Control of Autonomous Positioning System

Enhancing Crane Load Handling with Modern Control Algorithms

Project Report

E23-ITC5-1

Aalborg University
Department of Energy

Department of Energy
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Autonomous Positioning System

Theme:
Cyber-Physical Systems

Project Period:
Fall Semester 2023

Project Group:
E23-ITC5-1

Participant(s):
Adams Ali Gills
Hubert Dabrowski
Karlis Tregers

Supervisor(s):
Petar Durdevic
Daniel Ortiz Arroyo

Copies: 1

Page Numbers: 78

Date of Completion:
July 2, 2025

Abstract:

As the global shift towards environmentally sustainable energy sources gains momentum. The significance of renewable energy, specifically wind power, is becoming increasingly apparent. Sea-Sight Solutions produces a viable product for stabilizing wind turbine blades during installation. The product positions the wind turbine blades lifted on cranes autonomously. The project report focuses on an alternative control system for smaller applications in the realm of autonomous positioning systems. Subsequently, creating a control system for the model, the system stabilizes the yaw and sway of the lifted load. The resulting system modeling and simulation is created in Matlab and consequently implemented with a prototype. The project succeeds in creating a working control system for the provided model, while also exploring topics for further research.

Contents

Preface

Aalborg University, July 2, 2025

Hubert Dabrowski
<hdabro21@student.aau.dk>

Adams Ali Gills
<agills21@student.aau.dk>

Karlis Tregers
<ktrege21@student.aau.dk>

Chapter 1

Introduction

The Autonomous Positioning System (APS) developed and commercialized by SeaSight Solutions has achieved success within its target market for the installation of wind turbine blades. By effectively eliminating or supplementing tag-lines, the APS has not only simplified the installation process, but also enhanced the versatility and adaptability of these installations.

Given its success in wind turbine installations, there's growing interest in its potential adaptability for broader applications. Specifically, there's curiosity about the feasibility of scaling down this large-scale system for tasks such as lifting containers and managing other smaller-scale objects.

This report explores this very potential. With a focus on delivering a proof of concept for a scaled-down version of the APS including its control system, the project will, in collaboration with SeaSight Solutions, undertake an in-depth analysis of a similar system. Based on the findings, a model and control system will be constructed to ascertain the system's viability in these alternate scenarios.

$$\sqrt{\frac{2}{3+3}}$$

Chapter 2

APS-mini Analysis

This report is a continuation of a previously done project with Sea-Sight Solutions by the group, therefore the specifics of the large-scale commercialized system are mostly left out of this report as the focus will almost exclusively be on the small-scale prototype.

In order to, be able to proceed with a smaller scale model, the foundational characteristics of the prototype must be defined. One of the key elements is defining the scale of the prototype to delimit the components and set realistic targets considering the time and resource constraints.

A document given by SeaSight Solutions has provided specifications for the system's capabilities. These specifications are those of the final Mini-APS product. The report discusses a proof of concept model which can later be improved upon with more robust components to achieve the originally desired results.

Several experiments are conducted to find the correct motor choice according to an arbitrarily chosen load which must be controlled. This is discussed in more depth in a further section. This is done to specifically envision the scale of the prototype.

2.1 Components

The following section reviews a variety of options for prototype development consisting of different available components. Based on this analysis a delimitation of the most suitable components for the given purpose may be done.

2.1.1 Motors & Rotors

A significant part of the large-scale APS solution is the method of varying thrust. In the case of the APS, the speed of the rotor blades is constant, while the pitch of the blades is varied to change the thrust. While this solution is very efficient for such a large-scale model, for smaller applications an alternative option should be evaluated.

Two main options are possible:

- A system based on pitch variation, similar to that used by the full-scale system.
- A system based on varying motor speed with bi-directional capabilities.

Pitch variation

The current APS system uses a separate DC motor to vary the pitch of the rotor blades into either axial direction changing the amount of thrust. This is similar to the design of a helicopter pitching system. [1]

A way of controlling the pitch of helicopter rotors is a swash-plate system. This system has two different functions that can be performed concurrently; adjusting the pitch angle of all rotor blades uniformly and altering the pitch cyclically during each rotor revolution. By changing the pitch uniformly, the helicopter can ascend or descend. Meaning, increasing or decreasing the thrust in the axial direction of the blades. This is the function utilized by the APS. Conversely, by cyclically adjusting the blade pitch, the swashplate system directs the thrust vector in a specific direction, allowing the helicopter to tilt and move laterally. [1]

While cyclical control is not very relevant to the current application, this way of controlling the uniform pitch of the blades is an adequate method of increasing or decreasing thrust. This is also a heavily explored concept due to its wide application in helicopter systems.

Motor speed variation

Contrarily, using a symmetrical propeller design it is possible to simply reverse the direction of rotation of the motor to achieve the opposite thrust. Due to the very large scale of the main APS system, the weight and therefore moment of inertia of the propellers is very high. Because of this, a lot of energy and time would be needed to change the direction of travel. This hence makes the pitching system

more attractive for use. However, due to the project targeting a small-scale solution, this constraint is much less applicable. Although the pitching system would most likely provide a lower response time, the very high complexity and cost of the system makes it unsuitable given the objective of the project.

For the project, brushless DC motors (BLDC) were used because of their affordability and versatility. BLDC motors are also very efficient in terms of the provided torque even at low speeds. This combined with the ability to switch directions quickly makes this motor choice very suitable for the scale of the project.

Two Dualsky EG-12 motors were specifically selected for their capability to meet the project's objectives. A comprehensive analysis, which guided the determination of motor requirements, is elaborated in a subsequent section. It was crucial to strike an optimal balance between the motor's power and the propeller's efficiency to achieve the targeted output thrust. For this purpose, an APC 11x7E propeller was chosen, as it is specifically rated for the type and power of the motor used.

The propeller follows a distinct naming convention. APC 11x7E, is a propeller of diameter 11 inches and pitch 7 inches. (7 inches of travel in the normal direction of the propeller per rotation). As well as the propeller being designed for electric motors, specified by the "E".

The propeller is designed for use in RC planes, however is suitable for the given purpose. Although the propeller design is not symmetrical, meaning the RPM / Thrust ratio is not equal in the forwards and backwards rotation, due to there being two propellers on either side of the APS, the system is still able to adequately perform.

2.1.2 Micro-controller & Sensors

For this project, a combination of a microcontroller and an add-on board of sensors was used. The choice of microcontroller was delimited due to the limited availability of components. An STM32 Nucleo board (STM32F446RE) was used as opposed to an Arduino or Raspberry Pi due to its superior speed.[2] The STM32 also excels at interacting with real-time systems. This is necessary for the processing of data from multiple sensors as well as real-time system control. The Nucleo board was also chosen due to the availability of an add-on board of sensors (IKS01A2). The add-on board is made specifically to be used with boards following a pin-out such as that of the STM32 possessed by the group. It hosts a variety of sensors that were used for the position control of the system.

STM32F446 MCU

The major characteristics can be found on page 13 of the documentation [2]. However some of the more important features of the microcontroller for the scope of the project are listed in the table below.

Feature	Specification
Memory	256 kBytes
Timers	10
GPIO	50 (Including PWM pins)
Clocks	180 MHz
I2C Interface	4/1 FMP +
USART Interface	4/2

Table 2.1: Key Features of the STM32F446RE Microcontroller [2]

IKS01A2 Sensor board

The IKS01A2 contains a variety of sensors mounted on the adapter board. The IKS01A2 compatible with multiple devices including the STM32 MCU which is also developed by ST electronics. The board includes sensors such as: The LSM6DSL 3D accelerometer and 3D gyroscope, the LSM303AGR 3D accelerometer and 3D magnetometer, the HTS221 humidity and temperature sensor and the LPS22HB pressure sensor.

For the purpose of this project not all the provided sensors can be used. Specifically, the humidity, temperature, and pressure measurements can not effectively be used within the scope of the project. As the main use case of the senors is to determine changes in position of the APS prototype. A further analysis of the used sensors will be provided, discussing the purpose of each sensor separately and in combination.

Accelerometer and Gyroscope functionality

In order to fully understand how data is gathered for the project, the inner workings of an accelerometer and gyroscope should be understood. All the sensors used in the sensor board are MEMS (Micro-electro-mechanical systems).

An accelerometer uses a tiny proof mass attached to a spring. When the sensor experiences acceleration, due to the inertia the mass moves. The mass is positioned between fixed electrodes, therefore when moving the distance between the

mass and the electrodes changes, creating a change in the capacitance which is measured and translated into an acceleration.[3]

Similarly, a gyroscope measures the rate of rotation using capacitance. However the main principle is the Coriolis effect. The gyroscope measures the changes the vibration pattern of a vibrating mass.

LSM6DSL Accelerometer

The LSM6DSL Accelerometer can sense linear acceleration in three dimensions. All the sensors have four operation modes, power down, low power, normal and high performance. As these names suggest, it refers to different sampling speeds. This accelerometer has three different types of filters chained together: the analog to digital antialiasing, digital low pass, composite filter. So that it can provide adequate data readings (due to noise reduction) and adjust for drift. The filters used for such sensors are described in a later section.

LSM6DSL Gyroscope

The LSM6DSL gyroscope can sense rotational speed in three dimensions. The highest possible sampling speed which can be achieved by this sensor is 6.66 kHz. This gyroscope has four different types of filters chained together: the analog to digital antialiasing, digital high pass, as well as two digital low pass filters.

LSM303AGR Accelerometer

The LSM6DSL Accelerometer can sense linear acceleration in three dimensions. In these modes the resolution gets respectively better from 8 bits to 10 bits then to 12 bits. the accelerometer itself is able to achieve a sampling rate of around (1.344 kHz) for HR / Normal and (5.376 kHz) for High performance. This accelerometer has a high pass filter and some offset features that can be configured through registers.

LSM303AGR Magnetometer

The Magnetometer can sense the magnetic field in three dimensions. low power can handle a sampling rate of a 150hz and high resolution mode can use 100hz sampling rate. the magnetometer can use a low pass filter. When the low pass filter is enabled it creates a trade-off with the bandwidth it can detect. So when the LP (low pass) filter is off, the bandwidth is half the sampling rate. Otherwise, the bandwidth is one fourth of the sampling rate. Also, an offset can be enabled by editing a register.

2.2 Problem Delimitation

The main goal of the project is to create a control system for a scaled down version of the APS system by Sea-Sight solutions. Along with creating a working prototype to test and verify the validity of the model and control system. The main requirements from Sea-Sight solutions have been defined as reducing the cost-complexity of the project, however the main priority was the proof of concept that such a system is feasible scaled-down. Further delimitation's regarding the modelling and control of the system will be seen in 3.2.

Further seen in 2.2 and 2.1 are the definitions of the axis used for the prototype. The subsequent section also delimits the movement of the prototype that will be considered for this project.

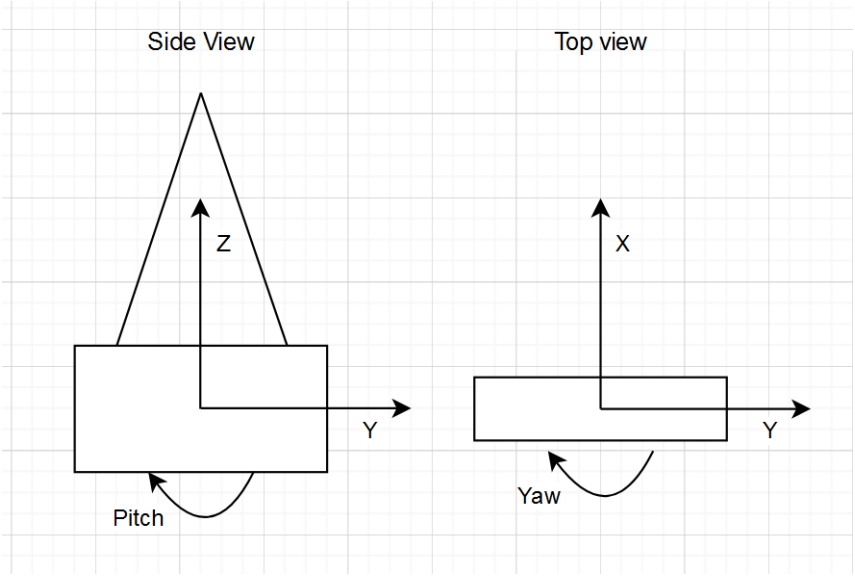

Figure 2.1: Definition of prototype axis

2.2.1 Delimitation of the movement of the prototype

For the purpose of this project, only 4 degrees of freedom are taken into account. That is the translational movement along the X axis (sway), and the rotational movement along the Z axis (yaw). This section will go on to delimit the remaining possible axis of motion in order to simplify the scope of the project.

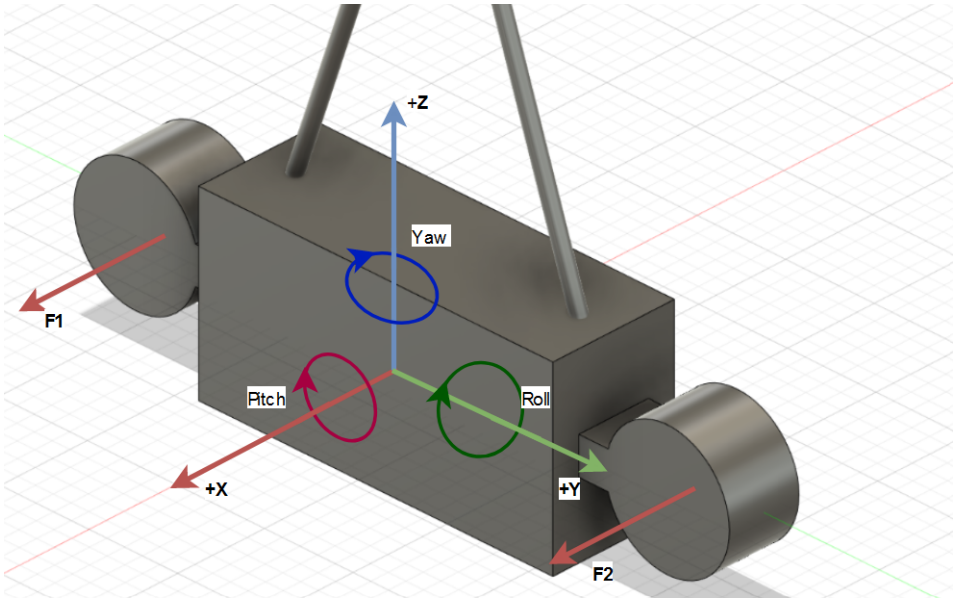

Figure 2.2: 3d representation of the axis definitions

Delimiting roll and pitch

In order to simplify the problem, a reasonable assumption was made. It is assumed that the rotational movement along the Y axis or the pitch is negligible. As the length of the rope and hence the distance from the anchor point is very high compared to the amount of pitch the system will experience, it is assumed that there is no rotational movement along this axis. Instead that the system moves linearly along the y axis. Which is further delimited as the project does not consider the translational movement in y. A graphical representation of the delimitation can be seen in 2.3.

However since the system is made to control sway, it is assumed that the movement in the translational x direction is significant. This movement can be calculated by recording the roll angle of the system and using trigonometry as the length of the rope is known.

Delimitation of movement in translational Y

Similarly, in order to further simplify the problem, it is assumed that the amount of disturbance acting to displace the prototype along the translational y axis is negligible. As the surface area on which the main force (wind) can act is extremely small relative to that in the x direction. Hence the force will not be considered.

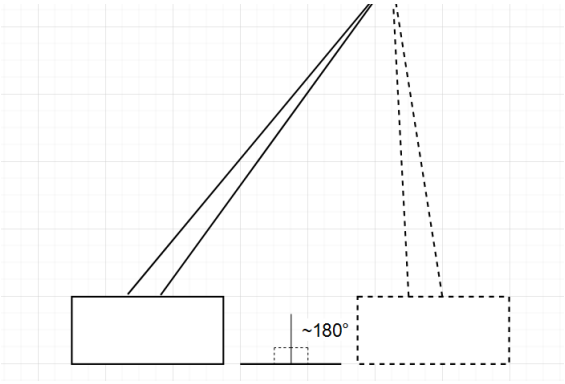

Figure 2.3: Delimitation of system pitching

2.3 Problem Formulation

While considering the defined limitations, a focus of the project can be narrowed down. A Main problem a solution to which can be explored during the project can be formulated. This problem can be summarized as:

> Creating a control system to control the sway and yaw of a model of a scaled down version of the Sea-Sight Solutions APS.

This formal definition helps guide the rest of the project by maintaining a clear direction towards the established goal. The following sections will serve to create a model of the given system and create a control loop to achieve the desired result.

Chapter 3

Prototype design

3.1 Wind disturbance experiment

In order to determine a suitable motor thrust output, several delimitations must be considered. The main consideration for the construction of the prototype is the load capacity or the weight which the system will be designed to stabilize. **This weight was delimited to be a 4kg load.**

To determine the necessary components to stabilize a 4kg load it was decided to conduct several experiments in real-world conditions. The experiment was delimited to only consider a load of a specific box shape with specific dimensions.

A nearby weather station stated that at the time of the experiment, a wind speed of 7m/s was seen with gusts up to 11m/s. These wind conditions will then serve as the control objective for the project. The control system must be able to stabilize the sway of the given load under these conditions.

Experiment analysis

Multiple experiments were conducted with different loads however a load of 4kg was proven to be the most realistic in terms of the time and budget allocated.

The objective of the conducted experiment was to find the resultant acceleration of the APS system caused by wind under certain delimitations:

- Aerodynamics were delimited to those valid for a box of size L50xW16xH39 cm
- Wind speed $7\frac{m}{s}$ with gusts of $11\frac{m}{s}$
- Load of 4kg

3.2 Experimental Considerations

To be able to achieve the goals of the project, the problem must be simplified to fit in the allotted time frame and knowledge. Many reasonable assumptions may be made which do not significantly decrease the accuracy of the results, however simplify the problem and decrease the complexity. This section describes some of the key delimitations and assumptions made.

Delimitation of experiment data

All subsequent sections of the report will refer to the axis and calculations using the body reference frame. However, during the experiments, the data collected uses the sensor reference frame which is not aligned with the axis of the body. Therefore the data is processed, with the report featuring the calculated resultant force which is approximated to be aligned with the body reference frame. However, this calculation is done by using components from both the x and y-axis accelerometer readings.

Calculation of maximum force acting on the load

As previously mentioned the total acceleration is calculated to take into account the force of wind acting to accelerate the load into the x and y directions. The z-direction was not taken into account.

3.2.1 Methods and Experimental Setup

Now that the basis for the understanding of the further experiment data has been established, the report can move on to displaying methods and further findings of the initial experiments.

