Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Measurement

# THRUST VECTOR CONTROLLED ROCKET MODEL

Bachelor's Thesis

Lukáš Málek

Field of study: Cybernetics and Robotics

Supervisor: Ing. Martin Šipoš, Ph.D.

May 2023

# Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Lukáš Málek

In Prague, 25. 5. 2023

# Acknowledgments

The completion of my bachelor's thesis was possible thanks to the support and contributions of several individuals. My supervisor Ing. Martin Šipoš, Ph. D, and Richard Šimeček, provided valuable guidance and assistance throughout the process. Radek Klesa and Matyáš Vašek helped with schematic design and PCB assembly. Kryštof Hájek played a crucial role in shaping my interests in model rocketry. I also acknowledge Ivo Lachman for sharing his knowledge and experience and the entire Avionics department in CRS for helping me to develop flight software for a real-time operating system Apache Nuttx. Lastly, I thank my family for providing me with the best conditions to develop this project effectively and freely.

# BACHELOR'S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Málek Lukáš** Personal ID number: **491987**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Measurement**

Study program: **Cybernetics and Robotics**

## II. Bachelor's thesis details

Bachelor's thesis title in English:

**Thrust vector controlled rocket model**

Bachelor's thesis title in Czech:

**Vektorování tahu modelu rakety**

Guidelines:

The goal of the bachelor thesis is design and realization of the Thrust vector controlled (TVC) rocket model. The TVC model rocket is a unique type of rocket that utilizes a control system to alter the direction of thrust, providing stability and control during flight. The design and realization should cover following steps:
- Modeling of the Rocket and the Thrust Vector Control Mount and their 3D print.
- Selection of Appropriate Sensors, Microcontroller Unit (MCU) and Optimal Rocket Motor based on parameter analyses.
- Design and realization of a Custom Printed Circuit Board (PCB).
- Evaluation of the Most Suitable Operating System, Outlining of the Code Structure, Design of a State Machine, Programming of the Flight Software.
- Simulation of the Proportional Integral Derivative (PID) Controller, Tuning of the PID Controller.
- Establishment of a Test Stand for Launches, Rocket Launch.
- Analysis of Flight Data, Improvement of the Rocket from Gathered Data, Interpretation of Results and Discussion.

Bibliography / sources:

[1] T. S. Taylor, Introduction to Rocket Science and Engineering. CRC Press, Taylor & Francis Group, 2017, ISBN: 9781498772327
[2] Vance, A. (2015). Elon Musk: How the Billionaire CEO of SpaceX and Tesla is Disrupting the Automotive, Aerospace and Energy Industries. Ecco, ISBN 9780753557525

Name and workplace of bachelor's thesis supervisor:

**Ing. Martin Šipoš, Ph.D. katedra měření FEL (13138)**

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **03.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until:
**by the end of summer semester 2023/2024**

| Ing. Martin Šipoš, Ph.D.<br>Supervisor's signature | Head of department's signature | prof. Mgr. Petr Páta, Ph.D.<br>Dean's signature |
|---|---|---|

## III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

| Date of assignment receipt | Student's signature |
|---|---|

# Abstract

To achieve a successful Thrust Vector Controlled (TVC) launch and potentially a vertical landing, several factors must be considered, including the rocket's structural design, the selection of the appropriate motor and sensors and the development of a flight computer and software. Furthermore, it is necessary to model the rocket's physical behavior and simulate the flight to find and fine-tune the optimal values for the PID controller. However, the process does not end with the launch; data logging during the flight is essential to analyze the rocket's performance and identify areas for improvement. By analyzing the data, one can gain insights into the rocket's stability and response to different conditions.

This thesis presents a comprehensive overview of the necessary steps involved in designing and launching a TVC rocket, including practical demonstrations of the process. The findings demonstrate the usage of TVC for model rocketry and highlight the importance of careful planning and execution in achieving a successful launch. The techniques outlined in this thesis provide practical guidance for those interested in implementing TVC in their projects.

Keywords: Thrust Vector Control, Model Rocketry, 3D Modeling, Flight Simulation, Flight Computer Design, PID Controller, TVC Rocket Launch

# Abstrakt

K dosažení úspěšného startu s vektorováním tahu (TVC – z anglického Thrust Vector Control) a potenciálního vertikálního přistání je třeba zohlednit několik faktorů, jako je strukturní návrh rakety, výběr vhodného motoru a senzorů, nebo vývoj palubního počítače a softwaru. Navíc je nezbytné modelovat fyzikální chování rakety a simulovat let, aby se našly a doladily optimální hodnoty pro PID regulátor. Proces nekončí startem; záznam dat během letu je klíčový pro analýzu letu a identifikaci oblastí pro zlepšení. Analýzou dat lze získat poznatky o stabilitě rakety a její reakci na různé podmínky.

Tato práce představuje komplexní přehled nutných kroků pro návrh a start vektorované rakety, včetně praktických ukázek procesu. Výsledky ukazují využití TVC v modelářské raketové technice a demonstrují důležitost pečlivého plánování a provedení pro dosažení úspěšného startu rakety. Techniky popsané v této práci poskytují praktickou ukázku pro ty, kteří se zajímají o implementaci TVC do vlastních projektů.

Klíčová slova: Vektorování Tahu, Model Rakety, 3D Modelování, Letová Simulace, Návrh Palubního Počítače, PID Regulátor, Start Vektorované Rakety

# Thesis Statement

This bachelor thesis aims to delve into the design and development of a Thrust Vector Controlled (TVC) model rocket. The TVC model rocket is a unique type of rocket that utilizes a control system to alter the direction of thrust, providing stability and control during flight. The following steps will be covered in this paper:

1. Modeling the Rocket
2. Modeling the Thrust Vector Control Mount
3. Three-Dimensional Printing of the Rocket and TVC Mount
4. Selection of Appropriate Sensors and Microcontroller Unit (MCU)
5. Analysis of Optimal Rocket Motor
6. Design of a Custom Printed Circuit Board (PCB)
7. Evaluation of the Most Suitable Operating System
8. Outlining of the Code Structure
9. Design of a State Machine
10. Programming of the Flight Software
11. Simulation of the Proportional Integral Derivative (PID) Controller
12. Tuning of the PID Controller
13. Create a Test Stand for Launches
14. Launch the Rocket
15. Improvement of the Rocket from the launches
16. Analysis of Flight Data
17. Interpretation of Results

# Abbreviations

| | |
|---|---|
| ACS | Altitude Control System |
| CG | Center of Gravity |
| COM | Center of Mass |
| CP | Center of Pressure |
| CRS | Czech Rocket Society |
| FC | Flight Computer |
| FSW | Flight Software |
| IMU | Inertial Measurement Unit |
| ISR | Interrupt Service Routine |
| MOI | Moment of Inertia |
| PCB | Printed Circuit Board |
| PID | Proportional-Integral-Derivative |
| RBFP | Remove Before Flight Pin |
| RCS | Reaction Control System |
| RTC | Real-Time Clock |
| RTOS | Real-Time Operating System |
| TVC | Thrust Vector Control |

# List of Tables

# List of Figures

# Contents

# Chapter 1: Project Introduction

## Introduction

Thrust Vector Control (TVC) is a technology that allows a rocket to change the direction of the thrust produced, which helps in controlling the vehicle's orientation and trajectory during the flight. This means that the rocket does not need fins to stabilize itself. Most of the mode rockets utilize aerodynamic forces to maintain a degree of flight stability.

TVC is commonly used during launch, orbit, and re-entry to make small corrections to the vehicle's path and maintain stability as the fins or any other movable parts of the rocket not producing the thrust cannot be used in upper atmosphere and space. Moreover, TVC is used in some rockets, like SpaceX Falcon 9 for landing capabilities. By adjusting the direction of the thrust during the descent, the rocket can make precise maneuvers to land safely and upright. In this bachelor thesis, I will focus on TVC for model rocketry. The active stabilization is achieved by using gimbaled TVC Mounts, which change the direction of the flight using two servos, each working for one axis.

## Motivation

Thrust vector control is a popular technique in model rocket engineering that enhances the capabilities of solid motor-powered rockets and tackles new challenges. In my thesis, I utilized my theoretical knowledge to design, program, and construct a rocket, while also acquiring new skills such as 3D modeling and PCB assembly.

Furthermore, this work provides a deeper understanding of TVC in model rocketry, which can be implemented in the Czech Rocket Society to develop advanced rockets capable of reaching higher altitudes and potentially landing. The successful control and landing of small-scale rockets serves as a steppingstone for future flights with larger rockets, where factors such as wind gusts, rocket drag, and air density have less impact, resulting in a more robust system. Moreover, continuing this project would provide university students with the opportunity to apply Control Theory in practice.

## Thesis Structure

In *Chapter 1: Project Introduction*, the bachelor thesis is introduced, including the motivation behind it, and the thesis's structure is outlined. *Chapter 2: Rocket Motor* focuses on the selection of the rocket motor and discusses its impact on the overall design and weight of the rocket. *Chapter 3: Thrust Vector Control* explores the design and testing of the gimballed TVC Mount, which plays a significant role in controlling the rocket's direction and stability. It also covers flight dynamics, rocket design, and mechanisms used for rocket stabilization, such as movable fins, grid fins or reaction wheel.

*Chapter 4: Rocket Body* is dedicated to the 3D modeling and printing of the rocket body, considering the challenges posed by weight restrictions, selection of appropriate materials, temperature control, heating bed configuration, layer width, and infill. *Chapter 5: Flight Computer* focuses on the flight computer,

which serves as the brain of the rocket, and explores the selection of sensors and microcontrollers to ensure a successful launch.

*Chapter 6: Flight Software* focuses on the flight software, which utilizes a real-time operating system (RTOS). This chapter covers diverse topics, including task distribution, state machines, and comparisons of different RTOS options. *Chapter 7: System Simulation* discusses system simulation, including the development of a physical model and the utilization of tools like Matlab and Simulink, addressing key factors such as the rocket model and PID control, including its tuning. These simulations play a crucial role in ensuring the success of the rocket.

*Chapter 8: Rocket Launch* explores the details of rocket launches, including preflight checks, the precise execution of rocket launches, and the thorough collection of data from each launch. Moreover, this chapter focuses on the comprehensive analysis and interpretation of the gathered data following the launch. Parameters such as altitude, rotation, and PID response are thoroughly examined to identify potential areas for enhancement in future flights.

The thesis concludes with *Chapter 9: Conclusion*, summarizing the most significant findings, discussing interesting results, and providing a comprehensive conclusion. While achieving a successful launch would be considered a significant accomplishment, the thesis recognizes the limitations of simulations and the variables that could affect the rocket's performance in the real world, aiming for as straight a flight as possible.

## Framework

The rocket was designed using Autodesk Inventor, and PID simulations were conducted in Simulink and tested in Python. Sketches were created using Arduino IDE, while the RTOS was programmed using WSL 2 and VS Code on Windows 11.

# Chapter 2: Rocket Motor

## Rocket Motors

Rockets ascend through propellant combustion and gas expulsion, following Newton's Third Law of Motion, resulting in propulsion based on an equal and opposite reaction, quantified by Newton's Second Law of Motion. While seemingly straightforward, the physics underlying rocket propulsion is in fact highly complex, involving numerous interrelated factors that affect the performance of the rocket motor, such as the shape and size of its nozzle and the amount and composition of the propellant. Additionally, the characteristics of the surrounding environment, such as atmospheric pressure and wind resistance, can also impact the rocket's performance, further underscoring the intricacy of the physics involved.

Solid rocket motors are a crucial component in the space industry, providing the necessary thrust to launch and propel spacecraft into orbit. They consist of a propellant known as a grain, which is usually a mixture of an oxidizer and the fuel itself. You can refer to the image below to visualize the internal structure of the motor. A common oxidizer used is ammonium perchlorate, and one of the commonly used fuels is atomized aluminum powder. While solid rocket motors were the only type used in the early days of rocketry, today they are mostly used as an additional and supportive booster due to their inability to control thrust and reignite the motor. However, they still provide the highest thrust among all types of rocket motors and are used in applications where high power is needed, such as launching heavy payloads into orbit. [1]

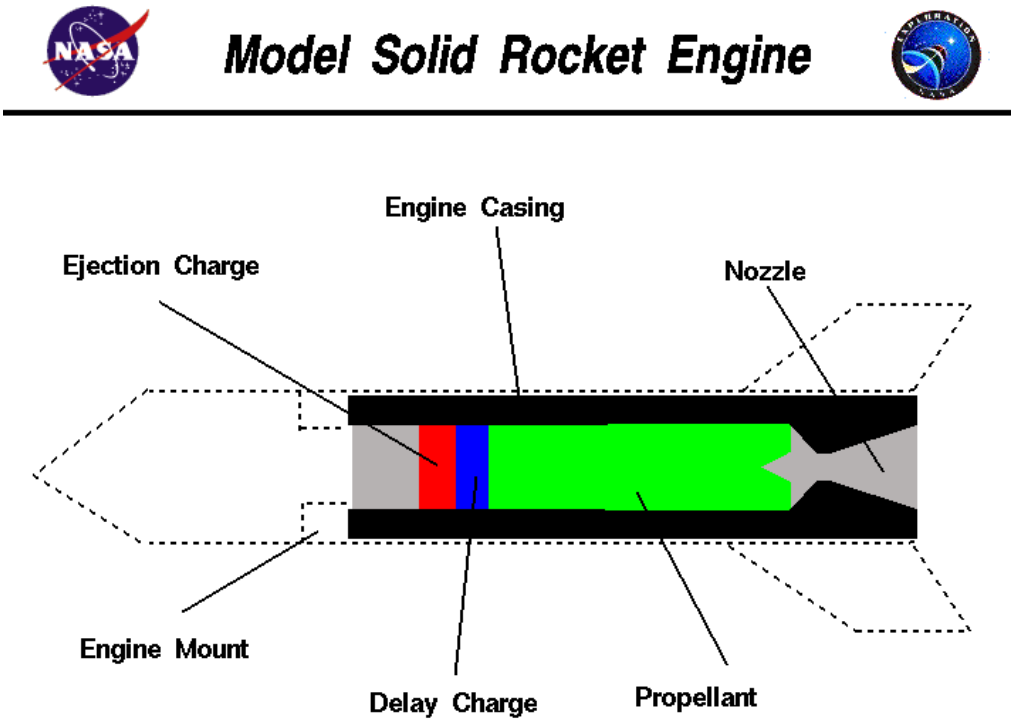


*Figure 1: Structure of a Solid Rocket Motor [2]*

Liquid-propellant motors are a widely used technology in modern launch vehicles as they solve the drawbacks of solid motors. They use liquid oxidizer and fuel, with liquid oxygen and kerosene or liquid

hydrogen as fuel. Recently, liquid methane has become more popular, with motors like SpaceX's Raptor and Blue Origin's BP-1 being powered by it. The main advantage of liquid motors is their ability to reignite the motor multiple times and throttle the motor, accordingly, making them suitable for achieving precise trajectories. However, they have several drawbacks, including the need for significantly more parts, as can be seen in Figure 2. This increases their complexity and the probability of potential failure. Despite this, liquid motors are the preferred choice for larger payloads and missions that require greater control over thrust. [1]

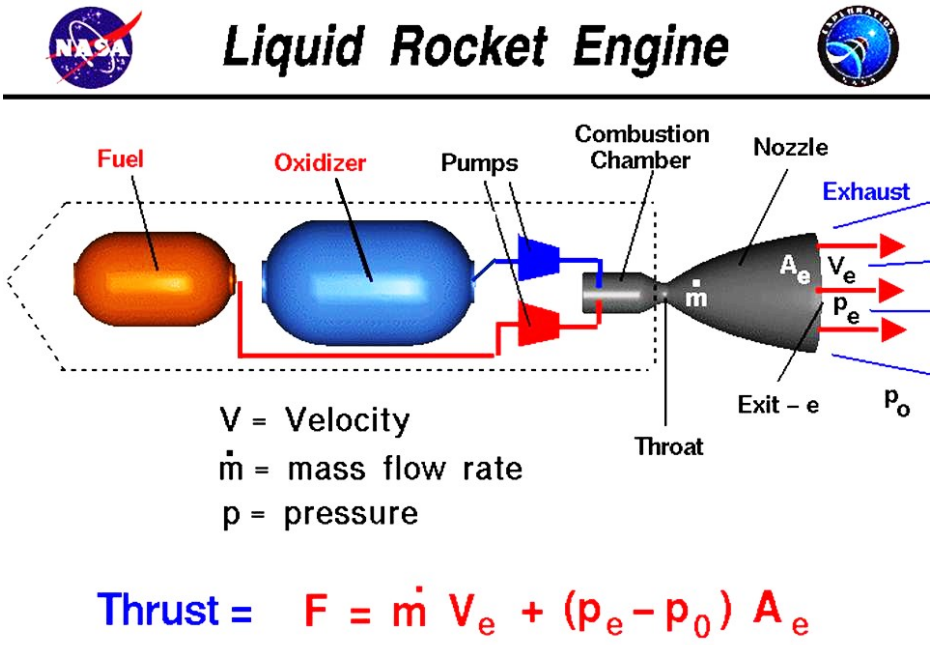


*Figure 2: Liquid Rocket Motor Theory [3]*

Hybrid rocket motors, which combine solid fuel and liquid oxidizer, offer a promising solution by combining the advantages of solid and liquid motors. The simplicity and reliability of solid motors are joined with the efficiency, high specific impulse, and reignition capability of liquid motors. below is an illustration that showcases the internal structure of the motor for better understanding. Specific impulse of a hybrid motor can be comparable to those of liquid motors under certain propellants. Though only small to medium-sized hybrid motors have been used on sounding rockets, several plans to implement them on orbital class rockets have been proposed in recent years. Notably, Scaled Composites, an American company planning to offer commercial spaceflights in the future, has used hybrid motors in spacecraft like SpaceShipOne and SpaceShipTwo. [4]

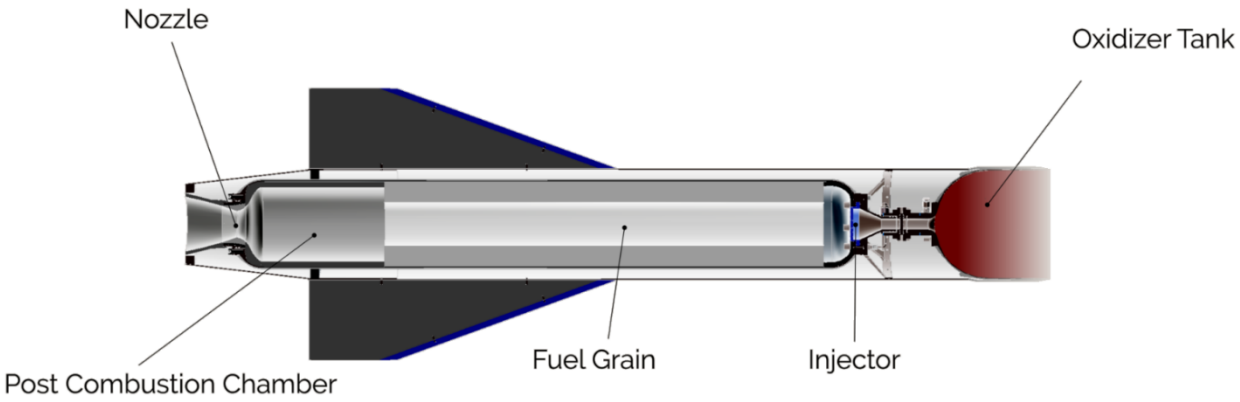


*Figure 3: Structure of a Hybrid Rocket Motor [4]*

## Choosing the Rocket Motor Type

This project aims to create a reusable rocket with a sturdy body capable of withstanding flight. The objective is to maximize liftoff weight and achieve the longest possible burn time, allowing for testing of active stabilization and maximum airtime. Ideally, the rocket will burn as close to the ground as possible, eliminating the need for a recovery system. The project will use simple solid motors instead of liquid-propellant motors due to their complexity and weight requirements.

The four crucial factors to consider when building a rocket are total impulse, thrust, burn time, and size. Total impulse is the total amount of impulse produced by the rocket motor, measured in Newton-seconds. Thrust is the force produced by the rocket motor, which can vary depending on factors such as motor design, propellant type, and nozzle configuration. Burn time is the duration of a rocket motor burn, influenced by motor design, propellant quantity, and other factors. Motor size is typically specified by its diameter and length.

The primary disadvantage of using solid fuel is the thrust curve, which typically has a high initial peak followed by a linear increase or decrease in total thrust. Achieving a more suitable thrust curve for the project requires significant expertise in cooking custom propellant, so the project will use pre-made rocket fuel readily available online. To compensate for the thrust curve, the project will dynamically adjust the PID values throughout the flight or lock the motor during the initial peak.

Although only tubular motors are currently available on the market, the shape of the grain used in a rocket motor can significantly affect its thrust curve, as depicted in the Figure 4. The thrust and chamber pressure generated by a rocket motor are related to the instantaneous burning area, which is determined by the initial shape and restricted boundaries of the grain. The figure below illustrates this concept using contour lines that represent the core shape at different moments during the burn. [5]

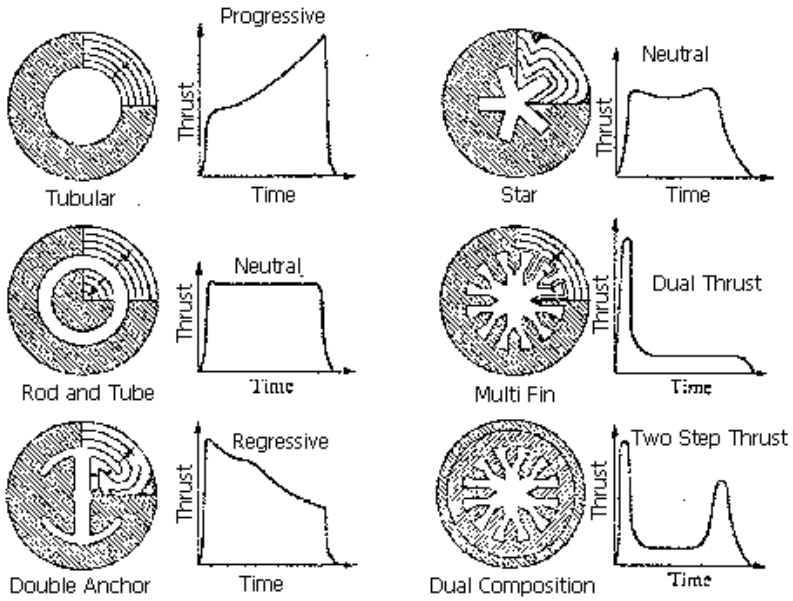


*Figure 4: Effect of Different Shapes of Grain [5]*

The thrust curve may not always follow motor theory, and there are a couple of reasons for this. The first issue is a sharp increase in pressure, which causes sudden combustion that stabilizes. This leads to a rise in thrust. The second issue relates to the fuel mixture used, which can be chemically treated for faster ignition. This might involve coating the inner walls with a substance to guarantee immediate ignition.

## Solid Motor Classification

The choice of a motor is critical since the rocket is specifically designed for the rocket motor dimensions, and it directly affects the rocket's ideal weight. Moreover, if the motor is too powerful, and the rocket reaches a high maximum altitude, I will need to use recovery systems.

Rocket motor construction has interesting naming conventions that describe the motor's properties. The codes consist of several letters and numbers that convey information about the motor's strength, thrust, and delay time for ejection charge. For example, a motor with the code B4-2 is a class B motor with an average thrust of 4 N and a delay time of 2 seconds. If the delay is zero, there is no delay mixture for ejection. Motors with codes ending in "P" are closed to prevent gases from escaping from the front end. They are designed for aircraft models powered by a rocket motor or for rockets powered by motors connected in parallel. The components of the model rocket motor are depicted in the picture below.

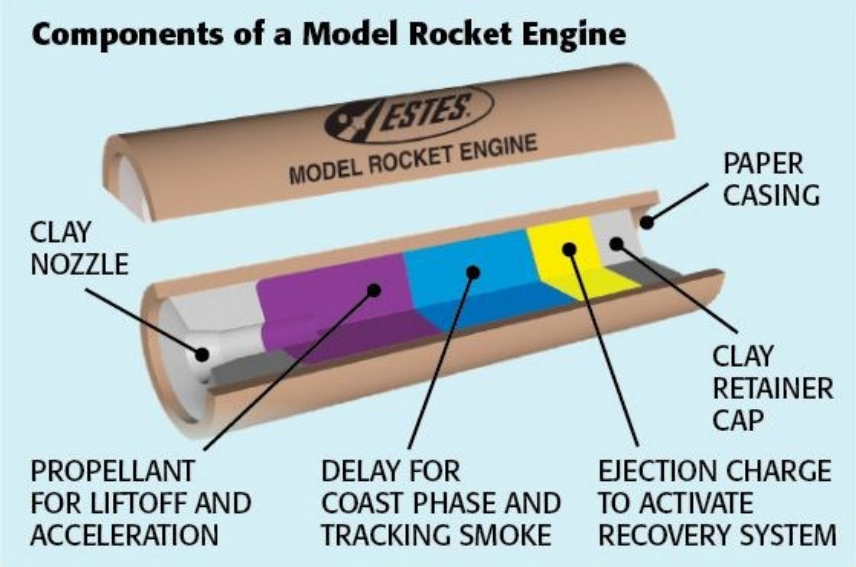


*Figure 5: Structure of the Solid Rocket Motor [6]*

The following are the numbers and restrictions for each motor class: [7]

- Motor class A: Total impulse range of 0.3125 - 0.625 Ns.
- Motor class B: Total impulse range of 1.25 - 2.50 Ns.
- Motor class C: Total impulse range of 5.0 - 10.0 Ns
- Motor class D: Total impulse range of 10.0 - 20.0 Ns
- Motor class E: Total impulse range of 20.0 - 40.0 Ns

I conducted a market analysis of available motors from the Estes and Klima brands. Graphs comparing the motors and their thrust is in Figure 6 and Figure 7. Until recently, it was not possible to order any rocket motor stronger than class D in the Czech Republic. Since I was searching for the most powerful motor with the longest burn time, the Klima Rocket motor D3-P was the most suitable option with no ejection charge. Moreover, students at Aalborg University tested this motor, so I have reliable measurements for simulations. [8] [9]

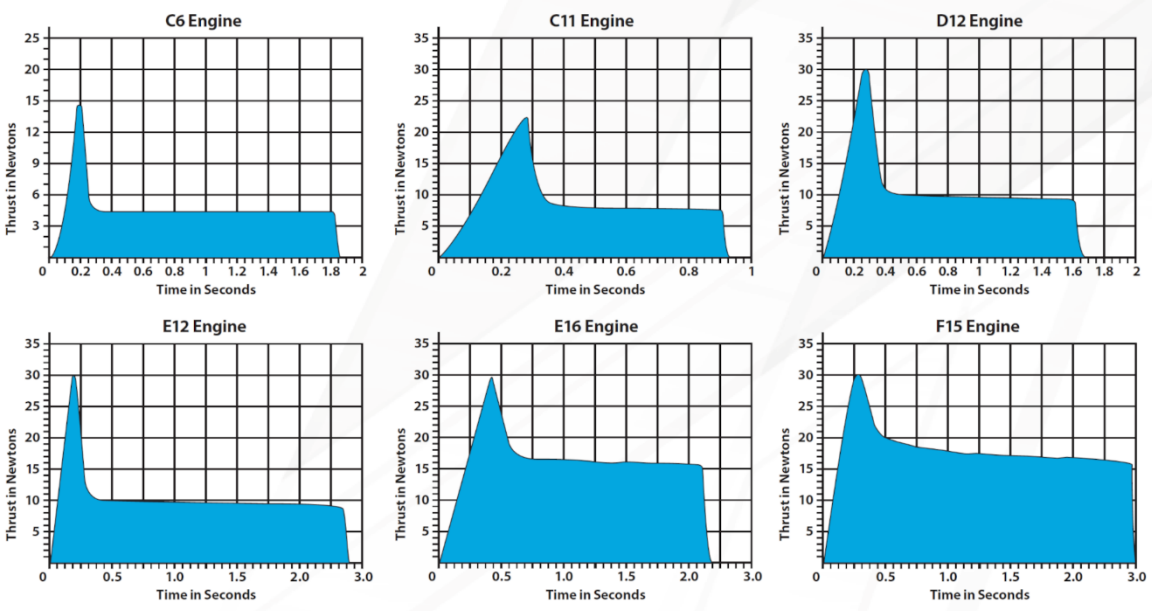


*Figure 6: Estes Rocket Motors Curves [10]*

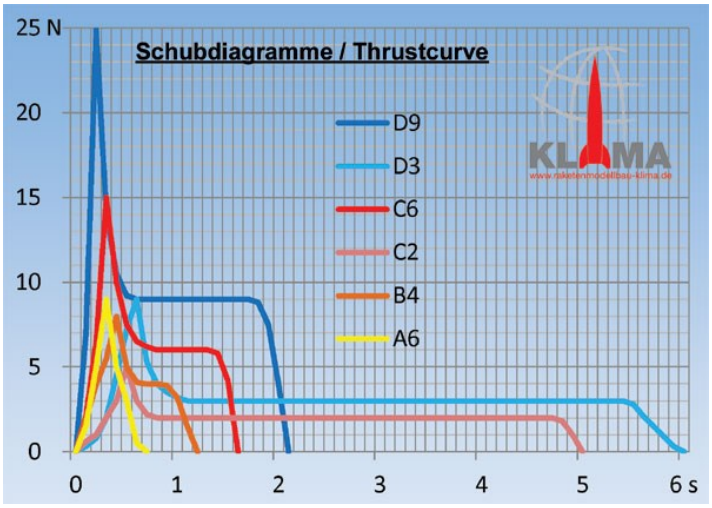


*Figure 7: Klima Rocket Motors Curves [11]*

Although new and more powerful Estes rocket motors have become available since the market analysis was conducted in September 2022, they have a shorter burn time, which is not what I am looking for. Furthermore, choosing a more powerful motor comes with the disadvantage of a larger size, which may require rebuilding the TVC Mount and rocket body if the diameter differs too much.

## Measured Data

These measurements are of utmost importance when simulating rocket behavior, as they allow us to accurately predict the ideal mass and flight altitude, as well as optimize the PID controller. It is worth noting that the thrust curve can vary significantly for each motor, and outside temperature, grain mixture, production imperfections, and other factors can all affect the motor's specifications and performance. Therefore, while the thrust curves obtained from the measurements may be similar, it is important to keep in mind that they may not be applicable to every single motor burned due to inherent differences and variations in motor construction and performance.

