CHALMERS

Starship Model 2025

Embedded System and Simulation for an Electrically Powered Reusable Rocket Model

Bachelor Thesis in Systems and Control

Oskar Gustafson, Sava Hama Aga, Gillis Magnusson, Eric Ringström, Fredrik Sandén, Sixten Thurfjell

DEPARTMENT OF ELECTRICAL ENGINEERING
Division of Systems and Control

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Starship Model
2025
Bachelor Thesis

Oskar Gustafson
Sava Hama Aga
Gillis Magnusson
Eric Ringström
Fredrik Sandén
Sixten Thurfjell

13th May 2025

Bachelor Thesis at the Division of Systems and Control
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
13th May 2025

CHALMERS
UNIVERSITY OF TECHNOLOGY

Starship Model 2025
Embedded System and Simulation for an Electrically Powered Reusable Rocket Model

OSKAR GUSTAFSON
SAVA HAMA AGA
GILLIS MAGNUSSON
ERIC RINGSTRÖM
FREDRIK SANDÉN
SIXTEN THURFJELL

© OSKAR GUSTAFSON, SAVA HAMA AGA, GILLIS MAGNUSSON, ERIC RINGSTRÖM, FREDRIK SANDÉN, SIXTEN THURFJELL 2025

Supervisor: Full Prof. Jonas Sjöberg, Chalmers University of Technology
Examiner: Prof. Nikolce Murgovski, Chalmers University of Technology

Bachelor Thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover picture: The current version of the Starship model and the SIMULINK model of the Starship model.

Written in LaTeX
Gothenburg 2025

Abstract

The bachelor thesis describes the design, analysis, and simulation of an electrically powered reusable rocket model. The model is designed with two rotors mounted on tiltable arms, enabling lift and maneuverability. The simulation is performed in a close to realistic environment using MATLAB Simulink and Simscape. The primary focus is improving the modularity of the embedded system and simulating the model's dynamic behavior.

The contributions made to the development of the Starship model are outlined at the start, detailing both hardware and software advancements. Key contributions include the design and implementation of a collision-resistant battery mount, a ground power cable for test bench applications, and the integration of a kill switch with multiple control modes. A planar motion detection system (optical flow sensor), a restructuring of embedded software, and position and orientation estimation using sensor data (lidar, IMU and Optical flow) are also developed.

The results demonstrate an accurate Complementary Filter for orientation estimation and a short term precise dead reckoning position estimation based on lidar, IMU, and Optical flow data. The optical flow camera provides the best position precision on rigid surfaces by identifying reference points. The restructuring of the software code results in a cleaner and more maintainable environment, enabling the seamless integration of additional features.

Additionally, the Simulink model accurately simulates the system's behavior under applied forces. However, the results indicate that while short term stability is achieved, the design requires modification for long term stability due to physical limitations affecting its controllability.

Preface

This report presents the findings and outcomes of bachelor thesis group EENX16-VT25-14, the Starship model. The thesis was conducted at the department of Electrical Engineering at Chalmers University of Technology in the spring of 2025. This thesis is a further development of two previous bachelor theses [1] and [2] which were conducted in the spring of 2023 and 2024.

We extend our gratitude to our supervisor Prof. Jonas Sjöberg who has been of great help in completing this thesis. We also thank the group of 2024's thesis for the discussions and help with finding a way forward, along with the CASE-lab at Chalmers for providing a place to work, necessary tools and materials for the project. Last but not least we thank our examiner Prof. Nikolce Murgovski.

Oskar Gustafson, Sava Hama Aga, Gillis Magnusson,
Eric Ringström, Fredrik Sandén and Sixten Thurfjell.
Gothenburg, May 2025

Explanation of Key Concepts

c.o.m - Center of mass (x,y,z), from the center at the bottom of the model.

Communication protocol - A set of rules describing how two or more entities transfer data between each other.

Complementary filter - A filter used to create an estimate from a combination of multiple measurements.

Dead reckoning - A method of estimating position based on the previous position and measured movement.

DOF - Degrees of freedom; the number of independent movements a system can make in space.

Gimbal - A mechanism allowing rotation about an axis.

IMU - Inertial measurement unit. A device measuring angular velocities and translational accelerations for three axes.

Kalman filter - An algorithm used to determine estimates of unknown variables, using measurements and mathematical models.

Madgwick filter – An orientation estimation algorithm that uses IMU data to compute orientation in quaternion form.

Kill switch - A mechanism allowing a user to switch off a system.

lidar - Light Detection and Ranging, a technology used to measure distance using lasers.

LiPo battery - Short for lithium-ion polymer battery. LiPo batteries consist of separate cells, #S, with # denoting the number of cells in the battery.

Optical flow sensor - A sensor that uses a camera facing down to measure translational velocity by tracking ground movement.

Sensor fusion - The process of combining data from different sensors to acquire more comprehensive and accurate information about a system.

SPI - Serial Peripheral Interface, standardized protocol for high speed communication with peripherals in an embedded system

Contents

1 Introduction **1**
- 1.1 Background . . . 1
  - 1.1.1 Starting Point of the Starship Model . . . 1
  - 1.1.2 Challenges in the Current Model . . . 2
- 1.2 Report Structure . . . 3

2 Contributions **4**

3 Technical Implementations **5**
- 3.1 Battery Mount for Collision Resistance . . . 5
- 3.2 Ground Power Cable for Test Bench Applications . . . 5
- 3.3 Kill Switch and Control Modes . . . 6
- 3.4 Planar Motion Detection Using Optical Flow Sensor . . . 7
  - 3.4.1 Hardware Implementation . . . 7
  - 3.4.2 Software Implementation . . . 8
- 3.5 Embedded Software Restructure . . . 8
- 3.6 Position and Orientation Estimation Using Sensor Data . . . 9
  - 3.6.1 Estimating Pitch and Yaw With Complementary Filter . . . 9
  - 3.6.2 Sensor Fused Dead Reckoning Position Estimation . . . 10
- 3.7 Control Simulation Using Simulink . . . 11
  - 3.7.1 Identifying the System Dynamics . . . 11
  - 3.7.2 Creating a State Space Model . . . 13
  - 3.7.3 Simulink Model . . . 15
  - 3.7.4 Thrust Vectoring . . . 16
  - 3.7.5 Trajectory and Attitude of Rocket . . . 16
  - 3.7.6 State Feedback . . . 16
  - 3.7.7 Linking Sensor Data to the State Vector . . . 17

4 Implementation Outcomes and Reflections **19**
- 4.1 Impact of Software Improvements for the Modularity of the System . . . 19
- 4.2 Results of the Ground Power Cable . . . 19
- 4.3 Sensor Improvements for Data Collection . . . 20
  - 4.3.1 Accuracy of Yaw and Pitch Orientation from IMU Data . . . 20
  - 4.3.2 Tuning of α Constant used in Pitch and Yaw Orientation . . . 22
  - 4.3.3 Accuracy of the Dead Reckoning Positioning . . . 22
  - 4.3.4 Tuning of Complementary Filter Constants used in Position Estimation . . . 24
- 4.4 Simulation . . . 26
  - 4.4.1 Limitations of Simulated Model . . . 26
  - 4.4.2 Testing and Verification of Forces and Motion . . . 26
- 4.5 Tuning the Control Algorithm . . . 32
- 4.6 Limitations of Construction . . . 32
  - 4.6.1 A Coupled Design . . . 33
  - 4.6.2 Belly Flop Maneuver . . . 34

5 Opportunities for Continued Exploration **35**
- 5.1 Hardware . . . 35
  - 5.1.1 Propulsion System . . . 35
  - 5.1.2 Aerodynamic Surfaces for Belly Flop Maneuver . . . 35
- 5.2 Control . . . 35
  - 5.2.1 System Identification . . . 35
  - 5.2.2 Working 3D Simulation . . . 35
  - 5.2.3 Replacing the IMU . . . 35

A Drawings **I**

1 Introduction

Automation plays a crucial role in advancing mobility solutions across various domains such as aerospace, automotive, and maritime transportation. Developing a control strategy for an autonomous system requires an understanding of mechanics, aerodynamics, system modeling, and feedback control. This makes vehicles with rocket characteristics an ideal platform for students and engineers to apply and test control theory.

Rockets represent one of the most challenging classes of dynamic systems to control due to their non-linear behavior, under actuation, and sensitivity to disturbances. *Starship* is an autonomous rocket engineered by SpaceX, which has the ability to land on its own with the purpose of being reusable. The rockets used previously in the industry were designed to burn up upon reentering the atmosphere, thus limiting the rocket to single use. The fundamental change of having the rocket being able to reenter the atmosphere and land eliminates this limitation. This technological advancement reduces costs [3].

Autonomy, control, and stability are key concepts to define the performance of a model or system. Autonomy is a concept that describes a system's ability to operate independently without human intervention. This includes operations such as takeoff, flight adjustments, and landing. The concept of control refers to the mechanisms and algorithms that actively guide the behavior of the system. The task of control is to ensure that the system follows the intended trajectory and responds to external disturbances. Stability relates to the system's ability to maintain or return to the desired state despite disturbances. While control involves actively adjusting inputs to reach a desired motion, stability ensures that the system does not deviate uncontrollably from the intended state.

1.1 Background

This bachelor thesis continues the development of an electrically propelled rocket, referred to as the Starship model. The work builds upon two previous bachelor theses completed in 2023 [1] and 2024 [2], which establishes the starting point of the model's structure and design. The term *Starship model* was introduced in the 2024 thesis and is used throughout this report to refer to the physical model of the rocket.

This thesis's main goal is to improve and evaluate the Starship model, focusing particularly on improving the modularity of the embedded system and simulating the model's dynamic behavior. The key changes include restructuring the onboard software for easier development, implementing new sensors for more accurate position and orientation estimation, while also building a simulation environment. The model uses two electrically powered, gimbaled engines with thrust vectoring to mimic the maneuverability of the full-scale SpaceX Starship, enabling both lift and attitude control for stable flight and landing [4].

1.1.1 Starting Point of the Starship Model

The results from the previous iterations of the Starship model are presented in this section, providing the starting point for this thesis. The foundation consists of the vehicle's frame and its electronic system. The mechanical hardware is demonstrated in Figure 1, and consists of eight rods arranged at the vertices of an octagon, connected with PLA plates. Two motors connected to propellers, are used to generate thrust. Each motor is connected to a servo allowing the motor to rotate about an axis. Additionally, two sensors, a lidar and an IMU, are used to measure position, acceleration as well as angular velocity of the Starship model. To perform computations and implement control, a microcontroller, Teensy 4.1, is used. The entire system is powered by LiPo batteries. For a more detailed description, see the 2024 thesis [2].

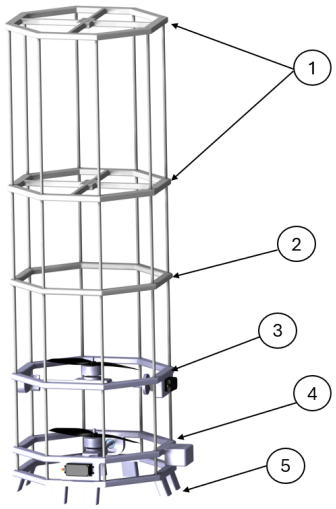


Figure 1: Previous design of the Starship model. The structure is still relevant but slightly modified. See 2024 thesis [2]. Reprinted with permission

Table 1: Component list [2], reprinted with permission

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

1.1.2 Challenges in the Current Model

The developed Starship model behaves as an inverted pendulum with six degrees of freedom (see Figure 2), wherein the center of mass is positioned above the point of thrust generation. This configuration mimics the dynamics of the real SpaceX Starship and leads to inherent instability.

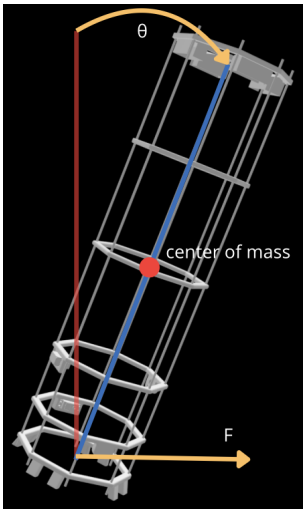


Figure 2: Visualization of how the Starship model acts as an inverted pendulum

Due to the high center of gravity, the system requires continuous and precise adjustments through thrust vectoring to maintain upright equilibrium. This introduces significant challenges in system stabilization, as rapid deviations from the vertical axis have the potential to escalate rapidly without corrective input. Consequently, high frequency and low latency sensor feedback is essential to enable the control system to react in real time. The control architecture must therefore be robust and tightly coupled with fast response sensor data to manage this inherently unstable system.

Furthermore, the unstable nature of the Starship model necessitates the integration of comprehensive safety mechanisms during development and testing. These safety features are crucial to prevent damage to equipment and ensure operator safety, especially when control errors or hardware failures occur.

1.2 Report Structure

This section outlines the structure of the bachelor thesis and briefly describes the content of each main part.

- **Section 2 – Contributions:** This section presents the technical contributions made to the Starship model.
- **Section 3 – Technical Implementation:** This part details the methodologies used to realize the contributions. The section begins with the hardware and software additions, followed by an explanation of the approach used to derive the state space model, how sensor data is mapped to the system states and the construction of the corresponding Simulink model.
- **Section 4 – Implementation and Results:** This section presents the outcomes of the implemented solutions. It starts by evaluating software performance, followed by the simulation results from the Simulink model. Finally, it discusses how certain physical limitations of the Starship model affect its controllability.
- **Section 5 – Continued Exploration:** The final section discusses possible improvements and future work that could further improve system performance and model fidelity.

2 Contributions

The following list shows the contributions with their corresponding implementation domain, in relation to the challenges presented in Section 1.1.2. All code related is available on Github[5].

- Battery Mount for Collision Resistance [Hardware]
- Ground Power Cable for Test Bench Applications [Hardware]
- Kill Switch and Control Modes [Hardware, Software]
- Planar Motion Detection Using Optical Flow Sensor [Hardware, Software]
- Embedded Software Restructure [Software]
- Position and Orientation Estimation Using Sensor Data [Software]
- Control Simulation Using Simulink [Simulation]

3 Technical Implementations

This section presents how the contributions for the Starship model are implemented.

3.1 Battery Mount for Collision Resistance

In the 2024 version of the Starship model the batteries were located on the top of the rocket and secured in place with velcro straps. As a result of this configuration the batteries would take the initial impact in the case of a nosedive crash. This is a safety concern given that damage to LiPo batteries poses a risk of fires or explosions [6].

To prevent the potential danger with unprotected LiPo 3S batteries a 3D printed protective mount is implemented on the top section of the Starship model, as shown in Figure 3. This is achieved by editing the existing top plate design in CAD, adding four shells to fit the batteries for protection during flight. Detailed drawing is found in A.4.