The load was hoisted using a rope attached to the load at two points, with a rope length of 2.5m (5 meters total combining both attachments). An accelerometer was attached to the load and the acceleration caused by wind disturbances was recorded.

Several parts of the experiment may be seen in 3.1 as the large spike seen at t-182 is indicative of the test load being manually stopped from swaying. This was used as an anchor point when examining the data. Following this, the load was fixated for a few seconds, this can be seen in the few small accelerometer values following. The figure then shows the acceleration in the x and y axis (using the reference frame of the accelerometer) due to wind. Figure 3.2 shows the combination

of the relative x and y forces. The total acceleration experienced in the direction along the translational X axis, see figure 2.2.

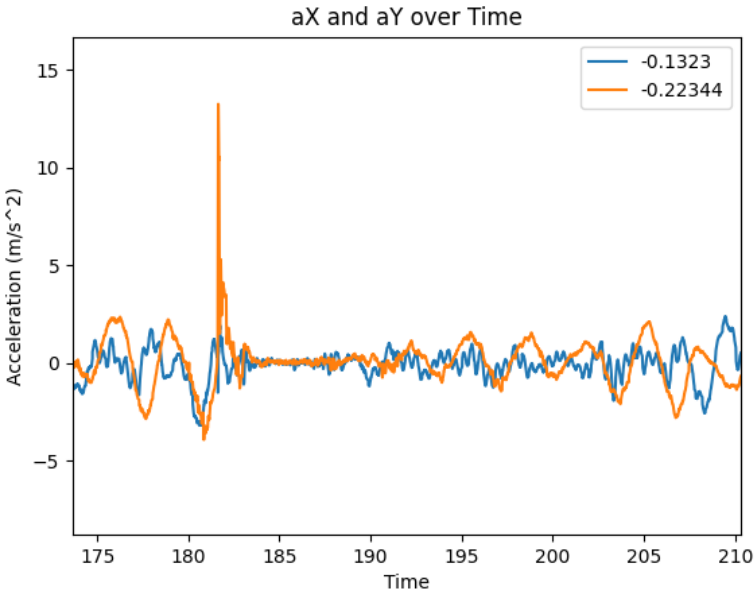

Figure 3.1: Acceleration in x and y direction

Furthermore, the resultant acceleration, denoted as a_T, is defined as the magnitude of the combined accelerations from the two axes: ax, ay. It can be calculated using the Pythagorean theorem as:

$$a_T = \sqrt{ax^2 + ay^2} \tag{3.1}$$

The maximum total acceleration experienced by the load under the current conditions has been 6.57m/s$\hat{2}$, as can be seen in Figure 3.2. This figure is in line with that of the earlier established experiment conditions (See 3.1). Given that the object has a mass m, the force F exerted by the wind on the object can be determined using Newton's second law:

$$F = m \cdot a_T \tag{3.2}$$

This acceleration was then used to calculate the maximum force acting on the load, using newtons second law of motion. With a load of 4kg, this is equal to 26.28N. This formula provides a calculation of the force based on the net acceleration on the object by the wind.

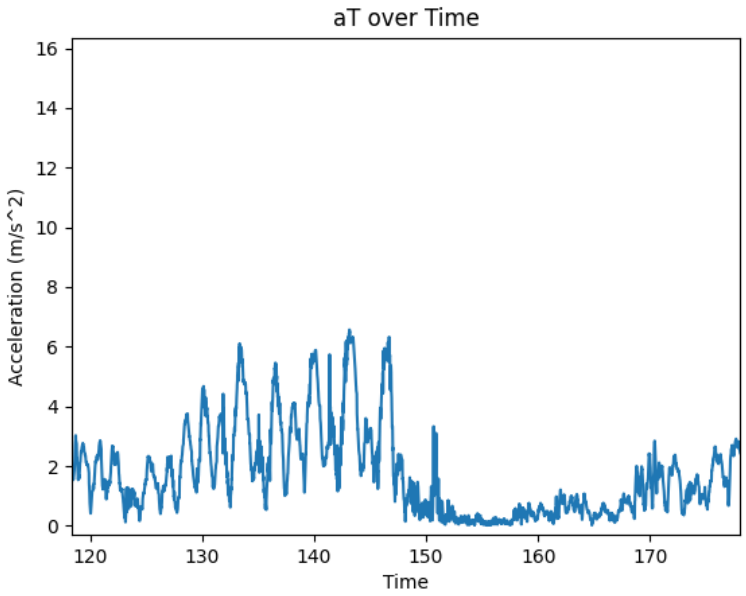

Figure 3.2: Total resultant acceleration over time

3.3 Required thrust and length of a beam

The experiment was used to collect data on specific wind conditions seen in a real-world environment. The collected data has now been used to calculate the force of the chosen load of 4kg. These experimental specifications can now serve as the basis of the prototype. The goal of the project is to control a built prototype fulfilling the previously defined conditions. In order to adequately achieve this goal, necessary components must be chosen such that the control of the load is possible while still maintaining a low cost and high efficiency. The length of the main beam (size of the prototype) must be considered due to the impact of the torque and moment of inertia. As well as the strength of the used thrusters as they must be able to counteract the given wind conditions.

As previously mentioned the project considers the 4 degrees of freedom. Aiming to stabilize the sway and yaw of the system. Each of these must be taken into account.

3.3.1 Sway

In order to adequately control the sway of the system, the thrusts must be able to produce enough force to counteract the worst possible scenario, that being the maximum wind disturbance acting to sway the system. As previously calculated

the thrusters must be able to produce :

$$F = 4kg \times 6.57\frac{m}{s^2} = 26,28N \tag{3.3}$$

3.3.2 Yaw

As the torque generated by thrusters depends on the length of a beam, it is important to consider different sizes. Disadvantages and advantages can be found in a table 3.1. A beam of a length 1m has been chosen because of its compact size suitable for further experiments.

Estimating Torque from Wind

Unfortunately, due to the unavailability of a gyroscope in our experiments, direct measurements of angular acceleration were not possible. However since the sway calculations assume that the wind is acting in the "worst possible scenario", such that all force from the wind is acting to sway the system, the same assumption can be made to calculate the yaw. The theoretical scenario where all the wind force is acting to yaw the system is considered.

$$\alpha \approx \frac{a_{\text{tangential}}}{r} \tag{3.4}$$

where α is the angular acceleration, $a_{\text{tangential}}$ is the tangential linear acceleration, and r is the radius of the circular path (in that case distance from the edge to a yaw rotation midpoint of an object represented in a figure **??**).

Table 3.1: Comparison of Long and Short Beams

	Long Beam	**Short Beam**
Advantages	Lower required force for torque; Increased mechanical advantage; Greater precision in control.	Structural rigidity; Faster response time; Compact and suitable for confined spaces.
Disadvantages	Susceptible to bending; Potential spatial constraints; Increased inertia leading to slower response.	Higher force required for same torque; Reduced mechanical advantage; Potential challenges in precise control.

$$\begin{aligned}\alpha &\approx \frac{a_{\text{tangential}}}{r} \\ &= \frac{6.57\,\text{m/s}^2}{0.25\,\text{m}} \\ &= 26.28\,\text{rad/s}^2\end{aligned} \tag{3.5}$$

Moment of inertia

The moment of inertia I of a rectangular box of dimensions height H 0.39m, width W 0.16m, and length L 0.5m, when rotating around the yaw axis through its center is represented at figure **??** and calculated as:

$$\begin{aligned}I &= \frac{1}{12}m(L^2 + W^2) \\ &= \frac{1}{12} \times 4\,\text{kg} \times ((0.50\,\text{m})^2 + (0.16\,\text{m})^2) \\ &= 0.09187\,\text{kg} \cdot \text{m}^2.\end{aligned}$$

It has been approximated that the moment of inertia of the lifted object used in the experiment is equal to $I \approx 0.1\,\text{kg} \cdot \text{m}^2$.

Estimating required Force from thrusters

In order to estimate the required force, we must firs calculate the torque due to wind:

$$\tau = I \times \alpha \tag{3.6}$$

In order to counteract this torque, the force F_1 and F_2 from the thrusters should produce an equal and opposite torque. For balance:

$$\sum \tau = \tau_{\text{wind}} + \tau_{\text{thrusters}} = 0 \tag{3.7}$$

$$F_1 \times L + F_2 \times L = \tau \tag{3.8}$$

Where L is the distance from each thruster to the yaw axis (half the length of the beam).

It must be noted that due to the design of the prototype the force produced by a thruster is not equal in the forward and backward directions due to a variety of previously described factors. In order to produce the maximum amount of yaw in the system, both thrusters must spin in opposite directions. And hence yawing the system in either direction, one thruster will produce less force than the other.

This thrust difference is further explored in a later section. However, this vastly complicates the development of the prototype and is delimited due to time constraints. Assuming equal thrust distribution for simplicity, $F_1 = F_2 = F$, the above equation simplifies to:

$$2 \times F \times L = \tau \tag{3.9}$$

$$F = \frac{\tau}{2L} \tag{3.10}$$

$$F = \frac{I \times \alpha}{2L} \tag{3.11}$$

$$F = \frac{0.1 \times 26.28}{2 \cdot 0.5} \approx 2.6N \tag{3.12}$$

The required force from the thruster has been estimated to be 2.6N.

3.4 Chosen Thrusters

Based on the analysis represented in chapter 3, the following components have been chosen:

- Propeller 11x7 e - which means a fan of size 11 inches in diameter with a 7-inch pitch. Although this propeller blade is not fully symmetrical, which is not optimal for the purposes of the project, it was deemed suitable due to its availability, low cost.
- Dualsky XM3844EG-12 810kv 16V with theoretical maximum rotational speed equal to 12960RPM. However, the actual maximum RPM is heavily reduced as the theoretical maximum is achieved with no load on the shaft of the motor. Whilst in reality, the attached propeller creates an exponentially higher drag due to air resistance with higher RPM values.

As shown earlier to stabilize the given load, a total of 27N of force must be produced by 2 motors. Assuming that the propellers are spinning in their primary direction, therefore producing the maximum thrust. The required force can be achieved at approximately 8000 RPM, as seen in the table 3.2. In the case that the swaying force is acting in the opposite direction, the RPM required will be higher. However, the mentioned motors were chosen with a safety margin such that, in the worst-case scenario only %80 of the motor thrust is required.

Conclusion

As can be seen from the above estimation, the sway that needs to be controlled is 26.7N, and the yaw is 2.6N. These values show that the force required for sway is much higher than that for the yaw. Hence the motor choice should be made such that enough thrust can be produced to control sway to satisfy all conditions.

Table 3.2: Thrust conversion from lbf to Newtons at different RPMs based on experimental data from producer [4].

RPM	Thrust (lbf)	Thrust (N)
1000	0.051	0.227
2000	0.206	0.916
3000	0.464	2.064
4000	0.826	3.677
5000	1.294	5.757
6000	1.868	8.314
7000	2.549	11.339
8000	3.340	14.861
9000	4.243	18.877
10000	5.259	23.404
11000	6.392	28.432
12000	7.645	34.013
13000	9.021	40.142

Chapter 4

Dynamic Modelling of The System

A dynamic model of a system is created as a basis for understanding and predicting the behavior of the system under various conditions. It will be used to analyze how the prototype will behave dynamically. The following section will explore the theoretical principles and create the basis for the control model.

4.1 Defining Parameters

Firstly the input and output of the system must be defined. As the two BLDC motors are controlled using a PWM signal with a frequency of 50Hz, the duty cycle of the PWM signal of each motor can be used as an input. Secondly as the position of the APS is to be controlled, the angular position of the yaw axis as well as the linear position of the x axis for swaying can be used as the output. Previously defined, constant and known parameters shall be used as the parameters of the transfer functions. Some of these coefficients are listed below.

- Length of the beam.
- Mass and moment of inertia.
- Voltage applied to motors.
- The thrust produced by the propellers.
- Motor constants expressed as the ratio between the angular velocity of the propeller and the voltage applied.
- Time constant of propellers.
- Non-linear function expressed in table 3.2 showing the expected relationship of RPM and thrust.

4.1.1 Mechanical equations

The torque produced by the thrusters can be represented in the following equation:

$$\tau_{\text{net}} = F_1 \times \frac{L}{2} + F_2 \times \frac{L}{2} \tag{4.1}$$

Newton's second law for rotational motion is expressed as:

$$\tau = I \cdot \alpha \tag{4.2}$$

Where I is total the moment of inertia of APS-mini and α rotational acceleration within the yaw axis.

4.1.2 Electro-mechanical equations

$$\omega_{\text{motor}} = KV \cdot V_{\text{PWM}} \tag{4.3}$$

The propeller rotational speed, where KV is the motor constant, showing the theoretical increase in RPM per every 1-volt increment (RPM/Volt) and V_{PWM} is a PWM signal expressed as voltage.

Conversion of PWM signal to analog voltage input signal.

The motors chosen for this project are operated by varying the duty cycle length of a constant frequency PWM signal. Below can be seen the actual PWM values used.

- Neutral PWM: $1500\mu s$
- Max Forward PWM: $2100\mu s$
- Max Reverse PWM: $900\mu s$

For a PWM signal P in microseconds, the corresponding voltage can be calculated by normalizing P around the neutral point ($1500\mu s$) as follows:

$$V_{\text{PWM}} = V_{\text{max}} \cdot \frac{P - 1500}{600} \tag{4.4}$$

Conversion of rotational speed of propeller to thrust

A transfer function needs to be created based on 3.2

4.1.3 Motor thrust transfer function

Only highly non-linear data was available (represented in table 3.2). It was needed to create a transfer function to approximate the given table for it to be included in the further calculations. The input for the transfer function must be ω_{motor} and output should be thrust. To best estimate the relationship of the data, non-linear regression can be used.

Matrix X represents data points of ω_{motor} which is the rotational speed of the propeller expressed in [RPM].

$$X = \begin{bmatrix} {\omega_{\text{motor}}}_1^0 & {\omega_{\text{motor}}}_1^1 & {\omega_{\text{motor}}}_1^2 & \cdots & {\omega_{\text{motor}}}_1^4 \\ {\omega_{\text{motor}}}_2^0 & {\omega_{\text{motor}}}_2^1 & {\omega_{\text{motor}}}_2^2 & \cdots & {\omega_{\text{motor}}}_2^4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ {\omega_{\text{motor}}}_N^0 & {\omega_{\text{motor}}}_N^1 & {\omega_{\text{motor}}}_N^2 & \cdots & {\omega_{\text{motor}}}_N^4 \end{bmatrix} \tag{4.5}$$

The transfer function is given by:

$$f(\omega_{\text{motor}}; w) = \sum_{k=0}^{4} w_k x^k \tag{4.6}$$

Where weights of a function can be found by solving an equation 4.8:

$$\hat{w} = (X^T X)^{-1} X^T t \tag{4.7}$$

The equations have been solved using python code.

The function is estimated to be:

$$f(x) = 4.60 \times 10^{-3} - 8.57 \times 10^{-6} x^1 + 2.32 \times 10^{-7} x^2 \tag{4.8}$$

The result of the given transfer function can be seen in the graph 4.1. Since as previously mentioned the results are non-linear, in order to be included in the dynamic model they must be linearized. This can be done by converting the given second order polynomial into a linear function, represented by a first order transfer function. In order to accomplish this some assumptions must be made in order to achieve the most accurate approximating.

A concrete way to delimit the data while not losing much accuracy in the model is to define an operating range of the model. Looking at the above table 3.2. As the relationship between RPM and Thrust is "exponential", the thrust produced at

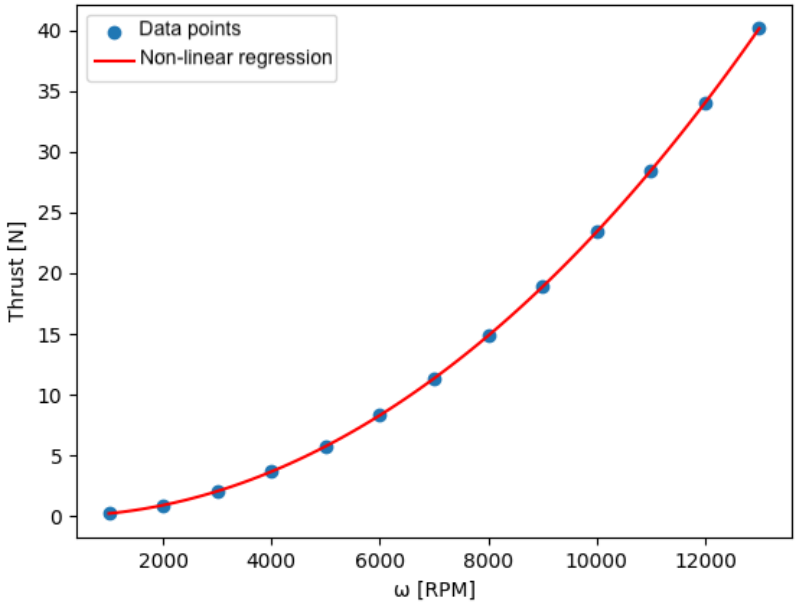

Figure 4.1: Non-linear regression of Thrust to RPM

low RPMs is extremely small, and therefore will not be able to contribute much to accelerating the prototype. Therefore values below 3000 RPM can be disregarded. As previously mentioned, the motors chosen have a safety margin and are more powerful then necessary to accomplish the control objectives. Because of this it is safe to assume that under the defined operating conditions, RPM values above 10000 will not be necessary and can therefore also be disregarded. By delimiting the operating range A more accurate linear approximation of the RPM - Thrust conversion can be achieved.

Creating a linear approximation of the data from 3000 - 10000 RPM based on the data points of the above table, the resulting function is displayed in 4.2. The function is equal to :

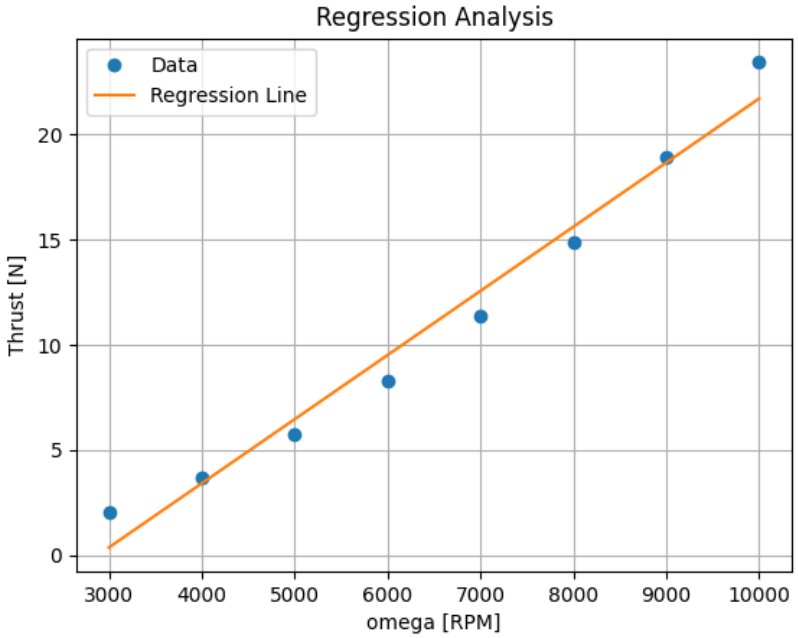

Figure 4.2: Linear Regression of Thrust to RPM

$$F(\omega) = 0.00304 \cdot \omega - 8.75 \tag{4.9}$$

4.2 State-space representation

In the previous section, the report states specific equations that will be further used for the creation of a state space model. Although the previous analysis of the motor speed has been done using the manufacturer specifications, which do not necessarily capture all the dynamics of this particular system, this was not accurate enough for the final model. Therefore although the analysis can help aid understanding, a further experiment was conducted for the specific data of the propellers and motors used in the project. The experimentally gained relationships between thrust and voltage (pwm) will henceforth be used.