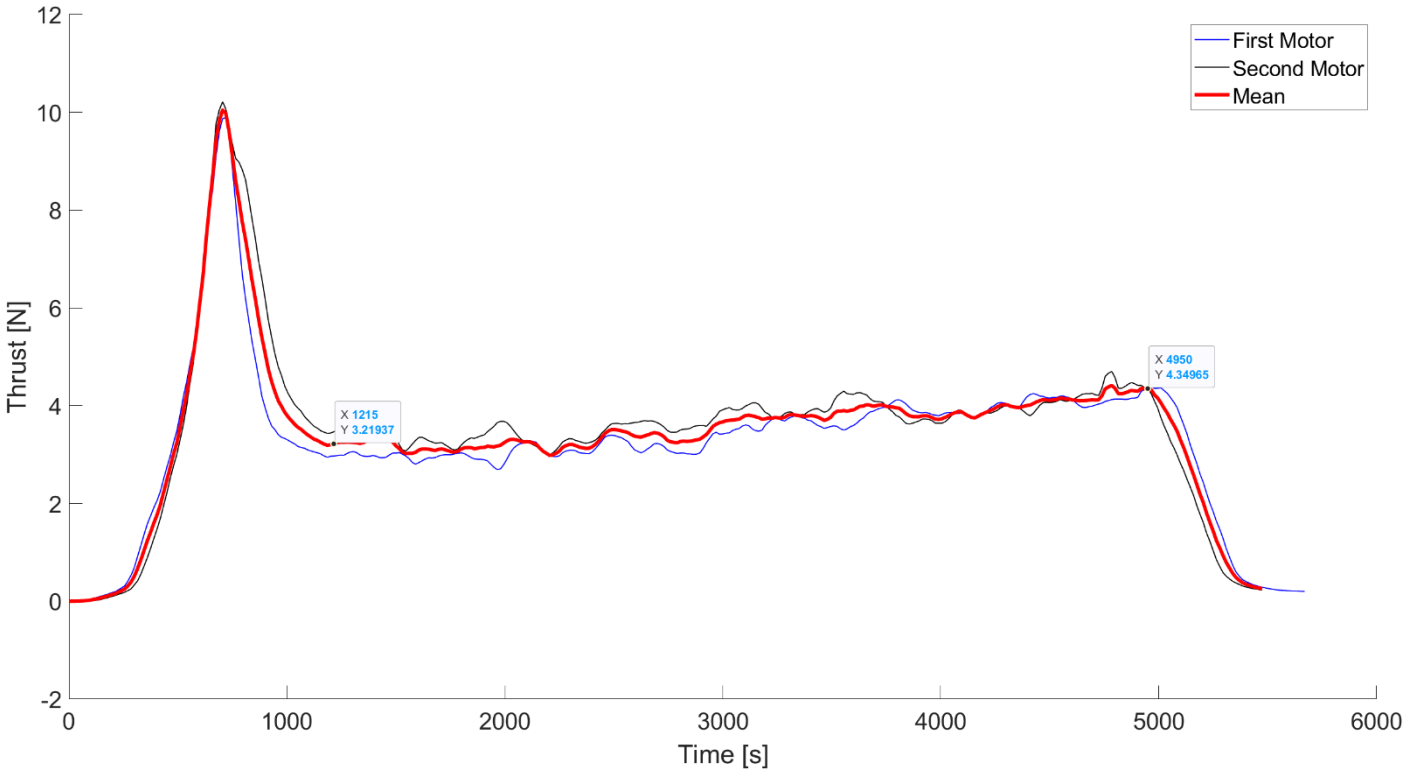


*Figure 8: Measured Thrust Curve for Klima D3-P Rocket Motor [8]*

## Influence of TVC on Vertical Thrust

Before considering the ideal weight that a rocket can lift, I must also take into consideration the decrease in thrust when the gimbaled motor is at its maximum angle. The graph below illustrates the decrease in vertical thrust as a function of the angle.

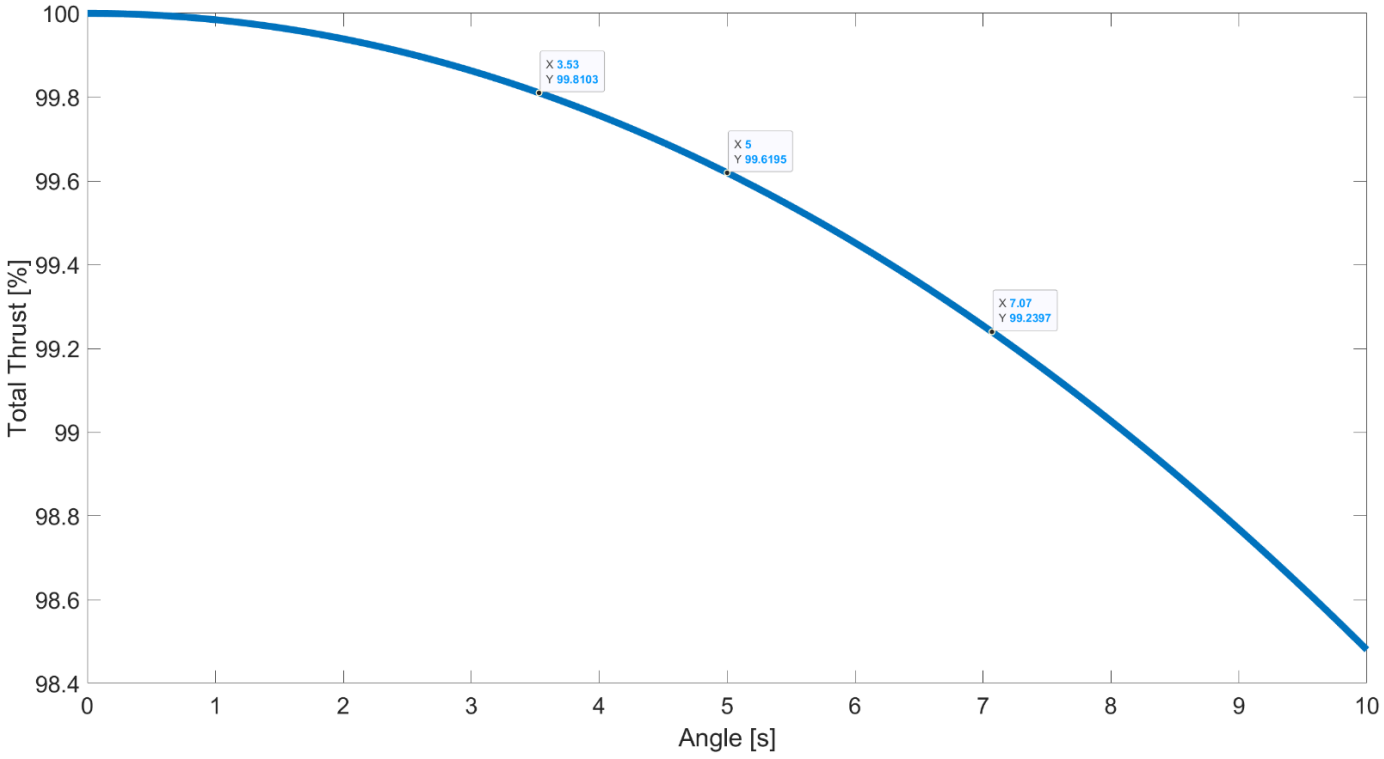


*Figure 9: Decrease of Vertical Thrust Based on the Gimbal Angle.*

Restricting the maximum angle to 5° while maintaining 99.62% of the total thrust strikes a favorable balance between these factors. It is important to note that when treating the axes of the servo separately, the motor movement follows a square pattern rather than a circular one, which may differ from common expectations. To simplify the tuning of the PID control for each axis individually, I will consider them as separate problems. Consequently, with a maximum angle of 5° for each axis, the maximum angle I can reach is about 7.07°. Thus, when both axes are at their maximum angles, the total vertical thrust is reduced to 99.24%.

Furthermore, during the PID control process, each axis is simulated independently. Therefore, if one axis reaches its maximum angle, the other axis remains unaware of this and continues assuming the maximum motor thrust without any reduction. By adopting 5° as the maximum angle for each axis, I can achieve 99.62% of the total thrust. Considering the motor's fluctuation and the absence of factors like air drag, wind, and other forces, any resulting uncertainties should not significantly affect the overall outcome.

# Chapter 3: Thrust Vector Control

## Rocket Stability

A rocket rotates around its center of gravity, and during flight, lift and drag forces act on the rocket's center of pressure.

Center of Pressure (CP) is an imaginary point where you can represent the sum of all aerodynamic forces, such as how one fin, another fin, the tip, the body, etc. turn with a single force. This simplifies the behavior of the rocket and eliminates the need to calculate all the aerodynamic forces individually. Its location can only be considered as a function of the shape [12]

The center of gravity (CG) known also as center of mass (COM) is the point where the weight of the rocket can be concentrated. It is the balance point where the gravitational forces on all parts of the rocket are equal and opposite. [13]

The location of the CG is important because it determines the stability of the rocket in flight. If the center of gravity is too far forward, the rocket will be nose-heavy and may become unstable. If it is too far back, the rocket will be tail-heavy and will also be unstable. Therefore, the center of gravity must be located within a certain range to ensure that the rocket is stable and flies straight. [14]

The stability of the rocket is determined by the relative positions of the center of pressure and center of gravity. A restoring force exists when the center of pressure is below the center of gravity, which causes the rocket to return to its initial position after any small displacement. This stability can be tested by swinging the rocket around with a string tied at the center of gravity. Engineers can adjust the stability by modifying the center of pressure or center of gravity, for example by increasing the fin area or adding weight to the nose.

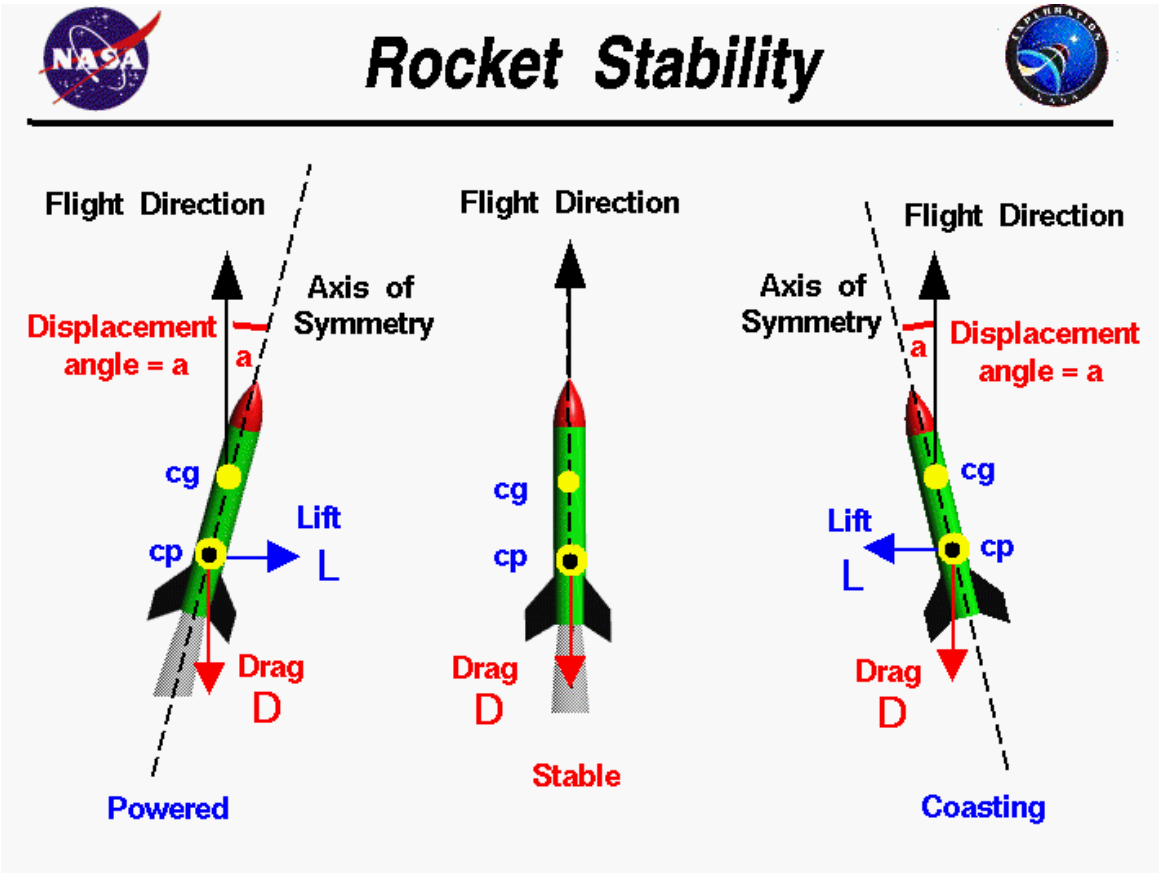


*Figure 10: Importance of Setting Up the CG and CP Points Correctly [15]*

While earlier rockets and model rockets followed this rule, advances in computer and autopilot technology in the second half of the 20th century allowed for rockets to be steered even if they were aerodynamically unstable. The Saturn V rocket is an example of this, with its CG located below the CP, as illustrated in Figure 11. The rocket has small fins located at its bottom, which serve to reduce instability in the event of motor gimbal failure. This is necessary to ensure that the structural load remains within safe limits and allows the crew enough time to act and ensure their safety. [16] For descent, the CP should be above the CG to ensure stability if there are no control surfaces or powerful motors to correct instability. This may require structural adjustments or active control actuators to provide stability during descent. [17]

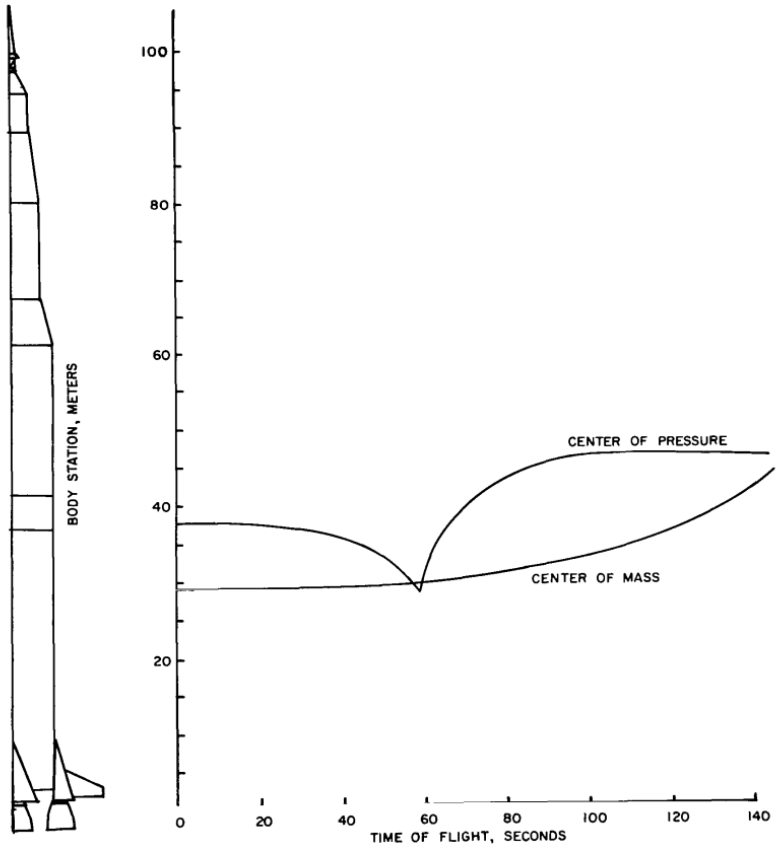


*Figure 11: Variations in the Center of Pressure and Center of Mass During Saturn V Flight [18]*

In our situation, incorporating fins into the rocket design would increase the bottom surface area, which, coupled with the lightness of the rocket, would make it challenging to control in windy conditions. Furthermore, since the rocket is in a hover state for most of the flight, the presence of fins would have minimal impact during the stable phase of the thrust curve. However, the absence of fins during the initial peak of the flight would cause instability, resulting in the center of pressure moving above the center of gravity and making it difficult for the rocket to fly straight. This is particularly crucial since tuning the PID controller for the frequently changing thrust curve during the initial peak with the TVC locked can be a daunting task.

## Rocket Control

Since the rocket is unstable as mentioned before, I have several systems that can be used to maneuver the rocket during flight, depending on the design and mission requirements. Different examples how to control the rocket are shown in Figure 12. Early rockets flied on movable aerodynamic surfaces, while later rockets designed for outer space employed small vanes in the nozzle exhaust to alter the direction of thrust. Modern rockets, such as the Saturn V moon rocket and the Space Shuttle, utilize a system called gimbaled thrust. With this system, the exhaust nozzle of the rocket can be swiveled from side to side, altering the direction of thrust relative to the center of gravity of the rocket and thereby enabling controlled movement. To stabilize the rockets, there are many different approaches.

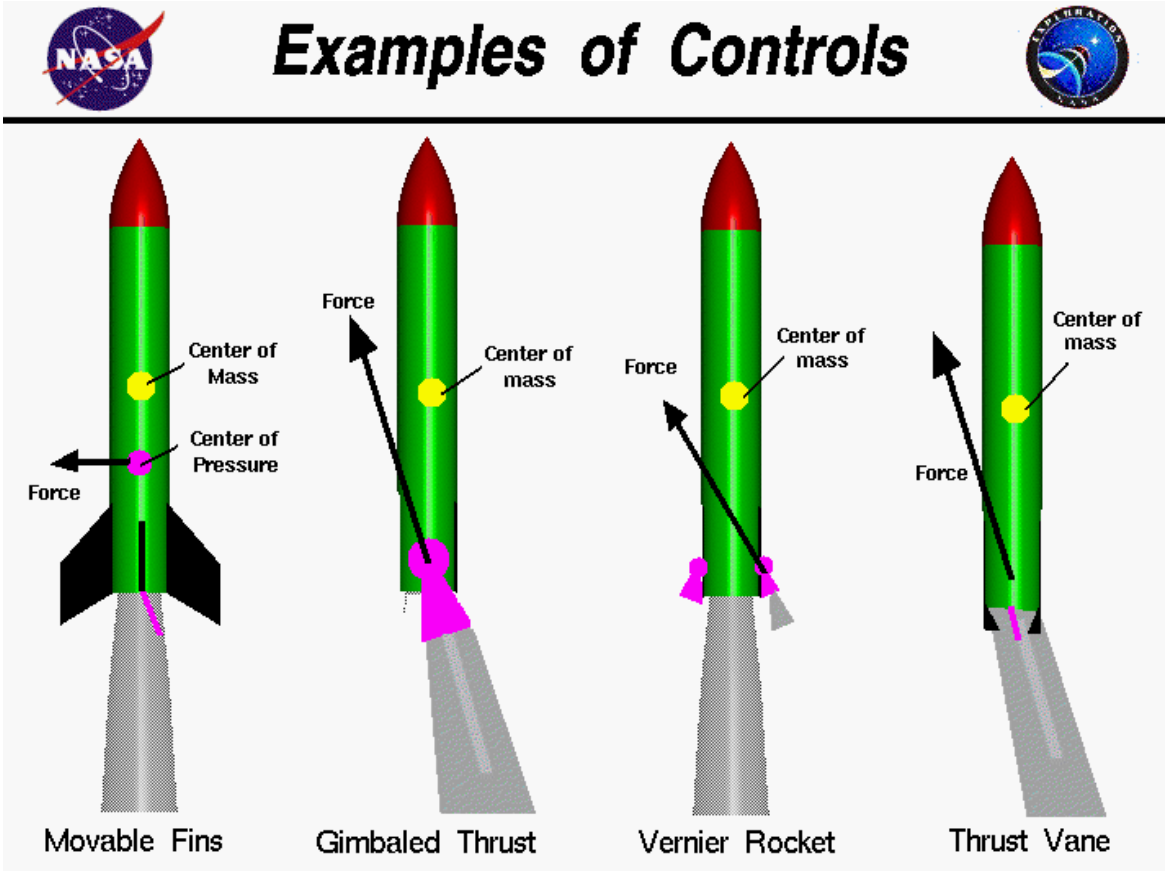


*Figure 12: Examples of Rocket Control [19]*

### Movable Fins

I can have movable fins, which I can adjust to maintain a straight flight, but they cannot be used in a space and decrease their effects in the less rigid parts of atmosphere.

### Thrust Vector Control

The physics behind TVC involves the manipulation of the rocket's CG and CP. By altering the direction of thrust vectoring, the CG and CP can be shifted to create a more stable flight path. This is achieved by moving the motor thrust in a specific direction, which generates a moment that acts on the rocket's mass. This moment causes the rocket to rotate, altering its trajectory. Additionally, TVC can be used to counteract forces acting on the rocket during flight, such as crosswinds or turbulence, by redirecting the thrust vector to compensate for these external factors.

### Grid Fins

Nevertheless, controllable fins cannot be placed at the base during descent because the CP would be below the CG. Foldable grid fins at the top of the first stage are used by SpaceX to stabilize and control the rocket during landing. Grid fins are specifically designed for supersonic and hypersonic flights and experience smaller atmospheric forces compared to planar fins. [17]

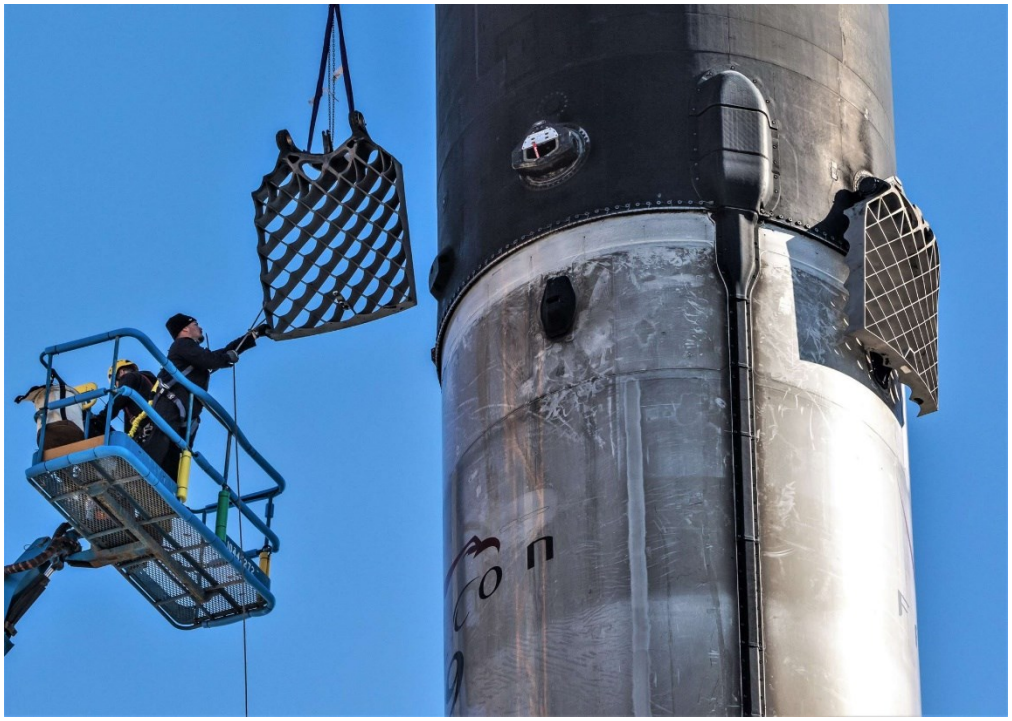

*Figure 13: Grid Fins in Falcon9 Rocket [20]*

### Reaction Control System

The grid fins are not effective for low speeds and landing. Instead, a viable option for maintaining stability during this phase is to utilize RCS (Reaction Control System). RCS is an actuator system that provides precise attitude control in a vacuum environment. It offers the ability to adjust the rocket's roll, complementing the pitch and yaw control provided by TVC. While RCS has traditionally been employed in spacecraft's Attitude Control System (ACS), SpaceX has also implemented it in their boosters. [17]

RCS comprises small thrusters that can be toggled on or off. In some cases, PWM (Pulse Width Modulation) can be used to enable thrust adjustments, like thrust control. During rocket landings, RCS plays a crucial role as stabilizing actuators. Positioned on each side of the upper section of the booster, they render the bottom jets unnecessary due to the presence of the main motor. An example is seen in Falcon 9 rocket's first stage, where the RCS cold gas nitrogen thrusters are fired for a "flip maneuver" to reorient the vehicle before initiating the descent. [17] You can observe this maneuver in the picture below.

*Figure 14: Falcon9 RCS during the Flip Maneuver [21]*

### Reaction Wheel

Another commonly employed option for controlling the roll angle in model rockets, which has not been mentioned yet, is the use of a reaction wheel. This technique involves spinning the reaction wheel on the ground to accumulate energy, which can then be harnessed to adjust the roll by either slowing down or speeding up the wheel's rotation. Due to its lightweight nature and ease of implementation, it has become a popular method for addressing roll angle control in model rockets that primarily rely on TVC for control. The reaction wheel offers an additional approach to finely adjust and stabilize the rocket's flight trajectory, providing an extra tool for controlling roll angle in model rockets that primarily rely on TVC.

*Figure 15: Fully Assembled Reaction Wheel [22]*

In conclusion, there are several techniques available for maintaining rocket stability. However, due to weight limitations, I will initially rely solely on TVC. Since the flight's airtime and maximum velocity are

low, the risk of experiencing rapid roll is minimal. Furthermore, both the reaction wheel and RCS are too heavy for our rocket. Although a reaction wheel would be a prudent choice in situations with high roll speeds.

## Design

The design of the gimballed TVC was inspired by various projects showcasing different approaches, some of them are listed below. After conducting a detailed analysis, I chose the approach with movable axes positioned in the middle of the motor, as it has been proven to effectively control the rocket by many individuals. This reliable mechanism was selected based on its successful track record.

Many of the available mounts are specifically designed for stronger motors in the E or F class and for rockets with a diameter of 74mm. However, since my rocket has a smaller diameter of 66mm, one of the most challenging tasks was to design a custom TVC mount that could accommodate the smaller size while still achieving the desired motor angle range.

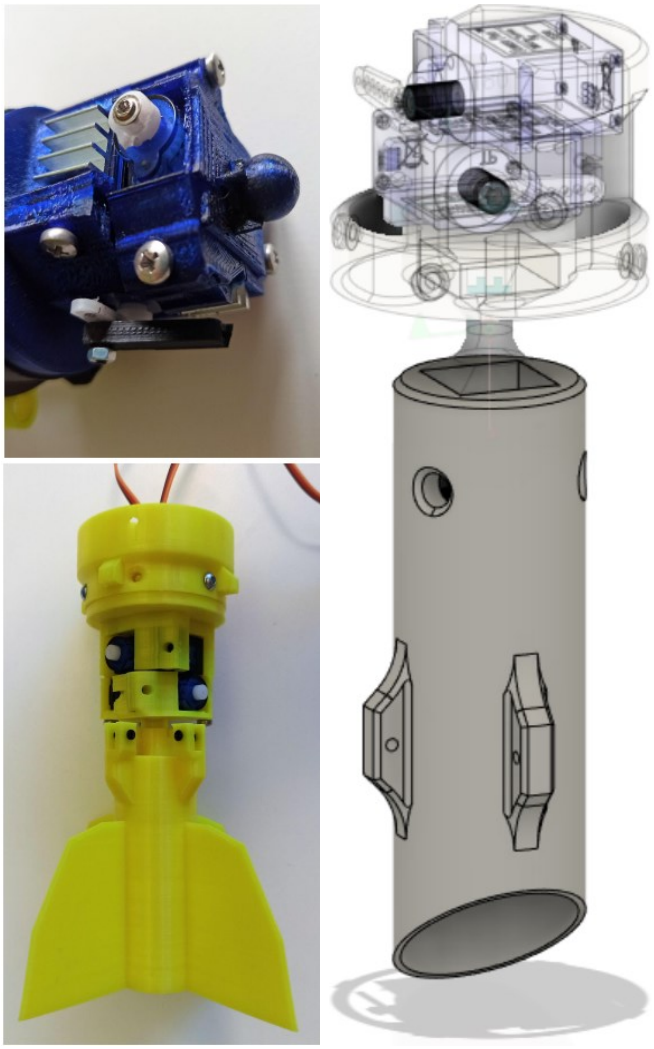

*Figure 16: Option 1: Using a Ball Joint [23] [24]*

This idea involves utilizing a ball joint with an attached servo. The advantage of this method is that the motor and servo are separate entities. However, it is worth noting that this setup is susceptible to damage and the 3D printing process for the ball joint can be more complex.

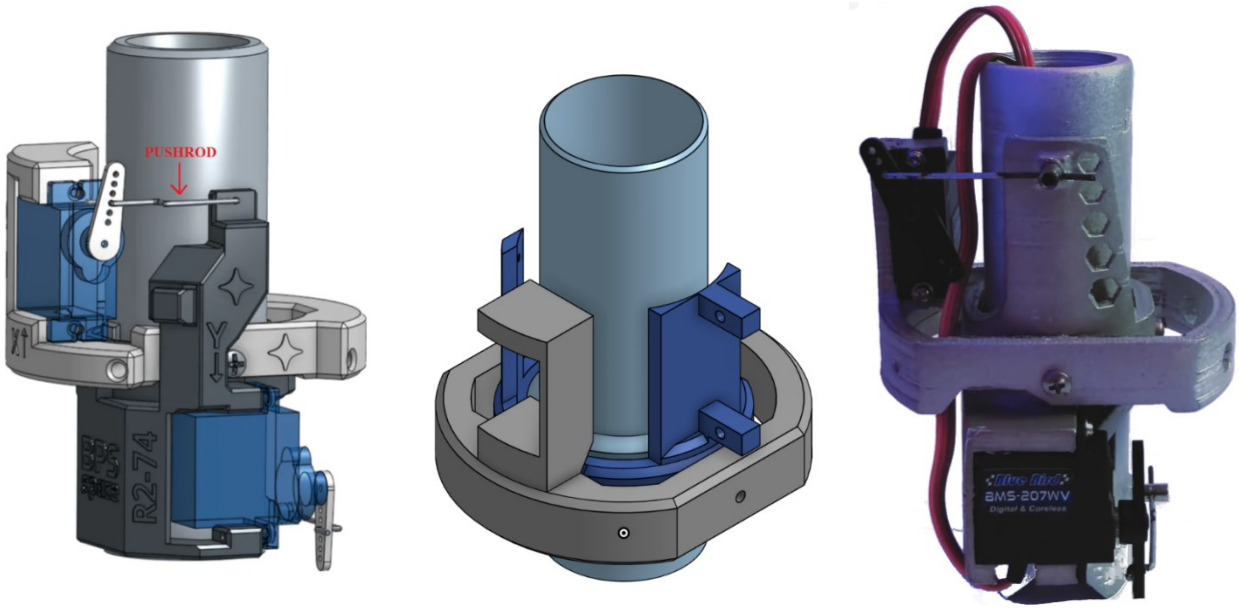


*Figure 17: Option 2: Using Movable Axes in the Middle, the Left Mount is from BPS Space [25], the Middle is from Moltech Space [26], and the Right Mount is from Delta Space Systems[27]*

In my design, I made a few adjustments. Firstly, I positioned both servos in the upper part to minimize the risk of material melting caused by the hot temperatures generated by the burning rocket motor. Additionally, the inner gimbal was designed to allow easy adjustment for a motor with a slightly larger diameter, providing flexibility in case the current motors prove to be insufficiently powerful.