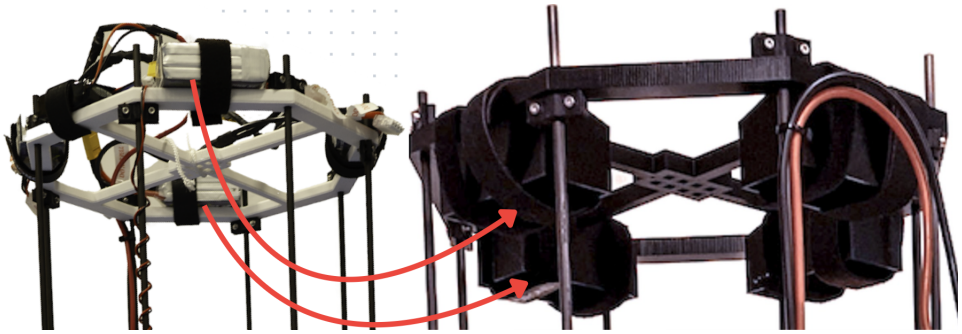


Figure 3: A battery mount is added to the Starship model in order to protect the previously exposed LiPo 3S batteries. The model on the left is the old setup with unsheltered batteries (reprinted with permission) and the model on the right is the current setup with protective mounts.

Another battery mount for the LiPo 2S is implemented on the top plate. This battery powers the electronics and was previously unshielded. This mount is designed in CAD from scratch to fit in a modular fashion for mounting on the top plate. The mount is fitted with screws and nuts on top of the plate in the middle with a shell to hold the battery. Detailed drawing is found in A.3.

3.2 Ground Power Cable for Test Bench Applications

To facilitate sensor functionality testing without relying on battery power, a ground power cable is installed. This allows the system to be powered externally, eliminating the need to remove, recharge, and reinstall the LiPo 2S battery between tests, this cable is demonstrated in Figure 4. During flight, the LiPo 2S battery supplies power to the sensors and microcontroller, but it is rapidly depleted during extended ground testing, making an external power source both practical and time efficient.

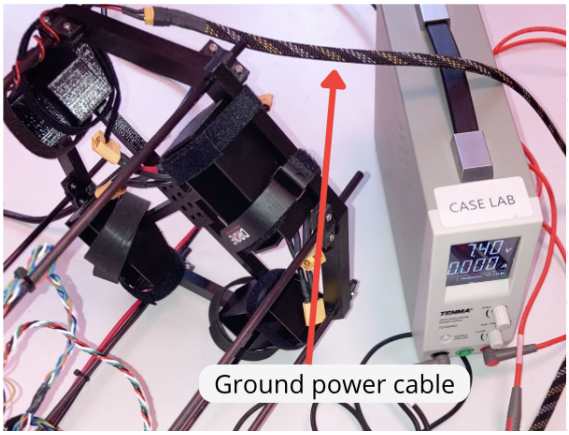


Figure 4: Ground power cable implementation for using ground power to the electronic board when performing sensor tests

3.3 Kill Switch and Control Modes

To address the safety concerns mentioned in Section 1.1.2, a kill switch is implemented to immediately shut down the Starship model in case of an emergency. Additionally, a manual control mode is added, allowing the system to be operated without active control algorithms for testing and debugging purposes. These features are implemented with a WFR09S receiver connected to a WFT09II transmitter on a radio controller [7].

The WFR09S receiver (seen in Figure 5) is connected to an interrupt vector similar to the Optical flow sensor 3.4. The code inside the interrupt handler has two flags that activate with a specific switch on the radio controller, an emergency stop flag and a manual mode flag. The emergency flag also activates if the radio controller is turned off. When the emergency stop flag is toggled the Starship model's motors will stop and all other functionality will pause until the emergency stop is deactivated. When the manual mode flag is toggled the autonomous control system is shut down and the user has the ability to control the thrust of the motors and gimbal angles of the servos with the radio controller.

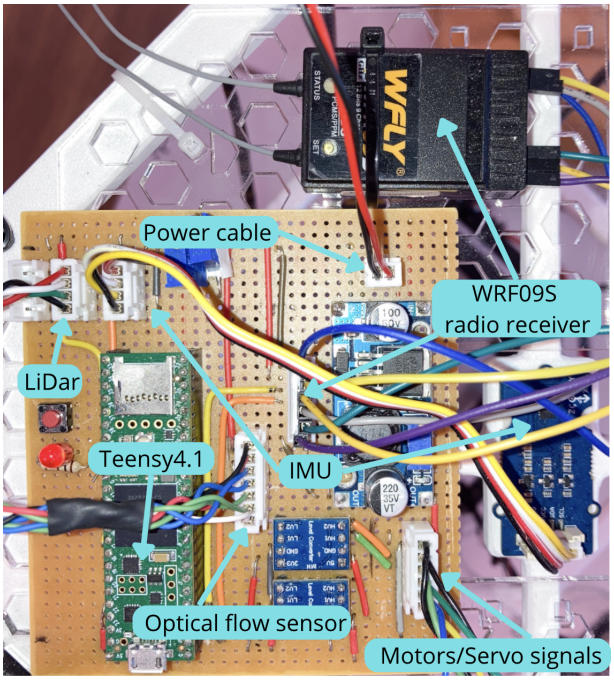


Figure 5: The electronic board connecting the sensors, motors and servos to the Teensy 4.1 microcontroller

3.4 Planar Motion Detection Using Optical Flow Sensor

In order to observe planar motion of the Starship model for position estimation, an optical flow sensor [8] added, which tracks motion in along the local X- and Y-axes $(\dot{x}, \dot{y})$, shown in Figure 9.

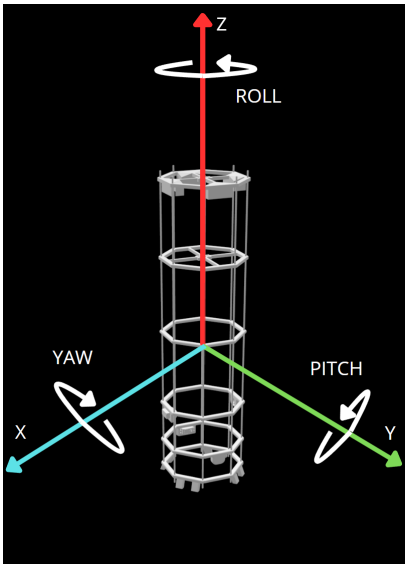


Figure 6: Local X, Y, Z coordinates and yaw, pitch and roll.

3.4.1 Hardware Implementation

The optical flow sensor is aligned with the rocket's Z-axis, pointing the negative direction (see Figure 7), like to the lidar sensor mentioned in 1.1.1. This is due to the motion tracking that comes from the camera observing the moving ground to estimate velocity. Ground with prom-

inent features produces more accurate estimations than one coloured smooth ground without detail.

As part of the integration process for the optical flow sensor on the rocket, a custom mount is designed in CAD and 3D printed. The mount features a modular design that fits onto the outer ring of the bottom plate. The mount consists of two parts, one with the space for the sensor to be fitted in, and one back plate for it to fit around the ring. These two parts are fastened using bolts and nuts. Detailed drawings are found in A.1 and A.2.

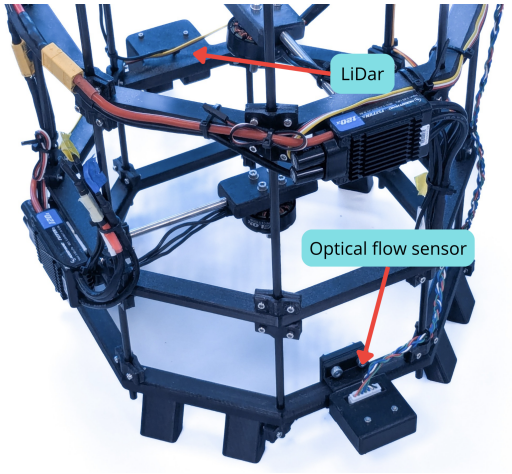


Figure 7: Mounted optical flow sensor at the bottom of the Starship model on the opposite side to the mounted lidar sensor.

3.4.2 Software Implementation

The optical flow sensor is connected to the Teensy 4.1 microcontroller. Communication between the microcontroller and the sensor is achieved through the SPI (Serial Peripheral Interface) protocol using the open source library *Bitcraze PMW3901*, available on GitHub [9].

An interrupt vector is connected to the Optical flow sensor for faster data collection. Every time the sensor collects data this interrupt vector is activated. When that happens an interrupt handler or interrupt service routine (ISR) connected to the interrupt vector begins running. The interrupt handler will temporarily pause the main code from being executed and runs the code in the handler instead [10].

3.5 Embedded Software Restructure

Accurate sensor data is an important aspect for the Starship model to operate autonomously, thus considerations have to be made for the code implementation. Creating a minimal, efficient and maintainable software solution is key to minimize the margin of human error when debugging and tuning parameters during testing.

The codebase from the 2024 version [2] is considered complex and difficult to interpret. To address this, major structural changes are implemented with the aim of improving readability and simplifying analysis, particularly when tuning filters and minimizing sensor noise. Therefore a modular approach is adopted, separating code related to individual components such as sensors, servos, and motors into dedicated files acting as libraries. These libraries are then accessed from the main file through clean and minimal function calls. This refactoring reduced the length and complexity of the main control script, making the code easier to debug and lowering the risk of unintentionally modifying critical logic during test flights and parameter tuning.

In the following Figure 8 a flow chart of the code structure is shown from the sensor data being received from the optical flow sensor, the IMU and the lidar to the calculated thrust speed

and gimbal angles going to the motors and servos. Each blue module is implemented in a separate library to maintain a structured code base. Depending on their respective roles, the libraries are either accessed by the main file or another library. If there is a need to tune the parameters of for example the lidar, one only has to access the lidar file instead of navigating a large codebase in main.

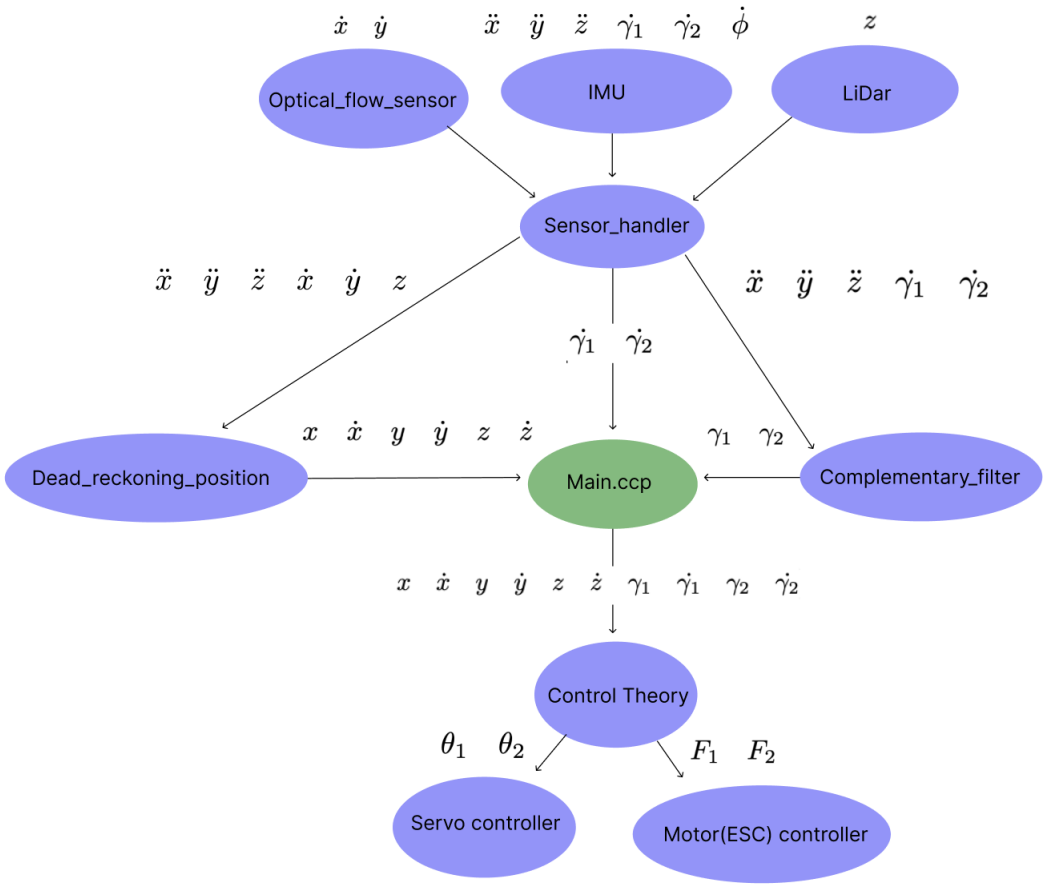


Figure 8: Flowchart over new file structure

3.6 Position and Orientation Estimation Using Sensor Data

This section presents the methods used to derive position and orientering estimates by fusing data from onboard sensors, including the IMU, lidar, and optical flow sensor.

3.6.1 Estimating Pitch and Yaw With Complementary Filter

To improve accuracy and reduce drift for the Starship's 6-DOF BMI088 IMU [11], sensor fusion is used. This is due to three reasons. First, over time the gyro slowly drifts more and more which must be compensated for. Second, the accelerometer is noisy and susceptible to interference from Starship model's vibrations. Third, with the help of sensor fusion it is possible to estimate quantities that are not directly measurable like pitch and yaw (γ_1 and γ_2).

To address these challenges a Complementary filter is implemented to fuse the gyro and accelerometer. A Complementary filter is a simple and effective filter that combines the short-term accuracy of the gyro with the long-term stability of the accelerometer [12] to calculate pitch and yaw. The Complementary filter works as a low-pass filter on the accelerometer and as a high-pass filter on the gyro. The calculations performed by the filter to achieve this is presented

in Equations 3.1 - 3.3.

$$\gamma_1 = \text{pitch}, \quad \gamma_2 = \text{yaw}$$

$$\ddot{x}, \ddot{y} \text{ and } \ddot{z} = \text{accelerometer readings}$$

$$\dot{\gamma_1}, \dot{\gamma_2} \text{ and } \dot{\phi} = \text{gyro readings}$$

$$\gamma_{1\text{acc}} = \arctan\left(\frac{\ddot{y}}{\sqrt{\ddot{x}^2 + \ddot{z}^2}}\right), \quad \gamma_{2\text{acc}} = \arctan\left(\frac{-\ddot{x}}{\ddot{z}}\right) \tag{3.1}$$

$$\gamma_{1\text{gyro}} = \gamma_1 + \dot{\gamma_1} \cdot dt, \qquad \gamma_{2\text{gyro}} = \gamma_2 + \dot{\gamma_2} \cdot dt \tag{3.2}$$

$$\gamma_1 = \alpha \cdot \gamma_{1\text{gyro}} + (1-\alpha) \cdot \gamma_{1\text{acc}}, \qquad \gamma_2 = \alpha \cdot \gamma_{2\text{gyro}} + (1-\alpha) \cdot \gamma_{2\text{acc}} \tag{3.3}$$

How much weight is given to the gyro ($\dot{\gamma_1}, \dot{\gamma_2}$ and $\dot{\phi}$) or accelerometer ($\ddot{x}, \ddot{y}$ and $\ddot{z}$) readings is decided by an α (Equation 3.3) constant which contains a percentage (0.0 to 1.0). If the $\alpha = 0.98$, the Complementary filter takes 98% of the estimation from the gyro readings and 2% of the accelerometer readings. The α constant is decided in trial and error testing presented in Section 4.3.2.