4.2.1 States of the system

The following section establishes a state space in order to model the rotational and swaying movement of the system. First, the notation of relevant variables is shown.

States are defined by:

$$x_1 = x \quad \text{(Linear position)} \tag{4.10}$$
$$x_2 = \dot{x} \quad \text{(Linear velocity)} \tag{4.11}$$
$$x_3 = \theta \quad \text{(Angular position)} \tag{4.12}$$
$$x_4 = \dot{\theta} \quad \text{(Angular velocity)} \tag{4.13}$$
$$x_5 = F_{\text{m1}} \quad \text{(Thrust of a motor 1)} \tag{4.14}$$
$$x_6 = F_{\text{m2}} \quad \text{(Thrust of a motor 2)} \tag{4.15}$$
$$\tag{4.16}$$

Angular velocity of motors and thrust

A first order transfer function H(s) will describe the dynamics of the propeller with input $V(s)$ as voltage and output $F(s)$ as Force produced. The transfer function will use experimental data, however as a start, it will use estimated constants.

$$H(s) = \frac{F(s)}{V(s)} = \frac{K}{\tau s + 1} \tag{4.17}$$

Where K is the DC gain of the motor and τ, the time constant. As previously mentioned both the parameters will be found at a later time experimentally.
Further expanding the above equation:

$$F(s) \cdot (\tau s + 1) = V(s) \cdot K \tag{4.18}$$

$$F(s)s\tau + F(s) = V(s) \cdot K \tag{4.19}$$

Taking Laplace inverse:

$$\tau \dot{F}_m(t) + F(t) = K \cdot V(t) \tag{4.20}$$

$$\dot{F}_m(t) = \frac{K \cdot V(t)}{\tau} - \frac{F_m(t)}{\tau} \tag{4.21}$$

Swaying

Now modelling the swaying of the system from the second Newton law, swaying acceleration becomes:

$$\ddot{x} = a = \frac{F_{total}}{m} = \frac{F_1 + F_2}{m} - \frac{F_{spring}}{m} - \frac{F_{air} \cdot \dot{x}}{m} \tag{4.22}$$

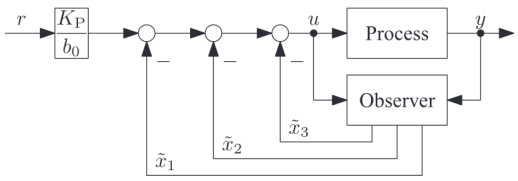

Figure 4.3: Enter Caption

Where F_{total} is a sum of thrust F_1 (the first propeller) and F_2 (second propeller). F_{spring} is the spring force acting by the attachment ropes and k_1 is its spring coefficient. F_{air} is the force caused by air resistance and b_1 is its coefficient.

From the second Newton law, swaying acceleration becomes:

$$\ddot{x} = \frac{F_1}{m} + \frac{F_2}{m} - \frac{k_1 \cdot x}{m} - \frac{b_1 \cdot \dot{x}}{m} \tag{4.23}$$

Yaw rotation

From the second newton low for rotational movement:

$$\ddot{\theta} = \alpha = \frac{\tau}{I} \tag{4.24}$$

Where α is angular acceleration of the APS system and θ is its angular position.

The maximum torque causing yaw rotation is achieved by each thruster acting in opposite directions, hence one force is set to be negative and the total force to yaw the system is found as the difference of the two torques.

$$\tau_{yaw} = \frac{F_1 L}{2} - \frac{F_2 L}{2} \tag{4.25}$$

Combining the two above equations we get:

$$\ddot{\theta} = \alpha = \frac{L \cdot F_1}{2I} - \frac{L \cdot F_2}{2I} - \frac{L \cdot F_{spring}}{2I} - \frac{L \cdot F_{air}}{2I} \tag{4.26}$$

$$\ddot{\theta} = \alpha = \frac{L \cdot F_1}{2I} - \frac{L \cdot F_2}{2I} - \frac{L \cdot k_2 \cdot \theta}{2I} - \frac{L \cdot b_2 \cdot \dot{\theta}}{2I} \tag{4.27}$$

For yaw rotation: F_{spring} is a force acted by the attachment ropes and k_2 is its coefficient. F_{air} is a force caused by air resistance and b_2 is its coefficient.

4.2.2 Differential equations

1. Swaying velocity $\dot{x}$ becomes:
$$\dot{x}_1 = x_2 \tag{4.28}$$

2. 2 Swaying acceleration $\ddot{x}$ becomes:
$$\dot{x}_2 = \frac{F_1}{m} + \frac{F_2}{m} - \frac{k_1 \cdot x}{m} - \frac{b_1 \cdot \dot{x}}{m} \tag{4.29}$$

3. Angular Yaw $\dot{\theta}$ velocity becomes
$$\dot{x}_3 = x_4 \tag{4.30}$$

4. Angular Yaw acceleration $\ddot{\theta}$ becomes:
$$\dot{x}_4 = \alpha = \frac{L \cdot F_1}{2I} - \frac{L \cdot F_2}{2I} - \frac{L \cdot k_2 \cdot \theta}{2I} - \frac{L \cdot b_2 \cdot \dot{\theta}}{2I} \tag{4.31}$$

5. Derivative of thrust produced by the first propeller $\dot{F}_1$
$$\dot{F}_2 = \frac{K \cdot V_2(t)}{\tau} - \frac{F_2}{\tau} \tag{4.32}$$

6. Derivative of thrust produced by the second propeller $\dot{F}_2$
$$\dot{F}_2 = \frac{K \cdot V_2(t)}{\tau} - \frac{x_6}{\tau} \tag{4.33}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -k_1/m & -b_1/m & 0 & 0 & 1/m & 1/m \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -k_2/m & -b_2/m & L/2I & -L/2I \\ 0 & 0 & 0 & 0 & -1/\tau & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/\tau \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K/\tau & 0 \\ 0 & K/\tau \end{bmatrix} \begin{bmatrix} V_1(t) \\ V_2(t) \end{bmatrix} \tag{4.34}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{4.35}$$

What is equivalent to:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{F}_1 \\ \dot{F}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -k_1/m & -b_1/m & 0 & 0 & 1/m & 1/m \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -k_2/m & -b_2/m & L/2I & -L/2I \\ 0 & 0 & 0 & 0 & -1/\tau & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/\tau \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ F_1 \\ F_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K/\tau & 0 \\ 0 & K/\tau \end{bmatrix} \begin{bmatrix} V_1(t) \\ V_2(t) \end{bmatrix} \tag{4.36}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ F_1 \\ F_2 \end{bmatrix} \tag{4.37}$$

4.2.3 State Space Discrete Representation

Later on, to develop a controller on the STM32 MCU the state space representation has to be discretized. This is because microcontrollers process data in time increments. Consequently, the continuous system should be modified by the following[5].

$$x(k+1) = A_d x(k) + B_d u(k), \quad A_d = e^{AT}, \quad B_d = \int_0^T e^{A\tau} B d\tau \tag{4.38}$$

Where $x(k)$ represents the current states and x(k+1) is the states at the next sample. The A_d and B_d are the discrete equivalent of the A and B. The discrete matrices are obtained using the formulas above (4.2.3), with the sampling being T. The full calculation of the discrete system will be done after obtaining the coefficients of the system.

4.3 Mass and Moment of Inertia

To fully model the system, it is essential to determine its specific coefficients. These coefficients provide a quantitative description of the system's physical properties, such as mass and moment of inertia, which are crucial for accurate state space representation.

The total mass of the system is a sum of all its components:

$$m = m_b + 2 \cdot m_c + 2 \cdot m_m + m_L = 6.85\text{kg} \tag{4.39}$$

Here, m_b denotes the mass of the beam, m_c is the mass of each plastic hub casing, m_m represents the mass of each motor, and m_L is the mass of the load.

Moment of Inertia

The moment of inertia, a measure of an object's resistance to changes in its rotation rate, is calculated for each component based on its mass and dimensions:

- Mass of an aluminium beam with electronics attached m_b = 1kg
- Length of the aluminium beam L = 1m
- Mass of a plastic hub casing m_c = 0.280kg
- Length of a Hub L_h = 0.29m
- Mass of a motor m_m = 0.163kg

Each component contributes to the total moment of inertia as follows, the following equation shows the moment of inertia of the beam.

$$I_b = \frac{m_b \cdot L^2}{12} = 0.12, \text{kg} \cdot \text{m}^2 \tag{4.40}$$

The moment of inertia of both hubs combined.

$$I_h = 2 \cdot (m_m + m_c) \cdot \left(\frac{L + L_h}{2} \right)^2 = 0.68\,\text{kg} \cdot \text{m}^2 \tag{4.41}$$

The moment of inertia of a load modeled as a box of width a, length b, and mass m_L:

$$I_L = \frac{1}{12} \cdot m_L \cdot (a^2 + b^2) = 0.13\,\text{kg} \cdot \text{m}^2 \tag{4.42}$$

Assuming the box has a mass of 5kg and dimensions of 0.4m x 0.4m, I_L is considered negligible for this project.

The total moment of inertia, combining all components is then:

$$I = I_b + I_h + I_L = 0.93\text{kg} \cdot \text{m}^2 \tag{4.43}$$

These coefficients, particularly the total mass and moment of inertia, are fundamental in defining the dynamical behavior of the system. They will be utilized in the state-space model to predict and control the system's response to various inputs, ensuring accurate and efficient performance of the control system.

4.4 Motor Dynamics

To identify the non-linear dynamics of a propeller, a stair-step response has been recorded for both directions. Each step size was 2 seconds long. This time was considered adequate as the prototype could reach a steady state in a much smaller time frame. Hence it could be assumed that no uncertainties or dynamics were inherited from the previous step. The motors also have a dead zone, meaning with a small enough input signal, the motor will not spin. The size of this dead zone was also determined using this experiment. A separate ramp response was recorded for this purpose.

Data acquisition

To acquire the data, a test setup was used. Specifically, the RC benchmark 1580 to measure the thrust output of such motors. The test setup was calibrated using known weights, subsequently, the motor was mounted upon the test stand. The test device contains load cells connected to a processing board. The load cells can accurately measure the thrust output of the motor.

4.4.1 Signal boundaries

To find the minimum PWM which causes the motor to spin, a ramp response has been recorded with an initial value of the pwm duty cycle width being set to 1500μs.

Forward thrust

The experiment was conducted, the results of which can be seen for the forward direction (positive direction) in Figure 4.4. It can be seen on the plotted data 4.4 that the motor starts producing thrust at PWM value of 1528. Values of input between 1500-1528 do not produce any rotation or thrust. The noise seen on the plotted data is due to inaccuracies of the load cells and should not be considered.

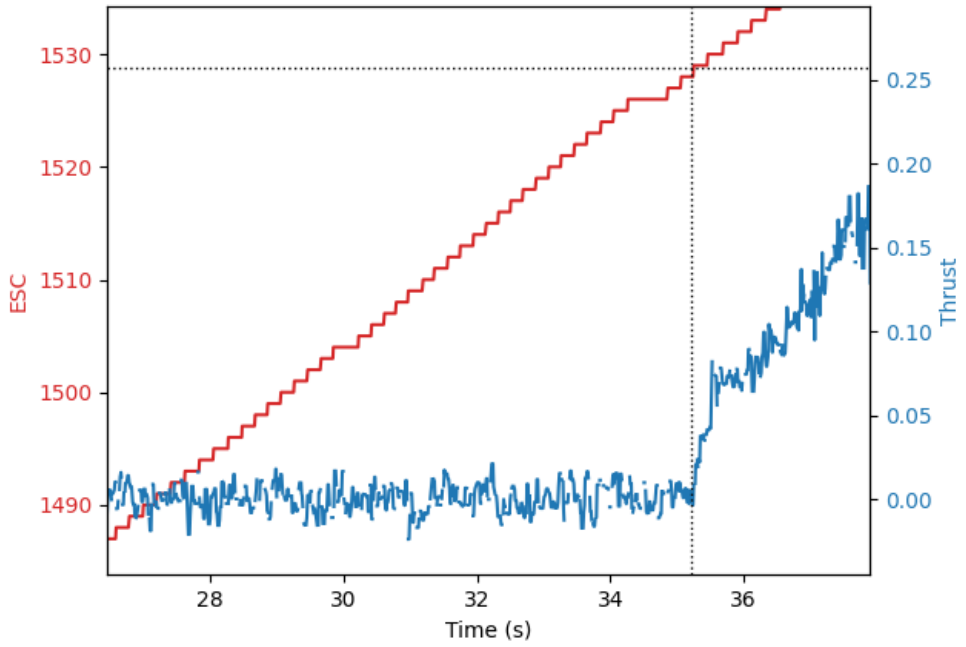

Figure 4.4: Experiment data from Forward direction

Reversed thrust

A similar experiment was conducted in the reverse direction. It can Observed that thrust started increasing at the the time marked by the vertical dotted line on plot 4.5 when the input signal was decreased to around 1480. Values of the input between 1500-1480 do not produce any rotation or thrust.

4.4.2 Dynamics identification

Depending on the chosen operating range, many step responses showing the different dynamic relations from the input signal to the output thrust can be used. Hence, to better estimate the time taken for the system to reach the steady state of a desired value, the experiment was conducted. Raw data recorded during a stair-step response is represented in plot 4.6 for the forward direction and in plot 4.7 for the reversed direction.

Data Normalisation

To better analyze the data relation and remove unwanted offsets, data normalization was performed. The resultant processed data can be seen in plot 4.8. To

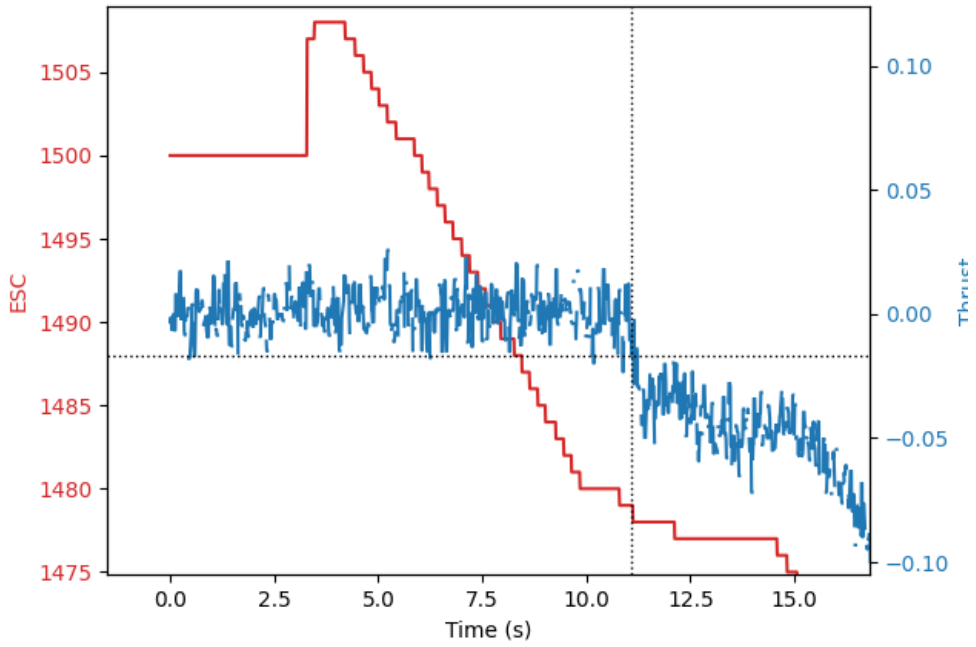

Figure 4.5: Experiment data from Reverse direction

normalize the data, multiple steps were taken, firstly, Mix-Max normalization was performed for the input (PWM signal). Since the original PWM values taken by the motors range from 900 to 2100 signal width, from maximum thrust backward to maximum thrust forwards. This is already artificially delimited to range from 1000 to 2000 for ease of calculations as well as to accommodate for the battery limitations. For the signal to be more accessible for the later developed control system, the signal was mapped from {1000 : 2000} to {-1 : 1}. Where a signal width of 1500 is equal to zero with the propeller being stationary. Also, the means of the output were removed for each interval to more easily view the relative change of the output.

Parameters by input interval

The time constant τ is going to be found by finding the required time to reach 63.3 percent of the steady-state value. This is done for all the different intervals, the result of this can be seen in the below table. The table shows a comparison between the different values depending on the interval taken.