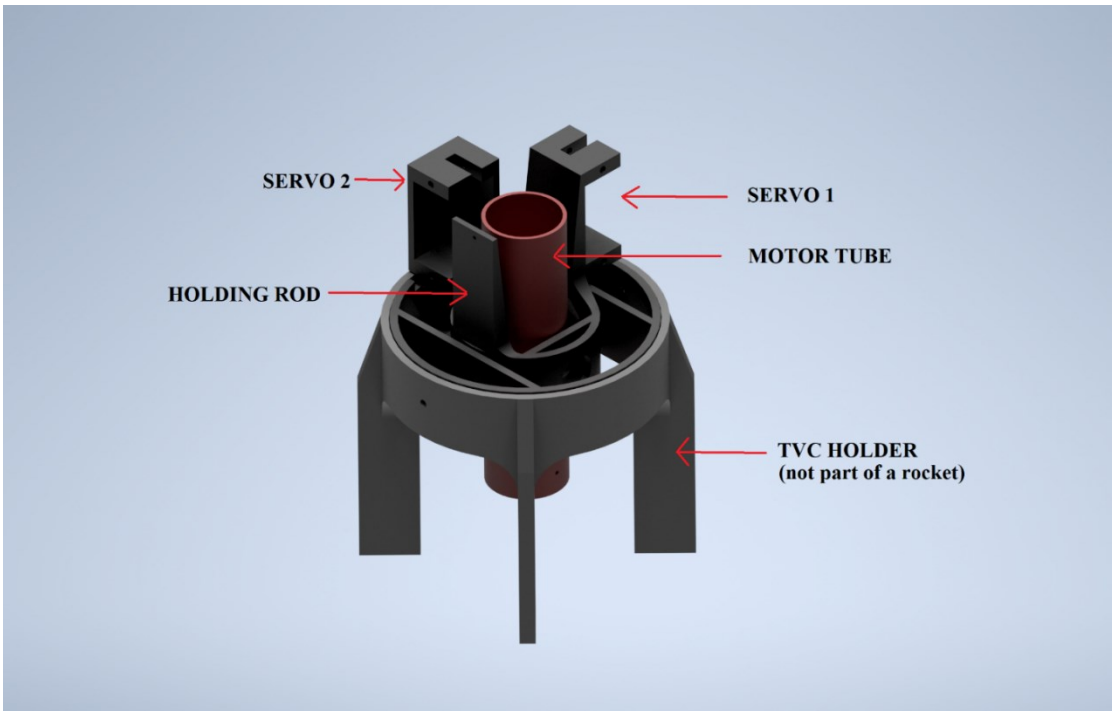


*Figure 18: New TVC Gimbal Design*

## Development

Developing my own TVC mount without any prior experience in 3D modeling software proved to be a more challenging task than expected. While there were many available mounts online, none of them met

all my requirements in terms of angle reach, design, and motor diameter. Additionally, all the 3D models I found were only available as STL files, which meant they were not editable.

Therefore, I had to design a brand new TVC mount that would be the centerpiece of the entire rocket. In total, I went through six iterations, with each one being printed and built to optimize every aspect of the design.

From the outset, I knew that the rocket diameter was 66mm as I was limited by the size of the Teensy microcontroller. Any extra millimeters in the design would result in a heavier rocket, which went against our main goal of making it as light as possible to use the Klima D3-P Rocket Motor, which dictated the motor size.

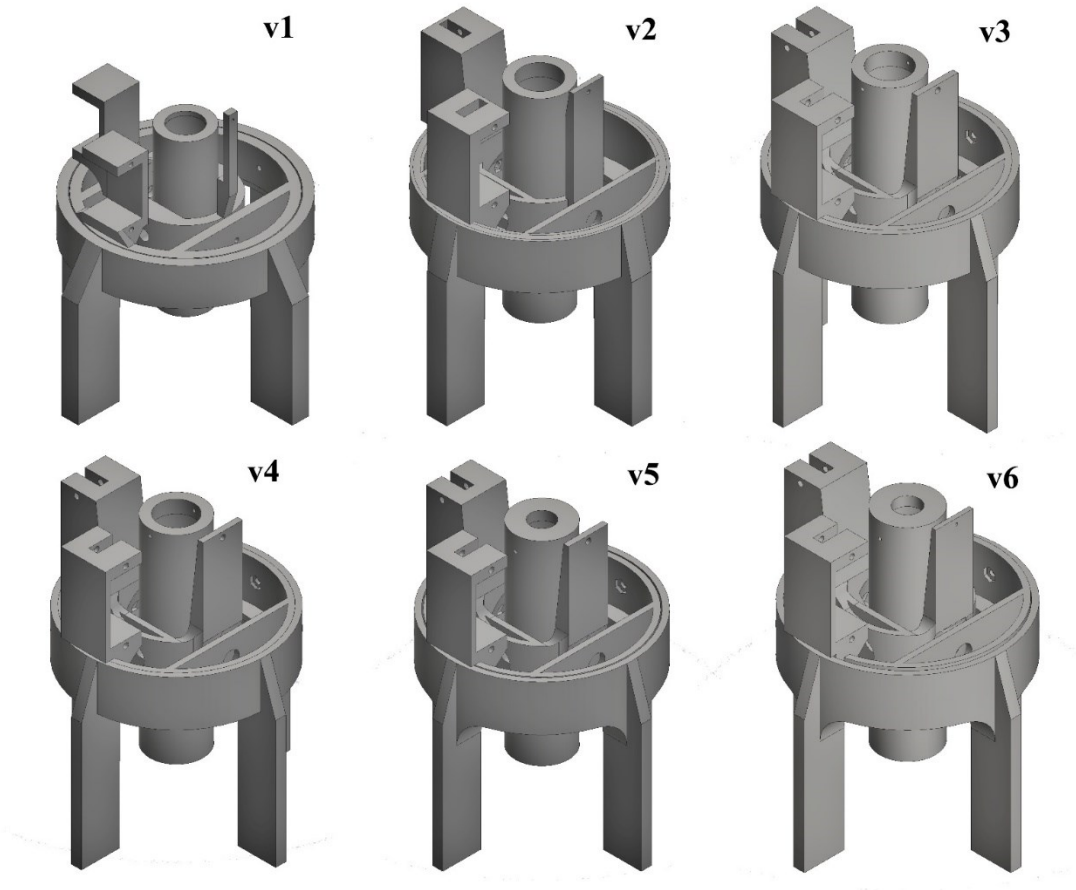


*Figure 19: The Evolution of My TVC Gimbal Design*

The initial stages of TVC development involved a process of testing and trial. In version 1, there were various issues discovered, including incorrect measurements where the motor tube was smaller than the motor itself, overly thick walls, weak servo, and pushrod holders that broke easily, incorrectly sized holes for bearings and nuts, misaligned screw holes, and wrong sizes for servo holders. These issues were addressed and fixed in version 2, with a focus on precise gap measurements, particularly between the inner and outer gimbal and the size of bearings.

From version 4 onwards, the focus shifted towards optimizing gaps for screws, nuts, bearings, and the servo offset to achieve the best position for the pushrods. It was crucial for everything to fit perfectly after printing, as any unnecessary gaps would result in dead zones where movements were imprecise. Some

screw holes were also found to be misaligned. In model version 6, a gap was added to prevent the collision between the servo in the inner gimbal and the ring. Additionally, the top of the motor was reinforced to prevent potential disaster if this part were to break during motor ignition, which could potentially damage the flight computer if the motor became trapped inside the rocket body. Furthermore, the nuts spot from the inside of the motor was removed, as it proved challenging to place the nut correctly, and future iterations aimed for a nut-free design.

The main challenge encountered towards the end was the servo movement, as it extended beyond the designated mount area. To address this, the rocket body was redesigned, with increased thickness above the TVC mount to allow more space for the servos to move freely.

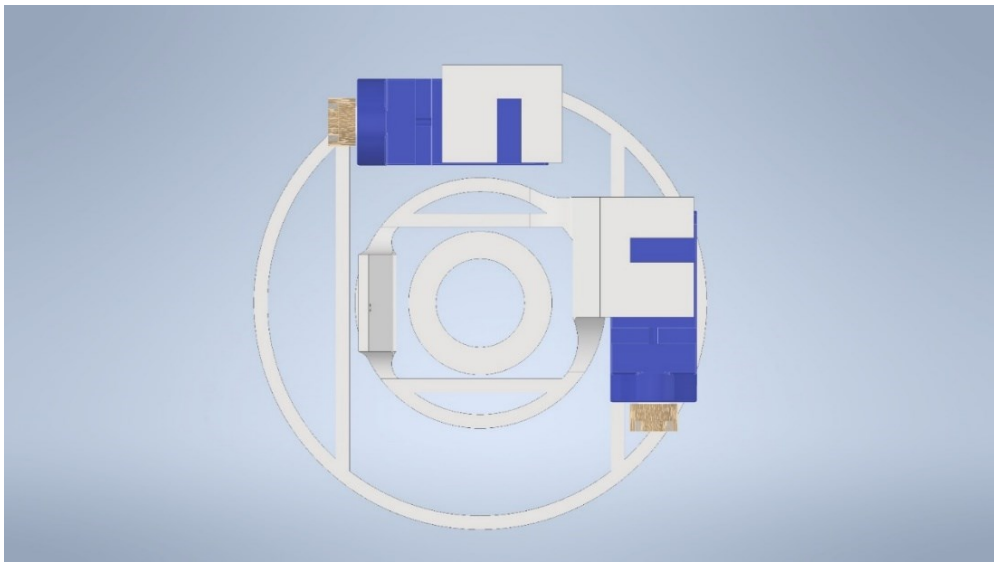

*Figure 20: Top View of the TVC Mount*

While the TVC design is now suitable for testing, ongoing experiments are being conducted to optimize the gap and 3D printing settings for the best possible outcome.

## Maximum Motor Reach

An essential factor to consider is the range of the motor, as it directly impacts the TVC's ability to correct the rocket's flight. The larger the maximum reach, the greater the misalignment of the rocket angle it can correct. The range was determined computationally using the 3D model, which enabled precise measurements between all the components.

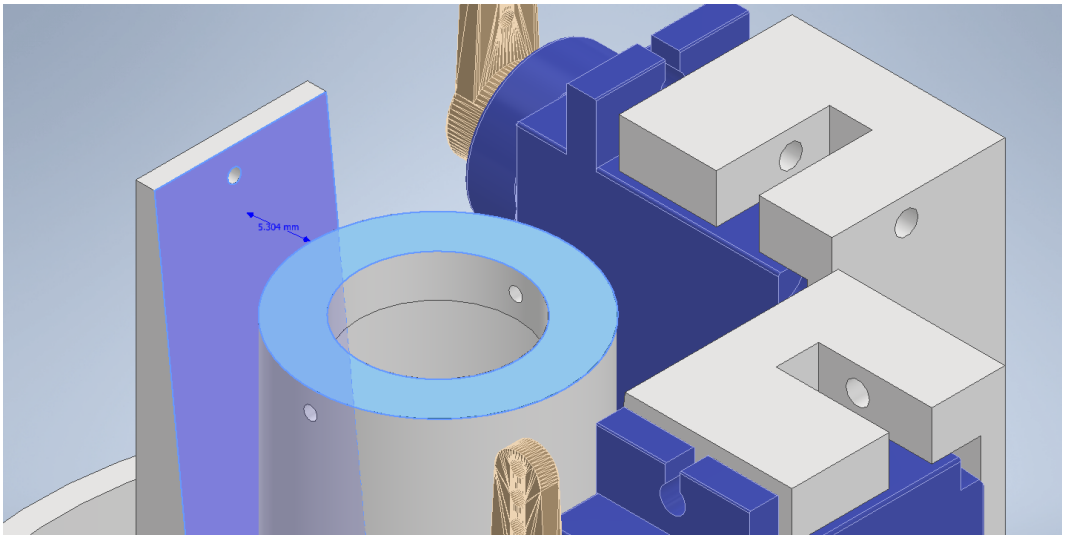


*Figure 21: Measuring the Minimal Distance Between the Motor Tube and the Holder for the X axis.*

During the computation, I focus on determining the smallest reach among all axes, as I simplify the motor's reach to be uniform for each axis.

In the X-axis, there are two parts that limit the motor's range: the servo itself and the axis holder. By knowing the motor's size (38.25mm) and the minimum distance between the motor tube and the obstacle, along with the triangle's other side dimensions, I can apply trigonometric functions to calculate the desired angle. The holder enables a maximum reach of $\alpha$=7.9663°, while the servo, positioned further away, allows for a total reach of $\beta$=9.6224°.

In the Y-axis, the servo itself acts as one limit, while the other side depends on the servo's maximum angle. There are no obstructions to the movement. The reach on this side is dependent on calibration, considering factors such as the servo arm insertion and the default position for the motor's upright orientation. Hence, no specific results can be provided for this side. However, on the opposite side, as shown in the picture, I measured and calculated a maximum reach of $\gamma$=15.38°.

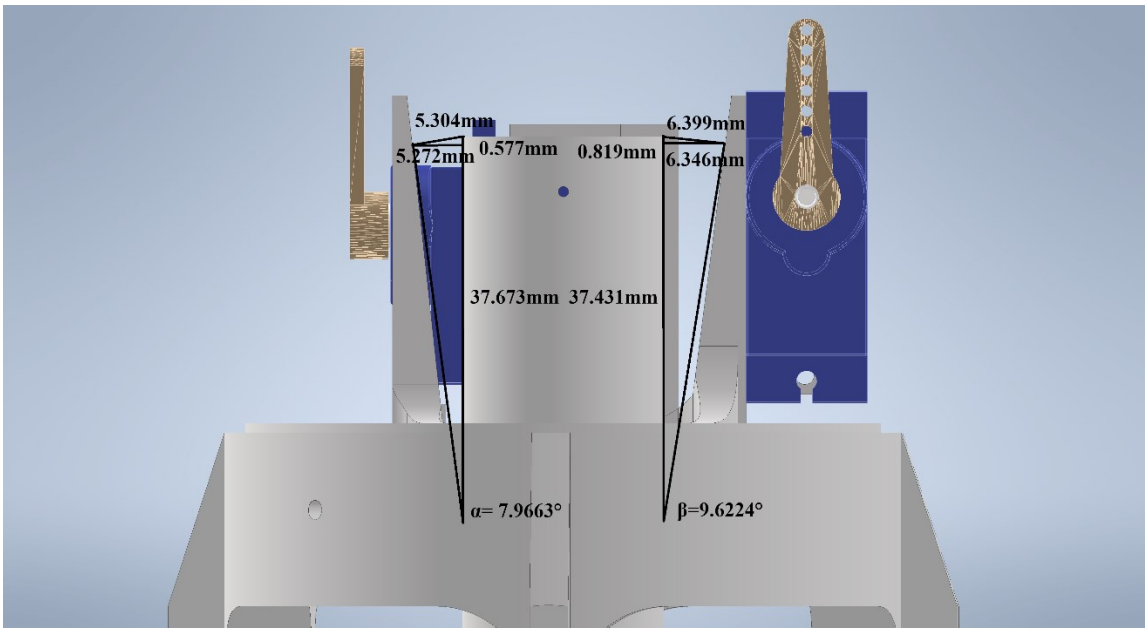


*Figure 22: Dimension for Servo 's X Axis*

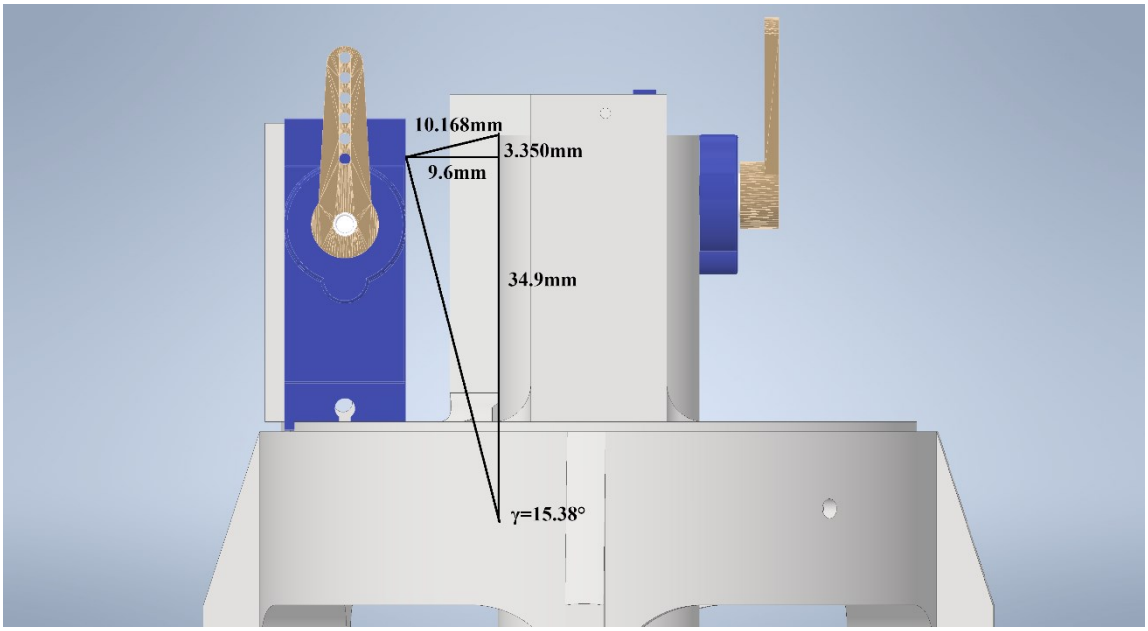


*Figure 23: Dimension for Servo 's Y Axis*

To summarize our findings, the servo can achieve a maximum reach of 7.97° in each axis, which fulfills our requirements. However, when using TVC inside the rocket, I need to consider potential constraints associated with the rocket's diameter. If the diameter of the rocket obstructs the servos, I can adjust the pushrod size to accommodate and modify the maximum servo reach. Nevertheless, this adjustment may lead to a reduced reach. As depicted in the picture, the servos are larger than the diameter, and to overcome this issue, I increased the diameter in this specific part of the rocket.

## Servo Transfer Function

As discussed in the Influence of TVC on Vertical Thrust, I have limited the maximum angle for a single axis to 5°. However, the servo can be moved from 0° to 180°. Therefore, I need to determine the relationship between the servo movements and the actual movement of the motor.

A simple approach would be to calculate the number of degrees the servo can move from its default position (which is when the motor is perfectly centered) to the maximum angle, and then create a linear function to represent how the motor angle changes based on the servo movement. However, since the servo moves along a circular path instead of a straight line, this function is not perfectly linear. Hence, I need to measure it to find the appropriate transfer function.

In the Figure 24, I can observe that y, m, and n are fixed, and the length of x represents the size of the pushrod (using trigonometry to eliminate the fourth dimension). Only the length of z varies. Initially, I need to calculate the default angles of the servo and the motor. Once the servo starts moving, I can calculate the size of z and then the motor angle using the cosine law.

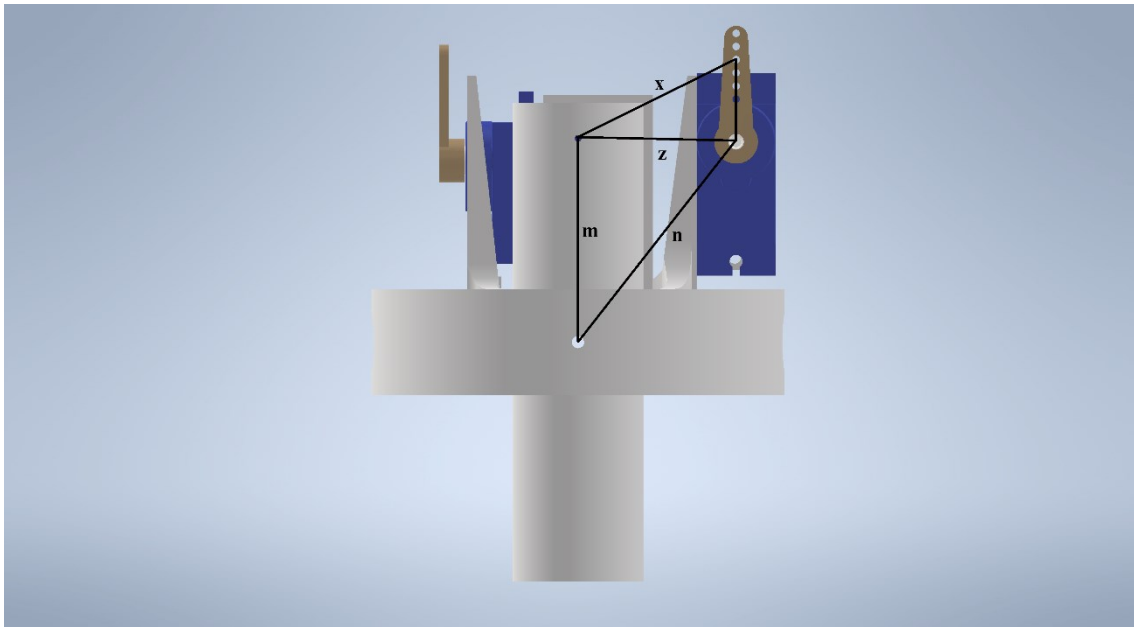


*Figure 24: Sketch of the Variables Used in the Script.*

The used parameters for the TVC model can be seen in Table 1:

| Parameter | X Axis | Y Axis |
|---|---|---|
| x | 20.9399 | 24.829 |
| y | 10.8 | 10.8 |
| n | 39.646 | 38.208 |
| m | 34.934 | 39.55 |
| z | 25.249 | 29.403 |

*Table 1: Parameters to Calculate the Transfer Function*

For the parameters above, the results are as follows:

- X axis: -0.0286x2 + 3.0245x – 0.0025
- Y axis: -0.0344x2 + 3.2925x – 0.021

As I can observe, the function is not linear. However, the deviation is not as substantial as anticipated.

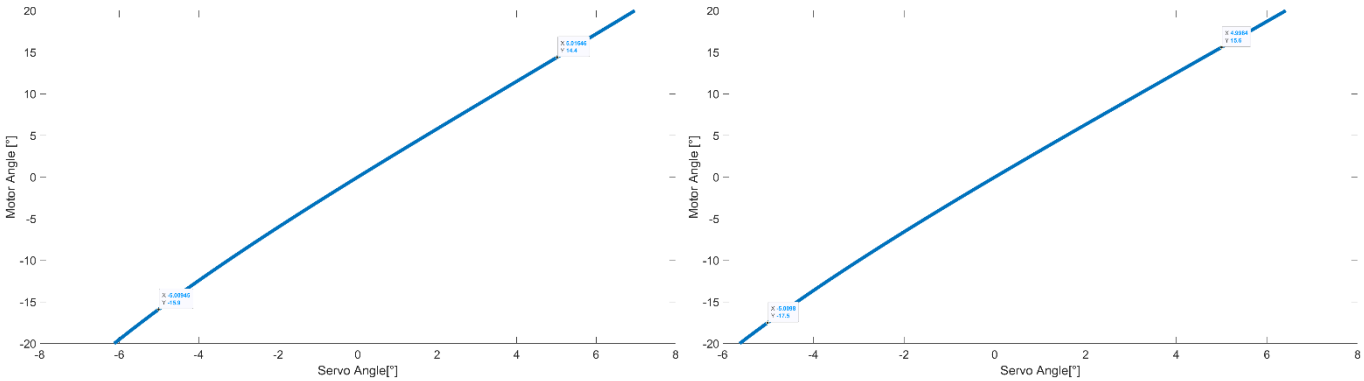


*Figure 25: Transfer Functions for X Axis (left) and Y Axis (right)*

## Improvements

Although the current model of the TVC Mount is suitable for the testing phase, there are several areas that could be improved to achieve a perfect design. By addressing these issues and implementing the suggested improvements, the TVC Mount can be optimized for better performance, functionality, and ease of assembly. These improvements can be summarized as follows:

- Increase the gap for the servo in the outer ring: By expanding the space in the outer ring, the servo can fit properly.
- Increase the space in the outer ring for the outer servo: The current design does not provide sufficient room for the outer servo. By expanding the space in the outer ring, the servo can fit properly, eliminating any issues with misalignment or restricted movement.
- Increase the space in the outer ring for the outer servo: The current design does not provide sufficient room for the outer servo. By expanding the space in the outer ring, the servo can fit properly, eliminating any issues with misalignment or restricted movement.
- Make the inner gimbal wider or add counter piece to the bearings. Now it touches also on the outer ring, resulting in the fact that the entire bearings are moving, not only the inside part.
- Optimize the holes for pushrods to use a new mechanism: The current mechanism for the pushrods may experience excessive friction.
- Reduce the gap for bearings: The existing gap for the bearings is too large, allowing them to move slightly. To enhance stability, it is advisable to decrease the gap so that the bearings are securely held in place and do not experience unintended movement.
- Eliminate the use of nuts and use friction-based holes instead: The current design includes small nuts holes, but the nuts are challenging to position accurately during assembly. To simplify the assembly process and ensure a precise fit, it is recommended to replace the nuts with holes where the screws can be securely fastened using friction.

# Chapter 4: Rocket Body

## Rocket Parts

I have previously discussed the most complex component of the rocket, the TVC. However, the rocket remains incomplete without its body. Before I delve into the details of the model and the 3D printing procedure, it is necessary to discuss the terminology used for this rocket model.

The rocket comprises several key components, namely:

- Rocket Body: The distances between the TVC, center of mass, and flight computers significantly impact PID tuning and rocket stability.
- Flight Computer: The flight computer serves as the brain of the rocket, enabling active control and overseeing all servo movements.
- TVC Mount with Servos: This movable part is responsible for adjusting the direction of the motor thrust, enabling rocket control.
- Rocket Motor: For our specific requirements, I have selected the Klima solid Motor D3-P, which has a total burnout time of 6 seconds.
- Stand (also known as the Launching Pad): This structure securely holds the rocket prior to liftoff. I used the rocket screws on the outside to properly insert the rocket into the stand.

The rocket body consists of five distinct parts:

- Cone: The top section of the rocket must be aerodynamically shaped to minimize drag against the air. To gain a better understanding of the rocket's behavior at higher altitudes, software such as OpenRocket [28] can be utilized.
- Upper Tube: This section serves as the mounting point for the Flight Computer.
- Middle Tube: Its purpose is to increase the overall height of the rocket.
- Bottom Tube Up: This part encompasses the servo and must be wider than the rest of the rocket to allow unrestricted servo movement within the TVC Mount.
- Bottom Tube Down: This section accommodates the TVC Mount.

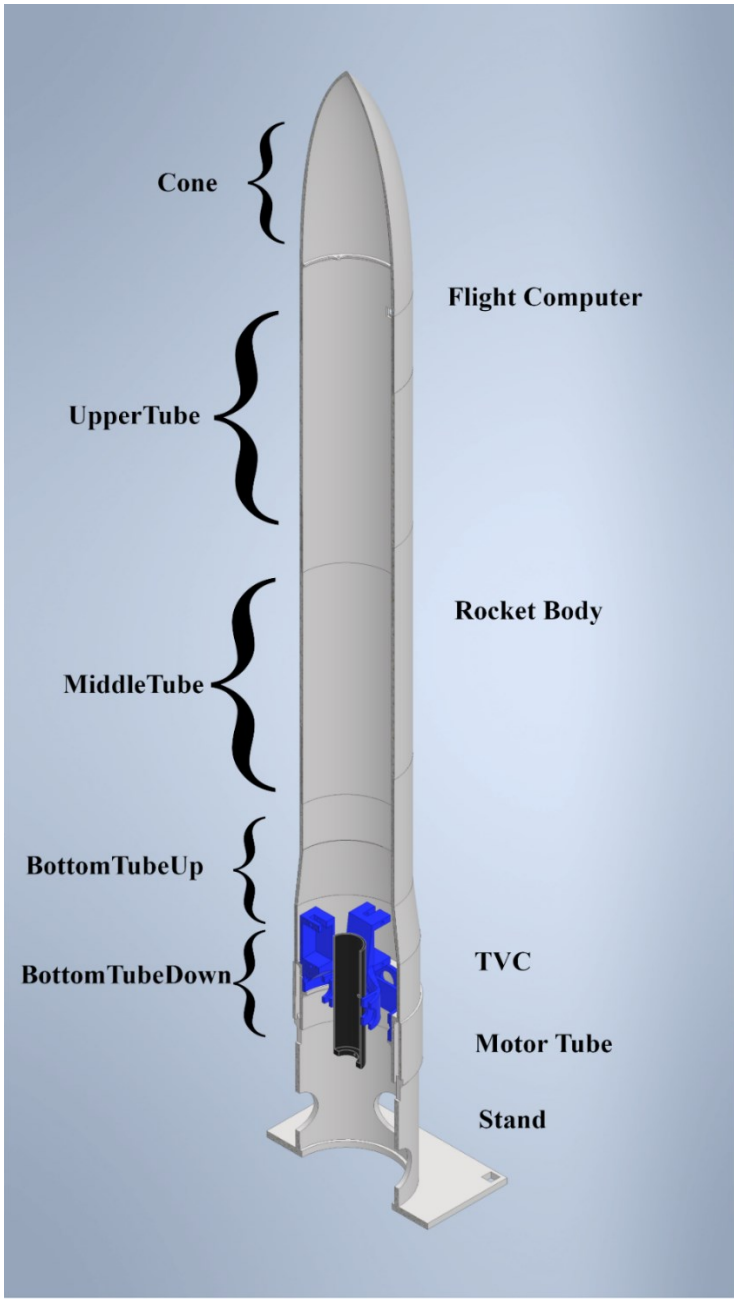


*Figure 26: Main Parts of the Rocker*

All these parts play a crucial role in the rocket's stability. Since I have no fins, which are the main stabilizing components and move the center of pressure below the center of gravity (center of mass), the goal was to put as much weight on the nose as possible. This is why the flight computer is located close to the cone.

It may appear that I am missing a crucial part, which is a recovery system with parachute deployment. However, given the rocket's maximum height of 1.34m with this motor and the weight of the rocket being around 410g, there is no need for parachutes. Eliminating the parachute system not only simplifies production but also reduces the weight of the rocket.

## Ideal Rocket Weight

To calculate the optimal weight of the rocket, I conducted a flight altitude simulation using Matlab. During the simulation, I disregarded physical parameters such as the rocket's drag coefficient and air density, as their impact on the rocket's flight is minimal. It is important to note that this scenario assumes a straight flight, meaning that any deviation from a straight path will result in a decrease in total thrust and an earlier touchdown.

Additionally, in the Third Launch, I observed that the rocket reached its highest point within 1 second, after which it lost control and tilted to the side before descending. This outcome aligns with our predictions. The total weight of the rocket in this test was 473g.