3.6.2 Sensor Fused Dead Reckoning Position Estimation

To estimate the Starship model's position in the x, y, and z directions and the $\dot{x}, \dot{y}$ and $\dot{z}$ velocities, a sensor-fused dead reckoning method is employed. This method combines data from the IMU, Optical flow sensor and lidar, to improve accuracy and minimize drift. For reference, the coordinate system is illustrated in Figure 6.

3.6.2.1 Horizontal position and velocity ($x, y, \dot{x}, \dot{y}$)

To estimate $x, y, \dot{x}$ and $\dot{y}$, a Complementary filter is used with numerically integrated $\ddot{x}$ and $\ddot{y}$ from the IMU's accelerometer and $\dot{x}$ and $\dot{y}$ from the Optical flow. The equation of the numerical integration of the IMU accelerometer is presented in Equation 3.4.

$$\ddot{x}_{\text{imu}}, \ddot{y}_{\text{imu}} = \text{IMU accelerometer horizontal readings}$$

$$\dot{x}(t)_{\text{imu}} = \dot{x}(t - \Delta t) + \ddot{x}_{\text{imu}} \cdot \Delta t \qquad \dot{y}(t)_{\text{imu}} = \dot{y}(t - \Delta t) + \ddot{y}_{\text{imu}} \cdot \Delta t \tag{3.4}$$

Thereafter the Complementary filterings of $\dot{x}$ and $\dot{y}$ from the IMU integration and from the Optical flow sensor are performed, shown in Equation 3.5.

$$\dot{x}_{\text{flow}}, \dot{y}_{\text{flow}} = \text{Optical flow reading}$$

$$\dot{x}_{\text{filtered}}, \dot{y}_{\text{filtered}} = \text{filtered velocity readings}$$

$$\dot{x}_{\text{filtered}} = \epsilon \cdot \dot{x}_{\text{imu}} + \dot{x}_{\text{flow}} \cdot (1 - \epsilon) \qquad \dot{y}_{\text{filtered}} = \epsilon \cdot \dot{y}_{\text{imu}} + \dot{y}_{\text{flow}} \cdot (1 - \epsilon) \tag{3.5}$$

The filtered velocity readings ($\dot{x}_{\text{filtered}}, \dot{y}_{\text{filtered}}$) are used as $\dot{x}$ and $\dot{y}$ for the state vector $\mathbf{x}$.

The ϵ constant decides the weighting between the IMU accelerometer and the Optical flow sensor in the filter. For details on the testing conducted to determine ϵ, see Section 4.3.4.

To estimate x and y, a second numerical integration is performed on the filtered $\dot{x}$ and $\dot{y}$ in Equation 3.6.

$$x(t) = x(t - \Delta t) + \dot{x}_{\text{filtered}} \cdot \Delta t \qquad y(t) = y(t - \Delta t) + \dot{y}_{\text{filtered}} \cdot \Delta t \tag{3.6}$$

These x and y values are used to determine the horizontal position and are saved for the state vector $\mathbf{x}$.

3.6.2.2 Vertical position and velocity $(z, \dot{z})$

For estimation of $\dot{z}$ and z another Complementary filter between the derivative of the lidar z position and numerical integration of the IMU accelerometer $\ddot{z}$ value is used. The derivative of the lidar is demonstrated in Equation3.7.

$$z_{\text{lidar}} = \text{lidar reading}$$
$$\dot{z}(t)_{\text{lidar}} = \frac{z(t)_{\text{lidar}} - z(t-\Delta t)_{\text{lidar}}}{\Delta t} \tag{3.7}$$

The numerical integration of the IMU's accelerometer is presented in Equation 3.8.

$$\dot{z}_{\text{imu}} = \text{IMU accelerometer vertical reading}$$
$$\dot{z}(t)_{\text{imu}} = \dot{z}(t-\Delta t) + \ddot{z}_{\text{imu}} \cdot \Delta t \tag{3.8}$$

The sensor fusion is defined in Equation 3.9 with a β constant that determines the dependency on the lidar against the IMU's accelerometer.

$$\dot{z}(t) = \beta \cdot \dot{z}_{\text{imu}} + (1-\beta) \cdot \dot{z}_{\text{lidar}} \tag{3.9}$$

The calculated $\dot{z}$ is saved for the state vector $\mathbf{x}$.

Numerical integration is used to calculate z from $\dot{z}$ in Equation 3.10.

$$z = z(t-\Delta t) + \dot{z} \cdot \Delta t \tag{3.10}$$

Finally the z value is saved for the state vector $\mathbf{x}$.

The β constant determining the weighting in the filter is decided through trial and error testing in Section 4.3.4.

3.7 Control Simulation Using Simulink

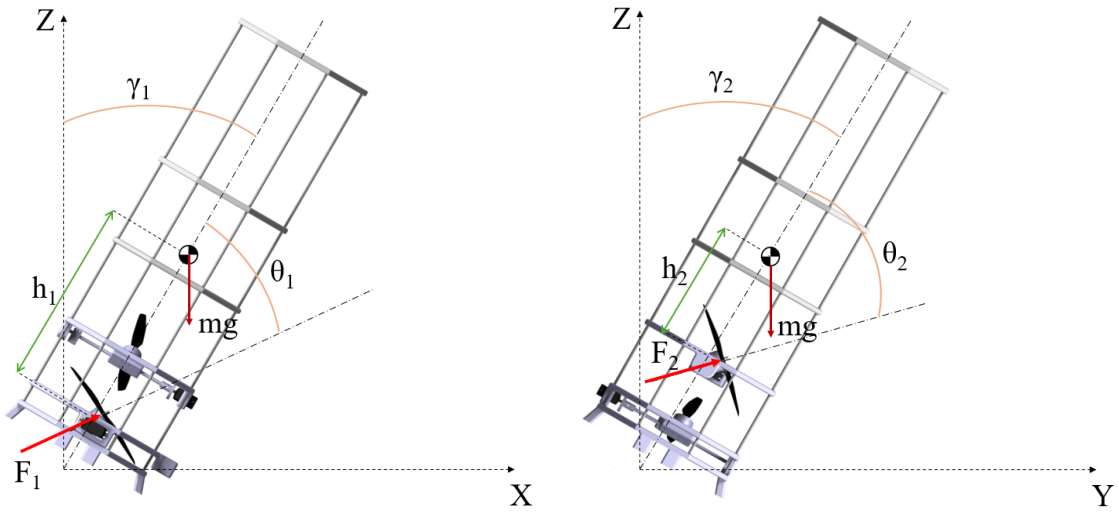
This section establishes the foundation of the simulation environment, enabling the testing and tuning of control mechanisms without the need for repeated physical flights, thereby avoiding external disturbances and easing diagnosis of regulator performance.

3.7.1 Identifying the System Dynamics

The first step in making a robust control system is to identify the system dynamics. The variables are defined in Table 2 and the system's forces are found in Figure 9.

Table 2: Variable definition

x	Position along the X-axis
y	Position along the Y-axis
z	Position along the Z-axis
P_1	Motor 1 power
P_2	Motor 2 power
ω_1	Angular velocity motor 1
ω_2	Angular velocity motor 2
τ_x	Torque about X-axis
τ_y	Torque about Y-axis
τ_z	Torque about Z-axis
γ_1	Pitch angle (rotation about Y-axis)
γ_2	Yaw angle (rotation about X-axis)
θ_1	Gimbal angle of motor 1
θ_2	Gimbal angle of motor 2
I_x	Moment of inertia about the X-axis at the $c.o.m$
I_y	Moment of inertia about the Y-axis at the $c.o.m$
I_z	Moment of inertia about the Z-axis at the $c.o.m$
ϕ	Roll angle (rotation about Z)
d	Diameter of the model
h_1	Distance between motor 1 and the center of mass along the Z-axis
h_2	Distance between motor 2 and the center of mass along the Z-axis
L	Length of the model
m	Mass of the rocket



(a) XZ-plane

(b) YZ-plane

Figure 9: Forces and angles of the system (figure originally from the 2024 thesis [2], reprinted with permission)

From this Equations 3.11-3.15 are derived. The acceleration in x and y direction are written as:

$$\ddot{x} = \frac{F_1}{m} \sin(\gamma_1 + \theta_1) \tag{3.11}$$

and

$$\ddot{y} = \frac{F_2}{m} \sin(\gamma_2 + \theta_2) \tag{3.12}$$

The acceleration along the z-axis are written as:

$$\ddot{z} = \frac{F_1}{m} \cdot \cos(\gamma_1 + \theta_1) + \frac{F_2}{m} \cdot \cos(\gamma_2 + \theta_2) - g \tag{3.13}$$

Furthermore the rotation about the X- and Y-axes, pitch and yaw, are described as:

$$\ddot{\gamma}_1 = \frac{F \sin(\theta_1) h_1}{\bar{I}_x} \tag{3.14}$$

$$\ddot{\gamma}_2 = \frac{F \sin(\theta_2) h_2}{\bar{I}_y} \tag{3.15}$$

A first order Taylor expansion is used to linearize the system about the operating point $x = y = 0$ [13] at any altitude any, meaning:

$$\sin(x) \approx x \tag{3.16}$$

$$\cos(x) \approx 1 \tag{3.17}$$

This in turn gives the simplified Equations 3.18-3.22:

$$\ddot{x} = \frac{F_1}{m}(\gamma_1 + \theta_1) \tag{3.18}$$

$$\ddot{y} = \frac{F_2}{m}(\gamma_2 + \theta_2) \tag{3.19}$$

$$\ddot{z} = \frac{F_1 + F_2}{m} - g \tag{3.20}$$

$$\ddot{\gamma}_1 = \frac{F \theta_1 h_1}{\bar{I}_x} \tag{3.21}$$

$$\ddot{\gamma}_2 = \frac{F \theta_2 h_2}{\bar{I}_y} \tag{3.22}$$

From Equations 3.18-3.22 the state space model is constructed, which is demonstrated in Section 3.7.2.

3.7.2 Creating a State Space Model

From the equations in Section 3.7.1 the system is written in state space form. In the case of the Starship model the inputs are defined as the thrusts of each motor, F_1 and F_2 and the gimbal angle θ_1 and θ_2 of each motor respectively.

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \gamma_1 \\ \dot{\gamma}_1 \\ \gamma_2 \\ \dot{\gamma}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} F_1 \\ F_2 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{3.23}$$

And now using Equations 3.18-3.22 the time derivative of the state space vector is written as Equation 3.24 by using the Taylor expansion explained in Section 3.7.1.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\gamma}_1 \\ \ddot{\gamma}_1 \\ \dot{\gamma}_2 \\ \ddot{\gamma}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{u_1}{m}(x_7 + u_4) \\ x_4 \\ \frac{u_2}{m}(x_9 + u_3) \\ x_6 \\ \frac{u_1 + u_2}{m} - g \\ x_8 \\ \frac{u_1 \cdot u_3 \cdot h_1}{I_y} \\ x_{10} \\ \frac{u_2 \cdot u_4 \cdot h_2}{I_x} \end{bmatrix} \tag{3.24}$$

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ any \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{u}_0 = \begin{bmatrix} \frac{mg}{2} \\ \frac{mg}{2} \\ 0 \\ 0 \end{bmatrix} \tag{3.25}$$

$$\mathbf{A} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(\mathbf{x_0}, \mathbf{u_0})} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.26}$$

$$\mathbf{B} = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{(\mathbf{x_0},\mathbf{u_0})} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{g}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{g}{2} & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & \frac{1}{m} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{mgh_1}{2I_y} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{mgh_2}{2I_x} \end{bmatrix} \tag{3.27}$$

$$\mathbf{C} = \left.\frac{\partial g}{\partial \mathbf{x}}\right|_{(\mathbf{x_0},\mathbf{u_0})} = I_{10\times 10}, \quad \mathbf{D} = \left.\frac{\partial g}{\partial \mathbf{u}}\right|_{(\mathbf{x_0},\mathbf{u_0})} = \mathbf{0}_{10\times 4} \tag{3.28}$$

The C matrix is an identity matrix since every state in the system is either measured or estimated. The D matrix contains only zeros due to the system's inputs not having a direct influence on the outputs. Following this the state space system is discretized using `MATLAB`s `Control System Toolbox`[14]. The following functions are used:

- `ss()` is used to create a state space system
- `c2d()` is used to discretize matrices, so that the continuous time matrices are implemented in code.
- feedback gain $\mathbf{K}$ is be computed using `[K,S,P]=lqr(Ad,Bd,Q,R)` where `Ad` and `Bd` are the discretized $\mathbf{A}$ and $\mathbf{B}$ matrices.

3.7.3 Simulink Model

This section describes the `MATLAB` based `Simulink`[15] and `Simscape` [16] (Copyright (c) 2020, The MathWorks, Inc. All rights reserved) model used to simulate the Starship model. The values in Table 3 are used for dimensions of the rocket in simulation.

Table 3: Parameters of final model

Parameter	Value
m (mass)	2.868 [kg]
I_x (moment of inertia)	0.215 $[kg \cdot m^2]$
I_y (moment of inertia)	0.226 $[kg \cdot m^2]$
I_z (moment of inertia)	0.042 $[kg \cdot m^2]$
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])

Figure 11 shows the Simulink model which consists of the following blocks:

3.7.4 Thrust Vectoring

The subsystem converting in the input signal **u** into force components along the rocket's local X-, Y-, and Z-axes is shown on the left side of Figure 11, and is labeled **Thrust**. It uses the dynamics set up in 3.7.1, and corresponds to the motors in the real Starship model. The block outputs forces, which are fed into the subsystem that dictates the movement of the rocket 3.7.5.

3.7.5 Trajectory and Attitude of Rocket

The subsystem dictating the movement of the rocket is shown at the top of Figure 11, labeled ***Movement, forces and torques***. It receives forces acting along the rocket's local X-, Y-, and Z-axes as input. It uses a 6-DOF joint, which permits translational and rotational motion, making it possible to replicate the dynamics of a rocket. State variables are measured by the 6-DOF joint, and are outputted and fed into the state feedback, see Subsection 3.7.6. This subsystem corresponds to the physical Starship model.

3.7.6 State Feedback

The blocks shown at the bottom of Figure 11 labeled **Bound inputs** and **State feedback** implement state feedback control according to Equation 3.29, provided by [17], and they correspond to the control algorithms implemented in the real system.

$$\mathbf{u} = \mathbf{u_0} - \mathbf{K}(\mathbf{x} - \mathbf{x_0}) \tag{3.29}$$

$\mathbf{x}_0$ denotes the reference state being a stable hover above ground, see Equation 3.25. $\mathbf{K}$ denotes the state feedback gain matrix calculated using a linear quadratic regulator, LQR. $\mathbf{u_0}$ denotes the control input when the system is at the reference state $\mathbf{x_0}$, see Equation 3.25.

Due to the physical constraints of the gimbaling angles, and the thrust generated by each motor, see Figure 10, the control input signal **u** is bounded according to Inequality 3.30 and 3.31.