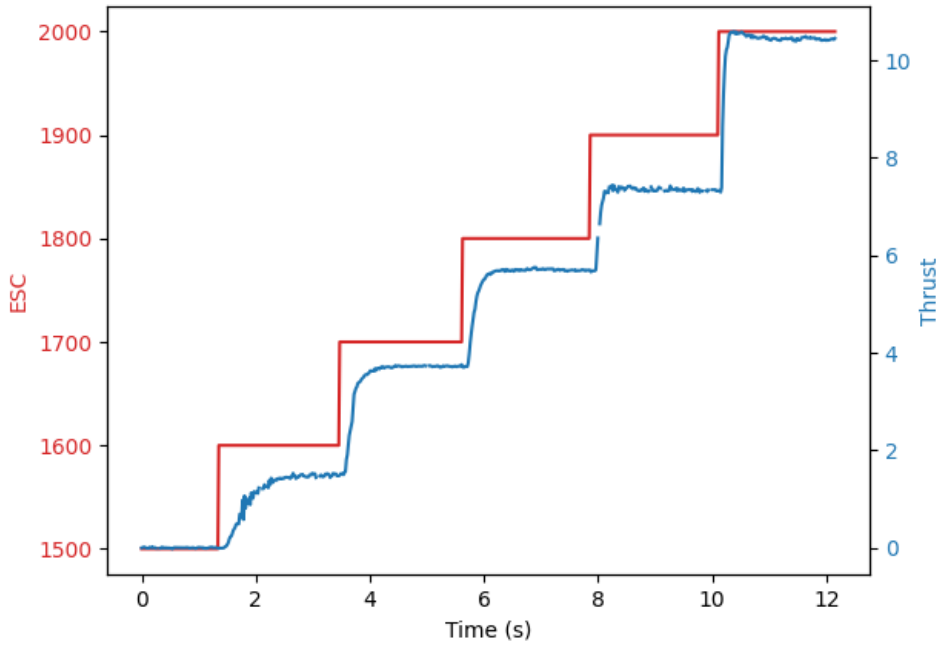

Figure 4.6: Cleaned data from forward direction

Input[μs]	Norm. input	DC Gain [N]	63.2% of DC gain [N]	Time Const. [s]
1500-1600	0.0-0.2	1.472	0.930304	0.445
1600-1700	0.2-0.4	2.200	1.390400	0.250
1700-1800	0.4-0.6	2.070	1.308240	0.239
1800-1900	0.6-0.8	1.720	1.087040	0.190
1900-2000	0.8-1.0	3.080	1.946560	0.100

Table 4.1: Experiment result derived parameters

4.4.3 Linear thrust dynamics

It has been found that the dynamics of the propeller are highly non-linear. However in order to implement the dynamics into the model, the non-linearities must be eliminated by linearizing. There are quite a few reasons for the existing non-linearities. Firstly, the utilized propellers are asymmetrical. hence, the thrust provided in the forward and reverse directions is not equal. These limitations are further expanded by the design of the mounting hubs, which contain support pillars. To provide structural rigidity, this structure inhibits the flow of air in the reverse direction even more. Because of this, the DC gain can not be the same for both directions. Secondly, as discussed in the previous section, the time constants

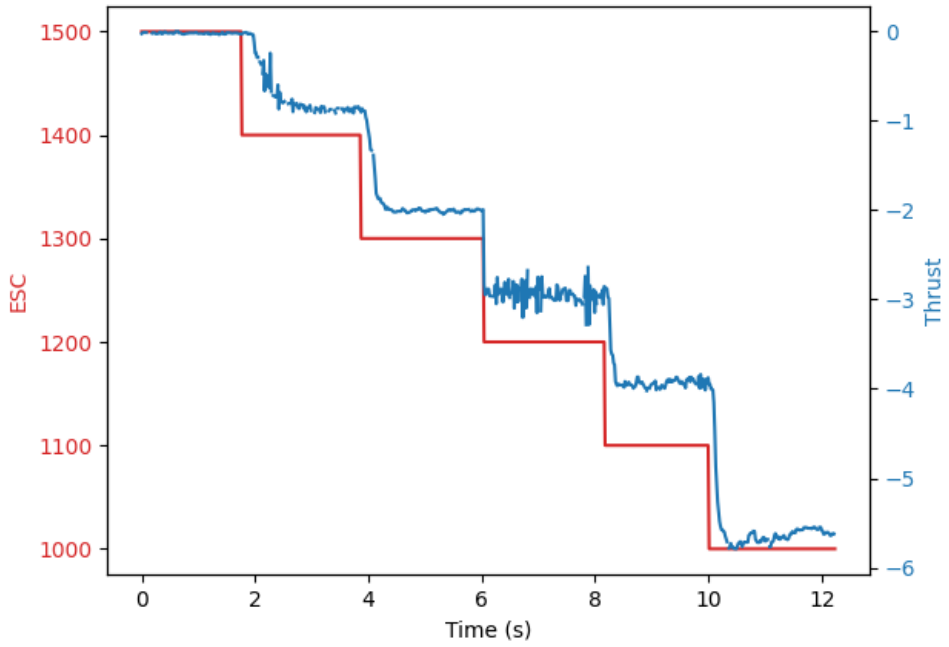

Figure 4.7: Cleaned data from reverse direction

differ depending on the input intervals. As well as the BLDC motor containing non-linear dynamics by nature.

Assumptions for linearisation

To delimit these non-linearities, some assumptions are made.

- Operating range for linearisation is going to be -0.40 to 0.40.
- Linear dynamics are described by a first-order transfer function.
- For both reversed and forward thrust, gains and time constant for forward direction is going to be used.

Data transformation for linearisation

For the chosen approximation method, the time constant for the first step has been used which is equal $\tau = 0.445$ which has been found for a range input 0-0.2. This approach was taken as the initial input range which is a more accurate representation of motor dynamics. Moreover, the control system will often need to efficiently respond to small deviations from the setpoint making the initial time constant

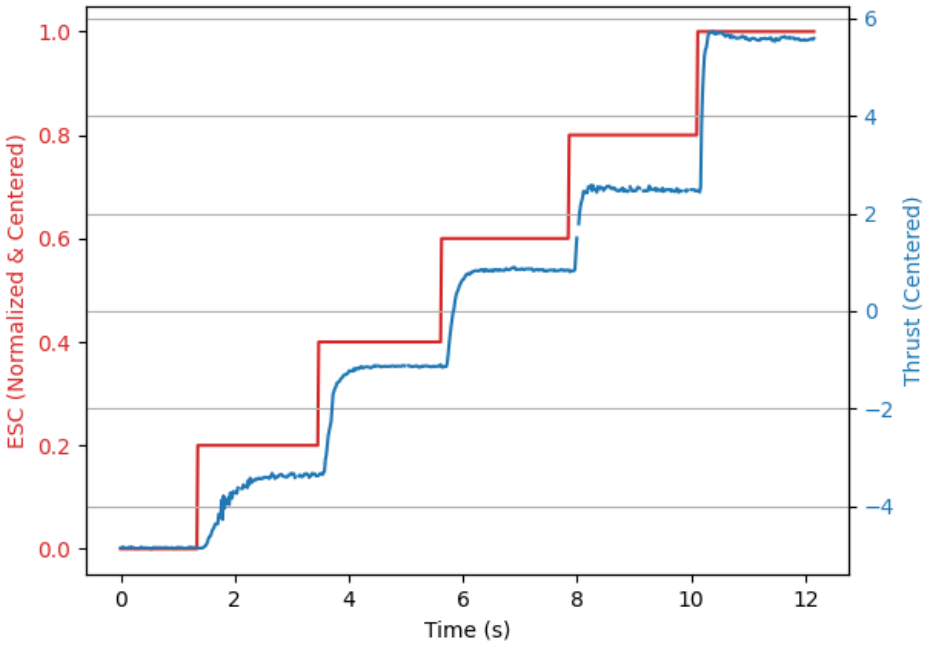

Figure 4.8: Cleaned data from forward direction

more reliable. However, the gain calculated over a wider input range (0-0.4) may provide a more accurate representation of the motor's steady-state behavior.

Combining parameters recorded from different operating ranges can give more comprehensive features of the motor's dynamics, accounting for both transient and steady-state behaviors. [6]
The DC gain is equal to:

$$K = DCGain = \frac{\triangle Output}{\triangle Input} = \frac{3.672}{0.4} = 9.18 \tag{4.44}$$

It has been approximated that the transfer function describing the linear relationship between Signal input (expressed in a range from -1 to 1) is equal to:

$$H(s) = \frac{K}{\tau s + 1} = \frac{9.18}{0.445s + 1} \tag{4.45}$$

4.5 Damping and restoring force experiments

The experiment was conducted by using a gyroscope and an accelerometer data of the prototype. This data was acquired in 2 scenarios.

- The prototype was deviated from its zero position in the yaw axis.
- The prototype was deviated from its zero position in the sway axis.

As such, the corresponding damping ratio and natural frequency of each situation was found. This ultimately can provide the spring force and air resistance coefficients that are included in the state space model.

4.5.1 Swaying

To find coefficients k and b which correspond to Spring (Restoring) force and Air resistance (Damping force) an experiment needs to be conducted and a second-order transfer function is identified with input as initial deviation and output as position. A second-order transfer function is given by:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$H(s) = \frac{x}{x_{in}} \tag{4.46}$$

Rearranging the second order transfer function where the input is the initial position x_{in} (deviation) and the output; the sway offset x.

$$x_{in}\omega_n^2 = xs^2 + 2\zeta\omega_n xs + x\omega_n^2 \tag{4.47}$$

Taking the Laplace inverse:

$$x_{in} = \ddot{x} + 2\zeta\omega_n\dot{x} + x\omega_n^2 \tag{4.48}$$

$$\ddot{x} = -\dot{x}2\zeta\omega_n - x\omega_n^2 + x_{in} \tag{4.49}$$

It can be observed that the coefficient $b_1 = \omega_n^2 \cdot m$ and the coefficient $k_1 = 2\zeta\omega_n \cdot m$

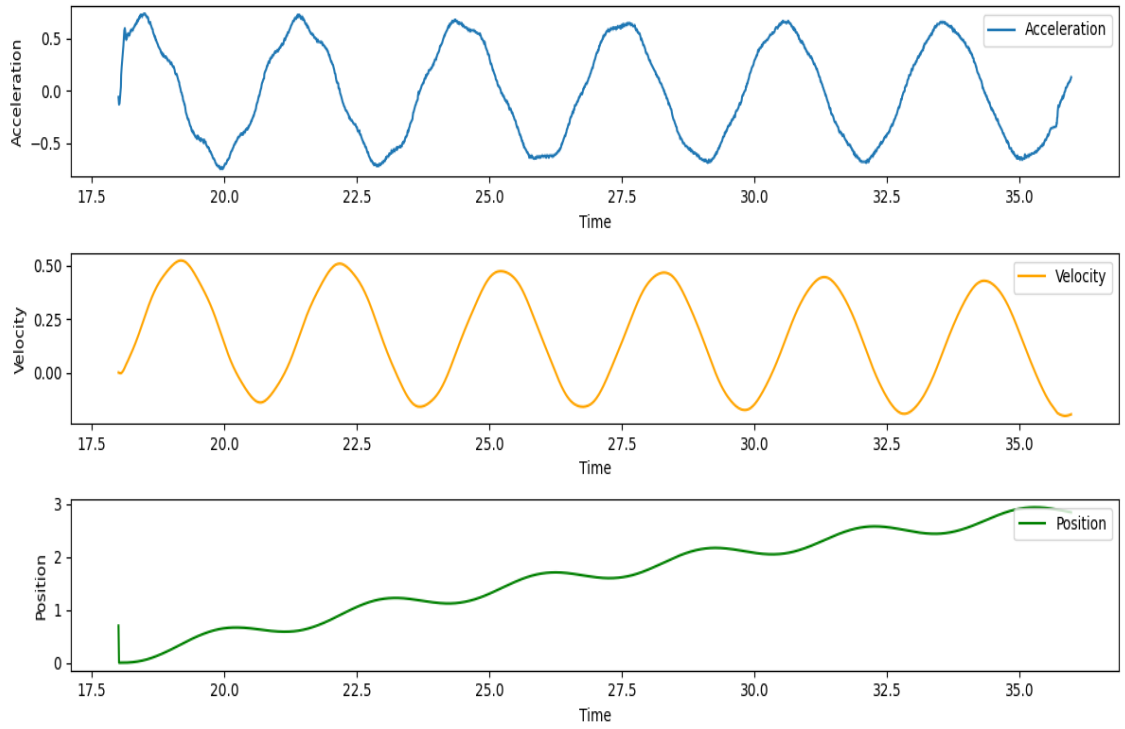

Figure 4.9: Accelerometer with Integration

Based on the results: period $T = 2.935s$ and natural frequency $\omega_n = \frac{2\pi}{2.935s} = 2.14$

It can be considered that $\zeta = 0$, because the system oscillates with the natural frequency.

$$k_1 = \omega^2 \cdot m = (2.14 rad/s)^2 \cdot 6.85 kg = 31.37 \tag{4.50}$$

4.5.2 Yaw rotation

On the other hand, the transfer function for the yaw can be given similarly, except using the initial angle (θ_{in}) as the input and output being the deviation (θ).

$$H(s) = \frac{\theta}{\theta_{in}} \tag{4.51}$$

Rearranging the second order transfer function :

$$\theta_{in}\omega_n^2 = xs^2 + 2\zeta\omega_n\theta s + \theta\omega_n^2 \tag{4.52}$$

Taking a Laplace inverse we get:

$$\theta_{in} = \ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \theta\omega_n^2 \tag{4.53}$$

$$\ddot{\theta} = -\dot{\theta}2\zeta\omega_n - \theta\omega_n^2 + \theta_{in} \tag{4.54}$$

We can observe that coefficient $b_2 = \omega_n^2 \cdot I$ and coefficient $k_2 = 2\zeta\omega_n \cdot I$

x_{in} and $theta_{in}$ need to be chosen according to the most probable operating range.

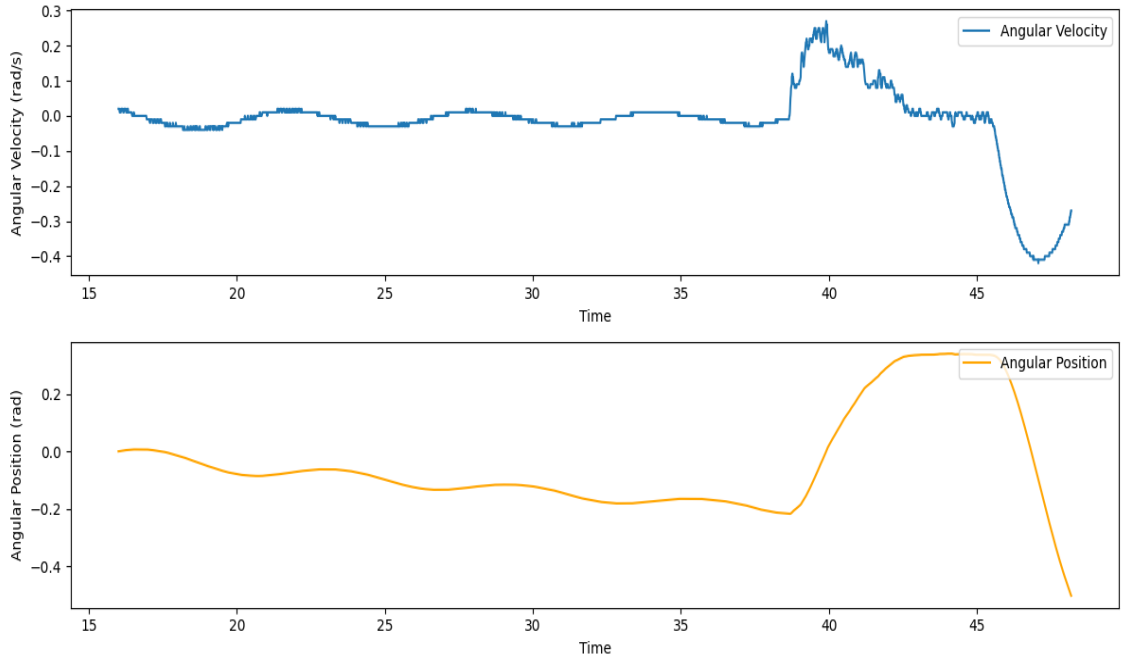

Figure 4.10: Gyroscope Data with integration

Based on the results $T = 6.5s$ and $\omega = \frac{2\pi}{6.5s} = 0.967$

Since again the system oscillates with the natural frequency, the ζ is zero.

$$k_2 = \omega^2 \cdot I = (0.967rad/s)^2 \cdot 0.93 = 0.869 \tag{4.55}$$

4.6 Final models Continuous and Discrete

After Obtaining the relevant coefficients, The Final Continuous model is represented as,

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{F}_1 \\ \dot{F}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -31.37/6 & 0 & 0 & 0 & 1/6.85 & 1/6.85 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -0.896/0.93 & 0 & 1/(2 \cdot 0.93) & -1/(2 \cdot 0.93) \\ 0 & 0 & 0 & 0 & -2.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2.25 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ F_1 \\ F_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 20.63 & 0 \\ 0 & 20.63 \end{bmatrix} \begin{bmatrix} V_1(t) \\ V_2(t) \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ F_1 \\ F_2 \end{bmatrix}$$

Due to the need for Discrete control in the microcontroller, the formula is applied 4.38 using the c2d function in Matlab to discretize the state space to a form represented below:

$$x_{n+1} = Ax_n + Bu_n$$
$$y_n = Cx_n + Du_n$$

```
sysd = c2d(sys, sampleTime, 'zoh');
```

Listing 4.1: c2d function

So, in this function, all the continuous matrices are used and this includes sampling time which is 100Hz or 0.01s. The final parameter shows that the ZOH (zero order hold) method is used. The ZOH is a well-documented standard way of discretizing a continuous system[7]. The method holds the same value until the next sample is sampled. As such, the A and B transform is:

$$A_d = \begin{bmatrix} 0.9772 & 0.0992 & 0 & 0 & 0.0007 & 0.0007 \\ -0.4545 & 0.9772 & 0 & 0 & 0.0130 & 0.0130 \\ 0 & 0 & 0.9953 & 0.0998 & 0.0025 & -0.0025 \\ 0 & 0 & -0.0933 & 0.9953 & 0.0481 & -0.0481 \\ 0 & 0 & 0 & 0 & 0.7987 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.7987 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0.0005 & 0.0005 \\ 0.0139 & 0.0139 \\ 0.0017 & -0.0017 \\ 0.0515 & -0.0515 \\ 1.8476 & 0 \\ 0 & 1.8476 \end{bmatrix}$$

$$C_d = C$$
$$D_d = 0$$

4.7 Model analysis

To have a strong foundation for further control system development, model analysis is performed.

4.7.1 Controlability

To control the system for any given initial state efficiently, there always needs to exist a piecewise continuous input signal such that within a finite period the LTI system will reach the original point from the initial state[8]. This property can be checked by the rank of the controllability matrix defined as:

$$T_C = [B \quad AB \quad A^2B \quad \dots \quad A^5B] \tag{4.56}$$

Where for given model:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -4.58 & 0 & 0 & 0 & 0.146 & 0.146 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -0.934 & 0 & 0.54 & -0.54 \\ 0 & 0 & 0 & 0 & -2.24 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2.24 \end{bmatrix} \tag{4.57}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 20.63 & 0 \\ 0 & 20.63 \end{bmatrix} \tag{4.58}$$

So the controllability matrix will be:

$$T_C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -6.2 & -4.1 & 13.9 & 9.2 & -6.7 & 1.9 \\ 0 & 1 & 0 & 0 & 0 & 0 & -4.1 & -6.2 & 9.2 & 13.9 & 1.9 & -6.7 \\ 0 & 0 & 1 & 0 & 0 & 0 & -2.8 & -1.8 & 0.0 & 0 & 10.9 & 10 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1.8 & -2.8 & 0.0 & 0 & 10 & 10.9 \\ 0 & 0 & 0 & 0 & 1 & 0 & -2.3 & 0 & 2.3 & -1.8 & -5 & 4.1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -2.3 & -1.8 & 2.3 & 4.1 & -5.2 \end{bmatrix}$$

We can observe that the above matrix is full rank and therefore is controllable.