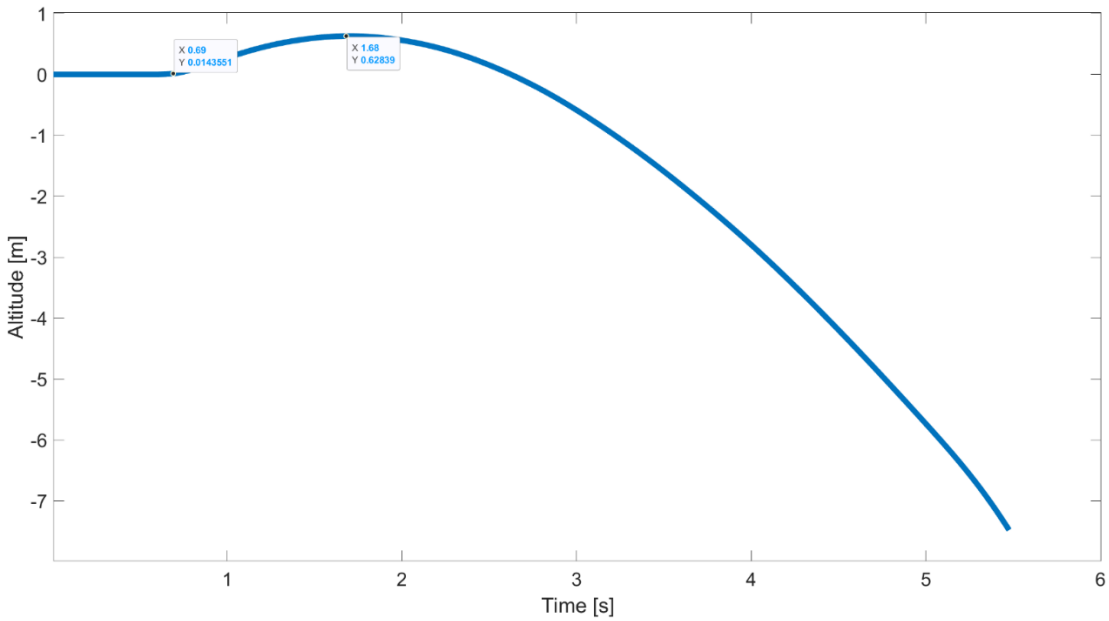


*Figure 27: Simulation of the Rocket Weight 473g*

Another aspect of this simulation was determining the optimal weight of the rocket, which is around 410g. With this weight, the rocket would descend close to the ground, aligning with our desired behaviour. This not only allows for potential landing but also ensures that the entire burn time is utilized for evaluating our control systems while being near the ground. This would expedite the testing phase as the rocket would not be destroyed. The graph below illustrates the flight trajectory for a rocket with a mass of 410g.

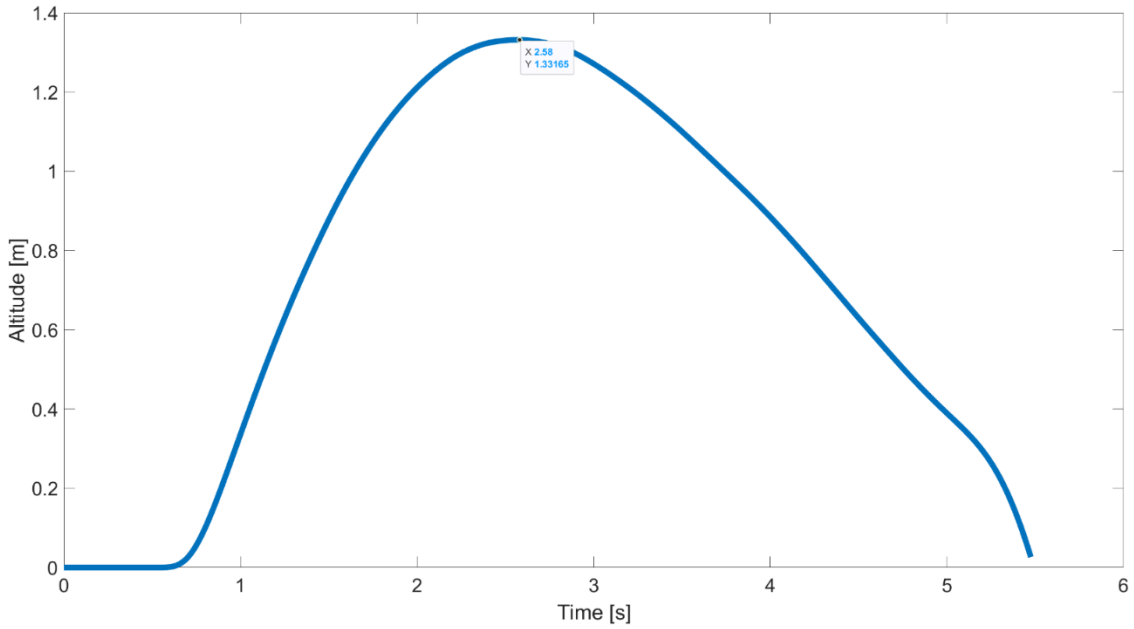


*Figure 28: Simulation of the Ideal Rocket Weight 410g*

Taking into consideration the impact of the angle on total thrust, which result in an average angular displacement of approximately 10° (accounting for both the motor and the rocket's orientation), there is a discernible reduction of 1.5% in the overall thrust as illustrated in Figure 9. Nevertheless, even in this scenario, the optimal weight of the rocket changes only to 404g. It is worth noting that the observed deviation in performance is not of considerable magnitude. Considering the presence of additional factors such as air drag and other unaccounted influences, it is reasonable to assert that a rocket weight falling within the range of 400-410g represents a desirable and efficacious range for achieving optimal performance.

## Modeling the Rocket

During the modeling, the most challenging task was to find the right thickness of the wall and gap between the parts. To be able to print the rocket, it was divided into five distinct parts, each part being no longer than 200mm tall, so it can easily fit into the printer. The parts are connected using only friction. The parts are slipped into each other, using no screws, resulting in an easy assembling process while allowing a little gap, making the rocket fit perfectly together. On the other hand, it needs a lot of tuning to find the perfect gap.

In the definitive version, I used the gap of 0.1mm, and the wall width of 2.1mm and 1mm for the connecting parts, which were the main issue if I made the rocket thinner, as they could easily break. For better understanding of the distances, in the following picture I can see the sketch for the upper tube. On the left, I can see the edge for PCB Holder, the thickest part in the middle is normal wall, and the thinner ones are the parts that are connecting the rocket. The 0.1mm which is missing after adding up is the gap which is used to fit the two parts of the rocket together. Without the gap, the parts are too close together and cannot slip in.

The development of the rocket body proved to be easier compared to the TVC. Only two parts underwent significant changes. Firstly, the stand was modified as the initial version posed complexities for 3D printing.

Secondly, the thickness of the BottomTube was reduced, as the previous diameter exceeded the necessary requirements, resulting in an unnecessarily heavy and aesthetically unconventional rocket design.

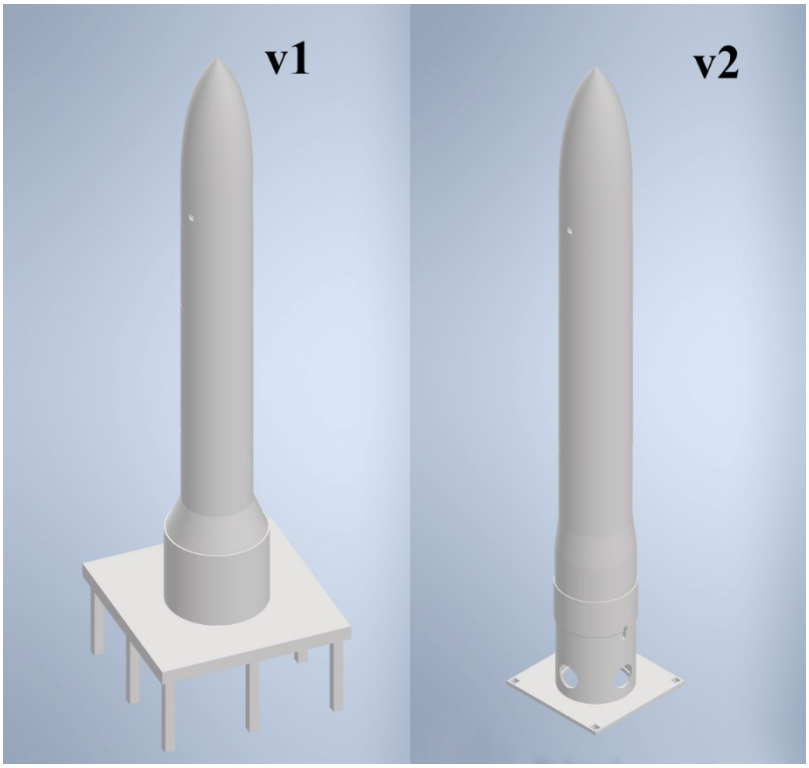


*Figure 29: Development of a Rocket Body and Launchpad*

During the design process, I minimized the use of nuts, as they posed challenges in proper placement. Instead, I utilized printing material with a smaller diameter to serve as screw holes, offering an elegant and simplified method of assembling the rocket. The model includes essential holes for various purposes, including: 2 screws for each TVC axis (total of 4 screws), 2 holes for attaching the movable pushrod from the servo (1 for each axis), a hole to attach the TVC inside the rocket (along with the holder), and additional holes for facilitating easier manipulation during the screwing process.

## Improvements

Although the rocket is ready for launches, there are still areas that could be enhanced in future iterations to improve the assembly process.

- Increase the size of the PCB screw holes.
- Avoid using a ring around the PCB, as it obstructs easy insertion of the Teensy.
- Utilize friction-based screw holes on the PCB instead of relying on nuts for fastening.
- Print a new and improved stand, as the current stand has an incorrect diameter.
- Expand the bottom diameter of the rocket to increase the radius for the servo.
- Resolve the problem of limited space for the SD card, leading to breakage, by incorporating an additional hole in the cone specifically designed for inserting the SD card.
- Integrate a battery holder on the inner side of the rocket, away from the servos, ideally positioned in the center.
- Ensure that the RBFP hole is positioned underneath the part used for connecting two tubes.

- Extend the length of the bottom tube downward to prevent the motor from touching the ground during servo movement.

In designing the body, our objective is to create a solid structure that can safeguard the flight computer and gimbal motor, while maximizing height without exceeding the desired total weight of approximately 410g. This necessitates careful consideration of the appropriate material.

I am using two different printers for printing: the Ultimaker Cura 3 Extended and the Prusa i3 MS3S+. As for the filament, I am using the one available at the PM-Filament website [29], and our chosen slicer is Ultimaker Cura. However, it is important to note that the slicer did not perform well on the Prusa printer. Therefore, it is strongly recommended to use a slicer specifically designed for that printer.

## Material

I am looking for the best takeaway between weight and strength. For printing, I was deciding between PLA, PETG, ASA and ABS. In the end, ASA was decided as the best solution. It is lighter than PLA and PETG, also stronger, it can withstand UV light, is easier to cut compared to PAL and PETG if I need to make small adjustments or the screws are not fitting perfectly, also I can use acetone to smooth the surface or connect two prints together. To sum up, ASA offers the best possible specs while being ideal for outside as it is UV stable, and less risky to lose the shape while cooling down compared to ABS, also it smells less during the printing process. In belowTable 2, a comparison of different printing materials can be seen:

Fracture toughness is a material property that assesses its resistance to crack propagation and fractures. It quantifies the energy required to initiate and propagate cracks within a material, playing a crucial role in engineering design and structural integrity analysis by determining its resistance to brittle fracture. In our specific case, lower fracture toughness indicates greater ease in cutting holes in the material.

Stiffness, on the other hand, measures a material's ability to withstand bending or flexing without deformation. PLA demonstrates excellent stiffness, making it highly resistant to deformation. However, it is important to note that stiffness and strength are distinct characteristics. While I do not have specific strength data for the materials, ASA is up to three times stronger than PLA against impact, making it an ideal choice for our requirements. [30]

| Material (1.75mm) | Temperature Hotend | Temperature Heating Bed | melting Temperature | Fracture Toughness | Stiffness | Density | Source |
|---|---|---|---|---|---|---|---|
| PLA | 200-220°C | 20-60°C | 55°C | 64.5kJ/m$^2$ | 3500MPa | 1.24g/cm$^3$ | [31] |
| PETG | 220-250°C | 80-90°C | 85°C | 11kJ/m$^2$ | 1800MPa | 1.27g/cm$^3$ | [32] |
| ABS | 220-250°C | 100-100°C | 94°C | 20kJ/m$^2$ | 1800MPa | 1.04g/cm$^3$ | [33] |
| ASA | 240-260°C | 100-110°C | 96°C | 12kJ/m$^2$ | 1800MPa | 1.07g/cm$^3$ | [34] |

*Table 2: Material Comparison*

When it comes to the TVC Mount, it is advisable to use PLA instead of ASA. The reason is quite simple: although ASA can withstand impacts well, it is more prone to bending. I conducted a test by printing both the rocket body and the TVC using ASA and PLA, respectively. Based on our findings, ASA is a better choice for the rocket body since it is the part that touches the ground first and needs to be flexible

enough to withstand crashes. However, when it comes to the servo linkages and the TVC itself, I want to avoid any flexing. In this case, PLA is a great option as it is less flexible, allowing for more precise movements of the TVC Mount.

To summarize this section, it is recommended to print the rocket body using ASA to save weight while ensuring a material that is better for impacts and more durable under sunlight. On the other hand, the TVC should be printed using PLA to minimize flexing, which could negatively affect the precision of the TVC Mount.

## Printer setting

The next step in the printing process involved setting up the printer with the appropriate parameters. For printing with PLA, I used a temperature range of 190-210°C, with the heating bed set to 50°C for the initial layer. With ASA, higher printing temperatures were required, ranging from 250-260°C, and the heated bed was set to 100°C for the initial layer and 50°C for the rest of the print.

To ensure maximum strength, I used a 100% infill for both the rocket body and the TVC. However, for the holder and test stand, a 60% infill was sufficient. The choice of layer width depends on the printer, but for the TVC, it is recommended to use a smaller width to achieve accurate details and precise fit. I used a layer width of 0.2mm for the TVC and 0.3mm for the rocket itself. It is worth noting that smaller layer widths result in better detail but longer printing times. The rocket body was also printed with a diameter of 0.2mm, resulting in a slightly rougher surface when using PLA, but it did not affect its functionality.

During the printing process, the speed was limited to a maximum of 60 m/s to minimize noise. The best support type for Ultimaker Cura Extended 3 was found to be zigzag, as it was easier to remove and provided optimal support.

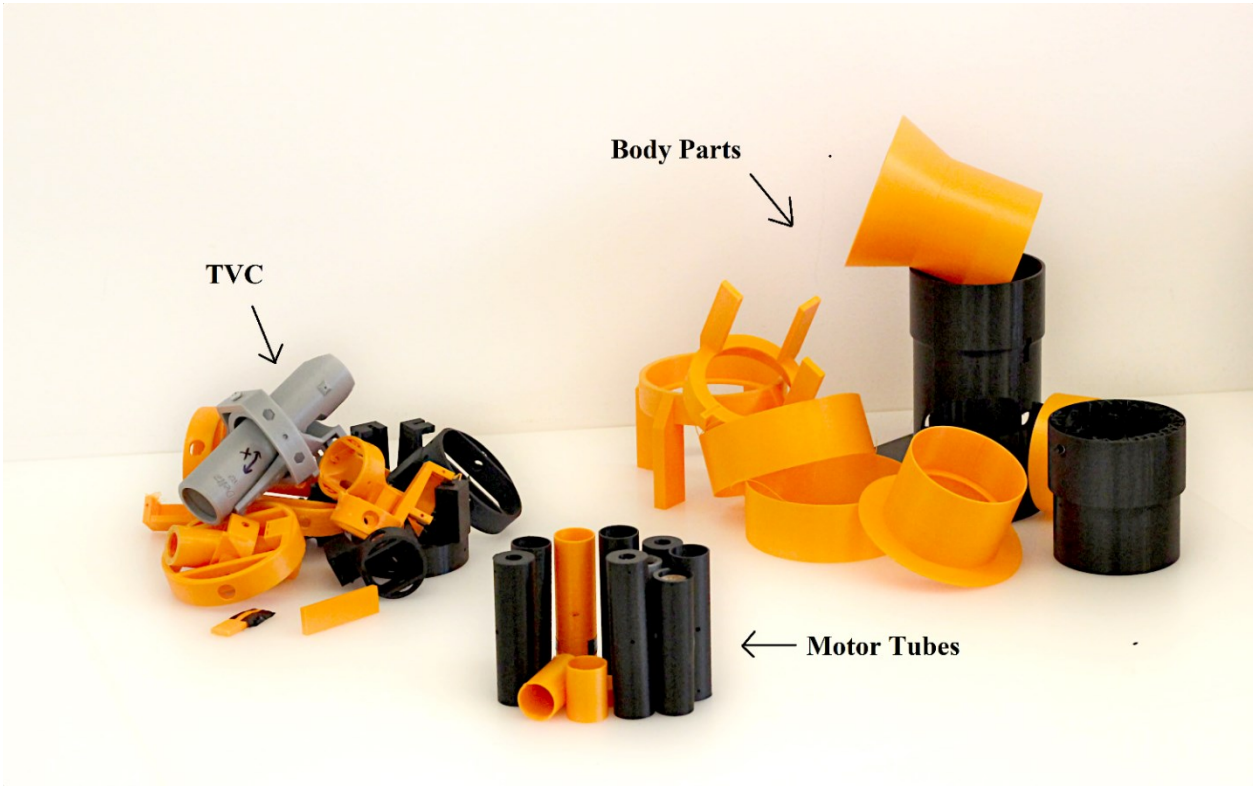


*Figure 30: 3D-Printed Parts and Optimization Process*

Some important notes to consider: In the Ultimaker Cura Slicer, it is recommended to turn off support towers, as they are unnecessary and difficult to remove. Additionally, when slicing the model, ensure that it is rotated by 45° horizontally to avoid the nozzle moving across the axis where the screws are placed. Consistent rotation angles for each part of the rocket create a helpful reference line during assembly, ensuring the proper alignment of the TVC with the flight computer's BNO sensor. When it comes to vertical orientation, experimentation is necessary to find the ideal settings that minimize the need for support structures inside the rocket. Printing supports on the outside can negatively impact the appearance and assembly process. Lastly, rotating the BottomTubeDown part to achieve better print quality for the inner section holding the TVC is recommended (avoiding support for this part).

In conclusion, achieving the best possible print quality requires time and experimentation to find the optimal settings for your specific printer and model. I conducted multiple tests during the printing process, particularly focusing on infill and layer thickness for the rocket body.

## Total Print Weight

Now that I am aware of using ASA material for printing, it is crucial to ensure that the total weight of the rocket does not exceed 410g. Let us consider the following weights:

- Battery: 28g
- Flight Computer with cables: 60g
- TVC Mount: 64g
- Rocket Motor: 28g

Therefore, I have approximately 230g remaining for the rocket body. To determine the weight using the current model, I can utilize a slicer and the correct density of the ASA material I am using, which is 1.07g/cm$^3$. The calculations yield the following results:

- Cone: 56g
- Upper Tube: 70g
- Middle Tube: 70g
- Bottom Tube (Upper Section): 55g
- Bottom Tube (Lower Section): 32g

This gives us a total rocket weight of 463g. Considering the addition of some color on top and stickers, the weight increases to around 470g, which exceeds the desired limit. To resolve this issue, I have two options. The first option involves reducing the wall thickness to decrease weight. Additionally, I can decrease the weight by 6g by removing unsoldered connectors on top of the PCB. Alternatively, the simpler solution is to fly without the middle tube, as it serves no specific purpose and can be easily eliminated.

## Building the Rocket

In preparation for the launch, the construction of the rocket is a necessary step. To accomplish this, the following components are required:

- Printed 3D Parts
- Two SG92R Servos or equivalent
- D3-P Klima Rocket Motor
- Two metal parts for the pushrods
- Flight Computer
- Servo Extension
- Four bearings
- Fourteen M2 screws
- Two M2 nuts
- Four female and two male-male jumpers

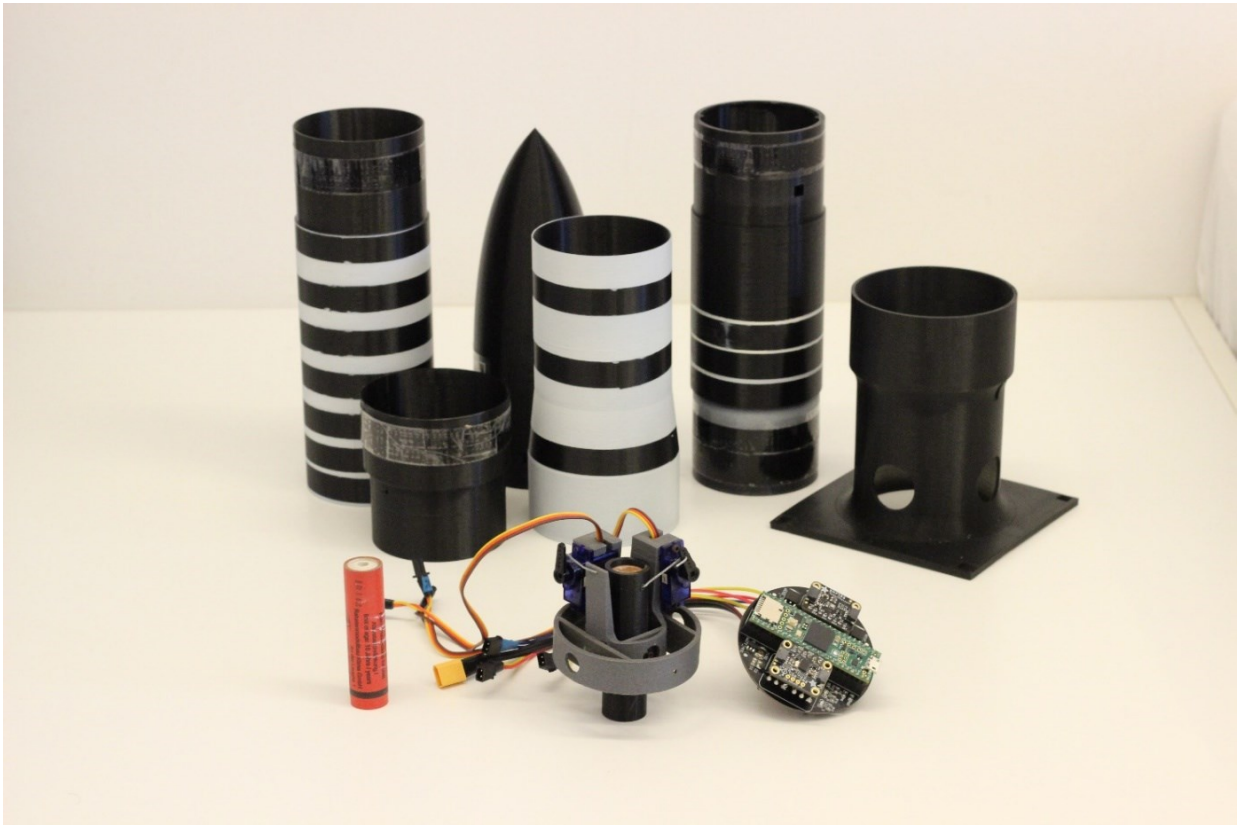

*Figure 31: All Rocket Body Parts, TVC, D3-P Klima Rocket Motor and Flight Computer.*

### TVC Mount

The assembly process for the TVC can be carried out by following these steps:

1. Place the bearings into the designated holes in the inner gimbal.
2. Insert the motor into position and use screws to securely tighten it.
3. Place the bearings into the outer gimbal.
4. Insert the inner gimbal from the previous step into the outer gimbal.
5. Utilize screws to firmly fasten the outer gimbal.
6. Insert the servos into their designated positions and securely fasten them using screws.
7. Remove the servo arms temporarily.
8. Insert the pushrods into the arm then into the motor tube and inner gimbal as shown in the provided picture.
9. Put the arm back on the servo.
10. Place the motor inside the motor tube and use slices of paper to achieve a precise fit around the motor tube.

Following these steps, your TVC mount should resemble the one depicted in the accompanying picture.

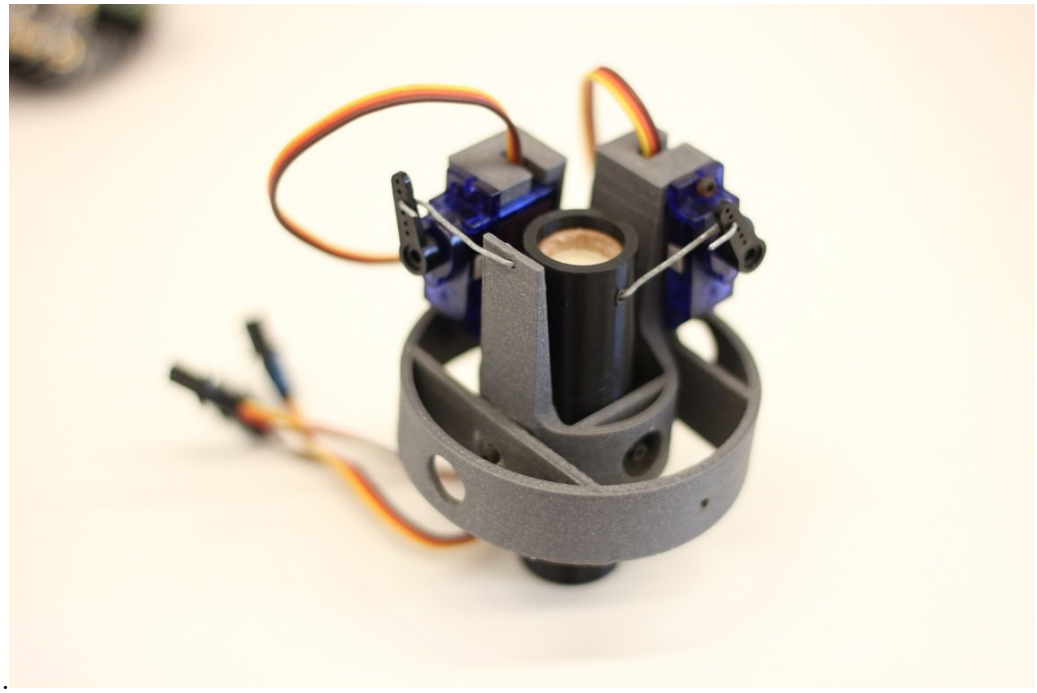

*Figure 32: Assembled TVC Mount in Detail.*

### Flight Computer

Next, I proceed with the preparation of the FC. In this step, I will insert the following cables into the bottom part of the PCB:

1. Connect the servo extenders for both the X and Y axes. Make sure to label them accordingly to indicate which one is for each axis, as well as label the servo cables themselves.
2. Insert the battery connector. The specific type of connector may vary depending on your battery; in my case, it is an XT-30 connector.
3. Insert two male jumpers for the Breakout.
4. Insert two male jumpers for the RBF (Remote Breakout Facility).

As you can see in the picture below, the servo's ground (brown cable) is positioned on the right side, the red cable is in the middle, and the yellow cable is on the left side. The ground connection for the battery is indicated by a white tape in the picture.

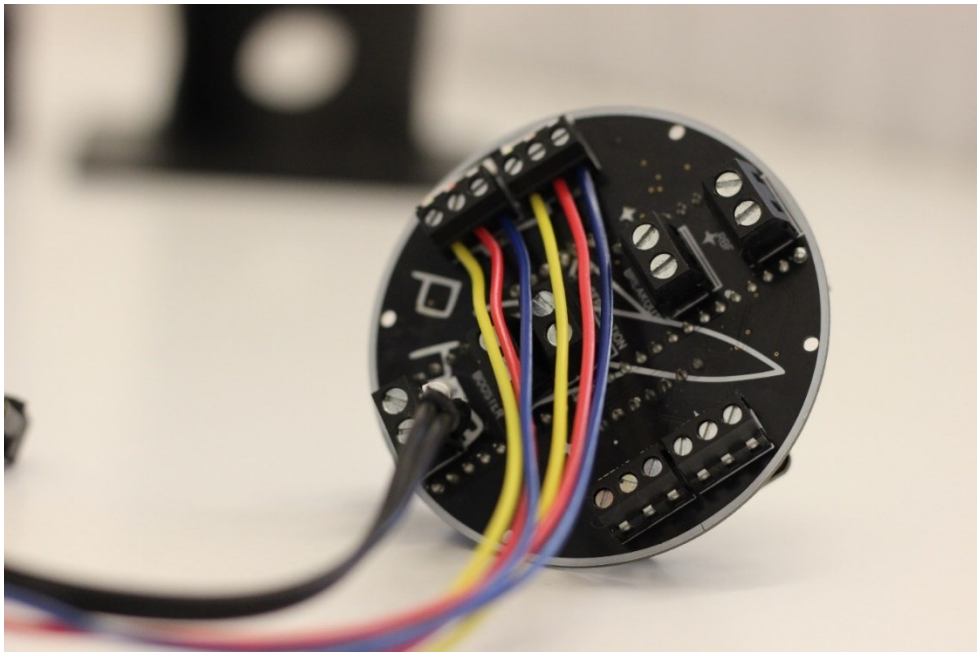

*Figure 33: Servo and Battery Cables*

Now, I will place the prepared flight computer on top of the upper tube, ensuring that the cables are inserted inside the tube. It is recommended to label the side with the SD Card slot for future reference. The assembly should resemble the configuration shown in the picture below.

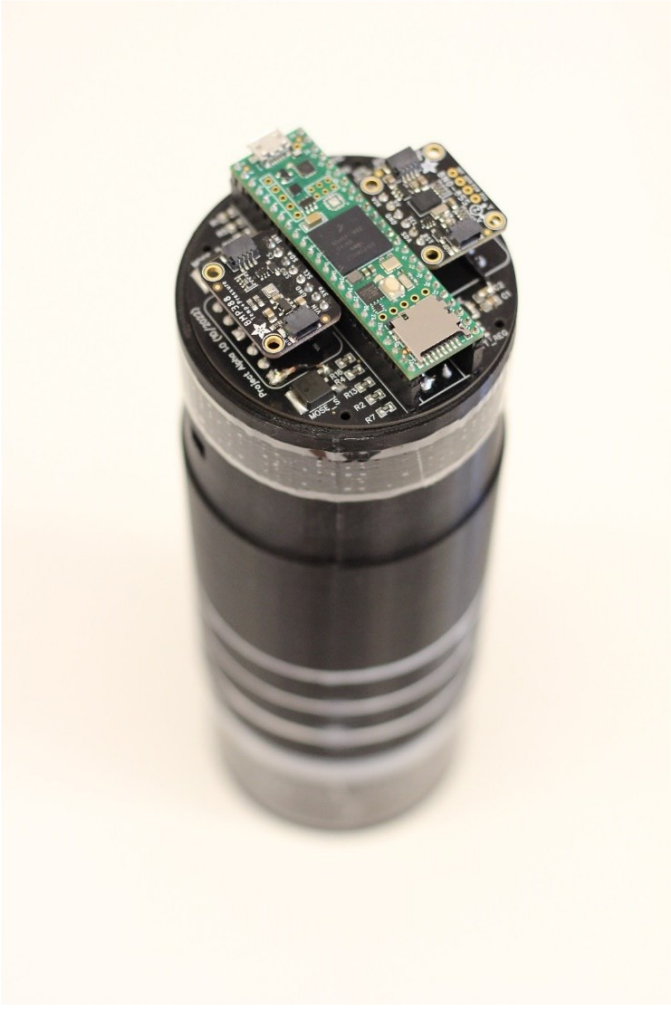

*Figure 34: Flight Computer on Top of the UpperTube*

In the last step, I will connect the servo extenders to the servos, insert the battery, and connect all the components together. I will leave the breakout and RBF cables hanging through the holes later use. Additionally, to improve aerodynamics, it is advisable to tape the end of the two female headers for both wires onto the outside of the rocket body. With the completion of the assembly process, the rocket is now fully prepared for calibration and final adjustments before the flight.


*Figure 35: Fully Assembled Rocket*

## Calibration

Before liftoff, I must do the following tasks:

- Calibrate Servos
- Find the moment of inertia.
- Compute the servo delay.
- Mark how to align rocket parts during the assembly.