The bounded control input **u** is fed into the thrust vectoring subsystem 3.7.4.

$$0 < u_{1,2} = F_{1,2} < \frac{3.450g}{2} = 1.725g \tag{3.30}$$

$$-30^\circ < u_{3,4} = \theta_{1,2} < 30^\circ \tag{3.31}$$

An LQR is a state feedback regulator, the control input uses the state vector instead of the output to compute the desired input. The regulator uses two weighting matrices, $\mathbf{Q}$ and $\mathbf{R}$, along with the state matrix $\mathbf{A}$, see Equation 3.26, and input matrix $\mathbf{B}$, see Equation 3.27, to solve for $\mathbf{K}$.$\mathbf{Q}$ penalizes deviations in state variables, and $\mathbf{R}$ penalizes the magnitude of control inputs [17]. The values for $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{K}$ are presented in Subsection 4.5.

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

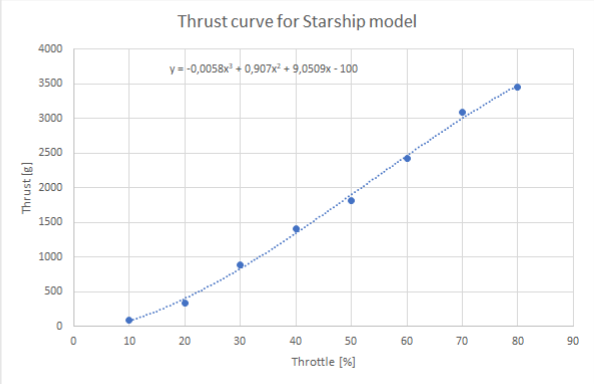


Figure 10: Thrust generated by the Starship model [2], reprinted with permission.

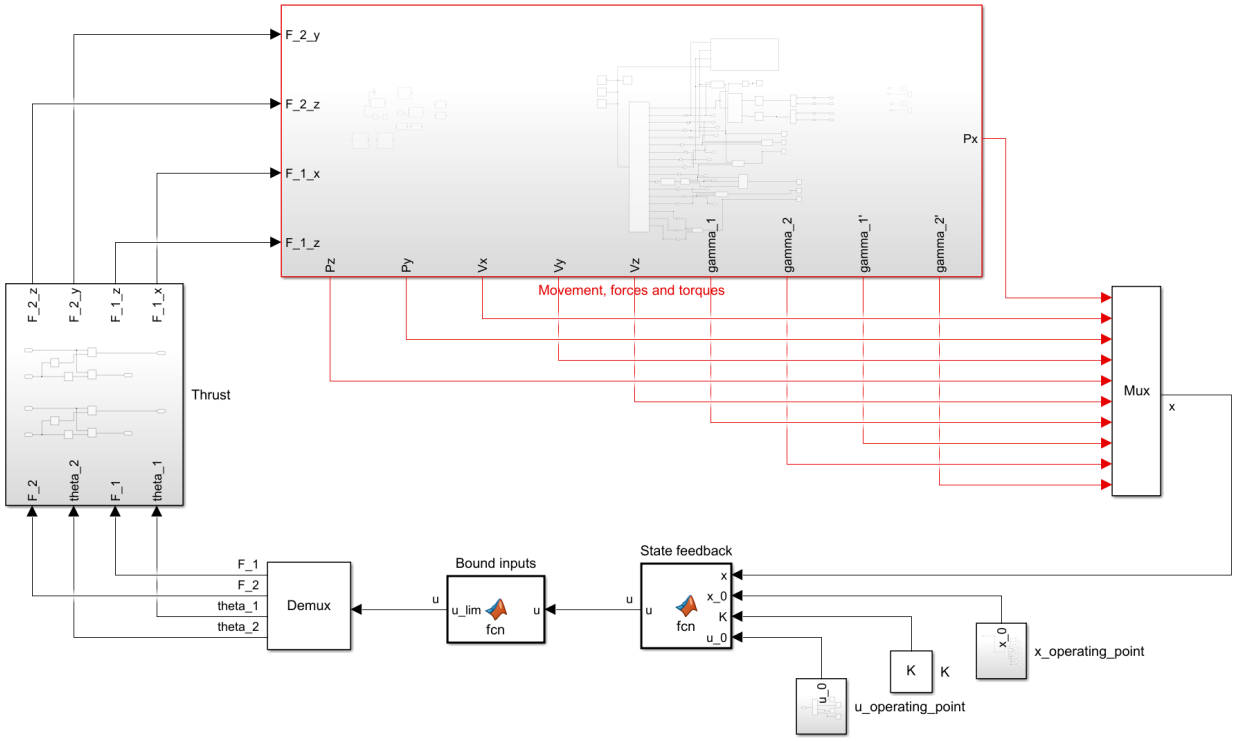


Figure 11: Current state of Simulink model

3.7.7 Linking Sensor Data to the State Vector

The states in the state vector $\mathbf{x}$ described in Section 3.7.2 must be estimated with the Starship model's sensors. The optical flow sensor estimates $\dot{x}$ and $\dot{y}$. The lidar sensor estimates z and the IMU estimates $\dot{\gamma_1}$ and $\dot{\gamma_2}$ with the gyro (angular velocity). That leaves $x, y, \dot{z}, \gamma_1$ and γ_2, which instead must be calculated. The estimations for $\dot{x}, \dot{y}$ and z can also be improved with further sensor fusion calculations.

The yaw and pitch (γ_1 and γ_2) are calculated with a Complementary filter mentioned in Section 3.6.1. The $x, \dot{x}, y, \dot{y}, z$ and $\dot{z}$ are calculated with two other Complementary filters described in Section 3.6.2.

- $\dot{\gamma_1}, \dot{\gamma_2}$ = gyro readings from the IMU.
- γ_1, γ_2 = Complementary filtering of the IMU described in Section 3.6.1
- $x, y, \dot{x}, \dot{y}$ = Complementary filtering of the Optical flow and IMU described in Section 3.6.2.
- $z, \dot{z}$ = Complementary filtering of the lidar and IMU described in Section 3.6.2.

4 Implementation Outcomes and Reflections

This section summarizes the results of the technical implementations, highlighting both successful outcomes and key challenges encountered. It reflects on the development process, the effectiveness of the chosen tools and methods, and insights gained during model construction and testing.

4.1 Impact of Software Improvements for the Modularity of the System

As a result of the code restructuring described in Section 3.5, the Starship software achieves modularity. The separation of functionality into sensor specific and function specific libraries has led to several benefits during development and testing. Firstly the time required to implement and test individual components is reduced. This is demonstrated by the straightforward software integration of the optical flow sensor as shown in Figure 12.

lib > Sensor_handler > Sensor_handler.cpp

```
20  SensorData SensorHandler::readSensors() {
21      SensorData data = {0};
22
23      // Read IMU
24      imu.read(data.imu_accel_x, data.imu_accel_y, data.imu_accel_z,
25               data.imu_gyro_x, data.imu_gyro_y, data.imu_gyro_z, data.imu_temp);
26
27      // Read Optical Flow
28      if (flowDataReady) {
29          flowDataReady = false; // Reset the flag
30          flow.readMotion(data.flow_x, data.flow_y); //, data.flow_quality);
31      }
32
33      // Read LiDAR
34      lidar.getData(data.lidar_dist, data.lidar_flux, data.lidar_temp); // Default I2C address for TFMini
35
36      return data;
37  }
```

(a) Demonstrates how the *sensor_handler* collects the sensor data

main.cpp

```
SensorData data = sensorHandler.readSensors();
```

(b) Call on *sensor_handler* in main file to access sensor data. Requires a single line of code and all sensor values are part of the data structure

Figure 12: Shows the simplicity of accessing all sensor data

Tuning specific parameters of a certain sensor is done in the dedicated file as mentioned in 3.5 without affecting other parts of the system. Due to this, conducting tests for tuning for example the yaw and pitch angles from the complementary filter is easier.

As mentioned in Section 3.5 the codebase from the 2024 thesis [2] is complex, which limits the ability to modify or extend its functionality. Notably, it includes a Madgwick filter for orientation estimation, which does not function correctly on the current system. Simulations performed in MATLAB indicate that the Madgwick filter offer superior performance compared to a Complementary filter, particularly in terms of dynamic response and drift correction. However, due to integration challenges and time constraints, a Complementary filter is chosen instead, as it is easier to implement and provides accurate and stable results (see Section 4.3.1).

4.2 Results of the Ground Power Cable

Another improvement related to easier testing is that of the ground power cable (see Figure 4). The extensive software testing performed on the system required frequent charging of the LiPo 2S

battery providing power to the microcontroller. This limited the testing capabilities and became a bottleneck. During an extended test session, the battery discharged beyond its safe limit, rendering it unable to recharge. Therefore a ground power cable became necessary for safely performing multiple tests.

4.3 Sensor Improvements for Data Collection

The change to have signal reading using interrupt request (IRQ) for IMU and radio transmitter resulted in a processing rate of around 195 packages of data per second. The processing rate is calculated by the number of operations over a time period showing a convergence to approximately 195 operations/second.

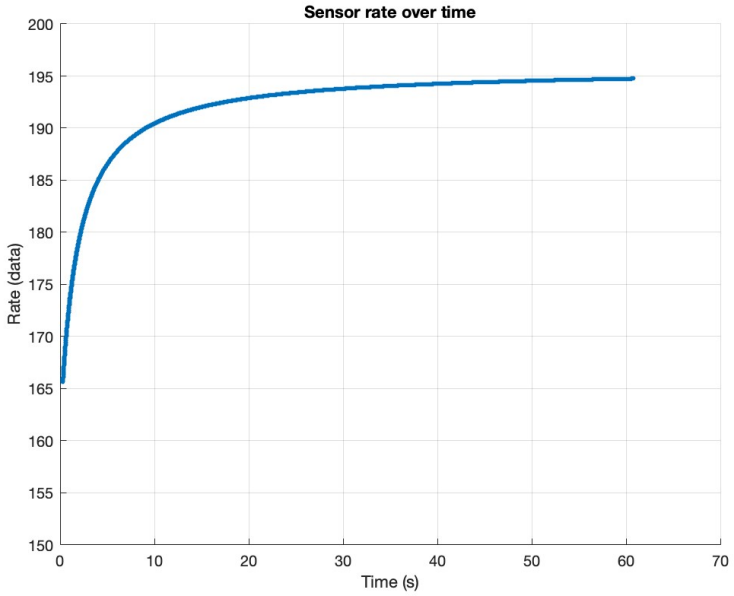


Figure 13: Graph showing sensor data readout by the Teensy 4.1 microcontroller per second

4.3.1 Accuracy of Yaw and Pitch Orientation from IMU Data

The resulting pitch (γ_1) and yaw (γ_2) estimated using the complementary filter described in Section 3.6.1 demonstrate good accuracy in short-term tests. The filter effectively reduces sensor noise from the IMU, as discussed in the 2024 thesis [2], and this performance is illustrated in a real time demonstration video available on YouTube[18] and in Figure 14.

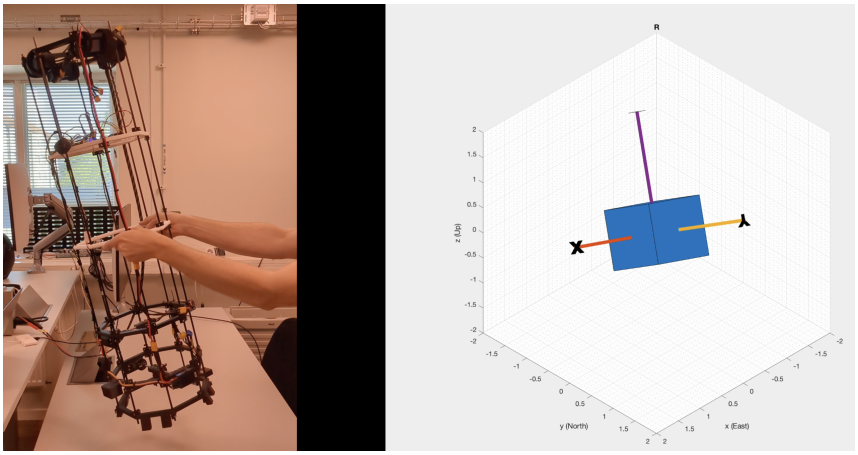


Figure 14: Real orientation of the Starship model compared with the complementary filter's estimated orientation

In the video, the orientation response appears smooth and closely follows the tilt of the rocket model with a slight delay. This delay is partially attributable to the serial communication between the Teensy 4.1 microcontroller and the PC, as well as the rendering time of the MATLAB visualization.

As described in Section 4.1 a Madgwick filter can perform better than the current Complementary filter if correctly calculated by the Teensy 4.1. Despite this, there are some advantages of the Complementary filter compared to a Madgwick. One advantage is that it is fast with simple calculations and provides low latency to the system. This results in faster sensor data collection.

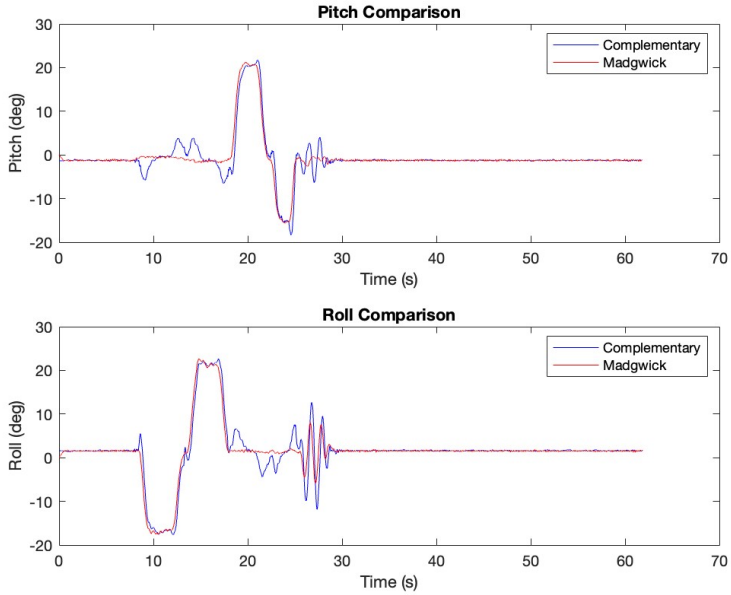


Figure 15: Madgwick filter compared to a Complementary filter. Test performed in MATLAB with built in Madgwick functionality. As seen in the graph, the Madgwick filter provides slightly less noise. [19].

Another filter that possess the potential to sensor fuse the accelerometer and gyro is a Kalman filter which first predicts a future state with some measurement and a state space model and then corrects the prediction with another measurement [12]. In the prediction step of the filter the gyro is used to predict the pitch and yaw angles and then compare those predictions with the

accelerometers estimation of pitch and yaw in the correction step. For estimating accurate pitch and yaw angles for the Starship model, the complementary filter provides more favorable results compared to those from a Kalman filter. This may be attributed to difficulties in parameter tuning.

Accurate orientation estimation is critical for precise control of the motor driven gimbal system, which is responsible for maintaining the Starship model in an upright position. Thus, achieving reliable attitude estimation marks a significant step toward autonomous control of the system.

4.3.2 Tuning of α Constant used in Pitch and Yaw Orientation

The tuning of the complementary filter constant α for pitch and yaw estimation, as described in Section 3.6.1, is carried out through empirical testing. Figure 16 shows the comparison of pitch response for different α values.

