie

**Key notes**

- Launch Operations:
  - Had a call with Paul from UKRA to ask if we can launch Pathfinder from MRC
  - He seemed quite positive about it, but had the following requirements:
    - Instead of going thorough TPS (Teams Project Support), we need to create a Facebook chat with him, Andy, Chris and Collin + all Aptos team
    - We need to send them documentation:
      - OpenRocket Simulations, CAD, further details about mechanical spec, servo motor spec (torque, movement, operating range) (list all parts, dimensions in mm)
      - Electronics, Firmware, Control, Telemetry overview
      - Failsafe mechanisms (mechanical, electrical, especially control)
      - Testing procedures
    - They want metal geared servos
    - We need to sign a waver (in case the rocket crashes and produces damage, it will be out fault rather than UKRA)
- For the mechanical side, focus more on the actual Aptos Launch rather than the 10th of March small academy rocket test launch

**Actions for next meeting**

- Set the general report structure and deadlines – Alex Posta

**Supervisor signature**

| **Meeting number**: 16 | **Date**: 23/02/2024 | **Attendance**: Alex Posta, Alex Monk, Antoine, Oliver, Sam |
|---|---|---|

**Agenda**

- Updates
- Deadlines + Report
- Targets over the next 2 weeks

**Progress since last meeting**

Antoine:

- Investigating new servos

Alex Monk:

- Use the air holes (drill additional ones) to mount the antenna; started the CAD
- Started soldering components for telemetry
- Written C code to work with transmitter
- CAD the antennas for the Academy rocket

Ollie:

- Firmware updates: SPI, sort out the delay function, system clock, watchdog running, LEDs, buzzer, UART

Alex Posta:

- Check deadlines and documents that need submitting
  - Get the MATLAB code running and start to look into hardware in the loop testing (HIL)
- Get a serial output in MATLAB

**Key notes**

- Cannot find servo, did not spec any servos
- Check deadlines document:
  - Deadlines.docx
- Plan for the next 2 weeks:
  - Finish Airbus presentation by the 27th of February (Tuesday)
  - Get the Academy rocket ready for the 10th of March
  - Finish poster between 11th - 13th of March

**Actions for next meeting**

- Friday:
  - Antoine to search workshop, living and Toby's room for servos
  - Alex Posta put PowerPoint together for Airbus
- Sunday:
  - If servos not found, Alex Posta and Antoine spec new servos
  - Antoine should buy academy motor (38mm, some H)
  - Ask Dom to launch it for us

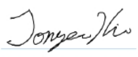
Antoine:

- Send Alex Monk CAD of Academy rocket

Sam:

- Get the MATLAB control working in C

**Supervisor signature**



| **Meeting number**: 17 | **Date**: 01/03/2024 | **Attendance**: Alex Posta, Alex Monk, Antoine, Oliver |
|---|---|---|

**Agenda**

- Airbus brief
- Updates
- Work on the following week

**Progress since last meeting**

Antoine:
- Look into buying servo motor
- Look into rocket motors

Oliver:
- Code: get the barometer data on Aptos
- Get the IMU to spit data

Alex Monk:
- Nothing this week

Alex Posta:
- Get the accelerometer data (in some form)
- General code flow

**Key notes**

Airbus:
- The propulsion was nice, we enjoyed the site. Enjoyed some of the talks. Rover arena wasn't very big, but still interesting.
- We are in a good position in terms of project compared to other teams. Quite happy to see our projects

Launch:
- Buy motor from the launch site. Add the I and J motors that we would like to launch with.
- Start the integration loop and write data to NAND Flash.
- Get the telemetry stuff inside the rocket for the small launch inside the rocket and still be approved by UKRA.
- Use PETR Gryphon as the rocket testing platform.

What frequency do we want to run the control to? 5-10 times closed loop bandwidth. 50Hz? They used 50ms time intervals for data reading, 10ms for gain updating.

**Actions for next meeting**

- We need a mounting solution/firmware development
- To Do:
    - Assembly ready for testing (Wednesday evening)
        - Bracket printed
        - Connection method to the rocket
    - Firmware ready for testing (Wednesday evening)
        - Accelerometer, IMU, barometer data
        - Store and read off NAND/SD card
        - Control converted to C
        - Initial code flow routine
    - Testing of assembly on (Thursday)

**Supervisor signature**

| **Meeting number**: 18 | **Date**: 08/03/2024 | **Attendance**: Alex Posta, Sam, Antoine, Oliver, Alex Monk |
|---|---|---|

**Agenda**

- Updates
- Launch Prep

**Progress since last meeting**

Antoine:
- Designed the board cage for the first launch
- Tried printing antenna for Alex Monk but had an issue, will try again.