4.7.2 Observsability

Observability defines if it is possible to find all states of a system from measured outputs. It means that any initial state can be determined from input U and output Y over a finite time interval.[9]

The observability matrix is defined as:

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^5 \end{bmatrix}$$

Where for given model: A: 4.57 and C = $\begin{bmatrix} 1 & 0 & 0 & 0\,0 & 0 \\ 0 & 0 & 1 & 0\,0 & 0 \end{bmatrix}$

The reduced Column echelon form of the observability matrix is

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.59}$$

Matrix O is full rank therefore system observable.

4.7.3 Stability

To better understand the model and its dynamics, it is imperative to observe the stability.

Continuous time system

The stability of a state space system is given by its eigenvalues, particularly the eigenvalues of the A matrix.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{k_1}{m} & -\frac{b_1}{m} & 0 & 0 & \frac{1}{m} & \frac{1}{m} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{k_2}{m} & -\frac{b_2}{m} & \frac{L}{2I} & -\frac{L}{2I} \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau} \end{bmatrix},$$

So we take the determinant of the characteristic equation

$$\det(A - \lambda I) = \begin{bmatrix} -\lambda & 1 & 0 & 0 & 0 & 0 \\ -\frac{k_1}{m} & -\frac{b_1}{m} - \lambda & 0 & 0 & \frac{1}{m} & \frac{1}{m} \\ 0 & 0 & -\lambda & 1 & 0 & 0 \\ 0 & 0 & -\frac{k_2}{m} & -\frac{b_2}{m} - \lambda & \frac{L}{2I} & -\frac{L}{2I} \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau} - \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau} - \lambda \end{bmatrix} = 0.$$

$$\lambda = \begin{bmatrix} 0.00 + 2.14i \\ 0.00 - 2.14i \\ 0.00 + 0.96i \\ 0.00 - 0.96i \\ -2.2472 + 0i \\ -2.2472 + 0i \end{bmatrix}$$

Solving for λ will give us the eigenvalues of matrix A. The dominant poles exhibit marginal stability as they are on the imaginary axis. To further observe the stability type, more calculations will be assessed.

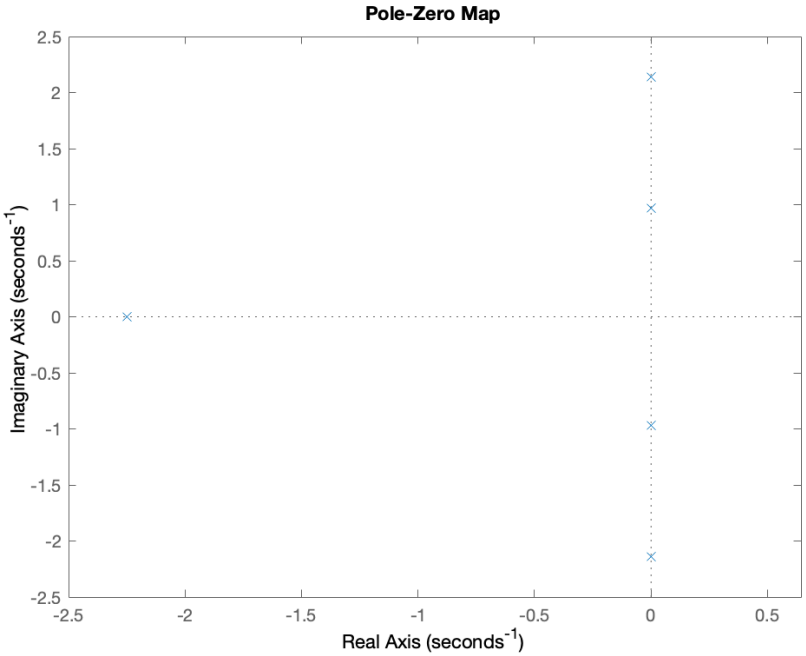

Figure 4.11: Pole Zero Map for continuous model

Lyapnov Stability

As each of the eigen-values on the imaginary axis are unique numbers even though they are 2 conjugate pairs, the multiplicity on the imaginary axis is 1. Meaning that each eigenvalue only appears once as the root of the characteristic equation. Next, the nullity or the dimension of the null space of this matrix is computed as

$$(A - \lambda_m I) = m = 1$$

The system is i.s.l stable (lyapnov). This proves that the system at some initial value remains near the system equilibrium point. Given an initial value, the system will never exceed a certain boundary. However, an objective boundary has to be defined to truly call it i.s.l stable. All in all, the equation above shows that the system stays near the equilibrium point (which is zero) without going to infinity. As for Asymptotic stability (the system goes to equilibrium as time goes to infinity), it can not be sufficiently proved as dominant poles are zero up to the 4th decimal place, and higher resolution calculations for the eigenvalues have not been made[10].

Discrete system

In a similar fashion, the eigenvalues are computed for the discrete system. However, they are evaluated based on the z domain. the eigenvalues are the following:

$$\lambda = \begin{bmatrix} 0.9998 + 0.0214i \\ 0.9998 - 0.0214i \\ 1.0000 + 0.0097i \\ 1.0000 - 0.0097i \\ 0.9778 + 0.0000i \\ 0.9778 + 0.0000i \end{bmatrix}$$

These eigenvalues indicate that the poles are basically on the unit circle, so like the continuous system, it would be classified as a marginally stable system. From a more visual perspective, the z-domain plot is given

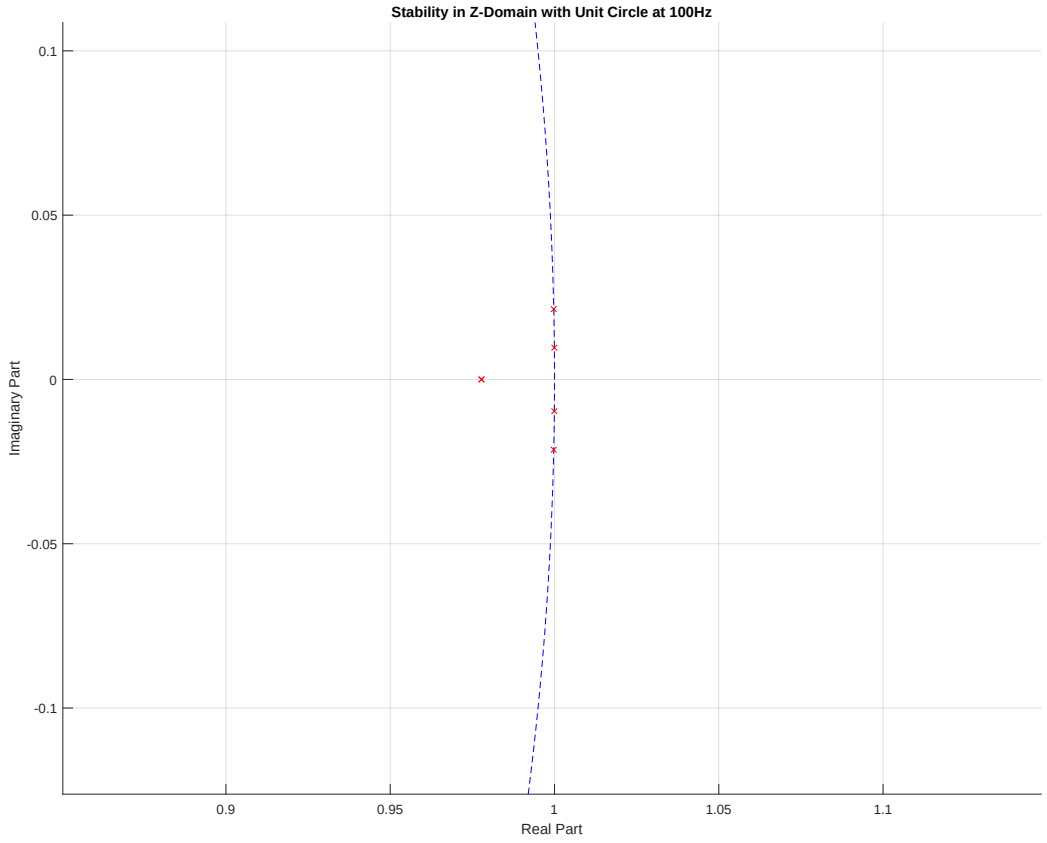

Figure 4.12: Z domain poles for 100hz

4.7.4 Frequency analysis

Determining the bandwidth of a marginally stable plant can be challenging because marginally stable systems have poles on the imaginary axis in the s-plane. That system does not settle down to a steady state state in its response. However other approaches can be taken to find a good sampling ratio for a system. The frequency at which the system gets a significant gain is going to be found. The sampling ratio should be many times higher than the frequency at which the system gets high gains. However, the system is considered to be slow so it is not a challenge.

Nyquist condition

The Nyquist frequency is half the sampling rate of the system. It is the highest frequency that can be accurately represented in a discrete system. For the chosen sampling speed 100hz, that frequency is equal to 50hz. It should be multiple times higher than the frequencies of the system, where it reaches significant gains.

4.7.5 Bode analysis

As the system is multiple input, multiple output. There will be multiple bode plots used for the frequency analysis, specifically the yaw rotation and sway are viewed separately.

Yaw rotation

The transfer function for yaw rotation is considered to be:

$$Yaw(s) = \frac{Output_2}{Input_1} = \frac{11.09}{s^3 + 2.247s^2 + 0.9344s + 2.1} \tag{4.60}$$

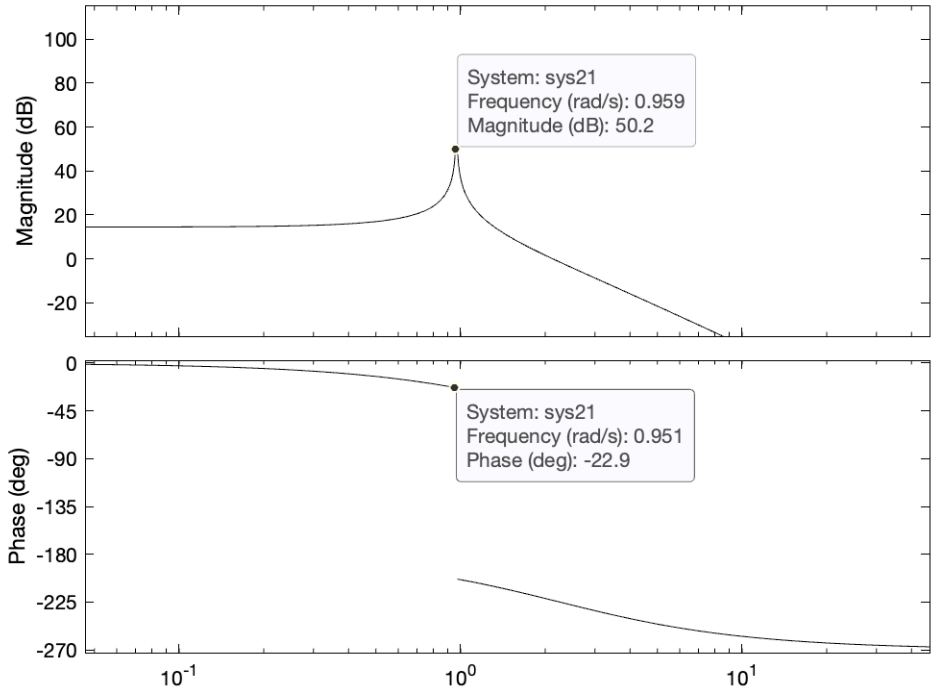

Figure 4.13: Bode plot for yaw rotation

The gain peak for yaw rotation can be observed at a frequency of 0.956 rad/s.

Swaying

The transfer function for swaying movement is considered to be:

$$Yaw(s) = \frac{Output_1}{Input_1} = \frac{3.012}{s^3 + 2.247s^2 + 4.58s + 10.29} \tag{4.61}$$

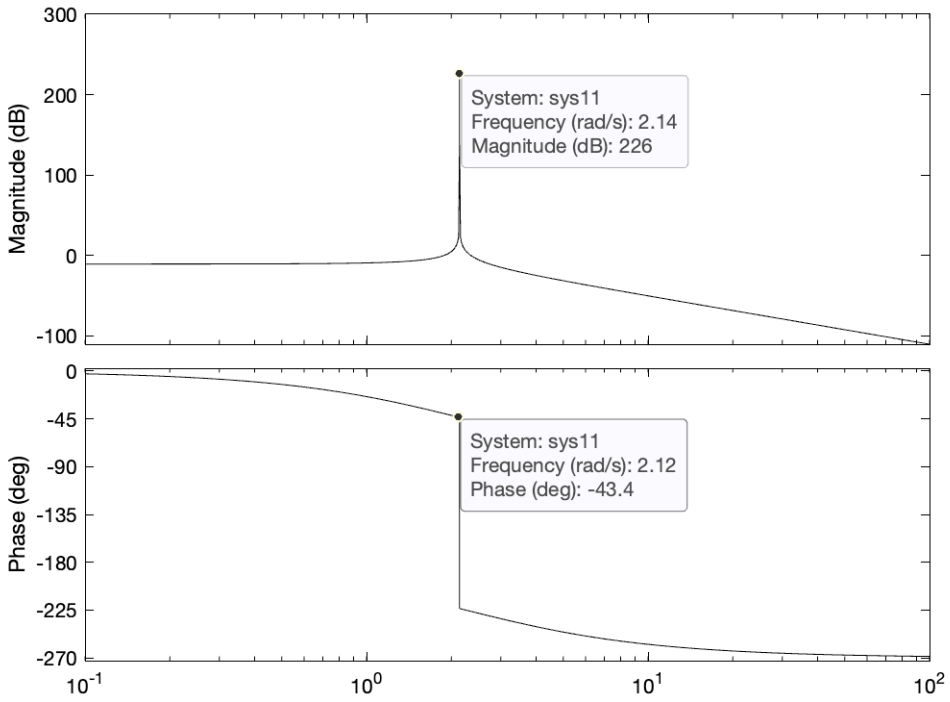

Figure 4.14: Bode plot for swaying

The gain peak for swaying can be observed at a frequency 2.14 rad/s.

Conclusion

The system seems to be quite slow compared to the chosen sampling frequency equal to 100hz. Considering Nyquist frequency equal to 50hz, aliasing will not occur.

In selecting a 100 Hz sampling frequency for the control system, it has been identified as an optimal middle ground that effectively balances system requirements with technological capabilities. The system has a relatively slow response, which negates the necessity for an excessively high sampling rate. While faster frequencies could theoretically provide better data, in practice, they would yield little additional benefit for the given application.

Chapter 5

Prototype Implementation

Using the previous sections the experimentation and modeling has been done. The following sections will discuss how the previously established knowledge was converted into a working prototype. The chapter is split into two sections, discussing the hardware and software implementation.

5.1 Hardware Implementation

The hardware implementation is heavily based on the implementation of the full sized APS system. As can be seen in the below figure 5.1.

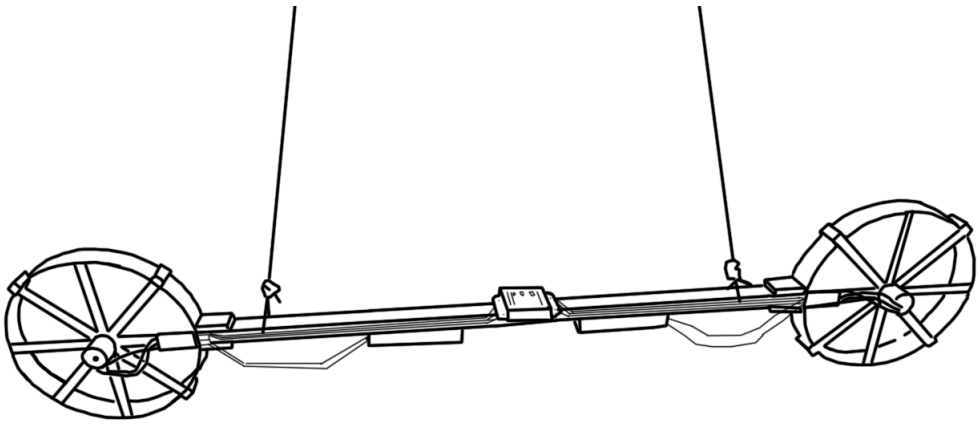

Figure 5.1: Prototype built for the project

Two hubs are connected to the main beam, to increase the moment of inertia as the thrusters producing force are positioned on the ends to produce the most torque.

However for smaller scale applications Sea-sight solutions follow a different approach, as for smaller scale solutions sway is a much lower priority, hence it is not necessary to have a balanced force applied over the translational axis. Hence the thrusters are positioned on one side with the system being mounted to the side of the payload. This is done for easier installation and a smaller more convenient system. However, the project explores the implementation of the APS system which considers sway.

Due to the very high maximum speed of the motors, producing large thrust proper safety measures must be put in place. An emergency stop is connected to both batteries to immediately disconnect the power from the motors. Safety tag lines are also anchored in order to prevent the prototype from overly swaying and yawing. Since the operating range / normal operation of the prototype should never exceed an arbitrary 20 degrees of yaw, dangling ropes were used to prevent the prototype from altering direction enough to hit surrounding objects. The ropes did not under normal conditions apply tension to the system and therefore it is considered that they do not impact the dynamics.

Furthermore, the motor speed was artificially limited in the code as the load of the system used for the tests described in the section 5.2.

The output of the STM32 PWM pins is 3.3 Volts as opposed to the desired 5 volts which the ESC requires in order to run the motors at full speed. In order to fix this, a level shifter was used. It takes in a 3.3 Volt signal as well as a 5-volt power input from the ESC and translates the PWM amplitude to 5 volts.

5.1.1 3D Printing

In order to create custom parts suitable for housing the motors, CAD models were created and the parts were 3D printed. Hub mounts were made for each motor as can be seen in Figure 5.2. These served as a stable way to mount the motors, as well as a safety precaution in the case that the propeller diss-attach from the motor while in operation. Although such an issue was not experienced during testing, it was noticed that the 3D-

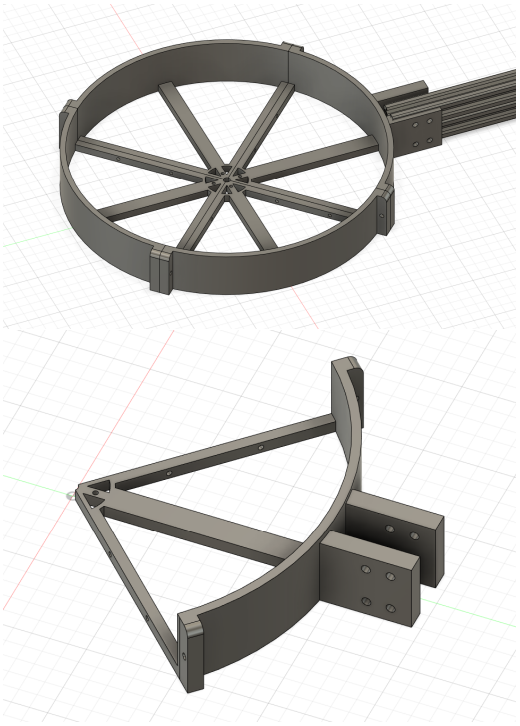

Figure 5.2: CAD Model of the full hub mount

printed parts are prone to vibrations. Therefore if more powerful motors or higher RPMs are used it should be considered that the PLA used for the 3D printing may shatter.

