CHALMERS

Starship Model

Modelling and control of an electric reusable rocket

Bachelor thesis in systems and control

Gunnar Edman, Oskar Haapalo, Gustav Haller, Nicholas Holmén, Emil Reinfeldt, Edvin Öhman

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg 2024
www.chalmers.se

Bachelor Thesis 2024

Starship Model

Modelling and control of an electric reusable rocket

Gunnar Edman, Oskar Haapalo, Gustav Haller, Nicholas Holmén,
Emil Reinfeldt, Edvin Öhman

CHALMERS

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg 2024

Starship Model
Modelling and control of an electric reusable rocket
Gunnar Edman
Oskar Haapalo
Gustav Haller
Nicholas Holmén
Emil Reinfeldt
Edvin Öhman

Supervisor: Dr. Hasith Karunasekera, Chalmers Tekniska Högskola
Examiner: Prof. Jonas Sjöberg, Chalmers Tekniska Högskola

Bachelor Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover picture: The constructed, electronically driven Starship Model.

Written in LaTeX
Göteborg 2024

Abstract

This project describes how a model of an electrical reusable autonomous rocket can be designed and controlled. It is a continuation on an existing model from a previous project where a rocket model was built from scratch. Flaws of the existing model are identified and improved by changing the rocket design, electronic components and the automatic control of the rocket. The most significant change to the rocket with regards to design is the new ability to gimbal the rockets motors in order to create stability and balance. A faster microcontroller is introduced, along with new sensors including a LiDAR and an IMU. The sensors are used together with sensor fusion methods such as a Madgwick filter for state estimation. The PID control used on the previous model is tested on the new model and compared to a new LQR controller introduced to improve the rockets flying capability. The final product was able to lift off, fly to a pre-defined altitude and hover for a few seconds.

The project helps to understand what physical components are needed together with what software to give the rocket autonomous flight abilities. Further development of the rocket is needed to make it fully autonomous and to give it the ability to fly to set points and land itself.

Preface

This report presents the findings and outcomes of bachelor thesis Starship Model. The thesis was conducted at the department of Electrical Engineering at Chalmers University of Technology in the spring of 2024. This project was a further development of a previous bachelor thesis [1] which took place in the spring of 2023.

Throughout this report, we delve into the design process of an autonomous electric rocket model, both structurally and electronically. Furthermore designing and optimising a controller for autonomous flight.

We extend our gratitude to our supervisor Dr. Hasith Karunasekera which has been of great help in completing this project. We also would like to thank Dr. Meng Yuang which has been of invaluable help when designing the control system. Furthermore, we would like to thank Alfred Aronsson for his guidance in modelling the control system. Additionally, we are gracious for the support Gideon Geelnard has given by presenting the previous thesis in depth. We also thank the CASE-lab at Chalmers for providing a place to work, necessary tools and materials for the project. Lastly we want to thank our examiner Prof. Jonas Sjöberg for contribution.

Gunnar Edman, Oskar Haapalo, Gustav Haller, Nicholas Holmén, Emil Reinfeldt and Edvin Öhman;
Gothenburg, May 2024

Nomenclature

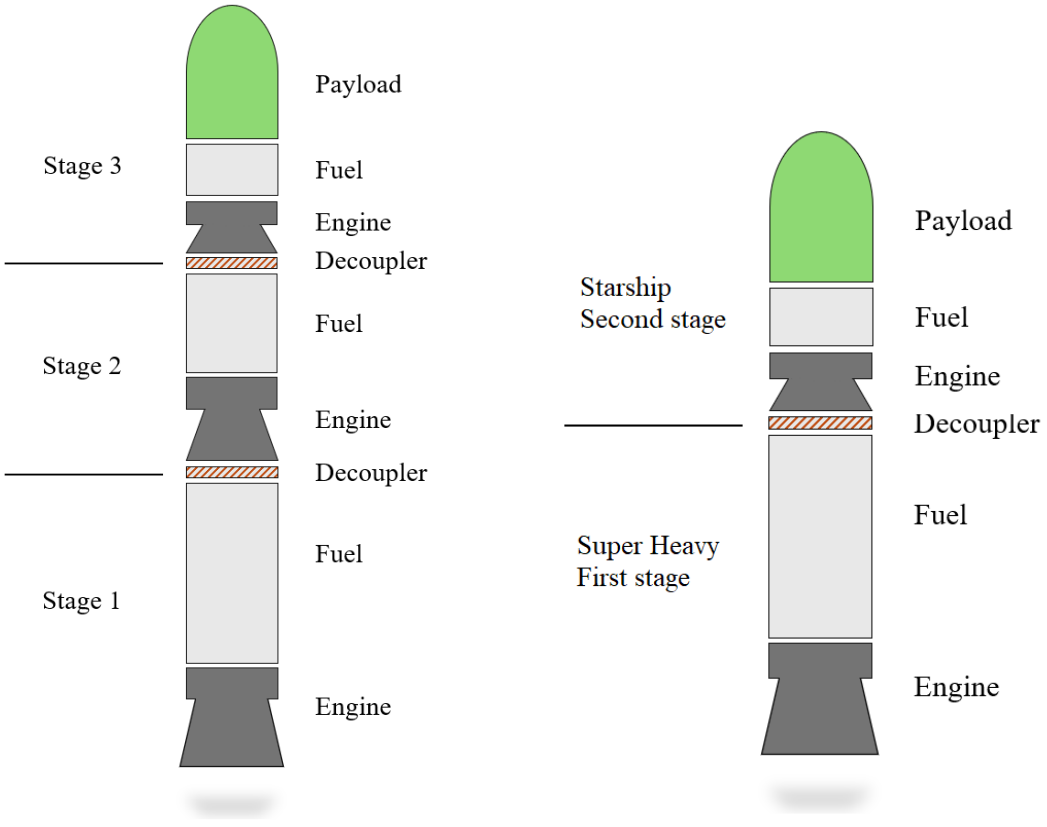
γ_1	Pitch angle (rotation around y)
γ_2	Yaw angle (rotation around x)
$\overline{I}_x$	Moment of inertia around the x-axis at the $c.o.m$
$\overline{I}_y$	Moment of inertia around the y-axis at the $c.o.m$
$\overline{I}_z$	Moment of inertia around the z-axis at the $c.o.m$
Φ	Roll angle (rotation around z)
$c.o.m$	Center of mass (x,y,z) from the center at the bottom of the model.
c	Speed of light
d	Diameter of the model
g	Gravity of earth
h_1	Distance between motor 1 and the center of mass
h_2	Distance between motor 2 and the center of mass
L	Length of the model
m	Mass of the rocket

Contents

List of Figures

1
Introduction

Rockets have been a disposable product since the first successful launch, nearly 100 years ago (1926)[2]. Made to carry crew and cargo into space, rockets consume an immense amount of liquid fuel that needs to be stored onboard. Conventionally, the fuel is divided and stored in several different stages as seen in Fig. 1.1a. Once a stage is out of propellant, it detaches itself from the payload to save upon the mass needed to be carried by the rest of the rocket. The stages generally burn up in the atmosphere on their way back to Earth or fall down in the sea, consequently rendering it useless for the next space mission.



(a) A conventional rockets different stages

(b) Starship's different stages

Figure 1.1: Different stages in conventional rocket's and the Starship.

In recent years, the company SpaceX[3] has gained a huge influence in the rocket industry with their semi-reusable rockets, saving time, money, and resources. How-

ever, SpaceX strives to further revolutionize the space industry by manufacturing a fully reusable rocket known as Starship. Contrary to conventional rockets with several stages, Starship will only consist of two as illustraded in Fig. 1.1b. The first stage "Super Heavy" that takes the rocket out of Earth's atmosphere, and the second stage "Starship" carrying the payload further into space and back to Earth. These rocket stages will be, and are already in some ways, reusable using a proprietary way of landing them[4].

1.1 Background

The following report extends from prior research [1] involving the construction of an electrically propelled model of the Starship-stage[5], denoted as the Starship model. The initial aim of model in [1] was to autonomously launch and land. However, the authors in [1] identified several critical weaknesses, elaborated upon in chapter 2.1.1 and 2.2.1, these serve as the foundation for the current study to improve the Starship model.

Constructing and researching a reusable rocket similar to the Starship-stage can provide valuable understandings and insights in the technology required to autonomously land rockets. Furthermore strengthening the vision of a more economically and environmentally sustainable future within the space travel industry.

1.2 Contribution

The main contribution of this study is to autonomously launch and hover the Starship model and thus improving on weaknesses from [1]. To achieve this, the following contributions are made:

- Designing and implementing a gimbal construction to enhance the models ability to redirect airflow and thus increasing controllability.
- Reducing weight of structural components to counteract the added weight from new parts.
- Implementation of lidar sensor to get a more accurate measurement of altitude.
- New Inertial measurement unit (IMU) and implementation of a improved Madgwick filter to improve attitude data (orientation of the rocket[6]).
- Implementation of a Linear Quadratic Regulator (LQR).
- Introducing the ability of inflight data logging for debugging and system identification.
- Providing all CAD-models, C++ code and SIMULINK simulations as open source material at the project GitHub page[1].

[1] `https://github.com/EENX16-24-50-Starship/starship-model-2024`

Collectively, these contributions signify a noteworthy progression from the groundwork laid by [1]. In addition, the group has gained valuable understandings and insights into the technology required to autonomously land rockets.

1.3 Societal and ethical concerns

Modeling and researching fully reusable rockets promotes a more sustainable future for the rocket industry by lowering material and economical resources required for space exploration. Consequently enabling space access for a wider variety of countries and companies. This in turn promotes more diverse science to be conducted, and thus accelerating research in more areas.

Using electrically controlled propellers for modelling instead of chemical propellant helps minimize CO_2 emissions, aligning with broader environmental considerations. Additionally, using electrical motors, testing becomes safer by eliminating concerns of chemical and fire hazards. However, using drone propellers at high RPM poses a risk of bodily harm if the motors where to start unexpectedly. Nevertheless, flights are conducted in a test rig where it won't endanger the surrounding environment during due to manual constraints.

1.4 Report Organization

In the following chapter, design of the new model is outlined. Highlighting modifications performed upon the initial model from [1]. The design modifications are split into two parts; structural design and electrical design where the improvements from each part is brought forth in the respective sections.

Following the rocket design chapter, the rocket control chapter explains the rockets dynamics and its controllers. This chapter also explains how the controller effect the real life rocket, this is done through a basic simulation model to give an overview of the dynamics of the complete system.

In the next chapter, tests conducted on the model are presented and their outcome discussed, additionally a conclusion of the results are given.

In the final chapter, a conclusion of the study is presented. Additionally, further work suggestions are given to facilitate an improvement process for this project.

2
Rocket design

This chapter of the report covers the new design of the model motivated by weaknesses of the initial model. Additionally the choice of new electrical components is explained and design of the new electrical circuit is presented.

2.1 Structural design

This section details changes made to the structural parts of the existing Starship model, mainly focusing on the gimbal construction and design of new parts.

2.1.1 Initial structural design

The previous design of the rocket model was made to replicate SpaceX's Starship, therefore the outer design of the model mimicked the same height to diameter ratio as Starship, which is 13.5:1. The model used two electrical drone motors to generate the thrust and rudders attached to a rudderplate at the bottom, which was used to redirect air thus making it possible to steer and angle itself. An illustration of the model with each part indicated can be seen in Fig. 2.1

Part	Part name	Quantity	Material
1	Rod	8	Carbonfiber
2	Rudderplate	1	CPE
3	Motorplate	2	CPE
4	IMU-plate	1	CPE

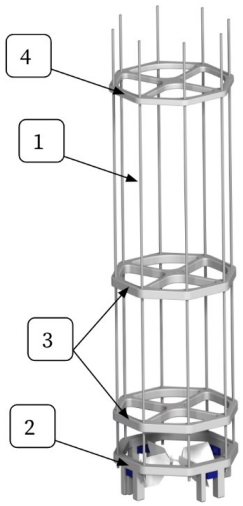


Figure 2.1: Previous design [1], modified with permission.

Three main problems with the previous design was identified:

1. **Difficulty in determining exact thrust**
   Controlling issues with the previous model stemmed from turbulence generated at its base, causing unpredictable forces that compromised the steering system's effectiveness. The model was steered by vectorized air accelerated by the drone motors using a system of rudders and servos. At each ruddeplate, see Fig. 2.1, two sets of rudders is attached, each set containing two rudders connected via an axis. These sets were placed perpendicular to each other at the bottom of the model. To control the angle of the rudders along with the desired thrust vector servomotors were utilized to rotate one set of rudders independently of the other.

2. **Model torsion and its effect on sensor data**
   Using only four plates as displayed in Fig. 2.1 made the distance between them are large enough to subject the model to torsion. Torsion will effect the sensor data by causing resonance in the structure. Counteracting this phenomena is vital since the motors exert torque on the structure to create the thrust needed for a flight. On the other hand, the layout effectively provided space for mounting components such as sensors and micro controllers.

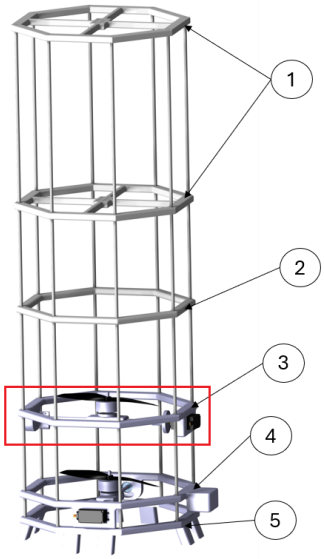
3. **Unnecessary loading**
   Lastly, every plate utilized in [1] was overdimensioned in thickness leading to unnecessary weight being added to the model.

2.1.2 Gimbal construction design

Using rudders is not how Spacex's Starship utilizes its motors to achieve stability and controllability of itself. Starship gimbals its motors, much like a ball joint, to a desired angle which in turn projects all the thrust in one direction at a time. Therefore a central part of working with the Starship model was to redesign the motor system to match that of the real Starship. In addition, by researching similar previous projects [7], the gimbal solution is the preferred way of vectorizing thrust on rockets. Thus, implementing a gimbal construction would address the first weakness of the previous model.

To simplify the construction, saving money and time but still achieve 360° of gimbal rotation, the motors are placed perpendicular to each other with the ability to rotate around its own horizontal axis. The gimbal construction consist of 9 different parts which can be seen in Fig. 2.2.

Part	Part name	Quantity	Material
1	Component plate	2	PLA
2	Stability plate	1	PLA
3	Gimbal construction	1	PLA
4	Gimbal construction with lidar housing	1	PLA
5	Bottom plate	1	PLA



Part	Part name	Quantity	Material
1	Gimbal plate	1	PLA
2	Servo motor	1	
3	Servo rod connector	1	CPE+
4	Engine pad	1	CPE+
5	Bearing	1	
6	D-rod Bearing to Engine pad	1	Silver steel
7	D-rod Servo to Engine pad		Silver steel
8	Engine	1	
9	Propeller	1	Glass fiber nylon

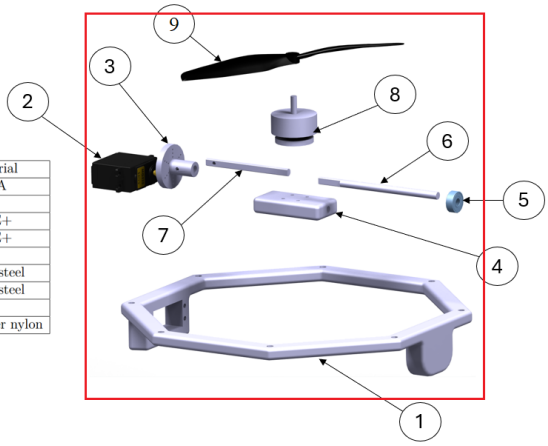


Figure 2.2: Parts of the Starship model and gimbal construction.

Design parameters of the Starship model's structure and parts include:

- The gimbal plate is 3D printed using polylactic acid (PLA) due to its lightweight properties.
- Parts 3 and 4 in Fig. 2.2 are 3D printed using co-polyester (CPE+). CPE+ was chosen due to its toughness, making it suitable for components subjected to loads from the motor.
- To secure the motor in place during servo rotation two D-shaped rods are attached at each side of the motor pad as seen in Fig. 2.2. These rods are crafted using silver steel which is a commonly used unhardened tool steel with tight tolerances that lends itself to machining high precision parts. One rod connects the servo to the motor, while the other links the motor to a bearing, enabling it to rotate freely when the servo operates.
- The gimbal construction is positioned as low as possible on the model to increase the lever arm to the center of mass, consequently reducing the required output from the servos and motors to adjust the model, thereby enhancing control.

- The propellers are counter-rotating to minimizing the impact of roll (rotation around the rockets own vertical axis) as shown in 3.3.
- The radius for this construction is dimensioned in order to fit the propellers when tilted without clash. Detailed measures can be seen in appendix A.
- The motors are placed at a distance of 125 mm, motivated by tests described in 4.2.

2.1.3 Plates design

Apart from the plate housing the gimbal construction, the remaining plates were redesigned with the same diameter and octagonal shape. Two component plates, one bottom plate and a stability plate was 3D printed using PLA. The redesign of all plates primarily addressed concerns about unnecessary load and accommodated space for new electrical components. Additionally, with reduced weight, an extra stability plate could be introduced, effectively mitigating torsion issues and their impact on sensor data.

Detailed drawings with measurements for each part presented in section 2.1.2 and 2.1.3 can be found in appendix A.

2.1.4 Parameters of the final model

The parameters of the final model are presented in Table 2.1.

Table 2.1: Parameters of final model

Parameter	Value
m (mass)	2,5 [kg]
$\overline{I}_x$ (moment of inertia)	0,215 $[kgm^2]$
$\overline{I}_y$ (moment of inertia)	0,226 $[kgm^2]$
$\overline{I}_z$ (moment of inertia)	0,042 $[kgm^2]$
d (distance)	275 [mm]
L (distance)	1000 [mm]
h_1 (distance)	456 [mm]
h_2 (distance)	331 [mm]
$c.o.m$ (point)	(66 [mm]; 38[mm]; 534 [mm])

The values in Table 2.1 are calculated through the CAD program Catia V5, where the exact weights and placements are corresponding with the real life model. All CAD models can be seen at the project GitHub page. Note the nomenclature for detailed descriptions of parameters.

The constructed Starship Model can be seen in Fig. 2.3.

Figure 2.3: Constructed Starship Model.

2.2 Electronic design

This section covers modifications of electronics, sensors and actuators of the model, mainly focusing on the components that were utilized. The complete component list can be found in appendix B.