During the calibration process, it is necessary to take note of the following values, which will be required for PID tuning in the next chapter:

- Minimal thrust of the rocket motor.
- Rocket weight.
- Moment of inertia.

- Both servo delays.

### Servo Calibration

This is a crucial step in determining the central position of our TVC system. The optimal method involves attaching an additional IMU sensor directly to the motor and calibrating it using a closed loop method. However, since this can be a challenging task, I will focus on manual calibration. For this purpose, it is highly recommended to use a sheet of paper placed at the end of the motor, which will extend its length and make it easier to identify any misalignments, as represented in theFigure 36. In the next step, open a script and begin adjusting the servo values for each axis to find the perfect central position.

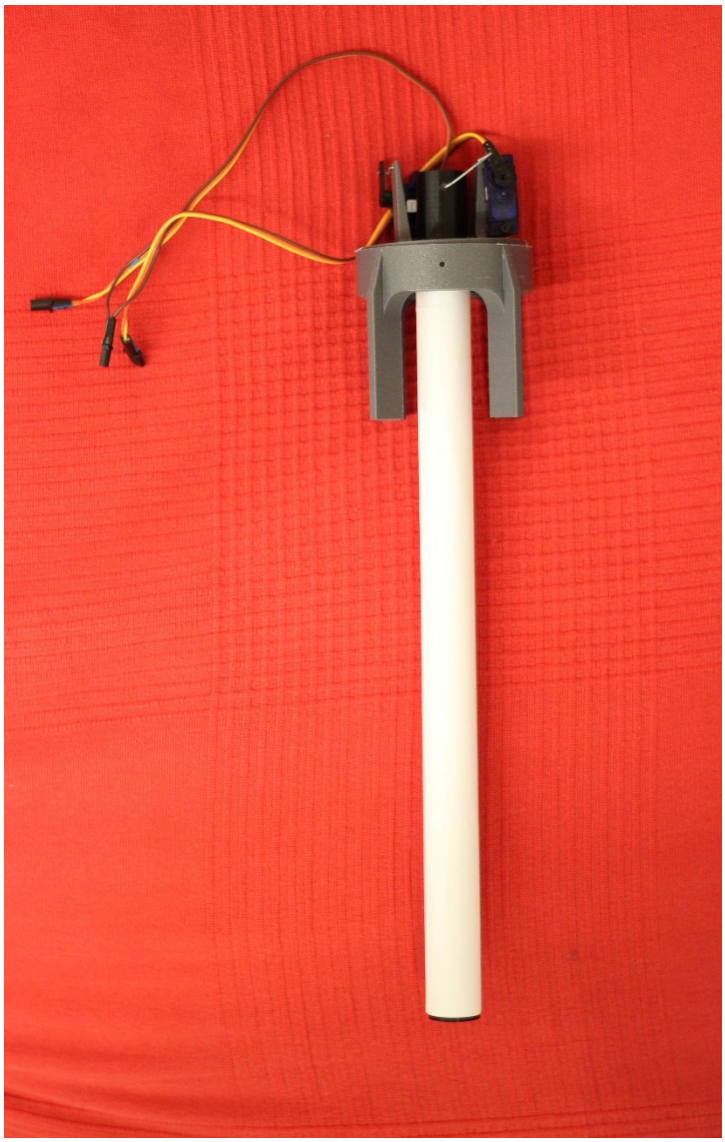

*Figure 36: The Ideal Central Position for the Servos Using a Paper Tube to Extend the Rocket Motor.*

Following the fifth flight, during which calibration emerged as one of the main causes of failure, I have developed a new TVC calibration tool. This tool can be easily attached to the bottom part of the rocket and effectively maintains the motor in a perfectly central position.

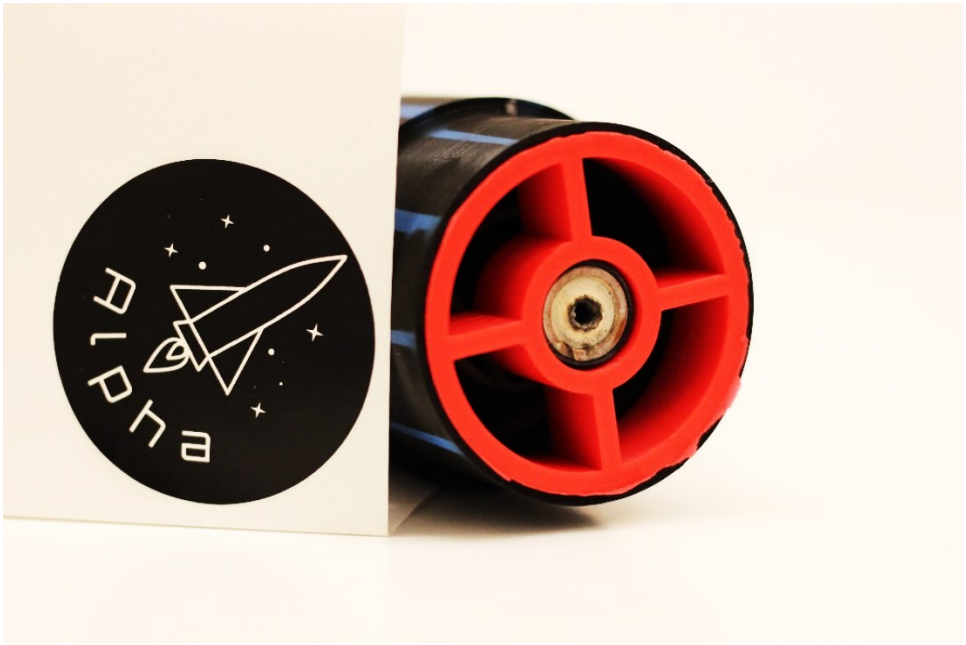


*Figure 37: New Tool to Calibrate Motor Upright Position*

### Moment of Inertia

To ensure I have all needed values for PID tuning, I need to follow these steps with fully assembled rocket, ready for flight: [35]

1. Weigh the Rocket
   Measure the weight of the rocket, including all its components, to obtain an accurate value. Extra or uneven weight, such as a missing battery, can affect the rocket's performance during the flight.

2. Find the Center of Mass (COM)
   Locate the center of mass by placing your finger at different points along the rocket until you find a position where it is stable. Once identified, mark this spot using a piece of duct tape or any suitable marker.

3. Hang the Rocket in a Vertical Position
   Use two parallel strings and hang the rocket vertically. Ensure that both strings are equidistant from the COM and provide stable support for the rocket.

4. Measure the Distance from COM to Strings.
   Measure and record the distance from the COM to each of the strings. This information helps determine the moments acting on the rocket during flight.

5. Measure the Distance from Strings to Support Surface
   Measure the distance between the strings and the surface (e.g., table) from which the rocket is hanging. This measurement is important for calculating the moments around the center of mass.

6. Find the Duration of One Oscillation
   Rotate the rocket around its COM while keeping the COM stationary. Observe the time it takes for one complete oscillation. For better accuracy, record the time for multiple oscillations and then calculate the average time for one oscillation.

7. Calculate the MOI
   Using the gathered measurements and data, perform the necessary calculations:

| | |
|---|---|
| m | 0.405 kg |
| oscillation | 1.246 s |
| COM to strings | 0.155 m |
| string length | 0.54 m |
| g | 9.81 m/s$^2$ |

$$\text{MOI} = \frac{m \cdot g \cdot oscilation^2 \cdot d^2_{\text{COM\_to\_strings}}}{d_{\text{string\_length}} \cdot 4 \cdot \pi^2}$$

Remember to document all the measurements and data obtained during these steps. This information will be valuable for subsequent analysis and calculations during the PID tuning and control process.

### Servo delay

To find the servo delay, specifically the reaction time and servo speed, you can use a slow-motion camera on your phone and follow these steps:

Step 1: Find the Reaction Time

- Set up a script which turns on the LED before moving the servo.
- Start a slow-motion recording and ensure both the LED and the servo are captured.
- Review the recorded video and note the number of frames between the LED turning on and the servo starting to move.
- Determine the frame rate per second (fps) of the slow-motion recording.
- Calculate the reaction time by dividing the number of frames by the frame rate (reaction time = number of frames / fps).

Step 2: Find the Servo Speed

- Record a slow-motion video of the servo moving a specified distance, for example, X degrees.
- Count the number of frames it takes for the servo to move the specified distance.
- Calculate the time it takes for the servo to move by one degree by dividing the number of frames by the frame rate and then dividing the result by the specified number of degrees (time per degree = number of frames / fps / X).

Both the reaction time and the time it takes for the servo to move by one degree are valuable measurements for the PID tuning process.

### Aligning Rocket Axes

To ensure that the axis of the servo matches the axis of the sensor, follow these steps:

Step 1: Build the Rocket with Proper Alignment

- Construct the rocket in a way that aligns the axes of the servo and the sensor.
- Ensure that the Y axis of the sensor corresponds to the Y axis of the IMU sensor.
- Mark the positions of each rocket part on the rocket body for reference.

Step 2: Verify Axis Alignment during Software Testing

- Run the flight software that controls the rocket and its servos.
- As you move the rocket, check if the servos are moving in the expected direction.
- Confirm that the Y axis of the rocket aligns with the movement of the servos and the desired direction of motion.
- If any discrepancies are noticed, adjust the alignment of the rocket parts and servo connections, as necessary.
- Repeat the testing process until you are confident that the axes are matching, and the servos are moving correctly in relation to the rocket's movement.

Verifying the axis alignment during software testing will help you detect and correct any misalignments or issues with the servo movements. This process will contribute to a smoother launch day and more accurate control of the rocket.

# Chapter 5: Flight Computer

The Flight Computer serves as the central brain of the system, constantly monitoring sensor inputs from IMU and Barometer to determine the rocket's orientation, motion, and altitude. Based on this information, the Flight Computer calculates and generates real-time control commands to adjust the servos, ensuring that it stays on its intended trajectory.

## FC Capabilities

The Flight Computer was designed to be as universal as possible, making it suitable for later and more complex projects. It was specifically designed for a two-stage TVC model rocket and incorporates the following functionalities:

- Collecting data from sensors, including a barometer and IMU.
- Controlling servos for both rocket stages.
- Initiating the separation between the first and second stages.
- Activating the second motor.
- Indicating the current state through LEDs.
- Signaling the activation of the Booster and Separations.
- Implementing the use of RBF (Remote Bus Fire) for safety.
- Utilizing a Breakout wire for precise liftoff detection.
- Storing data on an SD card.
- Powered by a LiPo battery.

This design allows the Flight Computer to serve a wide range of purposes, providing flexibility and reliability for various rocket projects, including those with increased complexity.

## Development

Due to the rocket's limited diameter, the Flight Computer board was designed to fit on top of a 3D printed piece and covered by a cone. This alternative placement was chosen to simplify rocket design, facilitate printing, and improve accessibility compared to the conventional vertical placement inside the rocket.

*Figure 38: UpperTube with PCB Holder*

The final design of the Flight Computer incorporates essential components and features. These include:

- Two MOSFETs for stage separation and activating the second stage motors.
- A reliable AMS1117-5V voltage regulator with a stabilization voltage of 5V is chosen. It has a voltage drop of 1.3V and can handle currents up to 1A. This choice is fully adequate for our purposes of providing a stable 5V output from the 2S LiPo battery.
- A resettable fuse with a rating of 15V/500mA to protect the MCU from short circuits when powered by the battery.
- Multiple pull-up resistors to ensure successful I2C transfer for the sensors.
- Convenient connectors for easy connection and disconnection of servos and wires, eliminating the need for soldering.
- Female headers used to connect both sensors and the MCU, facilitating easy replacement in case of any issues.

This design ensures efficient functionality and ease of maintenance, allowing for seamless integration of components and streamlined operation of the Flight Computer.

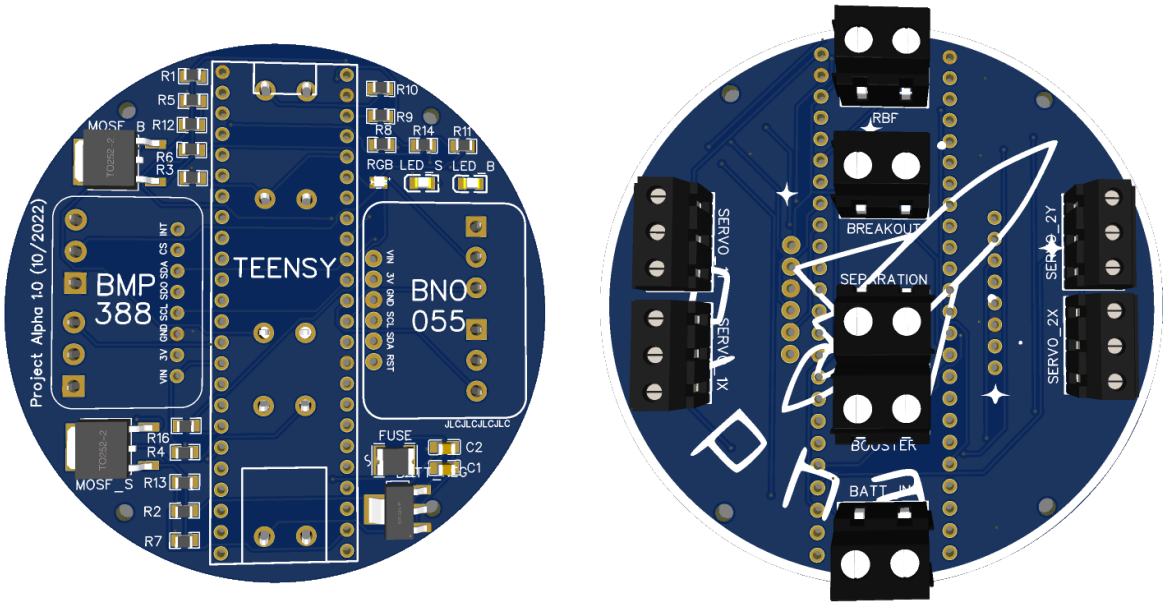


*Figure 39: Visualisation of PCB Using EasyEDA.*

It is important to consider potential vibrations during flight as they can have a negative impact on certain components. Using a relay or vertical switch may not be the best choice, as vibrations can negatively affect their performance and the vertical switch can turn off the rocket during the flight.

When it comes to developing the FC, starting with a breadboard or protoboard is more forgiving, especially for initial attempts, allowing for easier modifications and testing. Once the design is finalized, transitioning to a PCB offers advantages such as reduced weight, improved performance, and a cleaner look.

Initially, the flight computer was wired to a breadboard, but this was not ideal for flight due to the risk of contacts being lost due to vibrations. To address these concerns, the circuit was soldered onto a protoboard. Changes were made to the design, including the use of an RGB LED instead of separate LEDs, the addition of a regulator for a stable 5V output from the battery, and proper connection of the servos to the correct pins on the flight computer.

Unfortunately, during the last step, an error occurred when measuring the regulator, resulting in the accidental connection of two pins with the multimeter. This caused both the regulator and Teensy to burn out. Considering the time required to replace the MCU, a decision was made to design a PCB instead. This PCB has been in use since then and has proven to be a reliable solution for the flight computer.

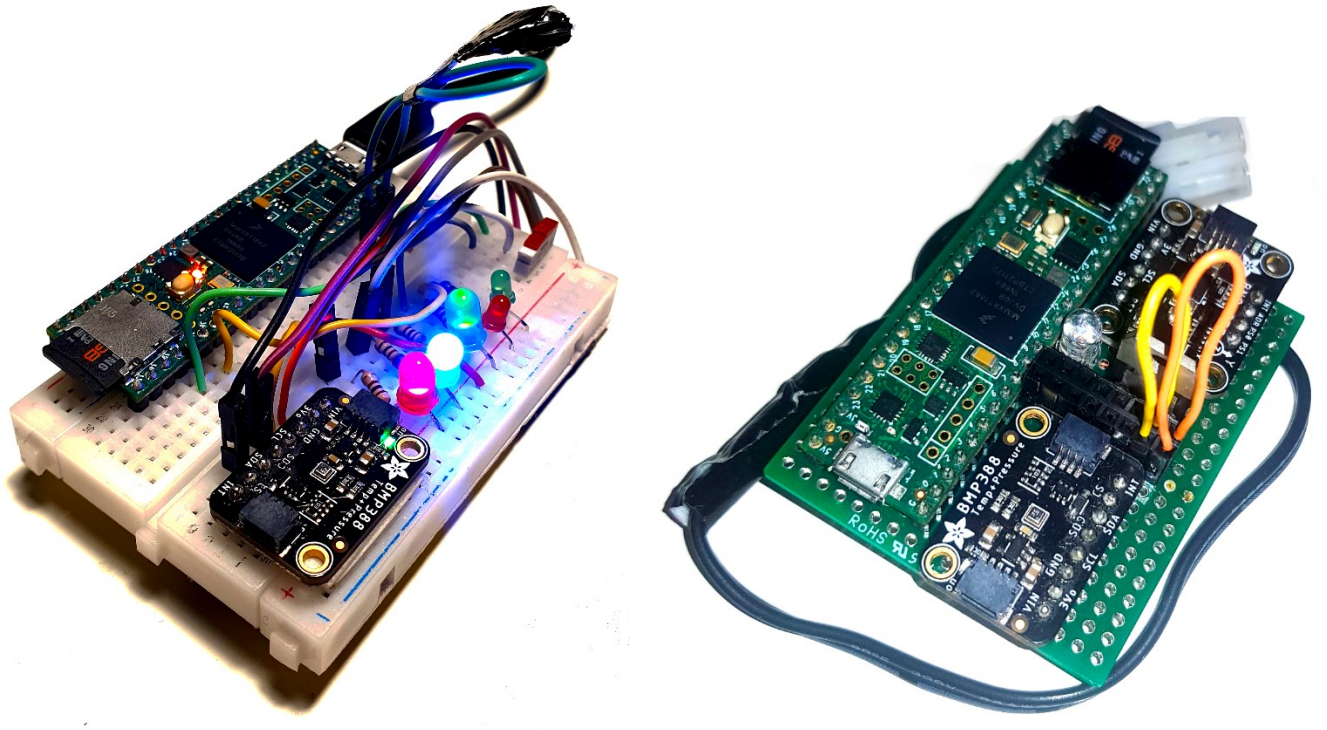


*Figure 40: The First Two Iterations of Flight Computer*

The PCB was designed using EasyEDA. However, there were some issues in the schematic that affected the power supply to the MCU from the battery and the state diode. Specifically:

- The battery ground (GND_BATT) was not connected to ground (GND).
- There was a mistake in the labeling of V_BATT (8V), resulting in a failure to connect the battery with the MOSFETs for separation and booster ignition.
- The State RGB LED configuration is not ideal and would benefit from using a transistor.

The red LED typically has a voltage drop of around 1.6V. With a difference of 1.7V between the 5V and 3.3V, the voltage across the series combination of resistor and LED is at its maximum. According to Kirchhoff's laws, the voltage across the resistor would be 0.1V. With a 150-ohm resistor, the current would be 600uA, which is not enough to cause damage. However, the red LED may not turn off completely.

Using transistors to drive the LEDs in the Flight Computer is a better approach. In the previous example, if the pin is LOW, the current flowing through the illuminated red LED would be 22mA. The Teensy allows a maximum load of 10mA per pin, which is already not ideal. This configuration puts excessive strain on the Teensy and could potentially lead to overheating or damage to the pin, although it should not cause a complete failure.

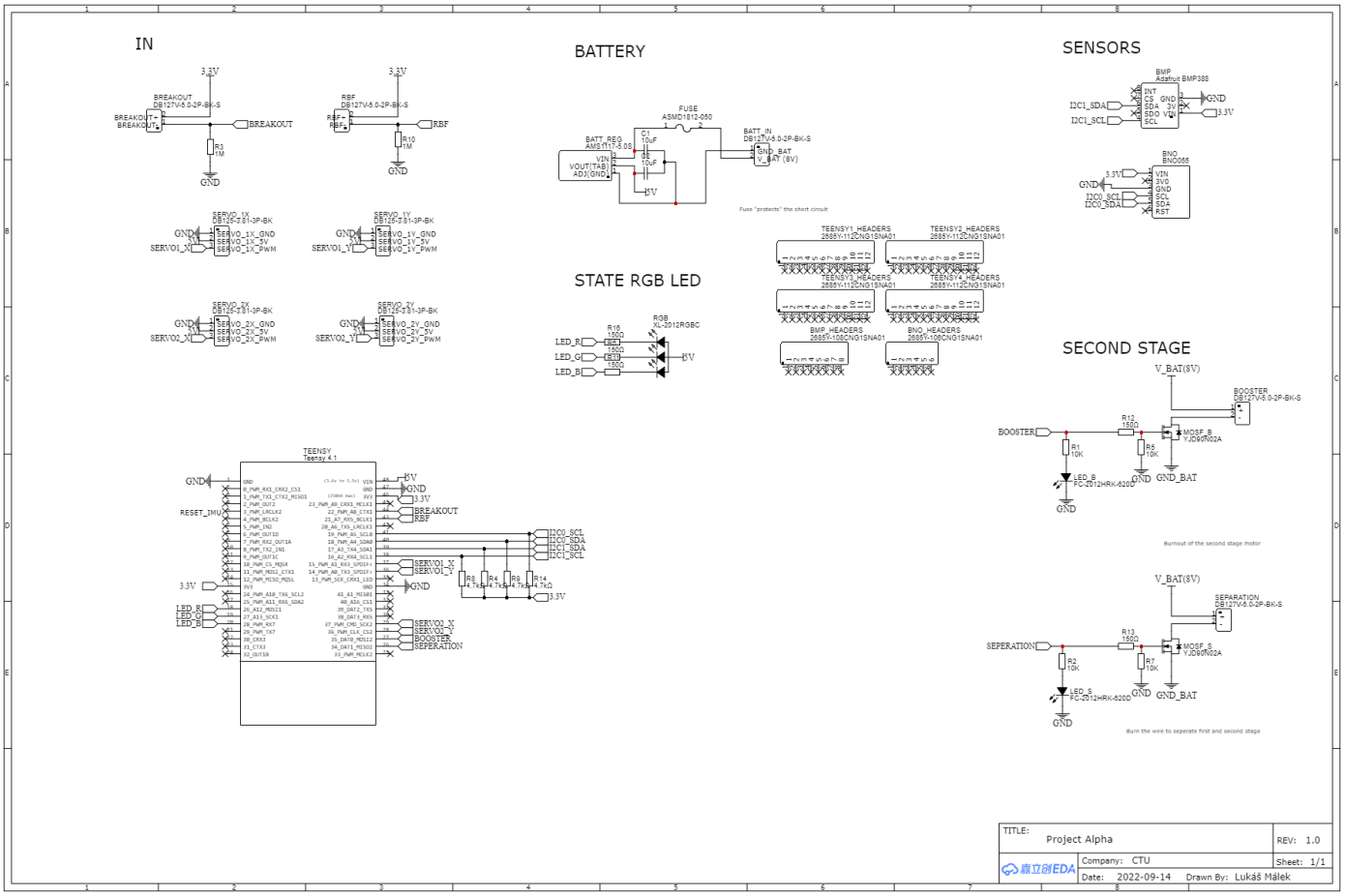


*Figure 41: Flight Computer Schematics*

The various components are connected to the pins of the central microcontroller, Teensy, using the interfaces that facilitate their communication with each other.

| **Module** | **Interface** | **Pin** |
|---|---|---|
| Adafruit BNO055 | I2C0 | 18,19 |
| Adafruit BMP388 | I2C1 | 16,17 |
| RBFP | ADC | 21 |
| Breakout | ADC | 22 |
| RGB LED Red | IO | 26 |
| RBG LED Green | IO | 27 |
| RGB LED Blue | IO | 288 |
| Separation | IO | 34 |
| Booster | IO | 35 |
| Servo1 X | PWM | 15 |
| Servo1 Y | PWM | 14 |
| Servo2 X | PWM | 37 |
| Servo2 Y | PWM | 36 |

*Table 3: Teensy 4.1 Pinout*

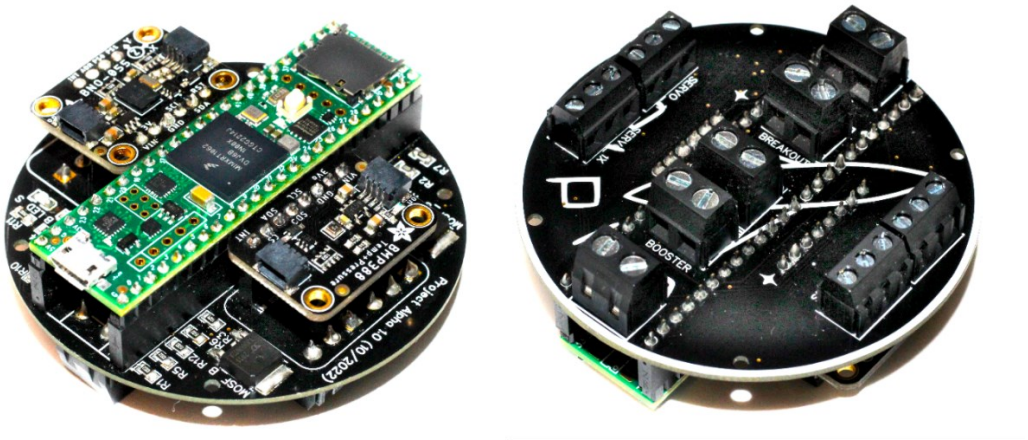


*Figure 42: Fully Assembled Flight Computer*

## Microcontroller Unit

The MCU is the heart of the flight computer, responsible for various tasks such as sensor data reading, servo control, state machine operation, PID computation, and data logging. I aimed to achieve a frequency of 100Hz during the flight, which is limited by the sensors. Hence, I needed a robust yet lightweight flight computer that could directly log data to a microSD card without the need for additional peripherals.

I considered options like Raspberry Pi, Beagle Bone, Arduino Uno-Zero, and Nucleo STM boards. However, they were too heavy, large, and more powerful than necessary for our needs. Raspberry Pi Pico, Arduino Micro, and Arduino Nano were also evaluated, but they lacked sufficient power and lacked the capability to directly log data to an SD card. Additionally, Teensy 3.2 and ESP8266 lacked a microSD card slot. Also, Teensy supports RTC, which allow us to measure the time even when the MCU is not powered using an external battery.

Ultimately, selecting Teensy 4.1 seemed like a wise choice as it provided enough computational power and fulfilled all our requirements.

| **Parameter** | **Value** |
|---|---|
| CPU | Arm Cortex M7 |
| Frequency | 600 MHz |
| Memory | 7936k Flash, 1024K RAM, 4K EEPROM |
| Input Voltage | 3.6-5.5V |
| Voltage | 3.3V |
| USB | Micro-USB B |
| Communication | 3x SPI, 3x I2C, 8x UART, 3x CAN, 55x digital I/O pins, 35x PWM output pins, 18x analog input pins, Ethernet |

*Table 4: Teensy 4.1 Parameters [36]*

The Teensy 4.1 board offers the option to add an additional memory flash chip. Currently, the board provides 8MB of flash memory for storing the running code, which may be sufficient for shorter flights.

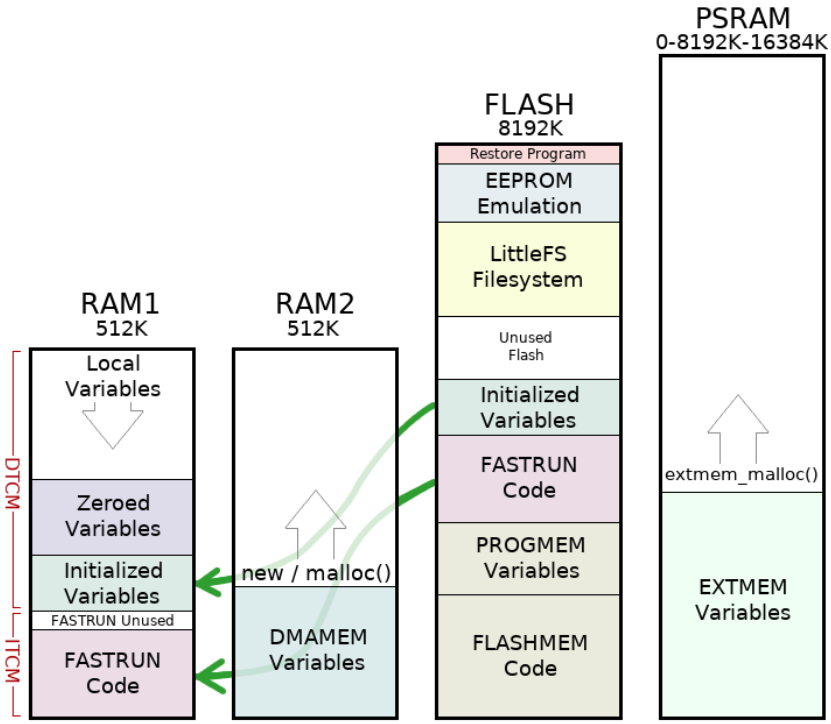


*Figure 43: Teensy 4.1 Memory usage [36]*

| Modul | Voltage | Current |
|---|---|---|
| Teensy 4.1 | 3.6-5.5V | 100mA |
| Adafruit BNO055 | 3-5V | 12.3mA |
| State RGB LED | 1.8-2.8V | 60mA |
| Adafruit BMP388 | 3-5V | <1mA |

*Table 5: FC Energy Consumption [36]*

## Peripherals

For both the IMU and Barometer, I decided to use breakout boards to simplify the process. Additionally, I prioritized sensors with a large user base for easier troubleshooting if any issues arise.

### Inertial Measurement Unit

In terms of the IMU, I considered several options such as the MPU6050, LSM6DS3, ICM20948, and BNO055. Despite being slightly more expensive, I chose the BNO055 as the best option for its user-friendly nature. It comes with built-in microcode that automatically converts data to quaternions, making orientation easy. The BNO055 is available as a breakout board from Adafruit and features an onboard Kalman filter.

To determine the orientation, I utilize an IMU based on the BNO055 chip from Bosch. This IMU combines a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer. It also includes a temperature sensor that I leverage in our system. The communication interface used for the IMU is I2C.

| Parameter | Value |
|---|---|
| Voltage | 3-5V |
| Gyroscope range | 125-2000°/s |
| Accelerometer range | up to ±16g |
| Magnetometer range | X, Y ±1300, Z ±2500 µT |
| Dimensions | 25 x 15 x 1.2mm |
| Frequency | 100Hz |

*Table 6: BNO055 Specs [36]*

The module has the capability to support two I2C addresses, which can be configured using a dedicated pin. This means that two identical modules can be connected to a single I2C bus.

One of the key advantages of this module is its integrated processing unit. The purpose of this unit is to perform reliable sensor data fusion. As a result, the IMU can provide diverse types of output, including:

- Absolute orientation in Euler Vector format (100Hz)
- Absolute orientation in Quaternion format (100Hz)
- Angular velocity vector in radians per second (100Hz)
- Acceleration vector in meters per second squared (100Hz)
- Magnetic field vector in micro-Tesla (20Hz)
- Linear acceleration vector in meters per second squared (100Hz)
- Gravity acceleration vector in meters per second squared (100Hz)
- Temperature in degrees Celsius (1Hz)

### Barometer

When considering barometric sensors, I evaluated several options such as BMP280, MPL3115A2, BMP180, LPS22HB, MS5611, and LPS25H. Among them, LPS22HB stood out as a top choice due to its superior performance compared to BMP280 and lower noise levels. However, after careful consideration, I opted for the upgraded BMP388. This choice was driven by its larger user base, slightly higher precision, and availability as a pre-made breakout board from Adafruit. It is worth noting that newer versions like the BMP390 are now available, offering even better performance.