A large enough 3D printer was not available to print Hubs in one piece which can fit the 11-inch propeller blades. Therefore the hubs were printed in 4 pieces and bolted together.

A mount for the Microcontroller was also printed to securely mount it on the aluminum beam, and ensure the correct orientation of the sensors relative to the body.

5.2 Software Implementation

Now that the hardware implementation has been discussed, the emphasis must now be put on the software. The main components of the prototype are the STM32 MCU and the sensor attachment board. As the MCU is the main component, it must interface with the sensor board, be able to retrieve and process the data, and subsequently utilize it to control the rotational speed and direction of two hubs in order to stabilize the prototype.

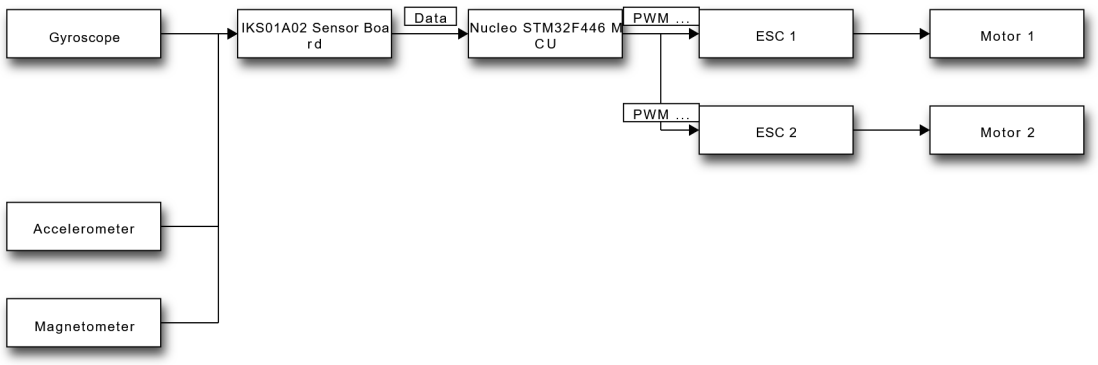

Figure 5.3: Overview of the interactions between the components

The general architecture of the code is seen in Figure 5.3. The Microcontroller receives data from the IKS01A2 sensor board. The data from the sensors is processed and subsequently passed as an input to the control loop. The output is mapped onto a PWM signal to actuate the motors.

The programming of this project was conducted in C as it is the language used by the STM microcontroller. Originally the STM32 cube IDE was used, however, the main library used for the interaction between the MCU and sensor board is not open source, and no documentation is available on the internal sensor processing. Hence, it was decided that due to time constraints, it was not possible to re-write the functionality of the library within the given time. A simpler implementation using Arduino-style code adapted for the STM32 microcontrollers used. This made it possible to focus on the implementation of the actual control and modeling.

All used code implementations can be seen on the Github repository **??**.

The structure of the code follows a standard Arduino-style implementation with a setup and loop functions. However, this description will focus on the project-specific aspects of the code. That is the sensor data acquisition and processing as well as the control implementation. Below can be seen an overview of the main functions used in the code.

- sensor_all() - Is called on every iteration of the main loop and used for sensor data acquisition as well as the preprocessing in converting the units of the data.
- complementaryFilter() - Implements a complementary filter to fuse sensor data to achieve a more precise result.
- ObserverUpdate(x_est_next) - Takes in the input and output to the plant and returns the output. Is used to update the input and output using the current states.
- InputX(u_e) - Is used to implement the integral term and accumulate error.
- pwmMap(input) - Maps the output of the control loop to a PWM signal, as well as sets the limits of the motor speed.

5.2.1 Sensor Data Acquisition

The Relevant data to be acquired from the sensors is the roll (later to be linearized for linear position) and yaw angle. The roll is achieved by performing sensor fusion between the accelerometer and gyroscope. Furthermore, a fusion between

the gyroscope and magnetometer is used to get the yaw angle. these sensors are fused by a complementary filter to combine the advantages of each sensor and mitigate some of the downsides. Since each of the individual sensors suffers from some limitations. The major limitations are the following, the gyroscope angles produce drift, the accelerometer angles are mostly valid at rest with no capabilities of measuring yaw, and the magnetometer is sensitive to noise from electronics and rotation in other angles besides yaw [11].

5.2.2 Code overview

The following section will go more into detail about the actual implementation of the above-mentioned functions in code.

sensor_all()

```
AccGyr.Get_X_Axes(accelerometer);
AccGyr.Get_G_Axes(gyroscope);

// Read magnetometer data from LSM303AGR_MAG_Sensor
Mag.GetAxes(magnetometer);

// Convert gyroscope data from mdps to radians/s and update the
  Sensor_angle_Gyro matrix
Sensor_angle_Gyro(0, 0) += (gyroscope[0] * mddps_to_rads) *
  deltaTime;
Sensor_angle_Gyro(1, 0) += (gyroscope[1] * mddps_to_rads) *
  deltaTime;
Sensor_angle_Gyro(2, 0) += (gyroscope[2] * mddps_to_rads) *
  deltaTime;

// Convert accelerometer data from mg to m/s^2 and update the
  Sensor_Acc matrix
Sensor_Acc(0, 0) = accelerometer[0] * mg_to_si;
Sensor_Acc(1, 0) = accelerometer[1] * mg_to_si;
Sensor_Acc(2, 0) = accelerometer[2] * mg_to_si;

// magnetometer data
Sensor_Mag(0, 0) = magnetometer[0];
Sensor_Mag(1, 0) = magnetometer[1];

// Calculate accelerometer angles (roll and pitch)
Sensor_angle_Acc(0, 0) = atan2(Sensor_Acc(0, 0), Sensor_Acc(2, 0));
Sensor_angle_Acc(1, 0) = atan2(Sensor_Acc(2, 0), g);

// Calculate magnetometer angle (yaw)
```

```
26  Sensor_angle_Mag(0, 0) = atan2(Sensor_Mag(0, 0), Sensor_Mag(1, 0));
```

Listing 5.1: Sensor all Implementation

Firstly a function to get the data from the sensor instance AccGyr of the LSM6DSL sensor is called. This instance is defined earlier in the code and can be used to get the data from the built-in gyroscope or accelerometer (AccGyr.Get_X_Axes()). Similarly, the magnetometer sensor instance is also called to retrieve the data from the sensor. Mag.GetAxes(magnetometer) retrieving magnetometer data from the LSM303AGR_MAG_Sensor. The following section is used to convert the sensor data into the proper units, meaning, Gyroscope data is converted from mdps to radians/s and accelerometer data from mg to m/s^2.

Lines 22 - 26 are used to calculate the roll, pitch and yaw angles using trigonometry.

complementaryFilter()

```
1  x_est_next(1, 0) = alpha1 * (x_est(1, 0) + Sensor_angle_Gyro(1, 0))
     + (1 - alpha1) * Sensor_angle_Acc(1, 0);
2  x_est_next(2, 0) = alpha2 * (x_est(2, 0) + Sensor_angle_Gyro(2, 0))
     + (1 - alpha2) * Sensor_angle_Mag(0, 0);
```

Listing 5.2: Complementary filter code implementation

The above code implements a complementary filter to fuse sensor data, achieving a more precise result in estimating orientation.

The complementary filter combines gyroscopic and accelerometer (and magnetometer for yaw) data to estimate the pitch and yaw angles. This method balances the short-term accuracy of the gyroscope with the long-term stability of the accelerometer and magnetometer, reducing noise and drift errors. This is done using a balancing factor Alpha, for this project, two alpha values have been used. It was found that better results could be produced if the yaw and sway had different alpha coefficients. Hence for calculating the pitch angle, an alpha of 0.1 was used, however for the yaw an alpha of 0.2 was preferable. The pitch angle uses the fusion between the angle from the gyroscope and the accelerometer. the yaw angle uses the magnetometer and gyroscope readings[12].

ObserverUpdate(x_est_next)

```
1 Matrix<2, 1, float>ObserverUpdate(const Matrix<2, 1, float>& y) {
```

```
    BLA::Matrix<6, 1> L_y = L * (y - (C * curr_state));

    // Compute the input effect
    BLA::Matrix<6, 1> B_u = B * input;

    // Update equation with Euler integration
    BLA::Matrix<6, 1> next_state = curr_state +  (((A * curr_state) +
    B_u + L_y) * 0.01f) ;

    // Update the x_est to the current calc
    curr_state = next_state;


    u_e = -K_f * next_state;
    u_e(0, 0) = (u_e(0, 0) > 1.0f) ? 1.0f : ((u_e(0, 0) < -1.0f) ?
    -1.0f : u_e(0, 0));
    u_e(1, 0) = (u_e(1, 0) > 1.0f) ? 1.0f : ((u_e(1, 0) < -1.0f) ?
    -1.0f : u_e(1, 0));

    return u_e;
}
```

Listing 5.3: Observer update function

This function implements a state observer using matrices A, B, C, L, and the current state curr_state. It calculates the estimated system state (next_state) and uses it to compute the control effort u_e.

The function uses the current state estimation to update the input and update using an integral controller and previously defined matrices.

All matrices used in the code implementation of the project are done using the BasicLinearAlgebra library which provides an efficient implementation of matrix structures as well as provides functionality to efficiently complete multiplication, concatenation, as well as other related matrix operations.

Finally the library adds a saturation to the output, from -1 to 1, as this range of values is accepted by the plant.

InputX(u_e)

```
  Matrix<2,1, float> InputX(const Matrix<2,1, float> u_e){
  BLA::Matrix<2,1, float> integralErrors = IntegError(x_est_next, r);
  input = integralErrors + u_e;
  BLA::Matrix<2,1, float> input1;
  return input1;
```

```
}
```

Listing 5.4: Input function code

The function is used to Integrate the error between the estimated state and the reference (r), then adds this to the control effort u_e. Similarly to the Observer function, Euler integration is used to accumulate the error over time. It must be noted that more advanced integration methods could be used to achieve more accurate results, however further research should be done to see how much of an effect this would have on the overall performance.

pwmMap(input)

```
int pwmMap(float inputs) {
  if (inputs < -5){
    return 1000;

  }
  else if (inputs > 5){
    return 2000;
  }
  else {
    int pwm = (inputs * 100) + 1500;
    return pwm;
  }
}
```

Listing 5.5: PWM Mapping

This function very simply maps the control signal to a PWM (Pulse Width Modulation) signal to actuate the motors, with limits set for motor speed. The limits of -5 to 5 are used due to the input variables being in radians. Hence it was arbitrarily determined that 5 radians in either direction is well outside the operating range. Hence the signals are limited for safety to prevent the motors from reaching dangerous speeds. 1500 is added as it is the neutral position of the motors. Hence when the input is zero the motors should not be spinning as no actuation is necessary.

5.2.3 Alternative Implementation

Although for the main code segment, a complementary filter is used, this implementation did not give optimal results. A complementary filter was not enough to give extremely consistent results. Some drift and inaccuracies were still present in the data. Because of this during the real-world testing of the prototype, an alternative implementation using a Kalman filter was tried. As previously mentioned a core library used for the interaction between the STM32 board and the sensors

(MotionFX) was not available in open source and hence was not explored in the scope of this report. However, to more effectively prove the validity of the control system created this library was still used for testing. Although the theory behind the Kalman filter is out of scope and this section must only serve as a foundation for further research. As to further read on this subject, refer to [13] [14] [15].

Chapter 6

Control

The following chapter will discuss the control of the model and system described in previous chapters. Open and closed-loop analysis is provided, simulations and a real world experiment will also be discussed in this chapter as a way to validate the acquired control system. The system will focus mainly on controlling the yaw of the prototype, however the sway is also considered.

Rotational position control for yaw is prioritized. However, swaying should also be minimized. Therefore, the following characteristics of yaw movement response should be reached:

- Operating range between 1 and 2 radians.
- Steady-state error equal 0.
- Overshoot less than 5%.
- Settling time lower than 5 seconds.
- Rise time has not been specified.

Disturbance rejection

This project does not focus on disturbance rejection. This matter can be developed in a future project.

The control objective is to be able to control the system using a state space model. The system can be seen in 6.1 and 6.2.

$$\dot{x} = Ax + Bu \tag{6.1}$$

$$y = Cx + Du \tag{6.2}$$

Where:

- x is the state vector,
- u is the input vector,
- y is the output vector
- A, B, C, D are system's matrices

6.1 Augmented system

To allow for integral control within the system, the system matrices must be augmented. This also has a secondary effect in helping remove constant disturbances. In order to add integral control, the system must be augmented by adding an extra state for the integral of the errors. All other matrices are also augmented to accommodate for the extra state. The augmented system matrices can be seen below and are calculated using 6.1 and 6.2.

$$\dot{x}_{\text{aug}} = A_{\text{aug}} x_{\text{aug}} + B_{\text{aug}} u \tag{6.3}$$

$$y_{\text{aug}} = C_{\text{aug}} x_{\text{aug}} + D_{\text{aug}} u \tag{6.4}$$

$$A_{\text{aug}} = \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix}, B_{\text{aug}} = \begin{bmatrix} 0 \\ B \end{bmatrix}, C_{\text{aug}} = \begin{bmatrix} C & 0 \end{bmatrix}, D_{\text{aug}} = D. \tag{6.5}$$

$$x_{\text{aug}} = \begin{bmatrix} x \\ \int (r - y)\, dt \end{bmatrix} \tag{6.6}$$

Furthermore, the disturbance parameters are added onto the augmented system shown in Equation 6.5. The disturbances caused by wind $w(t)$ is defined as :

$$\begin{cases} \dot{X} = AX + Bu + Ew \\ y = CX + Fw \end{cases} \tag{6.7}$$

The new integral part of the augmented state variable $X_I(t)$ is shown below. As the accumulation of error with time between the reference and actual output.

$$X_I(t) = \int_0^t (y(\tau) - r(\tau)) d\tau \tag{6.8}$$

The differential equation for displaying X_I with reference to the wind disturbance is also shown below. Followed by the full state space.

$$\frac{dX_I(t)}{dt} = y(t) - r(t) = CX(t) + Fw - r(t) \tag{6.9}$$

$$\begin{bmatrix} \dot{X}_I \\ \dot{X} \end{bmatrix} = \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_I \\ X \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} u + \begin{bmatrix} F & -I \\ E & 0 \end{bmatrix} \begin{bmatrix} w \\ r \end{bmatrix} \tag{6.10}$$

The full augmented matrices are displayed below:

$$A_{\text{aug}} = \begin{bmatrix} 0 & 0 & -1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4.5796 & 0 & 0 & 0 & 0.1460 & 0.1460 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.9344 & 0 & 0.5376 & -0.5376 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2.2472 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.2472 \end{bmatrix}$$

$$B_{\text{aug}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 20.6292 & 0 \\ 0 & 20.6292 \end{bmatrix}$$

$$C_{\text{aug}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$D_{\text{aug}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

6.2 Closed loop design

Further, the controller for the closed-loop system will be designed based on the previously created model of the system. Many control considerations have to be taken into account and an in-depth analysis of the results has to be conducted.

A full-state feedback controller is used based on the block diagram in Figure 6.1.

That structure allows for efficient response as well as eliminating steady-state error using integral control made possible by the augmented system.

Given the augmented system matrices A_{aug}, B_{aug}, C_{aug}, and D_{aug}, and the feedback gain matrix K_{aug}, the closed-loop system matrices $A_{c_{\text{aug}}}$, $B_{c_{\text{aug}}}$, $C_{c_{\text{aug}}}$, and $D_{c_{\text{aug}}}$ are defined as follows:

$$A_c = A_{\text{aug}} - B_{\text{aug}} K_{\text{aug}}$$

$$B_{c_{\text{aug}}} = \begin{bmatrix} \mathrm{I} \\ \mathbf{0} \end{bmatrix}$$

$$C_{c_{\text{aug}}} = C_{\text{aug}}$$

$$D_{c_{\text{aug}}} = D_{\text{aug}}$$

The matrices $A_{c_{\text{aug}}}$, $B_{c_{\text{aug}}}$, $C_{c_{\text{aug}}}$, and $D_{c_{\text{aug}}}$ represent the state-space model of the closed-loop system with state feedback and integral action.

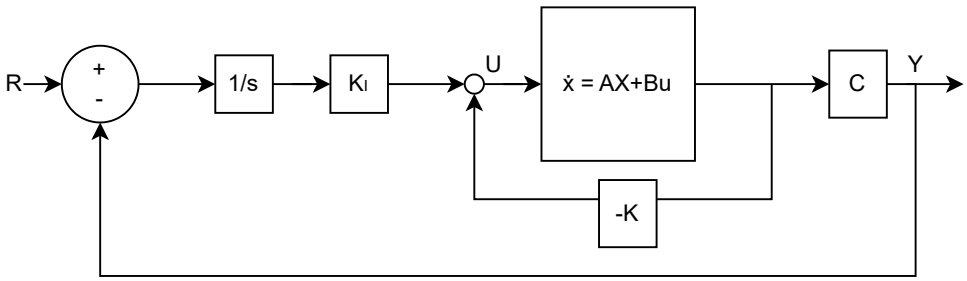

Figure 6.1: Block diagram of a closed loop system, where Ki is an integral gain, -K is State feedback gain

6.3 Full state feedback design

To build a full-state feedback closed loop system, these assumptions are taken for calculations.

- The augmented system is stable;
- The set-point $r(t)$ and disturbance $w(t)$ are some constants.