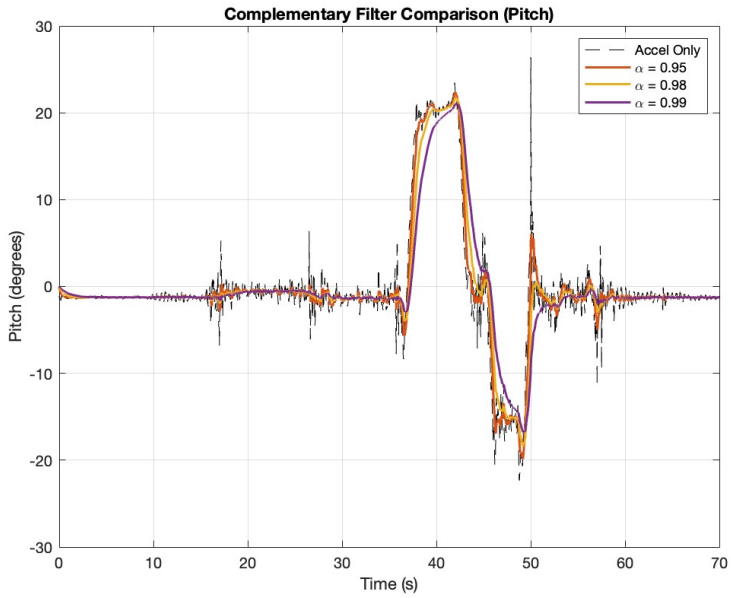


Figure 16: Pitch estimation comparison using different α values in the complementary filter. Accel Only means the α is set to 0.0.

From the graph, it is evident that a lower α value (e.g., 0.95) offers faster response but introduces noticeable noise, while a higher value (e.g., 0.99) smooths the output but adds delay in dynamic situations. The value $\alpha = 0.98$ (shown in yellow) is selected as the optimal trade-off, providing a smooth response while maintaining adequate sensitivity to rapid changes in pitch.

4.3.3 Accuracy of the Dead Reckoning Positioning

Dead reckoning position mentioned in Section 3.6.2 is a challenge to get accurate long-term, but short-term test results suggest favorable performance. The filtered x and y positions is relatively stable as seen in Figure 17.

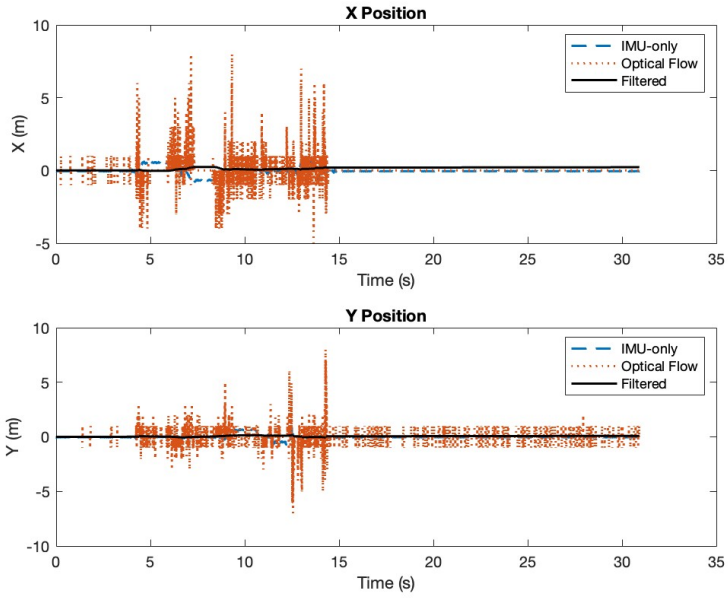
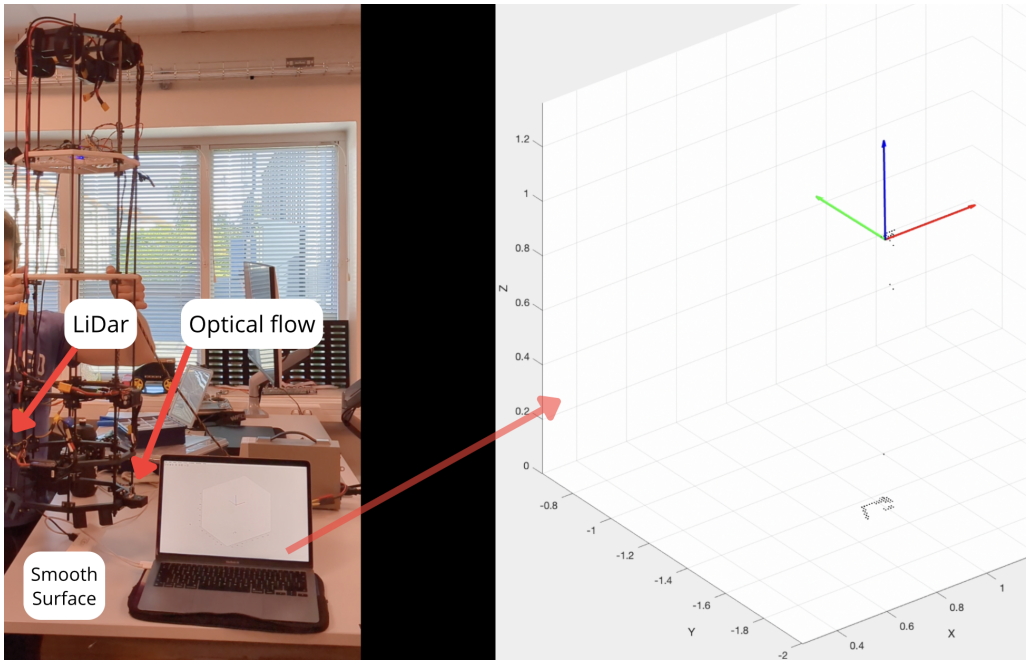
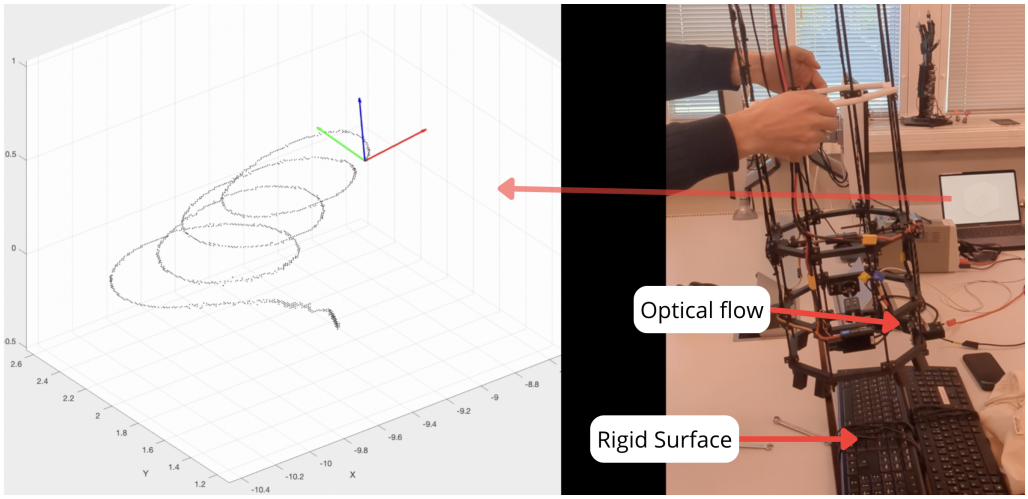


Figure 17: Graph showing raw IMU and Optical flow sensor position data vs the filtered positioning when tilting the Starship model at the same position.

Two real time demonstrations of the position estimation are presented on Youtube here[20] and here[21] (is also shown in Figure 18a and 18b). The first video demonstrates planar circular motion on a rigid surface, while the second shows similar motion on a smoother surface, including a moment where the lidar detects a height difference as the model moves off the table. The trajectory estimated on the rigid surface aligns more accurately with the actual path than the estimate obtained on the smooth surface. This discrepancy arises because planar motion is primarily detected by the optical flow sensor, which has limited accuracy in estimating velocity when it cannot reliably identify reference points on low-texture or smooth surfaces.



(a) In the figure the lidar detects a height difference as the model is moved off the table.



(b) Circular planar motion on a rigid surface.

Figure 18: Position estimation tests using the optical flow sensor and lidar. .

In the long-term dead reckoning positioning is not reliable as the sensor noise in each integration will add up and make the position estimation drift. For better position estimation, a well tuned Kalman filter would provide a more robust solution. A Kalman filter is described in the last paragraph of Section 4.3.1.

4.3.4 Tuning of Complementary Filter Constants used in Position Estimation

The Complementary filter constants β and ϵ, which is introduced in Section 3.6.2, are determined through an empirical trial and error process. Various values are tested to evaluate how well the estimated position matched the actual trajectory of the Starship model.

The constant β determines the weighting between the IMU's vertical acceleration data and the lidar's height measurements. After testing β values, the resulting height trajectories are compared to the true flight profile. As shown in Figure 19, tests revealed that lidar provides more

reliable height measurements than the IMU. Therefore, a low β value of 0.05 is selected to heavily favor the lidar readings.

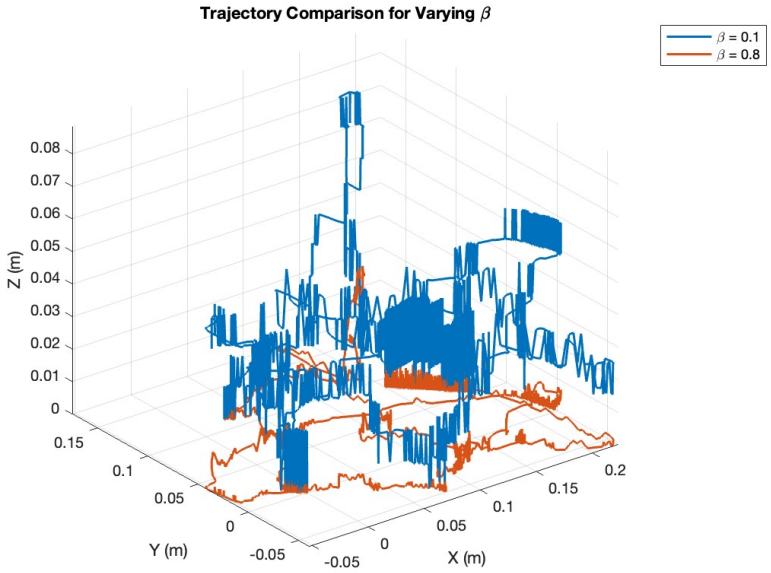


Figure 19: Comparison of position estimation using different β values.

A lower β emphasizes lidar data, while a higher value prioritizes IMU acceleration. The blue line shows better vertical accuracy compared to the red line, resulting in β being set to 0.1. The lidar provides measurements in centimeters, which results in discrete steps of 0.01 meters. This quantization introduces small jumps in the data, affecting the smoothness of the graph.

The constant ϵ is used to weight horizontal velocity estimates between the IMU and the optical flow sensor. Tuning ϵ is challenging due to the noisy nature of the IMU and the dependence of optical flow accuracy on surface texture. After testing various configurations, a value of $\epsilon = 0.8$ is selected. This setting favors the optical flow sensor, which generally provides better velocity estimates under most conditions.

Figure 20 shows the trajectory obtained using this final configuration, demonstrating reliable position tracking performance.

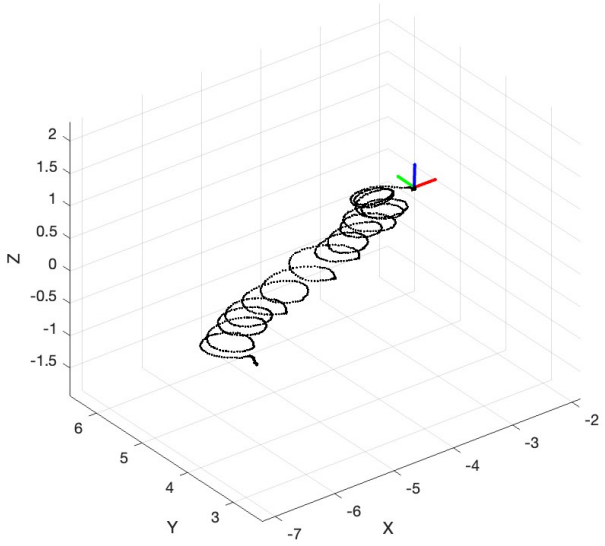


Figure 20: Trajectory of the position estimation with ϵ set to 0.8.

This favors optical flow data for horizontal velocity estimation. The graph depicts a trajectory that resembles the actual path recorded during testing, but with noticeable drift over time. Video is presented on YouTube[20].

4.4 Simulation

This section will describe the results from testing the simulation model. Simulations mirror the behavior of the real world system, suggesting that the underlying model structure is sound. However the simulation is not yet complete, and requires further development to be a reliable tool for experimenting with different control strategies and tuning controllers based on realistic system responses.

4.4.1 Limitations of Simulated Model

The simulated model has three notable limitations. First, it does not include roll. Since the model does not have rotors, there is no torque about the Z-axis, meaning the rocket does not initiate rotation about this axis. This is a simplification in the simulation, as torque is introduced in the real system due to the rotors, and controlling roll is essential for controlling the rocket. Second, the lack of rotors also excludes the unaccounted for perpendicular torque, which is explained more in Subsection 4.6.1. Meaning that the simulation is more decoupled than the real system. Third, and most critically, the forces applied are not fully working as described in Subsection 4.4.2.

4.4.2 Testing and Verification of Forces and Motion

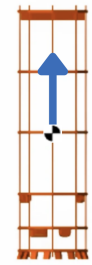
To verify that the system behaves as predicted by theory, a series of eight tests are conducted. These tests exclude gravity, and the moments of inertia about the rocket's local X- and Y-axes are set to one:

$$I_x = I_y = 1 \tag{4.1}$$

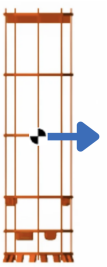
The updated values for the moments of inertia simplify calculations, and the exclusion of gravity, simplifies isolation of movement caused by specific forces.

Test 1 is a constant 1 N force applied along the rocket's local Z-axis from its geometric center and c.o.m. Resulting in acceleration along the global Z-axis. Test 2 applies the force along the local

X-axis. As shown in Figure 22, the rocket translates in the direction of the applied force, without motion along other axes. See visualization in Figure 21.

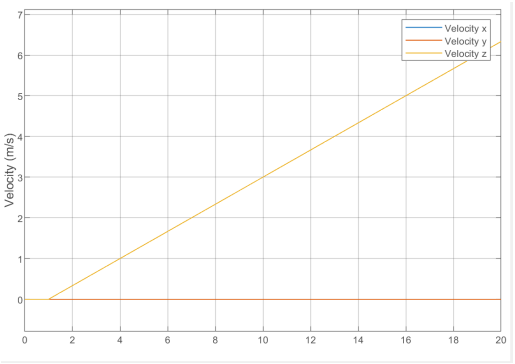


(a) Test 1, force applied along the Z-axis.