This unit provides altitude measurements with a relative accuracy of ±60 cm. The maximum measurement frequency is 200 Hz. The sensor communicates with the MCU via the I2C bus and supports High-Speed mode with a clock rate of 3.4 MHz. The MCU can be notified of new data through an interrupt pin, ensuring timely updates. [36]

| Parameter | Value |
|---|---|
| Powered | 3-5V |
| Frequency | <200Hz |
| Dimensions | 21.6 x 16.6 x 3mm |

*Table 7: Specification of BMP388 [36]*

### Servo

In the beginning, I decided to use the well-known SG-90 servos. During research, I found that SG92R is an updated version of SG90, coming with higher torque and the same price. Also, other options like MG90S and MG90D were considered. In the end of the day, I decided to use SG902R servos.

Nevertheless, later in the project the servos are the main limitation when it comes to operating frequency, running only on 50Hz, therefore for the future iterations BlueBird high-frequency servos like BMS-207 could be considered. But because of the bigger size entire TVC Mount must be redesigned.

Another interesting thing to experiment with Is the supplied voltage. Using higher voltage values, I can get not only higher torque, but also higher response time. Nevertheless, because of the complexity of the project this issue will not be solved as part of the current scope of the project.

| **Parameter** | **Value** |
|---|---|
| Operating Voltage | 4.8V |
| Stall Torque (4.8V) | 2.5kg/cm |
| Speed | 0.1sec/60° |
| Weight | 9g |
| Dimensions | 23 x 12.2 x 27mm |

*Table 8: Specification of SG92R*

### Data Recording

Initially, there were concerns about how vibrations would affect logging on the SD card. However, it turned out to be a great idea, as logging was successfully achieved without any data corruption. But using an SD card does have a significant drawback, which is its speed. Logging is not as efficient as logging into a small flash chip. However, the advantage is that the SD card can be easily unplugged.

An alternative option is to log the data onto the chip and transfer it to the SD card after landing, allowing for easy retrieval of the data. This is something that can be addressed at a later stage, as the logging speed is currently not a critical factor.

### Summary

In conclusion, the chosen sensors for the project are as follows:

- BNO055
- BMP388
- Teensy 4.1
- SG92R
- 2S LiPo Battery

## Improvements

There are three crucial updates that are needed to be done in the Schematics to make PCB fully functional.

- Rename "GND_BATT" to "GND" for clarity.

- Correct the labeling of "V_BATT (8V)" to "V_BATT(8V)" for consistency.
- Replace the LED, as its current configuration may be causing issues and potentially damaging the Teensy.

If the PCB is printed with the current schematics, I will need to connect these pins on the board using a wire and exclude the assembly of the RGB LED. I learned this the hard way by damaging a few transistors and two Teensy 4.1 boards. The first Teensy was damaged during the protoboard development phase when I accidentally connected two transistor pins while measuring the output voltage. This caused a direct flow of over 8V from the battery to the Teensy. The second Teensy was damaged in a different manner, when I forgot to isolate servo connectors during the tests, resulting in the ground and 5V touching each other.

To summarize the possible upgrades for the current setup, the following modifications are suggested:

- Upgrade to the newer BMP390 sensor.
- Use high-frequency servos for improved performance.
- Utilize lighter connectors to reduce weight.
- Replace connectors for the Teensy with a 24-pin long female headers or stackable 12-pin.
- Use for smaller holes for current black connectors for easier soldering.
- Switch to a smaller regulator, such as the AMS1117S.
- Add a small resistor before the pyro to protect against high voltage or circuit problems.
- Incorporate a switch on the side of the PCB for better battery power management.
- Mark the GND inputs for the battery, separation, and booster connectors on the board, potentially using a dot or other clear indicator.
- Clearly mark the servo inputs on the PCB.
- Move the BMP sensor closer to the Teensy board for improved connectivity.
- Verify the servo specifications and consider powering them with 7.4V if compatible.
- Include another fuse to provide full protection for the MCU.

These upgrades aim to enhance various aspects of the setup, including sensor performance, reliability, power management, and overall functionality.

# Chapter 6: Flight Software

## Real Time Operating System

The rocket is running on RTOS, which stands for Real-Time Operating System and is focused on matching tiny deadlines.

GPOS (General Purpose Operating Systems)

- Tiny deadline can be missed if human does not spot it
- Windows, Linux

RTOS (Real-Time Operating System)

- Match tiny deadlines
- FreeRTOS, RIOT OS, Apache Nuttx, Zephyr, Arm Mbed

BAREMETAL

- Only one or two deadlines
- C/C++, Arduino, STM32, Arm Mbed

*Figure 44: Comparison Between Operating Systems. [37]*

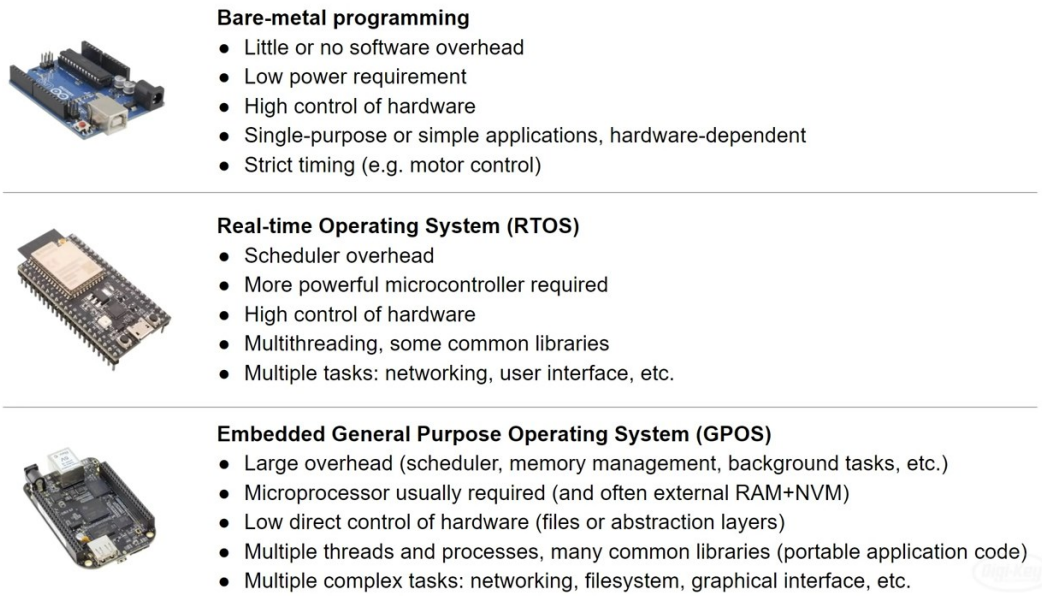


*Figure 45: Comparison of Different Operating Systems [38]*

Most people who start learning about microcontrollers use bare metal in the beginning. The code structure is simple to understand, as there is one main loop and setup function (in Arduino), which is called once before running the main loop.

In the field of microcontroller programming, interrupt service routines (ISRs) provide an elegant solution for handling unique events such as user interactions. Using interrupts can often be simpler than implementing a full-fledged RTOS since they execute immediately. However, as the number of interrupt routines increases, the code can become more difficult to read and maintain. Additionally, all interrupts have the same priority, making it challenging to determine which ones are most critical.

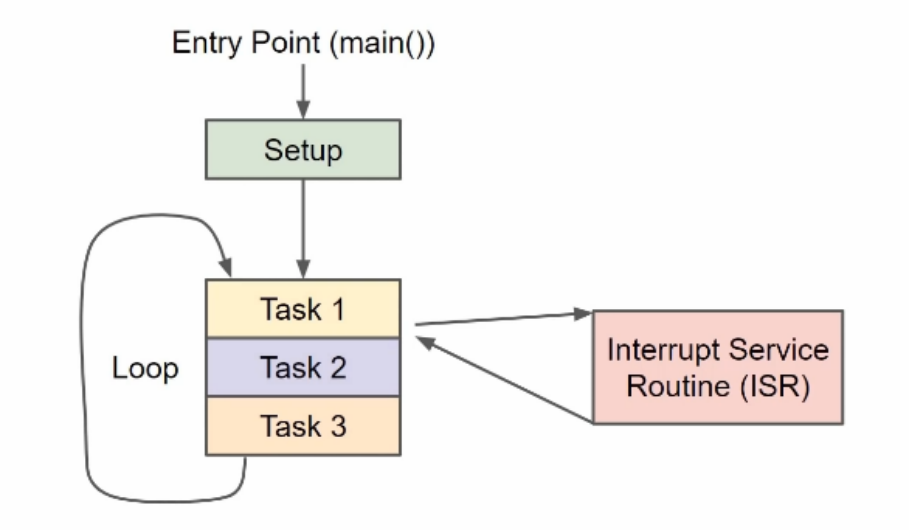


*Figure 46: Main Loop for BareMetal [38]*

An RTOS can effectively manage all interrupts by using multiple tasks at once while assigning them different priorities. This capability is particularly useful in industries such as autonomous cars, medical devices, or motor controllers, where tiny deadlines must be met. As a result, RTOS is widely used in these fields. Moreover, it can be useful while working in a team. After setting up the global variables, each task can be run separately no matter how the rest of the team is progressing. RTOS also offers the possibility to switch to a new board with the same code, making it easy to prototype and upgrade. For example, PlatformIO and Arm Mbed provide simple setups for different boards. [37]

The growing popularity of RTOS is evidenced by the many major companies investing in their own RTOS systems, including Amazon, which acquired one of the most significant open-source projects - FreeRTOS, Microsoft, which acquired Azure, and Google, which is developing its operating system.

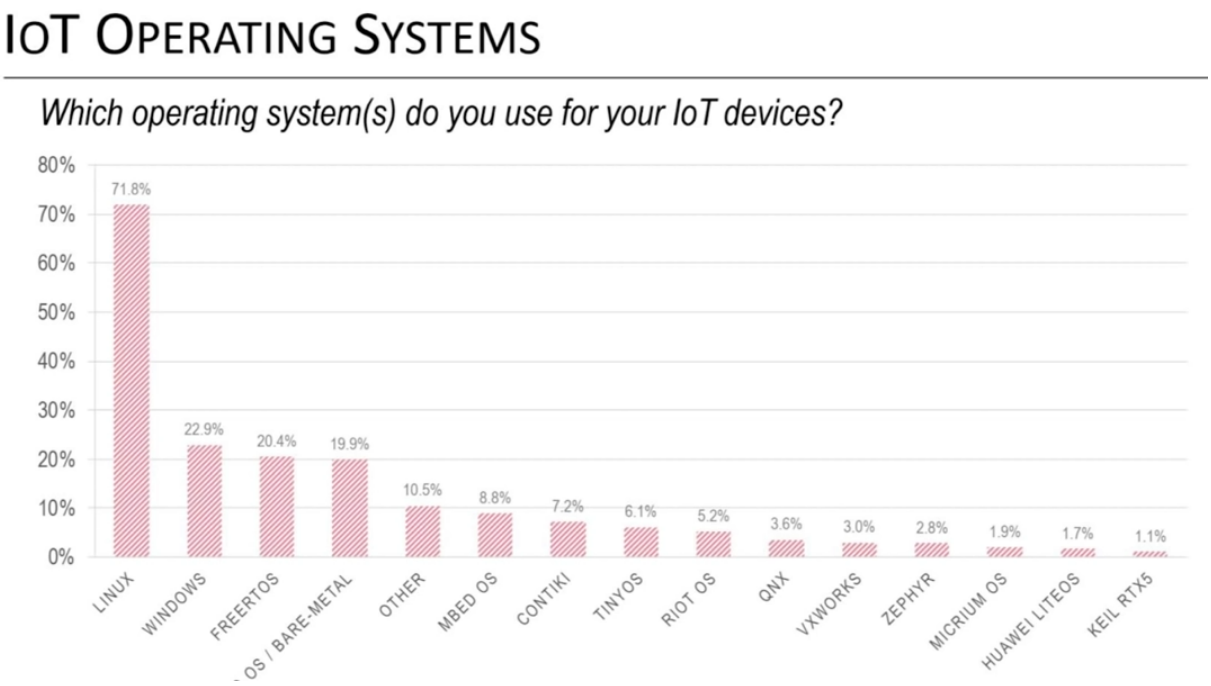


*Figure 47: Operating Systems Used for IoT [39]*

To avoid unexpected behavior when different tasks use the same variable, RTOS provides programming methods like mutex and semaphores, which can block specific code parts. Mutex works like a key to a bathroom, where the key blocks access to the room until it is returned. Similarly, mutex locks specific code parts until they are unlocked. Semaphores work like a library study room, where a limited number of students can use the same room. This method ensures that a limited number of processes use the same code at the same time, preventing conflicts. [37]

To sum up, RTOS opens an entirely new world for microcontroller programming. I can write more readable, general code that can be easily transferred between different boards and even between other operating systems. A simple summary of the positives and negatives is below.

Benefits:

- Board Variety
- Task Prioritization
- More readable (POSIX Compliant) code
- Possibility to run tasks in parallel.
- Easier cooperation within the team
- Growing market and increasing popularity
- Improved workflow after setting up the system.
- Any IDE can be used for programming (including VS Code)

Downsides:

- Task Scheduler consumes some computational power.
- Complex for beginners
- Most RTOS recommend Linux.
- Basic knowledge about the command line is needed (not always!)

## RTOS Comparison

When choosing an RTOS for a project, it is important to consider a wide range of factors. When it comes to handling rocket sensors, I need to prioritize the availability of sensor drivers over factors like low battery consumption or communication protocol support. To decide, I should focus on RTOS options that are actively developed, well-documented, community-supported, and easy to set up. To help us narrow down our options, here are some important criteria to consider:

- Project purpose
- License
- Number of active contributors
- Documentation
- Easy to follow tutorials.
- Board selection
- Examples
- Supported Sensors

- POSIX Compliance

The choice of Apache Nuttx as an RTOS for a project can be justified on several grounds. Firstly, Nuttx is a fully POSIX compliant RTOS, which means that it follows widely known function calls and can lead to more readable and understandable code. The use of POSIX compliant RTOS allows developers to use the same function calls as during programming in C/C++ on their own computer and while coding on the microcontroller. Moreover, the kernel can be debugged on computers using Valgrind or GCC debuggers because Windows and Linux are POSIX compliant.

Nuttx comes with very advanced configuration possibilities, where I can set up everything about the board. This advanced level of configuration makes it possible to customize the RTOS according to specific project requirements.

In summary, when it comes to choosing an RTOS for a project, Apache Nuttx is a strong contender due to its full POSIX compliance, versatility in board support, and advanced level of configuration. While RIOT OS and Zephyr are also viable options, Nuttx provides a unique set of features that make it the preferred choice for many developers. RIOT OS is best suited for IoT applications, while Zephyr is gaining popularity and is likely to receive official security certification.

When it comes to other RTOS options, FreeRTOS is a well-known choice, but its decreasing popularity and the fact that it is owned by Amazon may not make it the best fit for all projects. ARM Mbed (OS 6) offers a wide variety of compiler options and covers peripheral devices but is only compatible with ARM processors and has a larger code size and complex structure. Zephyr, supported by the Linux foundation, is popular for IoT usage, but may be difficult for beginners to set up. Overall, it is important to consider project needs, documentation, tutorials, board support, and compliance when selecting an RTOS.

All the functions would be called from the main loop in bare-metal programming, making it extremely hard to cooperate with the team at the same time. Having RTOS, everyone can work on their task independently, and when everything is ready, all the tasks can be run in parallel.

The only downside of using RTOS is the task schedular, which must be run every tick (tick = time when OS asks which task to run, around 1ms). The operating system needs to look at which task has the highest priority to be run, which takes extra computational power. Therefore, a bit more powerful microcontrollers are recommended for using RTOS (an excellent choice is 160+ MHz).

Using the schedular, I can run multiple tasks "in parallel" even on MCU with a single core. The schedular will switch between the tasks at high frequencies. Therefore, MCU behaves as if the tasks are running in parallel (although only one task is running simultaneously).

## Flight Software

The FSW runs on the Nuttx RTOS, compiled using gcc arm on Linux (or WSL), and is loaded into the Teensy using the Teensyloader. This is currently the only way to utilize Nuttx since it is not integrated

into other platforms like PlatformIO. Each process is executed with a specific periodic frequency defined for each state. The FSW structure is designed to accommodate future development of parallel processes. It is divided into four separate processes, as depicted in Figure 48. The software blocks are separated into four different tasks, each running separately with a different priority.

- State Machine: highest priority, responsible for switching between the states during the flight.
- Sensors Reading: reading from BMP and BNO
- TVC: using the measured data, use the PID to adjust the servo position.
- Output: logging to microSD card

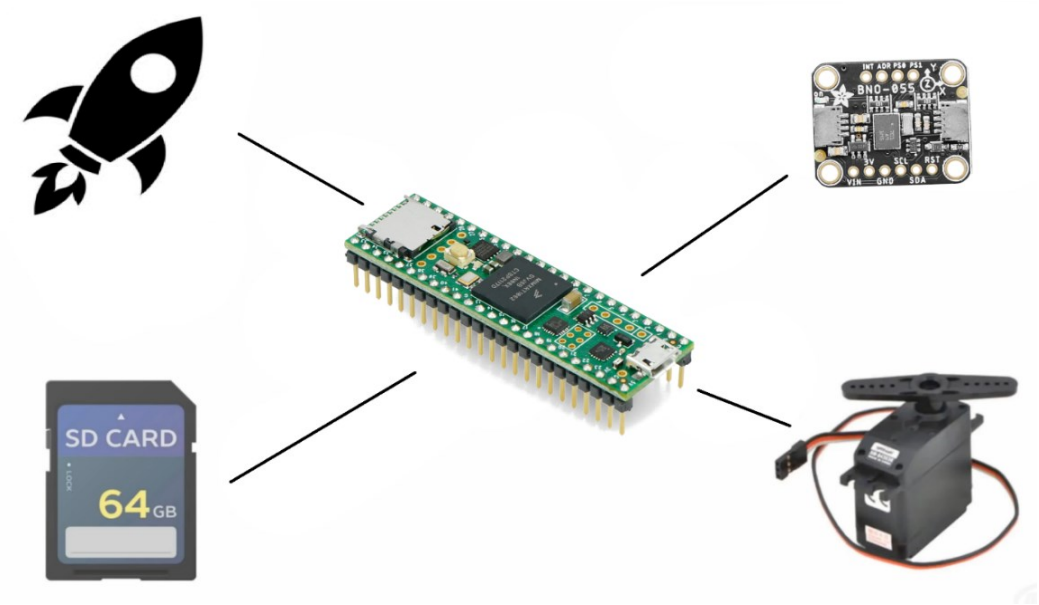


*Figure 48: Four Different Tasks for the FSW*

The Flight Software operates as a state machine, storing the current state in a variable and precisely defining transitions. The possible transitions are illustrated in the Figure 49.

- POWERUP (1Hz) – start of the system, first setup loop to initialize all variables, test all four corners with a max angle and then put motor into the middle position, if anything goes wrong (SD Card or RBF missing, goes into FAILMODE)
- IDLE (10Hz)– go around with the motor to check whether all the servos are working correctly, then wait for the sensors to read stable values for around one second and for the rocket to be in the upright position.
- READY (10Hz) – waiting for the remove before flight pin to be removed.
- ARM (100Hz) – signals the arm state with the external diode, starts logging data, waiting for the liftoff, detected by the BNO055 sensor accelerating in Z axis.
- FLIGHT (50Hz)– controlling the flight using servos, waiting for the rocket to stop accelerating, rotating and the values from BMP being stable (60cm precision)
- LANDED (1Hz) – while true loop, not logging data
- FAILMODE (1Hz) – external LED starts blinking inside while(true) state.
- RESET – checked every iteration, if the reset cable is inserted, goes back into the powerup state.

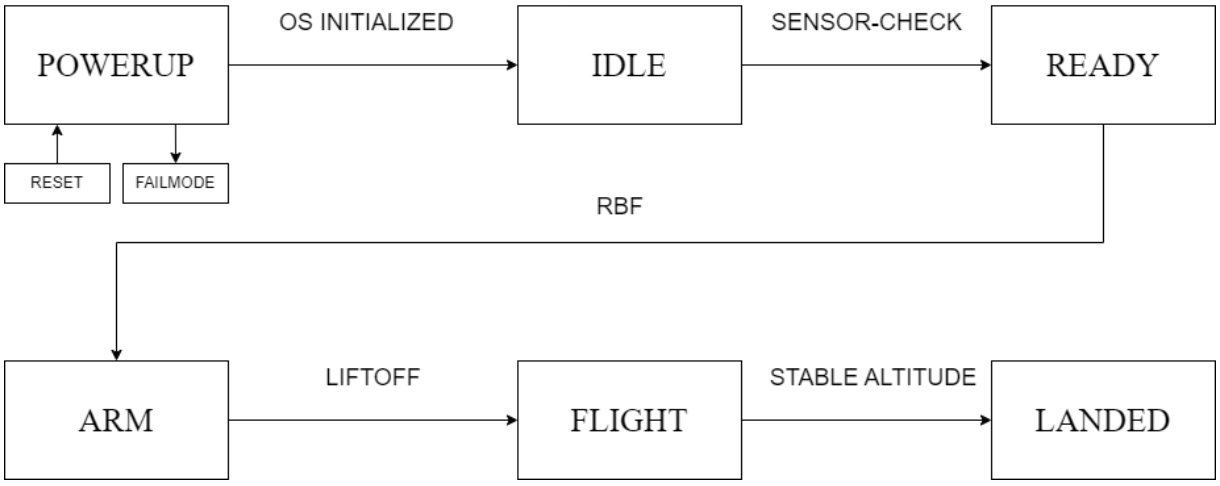


*Figure 49: State Machine Diagram*

The flight sequence runs at 50Hz due to the limitations of the servos, which cannot operate at higher frequencies. During each flight cycle, I perform the following tasks:

1) Read the sensor values.
2) Compute the error and pass it through the PID command.
3) Convert the PID output into values that can be understood by the servos.
4) Move the motors accordingly.
5) Repeat from step 1.

Note: This sequence of tasks is repeated continuously throughout the flight.

## Code Implementation

Now I will delve into the implementation and explain how the rocket operates through the code. The code is implemented in C, and the state machine operates as a list of function pointers.

```
int (*arr[11])(struct data *) = {
    &powerup,
    &idle,
    &ready,
    &arm,
    &flight
    &landed,
};
```

*Figure 50: State Machine Pointers*

The states of the rocket were originally intended to be signalled by different blinking patterns of the RGB LED. However, due to a mistake in the schematics, I am unable to use the RGB LED to indicate the states as intended. I store the names of each state into enum *states*. Then in the main while loop, I can easily call the necessary functions and update the states by utilizing the return values of the functions.

```
  while (data.fsw.state != END)
 {
   data.fsw.cycle++;
  data_processing(&data);
  data.fsw.state = arr[data.fsw.state](&data);
  logging(&data);
 }
```

*Figure 51: Main Loop*

An example of the "ready" state can be seen below

```
int ready(struct data *data)
{
   static bool last_readings[ARR_SIZE] = {0};
   data->fsw.cond = sum(last_readings, ARR_SIZE);
   if (data->fsw.cond >= PERCENTAGE * ARR_SIZE)
   {
      fprintf(data->fsw.fd_stdout, "%6d ARM - key removed\n", data->fsw.cycle);
      return ARM;
   }
   else
   {
      cond = (value_rbf == 0);
      index = check_condition(last_readings, cond, index, ARR_SIZE);
      return READY;
   }
}
```

*Figure 52: Example of Ready State*

Regarding the frequencies, I am operating at the maximum speed since the flight sequence from booting to launch is short. Therefore, there is no risk of exceeding the capacity of the SD card (up to 16GB) or draining the batteries during this time.

## Nuttx Setup

To prepare the SD card:

1. Format the SD card to FAT.
2. Create a file called DONT_TOUCH_IT.txt (case sensitive) on the SD card. Inside the file, write the log number, starting from one if the SD card is empty.

To upload the code to the Teensy:

1. Insert the SD card into the Teensy.
2. Connect the Teensy to your computer.
3. Open the Teensy loader and click on "Auto" to upload the code. The loader will use the binary file from the previous directory, so you do not need to move any files.

To configure Nuttx and compile the code:

1. Open the command line and navigate to the Nuttx directory.
2. Run the command: ./tools/configure.sh -l teensy-4.x:project_alpha (this configures the menu options).
3. Run the command: make (this compiles the code, which may take few minutes).

To upload the compiled code to the Teensy:

1. In the Teensy loader window, select the compiled hex file and upload it to the Teensy.
2. Press the button on the Teensy to transfer the code.

To check the output and run the application:

1. Open a serial port program (such as Kitty) and configure it to use the appropriate COM (Communication Protocol) port and baud rate (115200).
2. Press "enter" three times in the serial port program to open the NSH shell on the Teensy.
3. Type cansat_fsw_01 in the program to run the application.

Note: If the Teensy starts blinking, it indicates a critical error has occurred. [37]

## Improvements

While neither of these improvements are crucial, I can consider the following enhancements:

- Recording data to a chip and transferring it to the SD Card after landing.
- Finding a way to smoothly implement circular servo movement instead of the current square movement where each axis is treated separately.
- Adding code that moves the rocket out of the stand, ensuring it initially flies at a slight angle to avoid landing on top of the stand in future flights.

In terms of servo movements, limiting them to circular paths seems like a sensible choice, as it is easy to implement and ensures consistent behaviour in all directions. However, this requires determining how to assign appropriate angles to each servo. When one servo is at a zero angle, there is no issue. However, when one servo reaches its maximum degree and the other servo also needs to reach a specific angle (e.g., 5°), a logic is needed to dynamically adjust the maximum range for each axis.

# Chapter 7: System Simulation

Simulations play a critical role in rocket projects. However, launching without conducting thorough simulations is possible but not advisable. Rocket launches are expensive, and without proper simulations, large-scale rocket projects will not be feasible.

The purpose of simulations is to minimize the chances of any mishaps. If I can accurately simulate the rocket's behavior, the likelihood of something going wrong is reduced. Simulations help save costs by preventing damage to rocket parts and motors. In this section, I will delve into this topic further.

## Inspiration

Having an inaccurate simulation would be disastrous. Every launch incurs expenses for rocket motors, and there is also a risk of damaging rocket components, thereby prolonging the entire rocket development process. To mitigate this risk, I drew inspiration from Hanbergs Space Mission [40], which successfully launched a TVC Rocket Model. Therefore, there was a high probability that the PID Simulation for their patrons was effective and could serve as an excellent starting point for my thesis.

Several improvements were made to the original version:

- I fully discretized the simulation, running it on the same frequency as our Flight Computer to ensure that the simulation data reflects reality.
- I adopted a better approach to simulate the delay in servo movements. This involved considering the delay between the signal from the MCU and the actual servo movement, as well as accounting for the servo's speed by examining the angle difference between its last and current positions.
- The model was simplified by merging the addition and division blocks into one, resulting in a more straightforward and manageable structure.
- By employing a different approach to compute delays, I was also able to make the auto-tuner function work (although there may have been a license-related issue rather than an inherent flaw in the model).

## Measurements

To accurately simulate the rocket's behavior, I needed to measure several values, as discussed in Calibration. Throughout the testing phase, I utilized three setups: Rocket Model I, which was not actively stabilized and is not relevant to this chapter; Rocket Model II, used for the second and third flights until I discovered that the rocket was too heavy, leading us to remove the unnecessary middle part; and the remaining tests were conducted with Model III. Since the bottom part is the same, the servo delay remains unchanged for both rockets.

### Rocket Model II:

- Weight: 473g
- MOI : 0.0172 kg $m^2$
- Servo Reaction time 0.033333s
- Servo time delay: 0.007872s
- COM to TVC distance: 0.241m

### Rocket Model III:

- Weight: 405g
- MOI: 0.006951 kg $m^2$
- Servo Reaction time 0.033333s
- Servo time delay: 0.007872s
- COM to TVC distance: 0.155m

I will discuss only Rocket Model III in the rest of this chapter. To compute these values, I used the two blocks shown in the picture below. The weight and moment of inertia need to be manually entered into the 3-DOF block, while the servo delay results are automatically applied.

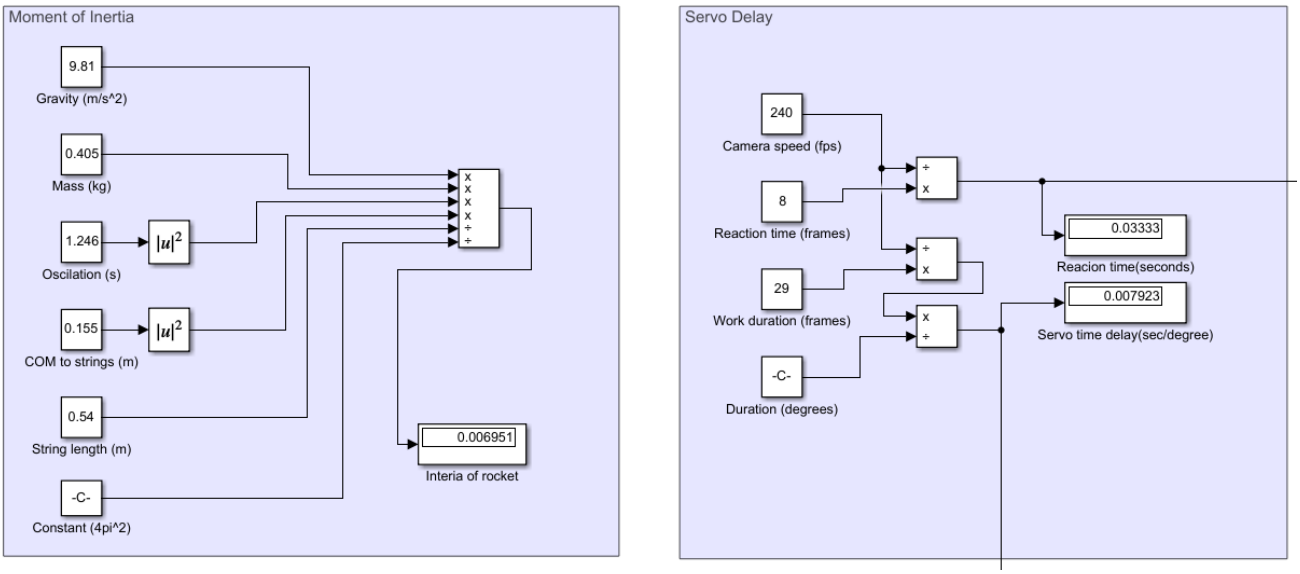


*Figure 53: MOI and Servo Delay Calculations*

## System Description

To achieve discrete control operating at a speed of 50Hz, I configured the model parameters with a fixed step of 0.02s. In the simulation, I considered both the servo delay and the discrete nature of the system, as it is not continuous. The rocket's physics are simulated using a 3-DOF block, where I apply forces to the rocket's TVC in the upward and sideways directions (X and Y axes). These forces are computed based on the Motor Angle, as shown in the picture below. Additionally, I provide the distance from the COM to the TVC, which is later needed in the 3-DOF block.