2.2.1 Initial electronic design

The electronic design outlined in [1] utilized two Arduino UNOs [8], a barometer, an IMU and two BLDC motors. This electronic design resulted in the following four weaknesses:

1. **Barometer sensitivity towards small changes**
   A barometer was used to measure the altitude but was sensitive to changes in air pressure. Minor fluctuations in the surrounding pressure substantially impacted measurement readings, leading to variability in altitude estimates.

2. **Poor IMU performance**
   The inertial measurement unit (IMU) utilized in [1] was of poor quality. When testing the IMU with an attitude estimator, the estimates drifted noticeably over time. This was the result of two problems. Firstly, the noise spectral density of the sensor readings was high compared to other similar IMUs, thus increasing the integrated error. Furthermore, the IMU was sensitive to vibrations, which poses a problem given that the motors on the rocket spin at 18 000 RPM. This sensitivity introduced even more noise which resulted in higher estimate drift.

3. **Unnecessary complexity with using two Arduino UNOs**
   The Arduino UNO is limited in processing power at high frequencies and struggles to manage multiple inputs and outputs. Its low frequency results in slow calculation times and reduces the efficiency of the system. Moreover, a complex dynamic model such as the Starship model necessitates multiple inputs and outputs rendering the Arduino UNO unsuitable.

4. **Telemetry**
   The lack of telemetry made it challenging to fine-tune the control system since no data could be retrieved, complicating post-flight data analysis. Additionally, without flight data, performance of sensors during flight couldn't be analyzed.

The following components was considered good enough and was therefore reused for this project:

- Brushless DC motors - had sufficient thrust to lift the rocket
- Electric Speed Controllers - compatable with servos and microcontroller
- Batteries to motors - compatible with the motors and servos

2.2.2 Sensors

When controlling complex dynamic systems like this rocket model, reliable sensor readings are crucial for enabling feedback control. To ensure the availability of the required data, the following sensors were utilized:

- **6-axis IMU consisting of:**
  - 3-axis MEMS (micro-electromechanical systems) accelerometer
  - 3-axis MEMS gyroscope

  To address performance issues with the IMU used on the previous model as described in section 2.2.1, a new IMU with higher performance was selected. It has mechanical LP-filtering making it more resilient to vibrations and thus a better fit for the Starship model. It also has a lower noise spectral density in the measurement data resulting in reduced drift in the state estimations as covered in section 3.4.

- **LiDAR distance sensor** The problem of measuring the altitude, mentioned in 2.2.1, was solved by acquiring a LiDAR (light detection and ranging) distance sensor. It achieves much higher precision in the readings compared to the barometer sensor used in the previous project. This is possible since it provides an absolute measurement, not based on pressure differences in the ambient air. This makes it much more resilient to environmental disturbances such as change of pressure due to wind or temperature fluctuations etc. It is mounted on the side of the rocket 10 cm above ground pointed straight downwards as can be seen in Fig. 2.2. As a consequence of the LiDAR being fixed on the side of the model, readings will be effected by pitch (γ_1) yaw (γ_2) as illustrated in Fig. 2.4.

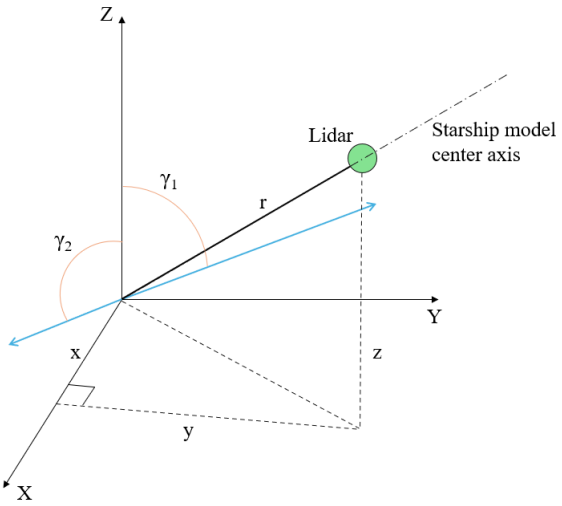


Figure 2.4: LiDAR reading's relationship with pitch and yaw.

To account for the aforementioned relation and still calculate the altitude z at all altitudes the following trigonometric relationships are used:

$$r^2 = x^2 + y^2 + z^2 \tag{2.1}$$

Where the distance x is calculated as:

$$x = r * sin(\gamma_2) \tag{2.2}$$

And the distance y is calculated as:

$$y = r * sin(\gamma_1) \tag{2.3}$$

Finlay the altitude z can be derived as:

$$z = \sqrt{r^2|(1 - sin(\gamma_1)^2 - sin(\gamma_2)^2)|} \tag{2.4}$$

- **Optical flow sensor**
  To address the lack of lateral position awareness when controlling the rocket, more sensors in addition to the IMU and LiDAR was necessary. To solve this, an optical flow sensor was acquired. It works by detecting motion of surfaces by using a small camera. The camera in turn provides a steady stream of picture samples to an algorithm that analyzes the motion of objects between the frames and calculates the optical flow field [9]. All of this is done on one circuit board, including the camera and a small microcontroller unit (MCU). The resulting output from the sensor is a Δx and Δy in rad/s. These can be used for better lateral awareness. To make the optical flow sensor fully compatible with the rocket, further development is necessary as discussed in chapter 5.

2.2.3 Microcontroller and communication protocols

The MCU Teensy 4.1 [10] with its ARM Cortex-M7 processor was chosen for the Starship model. The MCU handles all the real-time processing required to control the rocket, considering the sensor measurements. The main reason for choosing the Teensy 4.1 was:

- With the clock speed of 600MHz the Teensy 4.1 will have sufficient computational power to implement the control system described in chapter 3.2.

- The Teensy 4.1 offers multiple inputs/outputs (I/Os), as depicted in appendix C, facilitating the connection of the entire system to various protocols. Consequently, eliminating the need of using two Arduino UNOs.

- The inclusion of a micro-SD card slot in the Teensy 4.1 was critical in selecting the MCU, as it enables data storage on an external drive, thereby enabling telemetry.

To enable for communication between the MCU and sensors presented in section 2.2.2 an I2C protocol [11] was introduced. To achieve communication through I2C at our required speed of 400kbps without delays and corrupted data, a rise time for

a binary bit can be at most 400ns. This was accomplished by using pull-up resistors connected between the I2C bus and the 3.3 V supply of the MCU, ultimately lowering the resistance.

As per the manufacturers recommendation, the optical flow sensor where connected via a Serial Peripheral Interface (SPI) protocol [12] which allows for higher speeds than the I2C protocol.

2.2.4 Actuators

In order to propel the system described in 2.1.2, two, gimbaled, brushless DC motors with propellers are used. The motors are controlled by two electronic speed controllers (ESCs), one for each motor.

The gimbal mechanism, shown in Fig. 2.2, is actuated by two servos, one for each motor. When selecting servos, these two parameters were primarily considered:

- To enable fast and accurate control of the gimbal shaft position a servo with a high frequency control loop is necessary. Hence, the servo of choice was a digital servo running its internal control loop at 300 Hz instead of the 50 Hz of a typical hobby servo.

- A servo with sufficient torque had to be selected to handle the moment of inertia in the gimbal assembly. Also, it had to be be able to maintain its shaft position while the propellers are spinning at 18 000 RPM and possibly generating disturbance torque around the gimbal axis. Approximate calculations showed that a stall torque of 0.8 Nm was enough to achieve this.

Both ESCs and servos are controlled using 50 Hz Pulse Width Modulated (PWM) signals. For the ESCs, 1100 μs on-time corresponds to 0 % thrust and 1940 μs corresponds to 100 %. As for the servos, 900 μs corresponds to a shaft position of 0 degrees and 2100 μs corresponds to 120 degrees. The logic level of the MCU is 3.3 V, while it is 5 V on both the servos and the ESCs. To make the ESCs and servos work with the 3.3 V output of the MCU, logic level-sifters where utilized to increase the amplitude of the PWM signals to match the 5 V logic levels.

2.2.5 Power supply

When arranging the power supply to all different components, the following considerations were made:

- The ESCs receive a voltage of 25.2 V from four 3-cell LiPo batteries, connected according to the schematics in Fig. 2.5.

- To ensure continuous operation of the low voltage circuits, including the MCU and senors, a small 2-cell LiPo battery was used. This battery ensures that the control loop runs continuously, even in the event of sudden voltage drops

in the big batteries caused by peak current draw from the motors. This continuous operations is particularly important in the attitude estimation, as the Madgwick filter needs time to converge to a reliable estimate during startup.

- The ESCs featured battery eliminator circuits (BECs) which regulates the terminal voltage to 5.2, 6 or 7.4 V. This proved beneficial in this project, since the servos have a maximum rating of 6 V, thus matching one of the BEC modes. Additionally, this eliminated the need for dedicated batteries to power the servos, thus reducing the weight of the rocket.

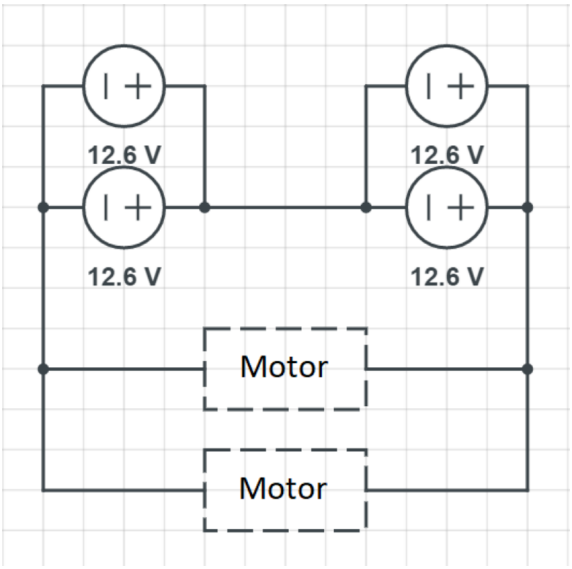


Figure 2.5: Arrangement of the batteries powering the propeller motors [1], reprinted with permission.

2.2.6 Circuit design

The circuit was designed on a protboard achieving the following strengths:

- No loose cables while in flight mode.
- Easy interface for connections.
- Light weight.

In Fig. 2.6 the circuit can be seen, with its main components indicated. All components used can be found in appendix B.

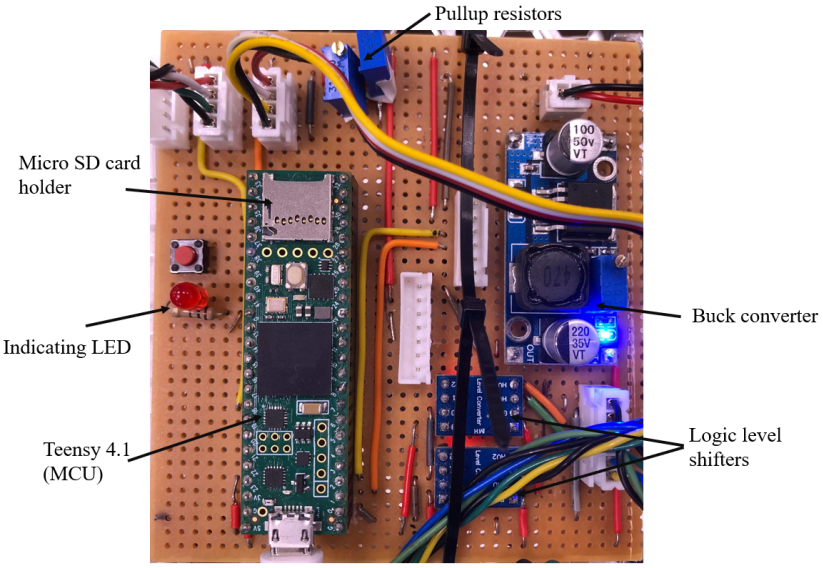


Figure 2.6: Proto board circuit.

2.2.7 Ground control

In efforts to enable remote control of the rocket, a ground controller was constructed. Its two primary objective was to:

1. **Send and receive data during live flights**
   Sending and receiving data during flights enables live analysis of results that can be used to improve the system.

2. **Manually override the control system in case of failure**
   The rocket model incorporates delicate components, meaning that a potential crash could result in costly and time-consuming repairs, impacting both the financial aspect and the project timeline. Therefor, its beneficial to have the ability to override the control system if it where to fail. The override commands are controlled using:
   - A joystick, used to control the direction the rocket would fly in.
   - A potentiometer slider to control the thrust
   - A switch to arm or disarm the rocket.

The ground controller includes an Arduino UNO connected to a computer running MATLAB to receive and view live flight data. The communication between the rocket (Teensy 4.1) and the ground control (Arduino UNO) were done wirelessly with the use of two transceivers *nRF24L01+PA+LNA* [13] connected via SPI to its respective MCU. However, an unresolved issue in communication between the Teensy 4.1 and the Arduino UNO, when using the aforementioned transceivers, caused the ground control to be unutilized, which is further discussed in chapter 5.

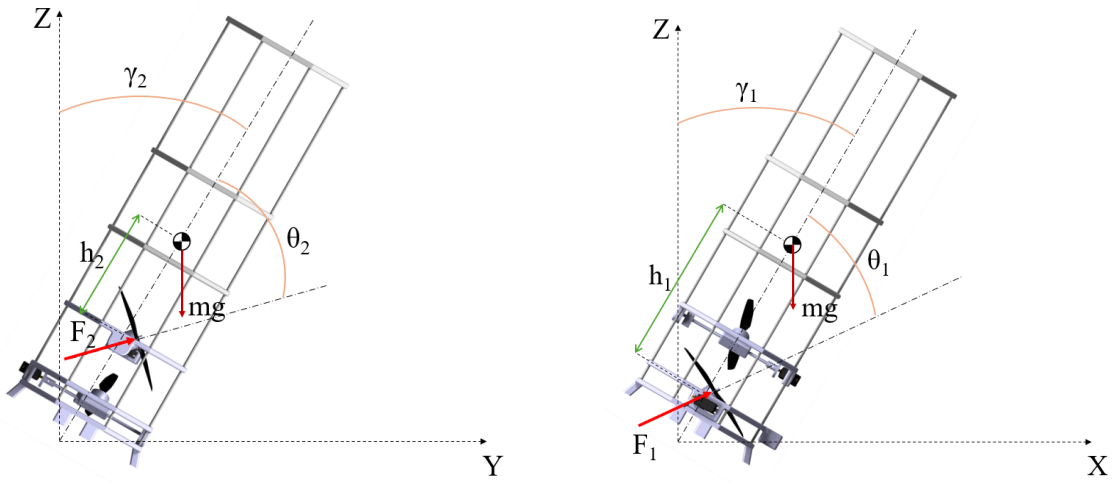
3

Rocket control

This chapter covers the dynamics of the rocket and how they can be used to express the system in a linear state-space model to use for the linear control design. Simulations are made for the system using MATLAB and simulink. The files for the simulations can be found at the projects GitHub page.

3.1 Rocket dynamics

The forces acting on the rocket are illustrated in Fig. 3.1.



(a) The rocket in the y-z-plane.

(b) The rocket in the x-z-plane.

Figure 3.1: Defined angles and forces acting on the rocket.

These forces can be described using newtons second law of motion which gives the equations 3.1-3.6. The motion along the x and y axis are described as:

$$\ddot{x} = \frac{F}{m}\sin(\gamma_1 + \theta_1) \tag{3.1}$$

and:

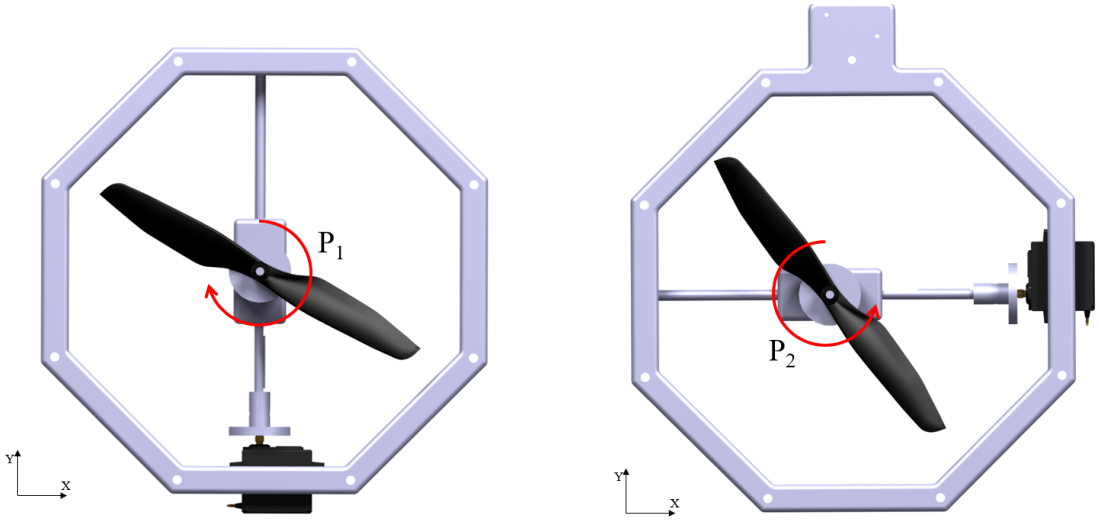
$$\ddot{y} = \frac{F}{m}\sin(\gamma_2 + \theta_2) \tag{3.2}$$

where the mass of the rocket is m. γ_1 is the rotation around the x-axis and γ_2 is the rotation around the y-axis. θ_1 and θ_2 are the gimbal angles of the motors. $F = F_1 = F_2$ is the force generated by the two individual motors. To counteract

rotation around the z-axis the motors should be operating at the same speed as can be seen in:

$$M_z = P_1 - P_2 \tag{3.3}$$

where M_z is the torque around the z-axis. P_1 and P_2 is the torque generated by motor 1 and 2 respectively. These forces are illustrated Fig. 3.2.



(a) The torque generated by motor 1.

(b) The torque generated by motor 2.

Figure 3.2: This Figure shows the torque generated by each motor.

The vertical motion of the rocket is described as:

$$\ddot{z} = \frac{F_1}{m}\cos(\gamma_1 + \theta_1) + \frac{F_2}{m}\cos(\gamma_2 + \theta_2) - g \tag{3.4}$$

and finally the rotational motions around the x and the y axis are described as:

$$\ddot{\gamma}_1 = \frac{F\sin(\theta_1)h_1}{\overline{I}_y} \tag{3.5}$$

and:

$$\ddot{\gamma}_2 = \frac{F\sin(\theta_2)h_2}{\overline{I}_x} \tag{3.6}$$

where $\overline{I}_x$ and $\overline{I}_y$ are the moment of inertia around c.o.m for respective axis, h_1 and h_2 is the distance between the rockets center of mass to motor 1 and 2 respectively. These equations are used to express the system as a state-space model as shown in section 3.3.