Hence utilizing these assumptions, the full state space can be written out and seen in equations 6.11 and 6.12. The variable definitions for the below state space equations are seen here :

- $X_{I_{ss}}$ are state values for the integral output states.
- X_{ss} are state values of the output states.
- u_{ss} refers to the control input at steady state. It is the constant input value that makes the state of the system unchanging over time (steady-state condition).
- F is the coefficient matrix multiplied with ω representing the strength of the disturbances at a specific time.
- E matrix is associated with how the disturbances or noise ω directly affect the state derivatives.
- I is the identity matrix.

$$\begin{bmatrix} \dot{X}_I \\ \dot{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} => \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_{I_{ss}} \\ X_{ss} \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} u_{ss} + \begin{bmatrix} F & -I \\ E & 0 \end{bmatrix} \begin{bmatrix} w \\ r \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.11}$$

$$\begin{bmatrix} F & -I \\ E & 0 \end{bmatrix} \begin{bmatrix} w \\ r \end{bmatrix} = - \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_{I_{ss}} \\ X_{ss} \end{bmatrix} - \begin{bmatrix} 0 \\ B \end{bmatrix} u_{ss} \tag{6.12}$$

The above equation is substituted into the augmented system model:

$$\begin{bmatrix} \dot{X}_I \\ \dot{X} \end{bmatrix} = \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_I \\ X \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} u - \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_{I_{ss}} \\ X_{ss} \end{bmatrix} - \begin{bmatrix} 0 \\ B \end{bmatrix} u_{ss} \tag{6.13}$$

$$\begin{bmatrix} \dot{X}_I \\ \dot{X} \end{bmatrix} = \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_I - X_{I_{ss}} \\ X - X_{ss} \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} (u - u_{ss}) \tag{6.14}$$

$$\begin{bmatrix} \dot{X}_I \\ \dot{X} \end{bmatrix} = \begin{bmatrix} 0 & C \\ 0 & A \end{bmatrix} \begin{bmatrix} X_I - X_{I_{ss}} \\ X - X_{ss} \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} (u - u_{ss})$$

The new state vector $z(t)$ and a new input $v(t)$ are defined as:

$$z(t) = \begin{bmatrix} X_I - X_{I_{ss}} \\ X - X_{ss} \end{bmatrix}, \quad v(t) = (u - u_{ss}),$$

$$\dot{z} = \hat{A}z + \hat{B}v$$

To design a full-state feedback controller for the above system, Linear–quadratic regulator (LQR) is used [16].

$$v = -\hat{K}z = -\begin{bmatrix} K_I & K \end{bmatrix} \begin{bmatrix} X_I - X_{I_{ss}} \\ X - X_{ss} \end{bmatrix}$$

6.4 Linear–quadratic Integral

The Linear–quadratic Integral known as (LQI) is a full-state feedback designing technique similar to that of LQR, however, incorporating the previously mentioned integral terms. The basis of this technique revolves around a cost function given as:

$$J = \int_0^\infty (z^T \hat{Q} z + v^T \hat{R} v) dt \tag{6.15}$$

Where the discrete form is given by

$$J = \sum_{n=1}^{\infty} (z[n]^T Q z[n] + v[n]^T R v[n]) dt \tag{6.16}$$

The main takeaway from this equation is to understand that the cost accumulates as it further deviates from the set point with given weights set to the state inputs. These numerical weights are in diagonal matrix form with $\hat{Q}$ and $\hat{R}$, respectively. These Matrices set the importance of each individual state and input [17].

6.4.1 Computation of K

After the parameters of the $\hat{Q}$ and $\hat{R}$ matrices are set, the formula for the full-state feedback controller is

$$K = R^{-1} \hat{B}^T P \tag{6.17}$$

Where the discrete equivalent is

$$K = (B^T P B + R)^{-1} (B^T P A) \tag{6.18}$$

Where P continuous is computed by the Algebraic Riccati Equation (ARE) [18]. given as such,

$$\hat{A}^T P + P\hat{A} - P\hat{B} R^{-1} \hat{B}^T P + \hat{Q} = 0 \tag{6.19}$$

And the discrete formula for P is

$$A^T P A - P - (A^T P B)(B^T P B + R)^{-1} (B^T P A) + Q = 0 \tag{6.20}$$

The ARE can give multiple solutions, whereas some solutions give an unstable system. So the last step is to analyze stability by checking the eigenvalues and using the respective stable controller[19].

6.4.2 Controller structure

As mentioned in the beginning of the chapter, given the APS prototype, the intuitive control prioritization requirements given yaw, sway, and energy of the system are.

- Stabilizing yaw rotation as a main priority (θ)
- Stabilizing the sway of the system as a last priority (X).

Energy at this point is not so important, as the goal is to achieve control and system stabilization with the APS mini as a proof of concept. However, the use of energy is important in the context of not reaching saturation and its relation with observer dynamics in the closed loop system (not all states are measurable thus they have to be estimated with the system model), as such it is given the highest weight out of all the parameters. .

6.4.3 Controller Tuning

As mentioned before, the Q and R matrices set weights to each state and input. Thus, a respective comparison is in order. Note that the weights will be set in the function lqr() and dlqr() using an augmented Q and R matrix in matlab. Lqr() is to get continuous model gains and dlqr() is to get discrete model gains.

	X_i	θ_i	X	$\dot{X}$	θ	$\dot{\theta}$	F_1	F_2	$Vpwm_1$	$Vpwm_2$
Q	0.45	0.45	3	0	6	0	0	0	-	-
R	-	-	-	-	-	-	-	-	17	17

Table 6.1: State Tuning

$$Q_{\text{aug}} = \begin{bmatrix} 0.45 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.45 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_{\text{aug}} = \begin{bmatrix} 17 & 0 \\ 0 & 17 \end{bmatrix}$$

Continuous time

Hence, the next step is to implement our parameters - A_{aug}, B_{aug}, Q_{aug}, and R_{aug} in the function lqr() part of Matlab.

```
[K_continous, S_aug, E_aug] = lqr(A_aug, B_aug, Q_aug, R_aug);
```

Listing 6.1: LQR function

The function returns the K controller (proportional and integral controller included), The solution to the ARE equation, and the eigen-values of the closed loop system. So the total controller (K) gains are:

$$\begin{pmatrix} -0.1150 & -0.1150 & -0.1785 & 0.1389 & 0.3350 & 0.4128 & 0.0711 & -0.0543 \\ -0.1150 & 0.1150 & -0.1785 & 0.1389 & -0.3350 & -0.4128 & -0.0543 & 0.0711 \end{pmatrix}$$

Where K_i is equal to:

$$\begin{pmatrix} -0.1150 & -0.1150 \\ -0.1150 & 0.1150 \end{pmatrix}$$

and K_f is equal to:

$$\begin{pmatrix} -0.1785 & 0.1389 & 0.3350 & 0.4128 & 0.0711 & -0.0543 \\ -0.1785 & 0.1389 & -0.3350 & -0.4128 & -0.0543 & 0.0711 \end{pmatrix}$$

Discrete time

For the chosen sampling rate of 100hz in the discrete-time domain, the gains only changed up to the second decimal place at most. The change was was not significant due to the relatively fast sampling rate. So the total controller (K) gains are:

$$\begin{pmatrix} -0.1148 & -0.1136 & -0.1808 & 0.1377 & 0.3294 & 0.4091 & 0.0707 & -0.0539 \\ -0.1148 & 0.1136 & -0.1808 & 0.1377 & -0.3294 & -0.4091 & -0.0539 & 0.0707 \end{pmatrix}$$

Where K_i is equal to:

$$\begin{pmatrix} -0.1148 & -0.1136 \\ -0.1148 & 0.1136 \end{pmatrix}$$

and K_f is equal to:

$$\begin{pmatrix} -0.1808 & 0.1377 & 0.3294 & 0.4091 & 0.0707 & -0.0539 \\ -0.1808 & 0.1377 & -0.3294 & -0.4091 & -0.0539 & 0.0707 \end{pmatrix}$$

6.5 Observer design

The estimator is designed in a way to keep a balance between good transient response and low enough bandwidth. The separation principle says that feedback controller and observer can be done independently, because the purpose of an observer is to estimate system's states and the controller's task is to regulate system's dynamic behaviour.[20]

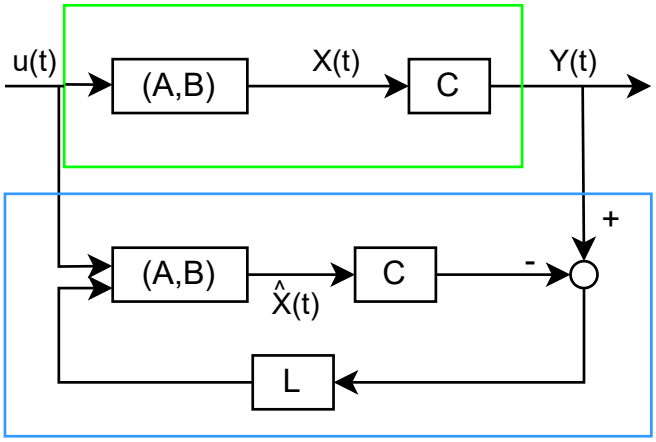

Figure 6.2: Observer structure block diagram, where $X(t)$ represents real system's states and $\hat{X}(t)$ represents estimated states. The observer part is marked by blue brackets and plant's part is marked by green brackets. Singal u(t) is an input to the plant and Y(t) represents measurable states of the plant

6.5.1 Observer Poles selection

For a pole placement "Place()" function is used in Matlab. That function computes a state-space feedback gain L. It determines the characteristic polynomial of the desired closed-loop system and adjusts gain L in such a way it suits eigenvalues of

$A - BL$, which are the poles.[21] that calculation can be performed using Ackermann's formula. Characteristic polynomial of an observer $\sigma_e(A)$ can be computed as follows:

$$\sigma_e(A) = A^n + \alpha_1 A^{n-1} + \alpha_2 A^{n-2} + \ldots + \alpha_n I \tag{6.21}$$

Where A is a state matrix, n is a number of states, α is the coefficient of the characteristic polynomial, chosen such that polynomial gains desired eigenvalues. I is an identity matrix and O is an observability matrix 4.59.

$$L = \sigma_e(A) O^{-1} \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix} \tag{6.22}$$

Continuous Domain Poles	Discrete Domain Poles	Discrete Poles Magnitude
-0.1360 + 2.1487i (Aug. pole)	0.9984 + 0.0215i	$\|z\|$=0.9986
-0.1360 - 2.1487i (Aug. pole)	0.9984 - 0.0215i	$\|z\|$=0.9986
-2.2546 + 0.0000i	0.9777 + 0.0000i	$\|z\|$=0.9777
-0.0663 + 0.0000i	0.9993 + 0.0000i	$\|z\|$=0.9993
-2.5638 + 0.0000i	0.9747 + 0.0000i	$\|z\|$=0.9747
-0.2663 + 0.0000i	0.9973 + 0.0000i	$\|z\|$=0.9973
-1.0018 + 1.6538i	0.9899 + 0.0164i	$\|z\|$=0.9900
-1.0018 - 1.6538i	0.9899 - 0.0164i	$\|z\|$=0.9900

Table 6.2: Continuous and Discrete Domain control Poles

.

Observer poles

The first pair of poles can be considered as poles created by augmentation. The estimator poles should be chosen to be faster by a factor of 5 - 10 than the control poles. As in the system plant control has to be slower than the observer so that the state can be correctly adjusted in the closed loop. Chosen poles and their magnitudes $|z|$ [22]:

- 0.9000 + 0.0000i, $|z|$ = 0.9000
- 0.8500 + 0.0000i, $|z|$ = 0.8500
- 0.9000 + 0.0000i, $|z|$ = 0.9000
- 0.8800 + 0.0200i, $|z|$ = 0.8802
- 0.8800 - 0.0200i, $|z|$ = 0.8802
- 0.8700 + 0.0000i, $|z|$ = 0.8700

Observer gain

$$L = \begin{bmatrix} 0.3267 & -0.0256 \\ 3.2360 & -0.5240 \\ 0.0073 & 0.3483 \\ 0.1756 & 3.6898 \\ 29.6574 & 1.7413 \\ 28.0739 & -16.9959 \end{bmatrix}$$

The chosen Poles are faster than the control poles as they are placed closer to the origin of the discrete pole plane axis.

6.6 Continuous time simulation

The simulation was conducted for control gains tuning. It was done by manually adjusting the values of Q and R matrices. They are represented in 6.1 table. The simulation was implemented in Matlab.

6.6.1 Code

The continuous closed loop system has been transformed to a discrete one by a function "c2d". By default, a 0 order method was used so it is assumed that the control inputs are piecewise constant over the sample time Ts.

```
% Parameters definition
{...)

%  State-space matrices definition
(...)

A_aug= [zeros(size(C,1), size(C,1)), -C;
                  zeros(size(A,1), size(C,1)), A];
B_aug = [zeros(size(C,1), size(B,2)); B];
C_aug = [zeros(size(C,1), size(C,1)), C];
D_aug = D;

% Redefine Q and R for the augmented system
Q_aug = diag([20, 20, 5, 0, 15, 0, 0, 0]); % Adjust weights as needed
R_aug = diag([8, 8]);

sys=ss(A_aug, B_aug, C_aug, D_aug);

[K_aug, S_aug, E_aug] = lqr(A_aug, B_aug, Q_aug, R_aug);

% Define the reference input vector for both outputs
ref_input = [0.1; 2]; % Unit step response for both outputs

```

```
24 % Closed loop system using state feedback control with integral action
25 Ac_aug = A_aug - B_aug * K_aug;
26 Bc_aug = [eye(size(ref_input, 1)); zeros(size(A, 1), size(ref_input,
       1))]; % Corrected
27 Cc_aug = C_aug;
28 Dc_aug = D_aug;
```

Listing 6.2: LQR function

In the above section of a code, Q and R are refined according to equation 6.5 for the augmented system. LQR computation is done with Matlab lqr function according to equation 6.15. Feedback law is used in line 25.

```
1
2 syscl_aug = ss(Ac_aug, Bc_aug, Cc_aug, Dc_aug);
3
4 dsys=c2d(syscl_aug, 0.01);
5 % Change in Sampling Time
6 Ts = 1/100; % 100 Hz sampling frequency
7
8 % Discrete-Time System with new sampling time
9 dsys = c2d(syscl_aug, Ts);
10
11 % Adjust the time vector for simulation
12 t = 0:Ts:20; % 20 seconds with 100 Hz sampling
13
14 % Simulate the closed-loop system response with the discrete-time
       system
15 [y_aug, t_aug, x_aug] = lsim(dsys, ref_input * ones(1, length(t)), t);
16 % Time span for the simulation
17 t = 0:0.01:20; % Adjust the time range and step size as needed
18
19 % Calculate the control input to the plant
20 u_control = -K_aug * x_aug';
21
22 %plotting
23 (...)
```

Listing 6.3: LQR function

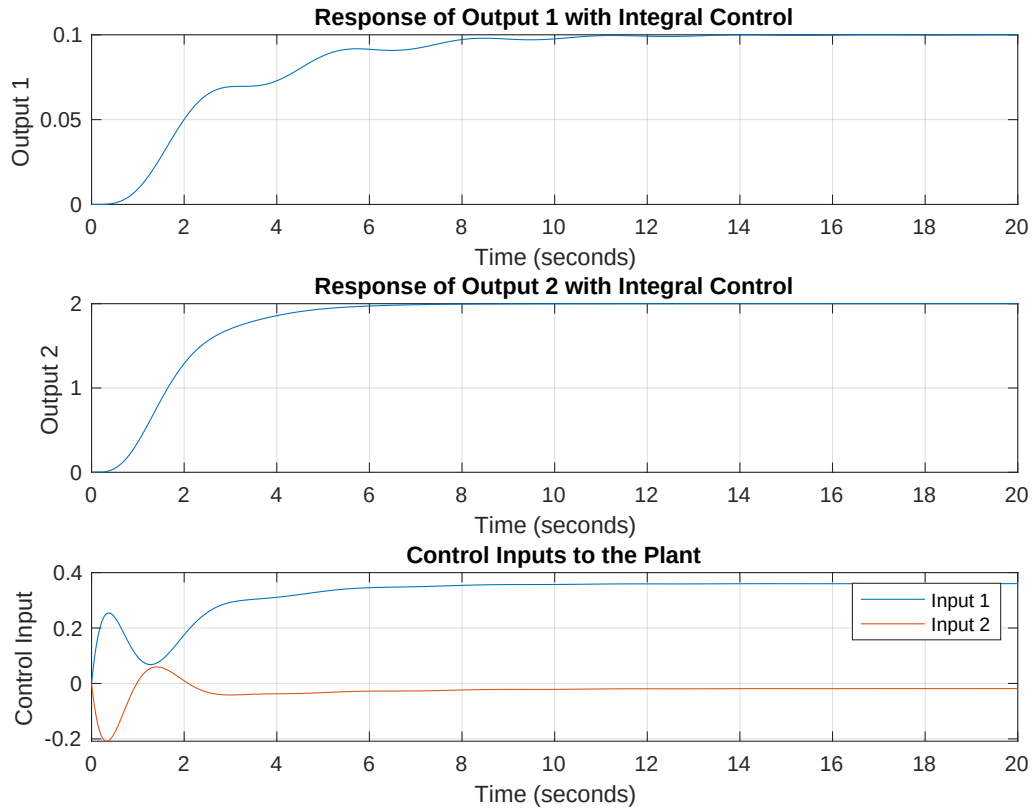

Figure 6.3: Simluation results where Output 1 is swaying and Output 2 is yaw rotation

The continuous simulation works with the tuned controller meeting all requirements, but it is important to understand that this simulation treats the forces as measurable. As a result, this simulation is not meant to serve as a final judgment for the controller gains and closed loop system. The continuous time simulation (without an observer) is a basis for the discrete time simulation that includes the observer. The continuous time simulation serves to view if the gains used are reasonable. After tuning the gains in continuous simulation, they are later evaluated in the full discrete system.

6.7 Discrete time simulation with observer

Discrete-time simulation has been performed in Matlab Simulink. Its block diagram is visible on a figure 6.4. The vector signal $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ indicates a reference where the first number is a set-point for swaying and the second one for yaw axis rotation. The simulation was conducted under 0.01s sampling time with discrete solver.

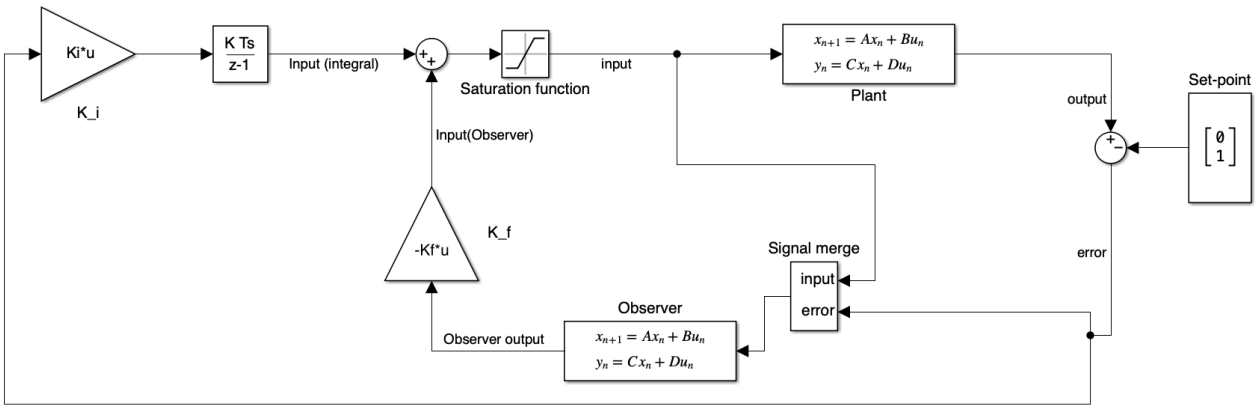

Figure 6.4: Closed loop system block diagram

6.7.1 Simulation results

Simulation results are analyzed based on 2 cases with different reference vector values:

1. 2 radians for rotational yaw position and 0.1m for swaying
2. 1 radian for rotational yaw position and 0m for swaying

Case 1.