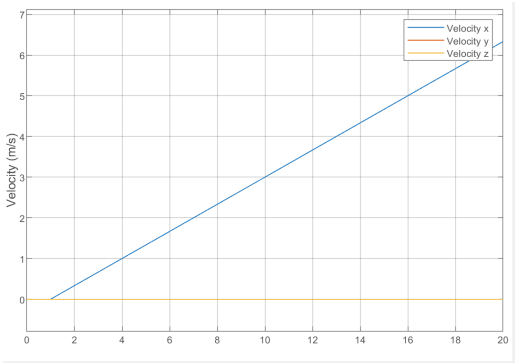


(b) Test 2, force applied along the X-axis.

Figure 21: Visualization for the forces applied in test 1 and 2



(a) Force applied along Z-axis. The rocket only moves along global Z-axis.



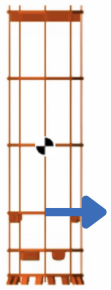
(b) Force applied along X-axis. The rocket only moves along global X-axis.

Figure 22: First and second test, time is shown on the horizontal axis.

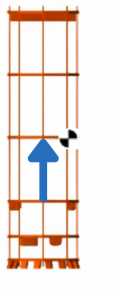
In Test 3, a constant 1 N force is applied along the rocket's local X-axis at the upper motor location (297 mm below the geometric center), see visualization in Figure 23a, producing a negative torque about the local Y-axis. As shown in Figure 24b, the torque magnitude agrees with Equations 4.2 and 4.3, given the force is orthogonal to the lever arm. The applied force also induces translation in the global-XZ plane, illustrated in Figure 24a.

$$\tau = Fr \tag{4.2}$$

$$\tau = I\alpha \tag{4.3}$$

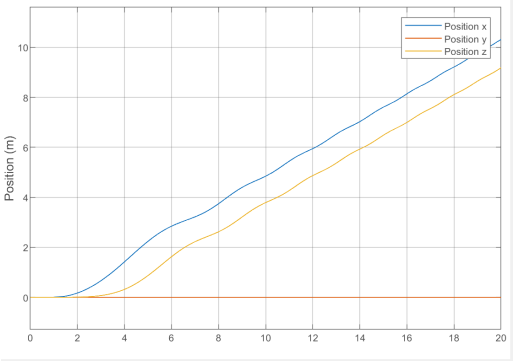


(a) Test 3, a constant force applied along the X-axis.

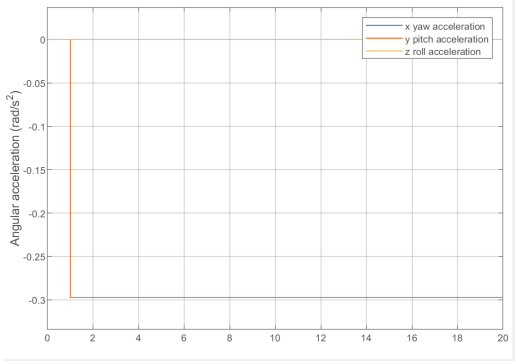


(b) Test 4, a constant force applied along the Z-axis.

Figure 23: Visualization for the forces applied in test 3 and 4



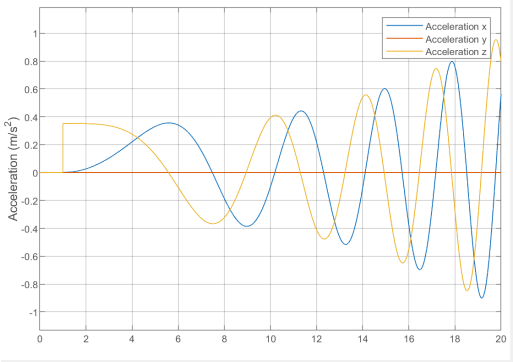
(a) Position of the rocket.



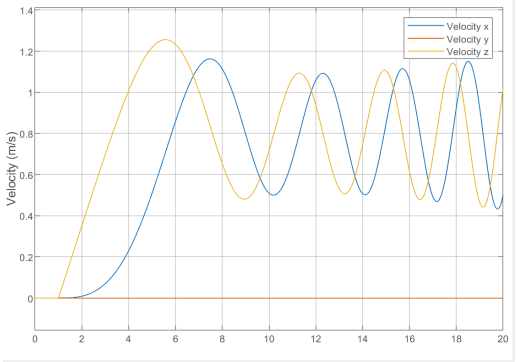
(b) Constant angular acceleration of the rocket.

Figure 24: Third test, applied force generates negative torque about Y-axis, time is shown on the horizontal axis.

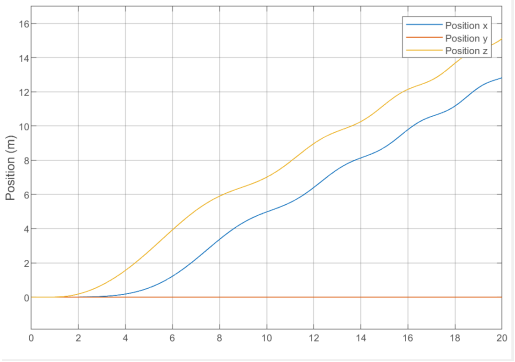
In Test 4, the center of mass is shifted 136 mm along the rocket's X-axis to (136 mm, 0 mm, 500 mm), and a constant 1 N force is applied along the rocket's local Z-axis at the upper motor location, see visualization in Figure 23b). As the thrust is no longer aligned with the center of mass, it generates a torque about the local Y-axis due to its perpendicularity to the local X-axis. Figure 25 shows translation in the global XZ plane, resulting from this induced rotation. This result confirms that the rocket responds to the change in the center of mass, as it has a curved trajectory.



(a) Acceleration of the rocket



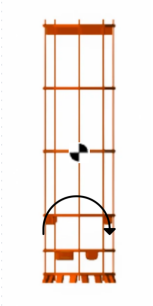
(b) Velocity of the rocket



(c) Position of the rocket

Figure 25: Fourth test, the rocket's trajectory curves as its c.o.m. is shifted, time is shown on the horizontal axis.

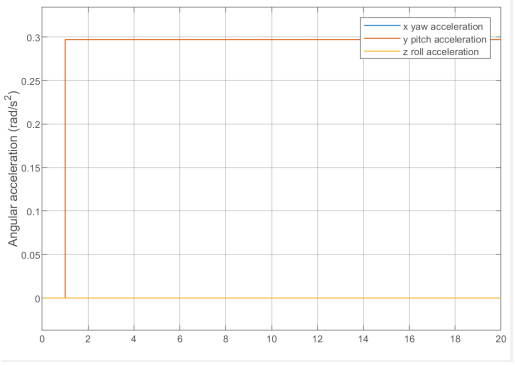
Tests 5 and 6 apply torque directly to the rocke without any external force, resulting in pure rotational motion without translation. In Test 5 a 0.297 Nm positive torque is applied about the local Y-axis at the upper motor position, with results shown in Figure 27. In Test 6, the torque origin is moved to the geometric center. As predicted by rotational dynamics, the motion remains unchanged, as the applied torque is identical, see Figure 28. See visualization in Figure 26.



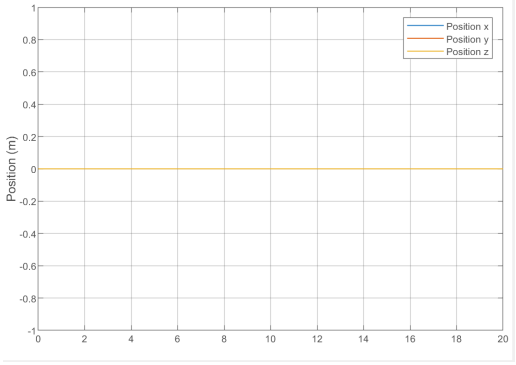
(a) Test 5, torque test with torque applied at upper motor

(b) Test 6, torque test with torque applied at c.o.m.

Figure 26: Visualization for the forces applied in test 5 and 6



(a) Constant angular acceleration of the rocket

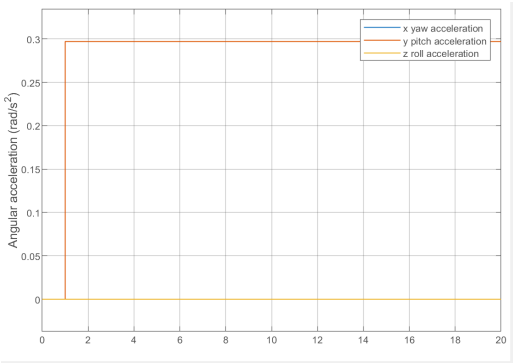


(b) Geometric center of the rocket does not move

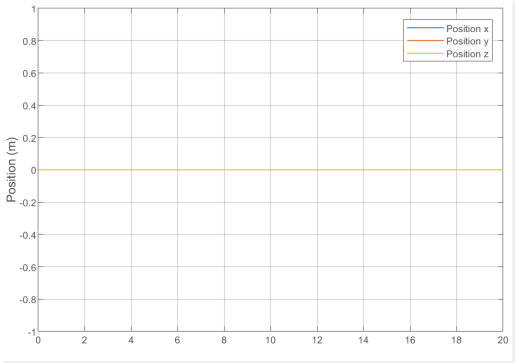
Figure 27: Fifth test, positive torque applied about Y-axis, time is shown on the horizontal axis.

Test 7 investigates whether two antiparallel forces, applied at the motor locations, produce canceling torques. See visualization in Figure 30a. A 1 N force is applied along the rocket's local X-axis at the upper motor, and a -0.704 N force along the same axis at the lower motor. According to Equation 4.2, the resulting torques are equal in magnitude and opposite in direction, yielding near-zero net torque, as confirmed by the negligible angular acceleration in Figure 29b (note the vertical axis scale of 10^{-11}). The net force results in translational acceleration, as shown in Figure 29a and calculated by Equation 4.4:

$$\frac{F_{res}}{m} = \frac{0.296}{3} \tag{4.4}$$

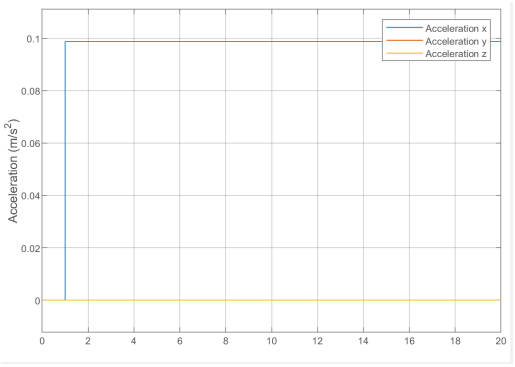


(a) Constant angular acceleration of the rocket

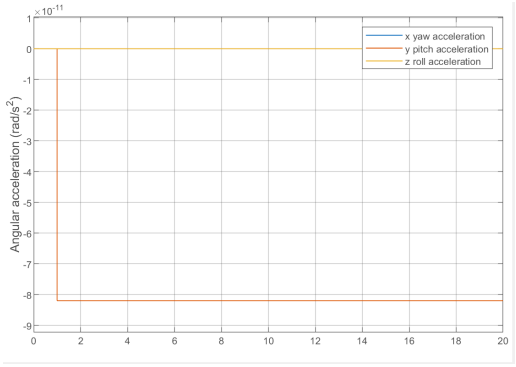


(b) Geometric center of the rocket does not move

Figure 28: Sixth test, positive torque about Y-axis applied in geometric center, time is shown on the horizontal axis.



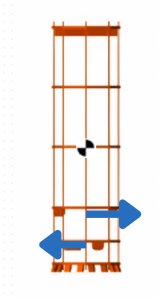
(a) Constant acceleration of the rocket



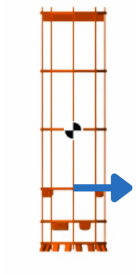
(b) Constant angular acceleration of the rocket

Figure 29: Seventh test, two opposite torques applied to cancel each other out.

Test 8 applies a constant 1 N force along the rocket's local Y-axis at the upper motor location (297 mm below the geometric center), generating a positive torque about the local X-axis. While the torque magnitude aligns with Equation 4.3, its sign deviates from the expected result based on Section 3.7.1, where $F_y > 0$ should result in $\gamma_2 < 0$. Results are shown in Figure 31, with signals initiated at $t = 0$ instead of $t = 1$. See visualization in Figure 30b.



(a) Test 7, two constant anti parallel forces



(b) Test 8, a constant force applied along the Y-axis.

Figure 30: Visualization for the forces applied in test 7 and 8

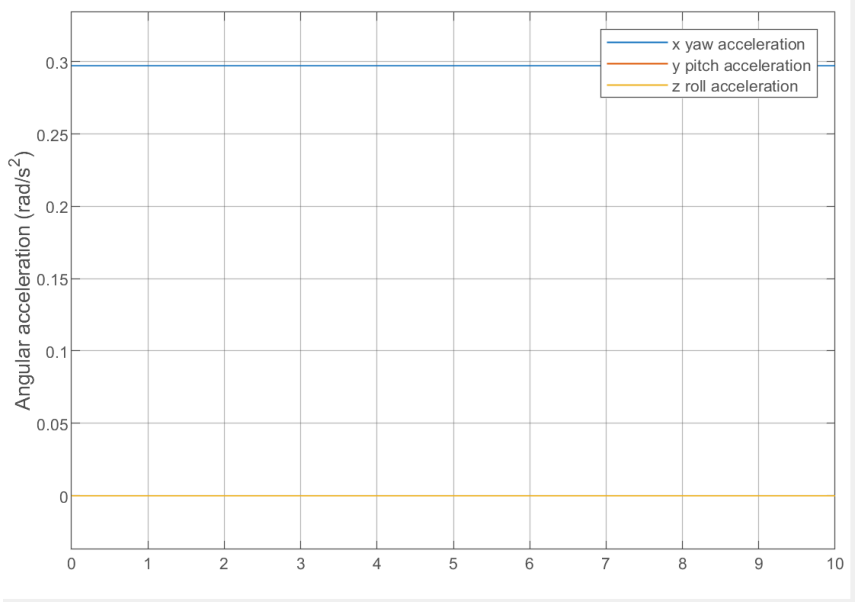


Figure 31: Test 8, positive force along Y-axis results in positive torque about X-axis, time is shown on horizontal axis

These tests demonstrate that the forces are not entirely correctly applied, since a positive force along the rocket's local Y-axis, results in a negative torque, when it should result in a positive torque. Other than that, the system behaves as expected according to the principles of the dynamics.

4.5 Tuning the Control Algorithm

In this section the values of the Q and R matrices are presented. The values are

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

The reasoning behind the values are as followed:

The code is available on Github[5] in the folder Simulation_and_Matlab. The values of Q and R, seen in Equation 4.5 are determined by the desired behavior, see Table 4. The reasoning follows the approach outlined in [17]. No further parameter optimization is performed using simulation or flight testing.

4.6 Limitations of Construction

The mechanical design of the Starship model imposes significant limitations on the effectiveness and applicability of traditional control theory. Several structural choices deviate from conventional rocket architecture, introducing coupling effects and actuation constraints that complicate the modeling and control. These limitations are outlined below.