3.2 Controlling the rocket

In this study the rockets controller is designed to take the rocket to a pre-defined altitude and hover stationary in place. To achieve this the controller uses the thrust

(F) and the gimbal angles (θ_1 and θ_2) as inputs to control the outputs $\dot{x}$, γ_1, $\dot{\gamma}_1$, $\dot{y}$, γ_2, $\dot{\gamma}_2$, z and $\dot{z}$.

The linear quadratic regulator (LQR) is a full state feedback controller that can handle multiple inputs and outputs (MIMO) to a system. Because the system to be controlled is a MIMO system, an LQR controller is suitable to control the system. Using LQR for controlling similar systems to this projects rocket model can also give more stable control compared to PID according to [14].

The LQR takes the linear system in a state-space format which is further explained in 3.3. Then two cost matrices Q and R are defined. The Q matrix defines how much a deviation for the different state-variables are penalized. For example if a state is considered more important than another, this state can be penalized more for deviating the desired value. The R matrix penalizes how much the inputs are used, for example maybe fuel is an issue so the thrust is more penalized so it won't use it as much. These cost matrices are used in the cost function to calculate the gain matrix K. The cost function is defined as:

$$J = \int_0^\infty x^T Q x + u^T R u \, dt \tag{3.7}$$

and to obtain the optimal gain matrix K the equation is solved to minimize J with the substitution

$$u = -Kx \tag{3.8}$$

For further reading about LQR, see chapter 9 in [15]. In this project the MATLAB command *lqr*[1] was used to calculate K. This means that K can quickly be calculated for different Q and R matrices. So tuning the controller means changing Q and R instead of choosing the gain values directly as for PID.

The PID control used in [1] used three different PID controllers to control the γ_1 and γ_2 angles and the altitude z. For this project a new LQR was designed to control more states to see if it would improve the control. The following sections describes the new LQR design, for further reading about the PID control from the previous project see Appendix D.

3.3 State-space model

To express the system as a linear state-space model the standard notation is used:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{3.9}$$

where x is the state vector, u is the input vector and y is the output. The x and u vectors needs to be defined as well as the A, B, C and D matrices. The u vector is

[1] `https://se.mathworks.com/help/control/ref/lti.lqr.html`

defined by the inputs of the system which is the thrust (F) and the gibmbal angles (θ_1 and θ_2). Therefore there are three inputs defined as:

$$u = \begin{bmatrix} F \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{3.10}$$

As shown in section 3.1, the system is described by multiple non-linear differential equations. Therefore to make the model linear the angles are assumed to be small so:

$$\sin(\gamma_1) \approx \gamma_1 \tag{3.11}$$

and:

$$\cos(\gamma_1) \approx 1 \tag{3.12}$$

This makes it possible to separate the incline angles $\gamma_{1,2}$ from the gimble angles $\theta_{1,2}$ and describe the system on state-space form. The state vector x can be defined as:

$$x = \begin{bmatrix} \dot{x} \\ \gamma_1 \\ \dot{\gamma}_1 \\ \dot{y} \\ \gamma_2 \\ \dot{\gamma}_2 \\ z \\ \dot{z} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} \tag{3.13}$$

and with the small angle assumption $\dot{x}$ can be formulated as:

$$\dot{x} = \begin{bmatrix} \ddot{x} \\ \dot{\gamma}_1 \\ \ddot{\gamma}_1 \\ \ddot{y} \\ \dot{\gamma}_2 \\ \ddot{\gamma}_2 \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{u_1}{m}(x_2 + u_2) \\ x_3 \\ \frac{u_1}{I_x} h_1 x_2 \\ \frac{u_1}{m}(x_5 + u_3) \\ x_6 \\ \frac{u_1}{I_y} h_2 x_5 \\ x_8 \\ \frac{u_1}{m} - g \end{bmatrix} \tag{3.14}$$

Note that the x and y positions are not included in the state vector. The choice to control the velocities $\dot{x}$ and $\dot{y}$ and not the position is made due to the limited sensors and state estimation accuracy, as discussed in section 2.2.2 and 3.4. To write the system as a state-space model, a Jacobian linearization, which is explained in [16], was performed with the equilibrium point:

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ any \\ 0 \end{bmatrix} u = \begin{bmatrix} \frac{mg}{2} \\ 0 \\ 0 \end{bmatrix} \tag{3.15}$$

this equilibrium point reflects the rocket hovering at any altitude. The A and B matrices are calculated as:

$$A = \begin{bmatrix} 0 & \frac{g}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{g}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & g & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{mgh_1}{2\bar{I}_y} \\ 0 & 0 & g \\ 0 & 0 & 0 \\ 0 & \frac{mgh_2}{2\bar{I}_x} & 0 \\ 0 & 0 & 0 \\ \frac{2}{m} & 0 & 0 \end{bmatrix} \tag{3.16}$$

and the C and D matrices are:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.17}$$

The C matrix is the identity matrix because all the states are measured or estimated. The D matrix is full of zeros because the inputs of the system doesn't affect the outputs directly.

For the state-space model to be useful it has to be controllable. This means that its possible to control each state with the defined inputs. If it is not controllable the gain K cannot be calculated for the LQR. To check the controllability of the system, the controllability matrix can be calculated. If the rank of this matrix is equal to the number of states in the system, the system is controllable as explained on page 45 in [15]. To check this the simple MATLAB command $rank(ctrb(A, B))$[2][3] was run. The command returned the rank 8, indicating that the system is controllable.

3.4 Sensor fusion and filters

The readings provided from the sensors are:

- IMU
  - Accelerometer
    - $\ddot{x}$, $\ddot{y}$, $\ddot{z}$
  - Gyroscope
    - $\dot{\gamma}_2$, $\dot{\gamma}_1$, $\dot{\Phi}$
- Lidar
  - z

[2] `https://se.mathworks.com/help/matlab/ref/rank.html`

[3] `https://se.mathworks.com/help/control/ref/statespacemodel.ctrb.html`

- Optical flow sensor
  - $\dot{x}$, $\dot{y}$

All of the sensors contribute in estimating the full state vector x that can be seen in equation 3.13. Observe that all states cannot be measured directly with these sensors. Additionally, testing of the sensors showed that the raw sensor data contains noise. To obtain robust LQR control, it is essential that the estimated state vector $\hat{x}$ is as precise as possible. The initial plan to estimate these states and filter out the noise was to use a combination of a Madgwick filter and a Kalman filter, along with calculations. In Fig. 3.3, an overview of the different states, sensors and filters can be seen.

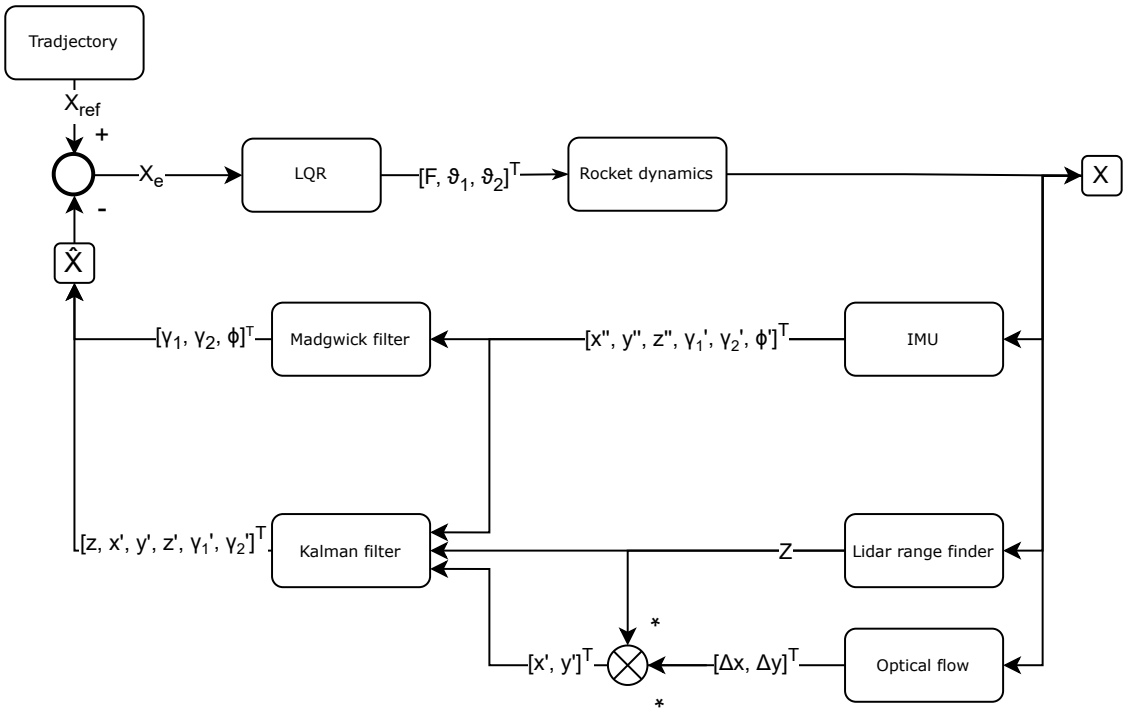


Figure 3.3: Block diagram of states, sensors, filters and controller.

3.4.1 Madgwick filter

In order to estimate the attitude of the rocket $[\gamma_2, \gamma_1, \Phi]$, a Madgwick filter is used. It is a complementary filter that manages attitude estimation with low drift in the estimates [17]. Drift, in this context, referees to a steady growth in error between the estimated value and the actual real value. It is caused by integrating the noise in the rotational speed-measurement from the IMU, when estimating the angle. The Madgwick filter is simpler to set up and uses an optimization algorithm (gradient descent) to estimate the states compared to filters with a probabilistic approach such as Kalman filters. Madgwick was chosen for attitude estimation since it requires less tuning than probabilistic filters to achieve high performance.
The input to the filter is the following:

$$\begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z} & \dot{\gamma}_2 & \dot{\gamma}_1 & \dot{\Phi} \end{bmatrix}^\top \tag{3.18}$$

There is essentially only one filter parameter that can be tuned in a Madgwick filter (the β-parameter), which influences the trade-off between filter speed and accuracy over time. Testing by trial and error found that $\beta = 0.038$ worked best in combination with an IMU sample rate of 400 Hz and a filter frequency of 2000 Hz. The resulting output from the madgwick filter is:

$$\begin{bmatrix} \gamma_2 \\ \gamma_1 \\ \Phi \end{bmatrix} \tag{3.19}$$

3.4.2 Kalman Filter

The Madgwick filter only estimates a subset of the state vector x, leaving the following states to be estimated and filtered in other ways:

$$\begin{bmatrix} \dot{x} \\ \dot{\gamma}_1 \\ \dot{y} \\ \dot{\gamma}_2 \\ z \\ \dot{z} \end{bmatrix} \tag{3.20}$$

All of states in 3.20, except $\dot{z}$, can be measured directly from the sensors. $\dot{z}$ is obtained by taking the discrete derivative of z, resulting in:

$$\dot{z} = \frac{z(n+1) - z(n)}{\Delta t} \tag{3.21}$$

As previously mentioned, the sensor readings contain noise. This can be filtered out using a Kalman filter [18]. It produces an estimate of the system states based on a prediction combined with the noisy sensor data. Using the covariance of the measurements and process disturbances, the Kalman filter weights the prediction and measurements together to obtain an effectively filtered state vector.

To implement the Kalman filter, a number of parameters are required:

- The system model A (3.3) for the states to be estimated.
- Sensor noise covariance matrix
- Covariance matrix for process disturbances

The system model A is known from the state-space model as discussed in chapter 3.3. The sensor noise covariance for the IMU is provided by its data sheet. However, the noise covariance for the other sensors will have to be derived experimentally in further research. Additionally, the covariance for process disturbances needs to be tuned experimentally in a continuation of the project as outlined in chapter 5. The Kalman filter was thus not utilized in this project.

3.5 Simulations

To design the LQR controller, simulations in simulink were made. The following simulink model was created with the state-space model in chapter 3.3.

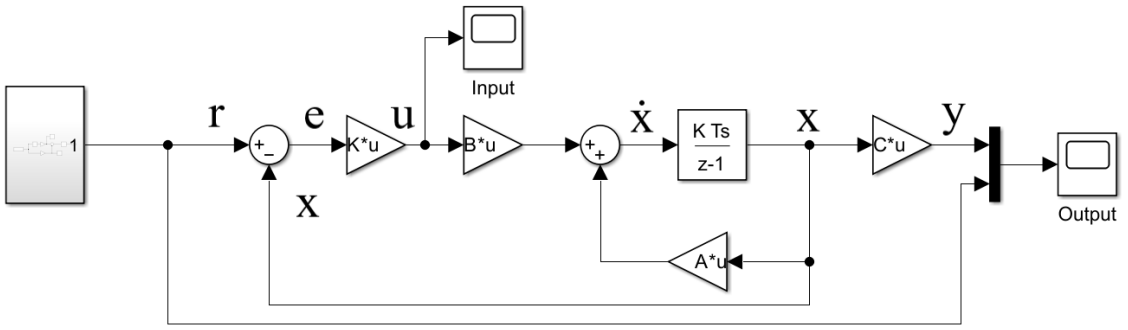


Figure 3.4: This is a simple LQR model in simulink. Here r is the reference value and e is the error. $\dot{x}$ is the result of the second sum that adds up Bu and Ax. The output y is Cx, just like the state-space model in equation 3.9.

This model gives a more intuitive understanding of how the linearized model works. One problem this model has is that it is linear and the system to be controlled is non-linear. Because of this the the model in Fig. 4.2 was mainly used to test the system and see if the output was reasonable before moving on to a non-linear simulation. The new model in Fig. 3.5 was made to simulate the system with the non-linear equations thereby providing a more accurate simulation.

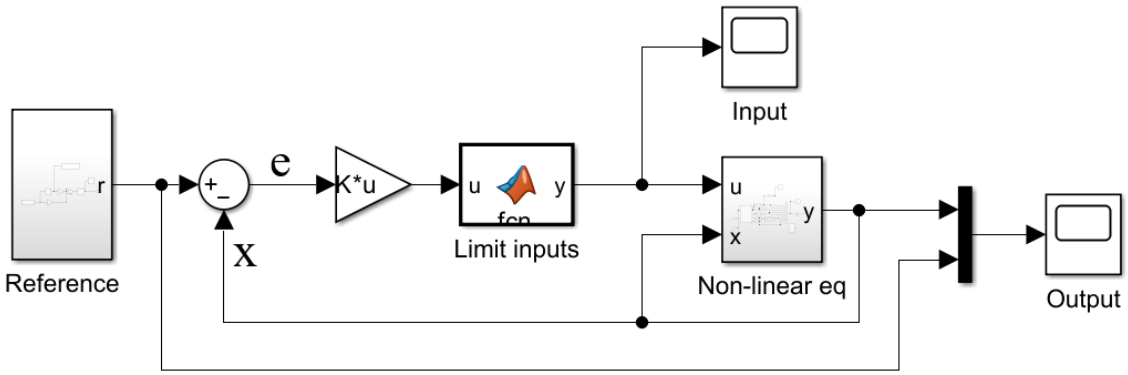


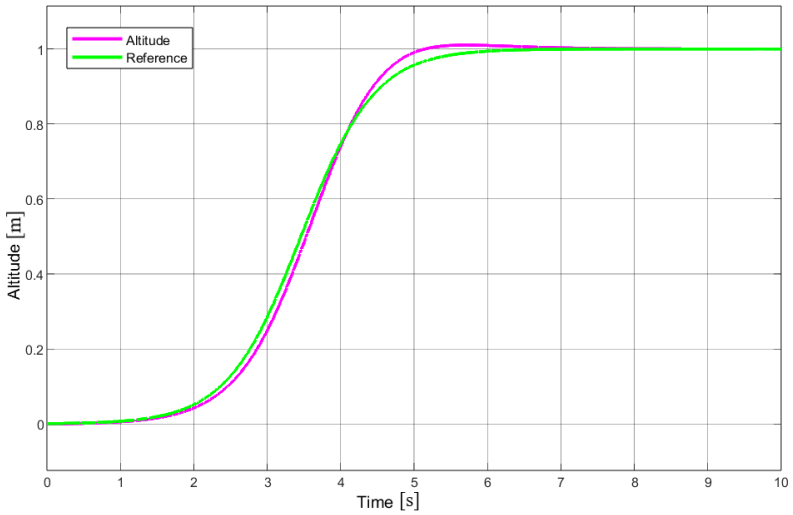
Figure 3.5: The reference r is subtracted by x giving the error e. e is then multiplied by the matrix k giving the u vector. The "Limit inputs" block constrains the u vector values to reflect the capabilities of the rocket model, i.e. it limits the thrust and gimble angles. The "Non-linear eq" block then runs the values through the non linear equations resulting in the output values.

Because both the velocity ($\dot{z}$), and altitude (z), are controlled, the reference for these states need to change over time in order not to contradict each other. In order to do this, different set points for the altitude was calculated to provide a reasonable

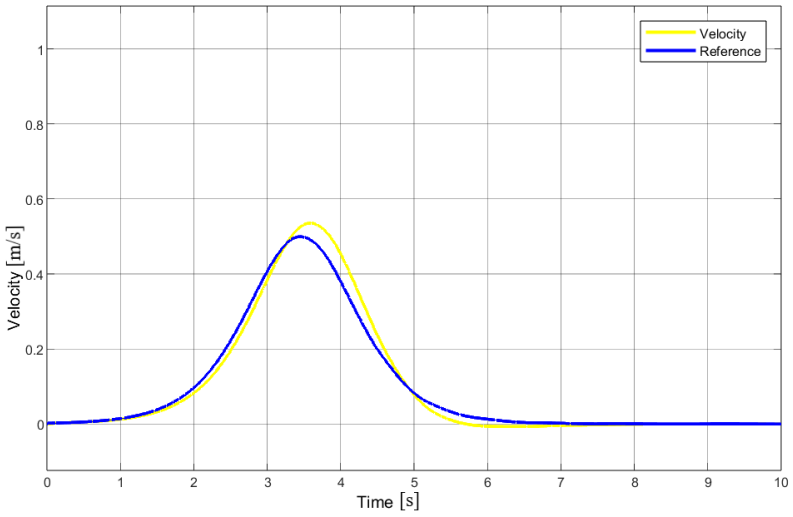
flight path. This was done with the logistic growth [4] function:

$$\dot{a}(t) = ra(t)(1 - \frac{a(t)}{a_g}) + C \tag{3.22}$$

where the constant C was added to stop $\dot{a}(t)$ from initially being zero. r is a constant that determines how fast the altitude should reach its goal, $a(t)$ is the altitude at time t and a_g is the goal altitude. In Fig. 3.6 the reference is plotted together with the output from the simulink model in Fig. 3.5. In this simulation the constants are $r = 2$, $C = 0.001$ and $a_g = 1$.



(a) The altitude plotted with the reference value.