Ollie:
- NAND flash code has been improved and test.
- Code written to get data off in CSV format.

Alex Monk:
- Can see signals showing up from transmitter to receiver. Plans to attach barometer for the launch

Alex Posta:
- Firmware. A lot of updates to the code.
- Overview of the structure and flow.
- Data from barometer and Accelerometer
- Data buffer for the last 50 readings.

Sam:
- Looked at generating C code.

**Key notes**

First launch will just be logging data not running any control code.
- Software flow is nearly ready for first launch.
- NAND flash is working
- Discussion around the format of input data the control algorithm need.
- Demo of telemetry progress
- Next launch could be April 7th in Cambridge
- Would we want to build up the second PCB
- Possibility of using university drones or Sam's drone to do testing.

**Actions for next meeting**

Oliver and Alex Posta:
- We need IMU driver complete for the control. Does IMU output angle or angular velocity

Sam:
- Check what is raw data needed in the control
- Conversion between CSV and open rocket data.

Alex Posta:
- Generate new frame array structure

Antoine:
- Print antenna & cage

Alex Monk:
- Details of all tests needed for the telemetry.

All:
- Meeting tomorrow 10am to complete assembly and procedures for the Sunday launch.
- Think about integration between main board and telemetry
- Poster due on Wednesday.

**Supervisor signature**

| **Meeting number**: 19 | **Date**: 08/03/2024 | **Attendance**: Alex Posta, Sam, Antoine, Oliver, Alex Monk |
|---|---|---|

**Agenda**

- Updates
- Launches

**Progress since last meeting**

Antoine:

- Wind Tunnel testing meeting with Sam. Going to be a few more weeks as they are testing a new equipment. Need to put pressure on Sam to do it asap.

Ollie:

- Will work with Sam to get the code working and changes needed to adapt legacy code to our new boards.
- Gyro data is pretty good.
- Accelerometer on the IMU is working.
- Tried to figure an angle from the axis of gravity. However, it uses the arctan function, which needs floating points that we don't have. Gives an approximate, but not close enough.
- Missing the BME280 and the servo drivers.
- Needs to do servo driver, BME driver and arctan problem.
- Might do low pass filters, but the data we get is good enough.
- Will check if the boards can fit horizontal in Aptos.

Alex Monk:

- Tried to demodulate the signal, but there is a lot of noise. Maybe the data rate is not correct? Not using an impedance match, so might have an impact.
- Need to try using a standard antenna to see if the problem isn't his antenna.
- Once demodulation is done, need to find a way to automatically read the data coming from the antenna. Need to copy the binary code from antenna into a .txt file before decoding by hand.

Alex Posta:

- Has been a bit ill. Poster has been submitted

Sam:

- Looked at the formulas for MATLAB and went over the code from last year to see what needs to be improved. Can't currently do floating points, which could be a problem for gains.
- Will work with Ollie to get the code working and changes needed to adapt legacy code to our new boards.
- Legacy was doing comms using Bluetooth. Getting rid of it and coming with an alternative solution to that.
- Need to work on servo drivers, and update controls from the Legacy.
- Need to implement changes of the updated Pathfinder to the simulations.

**Key notes**

- Servos are on their way to uni, and bushings are already here, waiting to be picked up.
- Launches:
  - G2 team to do launch on the 14th of April from MRC.
  - Can go to EARS on the 7th to do a test launch, do a small bottle test in the field?
  - Test telemetry in a car?
  - Can put it on a drone and fly it. Sam has a drone. Can test on Sam's commercial drone.
- People will be back before the 14th, but not too sure how long before. Can go to Peak District on the 5th to do testing.

**Actions for next meeting**

**Supervisor signature**

| **Meeting number**: 20 | **Date**: 08/03/2024 | **Attendance**: Alex Posta and M, Sam, Antoine, Oliver, Dr Jongrae Kim |
|---|---|---|

**Agenda**

- Updates
- Ask questions about report

**Progress since last meeting**

- See previous table – Meeting on the same day as previous

**Key notes**

- Next meeting: Friday 3pm

- If we want to reference the work of others, include a footnote with their names. Make sure everything is transparent.