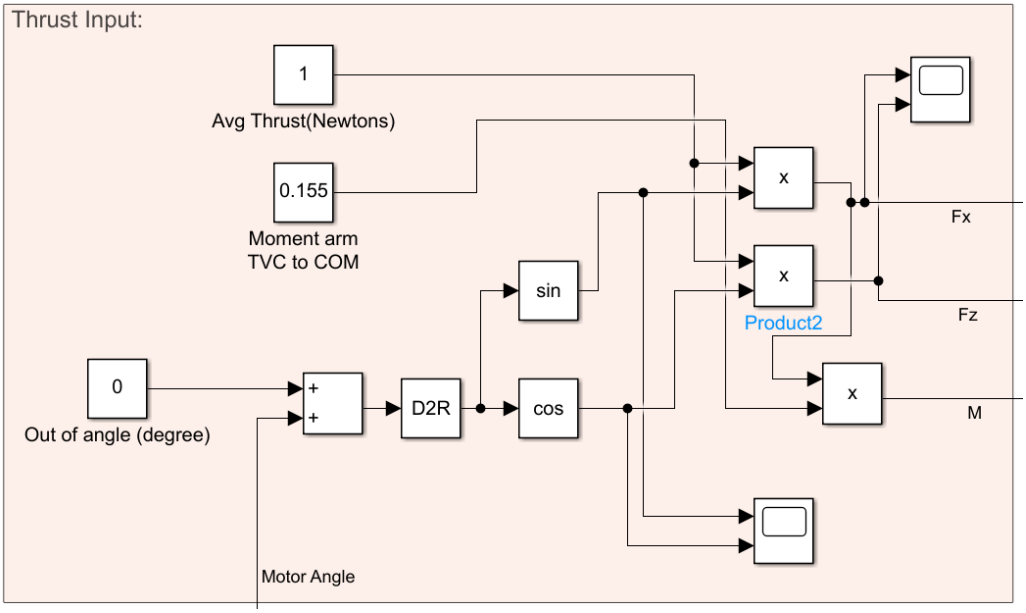


*Figure 54: Thrust Input*

Among other outputs, the 3-DOF block provides the rocket's orientation in radians. Together with other variables such as drift or altitude, I visualize these outputs using scopes, which can be found below.

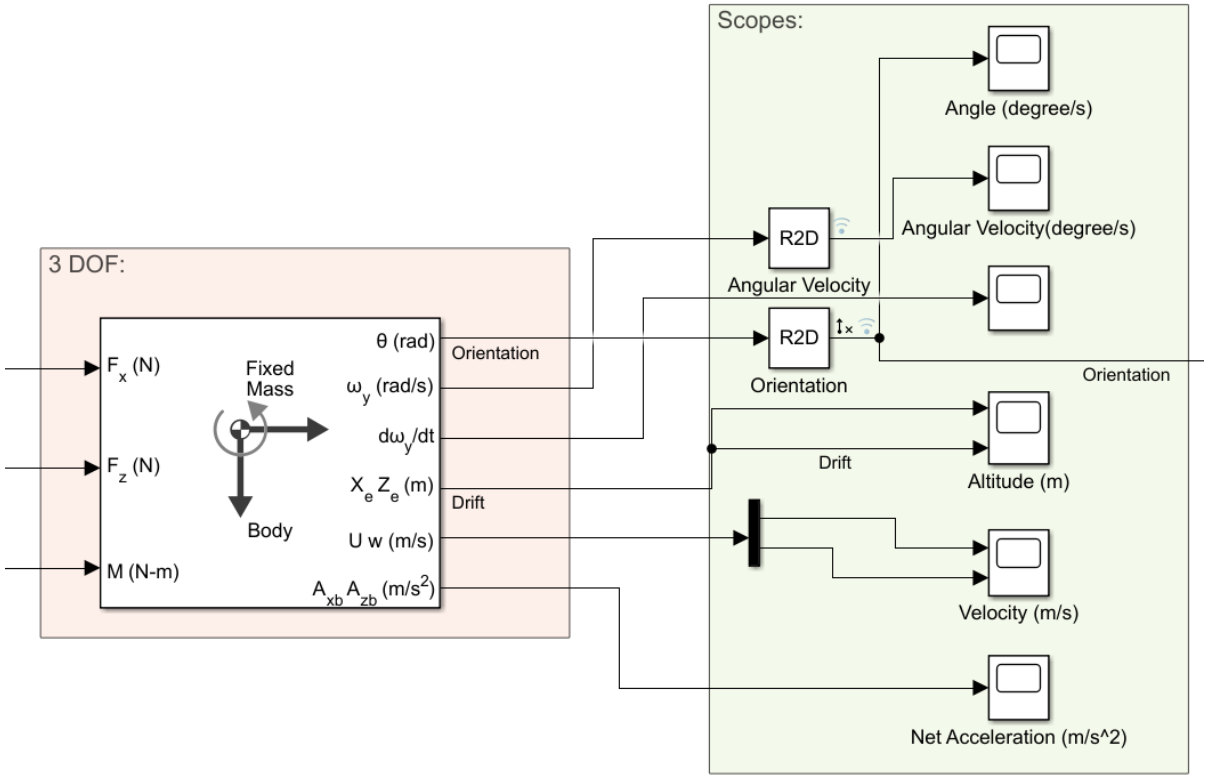


*Figure 55: Three Degrees-of-Freedom Block and Scopes*

The rocket's orientation is crucial for determining the ideal TVC response. I measure the orientation using the BNO055 sensor. In real-life scenarios, I aim to have the orientation equal to zero, making the rocket orientation the error input to our PID controller. However, for simulation purposes, I simplify the problem by trying to move the rocket to a desired angle called the Target Setpoint. At the beginning of the simulation, the rocket is perfectly straight, and I tune the values to achieve the fastest and most realistic

response of the motor angle, orienting the rocket to face 3 degrees instead. Although the Target Setpoint would be 0 and the orientation might be, for example, 3 degrees, the difference between these two values remains the focus of our simulation.

Once I obtain the orientation error, I input it into our PID controller. Here, I use tuned values and delay this signal based on the servo reaction time and the servo speed. The servo reaction time represents the time between sending the signal from the MCU (Microcontroller Unit) and the first movement of the servo, which in our case is 0.033333s. Then, based on the degrees that I need to travel (output from the Difference block), I multiply it by the time the servo needs to travel one degree, which is 0.007872s. These two delays are summed up and fed into a discrete delay block. Without these delays, the motor would be able to react immediately to any error in the rocket's orientation, resulting in a very stable system with much better response, but unrealistic. The block representing this delay can be seen below, along with a tuned response for a 10N thrust input, which is our peak value during flight.

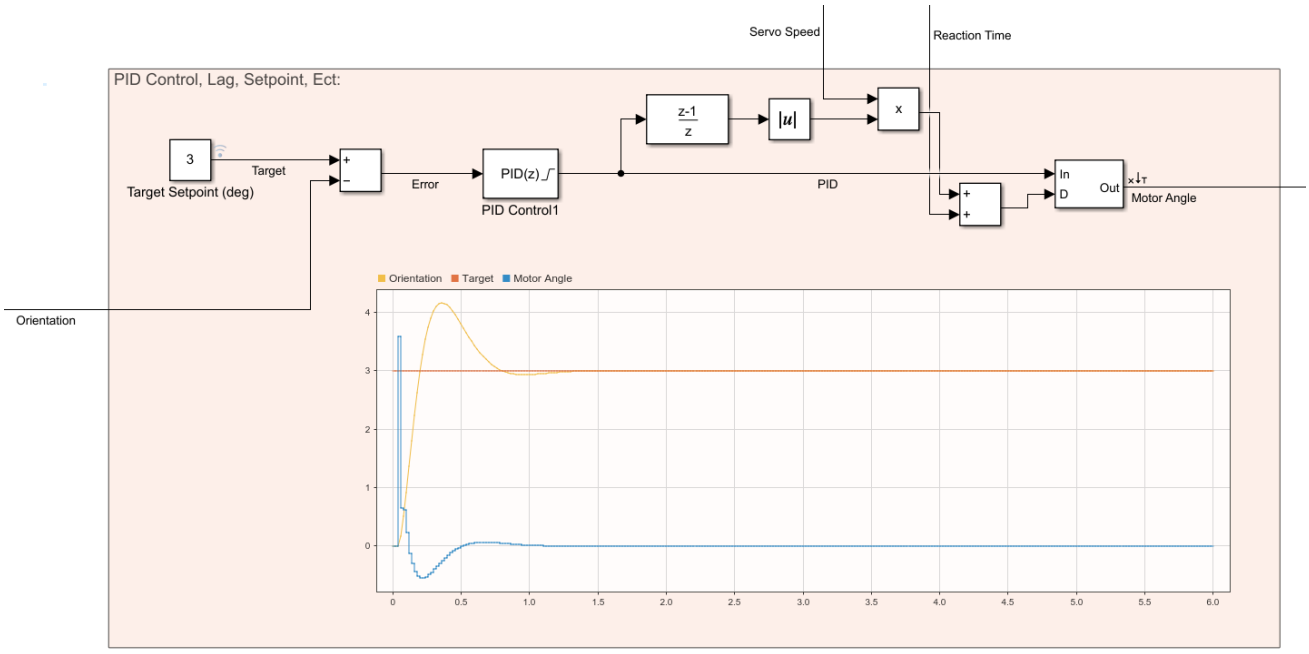


*Figure 56: PID Response*

From this system, I obtain the desired Motor Angle, which is then fed back to the first block for computing the horizontal and vertical forces applied to the rocket. Using the mentioned transfer function, I control the movement of the servos. The entire system is below.

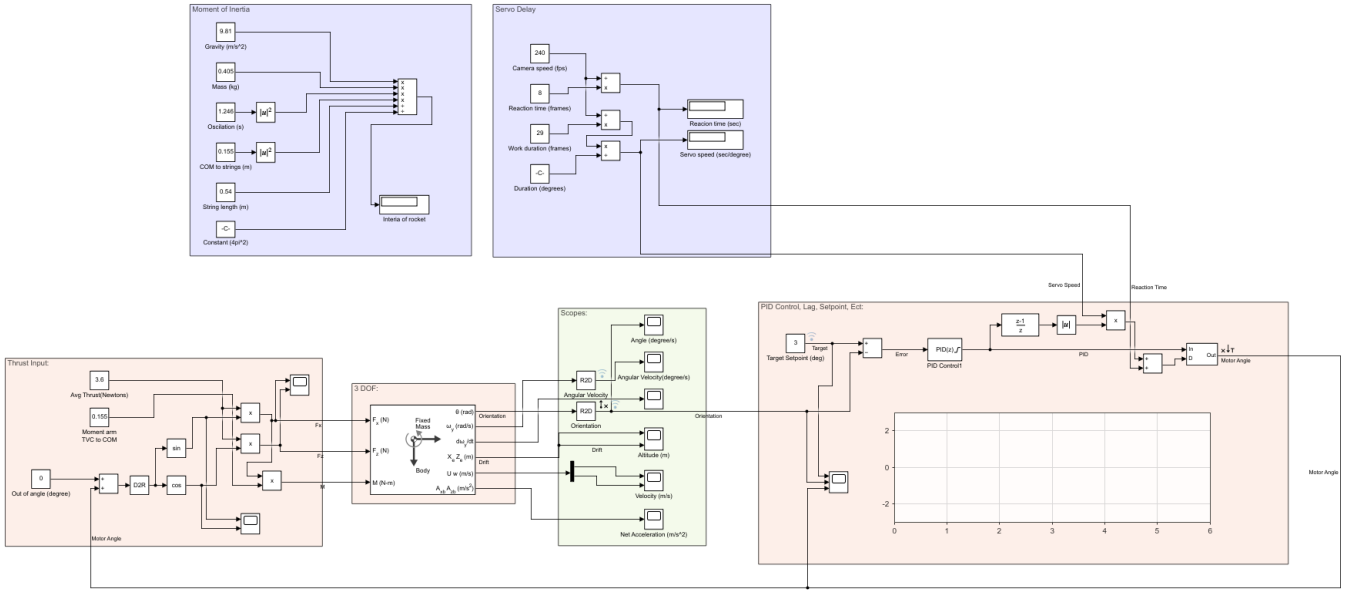


*Figure 57:Complete Simulink for Rocket Simulation*

## Linearization

To enhance comprehension and facilitate smoother simulation of our system, it is beneficial to linearize it into a transfer function. I achieved this by employing the Model Linearizer, which is included in a linear control toolbox. The transfer function was obtained using Linear Analysis Points, with the motor angle as the input and the rocket orientation as the output for our open-loop analysis. The toolbox automatically selected the operating point for linearization. While I cannot solely rely on the result, after thorough testing and comparing the response of the linearized model transfer function with the actual system, I observed that both systems exhibited the same response.

It is worth noting that our model belongs to the category of a Linear Parameter Varying System (LPV), which means that the transfer function varies based on the thrust parameter. For instance, let's consider the transfer function for a motor thrust of 1N, which is as follows:

$$H(z) = \frac{0.00446z + 0.00446}{z^2 - 2z + 1}$$

Since I need to simulate the thrust that I am tuning our PID controller for, I can easily rewrite the transfer function as follows:

$$H(z) = F_{motor\_thrust} \frac{0.00446z + 0.00446}{z^2 - 2z + 1}$$

Using this transfer function, I can simplify the system as could be seen in the picture below.

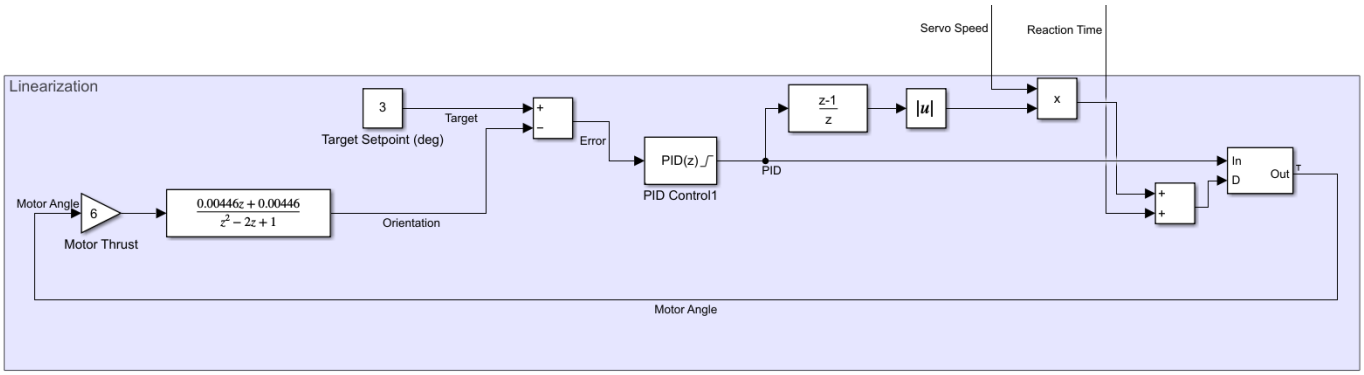


*Figure 58: Linearized Model*

From the transfer function, I can observe that the system has:

- A zero at -1
- A double pole at +1

Since the system is discrete, the stability of the system is determined by the location of the poles inside the unit circle. In this case, both poles are on the boundary of the unit circle. Further analysis and considerations are required to assess the stability of the system.

## PID Tuning

Before delving into the tuning procedure, it's important to mention that the following values need to be adjusted based on your specific rocket specifications:

- 3DOF Block
- Average Thrust
- Servo Reaction Time
- Servo Delay
- Transfer Function
- TVC to COM

When it comes to PID tuning, I have two options: manual tuning or automatic tuning. Due to the dynamic changes in the thrust curve during the flight (ranging from 3.2N to 10N), finding values that remain stable throughout the entire flight duration can be challenging. However, our primary focus is on the range between 3.2N and 4.3N, as the high peak of 10N occurs only at the beginning.

For PID tuning, it's crucial to note that I need to limit the PID output to a range of -5 to +5 degrees since that corresponds to the maximum motor reach. While clamping the integrator is recommended, it's not mandatory.

## Automatic Tuning

If I opt for automatic tuning, I can utilize the built-in autotune feature. However, after conducting multiple experiments, the tuned values were not satisfactory. The reason behind the auto-tune's inefficiency lies in the fact that it does not account for time delays, which are essential for accurately representing the behavior of our system. Moreover, the tuned values tend to heavily depend on the derivative term, which

is the easiest to tune. Consequently, after careful consideration and unsuccessful attempts, including experimentation with a low-pass filter, I have decided to switch to manual tuning as the auto-tuning methods did not yield satisfactory results.

One potential solution to address this issue would be to force the auto-tuner to use values for each term within a specified range. Despite the PID block's capability to limit these values, the auto-tuner surpasses those limits. Here are examples of the best responses obtained from the auto-tune, including the filter coefficient N:

Average Thrust 3.2N

- Proportional (P): 0.0156278124028587
- Integral (I): 0.000817563487792593
- Derivative (D): 0.074681821089624
- Filter Coefficient (N): 15

Average Thrust 10N

- Proportional (P): 0.0620795512830916
- Integral (I): 0.021487467978605
- Derivative (D): 0.0417260881931518
- Filter Coefficient (N): 49.9178863294947

However, all these values prove to be unstable when the servo delay is applied. As a result, I attempted auto-tuning using only a PI controller, but it was not possible to find a stable solution, indicating the necessity of the derivative term.

## Manual Tuning

I achieved greater success with manual tuning. Initially, I experimented with the Ziegler-Nichols approach, but it was not possible to stabilize the system using only the Proportional term. Nevertheless, after trying different parameters, I discovered alternative solutions with broader gain values.

Each component of the PID controller is tuned for a specific average thrust. For simplicity's sake, until I achieve at least one successful flight with thrust vector control, I will employ the same gains for the entire flight. Thus, I need to select gains that remain stable during the initial peak in thrust while effectively controlling the rocket throughout the later phases of the flight. Once this approach delivers satisfactory results, I can start experimenting with dynamically changing the gains.

All manually tuned values can be seen in Table 9. I am searching for the best gains for the entire thrust, aiming for the most stable response.

| Average Thrust | Proportional (O) | Integral (I) | Derivative (D) |
|---|---|---|---|
| 3.6N | 0.541875 | 1.265625 | 0.1122 |
| 6N | 0.20825 | 0.32 | 0.04985 |
| 10N | 0.2046875 | 0.46875 | 0.0403125 |

*Table 9: Tuned PID Values*

To find the best possible gain, I need to examine the system response. In the graphs provided below, I simulated the system for 6 seconds, which was chosen as an optimal time to observe the system's response. It is also worth noting that the motor's burn time is 6 seconds. The objective is to start from 0 degrees and achieve stability at 3 degrees. In the provided graphs, there are three different levels of thrust. The first three graphs correspond to an average thrust of 3.6N, the following three graphs represent a thrust of 6N, and the final three graphs represent a thrust of 10N.

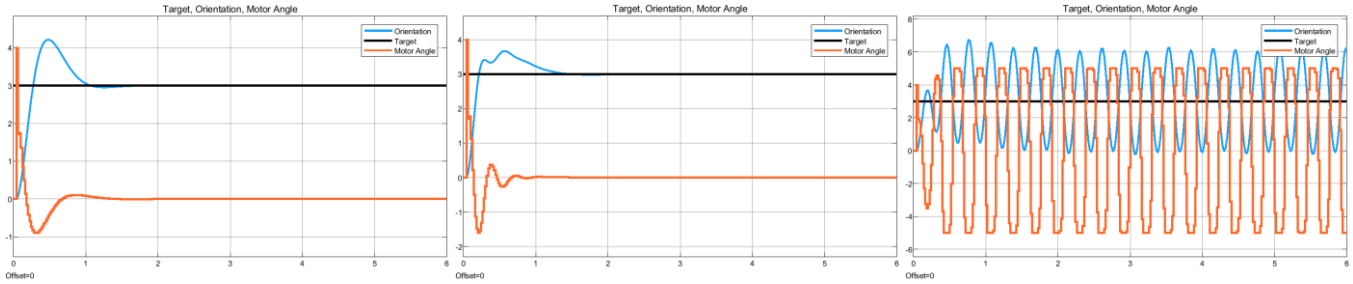


*Figure 59: System Response for a 3.6N Thrust.*

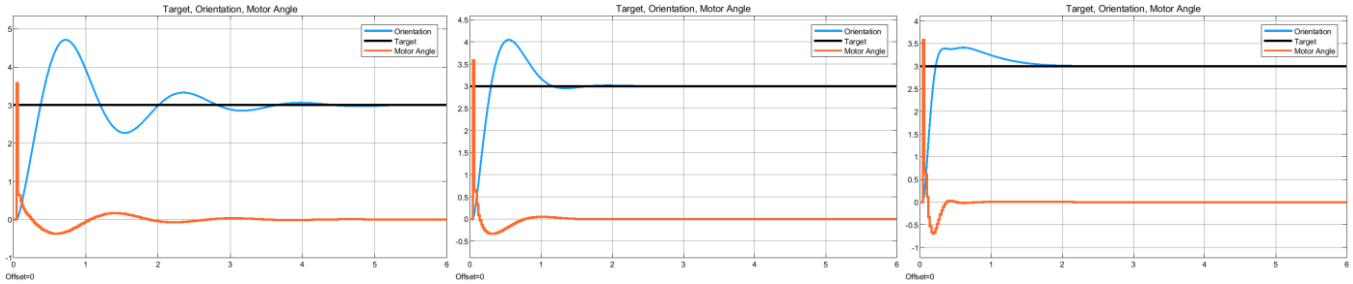


*Figure 60: System Response for a 6N Thrust.*

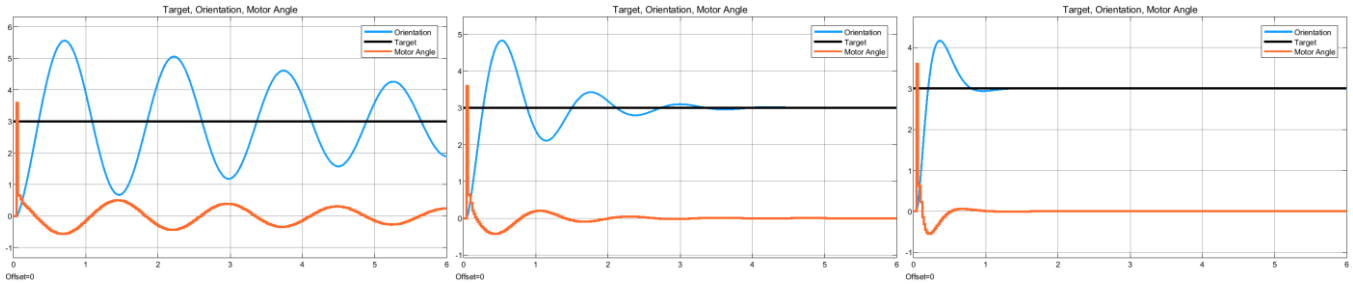


*Figure 61: System Response for a 10N Thrust.*

The values for 3.6N thrust work well until reaching 6N, but once I reach the maximum peak, the oscillations become too risky. Considering the potential impact of external forces like wind gusts, it would be prudent to explore alternative options. On the other hand, the graphs for 10N thrust display excellent performance during the initial peak. However, once the thrust stabilizes at 3.2N, there is a significant risk of losing stability. The 6N option strikes a perfect balance between the previous two. It stabilizes quickly in the initial phase of the flight and maintains stability even after reaching the linear and more stable portion of the thrust curve. By utilizing this setup, I can achieve great results. Therefore, I have decided to use these values as the best option I have tuned so far.

## Model Verification

To validate our tuned PID values, I came across an online tool that offers even more comprehensive simulations, incorporating factors such as wind, noisy sensors, and flex of the servo linkages. You can access the tool through the following [41]. After inputting the measured values, I obtained identical responses, reaffirming the accuracy of our simulation and tuned values.

With the successful verification of our simulation and tuned values, I can confidently consider them as correct and reliable for our project.

## Improvements

There are two major improvements that I can experiment with:

- Simulating the entire motor thrust curve: Currently, our simulations use an average thrust value. However, since I have measured the actual thrust curve, I can easily incorporate this data into our simulations. By doing so, I can gain a better understanding of the rocket's behavior throughout the entire flight.
- Dynamic changes in PID values: Another improvement I can explore is dynamically changing the PID values since liftoff. With precise measurements of the rocket's liftoff time and the time it takes to reach the thrust peak from the measured data, I can actively adjust the PID values accordingly. However, there is one challenge to consider. Rapid changes in PID gains can result in abrupt changes in motor angles, potentially causing instability. To mitigate this issue, I can implement a method called "transient-free switch," gradually transitioning between the gains over a certain period. Additionally, I can create a function to interpolate the gains based on the time since liftoff (i.e., thrust), allowing for smoother and more controlled adjustments.

# Chapter 8: Rocket Launch

## Expected Flight Phases

The flight is divided into five phases, including a landing phase. The points below illustrate an ideal flight.

- PHASE I – Liftoff: At this moment, the solid motor ignites, producing the highest thrust throughout the flight. The rocket begins ascending towards its apogee.
- PHASE II – Apogee: The rocket reaches its highest point during the flight. The motor thrust peak is over, and the rocket starts a gradual descent.
- PHASE III – Descent: The rocket slowly descends while maintaining upright stability.
- PHASE IV – Burnout: The rocket's motor ceases burning, indicating that the rocket is close to the ground.
- PHASE V – Landing: The rocket contacts the ground.

## Preflight Check

Before moving to the launchpad, the following tasks need to be completed:

- Verify the center position of the motor.
- Check that the data is logging correctly.
- Place the battery inside the rocket on the opposite side of the servos.
- Ensure both servos are moving the motor within the range of -5° to +5°
- Check whether the battery is fully charged.
- Set up the cameras for each axis to record the flight.
- Use measuring tools to ensure the stand is perfectly facing up, do not rely on visual estimation.
- Assemble the rocket without inserting the SD Card.
- Double-check that the rocket is assembled correctly.
- Insert the SD Card.
- Cover the SD Card with tape.
- Connect the battery.
- Secure the cables to prevent them from getting stuck in the TVC mount.
- Reconfirm that the rocket is assembled correctly.
- Turn on the cameras.
- Calibrate BNO055 by moving the rocket in all directions.
- Insert rocket inside the stand.
- Remove RBFP
- Immediately after the landing save the log file.


*Figure 62: Remove Before Flight Pin.*

## First Flight

Date: 7.7.2023

The very first flight was conducted as a preliminary test to determine the feasibility of the project. I used a rocket model from Czech Rocket Challenge with adjusted fins. These fins were used because I was uncertain if the TVC Mount would be ready for testing at the scheduled time. As a substitute for TVC, I downscaled the available online Delta R2 model [27].

### Video

*Figure 63: First Flight Video https://www.youtube.com/watch?v=SHa_8S-sUho*

### Goals

- Test the Klima D3-P Motor.
- Experiment with 3D printing.
- Buy all the necessary tools for building the rocket.
- Gain a basic understanding of the project's complexity.
- Complete the protoboard Flight Computer.

### Rocket Model

This flight served as the first and only launch for this rocket. In preparation for the upcoming flight, I have upgraded the design of both the rocket and the TVC Mount. The weighted parts for this specific flight are listed below.

- 202g ..... 3D printed parts (266g including the middle tube that was excluded from the flight).
- 25g ..... Flight Computer (excluded from the flight).
- 28g ..... Rocket Motor.
- 62g ..... LiPo battery.
- 58g ..... TVC.
- 30g ..... Screws, nuts, and pushrods

Total weight: 380g.

No calibration was necessary for this flight as the TVC was not utilized due to issues with incorrect dimensions for the servo holders and a malfunctioning flight computer.

### Flight

The flight was successful. I achieved all our goals except for completing the Flight Computer. I accidentally burned the regulator while measuring the voltage, resulting in the destruction of the Teensy 4.1. Additionally, since I downscaled the online STL file, I did not realize that the servo mounts would be smaller than the servo itself, making it impossible to fit. Moreover, the rocket's diameter was too small to accommodate the TVC mount with the current servo setup, making the TVC unusable.

However, despite not using TVC for this flight, I gathered valuable information:

- The motor was strong and reliable enough to lift the rocket and ensure a successful flight.
- The 3D prints were of high quality, demonstrating the effectiveness of 3D printing for rocket body construction.
- I had all the necessary resources for future rocket development.
- I recognized that the project posed complex challenges but decided to take the risk and continue its development.

*Figure 64: First Launch Rocket Ready for a Lift-off*

### Improvements

For the next flight, our focus will be on:

- Developing our own rocket model.
- Designing a brand new TVC Mount.
- Creating a PCB for a new flight computer unit.
- Modeling a launchpad.

## Second Flight

Date: 30.4.2023

Prior to liftoff, I addressed all previous improvements. I completely built and modeled a new rocket with a launch pad, developed a new flight computer, and created a TVC mount. However, I faced an issue when the SD Card broke due to insufficient space. I had to replace it with another one to ensure the software could run. The second attempt was successful, but I recognize the need for improvement in this aspect before the next flight.

### Video

*Figure 65: Second Flight Video https://www.youtube.com/watch?v=SV80jhYoOVQ*

### Goals

The second flight had the following goals:

- Test the starting procedure.
- Verify if the TVC can effectively vector the motor as intended.
- Evaluate the software sequence.

### Rocket

The flight did not use the calibrated PID values due to a human error. The PID tuning was incorrect because the values were not updated within the 3DOF block in Simulink, where an incorrect mass and inertia for the rocket were used.

- Mass: 0.6kg (real mass 0.473g)
- Inertia: 0.03 kg $m^2$ (real inertia 0.0172 kg $m^2$)
- Servo Reaction time 0.033333s
- Servo time delay: 0.007872s
- Avg Thrust: 3N
- TVC to COM distance: 0.241m

The calibrated values for the PID Controller were as follows:

- P: 1.05
- I: 1.5
- D: 0.285

### Flight

While I learned valuable lessons during the flight, excitement during the launch caused a lapse in concentration, leading to the failure of removing the "Remove before flight" pin. Consequently, the TVC remained blocked at an angle of 5 degrees for each axis, causing the rocket to flip immediately after liftoff.

Nonetheless, I achieved noteworthy progress since the last launch:

- All the planned improvements from the previous flight were implemented.
- Data was successfully logged onto SD Card.
- The entire starting procedure was tested, including camera setup and launch pad leveling.
- The durability and reusability of the TVC Mount was confirmed, demonstrating its capability to withstand motor thrust for subsequent flights. Only the motor tube needs to be replaced due to the heat changing the shape of the tube.
- Determined that the rocket remains reusable even after a hard impact with the ground.

*Figure 66: Second Launch Rocket during and after Lift-off*

### Improvements

During the launch, the following notes were made regarding improvements for upcoming flights:

- Redesign the cone to prevent the SD Card from breaking. Ensure that there is enough space and that the card does not come into direct contact with the cone.
- Develop a pre-flight checklist to ensure all necessary tasks are completed before launch.
- Include an additional hole in the upper tube, positioned underneath the part used for connecting two tubes. Currently, the hole is in the connecting part, which causes the cable to be cut when inserting the cone.

## Third Launch

Date: 1.5.2023

The third launch did not feature significant improvements and took place the following day, with the hope of successfully launching the rocket after removing the RBFP.