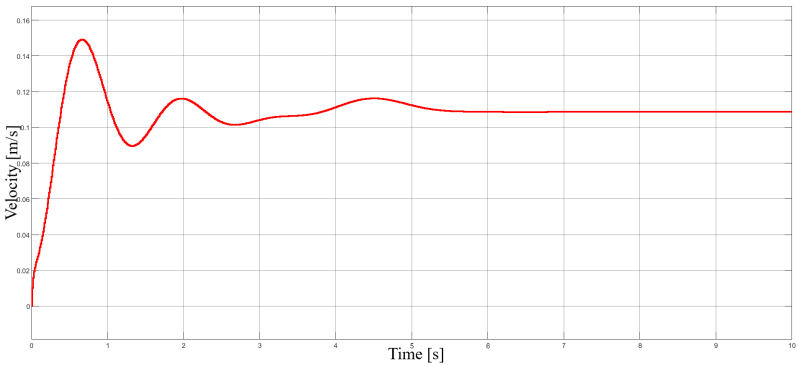


(b) The velocity plotted with the reference value.

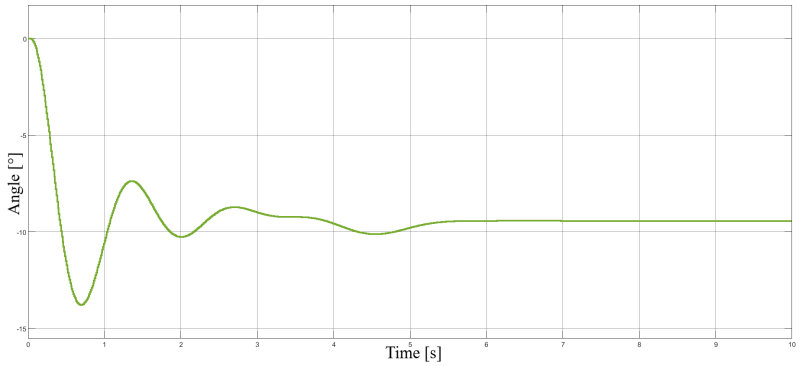
Figure 3.6: This figure shows the reference and output for the altitude (z) and ($\dot{z}$).

[4] `https://services.math.duke.edu/education/ccp/materials/diffeq/logistic/logi1.html`

This simulation gave promising results but it does not take disturbances into consideration, such as turbulence wind or ground effect (a phenomenon that occurs for aircraft's close to the ground). Without these disturbances the controller isn't really being tested with regards to the gimbal mechanism because the angles $\gamma_{1,2}$ are always zero unless disturbances are introduced. Therefore another simulation was made with a disturbance added to the $\ddot{x}$ and $\ddot{\gamma_1}$ state by adding a constant $w = 1$ to equations 3.1 and 3.5 to simulate wind.



(a) The velocity $\dot{x}$ plotted with respect to time.



(b) The angle γ_1 plotted with respect to time.

Figure 3.7: This figure shows how the state $\dot{x}$ and γ_1 behaves with added disturbance.

Fig. 3.7 shows the results for the states $\dot{x}$ and γ_1 with the added disturbance. The $\dot{x}$ velocity is reduced from 1 m/s to around 0.11 m/s by the controller and the γ_1 is stabilizing around -10 degrees to compensate for the disturbance. The disturbance of $w = 1$ probably doesn't reflect a real disturbance due to the complexity of the system, but it shows that the controller reacts to a disturbance and reduces its effects on the system. The steady state error that occurs in Fig. 3.7 (a) can be minimized by tuning the Q and R matrices through trial and error. This was not done on the simulated model due to the unknown disturbance parameters that would effect the real system wouldn't be accounted for anyway.

3.6 System identification

System identification is a method used to estimate a mathematical model of a system. It uses input and output data to estimate different parameters of a system. It can provide a better model of the system if performed correctly, especially if the system is hard to model analytically. Due to the complexity of this system with two propellers with unknown individual characteristics and disturbances (such as exact thrust and turbulence) system identification was performed to see if it would be better than the analytical state space model in section 3.3.

For system identification to be useful, the system should be experiencing as many different inputs and outputs as possible in order to get a good estimation of the systems model. This input and output data needs to be collected during a test and the test environment needs to be as similar to the real environment as possible. The results of this and how it was performed is further explained in section 4.4.

4

Test flights and discussion

This chapter details the rocket testing procedures, parameter adjustments, and their corresponding outcomes. It also encompasses system identification techniques employed to enhance the estimation of a more accurate state-space model. All flight tests presented were recorded by video[1].

4.1 Software implementation

The flight controller, including sensor fusion, control and telemetry logging was implemented in the Arduino variant of c++. This was used due to its ease of programming and implementation on the hardware. The code is available in the projects GitHub page. The code for wireless communication protocol between the rocket and the ground controller can also be found there, but is inactivated when using the Teensy 4.1.

4.2 Thrust tests

Moving the propellers closer, as described in section 2.1.2, was an essential modification. However it could effect the total thrust generated by the rocket due to air at greater velocity entering the second motor. A test to assess performance at various distances was conducted on the initial model to ensure thrust would remain sufficient enough to lift the rocket.

The test was done by systematically reducing the distance between the top and bottom engines until reaching a minimum of 100mm, ensuring both motors would be able to gimbal without colliding. This minimum distance was estimated, considering that the gimbal construction was not yet constructed. At various distances the initial model was run upside down on a scale with a constant PWM signal of 1800 μs, equivalent to 92% throttle. The results are shown in Table 4.1. A negligible difference in thrust was observed confirming significant thrust when mounting the motors at a distance of 125mm. This distance, although not tested, was chosen when constructing the new model for safety reasons (prevent propeller clash) and due to fitting of electrical cables.

[1] `https://www.youtube.com/playlist?list=PLLyyVZOFIb_I-XsQ4xieir3YfaWF3Y6mQ`

Table 4.1: Distance and thrust test on previos model

Distance [mm]	**Throttle [μs]**	**Thrust [kg]**
315	1800	3.25
167	1800	3.2
100	1800	2.95, 3.2

Furthermore, several thrust tests were conducted on the final Starship model.

1. Running the engines at increments of 10% of throttle from 10% to 80% a thrust curve could be obtained and used in the LQR to convert input signal F (thrust) to a PWM signal that is sent to the motors. Furthermore this gave an overview of how the final Starship model perform at different throttle percentages with both motors running. The results are presented in Fig. 4.1

Trottle [%]	Thrust [g]
10	90
20	340
30	880
40	1410
50	1810
60	2430
70	3090
80	3450

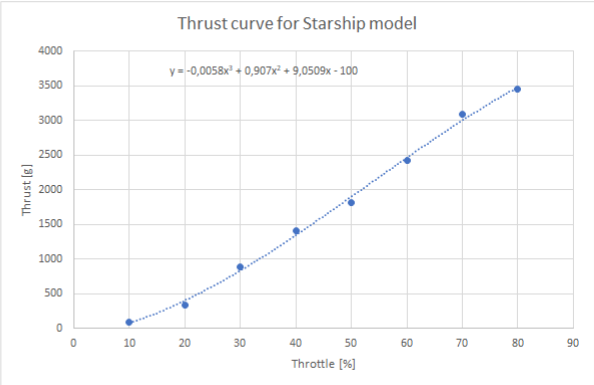


Figure 4.1: Thrust generated by the final Starship model.

2. To evaluate the impact of gimbal angle upon thrust, tests were conducted at two different throttle settings and three angles. The gimbal angles θ_1 and θ_2 where set to the same values. The engines were running at the specified throttle while the model was pointed downwards onto a scale to measure thrust. The results are presented in Table 4.2, where 40% throttle and 5° results in 1330g of thrust.

Table 4.2: Angle's (θ_1 and θ_2) effect on thrust

Angle (θ_1 and θ_2) [°] / Throttle [%]	**5**	**10**	**20**
40	1330 [g]	1330 [g]	1290 [g]
80	3460 [g]	3400 [g]	3130 [g]

When comparing thrust at 40% and 80% throttle with the results in Fig. 4.2 a difference is observed. However, at 40% throttle the difference is negligible and the discrepancy at 80% throttle isn't detrimental. Since the model mostly will

be flown at or over 80% throttle this is the most interesting results confirming that the model will have sufficient thrust even when the motors gimbal at 20°.

4.3 Flight tests

To test the rockets flying capabilities without risking a crash, a test rig was used. This test rig consisted of two ropes attached to the rocket. One was attached to the top of the rocket and hung up in a fixed point above the rocket and the other was attached to the bottom of the rocket. This way one could easily save the rocket if needed by pulling in each rope. A picture of the test rig can be seen in Fig. 4.2. The point of the flight tests was to see if the rocket could fly to a pre-defined altitude and hover in place. The reference values explained in section 3.5 was used for the flight tests with LQR and for the PID testing a reference of $\gamma_1 = \gamma_2 = 0$ and $z = 1$ was used.

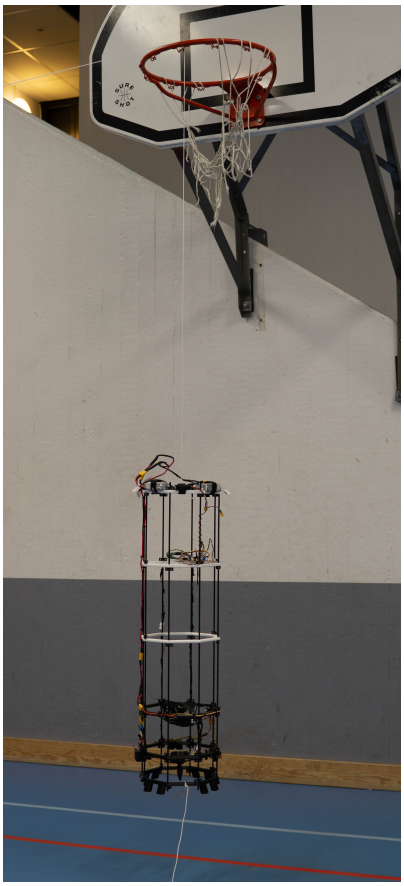


Figure 4.2: Test rig where the model could be saved by pulling on either the upper or lower string.

The first flight test was done with the PID control, used in [1] with tweaked gain values. The PID gain values can be seen in Table 4.3.

Table 4.3: Here the gain values used for the PID controllers are presented.

	Atitude PID	Angle PID (γ_1)	Angle PID (γ_2)
K_p	20	2	0.5
K_i	5	0.01	0.01
K_d	18	10	10

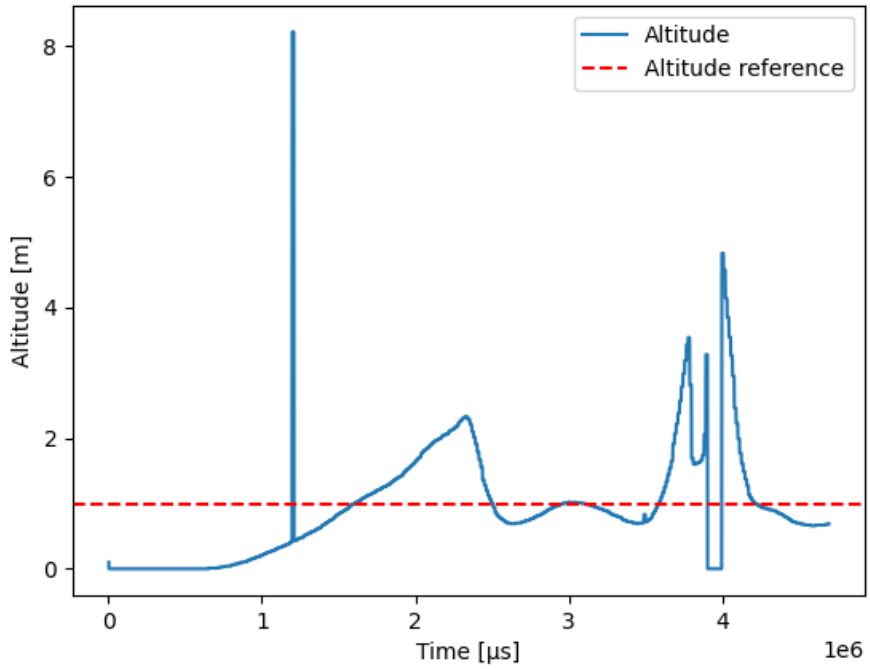
The angle PIDs gains differ because of the motors distance to the c.o.m. are not the same. Therefore the K_P gain for the top motors angle PID was increased to make up for the leverage. The LQR controller was tested and tuned through trial and error, prioritizing altitude and angles resulting in these Q and R matrices:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \tag{4.1}$$

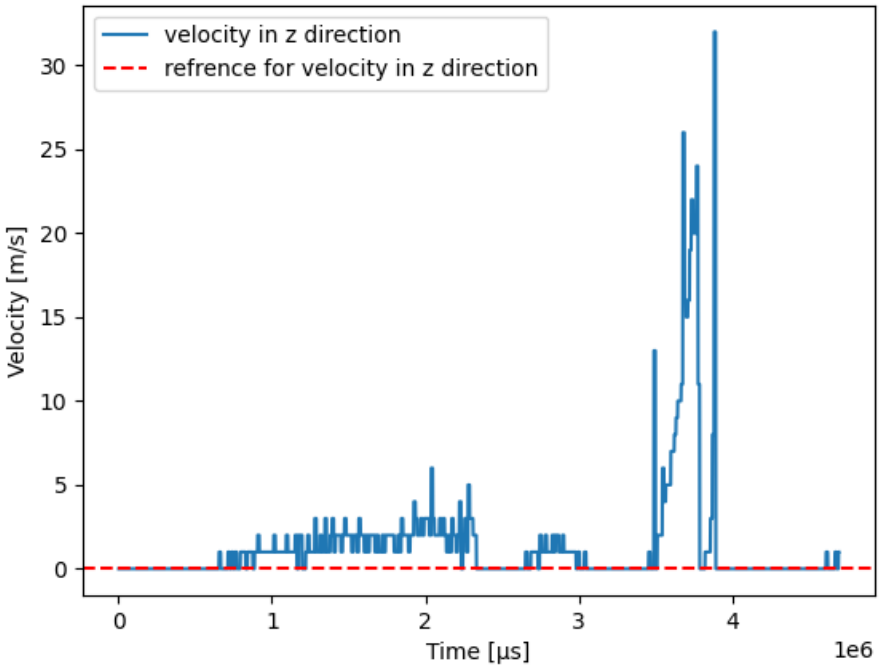
used to calculate the gain matrix K used for the test, plotted in Fig. 4.4. The following matrices was used to calculate the gain matrix K used in the test plotted in Fig. 4.5.

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \tag{4.2}$$

The data collected from the flight tests are plotted together with the reference for the states in Fig. 4.3 for the PID flight and in Fig. 4.4 and Fig. 4.5 for both LQR flights.



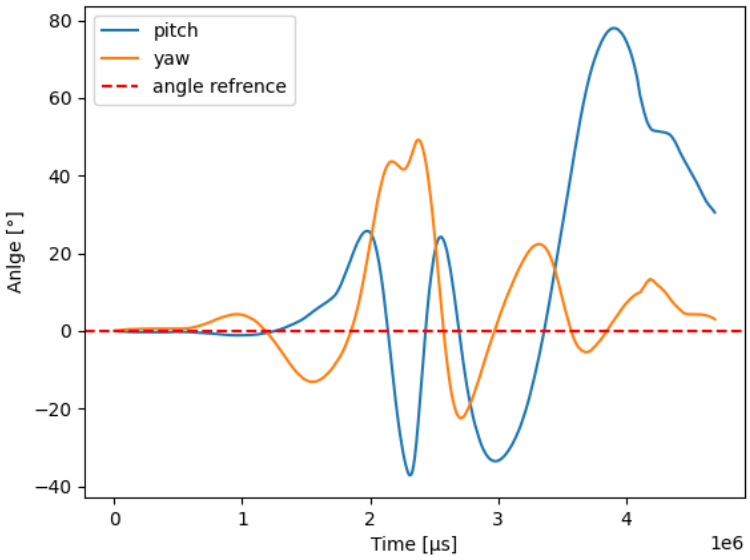
((a)) Altitude plot



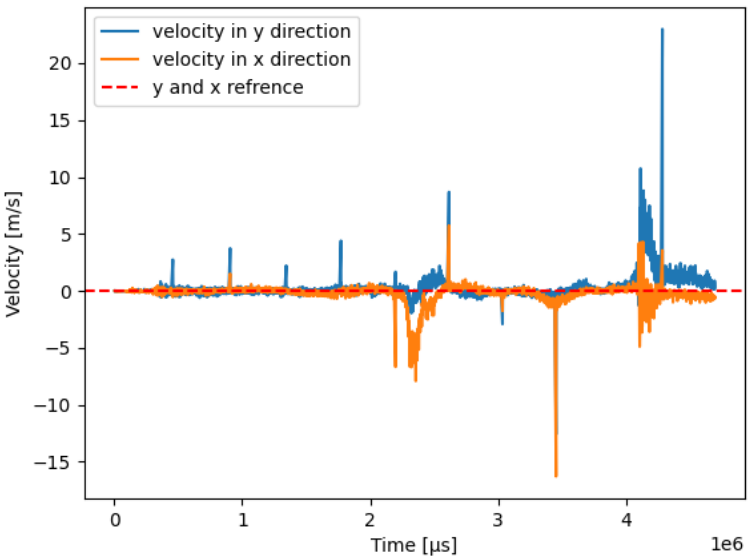
((b)) Plot for velocity in z

Figure 4.3: The first two figures of the test flight using a PID controller.