Table 4: Design rationale behind the weighting of states in Q and the input cost matrix R.

State/Matrix	**Rationale**
Altitude z	Heavily weighted, as maintaining vertical position is a primary control objective.
Orientation angles γ_1, γ_2	Assigned high weights due to their critical role in ensuring the stability of the Starship model.
Horizontal positions x, y	Given low priority, since precise lateral positioning is secondary to maintaining a stable and safe flight.
Angular rates $\dot{\gamma}_1$, $\dot{\gamma}_2$	Less emphasized, as stability is primarily enforced through control of the angles themselves.
Velocities $\dot{x}$, $\dot{y}$	Assigned moderate importance, as they influence the dynamic response of the vehicle in the horizontal plane.
Cost matrix R	Defined to penalize excessive gimbal actuation, thereby discouraging rapid or aggressive control inputs that could induce undesirable oscillations or structural stress.

4.6.1 A Coupled Design

The unmodeled torque of the Starship model will be explained in this section. The Starship gimbals that are introduced in the Bachelor Thesis (2024) [2]. One such assumption concerns the torque generated about the Z-axis, as expressed in Equation 4.7. From fundamental mechanics [22]:

$$P = \tau \cdot \omega \tag{4.6}$$

This means

$$\tau_z = \frac{P_1}{\omega_1} - \frac{P_2}{\omega_2} \tag{4.7}$$

The rotors are contra rotating, therefore their torques are counteracting each other. Under the condition that both motors operate at equal angular velocities and the gimbal angles are zero (i.e., $\theta_1 = \theta_2 = 0$), the resulting torque about the Z-axis simplifies to:

$$\tau_z = 0 \tag{4.8}$$

This formulation neglects the additional torques induced when the motors are tilted. Specifically, tilting motor 1 produces a torque about the X-axis, perpendicular to the illustrated force seen in Figure 9a, while tilting motor 2 produces a torque about the Y-axis, perpendicular to the illustrated force seen in Figure 9a. Meaning a change in θ_1 will cause a movement in γ_2 and the reverse also applies. This is not accounted for in the state space model and is therefore unobservable and consequently uncontrollable.

The torques about the X- and Y-axes are estimated using Equations 4.9 and 4.10

$$\tau_x = \frac{P_1}{\omega_1} \cdot \sin(\theta_1) \tag{4.9}$$

$$\tau_y = \frac{P_2}{\omega_2} \cdot \sin(\theta_2) \tag{4.10}$$

and due to the inability to directly measure power (P) and angular velocity (ω) another limitation is introduced. They are therefore not incorporated into the state space model used in this thesis. When the motors are tilted, the torque has the ability to be decomposed into components, resulting in a torque in the perpendicular plane. This is not accounted for in the state space model of the control system. Using Equation 4.10, together with data from [23] yields

$$\frac{1600\,\mathrm{W}}{18000\,\mathrm{rpm}} \cdot \sin(5^\circ = 0.087\,\mathrm{rad}) = \frac{1600\,\mathrm{W}}{1885\,\mathrm{rad/s}} \cdot 0.087 = 0.848\,\mathrm{Nm} \cdot\ 0.087 = 0.075 Nm \tag{4.11}$$

The torque is small and increases linearly with the angle θ, this is shown in Figure 32. In a worst case scenario when $\theta = 30°$ the torque $\tau_{max} = 0.8Nm$. Since the torque is smaller than the produced thrust, see Figure 10, neglecting the torque resulting from tilting the rotors is deemed a reasonable simplification.

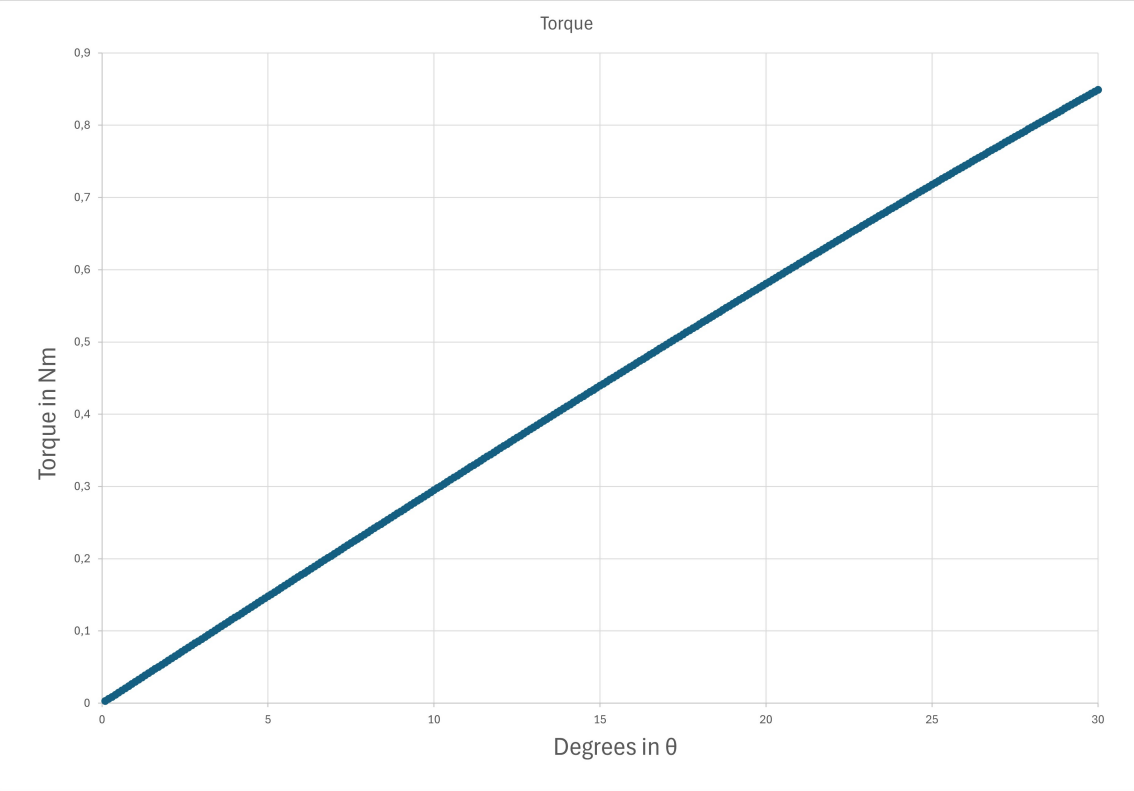


Figure 32: The unmodeled torque about the X- and Y-axes

A rocket does not have propellers and does not require roll control in the same degree that the Starship model does. A rocket only has one origin of thrust and the Starship model has two separate origins of thrust, see Subsection 1.1.1. The changed mechanical design of the Starship model may result in behavior less characteristic of a traditional rocket, more difficult to control, and coupling the control of lift forces with the lateral forces, see Section 4.6.1.

4.6.2 Belly Flop Maneuver

The belly flop is explained in detail in the Appendix of the bachelor thesis 2024 [2]. The maneuver requires air resistance to be effective. The current Starship model is most likely unable to have body panels without interfering with the thrust vectoring. This implies that before the belly flop maneuver is possible to implement in control, the design of the propulsion system of the Starship model must support body panels, which it currently does not.

5 Opportunities for Continued Exploration

This section outlines potential directions for further development of the Starship model. These opportunities span hardware, software, and control domains, and aim to improve system performance, reliability, and fidelity. The suggestions are intended to guide future project teams in advancing the platform both experimentally and analytically.

5.1 Hardware

This subsection outlines potential mechanical improvements that may simplify control and more closely resemble rocket characteristics.

5.1.1 Propulsion System

Future iterations of the platform should explore mechanically decoupling the control inputs to simplify the control problem and improve stability. Implementing a single 2-axis gimbal using contra-rotating coaxial propellers. This configuration could minimize interaction between control axes and reduce the need for compensation in software. Design efforts should also evaluate the use of a centralized thrust point to better emulate rocket behavior. Attention should be given to symmetry, weight distribution, and mechanical robustness to ensure that any new design remains practical for testing and experimentation. Investigating the trade-offs between mechanical complexity and control simplicity will be crucial in this process.

5.1.2 Aerodynamic Surfaces for Belly Flop Maneuver

To perform the belly flop maneuver, several key aspects of the Starship model would need to be redesigned, particularly to account for aerodynamic surfaces and structural integration. Due to the complexity of this maneuver and its limited relevance to the current platform's capabilities, future projects are advised to focus efforts on more achievable objectives unless a dedicated redesign is undertaken.

5.2 Control

This subsection presents ideas for advancing the control strategy, including improvements in modeling, estimation, and regulation to achieve more robust and autonomous flight.

5.2.1 System Identification

Future development of the control system would benefit from refined system identification. Collecting flight data from stable hovering and controlled step responses could enable data driven modeling approaches, improving accuracy beyond the current physics based model. Incorporating machine learning or adaptive estimation techniques may also help capture unmodeled dynamics such as coupled torques and nonlinearities. To enable this, ensuring reliable and noise resistant sensor measurements is critical. A more comprehensive identification process could unlock advanced control strategies and better simulation fidelity.

5.2.2 Working 3D Simulation

To achieve fully functioning simulation environment, it is necessary to incorporate roll, model torques perpendicular to the rotors, and ensure that forces behave in accordance with the laws of dynamics, are all needed to create a high fidelity simulation.

5.2.3 Replacing the IMU

Replacing the current 6-DOF IMU with a 9-DOF IMU, including a magnetometer, would allow for roll estimation. The magnetometer allows full 3D orientation estimation (roll, pitch, yaw) including an absolute reference for yaw, which is not available with a 6-DOF IMU.

References

[1] W. Dahlin, E. Fabricius, G. Geelnard, A. Gleisner, D. Klerebladh and S. Källhammar. 'Modellering och reglering av eldriven raket - För autonom uppskjutning, flygning och landning.' (May 2023), [Online]. Available: https://odr.chalmers.se/items/2c6cbece-f72c-45bb-ae64-2de4307e59ff (visited on 02/04/2025).

[2] G. Edman, O. Haapalo, G. Haller, N. Holmén, E. Reinfeldt and E. Öhman. 'Starship Model.' (May 2024), [Online]. Available: https://odr.chalmers.se/items/125634e5-7b90-48d7-811e-d45ce1ff6923 (visited on 02/04/2025).

[3] A. S. Teitel. 'How does SpaceX build its Falcon 9 reusable rocket?' (Sep. 2017), [Online]. Available: https://www.sciencefocus.com/space/how-does-spacex-build-its-falcon-9-reusable-rocket (visited on 02/04/2025).

[4] SpaceX. 'STARSHIP.' (May 2025), [Online]. Available: https://www.spacex.com/vehicles/starship/ (visited on 12/05/2025).

[5] O. Gustafson, S. H. Aga, G. Magnusson, E. Ringström, F. Sandén and S. Thurfjell. 'starship-model-2025.' (May 2024), [Online]. Available: https://github.com/TheRealQuasi/starship-model-2025 (visited on 13/05/2025).

[6] iCharger NZ. 'LiPo battery safety guide.' (2022), [Online]. Available: https://www.icharger.co.nz/lipo-battery-safety-guide (visited on 02/04/2025).

[7] WFLY. 'WFT09S RADIO CONTROL SYSTEM PCMS 4096 PCMS RECEIVER DUAL MCU DUALANTENNA INSTRUCTION MANUAL v01.' (), [Online]. Available: https://www.himodel.com/en/info/manual/WFT09S_instruction_manual_090716.pdf (visited on 26/04/2025).

[8] M. Electronics. 'Optical flow sensor PIM453.' (Mar. 2025), [Online]. Available: https://www.mouser.se/ProductDetail/Pimoroni/PIM453?qs=PzGy0jfpS%20MuJnlsymXyulA%3D%3D (visited on 02/04/2025).

[9] Bitcraze. 'Bitcraze PMW3901.' (Dec. 2022), [Online]. Available: https://github.com/bitcraze/Bitcraze_PMW3901/tree/master (visited on 01/05/2025).

[10] M.-A. Pedro, L.-d.-F. L. Eduardo and D.-R. Arnoldo, 'Interrupts mechanism,' in *Interrupt Handling Schemes in Operating Systems*, Gewerbestrasse 11, Cham, Ch 6330, Switzerland: Springer International Publishing, 2018, pp. 1–13. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-94493-7_1 (visited on 26/04/2025).

[11] Bosch. 'BMI088 IMU.' (Jan. 2024), [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi088-ds001.pdf (visited on 13/05/2025).

[12] G. Pengfei, T. Liqiong and S. Mukhopadhyay. 'MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion.' (Jun. 2015), [Online]. Available: https://ieeexplore.ieee.org/document/7334442 (visited on 26/04/2025).

[13] B. Lennartson, *Reglerteknikens grunder*. Lund: Studentlitteratur, 2006, pp. 83–98.

[14] MathWorks. 'Control systems toolbox.' (2025), [Online]. Available: https://se.mathworks.com/help/control/index.html (visited on 12/05/2025).

[15] I. The MathWorks. 'Simulink is for Model-Based Design.' (2025), [Online]. Available: https://se.mathworks.com/products/simulink.html (visited on 27/04/2025).

[16] I. The MathWorks. 'Simscape Multibody.' (2025), [Online]. Available: https://se.mathworks.com/help/sm/index.html (visited on 28/04/2025).

[17] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*. 29 West 35th Street, New York, NY 10001: CRC Press, 2010, pp. 45, 239–265.

[18] E. Ringström and F. Sandén. 'Pitch and yaw orientation from complementary filter.' YouTube video. (May 2025), [Online]. Available: https://youtu.be/k0ntBAkZBpg (visited on 06/05/2025).

[19] ahrs read the docs. 'Madgwick orientation filter.' (2025), [Online]. Available: https://ahrs.readthedocs.io/en/latest/filters/madgwick.html (visited on 12/05/2025).

[20] E. Ringström and F. Sandén. 'Position estimation with optical flow, imu and lidar.' YouTube video. (May 2025), [Online]. Available: https://youtu.be/R1Grn2xolLU (visited on 13/05/2025).

[21] E. Ringström and F. Sandén. 'Dead reckoning position estimation.' YouTube video. (May 2025), [Online]. Available: https://youtu.be/UQXmZCBLHkI (visited on 08/05/2025).
[22] P.-Å. Jansson, G. R. and M. Enelund, *Mekanik - Statik och dynamik*. Studentlitteratur AB, 2006.
[23] Pyrodrone. 'T-motor velox victory v3008 cinematic fpv drone motor - 1350kv.' (May 2025), [Online]. Available: https://pyrodrone.com/products/t-motor-velox-v3008-cinematic-fpv-drone-motor-1350kv (visited on 12/05/2025).

A Drawings

In this appendix, the drawings of the components created and used are presented in figures.