### Video

*Figure 67: Third Flight Video https://www.youtube.com/watch?v=ItLl7w50pAo*

### Goals

- Speed up the launch procedure.
- Verify the proper tuning of the PID controller.

### Rocket

The issue of the block not being updated inside Simulink persisted, as it was discovered after this flight. Unfortunately, the computed moment of inertia and the distance from the COM to the TVC were lost. However, it is possible to measure the values again. The rocket utilized all printed parts as can be seen in Figure 26, and was powered by 450mAh 2S LiPo Tattu battery.

- Mass: 0.6kg (real mass 0.473g)
- Inertia: 0.03 kg $m^2$ (real inertia 0.0172 kg $m^2$)
- Servo Reaction time 0.033333s
- Servo time delay: 0.007872s
- Avg Thrust: 3N
- TVC to COM distance: 0.241m

The calibrated values remained unchanged:

- P: 1.05
- I: 1.5
- D: 0.285

### Flight

The decision to proceed with the flight without significant improvements, solely to test the flight sequence, proved to be a valuable one. I obtained crucial feedback and identified several areas that require attention before the next flight.

- The starting procedure was executed correctly.
- Sufficient data was gathered to validate the maximal weight liftoff simulation, enabling us to make better weight adjustments for optimal performance.
- The PID values proved to be incorrect, as the rocket lost stability before reaching its apogee.
- From a software perspective, the flight sequence performed flawlessly.
- Unfortunately, the logging system failed once again, with the SD card breaking after landing.

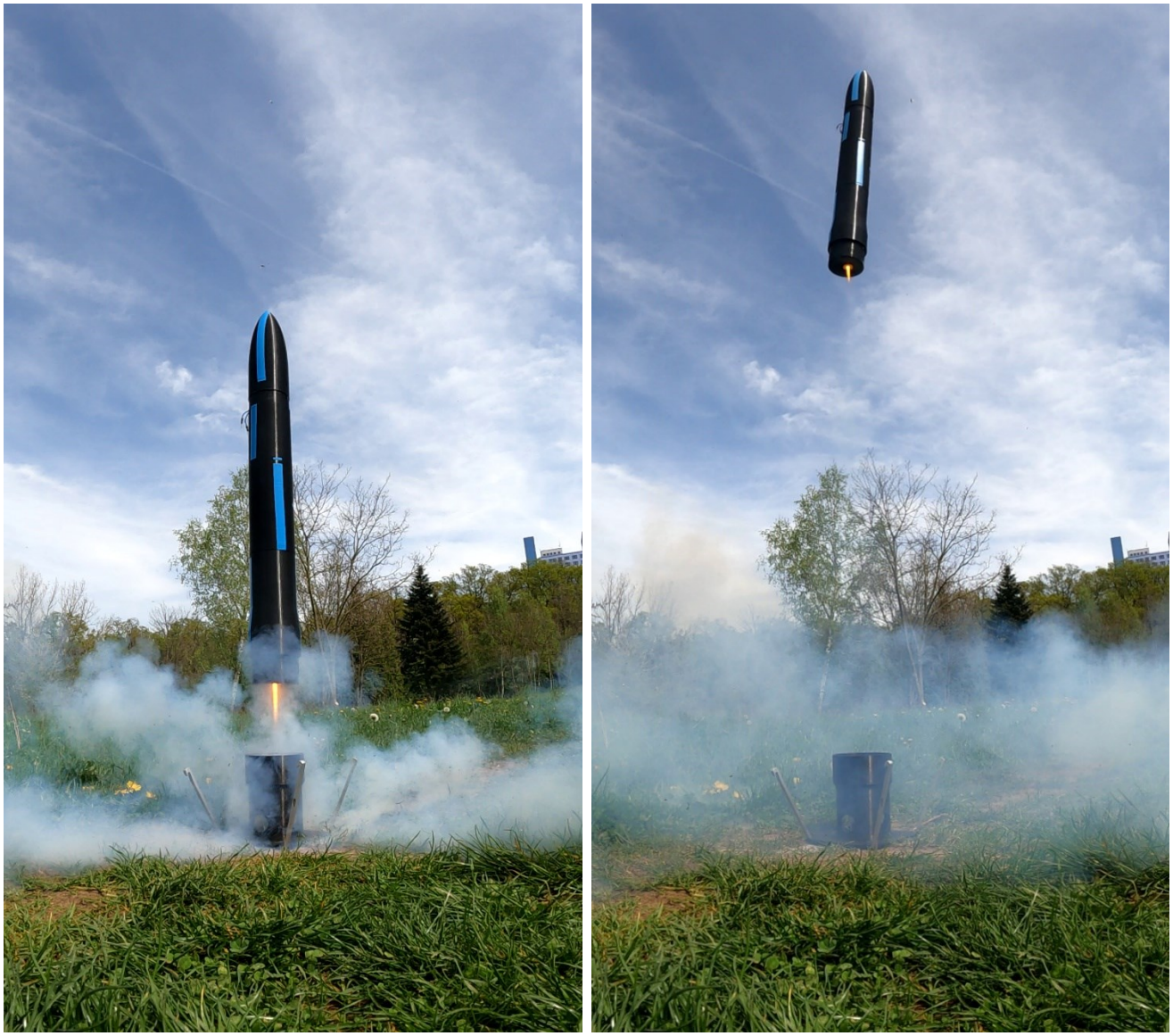

*Figure 68: Third Launch Rocket during and after Lift-off*

### Improvements

After this flight all the improvements from the previous flight remained unchanged, namely:

- Addressing the SD card issue by either printing a new cone or creating a hole in the current one to prevent damaging the SD card during the flight or a card insertion.
- Developing a pre-flight checklist to ensure all necessary tasks are completed before launch.
- Including an extra hole in the upper tube as the hole is in the connecting part right now, causing the cables to be cut after inserting the cone.

This flight has been the most valuable one so far, as it has allowed us to identify numerous areas that require improvement. The insights and feedback gathered during this flight are invaluable and will contribute to enhancing our future flights.

- Implement a 1-second TVC lock at the beginning of the launch to account for the peak thrust. The PID simulation is based on an average thrust of 3N, but at the start, there can be thrust up to 15N, potentially causing the rocket to overshoot.
- Also block the motors during liftoff to ensure that any contact with the launchpad does not introduce incorrect orientation values. Touching the launchpad can add extra G-forces to the IMU, leading to erroneous integration in the PID controller.
- Improve the precision of servo calibration, particularly for the central position of the servo, as setting up the incorrect upright position may play a significant role during unstable flight.
- Synchronize the maximum operating frequencies to 50Hz, considering the servos as the main limitation:
    - Teensy 4.1: 150Hz (with a basic loop with no heavy calculations)
    - BNO055: 100Hz (while in NDOF mode)
    - SG92R: 50Hz (operates with a 20ms cycle and 5-10% duty cycle)
- Determine a transfer function for servo movements, as their current movement is non-linear.

## Fourth Launch

Date: 13.5.2023

For this flight, I addressed the weight issue by removing the middle part of the rocket. However, this caused the cables to become too long. To solve this problem, I produced a solution using a string connected to the wires. The string lifted the cables after assembling the rocket and was securely attached to the rocket body using duct tape. With the structural changes, new measurements of the MOI had to be conducted. Additionally, I discovered that the Simulink 3-DOF block had not been updated. To further improve the rocket's performance, I implemented several modifications:

- Using a soldering machine, I created three new holes in the current rocket model: one for the SD card and two for cables (RBFP and Breakout).
- The software was updated to utilize the breakout for accurate liftoff detection and to ensure the TVC remained blocked for one second.
- The servo calibration process was enhanced, utilizing a paper tube to extend the rocket motor and a leveled calibration tool to determine the center position more accurately.
- A servo transfer function was developed to improve the precision of servo movements.
- The frequency and logging settings were adjusted to run at 50Hz.
- The weight of the rocket was successfully reduced to 405g, which aligns with our target weight for optimal performance.

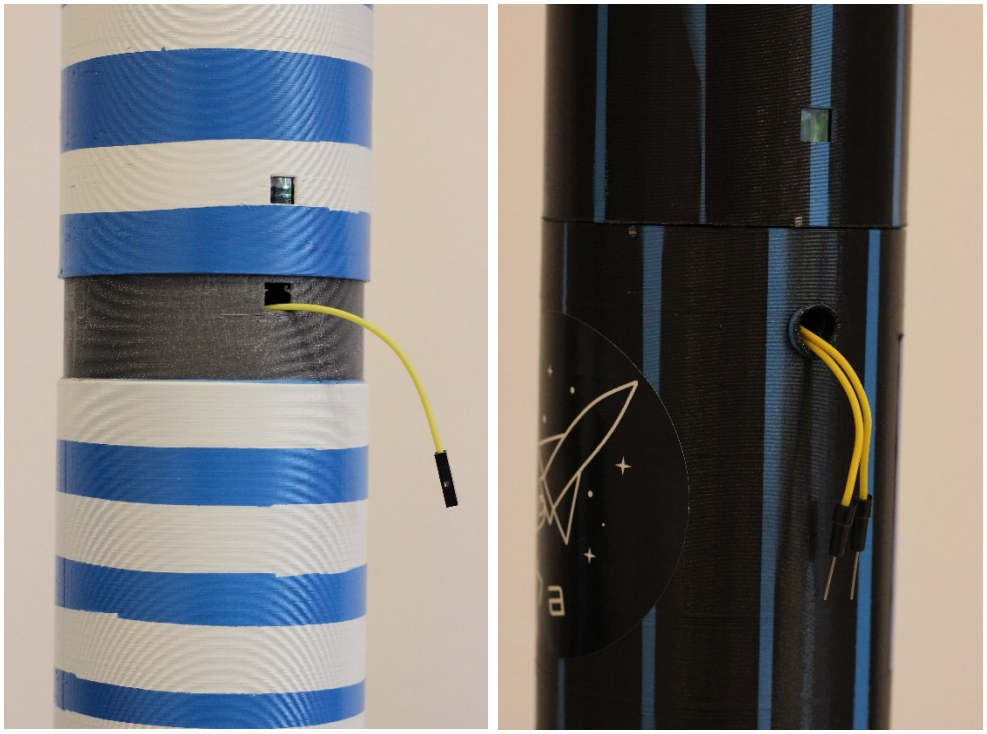

*Figure 69: Problem with the Cable (left), Soldered Hole (right)*

## Video

*Figure 70: Fourth Flight Video https://www.youtube.com/watch?v=GUQc64a9LDQ*

## Goals

- Test the new breakout system for liftoff detection.
- Achieve a longer duration of flight.
- Verify the PID tuning.

### Rocket

The issue with the simulations has been resolved, ensuring accurate results this time. The simulations were fine-tuned to reflect the real average thrust of 3.6N, as determined from our thrust curve.

Rocket specifications:

- Mass: 405g
- Oscillation: 1.246s
- COM to string distance: 0.155m
- String length: 0.54m
- Inertia: 0.006951 kg $m^2$
- Servo reaction time: 0.033333s
- Servo time delay: 0.007872s
- Average thrust: 3.6N
- TVC to COM distance: 0.155m

The PID controller has been tuned with the following values:

- P: 0.541875
- I: 1.265625
- D: 0.1122

### Flight

With each flight, our preparation process becomes more organized. While the flight successfully addressed previous issues above, there is still work to be done.

- I need to improve the pre-flight procedure to avoid damaging the SD Card. In this instance, the Card was accidentally left inside while assembling the Cone, resulting in its destruction.
- Better implementation of liftoff detection is crucial. The breakout wire caused higher friction than expected, leading to excessive changes in the rocket's direction and hindering stabilization after TVC mount activation.
- The transfer function for the Y axis requires correction as it was limited to approximately 4 degrees instead of the intended 5 degrees. To resolve this, I used a linear approximation like the first three flights.
- Usable data were logged, compared to the previous log where only the value of the unremoved RBFP was stored. However, one more flight is important to obtain better data and observe the PID Controller's response.

After a motor ignition, the rocket experienced an abrupt change in direction. During the initial 0.6 seconds, the motor TVC was locked, making the stabilization system inactive. Consequently, when the stabilization was eventually activated, it was too late to correct the rocket's sideways trajectory. This deviation is clearly visible in the picture on the right, where the rocket veers off to the side instead of ascending straight. The delayed activation of the stabilization system prevented correction of the rocket's orientation immediately after the launch.

*Figure 71: Fourth Flight Rocket during and after Lift-off*

### Improvements

To improve the next flight, I need to address the following:

- Improve the pre-flight check process.
- Implement BNO055 for liftoff detection instead of the breakout wire.
- Conduct more simulations to accurately retune the PID values.
- Fix the transfer function for the Y axis.
- Introduce a battery or reset switch to avoid unnecessary rocket disassembly.

## Fifth Launch

Date: 24.5.2023

This was expected to be the last test flight before submitting the thesis. For this flight, I aimed to address all the previously mentioned issues. I also considered dynamically changing the PID values based on the current thrust, which would be computed from the time since liftoff. However, for the sake of simplicity and incremental improvements, I decided to leave these adjustments for future iterations and instead

launch the rocket by locking the TVC for 0.6 seconds after liftoff. Since the last flight, I have taken the following actions:

- Double-checked our PID tuning using a Simulink model that I found online [41], ensuring the correctness of our model and tuned values.
- Fixed the transfer function for the Y axis.
- Resolved the liftoff detection issue.
- Implemented a software reset, eliminating the need to disassemble the rocket.

Using another simulation, I have confirmed that the rocket's response remained consistent. Furthermore, that model allows for more sophisticated simulations incorporating factors such as wind, air drag, uncertainty, and noisy sensor readings, which could be valuable for future tuning.

Throughout the tuning process, I discovered that it is unnecessary to use different PID values for each axis. I can treat these two problems separately. The code controlling the PID only outputs the motor angle, which is then transformed into servo movements using the transfer function above.

### Video


*Figure 72: Fifth Flight Video https://www.youtube.com/watch?v=vylj74kLUFc*

### Goals

The objectives for the upcoming flight are as follows:

- Log data showing the PID response.
- Activate the TVC Mount in a situation where it is still possible to stabilize the rocket.
- Remain airborne for more than 2 seconds.

### Rocket

Since the moment after the peak is crucial for us, the simulations were slightly adjusted to better accommodate the period when the rocket can tilt slightly due to the locked TVC and lack of fins, resulting in instability. Additionally, I inadvertently omitted the inclusion of the servo transfer function.

### Flight

The flight had a fixed date; unfortunately, a few issues arose before the launch:

- The transfer function for the y-axis was still incorrect, although I identified a bug in the measurement process.
- The calibration tube for measuring the default servo position was not as accurate as expected.
- The Y-axis servo holder was slightly moving, which may have contributed to the incorrect transfer function.

Ideally, it would have been better to postpone the launch to address these issues. However, due to time constraints, I decided to proceed with the rocket launch, albeit without achieving absolute perfection. The launch did not proceed as anticipated; nevertheless, I gained valuable insights from this flight. Our goal was to reach the PID control phase, which occurs 0.585 seconds after liftoff, enabling us to assess whether the PID values are properly tuned using the logged data. Unfortunately, the rocket lost stability prematurely. Locking the TVC for 0.585 seconds proved to be too long, resulting in the rocket losing stability.

*Figure 73: Fifth Flight Rocket during and after Lift-off*

### Improvements

The primary focus is ensuring the rocket's stability during the initial thrust peak. The following tasks will help us achieve this goal:

- Determine the optimal transition point between optimal functionality at 3.6N and 10N, utilizing PID tuning from the beginning and avoiding leaving the rocket uncontrolled after liftoff.
- Fix the transfer function for the Y-axis.
- Implement the use of a launching tube.
- Develop a mechanism to ensure perfect alignment of the stand.
- Print tool to facilitate easier calibration.
- Print a 3D extender to prevent the rocket motor from touching the ground, which could result in unintended servo movements.

The tool has been printed and can be viewed in the calibration section  Servo Calibration. The extender has also been developed in the meantime and is shown in the accompanying Figure 74. The extender ensures that the rocket motor remains elevated from the surface during the preflight check, which is performed every time the software is restarted. It rotates the motor, ensuring that all servos are functioning correctly. However, due to direct contact with the surface, it may cause slight servo movements, resulting in inaccuracies.

*Figure 74: Rocket Extender*

The most significant improvement since the last launch is undoubtedly the tube that will serve as an enhanced launching pad. Initially, I used a 100mm diameter stove pipe, which was not the ideal choice

given the rocket's outer diameter of only 77mm. To address this issue, I inserted a piece of carpet inside the tube to narrow it, transforming it into a perfect launch pad for future flights.


*Figure 75: New Launchpad (left) and the Rockets (the middle one was used for all launches)*

Additionally, I have re-tuned the values once again, aiming for a stable system within the thrust range of 3.2-10N. This enables us to activate the PID from the beginning of the flight, allowing successful logging and analysis of the PID response data. This is a significant milestone as it means the rocket is prepared for thrust vectoring right from the start of the flight. After conducting multiple simulations, accounting for factors such as wind and the conclusion of the TVC Mount, I have identified a substantial margin within which the rocket should remain stable. The new launch system addresses all the previously mentioned issues, resulting in a vectored flight with a stable simulation for the entire thrust curve. This contrasts with the third flight, where the instability during the initial phase was completely overlooked.

## Data Analysis

Throughout the flight, I recorded the following data:

- State
- Current Time
- X-axis acceleration and Y- and Z-axis rotation from the IMU (BNO055)
- PID gains for each axis.
- Motor Angle Output for each axis
- Servo output for each axis
- Previous error
- Time since liftoff
- Time difference between the loops
- Remove before flight pin value.
- Origin for the perfect BNO055 sensor alignment
- Pressure from the barometer (BMP388)

The barometer was primarily utilized for flights with more powerful motors, specifically in situations involving staging or deploying parachutes. However, for this project, the barometer data is not crucial.

On the other hand, data from the IMU is important, especially regarding the response of our PID controller. Unfortunately, I only managed to actively stabilize the rocket during the third flight when I implemented stabilization from liftoff. Unfortunately, the SD card broke upon impact, resulting in the loss of data from this flight. Consequently, I decided to lock the TVC to mitigate the initial peak since the PID gains were unstable during that phase.

However, subsequent flights demonstrated that this approach was not the correct direction. I was unable to maintain stability, and by the time the TVC attempted to correct it, the rocket had already deviated off course. Although the servo reactions were satisfactory, they did not provide sufficient data for analysis in this context.

For the sixth flight, I will revert to the previous approach. With fine-tuned PID values and an improved stand that should ensure more stable launches, I have high hopes that the next flight will provide valuable data.

# Chapter 9: Conclusion

## Future

This project is far from over. I aim to collaborate with CRS to further develop and enhance this project, making it more robust, reliable, and capable of achieving higher altitudes with larger rockets. Here is a list of important tasks that will propel this project to the next level:

- Add recovery system to ensure the safe return of the rocket.
- Upgrade to faster and more precise servos, such as the Blue Bird servo BMS-127WV (11g/0.05s/4.8kg). This will provide higher resolution (12 bits) for up to 4068 different servo positions and allow for operation at higher voltages (up to 8.4V) for increased response time and torque. However, careful attention must be given to prevent overheating or burning.
- Incorporate additional Klima D3-P motors to increase the total liftoff weight or redesign the TVC Mount to accommodate multiple motors for active stabilization. Exploring options like raising the motor height or placing the motors outside the rocket can be considered.
- Install onboard cameras, such as the RunCam Hybrid 4K camera, to capture footage of the rocket's flights.
- Implement a two-stage rocket system capable of controlled landing.
    - Develop software to compute the optimal ignition timing for the second stage.
    - Introduce movable fins on the top of the rocket to ensure stability during descent.
    - Create a mechanism that can adjust the total thrust, like modifying the position of the motor inside the motor tube to change the thrust curve or finding a way to block the motor.
    - Explore methods to reduce motor thrust by employing rotational movements to compensate for the angle of the motor.
- Apply a spray or coating to the flight computer to protect against the risk of short circuits caused by humid air. A silicon conductor could be used to combat humidity.
- Devise a closed-loop method for centering the TVC during the calibration phase.
- Develop a custom solid motor for the rocket.
- Build a secondary flight computer with emergency timers and automatic flight termination capabilities in case of unexpected situations.
- Integrate Bluetooth communication for servo calibration.
- Enhance the launchpad design with clippers that can control the ignition of multiple motors.
- Create a smartphone or computer application to facilitate communication with the rocket and enable data transmission, such as using OpenMCT Software.
- Optimize the 3D model to be printable without the need for additional support structures, reducing costs and improving print quality.
- Fine-tune the PID controller, considering the influence of the X-axis angle on the total thrust of the other axes. This may involve distributing the PID values for each axis based on motor inclination and time.

By addressing these tasks, I can propel this project forward, pushing the boundaries of rocketry and advancing our understanding of aerospace engineering. With each step, I strive to make significant improvements and contributions to the Czech Rocket Society.

## Results

Looking back at the Thesis Statement, I am pleased to confirm that I have successfully fulfilled all the requirements. Throughout my bachelor thesis, I undertook the design, construction, printing, and multiple launches of the rocket with active stabilization. Each flight provided valuable knowledge and logged data for future improvements. The simulations have proven effective, bringing the rocket closer to achieving straight flight.

In total, I conducted five flights, with each one contributing to our understanding and progress towards a fully controlled rocket capable of landing. The most successful flight was the third one, where I confirmed the functionality of the TVC Mount and thrust vectoring in general. This flight demonstrated the rocket's ability to change its flight direction. The rocket was stabilized since liftoff, but the PID was tuned for an average thrust, making the rocket unstable during the initial peak. As a result, I decided to lock the TVC to mitigate the initial peak caused by unstable PID gains during that phase.

However, subsequent flights revealed the limitations of this approach. Maintaining stability during the initial phase of the flight proved challenging, and by the time the TVC attempted to correct deviations, the rocket had already veered off course.

In preparation for the sixth flight, scheduled after the thesis submission, our plan is to vectorize the rocket from the beginning of the flight. This will provide us with comprehensive and valuable data for thorough analysis. After the fifth flight, I have already gained sufficient information about the rocket's behavior. I have made improvements to the launching pad structure and fine-tuned the PID values, ensuring the rocket will overcome the initial thrust peak in the sixth flight.

While not every decision made along the way was optimal, careful analysis has shown that the most important ones were indeed the right choices. The rocket has proven to be reusable, successfully completing at least five launches with the same rocket body and TVC Mount. Opting for a less powerful motor with a longer burn time was a wise decision, as the rocket reached its weight limits even without the recovery system. This also provided us with a longer airborne duration, allowing for more extensive testing and data gathering.

Additionally, I carefully selected materials such as ASA for the rocket body and PLA for the TVC, optimizing weight, impact resistance, durability, and precision. This meticulous material selection greatly enhanced the overall performance and reliability of the rocket.

The rocket design has proven reliable, and the servo has exhibited precision without significant issues throughout all five flights. The PID simulations conducted in Matlab were successful and validated by external sources. Additionally, all potential areas for improvement have been documented, enabling future project enthusiasts to learn from these mistakes.

Furthermore, the decision to forgo the reaction wheel has proven beneficial, as the rocket's flight did not generate a fast enough spin to adversely impact its trajectory. Keeping the project as simple as possible was a key objective, as complex solutions often introduce more intricate problems. Simplicity has played a crucial role in delivering the project on time.

Throughout the process, I successfully printed three fully functional rockets, which could serve as representations of our school. While not initially part of the project scope, this achievement brings me great satisfaction.


*Figure 76: Rockets for a Project Presentation*

## Conclusion

Over the past two semesters, I embarked on an ambitious rocket project that encompassed various aspects of aerospace engineering. Despite lacking prior experience, I took on the challenge of designing and developing a flight computer, modeling the rocket in Autodesk Inventor, mastering 3D printing, soldering,

and assembling PCB for the first time in my life. The culmination of these efforts resulted in the successful launch of an actively controlled rocket model.

In conclusion, the rocket project has been an exhilarating and demanding endeavor that allowed me to delve into the realm of aerospace engineering. Throughout the project, I encountered numerous obstacles and setbacks, but with persistent determination, I managed to overcome them.

I achieved a significant milestone by designing and constructing a rocket that showcased impressive flight capabilities. Through multiple test flights, I gained invaluable insights into stability and control systems. Each flight provided us with essential data that guided us in refining our design and making necessary adjustments.

The project nurtured a strong sense of creativity and problem-solving. I learned the importance of effective time management, adaptability, and attention to detail in every aspect of the project. Furthermore, it deepened my understanding of rocketry principles and expanded my practical skills in areas such as design, fabrication, and testing.

The knowledge and experience gained from this project will undoubtedly serve as a solid foundation for the future. The lessons learned and the successes achieved will inspire me to continue pushing boundaries and exploring new frontiers in rocket science.

Overall, the rocket project has been an incredible journey characterized by personal growth, continuous learning, and innovative thinking. It has instilled in me a profound passion for aerospace engineering and a steadfast commitment to further contribute to the field. I take pride in our accomplishments and eagerly anticipate the future advancements that this project will inspire.

*Figure 77: Assembled Rocket Model II before the First Lift-off*

# References


[1] T. S. Taylor, Introduction to Rocket Science and Engineering. CRC Press, Taylor & Francis Group, 2017, ISBN: 9781498772327.

[2] NASA Glenn Research Center, Rocket Engine, [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/rktengine.html.

[3] NASA Glenn Research Center, Rocket Launch, [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/lrockth.html.

[4] HyEnD, Hybrid Rocket Engines, [Online]. Available: https://hyend.de/index.php/hybrid-rocket-engines/.

[5] J. Nakka, Solid Rocket Motor Grain Geometry, [Online]. Available: https://www.nakka-rocketry.net/th_grain.html.

[6] Estes Rockets, Get Started, [Online]. Available: https://estesrockets.com/get-started/.

[7] National Association of Rocketry, Standards and Testing Committee, [Online]. Available: https://www.nar.org/standards-and-testing-committee/.

[8] R. Casimir, kord-rcs, GitHub, [Online]. Available: https://github.com/raphaelcasimir/kord-rcs..

[9] Raphaël Casimir, RCS Thruster Testing with the Space Simulator, 2019. [Online]. Available: https://www.youtube.com/watch?v=mpiw7nPdZas.

[10] Astramodel.cz. (n.d.). Přehled raketových motorů, [Online]. Available: https://www.astramodel.cz/cz/blog/prehled-raketovych-motoru.html.

[11] Modelrockets.co.uk, B2 Six Pack 18mm Rocket Motor , [Online]. Available: https://www.modelrockets.co.uk/shop/model-rocket-motors/b2-six-pack-18mm-rocket-motor-p-3703.html?osCsid=eou8eb8nt5tq88jppq9a35p203.

[12] NASA Glenn Research Center, Rocket Thrust Calculator, [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/rktcp.html.

[13] NASA Glenn Research Center, Gimbaled Thrust , [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/gimbaled.html.

[14] NASA Glenn Research Center, Rocket Thrust Chamber Geometry Calculator, [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/rktcg.html.

[15] NASA Glenn Research Center, Rocket Stability Calculator, [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/rktstab.html.

[16] Dr. Wernher von Braun explains: Why Rockets Have Fins, Popular Science, vol. 185, no. 3, pp. 68-69/184-185, 1964, ISSN: 0161-7370.

[17] R. Bláha, Stabilization and control of a vertically landing launcher, [Online]. Available: https://dspace.cvut.cz/handle/10467/87890?show=full.

[18] NASA Technical Reports, Ejectable Rocket Motor Ignition and Launch, [Online]. Available: https://ntrs.nasa.gov/citations/19700023342.

[19] Wikimedia Commons. (n.d.). Rocket Controls [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/0/0f/Rocket_Controls.gif.

[20] Teslarati, SpaceX's Starship Super Heavy Grid Fins Could Transition from Titanium to Steel, [Online]. Available: https://www.teslarati.com/spacex-starship-super-heavy-grid-fins-titanium-to-steel/.

[21] Headed for Space, Reaction Control System (RCS), [Online]. Available: https://headedforspace.com/reaction-control-system/.

[22] Charleslabs, Reaction Wheel Attitude Control, [Online]. Available: https://charleslabs.fr/fr/project-Reaction+Wheel+Attitude+Control.

[23] Cassiopeia Space Program, The Development of Rockets with Flight Correction, Unpublished scientific paper for the 41st Beijing Youth Science Creation Competition. [Unpublished].

[24] FZ Model Rocketry, TVC Dev Blog 1: Dec 28th, [Online]. Available: https://fzmodelrocketry.ca/2021/12/30/tvc-dev-blog-1-dec-28th/.

[25] BPS.space, Thrust Vector Control, [Online]. Available: https://bps.space/products/thrust-vector-control.

[26] Cults, Motech Space Thrust Vector Control Mount, [Online]. Available: https://cults3d.com/en/3d-model/various/motech-space-thrust-vector-control-mount.

[27] Thingiverse, Thrust Vector Control Nozzle, [Online]. Available: https://www.thingiverse.com/thing:4609845.

[28] OpenRocket, Fully Featured Model Rocket Simulator Program, [Online]. Available: https://openrocket.info/.

[29] Filament-pm.cz, [Online]. Available: www.filament-pm.cz.

[30] CNCKitchen, S. Karmali, Comparing PLA, PETG, & ASA feat. Prusament, [Online]. Available: https://www.cnckitchen.com/blog/comparing-pla-petg-amp-asa-feat-prusament.

[31] Filament-pm.cz, PLA černá 1.75 mm 5 kg, [Online]. Available: https://www.filament-pm.cz/pla-cerna-1-75-mm-5-kg/p304.

[32] Filament-pm.cz, PETG černá 1.75 mm 5 kg, [Online]. Available: https://www.filament-pm.cz/petg-cerna-1-75-mm-5-kg/p305.

[33] Filament-pm.cz, ABS černá 1.75 mm 1 kg, [Online]. Available: https://www.filament-pm.cz/abs-cerna-1-75-mm-1-kg/p16.

[34] Filament-pm.cz, ASA černá 1.75 mm 2 kg, [Online]. Available: https://www.filament-pm.cz/asa-cerna-1-75-mm-2-kg/p285.

[35] BPS.space, Rocket Tuning - Build Signal R2, 2018. [Online]. Available: https://www.youtube.com/watch?v=w8pu5PhD2B8.

[36] Czech Rocket Sociery, Documentation for Cansat Avionics 2022, [Internal Technical Documentation].

[37] Lukáš Málek, Real-Time Operating System Report for Microcontroller Development for Advanced IoT Using Embedded C, Danish Technical University. [Unpublished].

[38] Digi-Key, What is a Real-Time Operating System (RTOS), [Online]. Available: https://www.digikey.com/en/maker/projects/what-is-a-realtime-operating-system-rtos/28d8087f53844decafa5000d89608016.

[39] Eclipse IoT, IoT Developer Survey 2018, [Online]. Available: https://iot.eclipse.org/community/resources/iot-surveys/assets/iot-developer-survey-2018.pdf.

[40] Hanbergs Space Mission, PID Controller design, [Online]. Available: https://www.patreon.com/user?u=48967593. [Under Paywall].

[41] CodeSandBox, Rocket Sim Final with Position Control (Dual Motor), [Online]. Available: https://codesandbox.io/s/rocket-sim-final-with-position-control-dual-motor-0oq6u.