During the flight test with PID control the rocket quickly lost control but was saved



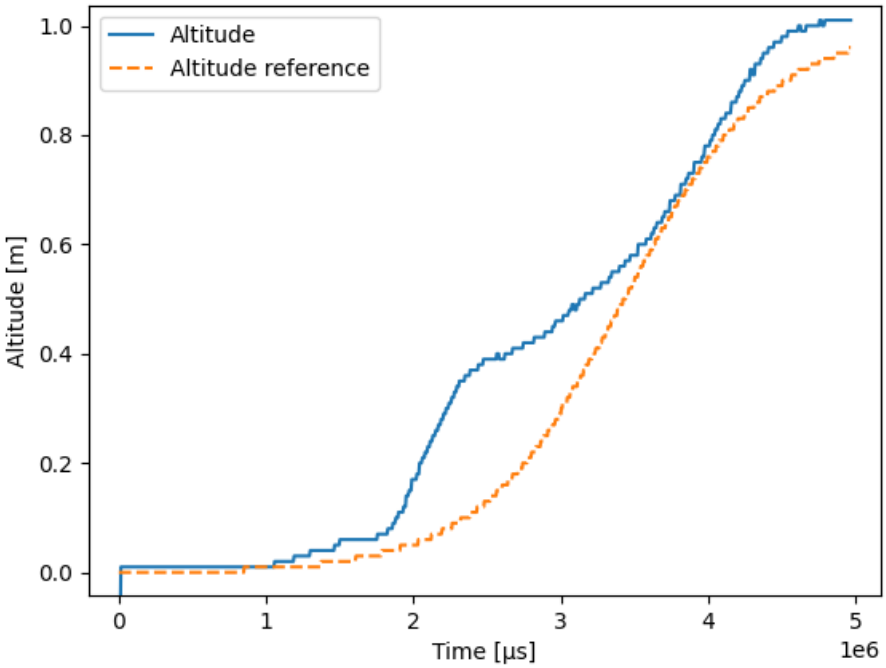
((c)) Angles plot



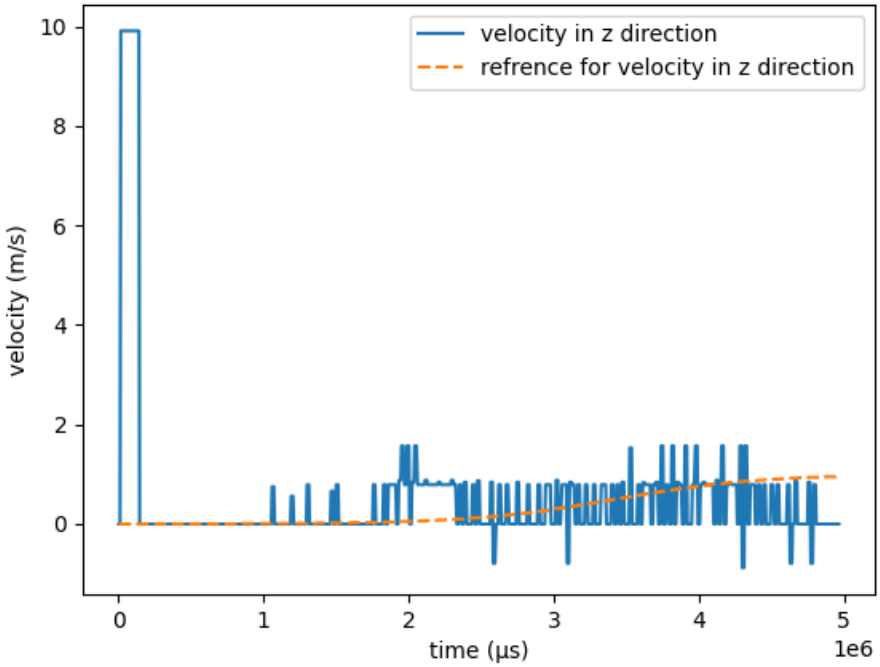
((d)) Velocity in x and y direction plot

Figure 4.3: The last two figures of the test flight using a PID controller.

by the test rig, as can be seen in the flight data plotted in Fig. 4.3. It was very difficult to see what gain values needed to be tuned to get a better result, confirming the troubles with PID control stated in section 3.2.

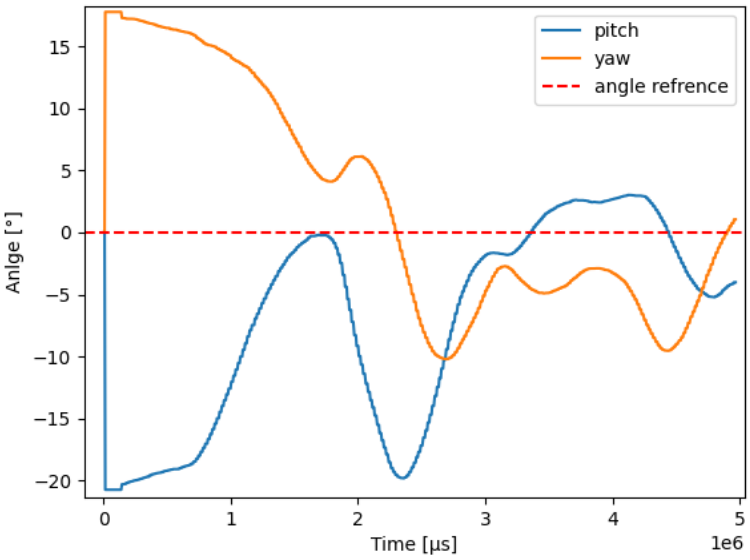


((a)) Altitude plot

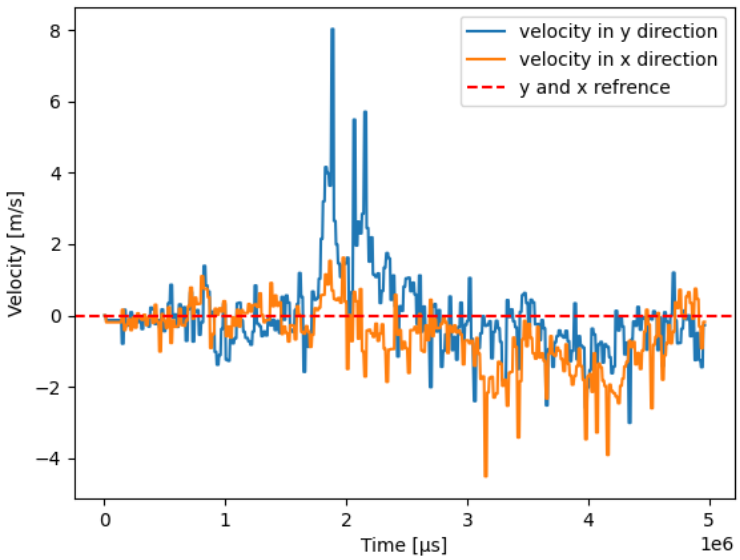


((b)) Plot for velocity in z

Figure 4.4: The first two plots for test flight with the first LQR using K calculated from Q and R shown in (4.1)



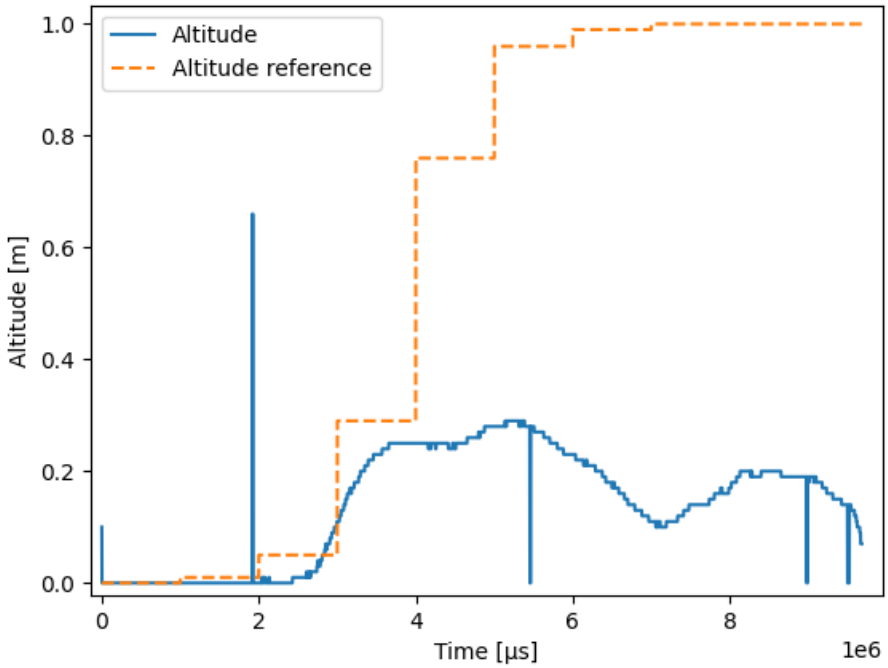
((c)) Angles plot



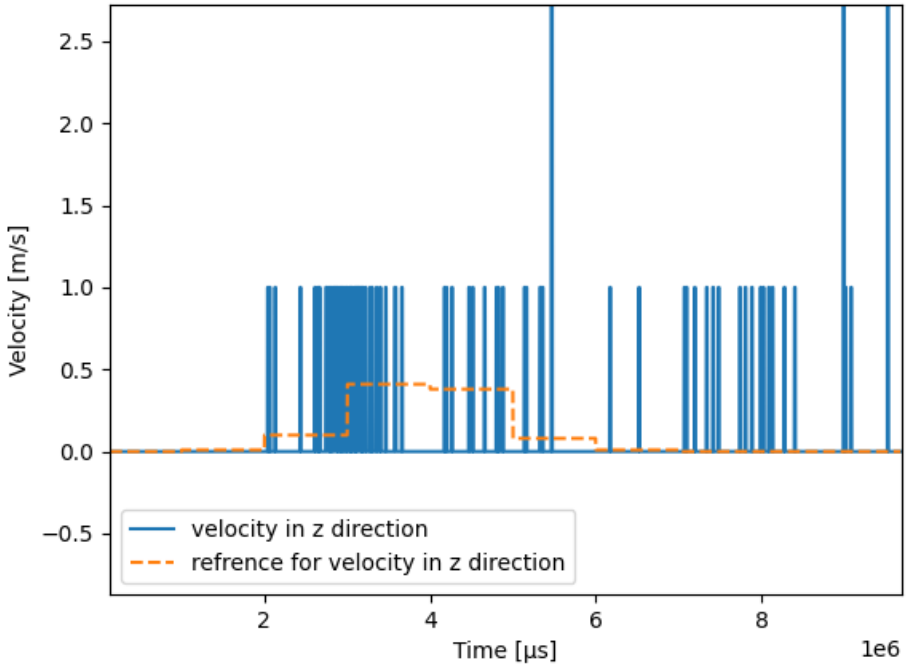
((d)) Velocity in x and y direction plot

Figure 4.4: The last two plots for test flight with the first LQR using K calculated from Q and R shown in (4.1).

In the test with the LQR controller in Fig. 4.4, the (a) plot shows the altitude being controlled along the reference. The altitude line growths a bit faster in the beginning compared to the reference which is due to ground effect.

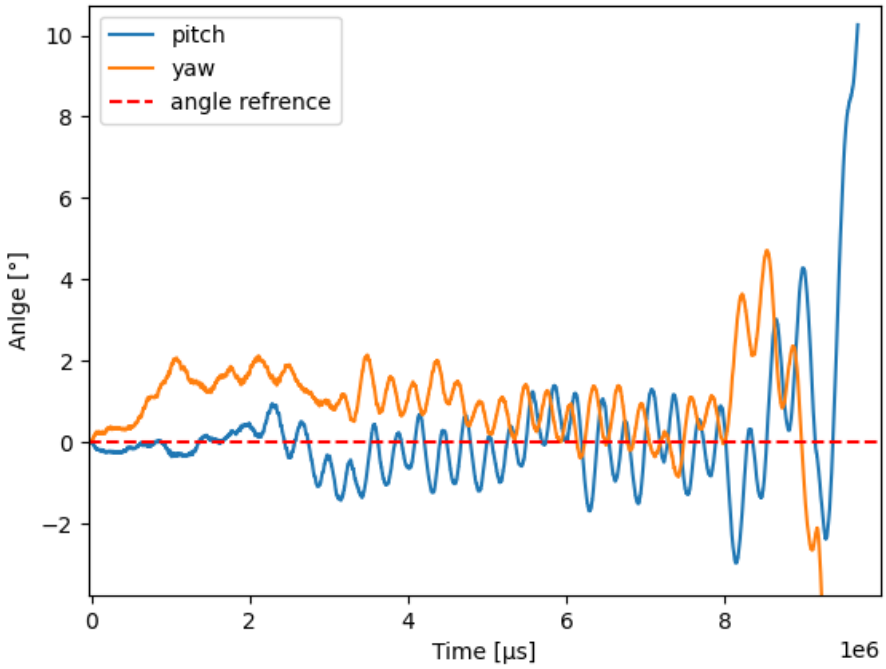


((a)) Altitude plot

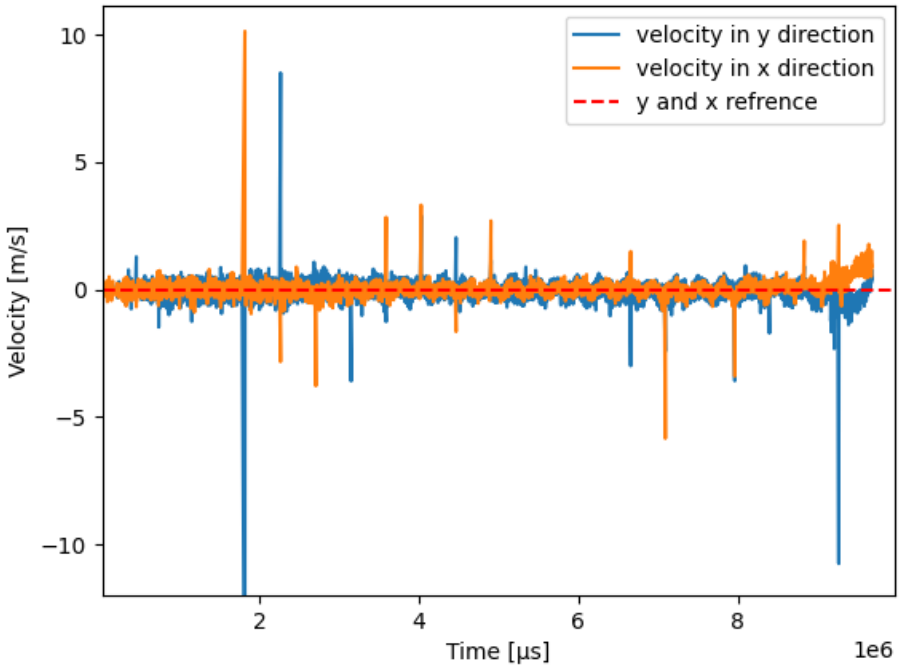


((b)) Plot for velocity in z

Figure 4.5: The first two plots for test flight with the second LQR using K calculated from Q and R shown in (4.2).



((c)) Angles plot



((d)) Velocity in x and y direction plot

Figure 4.5: The last two plots for test flight with the second LQR using K calculated from Q and R shown in (4.2).

The LQR controller was able to achieve a stable hover for a few seconds in the flight test corresponding to Fig. 4.5. However due to low battery levels it was unable to generate enough thrust to track the altitude reference correctly.

4.3.1 Conclusion of flight tests

When performing the flight tests with LQR the results quickly improved by tuning the Q and R matrices between each flight. It was more intuitive what states needed to be penalized when observing the behaviour of the rocket during each flight compared to the PID approach. Also, by analyzing the recorded flight data the tuning process could be optimized. Tuning a PID would have required more time since there are three separate gain values to adjust for each PID. Due to the noisy flight data that can be observed in Fig. 4.3, 4.4 and 4.5 it was hard to know whether or not the tuning could be improved more. The noisy data indicates that the optical flow sensor and kalman filter needs to be implemented in order to achieve better flight results. This in turn would have provided the LQR with correct data and thus resulted in better flight results.
Due to the aforementioned results, the LQR is preferred due to better results and a quicker tuning process.

4.4 System identification results

All the inputs and outputs explained in 3.2 are stored on the rockets micro SD card during the flight tests to be used in system identification. The "System Identification Toolbox"[2] app in matlab was used to estimate the the new state-space model. The new A matrix is:

$$A = \begin{bmatrix} 0.983 & 0.027 & 0.076 & -0.053 & 0.000 & -0.024 & -0.036 & -0.018 \\ 0.005 & 0.953 & -0.245 & 0.120 & 0.009 & 0.001 & 0.074 & 0.025 \\ -0.051 & 0.216 & 0.533 & -0.725 & -0.166 & 0.164 & 0.184 & 0.235 \\ 0.036 & 0.077 & 0.672 & 0.649 & -0.041 & 0.138 & 0.004 & -0.008 \\ 0.036 & -0.004 & 0.019 & -0.125 & 0.948 & 0.192 & 0.096 & -0.064 \\ -0.011 & -0.014 & -0.098 & 0.045 & -0.098 & 0.894 & 0.190 & -0.266 \\ 0.041 & -0.025 & -0.038 & -0.080 & -0.191 & 0.055 & 0.736 & -0.190 \\ 0.022 & -0.011 & -0.000 & 0.005 & -0.026 & 0.313 & -0.379 & 0.593 \end{bmatrix} \tag{4.3}$$

and the new B matrix is:

$$B = \begin{bmatrix} -0.001 & -0.000 & 0.000 \\ 0.005 & 0.001 & -0.001 \\ 0.009 & 0.001 & -0.000 \\ -0.014 & -0.001 & 0.001 \\ 0.002 & -0.001 & 0.001 \\ 0.016 & -0.000 & 0.000 \\ 0.004 & -0.003 & 0.003 \\ 0.003 & -0.005 & 0.006 \end{bmatrix} \tag{4.4}$$

[2] `https://se.mathworks.com/products/sysid.html`

This model was used to calculate a new optimal gain matrix K. When this matrix was used in the simulation in Fig. 3.5, it gave the following output seen in Fig. 4.6.

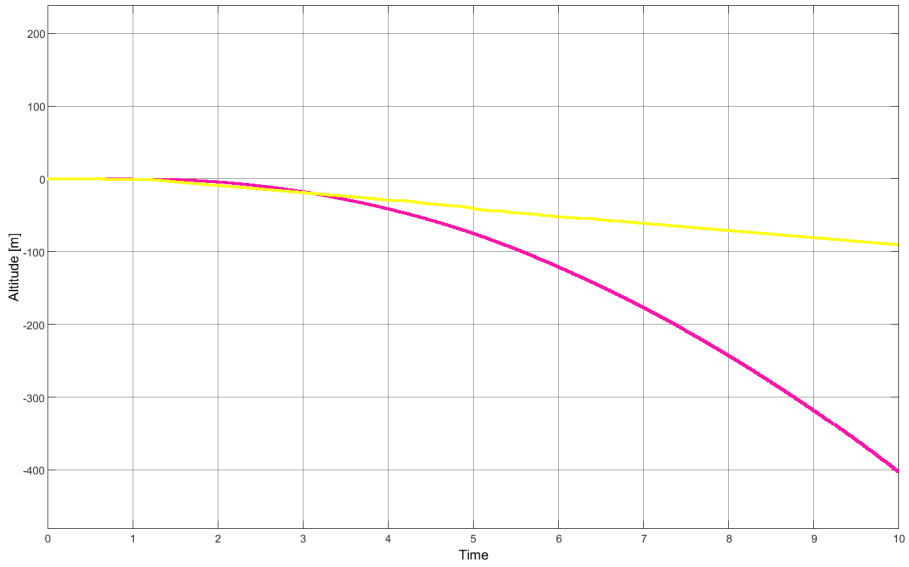


Figure 4.6: The altitude (pink line) and velocity (yellow line) output from the non-linear simulink model, both with respect to time.