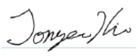
- Pick a literature paper and use the style of that paper, general style. Common mistakes:
  - Define acronyms (even 3D). Define when it first appears
  - Abstract is independent from all the report, define acronyms twice if they appear there
  - When you have formulas, define all variables under the equation. Examples when the variables are. All symbols need to be defined. If they appear after, it is ok
  - For Figures: put AXIS names and UNITS
  - X axis is something… which axis is there? Even put them on the figure. X,y axis. Add legend
  - Use IEEE reference
  - Figure and tables must be refereed in text before they appear
  - If figure is big, put it over two columns
  - Formulas, everything needs to be defined

Arxiv: Contains drafts papers https://arxiv.org/pdf/2311.11372.pdf

SERVOS: Antoine has two, Oliver has two

**Actions for next meeting**

**Supervisor signature**



| **Meeting number**: 21 | **Date**: 22/03/2024 | **Attendance**: Alex Posta, Alex M, Sam, Antoine, Oliver |
|---|---|---|

**Agenda**

- Updates
- Work to be done for next week

**Progress since last meeting**

Oliver:

- Managed to make the servos move independently through the board. Need to test the accuracy. The input gives the absolute position. The input is millidegrees. Can set the neutral position as we want, for now it's been set in the middle of the 4000 available values. Needs more testing to know which side is clockwise and anti-clockwise. Can be tested this weekend before Ollie leaves
- Formatted the IMU driver for readability
- Set min and max angle functions. If input a value higher than the max angle, it shouldn't go above the 15°
- Plugging in the servos uses the UART board. Which results in slowing down figuring out which angle it is at

Alex Posta:

- Converted the MATLAB control into C. At the moment it is pure maths, so no problems so far
- Created LQR controller
- Created matrix operations to translate matrices into coordinate systems
- Created a new Simulink model for the loop testing. Using serial blocks. Broke the control loop and added some serial blocks
- Send the roll, pitch and yaw of rocket directly into code

Sam:

- Has not touched the gains. Added filters on the yaw and pitch angle, and added PI controller
- From OpenRocket you can take the pressure rate and plopped that into the Simulink, which is more representative of how the speed of the rocket will be simulated

Antoine:

- Have all the parts ready for the first assembly test

Alex Monk:

- Antenna works better! Still not perfect. Little demo of it
- Have not just noise, but peaks showing the bits. There is a lot of reflection
- However, it's not centred around 433 MHz. It is not calibrated properly
- Also has a demodulator for it

**Key notes**

- Try to test the module while spinning
- Telemetry needs a better antenna
    - The current one is not good, so going to buy the one he needs on eBay
- Aptos boards don't fit inside the Aptos Module. Mount needs to be redesigned to have it vertical. Can use the old mount that was meant for Petr Griffin (Academy small rocket)
- The gyroscope drifts over time. Can be offset from the get-go by looking at the standard deviation, and also look at the accelerometer data to know where the gravity field is pointing towards

**Actions for next meeting**

Oliver:

- Verify the servo positions before leaving

Alex Posta:

- Get Ollie's code working to see the canards moving
- Test the controller on microprocessor, and using the loop in Simulink
- Get floating point to work

Sam:

- Need to check that modifications make sense, and are the correct representation of how it will be simulated. More testing and experiments

Antoine:

- Need to finish prepping all the parts and assemble them together. Will be done by Sunday morning
- Need to find an alternative for the wind tunnel. (IPSA? Need to ask one of his old teachers.)

Alex Monk:

- Get the oscillator going. Take out all the wrong decode/noise data
- Modify the PCBdesign
- Buy a new antenna

**Supervisor signature**

| **Meeting number**: 22 | **Date**: 22/03/2024 | **Attendance**: Alex Posta, Alex Monk, Sam, Antoine, Oliver, Dr Kim Jongrae |
|---|---|---|

**Agenda**

- Updates
- Questions about report

**Progress since last meeting**

See previous table

**Key notes**

- Need more specific titles/more details for the chapters. Have to have a specific font and size.
- Do not use webpages as reference unless it is the only source
- Do not use excuses such as “time limitations…”, “budget…”
- We should add a section about the rocket launch