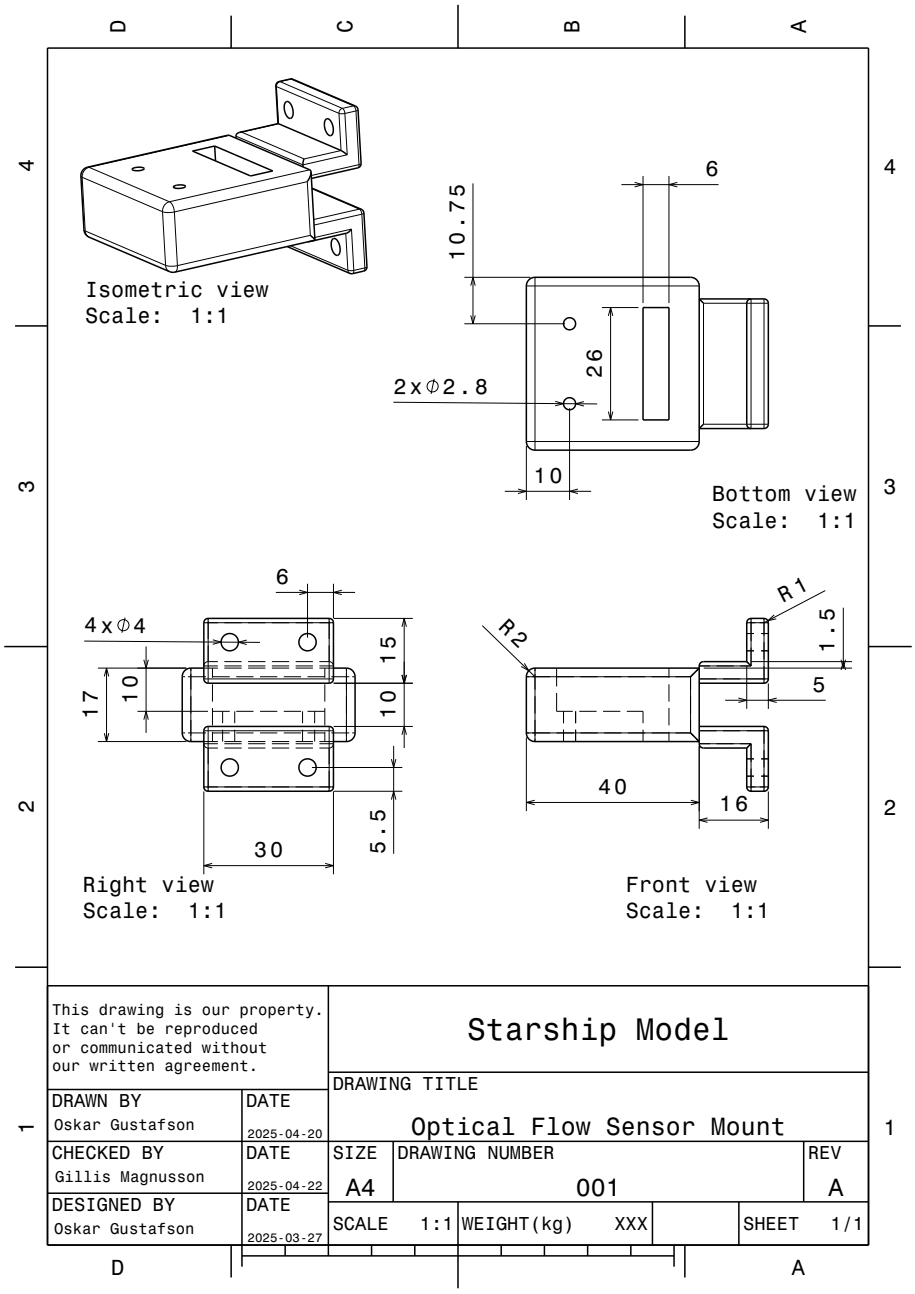


Figure A.1: Drawing of the optical flow sensor mount.

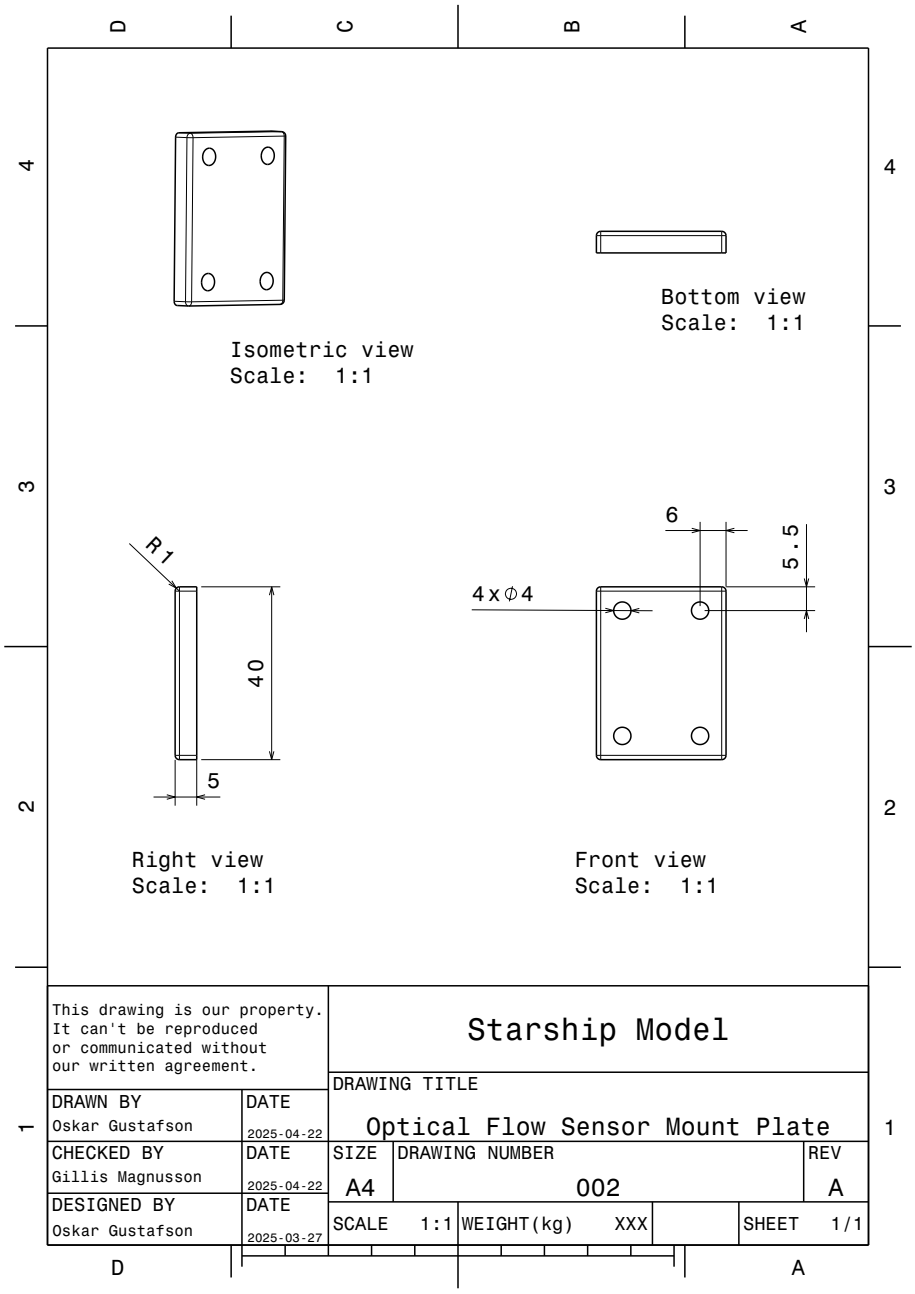


Figure A.2: Drawing of the back plate of the optical flow sensor mount.

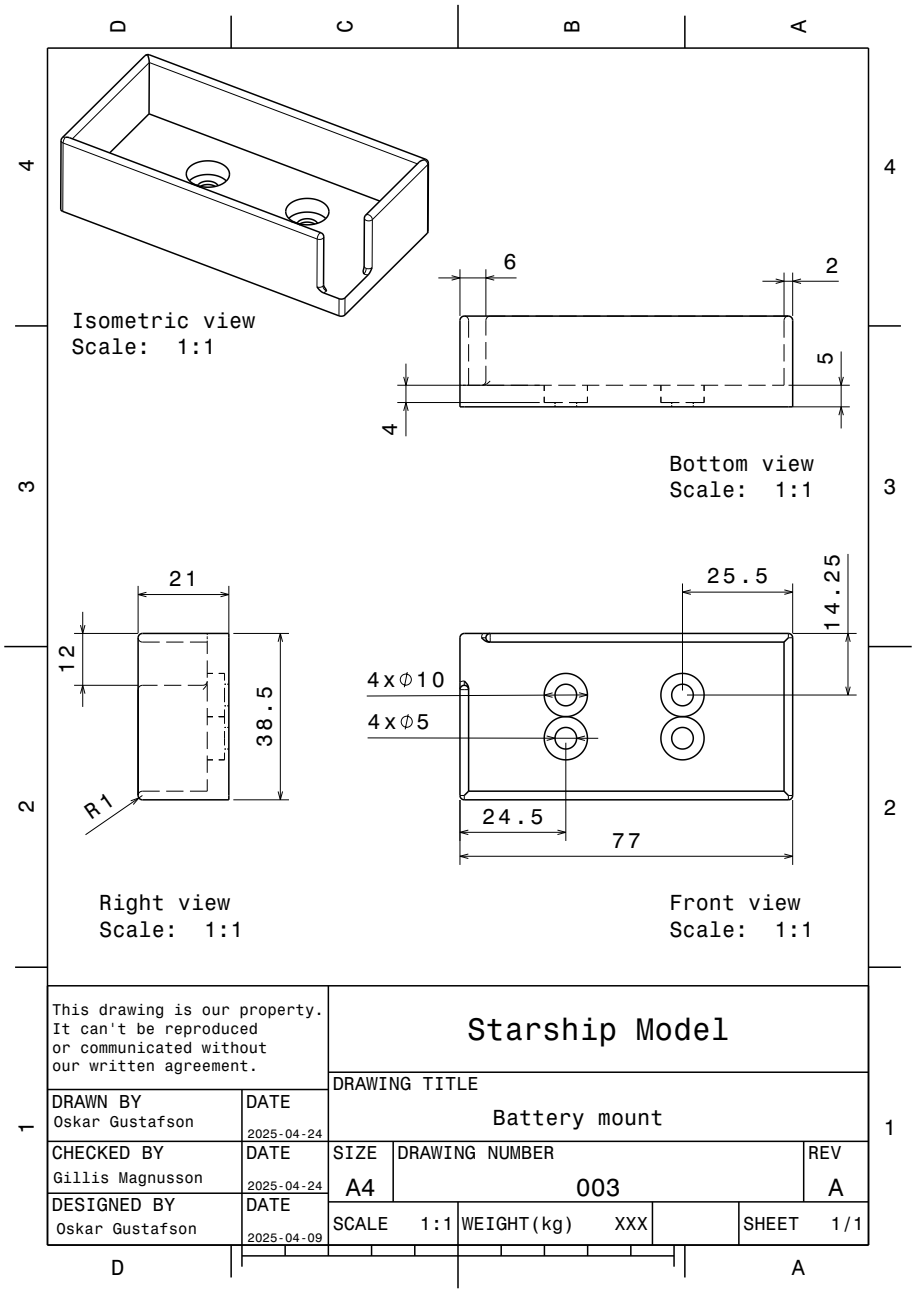


Figure A.3: Drawing of the top battery mount, for the LiPo 2S battery.

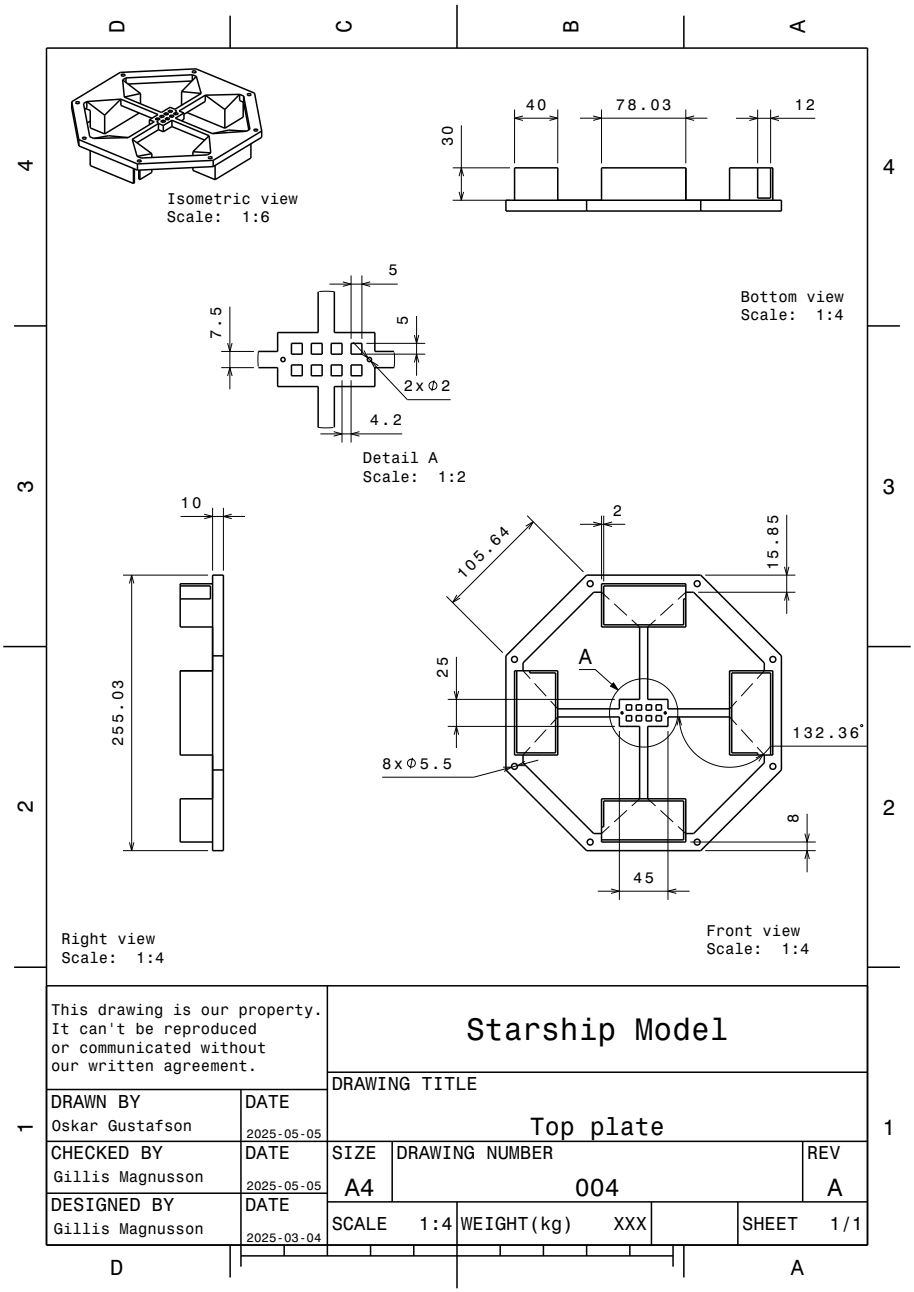


Figure A.4: Drawing of the top plate with the additions of battery casings for the LiPo 3S batteries.

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

CHALMERS