The reference for the altitude and velocity where the same as in the simulation in section 3.5 but are not plotted in Fig. 4.6 as they appear to be constant 0 due to the extreme y-axis. This simulation is obviously far worse than the previous one and was for this reason not tested on the real rocket model.

It is observed that the matrices differs a lot from the model in section 3.3. This might be due to the system not being put through a good enough test environment because of the ropes in the test rig interfering with the system during the flight. Additionally, the poor sensor data collected affected the system identification negatively. The test flight might also have been too short to collect a meaningful amount of data for the System Identification Toolbox to give a decent model. The low values in the B matrix in equation 4.4 might occur due to the inputs to the system not changing enough during the test flight.

5
Conclusion and further work

This chapter concludes the project and gives suggestions for further improvements in case of future development of the rocket model.

This report has covered a bachelor thesis project of an electric reusable rocket. By implementing a gimbal construction on the rocket model, the gimbaling motors could facilitate improved control of the rocket. Adding additional sensors and a robust circuit improved precision and reliability, therefore enabling the development of an enhanced control system. The tuning of LQR control proved to be easier and gave better results compared PID control for this application. System identification was attempted to improve the LQR control but it failed and did not improve the controllers performance. By implementing these design changes and using a MIMO system, the rocket model was able to track the state references and perform a stable hover for a few seconds.

By tuning the controller to a finer precision, better results could be achieved, i.e a more stable flight. Furthermore, implementing the following changes would improve the rocket model.

- **Optical flow sensor and Kalman filter implementation**
  The optical flow sensor was not mounted in time for the final tests. Instead, $\dot{x}$ and $\dot{y}$ had to be calculated based on the raw, noisy IMU data. This in turn lead to poor estimates as can be observed from the telemetry logs in section 4.3. Had the optical flow sensor been mounted in time for flight, $\dot{x}$ and $\dot{y}$ would have been much more accurate. This in turn would have enabled the LQR to notice the lateral movement and controlled it to zero, thus maintain a hover within a constrained area. Combining this with the implementation of a Kalman filter would have improved the state estimates even more and thus made for more accurate telemetry.

- **Usage of system identification**
  System identification for this project did not give promising results. It should however not be ruled out for potential future work as it can be performed better. The problems pointed out in section 4.4 can all be fixed by implementing better control, sensor fusion and doing longer test flights. It should be noted that the analytical model might be good enough to control the rocket as shown in section 4.3. Because of this system id might not be needed to model the system accurately for an autonomous flight.

- **Ground controller implementation** Due to some bug in the MCU Teensy 4.1, the wireless communication to the ground controller was not working. The Teensy 4.1 could send packets to an Arduino UNO without problems but could not receive the acknowledgement packet or its data, thus returning "Transmission error". It works flawlessly when using two Arduino UNOs together but not with the Teensy 4.1. Even testcode from the library authors between the two did not work. Additional research why this occurs is needed to have a ground controller to safely command the rocket from a far, especially when testing outdoors.

- **Control** The controller used in this project performed decently, but is not sufficient to perform complicated maneuvers such as the belly-flop maneuver described in appendix F. It can also be expanded to include states for the x and y coordinates. This could make the hover capability more robust but would probably require more sensors such as GPS used together with more more sophisticated sensor fusion algorithms. It also might not be sufficient to land the rocket due to disturbances such as ground effect. To make the landing easier one could try implementing integral action in the LQR to make the landing process smoother. The current controller also does not control the rockets roll parameter (the spin around its own axis). An initial design of a P regulator was made to change the RPM of one of the motors to control roll but it was never tested. The details about this controller can be found in appendix E.

  The use of a model predictive controller (MPC) was reached during this project. The MPC showed promising results in other works and would likely be beneficial for the belly flop maneuver. However the MPC is computational heavy and the current hardware design with the chosen micro controller would presumed not have been able to support a MPC at a desired sampling frequency.

- **Two-axis gimbal with coaxial counter-rotating propellers** To minimize the impact the engines have on each other they could be mounted on the same axis in a two-axis gimbal construction. Much like SpaceX's Starship this would enable the motor to gimbal 360° by itself. By not coupling the motors i.e having only one motor, the engine construction could achieve a greater efficiency. Thus resulting in better and more uniform thrust vectorization which is desirable, this would also minimize the weight of the rocket as an entire gimbal construction would be removed.

- **Belly flop** Another development area for the rocket is to develop the ability to perform a Belly flop maneuver. This makes it more similar to SpaceX's Starship. This manever is explained in appendix F.

References

[1] W. Dahlin, E. Fabricius, G. Geelnard, A. Gleisner, D. Klerebladh, and S. Källhammer, "Modellering och reglering av eldriven raket," Avdelningen för System- och Reglerteknik, Chalmers tekniska högskola, 412 96 Göteborg, Tech. Rep., 2023.

[2] R. E. Guide. "A pictorial history of rockets." Accessed: Mar. 4, 2024. (2012), [Online]. Available: https://www.nasa.gov/wp-content/uploads/2012/03/rockets-guide-20-history.pdf.

[3] SpaceX, *Homepage*, Accessed: Feb. 1, 2024. [Online]. Available: https://spacex.com/.

[4] SpaceX, *Starship SN15 high-altitude flight test*, Accessed: Feb. 1, 2024. [Online]. Available: https://www.youtube.com/watch?v=z9eoubnO-pE&t=777s.

[5] SpaceX, *Starship overview*, Accessed: Feb. 5, 2024. [Online]. Available: https://www.spacex.com/vehicles/starship/.

[6] V. Technologies, *Inertial navigation primer*, Accessed: May. 9. Chapter 2, Section 2.3: Attitude Representation, VectorNav Technologies, n.d. [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-attituderep.

[7] BPS.space, *Bps.space youtube channel*, Accessed: Feb 10, 2024. [Online]. Available: https://www.youtube.com/BPSspace.

[8] Arduino, *Uno r3*, Accessed: Jan. 25, 2024. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3/.

[9] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, Sep. 1995, ISSN: 0360-0300. DOI: 10.1145/212094.212141. [Online]. Available: https://doi.org/10.1145/212094.212141.

[10] PJRC, *Teensy 4.1 development board*, Accessed: Jan. 25, 2024. [Online]. Available: https://www.pjrc.com/store/teensy41.html.

[11] T. Instruments, *A basic guide to i2c*, Accessed: Feb. 06, 2024, 2022. [Online]. Available: https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=

1707199994601&ref_url=https%253A%252F%252Fwww.google.com.hk%252F.

[12] jagan716, *What is serial peripheral interface (spi)?* Accessed Feb. 1, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-serial-peripheral-interface-spi/.

[13] Lastminuteengineers, *How nrf24l01+ wireless module works interface with arduino*, Accessed Feb. 1, 2024. [Online]. Available: https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/.

[14] P. Saraf, M. Gupta, and A. M. Parimi, "A comparative study between a classical and optimal controller for a quadrotor," in *2020 IEEE 17th India Council International Conference (INDICON)*, 2020, pp. 1–6. DOI: 10.1109/INDICON49873.2020.9342485.

[15] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*. 29 West 35th Street, New York, NY 10001: CRC Press, 2010, p. 45.

[16] B. Lennartson, *Reglerteknikens grunder*. Lund: Studentlitteratur, 2006, pp. 93–98.

[17] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, 2011, pp. 1–7. DOI: 10.1109/ICORR.2011.5975346.

[18] G. Welch and G. Bishop, "An introduction to the kalman filter," *Proc. Siggraph Course*, vol. 8, Jan. 2006.

[19] Z. C. Charles Cox. "Thrust vector control of a hypersonic re-entry vehicle." Accessed: Apr. 7, 2024. (), [Online]. Available: https://charm.stanford.edu/ENGR1052016/CharlesCoxZacClausing.

A

Drawings

Drawings of components designed in this project can be seen in the following figures.