Response represented on figure **??** shows characteristic of yaw rotation:

- Steady-state error equal to 0.
- Overshoot magnitude equal to 0.19 rad which corresponds to a 1.5% larger value than a steady state value,
- The rise time to reach 90 % of steady state value is equal to 2.19 seconds.
- Settling time to remain within 2% of the steady-state value is equal to 4.11s.

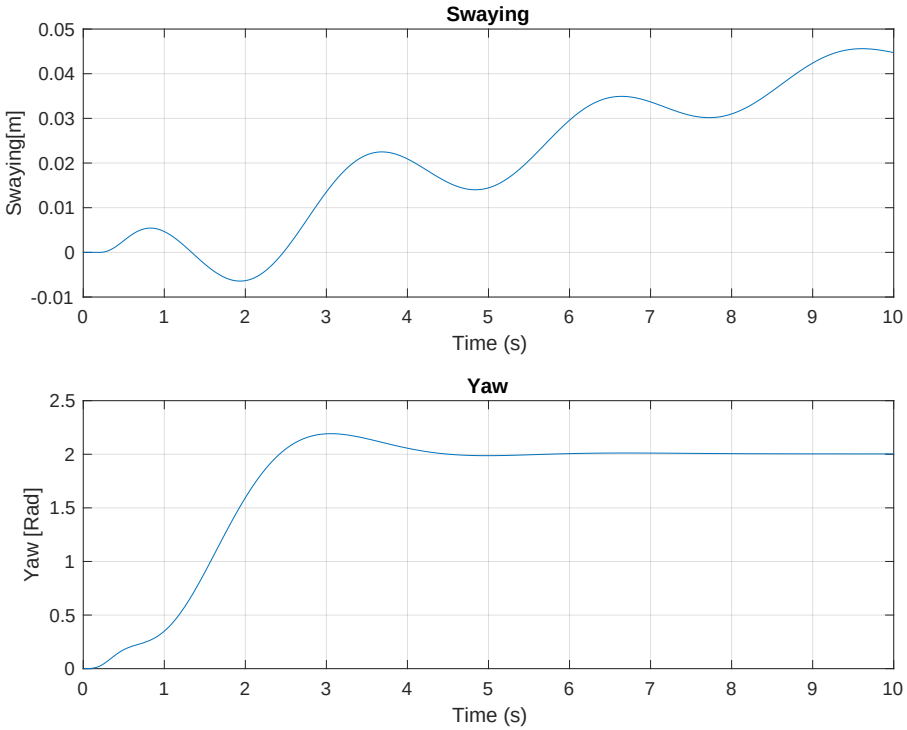

Figure 6.5: Discrete closed-loop model step response with set-points for yaw position 2 radians and 0.1m for swaying, where blue line indicates yaw rotational position and yellow swaying.

The simulation showed that the controller does not handle swaying well. Even small values like 10cm is not reached within response simulation time. Swaying reached 0.06m after 10 seconds. Zoomed swaying response is represented in Figure 6.5.

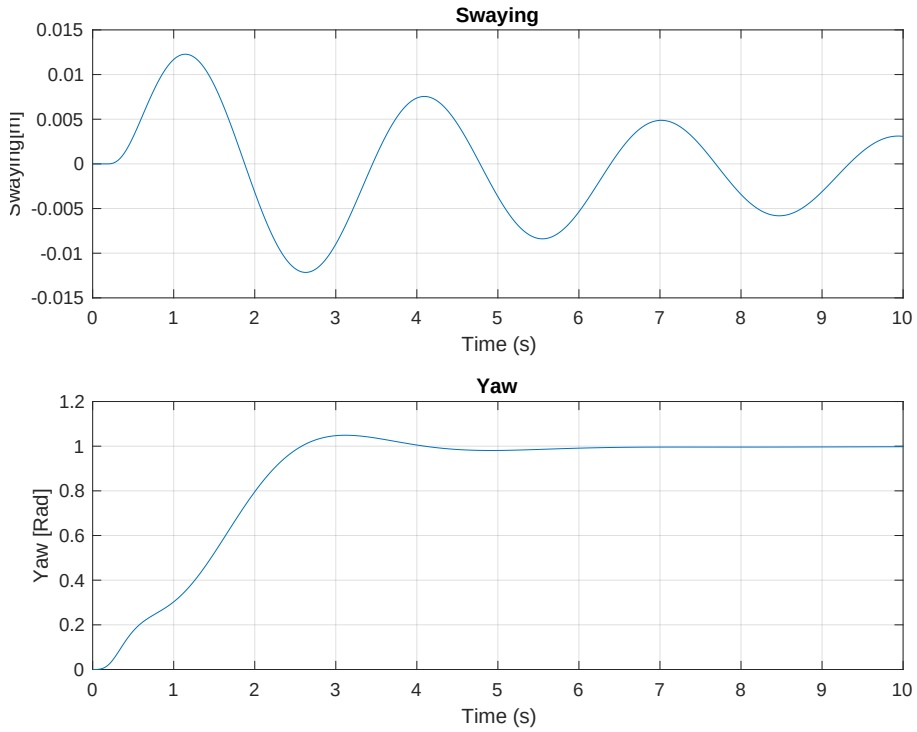

Figure 6.6: Discrete closed-loop model step response with set-points for yaw position 2 radians and 0.1m for swaying zoomed for sway response, where blue line indicates yaw rotational position and yellow swaying.

Case 2.

Response represented on figure 6.6 shows characteristic of yaw rotation:

- Steady-state error equal to 0.
- Overshoot magnitude equal to 0.045 rad which corresponds to a 4.5% larger value than a steady state value.
- The rise time to reach 95 % of steady state value in 2.48 seconds.
- Settling time to remain within 2% of the steady-state value is equal to 3.76s.

The oscillating movement has been observed for swaying with a magnitude of 0.012m.

As seen in the simulations above, the control objectives are met. The created control system is able to control the yaw of the particular system with adequate speed and accuracy. However it must be noted that the simulation is not a perfect representation of the real world and hence more tests should be done using the real world

prototype to further validate the control system. The control objectives specifically for yaw have been full filled well, however with sway more improvements could be made as the time taken for the system to stabilize is very large.

Chapter 7

Conclusion

In conclusion, a scaled-down system inspired by the APS is a viable solution to achieve system stabilization during lifting. The report aims to prove this by testing a real-world system as well as creating a simulation. A large portion of the project delved into the processes of modeling a state space system using mechanical modeling and data logging techniques. Whereby extension, the project looked into integral and full state feedback control tuned by LQI.

Not all the states of the system were measurable (specifically propeller thrust). This posed a challenge to the full state feedback controller for the real system, which was solved by implementing a state estimator (Luenberger Observer) that incorporated the model in the MCU to estimate system states.

The viability of the controller developed using all the aforementioned techniques was tested and verified using a prototype. However, before testing the controller on the prototype, the project looked into the system response in Simulink to validate the system controller.

A prototype was built from scratch which included two BLDC motors, a microcontroller, and a sensor board (magnetometer, accelerometer, gyroscope). The sensor data on its own from individual components proved to be inadequate for control purposes. As a result, a complementary filter was implemented. It is a sensor data acquisition technique that tries to take the best attributes of each component statically, with future work being hinted at employing a Kalman filter as it dynamically fuses the sensors. All in all, the project successfully details the implementation of the fundamentals of state space techniques onto a miniature version of the APS.

Chapter 8

Discussion

The project was not without its challenges, and there were different ways that each one was approached. The sensor data acquisition process was pressed forward due to time constraints. In order to develop the prototype, originally, a non-open source library featuring an easy-to-implement Kalman filter was used. However it was later realized that this approach would not be suitable for the learning objectives of the project and hence although the Kalman filter provided higher-quality sensor data, it was substituted for a manually implemented complementary filter. Although most of the project work was done using the Cube IDE, built for the STM32 Microcontrollers, the final implementations of the code used for testing were done using the Arduino IDE due to its ease of use and the availability of alternative libraries. A more comprehensive approach to sensor data implementation is potentially an intuitive project goal for next semester.

Furthermore, some considerations were made during modeling. When stabilizing sway movement, if the propellers are heavy enough, they could move the system unfavorably. The propellers were assumed to be too small to introduce any important dynamics under this context. Due to the nature of the propellers and the motor dynamics, the thrust was not the same in both directions. The state space system however did assume that the force in relation to the PWM is the same in both directions to retain a linear model. Lastly, The thruster dynamics in the state space model were based on a 0% - 40% of max force operating range where the dead zone PWM frequency was not directly added to the model as it introduced non-linear dynamics. The need for a linear model is needed as the control techniques are specifically meant for linear systems.

Chapter 9

Github Repository

Please see all the available code for the project in the following github repository.

https://github.com/Ka-rlis/APS_ITC5/tree/main

Chapter 10

Final prototype

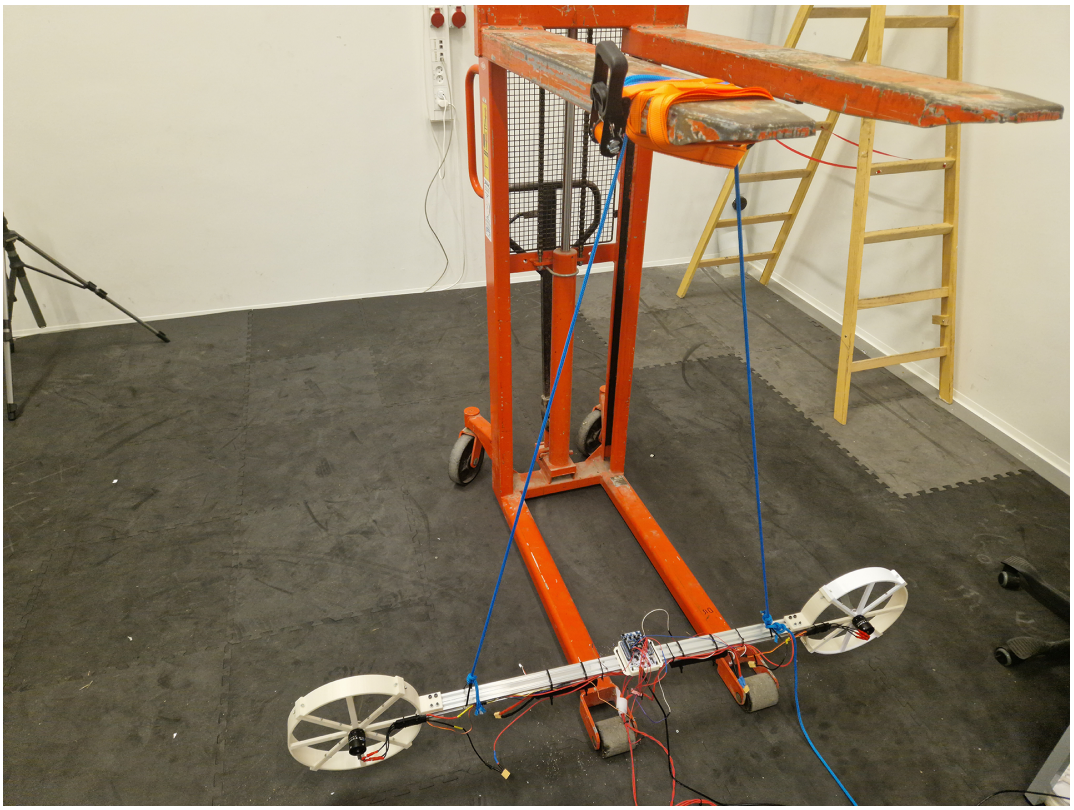

Figure 10.1: Pictures of the prototype during final testing

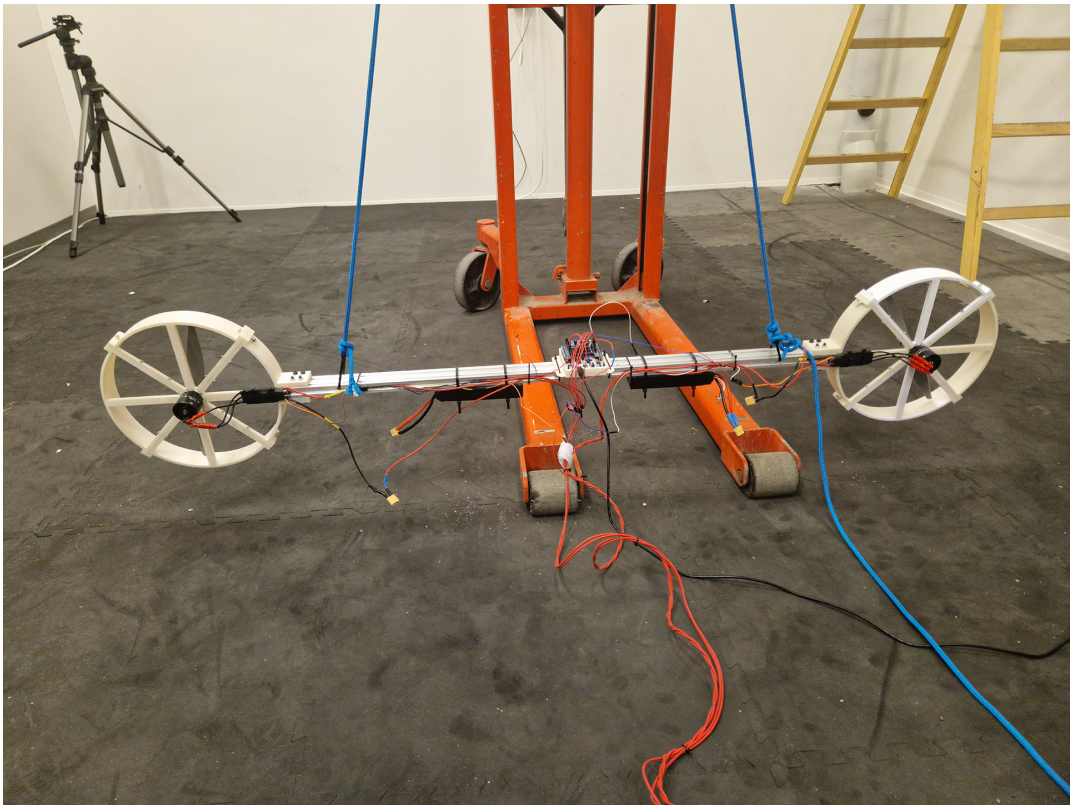

Figure 10.2: Pictures of the prototype during final testing

Bibliography

[1] GooMotion. *Pitch Control*. 2007. URL: http://www.helistart.com/pitchControl.aspx.

[2] ST Electronics. *STM32F446xC/E Datasheet*. Tech. rep. 2021. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html.

[3] Silicon Sensing Systems Limited. *MEMS Accelerometer*. 2017. URL: https://www.siliconsensing.com/technology/mems-accelerometers/#:~:text=All%20accelerometers%20work%20on%20the,corresponds%20to%20the%20applied%20acceleration..

[4] APC Propellers. *APC Performance data*. URL: https://www.apcprop.com/technical-information/performance-data/.

[5] Karl J Åström and Björn Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[6] Valerie Jardon and Sara Wilson. *4.1 Utilizing Transfer Functions to Predict Response*.

[7] Roland Tóth et al. "Crucial aspects of zero-order hold LPV state-space system discretization". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 4952–4957.

[8] Frederick Walker Fairman. *Linear control theory: the state space approach*. John Wiley & Sons, 1998.

[9] Zhenyu Yang. *Modern Control Theory - mm4*. Tech. rep. Esbjerg, Aug. 2023, p. 6.

[10] Hassan K Khalil. "Lyapunov stability". In: *Control systems, robotics and automation* 12 (2009), p. 115.

[11] Josef Justa, Václav Šmídl, and Aleš Hamáček. "Fast AHRS Filter for Accelerometer, Magnetometer, and Gyroscope Combination with Separated Sensor Corrections". In: *Sensors* 20.14 (2020). ISSN: 1424-8220. DOI: 10.3390/s20143824. URL: https://www.mdpi.com/1424-8220/20/14/3824.

[12] Hassen Fourati et al. "Position estimation approach by complementary filter-aided IMU for indoor environment". In: *2013 European Control Conference (ECC)*. 2013, pp. 4208–4213.

[13] Dan Simon. "Kalman filtering". In: *Embedded systems programming* 14.6 (2001), pp. 72–79.

[14] Gregory F Welch. "Kalman filter". In: *Computer Vision: A Reference Guide* (2020), pp. 1–3.

[15] Karlis Tregers, Hubert Dabrowski, and Adams Ali Gills. *ITC5 - APS Github Repository*. 2023. URL: https://github.com/Ka-rlis/APS_ITC5/tree/main.

[16] Russell D Smith and William F Weldon. "Nonlinear control of a rigid rotor magnetic bearing system: Modeling and simulation with full state feedback". In: *IEEE Transactions on Magnetics* 31.2 (1995), pp. 973–980.

[17] Michael F Everett. "LQR with integral feedback on a Parrot Minidrone". In: *Massachusetts Institute of Technology, Tech. Rep* (2015).

[18] Vinodh Kumar E and Jovitha Jerome. "Algebraic Riccati equation based Q and R matrices selection algorithm for optimal LQR applied to tracking control of 3rd order magnetic levitation system". In: *Archives of Electrical Engineering* 65.1 (2016), pp. 151–168.

[19] Katalin György, László Dávid, and András Kelemen. "Theoretical study of the nonlinear control algorithms with continuous and discrete-time state dependent riccati equation". In: *Procedia Technology* 22 (2016), pp. 582–591.

[20] Zhenyu Yang. *MM-4 Control Design Using Estimator*. Esbjerg, 2023.

[21] Inc. The MathWorks. *Matlab documentation, place*.

[22] Kumar E Vinodh, Jerome Jovitha, and S Ayyappan. "Comparison of four state observer design algorithms for MIMO system". In: *Archives of Control Sciences* 23.2 (2013).