Questions

- Do we need to have the same title?
    - No need for the same title for the individual report. Add the main title it as a subtitle under your individual title

- How do you add and reference code?
    - If you have a code snippet, add it as a figure. You do not need to add a reference to your own code. Make it clear if you took inspiration from somewhere. Add a link to GitHub, we do not have to make it public, add a footnote that says that code is not public and give access key or something

- Table of components with features for each component (all evaluation come from different sources)
    - You basically need to add references for all of those difference. Or, if you reference Mouser in multiple locations, add link to Mouser and say (see price on Mouser)

- If you made modifications, new versions
    - Tell the “story”. In an engineering report, you need to show off all of the process. That is one of the most important points in the report

- Should we use the ECSS standard?
    - Would be good to use standards, find improvements for next phase. We need to explain what those are before we use them. Take all documents as separate

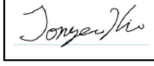
- Is he looking for a specific structure for the chapters?
    - No

- How much can he review our reports? Is it only one page or the whole report?
    - 20% of each report can be send for review. We can approach other academic as well for review.

- Do we need to post it on LinkedIn/post the screenshot of the LinkedIn post?
    - I do not know… maybe required, but not marked. Probably yes.

- How many references is he expecting?
    - 20-30 references. Include majority of them from journal papers. Single space, smaller font.

**Supervisor signature**



| **Meeting number**: 23 | **Date**: 05/04/2024 | **Attendance**: Alex Posta, Alex Monk, Sam, Antoine, Oliver |
|---|---|---|

**Agenda**

- Updates
- Drone and car tests
- Launch Operations

**Progress since last meeting**

Antoine:

- 3D printed the model of the drone support
- Got the data off the wind tunnel in France
  - Stationary canards with different angles of attack (from 0 degrees to 15 degrees)
  - Increase the wind speeds by increments of 5m/s up to 40m/s
- Redesigned the transmission system of the Aptos module; currently the canards are not attached properly to the servos

Alex Monk:

- Telemetry is not ideal; mostly working in the past as it transmits data; but frequency shifts every time when you turn it on
  - The oscillator was 4MHz instead of 40Mhz
  - A new oscillator was fit, registers are read correctly, but still does not get the frequency right => prob because the voltage input is not stable enough (it is not stable from Teensy/Power Supply/AAA Batteries), you get a drop in voltage when the current is drawn for the transmission => get a circuit to speed up the voltage set
    - Order a voltage regulator and some regulators

Sam:

- Look into Kalman filter between accelerometer and gyroscope to stop the gyro drift in midflight. Keep this for his report

Oliver:

- Further developed the servo driver; got the input as millidegrees
- Initial orientation of the board is worked out using the accelerometer; therefore, board can be initialized on the pad rather than ground
  - Due to the gyroscope reading; initially gyro was calibrated by lying on flat ground, but we cannot do that on a field.
  - Additionally, when rocket is stationary, remove the gyro drift using the acceleration data (if stationary the acceleration should reveal the orientation of the rocket on pad).
- Currently working on the update of the orientation based on accelerometer
- Try to set the servos to the orientation of the board to see if the Euler angles work, some issue with the char pointer

Alex Posta:

- Check the controller code from C that was translate from MATLAB using the hardware: faced multiple issue with the way in which the data was passed from one function to others; the gyroscope data was not calibrating after a time; the servo deflections were not correct angles; look at the servo transmission mechanism->canards are not attached properly
  - Solve the C pass by reference issues in various functions.
  - Got to the point in which the orientation function outputs some Euler angles and they are passed on the controller to receive servo deflections.
  - The servo deflections react to yaw/pitch but did not conclude whether the output is correct or not.
- Change the frameArray structure to reflect the new sensors.
  - FrameArray contains a maximum of 128 bytes
  - Included the majority of the sensors + Euler angle and rates
  - Need to talk to Ollie to confirm that structure is what is needed; Sam also mentioned two additional variables that he needs

**Key notes**

- For Antoine, try to get a mathematical equation for the canards; would be extremely beneficial for the controller in the future
- For Alex Monk, get a voltage regulator fitted; regulator arrives tomorrow (Amazon), another one comes on Monday (Mouser)
- Sam: give us a csv file of the Euler angle / rates / velocity / altitude
- Alex Posta: get the velocity out of barometer; change the NAND flash

Issues:

- Servo 1 works as long as you use it with ID 1 instead of 101
- For csv printing, do not use the equal sign; talk further about the NAND Flash storing procedure (Alex Posta + Ollie)
- Extra 96 bits available on the NAND Flash: Sam needs two values for Roll and Pitch
- Alex needs SPI1 (for telemetry) in mode 0
- Canard deflections: bump them to int16 and change the orientation to use the struct instead of the chart; store it in millidegrees

**Actions for next meeting**

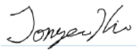
Drone test:

- Try to do a drone test on Wednesday.
  - If system does not look good, do further drone testing the week after the 14th
  - if weather does not improve by Tuesday, decide whether we want to do the launch
- Total payload test: approx. 500g
  - If needed; fly the telemetry assembly separate from the avionics

Tuesday meeting:

- 6:30pm Tuesday; decide what to do this week.

**Supervisor signature**



| **Meeting number**: 24 | **Date**: 19/04/2024 | **Attendance**: Oliver, Antoine, Alex Posta |
|---|---|---|

**Agenda**

- Updates
- Report structure
- Split sections to write for group report
- Next week plan

**Progress since last meeting**

All:

- **We launched a rocket!!!**
- Attempt a drone test, unsuccessful

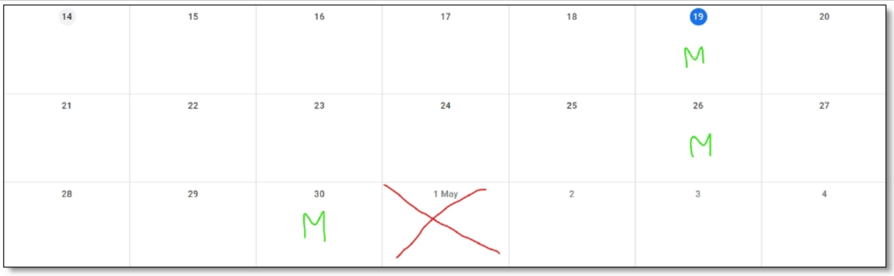
Oliver + Alex Posta:

- Eliminated gyro drift using accelerometer data
- Get the LQR to work when board was setup on the table and then reorientate axis of gyro for vertical velocity
- Optimise code running to get the main at 100Hz and faster
- Redesign flight loop: add buzzers, LEDs, trigger between flight stages slightly differently to make them more consistent
- Update vertical velocity calculation and check for landing using gyro data
- Vacuum chamber testing
- Get data off Flight computer after launch
- Change db and web structure to reflect the new frameArray

Antoine:

- Printed the PCB mount
- Reprint the servo mount, test fit and assembly
- Added slots for bushings and glued them in place
- Ran OpenRocket Simulations with the new weighted parts
- Look at mathematical model of the canards
- Looked at the wind tunnel data

**Key notes**

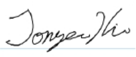



- Less than 2 weeks to submit
- See group word document for section splits
- Do final tests on Monday: run another vacuum test, try to do a drone test. Telemetry?
- Meet on Wednesday, the $24^{th}$, to check first draft of all sections for group report; meet at 2pm
- Late long meeting on the $30^{th}$ of April to submit the group report

**Actions for next meeting**

- Write report

**Supervisor signature**



| **Meeting number**: 25 | **Date**: 26/04/2024 | **Attendance**: Oliver, Antoine, Alex Posta, Alex Monk, Sam |
|---|---|---|

**Agenda**

- Updates
- Check meeting log
- Report
- Website/LinkedIn

**Progress since last meeting**

All:

- Work on report

Alex Monk:

- Two more iterations of the transceiver board; amazon oscillator did not oscillate at the correct rate; had to resolder new ones
  - Connect reset pins to incorrect voltage, resolder new board
  - Board goes into transmit mode, regulators work, does calibration and power amplifier
  - Antennas are printers, run tests

Ollie:

- Did a drone test and looked at results, had multiple issues: barometer is affected by prop wash, accelerometer and gyro faced too many vibrations; would be worth adding extra filters
- Found a prone app that works at 100Hz that does accelerometer, gyro and orientation (does quaternions into Euler, exactly as us); match the phone test to the flight computer: Sensor Logger

Alex Posta

- Small test bench for the database ingestion rate

**Key notes**

- Look through meeting logs and send them for checking

**Actions for next meeting**

- 

**Supervisor signature**