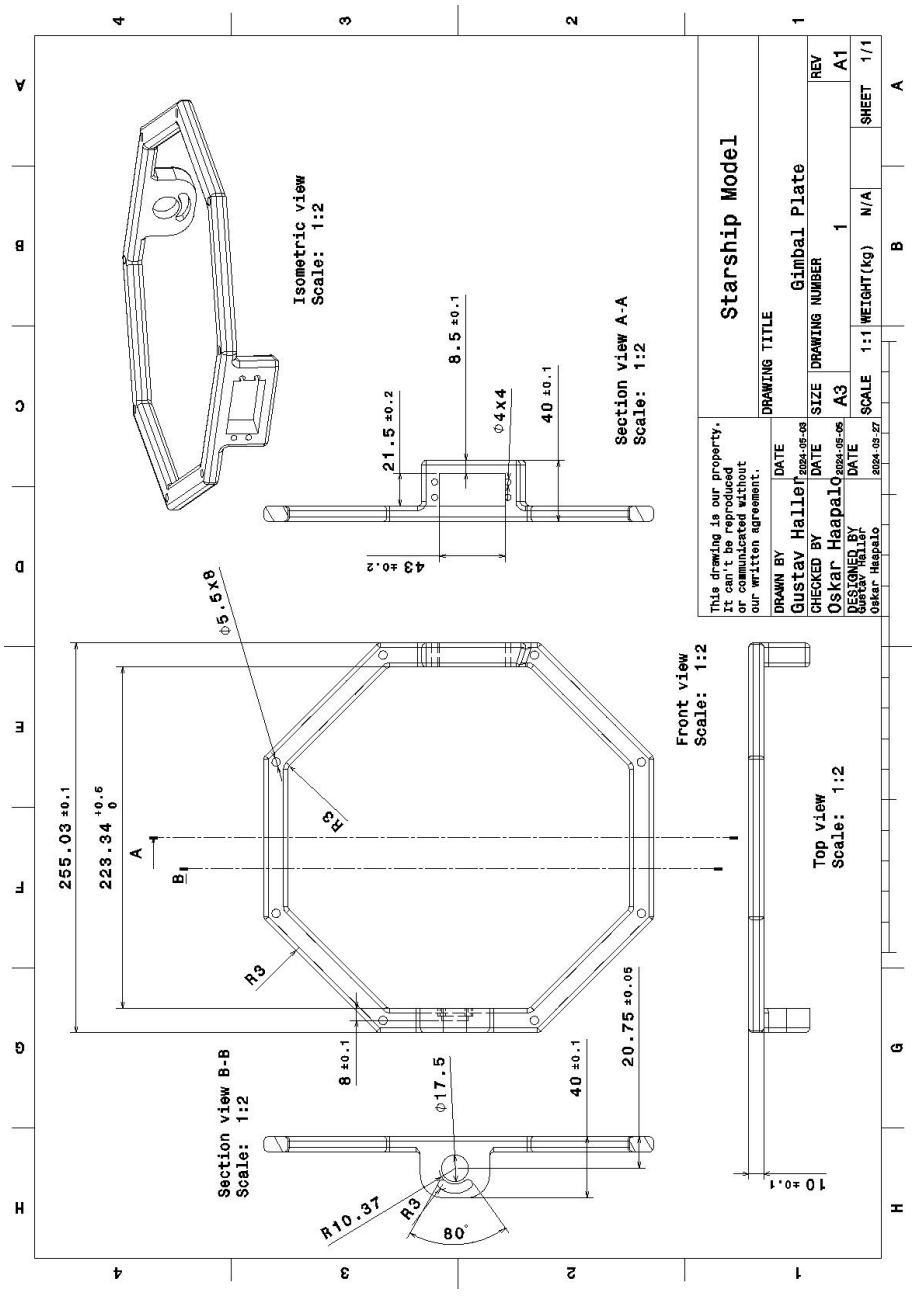


Figure A.1: Gimbal plate drawing

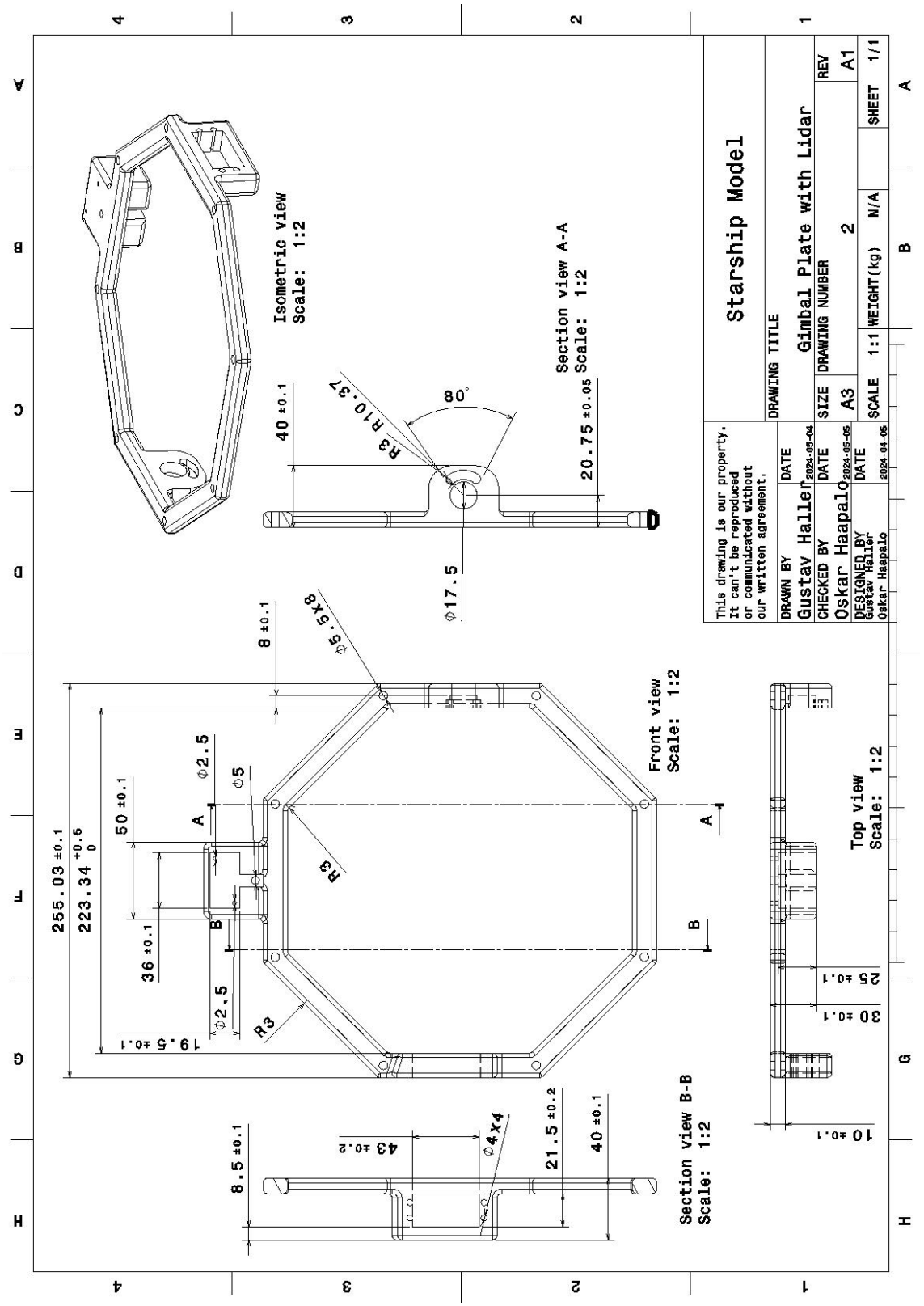


Figure A.2: Gimbal plate with lidar drawing

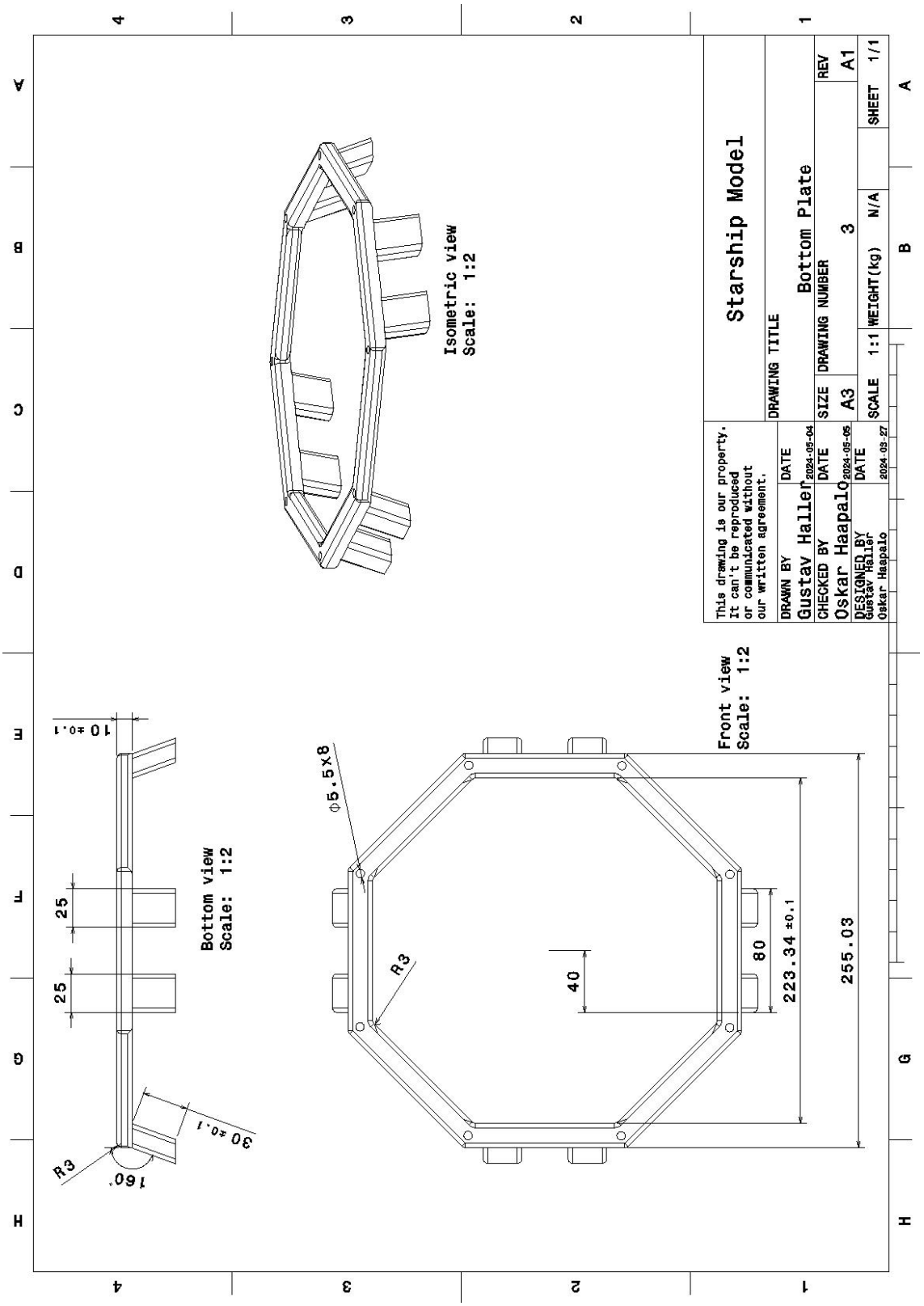


Figure A.3: Bottom plate drawing

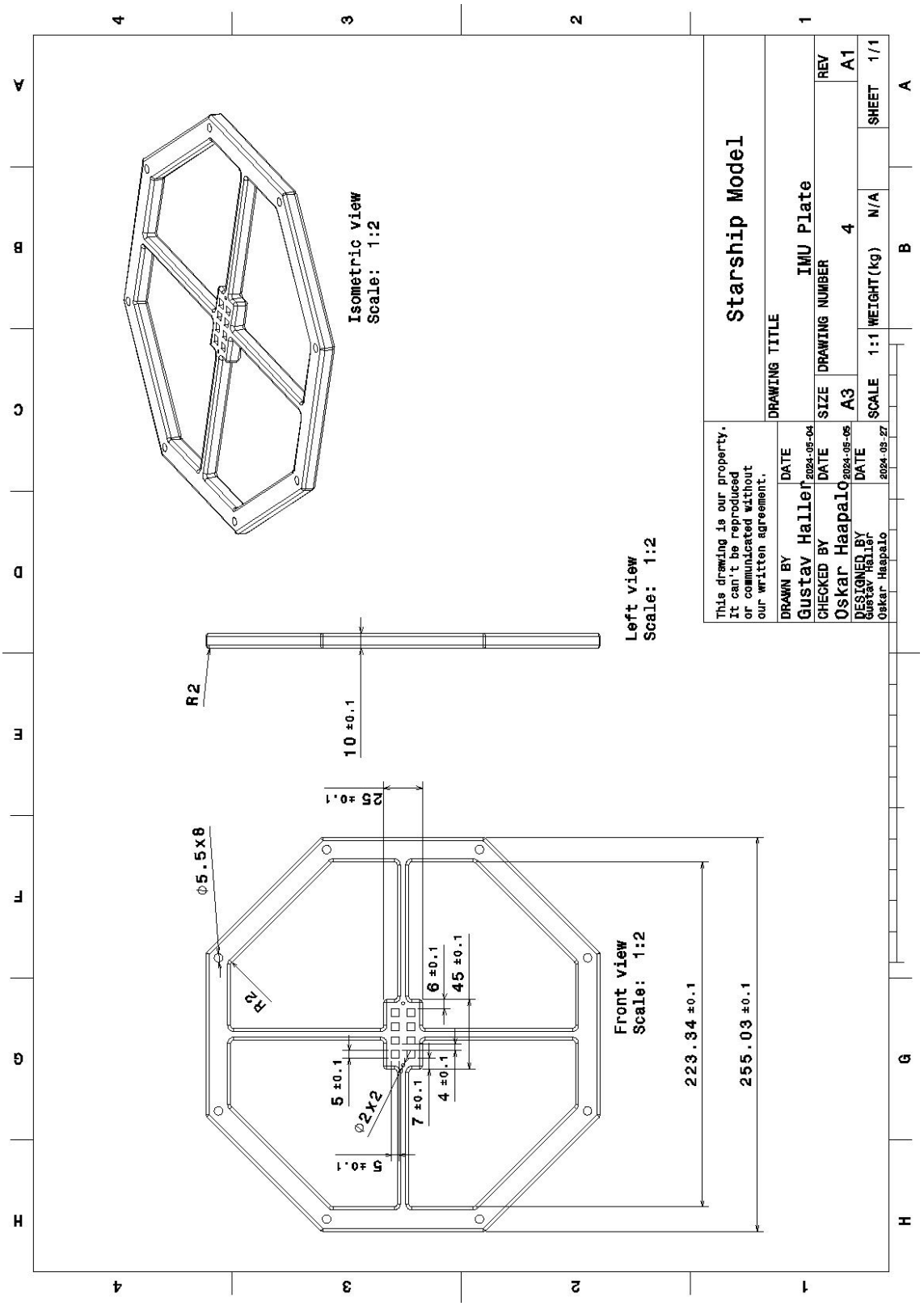


Figure A.4: IMU plate drawing

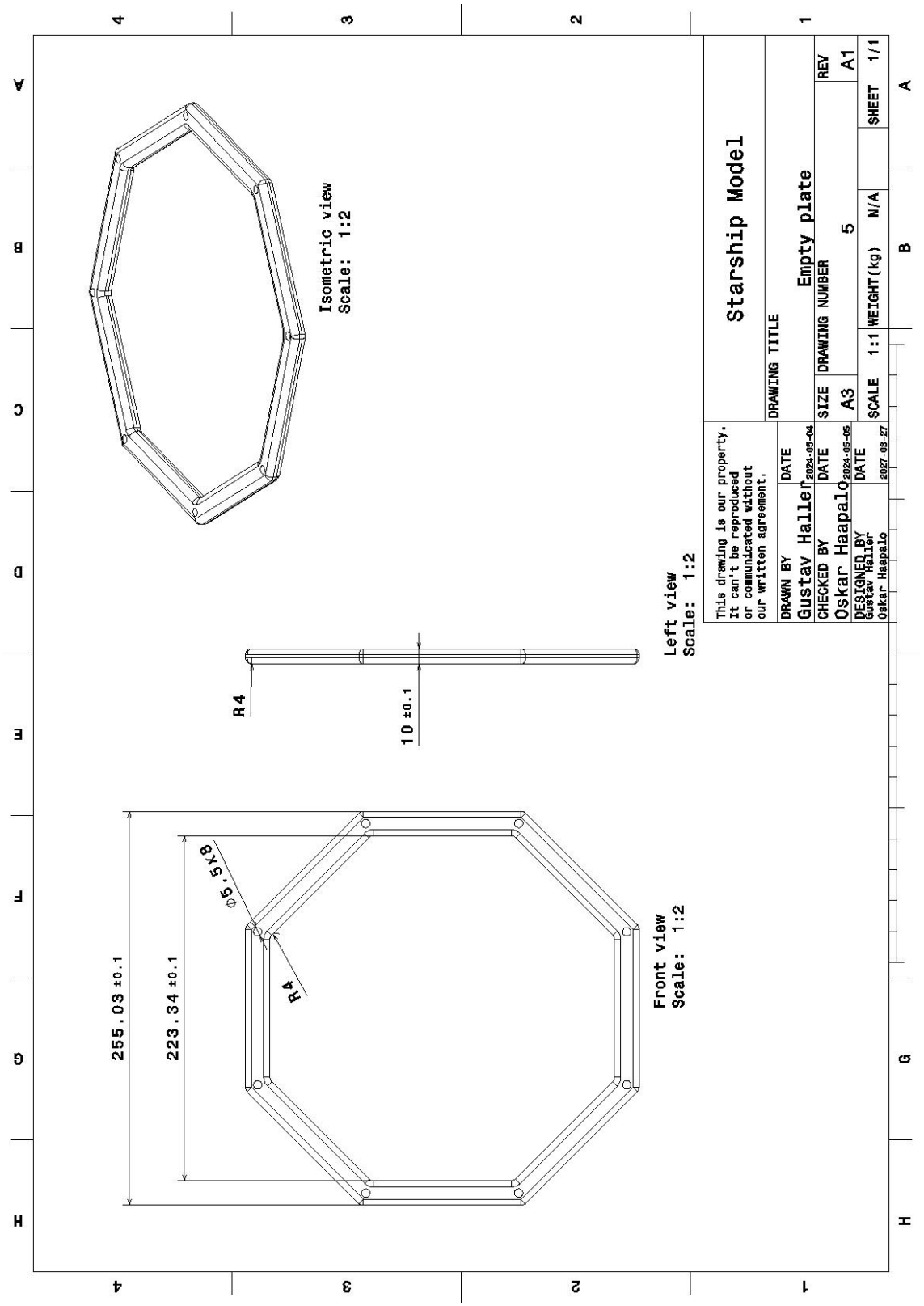


Figure A.5: Empty plate drawing

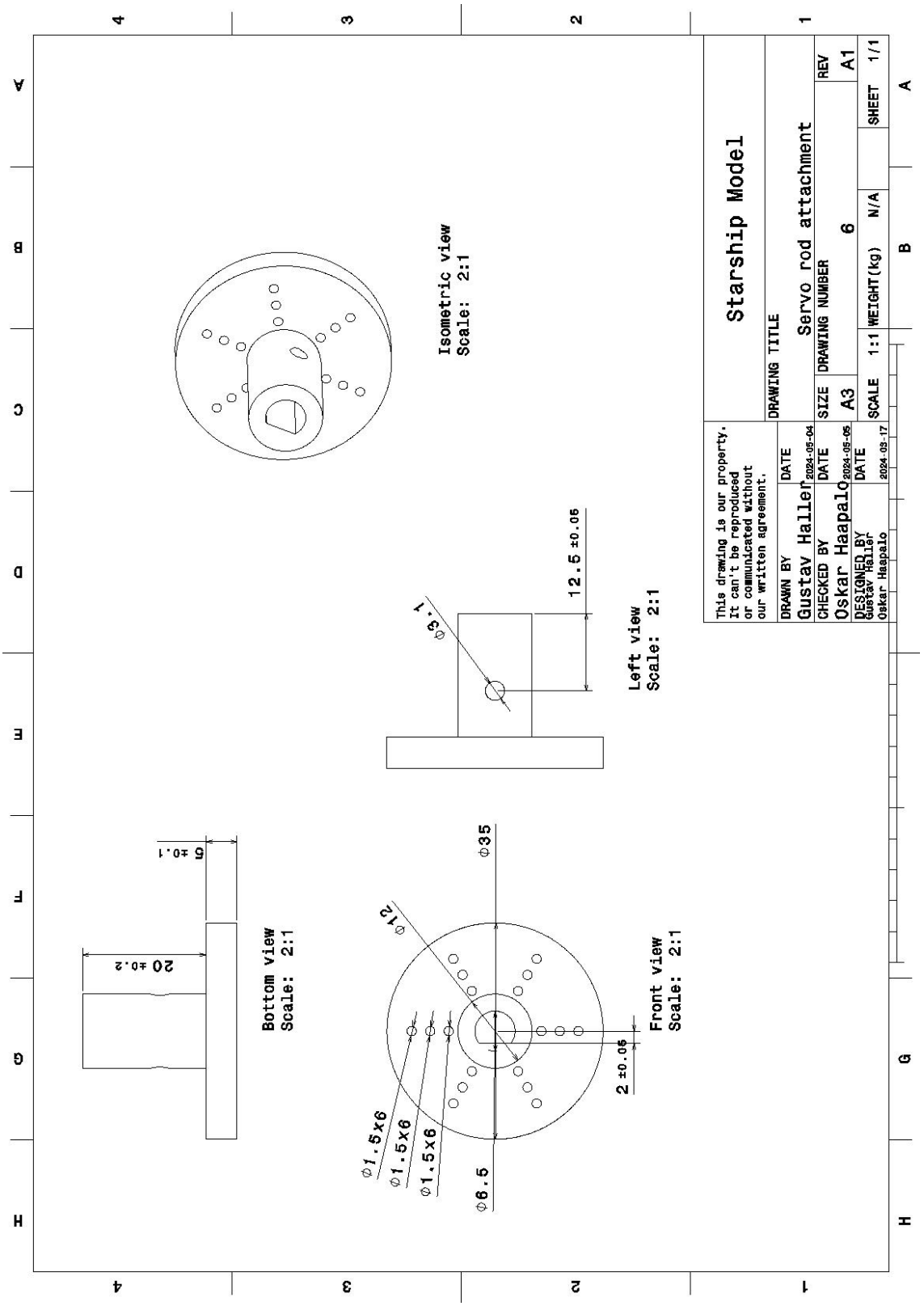


Figure A.6: Servo rod attachment drawing

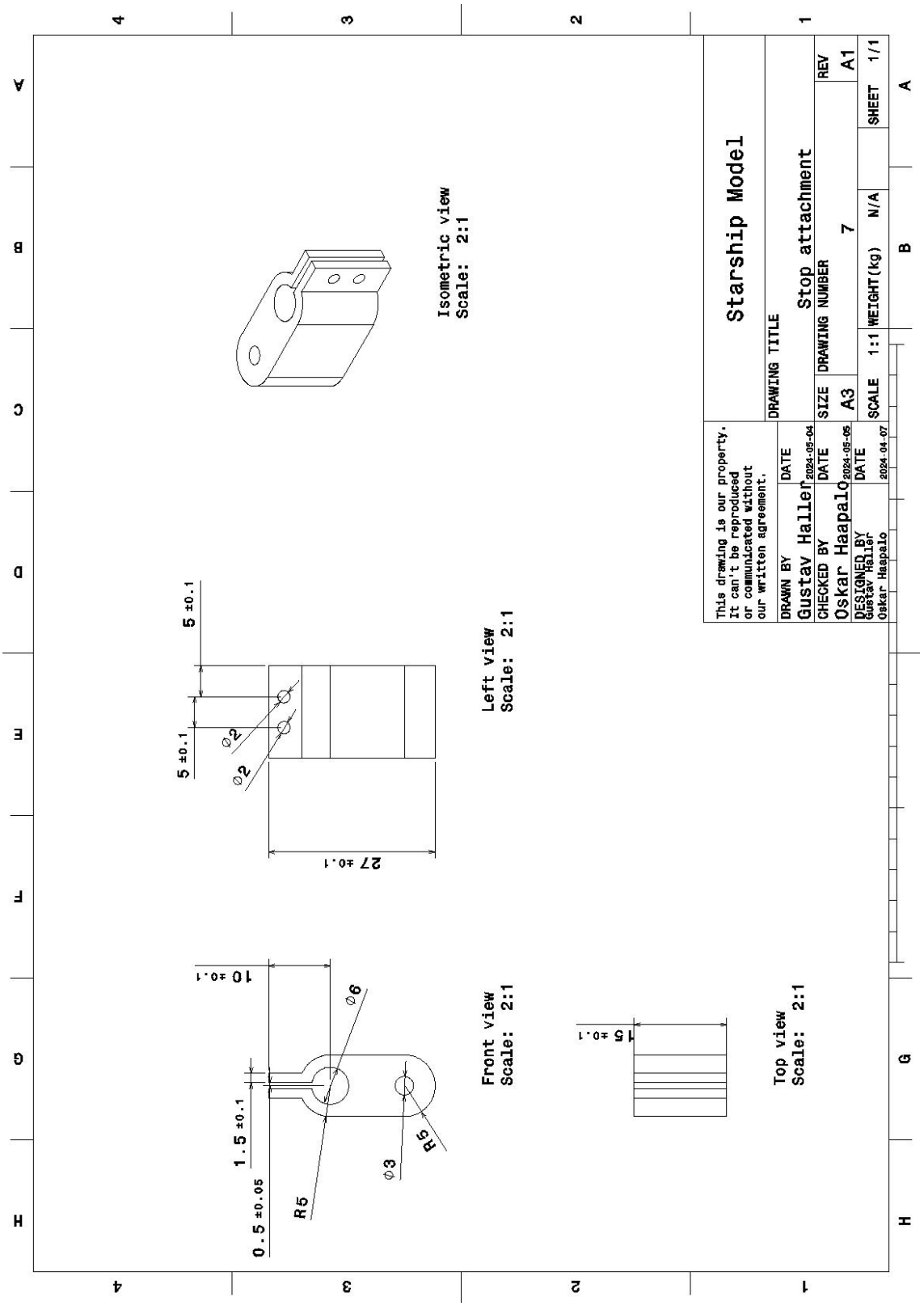


Figure A.7: Stop attachment drawing

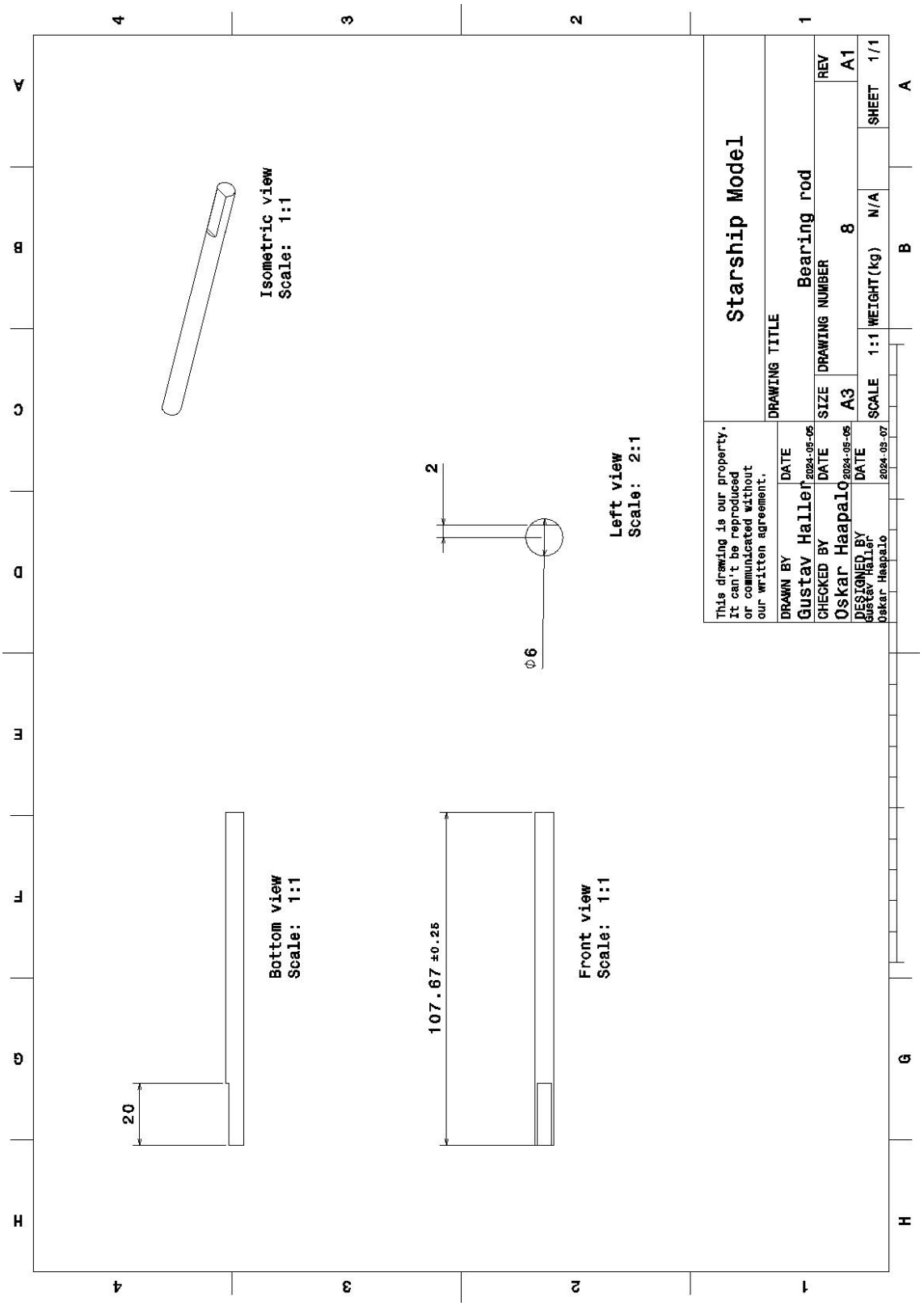


Figure A.8: Bearing rod drawing

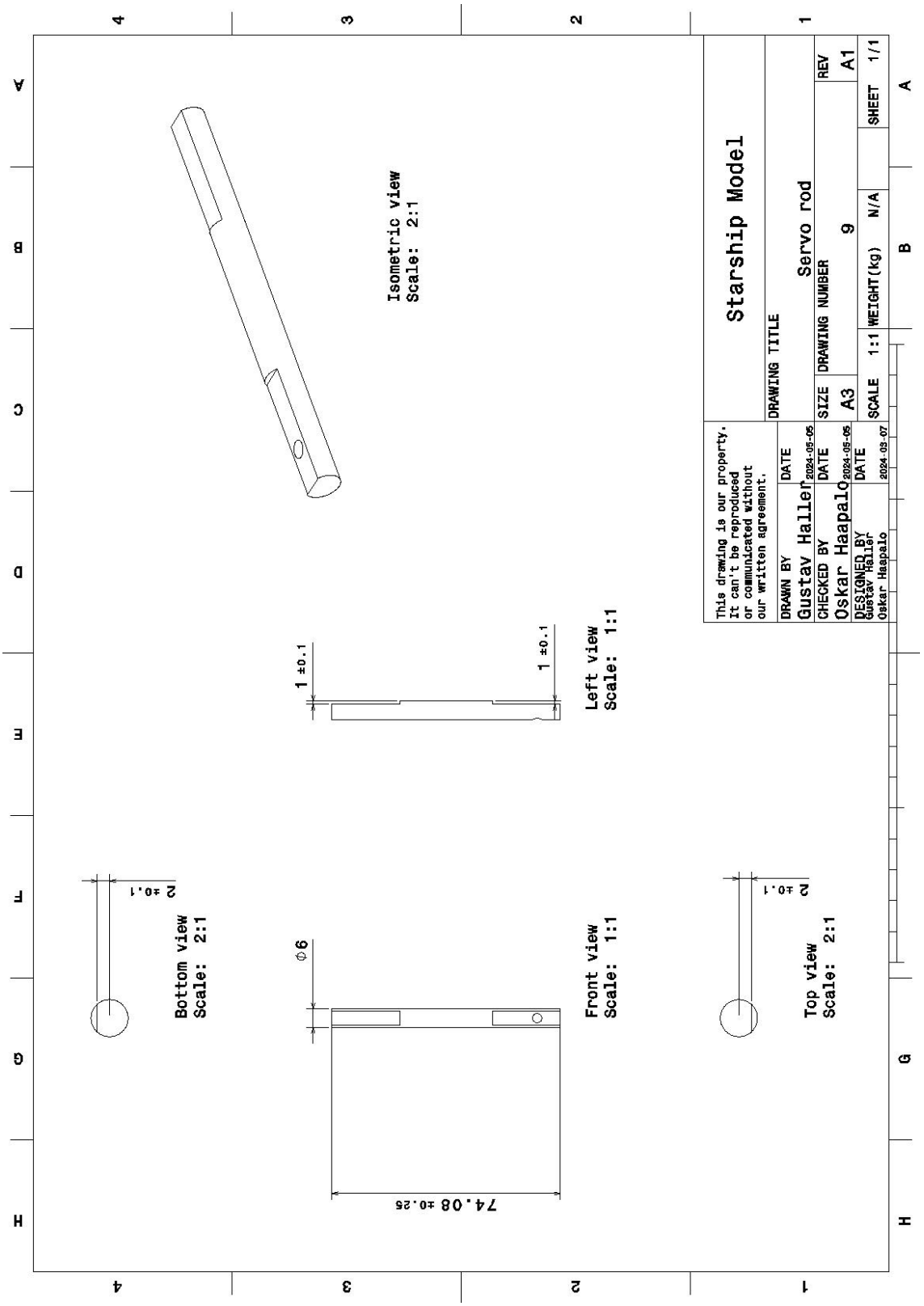


Figure A.9: Servo rod drawing

B

Component list

All the components used in the project are listed here:

- Microcontroller Teensy 4.1:
  https://www.pjrc.com/store/teensy41.html
- BLDC motor - T-motor V3008 1350KV:
  https://pyrodrone.com/products/t-motor-velox-v3008-cinematic-fpv-drone-motor-1350kv
- ESC - Hobbywing FlyFun V5 120A 3-8s:
  https://www.elefun.se/p/prod.aspx?v=35885
- Propeller:
  https://www.drone-fpv-racer.com/en/gemfan-8040-3-glass-fiber-nylon-for-cinelifter-macro-quad-9981.html
- 3S LiPo Battery 1550mAh 100C Tattu FunFly:
  https://www.drone-fpv-racer.com/en/tattu-funfly-lipo-battery-3s-1550mah-100c-5540.html
- 2S LiPo Batteri 600mAh 7.4V:
  https://rcdelar.se/sv/2s-lipo-batteri-74v-600mah-25c-jst-rcy-bec-vapex-lp0005f-21754
- IMU (BMI088) - Seed studio 6DOF IMU:
  https://www.mouser.se/ProductDetail/Seeed-Studio/101020584?qs=0lSvoLzn4L9bfYhcxsaCqA%3D%3D
- Buck - LM2596:
  https://www.ti.com/lit/ds/symlink/lm2596.pdf
- Gimbal servos - Modr:
  https://www.hobbex.se/el-elektronik/servo/pdi-6208mg.html
- Transceiver - nRF24L01+PA+LNA:
  https://www.amazon.se/DollaTek-NRF24L01-transceiver-modul-antistatisk-skumantenn/dp/B07PQPFTW

- Transceiver adapter 5V:
  https://www.amazon.se/AZDelivery-NRF24L01-adapter-s%C3%A4ndtagarmodul-inklusive/dp/B086V1YTTJ/ref=sr_1_5?crid=2PCEX4BOIQ2OY&dib=eyJ2IjoiMSJ9.e3s_G5BIrlgWXo0SZZgl5SUkbiggq9PITHTx8Zqv8KIUN6tSM5vmOQgKe2TSyYqc5L_kRzT01_WkApMwrU1EIzyxkJ6EwIZvCYOxRLUFibALC53ouJC2ysIVbljcGnK0d6xqBH9iK1FQASIusnU_FcQMwnNYTY9k5rTJUv4YZ7pqUbi9s46xnwrsQ1OOixXdTM2GqlinZGIsJZJhjfOIxXZXWS5iV7WPXgkPCu_0uMSeddJ84ptadS4GvhDVXFbMZChpwV5LyAhNpJohm9yeK-DfH-CemjA_v0BVduKt9uM.GpYWfhIZftxnd5b-1uxa7rnxlFN1J2J7i-bBIRJV11g&dib_tag=se&keywords=nrf24l01%2B%2B%2Badapter&qid=1710285152&sprefix=nrf24l01%2Badapter%2Caps%2C70&sr=8-5&th=1

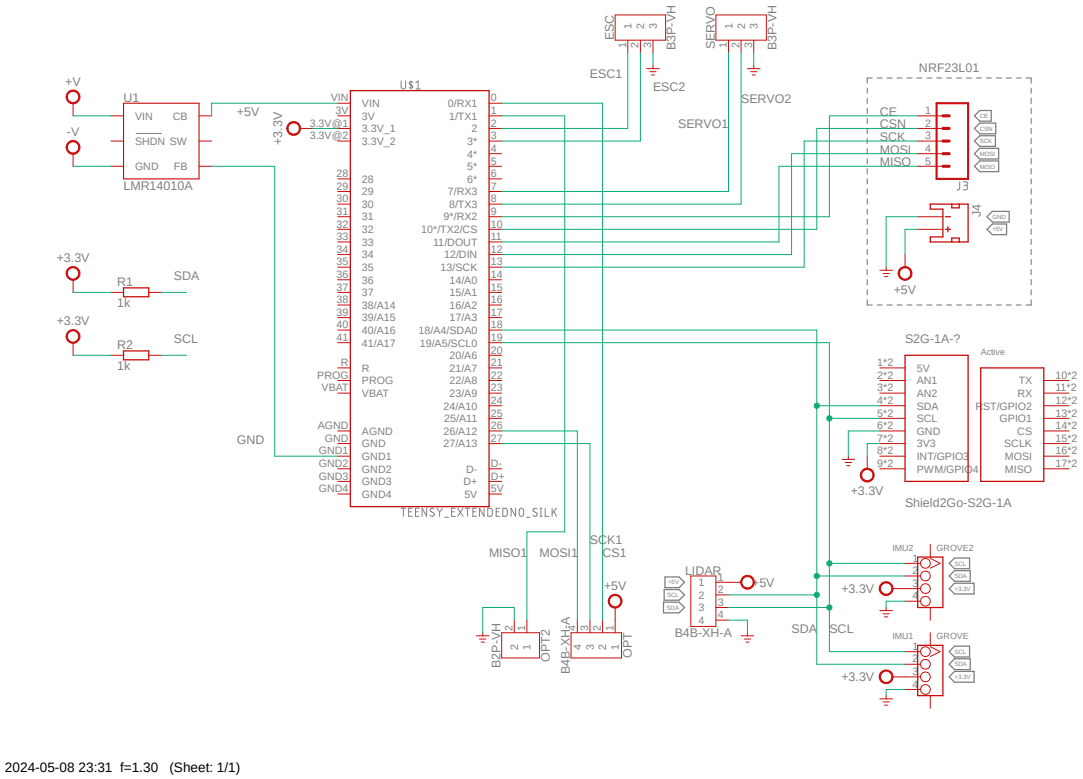
- Lidar - TF mini I2C:
  https://www.mouser.se/ProductDetail/Benewake/TFmini-Plus-I2C?qs=DPoM0jnrROWP3E1Fkud5uw%3D%3D

- Optical flow (PIM453):
  https://www.mouser.se/ProductDetail/Pimoroni/PIM453?qs=PzGy0jfpSMuJnlsymXyulA%3D%3D

C

Circuit diagram

This shows a detailed diagram of the circuit used in the project.



2024-05-08 23:31 f=1.30 (Sheet: 1/1)

Figure C.1: Circuit diagram for the rocket

D

PID control

Here the PID controller used for the tests is presented and explained.
The height control will be the same as in [1] since the rocket will closely mirror each other in regards to height adjustments. The height controller is given by the following feedback loop:

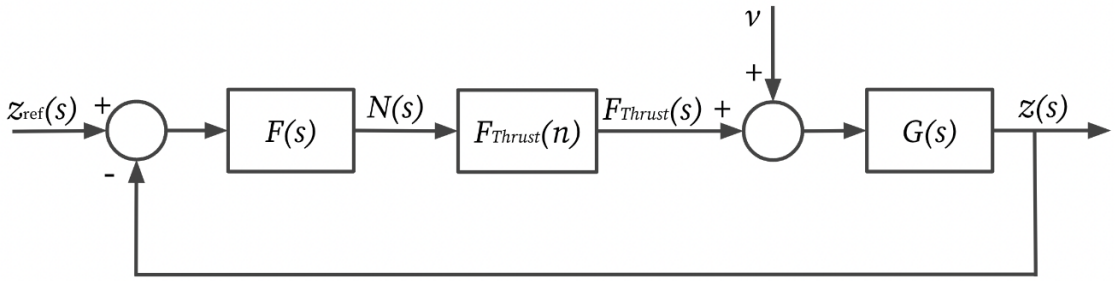


Figure D.1: Feedback system for altitude regulation[1], reprinted with permission

where z is the height of the rocket and $F_{Thrust}(n)$ is the relation between the rpm of the motors and lift force. F(s) are the gain values for the PID controller K_p, K_i/s and K_ds. The transfer function[1] is given by:

$$G(s) = \frac{Z(s)}{F_{Thrust}(s)} = \frac{1}{ms^2} \tag{D.1}$$

For stabilizing around its c.o.m the PID from [1] will be used, this model is based upon a model with thrust gimbaling [19], therefore it should function adequately for this model, potentially with superior performance. The only parameters that has to be changed is the height from which the motor is operating from the center of gravity. The following feedback system is given by [1]:

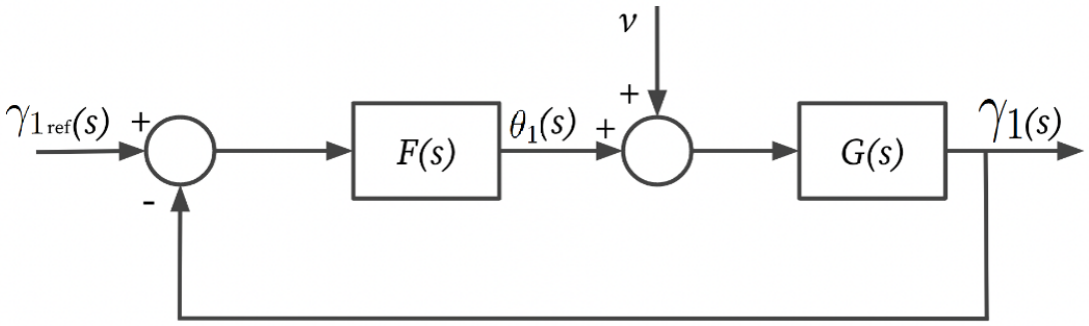


Figure D.2: Feedback system for angular regulation[1], adapted with permission

where F(s) is the gain function from the PID controller and G(s) is given by[19]:

$$G(s) = \frac{\gamma_1}{\theta_1} = \frac{h_1 * F}{\overline{I}_x * s^2} \tag{D.2}$$

E

Roll control

Here a P controller to control the rockets roll parameter is presented.

In Fig. E.1 a block scheme for a P controller is shown. It works by giving a reference signal r that is the desired roll angle. The reference is then subtracted by the output to create the error. The controller, *F(s)*, is the P controller that based on the error adjust the pwm signal for one of the motors, *G(s)*, to control the torque resulting in a new roll angle. Since the pwm singal is only calculating the diffrence in rpm from the other motor, the base line pwm signal needs to be added after the error is multiplied with the gain. The baseline pmw signal is represented as v.

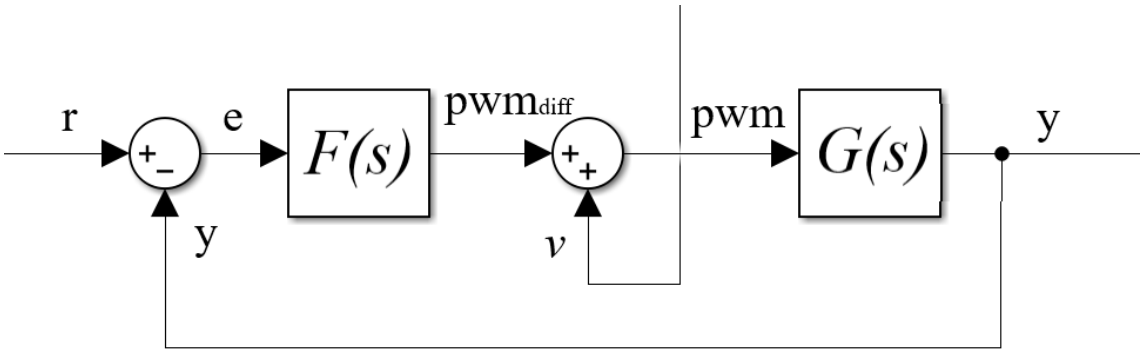


Figure E.1: Feedback system for roll control.

This in practice means that the LQR controls the thrust of one of the motors and this P controller controls the other. Note that the P controller F(s) output needs to be constrained in order not to differ to much from the other motors pwm signal. This is necessary to reduce the change in thrust that can occur with this control strategy.

F

Belly flop

In this appendix, the belly flop maneuver is described in detail.

The core idea of the belly flop maneuver is to increase the drag for the rocket when it is acceding, this is further explained below along with a Fig. F.1. This maneuver makes it so the rocket requires less thrust to decelerate and land, making the model more fuel-efficient. To make this principle apply more efficiently for the current rocket design a hull is needed.

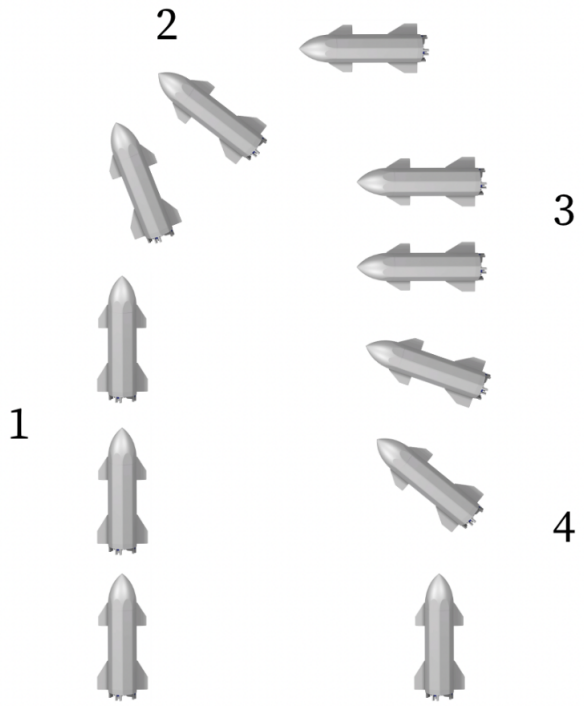


Figure F.1: Steps in a Belly flop maneuver

SpaceX have found a way to use rudimentary physics and state of the art automatic control engineering to bring their rockets back to the same spot they launched. The belly flop maneuver can be divided into four steps as seen in Fig. F.1.

1. The rocket is launched into space or coming back from outer orbit perpendicular to earth

2. The Rocket, autonomously, rotates itself, at a pre-defined height, horizontal towards earth
3. The rotation induces a larger amount of drag, slowing and cooling the rocket, consequently preventing it from burning up while entering the atmosphere contrary to conventional rockets.
4. At a given height away form the landing zone the rocket once again rotates itself perpendicular to earth and proceeds to execute a landing protocol

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS