Self-Balancing Robot for Painting

Martin Michael Lau

Nikolai Lund Christensen

Victor Højriis Olesen

Master's Thesis - Aalborg University

Department of Materials and Production

Denmark

May 31, 2024

Department of Materials and Production
Aalborg University

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Self-Balancing Robot for Painting

Theme:
Master's Thesis

Thesis Period:
Spring semester 2024

Thesis Group:
EMSD-4.1

Participants:
Martin Michael Lau
Nikolai Lund Christensen
Victor Højriis Olesen

Supervisors:
Dao Zhou

Page Numbers: 74
Appendix pages 60

Date of Completion:
May 31, 2024

Summary

This thesis concerns the analysis and control of a mobile wheeled inverted pendulum (MWIP), a dynamic system that exhibits nonlinear and unstable open-loop characteristics. The main objectives of this research are to develop a dynamic and electrical mathematical model of the MWIP and use it to design robust control schemes that can control the MWIP under varying load and external conditions.

Motivation

The reason for examining such a system is its possible use case for the local company Turf Tank which design line marking robots that automatically paints line markings for the sports industry. The company is facing problems with the its current solution, where it applies two Permanent Magnet Synchronous Machines (PMSMs) at the front and two caster wheels at the rear. The caster wheels produce undesirable characteristics and the company wishes to explore the possibility of removing the caster wheels altogether, while keeping the same painting accuracy and capabilities.

Structure

The thesis is split into four parts: the analysis of a pre-existing prototype and its limitations; the derivation of the mechanical and electrical models that govern the behaviour of the MWIP; using those models to design a linear quadratic controller that controls the pitch, yaw, and speed of the MWIP; and lastly applying the designed controller on the prototype to validate the model and control.

Preliminary Analysis

The preliminary analysis of the pre-existing prototype reveals electrical components that are deficient to the use case. The microcontroller unit (MCU) is deemed to slow for the LQR controller and more advanced control strategies, and the I/O capabilities of the existing MCU is insufficient for further development of the prototype. The Inertial Measurement Unit (IMU) that provides the orientation and position of the MWIP is deemed to be too noisy due to the mount. Furthermore, the IMU is replaced with another unit which has a sensor fusion algorithm. The MWIP is powered by two PMSM hub motor that are driven by an ODrive v3.6 motor driver.

Modelling

The mathematical model includes a three-dimensional dynamic model of the MWIP and is derived using Lagrangian mechanics. The mechanical model is limited to level and even surfaces, and the sloshing effect of the paint within the paint tank is not considered. The PMSM motor is modelled in the rotating reference frame using the Clarke and Park reference frame transformation and field oriented control (FOC) is applied in order to control the current and the torque of the motor. The motor driver is

using two three-phase six step inverters which are also modelled to establish its impact on the control of the motors.

Control

The dynamic model of the mechanics is linearised about the upright standstill position for the design of the full state feedback LQR control. The controller is applied to the dynamic model of the electrical and mechanical system to validate it using simulation.

Experiments

Multiple experiments are conducted on the MWIP to investigate the performance of the prototype. The hall effect sensors fitted to the motors are validated. The FOC loop of the current is validated using both linear and nonlinear simulation and experiments on the prototype. The motor parameters are found using optimisation that seeks to minimise the error between the model and tests executed on the prototype. The control is implemented on a Teensy MCU and tested under repeatable conditions in the laboratory.

Preface

This thesis documents a master thesis completed by a group consisting of three 4th semester students at M.Sc education in Mechanical Engineering specialised in Electro-Mechanical System Design at Aalborg University.

Throughout the thesis, source references are used and referred to the bibliography, placed at the end of the thesis. In the thesis, a source is referred to using the IEEE Reference Style. In the bibliography, books are specified with the author, title, edition, and publisher, while web sources are specified with the author, title, address, year and last visit. Figures and tables are numbered according to chapter, which means that the first figure in chapter four, has number 4.1, the second 4.2, and so on. Captions to tables are found above the given table and figure captions are found below the figure. Chapters in the appendices are enumerated using letters.

Bold font is used for vector and matrix notation. Derivatives with regard to time are denoted with a dot of ascending order based on the order of derivation. Hence $\frac{dx}{dt}$ becomes $\dot{x}$, $\frac{d^2x}{dt^2}$ becomes $\ddot{x}$ and so forth.

The authors want to express their gratitude to Torben N. Matzen, contact at Turf Tank, for great cooperation during the thesis.

Martin Michael Lau | Nikolai Lund Christensen | Victor Højriis Olesen

Nomenclature

Abbreviations

2WMRs	Two-wheeled mobile robots
AC	Alternating current
ACO	Ant colony optimisation
BLDC	Brushless DC
CM	Centre of mass
DC	Direct current
etc	Et cetera
FIFA	Federation Internationale de Football Association
FOC	Field oriented control
GA	Genetic algorithm
GPS	Global Positioning System
HODOSMC	High-order disturbance-observer-based sliding mode control
IAE	Integral of absolute error
IMU	Inertial measurement unit
LQG	Linear quadratic Gaussian
LQR	Linear quadratic regulator
LTI	Linear time-invariant
MCU	Microcontroller unit
MIMO	Multiple-input-multiple-output
MPC	Model predictive control
MTWIP	Mobile two-wheeled inverted pendulum
MWIP	Mobile wheeled inverted pendulum
PI	Proportional-integral

PID Proportional-integral-derivative

PMSM Permanent magnet synchronous machine

PP Pole pairs

PSO Particle swarm optimisation

PWM Pulse width modulation

RMS Root mean square

SA Simulated annealing

SISO Single-input-single-output

SMC Sliding mode control

SVPWM Space vector pulse width modulation

Constants

g Gravitational acceleration $9.82\,\mathrm{m/s^2}$

Symbols

α Yaw angle of the frame

β Pitch angle of the frame

$\boldsymbol{\lambda}_{abcs(f)}$ Flux linkage contribution to the Stator flux linkage of the windings from the permanent magnet

$\boldsymbol{\lambda}_{abcs(s)}$ Flux linkage contribution to the Stator flux linkage of the windings from the stator currents

$\boldsymbol{\lambda}_{abcs}$ Stator flux linkage of the windings

$\boldsymbol{\tau}$ Joint torques

$\dot{s}$ Linear velocity of the MWIP system

λ_f Flux linkage of the permanents magnet

$\tilde{\mathbf{x}}$ Error between the reference and the feedback

$\mathbf{A_i}$ Three-dimensional rotation matrix for $i = \{1,2,3\}$

$\mathbf{A}$ System matrix

$\mathbf{B}$ Input matrix

$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ Coriolis and centripetal vector

$\mathbf{C}$ Output matrix

$\mathbf{D}(\mathbf{q})$ Inertia matrix

$\mathbf{D}$ Direct coupling matrix

$\mathbf{e}_{i,\max}$ Maximum allowed control signal of each input $i = \{1,...,5\}$

$\mathbf{e}_{i,\max}$ Maximum allowed deviation of the state $i = \{1,...,5\}$

$\mathbf{G}(\mathbf{q})$ Gravity vector

$\mathbf{J}$ Inertia tensor

$\mathbf{J}_c$ Inertia tensor of the container

$\mathbf{J}_T$ Inertia tensor of the tank

$\mathbf{K}$ Feedback gain matrix

$\mathbf{P}_{CMF,local}$ Centre of mass of the frame in the local coordinate system

$\mathbf{P}_{CMT,local}$ Centre of mass of the tank in the local coordinate system

$\mathbf{P}_{CMW1,local}$ Centre of mass of wheel 1 in the local coordinate system

$\mathbf{P}_{CMW2,local}$ Centre of mass of wheel 2 in the local coordinate system

$\mathbf{Q}$ Non-conservative externally applied torques

$\mathbf{Q}$ Relative importance of the error

$\mathbf{q}$ Generalised coordinate vector

$\mathbf{R}$ Relative importance of the energy expenditure

$\mathbf{R}_s$ Stator resistance

$\mathbf{S_i}$ Local lengths for $i = \{1,2,3\}$

$\mathbf{u}$ Input vector

$\mathbf{x}$ State vector

$\mathbf{y}$ Output vector

μ Coulomb friction coefficient

μ_0 Permeability of air

ω_c Crossover frequency

ω_e Electrical angular velocity

ω_r Mechanical angular velocity

Φ_m Magnetic flux

ρ_p Density of the paint

τ	Time constant
θ_e	Electrical angle
θ_r	Mechanical angle
A	Surface area
B	Magnetic field density
b	Viscous friction
B_v	Viscous friction in wheels
B_{mf}	Magnetic field
CM_F	Centre of mass of the frame
CM_T	Centre of mass of the tank
CM_{W1}	Centre of mass of wheel 1
CM_{W2}	Centre of mass of wheel 2
d	Depth of the container
d_H	Width of the Hall element
e_i	Back emf for winding $i = \{a, b, c\}$
F_1	Force applied by motor 1
F_2	Force applied by motor 2
G_c	Current controller
G_p	Plant for q-axis current
$G_{ol.c}$	Transfer function of the open loop q-axis current
h	Height of the container
I_H	Current flowing through the Hall element
i_d	d-axis current
i_{is}	Stator current for winding $i = \{a, b, c\}$
$I_q(s)$	q-axis current in the Laplace domain
J_F, xy	Inertia of the frame with respect to xy axes
J_W, xy	Inertia of the wheel with respect to xy axes
K	Kinetic energy

K_i Integral gain

K_p Proportional gain

K_R Rotational translation energy

K_T Kinetic translation energy, Torque constant

K_v Kv rating

L Lagrangian

L Length of the pendulum

L_P Paint level

L_s Stator inductance

L_{ACMF} Distance from the origin to the CM of the frame

L_{ACMT} Distance from the origin to the CM of the tank

L_{ACMW} Distance from the origin to the CM of the wheel

L_d d-axis inductance

L_{ii} Mutual inductance $i = \{a, b, c\}$

L_{ij} Self inductance $i = \{a, b, c\}$, $j = \{a, b, c\}$ where $j \neq i$

L_{ls} Leakage inductance

L_m Magnetising inductance

L_q q-axis inductance

L_s Stator inductance

M Mass of the cart

m Mass of the pendulum

m_F Mass of the frame

m_W Mass of the wheel

m_c Mass of the container

m_p Mass of the paint

MI Modulation index

n_{cpr} Hall effect sensor count-per-revolution

P Potential energy

P_e	Electrical power
R_H	Hall constant
R_s	Stator resistance
r_w	Wheel radii
r_{dw}	Distance to the surrounding wire which produce the magnetic field
s	Distance travelled
s	Laplace operator
T_1	Torque delivered by motor at wheel 1
T_2	Torque delivered by motor at wheel 2
T_e	Output torque
T_s	Modulation period
T_{diff}	Difference torque
T_{sum}	Sum torque
u	Input to the system
V_H	Voltage output of the hall effect sensor
V_i	Space voltage vector $i = \{0, 2, ..., 7\}$
$v_{d,ff}$	Feedforward term for the d-axis voltage
v_{is}	Stator voltage for winding $i = \{a, b, c\}$
V_p	Volume of the paint
$v_{q,ff}$	Feedforward term for the q-axis voltage
$V_q(s)$	q-axis voltage in the Laplace domain
w	Width of the container
x	Horizontal displacement of the cart

Contents

Chapter 1

Introduction

The use of robotics is ever-increasing and becoming more entangled in our everyday life. According to [1], the forecast for the installation of industrial robots projects growth in the coming years. An area that is widely used in the industry is field robotics. This field of robotics is defined as robots intended for outdoor applications, often operating in unstructured or semistructured environments that can change over time [2].

The machines are used in a multitude of areas including: unmanned aerial vehicles for surveillance and package delivery [3, 4], surgery [5], building and construction [6, 7], mining [8, 9], space exploration [10], forestry [11], agriculture [12, 13], and autonomous driving of automobiles and trucks [14, 15]. To operate a controlled field robot either automatically, or remotely, the following three parts are required: a guidance system which consists of the robot's sensors and computes the reference determining the position, velocity and acceleration to be used in the motion control; a navigation system that uses the data collected from the available sensors to determine the state and motion of the robot; and finally, a control system, or motion control system, to determine the amount of force or torque required by the actuators to satisfy the intended control objective [16].

Various automatic guidance systems have been developed to make systems capable of functioning autonomously with one shown to have an accuracy of less than 12 mm [12, 17]. These systems employ sensors such as global positioning system (GPS), machine vision, etc. [12]. The affordability of these technologies has greatly improved, and progress is being made in navigation to provide more precise data [18]. Such advancements are enabling the expansion of the technology and its application.

[19, 20] suggest to expand the range of application of field robotics to sports, which is a relatively new area with scarce amount of publications compared to the aforementioned areas. In [20], it is proposed that the robots can perform periodic tasks such as cutting grass and line marking of a football pitch. These tasks are time-consuming and mostly performed manually by skilful personnel as the football pitch is required to meet certain standards issued by Federation Internationale de Football Association (FIFA).

To reduce manual labour companies such as Turf Tank makes line marking robots. Turf Tank's robots uses GPS to determine the position of the robot and the layout of the pitch. The robots are used for sports such as football, baseball, tennis, and more. The major components of the robots consist of: two wheels actuated by two permanent magnet synchronous machines (PMSMs), two castor wheels, a paint container and nozzle, a battery pack, and a casing.

An example of one of Turf Tank's robots is the Turf Tank Two, which is seen in Figure 1.1. Turf Tank uses control algorithms to automate the line marking process, minimising the risk of crooked lines as

well as reducing paint consumption by 50 % as compared to manual line markings according to their own data [21].

(a) View of Turf Tank Two

(b) Side view of Turf Tank Two

Figure 1.1: Pictures of Turf Tank two.

Currently, Turf Tank faces problems with their castor wheels. The castor wheels reduce the normal force on the traction wheels leading to an increased chance of slip. When slip occurs it increases the line marking tolerance, especially for substrates with low traction as e.g. clay tennis courts. Furthermore, as the robot turns, the friction in the ball bearing swivels yields an undesirable impact on the robots orientation. The castor wheels are also prone to collect dirt and grass which further increases the friction in the ball bearings which results in increasingly unpredictable behaviour during turning.

To resolve these issues, a new design without caster wheels is proposed as seen in Figure 1.2, which in addition also reduces the material and production cost of the design. The proposed design is a two-wheeled self-balancing robot, which falls under the category called mobile wheeled inverted pendulum (MWIP), also known as mobile two-wheeled inverted pendulum (MTWIP) or two-wheeled mobile robots (2WMRs) in some literature. A previously constructed prototype is available [22] limiting this thesis to the modelling and control of said prototype.

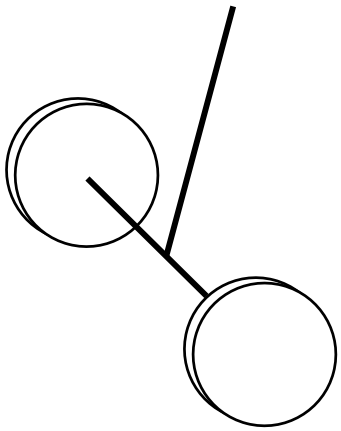


Figure 1.2: Sketch of proposed design consisting of two wheels and a rod representing the upper body of the robot.

The robot is an underactuated system as it has three degrees of freedom while only only having two control inputs from two motors. The paint used by the robot during operation results in a change in mass, change in centre of mass (CM), and introduces sloshing of the liquid during accelerations. As the

robot is subject to tight tolerances these dynamics must be considered when creating control schemes.

The study of MWIPs and control thereof originates in 1987 where [23] studied the two-wheeled balance control technology. Since then, a myriad of MWIPs have been developed with each of them utilising different control strategies. Earlier published works use linear control strategies such as [24] where a single-input-single-output (SISO) proportional-integral-derivative (PID) controller is applied by considering the problem as a 2D case, or as in [25] where multiple SISO-controllers are applied for a 3D case. However, due to the complexity of the system and uncertainties the majority of recent publications proposes more complex control structures such as linear quadratic Gaussian (LQG) control [26], fuzzy logic control [27], high-order disturbance-observer-based sliding mode control (HODOSMC) [28], adaptive super-twisting control [29], model predictive control (MPC) [30], adaptive fuzzy logic proportional-integral control [31]. To get an overview of the various control methods related to MWIPs, [32] suggests to divide methods into four distinct categories:

1. Linear control methods:
   Firstly, the dynamics of the system are derived mainly based on either Lagrange method or Newton-Euler method [32]. Secondly, the dynamics are linearised around a linearisation point to obtain the approximated linear model. Finally, using the approximated linear model, linear controllers are derived and applied to control the system, for example linear quadratic regulator (LQR), feedback linearisation control, etc.

2. Robust control methods:
   Robust control is widely used for MWIP systems due to model uncertainties, varying work conditions, external disturbances, characteristics of underactuated robot, and the nonlinear instability. Similar to the linear control method, firstly the dynamics of the system are derived mainly based on either Lagrange method or Newton-Euler method. Thereafter, methods such as H_2, H_∞, LQG, sliding mode control (SMC) can be applied to realise robust control.

3. Intelligent control methods:
   An intelligent control system is able to adapt to the changes in its environment by evaluating it. Based on the evaluation the control system alternates the controller to improve its characteristics accordingly [33]. These methods include adaptive control, fuzzy logic, neural network amongst others. Occasionally, a combination of intelligent techniques are utilised, for example fuzzy adaptive controller, fuzzy neural network controller, etc. [32].

4. Combination of intelligent control and optimisation algorithms:
   Typically, the intelligent control methods have a number of parameters, which the control performance depends on. Thus, it is not uncommon that the intelligent control is combined with optimisation algorithms which are responsible for optimising the control parameters to achieve optimal control. Examples of optimisation algorithms include particle swarm optimisation (PSO), genetic algorithm (GA), ant colony optimisation (ACO), and simulated annealing (SA).

The initial phase of analysis involves an examination of a prefabricated prototype used for initial testing of the feasibility of the solution. Thereafter, the prototype is to be modelled and control strategies suggested to ensure the desired capabilities.

Chapter 2

Problem Statement

This thesis is carried out in order to investigate the feasibility of an MWIP solution to automatic field line marking. This is done by investigating an MWIP prototype and applying linear control schemes to establish the applicability of the solution.
This leads to the following problem statement:

> *How can a prototype of a mobile wheeled inverted pendulum with varying payload be modelled and controlled to be self-balancing?*

Key Objectives

The key objective of the thesis is as follows:

- Analyse existing prototype with the aim of improving upon it.
- Establish requirements for further development of the prototype.
- Development of system model.
  - Mechanical model of MWIP.
  - Electrical model of motors.
  - Verification of models.
- Establishment of linear controller.
- Evaluation and comparison of controllers in real world applications.

Evaluation Criteria

The controllers are evaluated based on the performance of the existing prototype's control and compared to each new developed controller. This is done by following a predetermined trajectory, both on the nonlinear model and the prototype, and evaluating the difference between the desired and actual trajectory. Specifically the root mean square (RMS) tracking error, the Integral of Absolute Error (IAE) and the peak tracking error are evaluated. The RMS error quantifies the average tracking error over time, while the IAE further penalises large and sustained errors. The peak tracking error defines the overall tolerance of the line tracking. Furthermore the controllers are tested under disturbance conditions seen in real world applications like grass substrate and slopes.

Delimitations

- Fitting the painting nozzle to the robot is out of scope.
- Implementation of GPS feedback

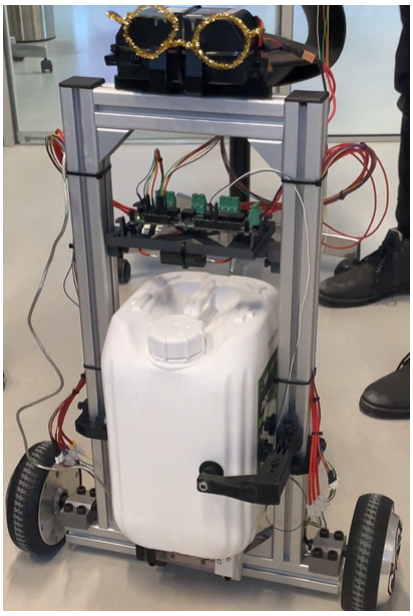
Chapter 3

Design of Prototype

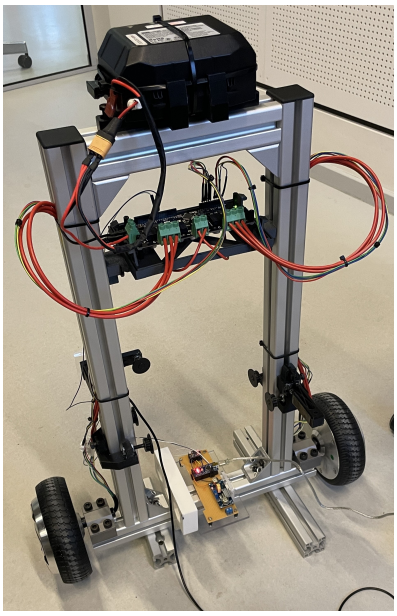
In this chapter, the previous prototype is analysed to achieve an understanding of the system in order to build upon the pre-existing design. Subsequently, the solutions to the encountered problems with the existing design of the prototype and modifications are described. Lastly, the system requirements and the final prototype are presented.

3.1 Original Prototype

In this section, the initial prototype developed by [22] is presented and analysed in order to achieve an understanding of the solution, and gain knowledge for further development of the prototype. The original prototype without castor wheels is seen in Figure 3.1.



(a) Front view with paint tank



(b) Back view without paint tank

Figure 3.1: The constructed prototype viewed from the front and back.

The frame consists of joint aluminium extrusions held in place by angle brackets. The paint tank is fitted between the aluminium extrusions and held in place by a set of 3D printed supports, the two upper supports have broken and are therefore not available in this thesis.

The primary electrical components of the prototype and their purpose are listed in the following:

- **Power source**
  The power source is a lithium ion battery [34] consisting of 14 ICR18650 batteries with a total

nominal voltage of 25.9 V and a capacity of 4 A h. The battery is mounted at the top of the prototype using two black 3D printed parts and zip ties.

- **Inertial measurement unit**
  The prototype features a GY-91 [35] inertial measurement unit (IMU). The IMU provides a three axis gyroscope, three axis accelerometer, three axis magnetometer, and a barometer. The IMU is used to determine the orientation, velocity and acceleration of the prototype. The IMU receives 5 V through a buck converter, and draws 4 mA. The IMU is seen mounted at the bottom of the frame on a perfboard along with the other electrical components.
- **Microcontroller unit**
  The microcontroller unit (MCU) is an Arduino Nano [36] with a clock speed of 16 MHz and one serial communication port. The MCU is responsible for receiving and interpreting signals received by the IMU and the motor driver. It is powered with 5 V through the buck converter and draws 19 mA. The MCU is also fitted to the perfboard.
- **Buck converter**
  To convert the 25.9 V output of the battery to the 5 V needed for the MCU and IMU, a DC-DC 5 A buck converter [37] is implemented. The buck converter has a maximum output power of 75 W with a maximum conversion efficiency of 96%. The buck converter is also fitted to the perfboard.
- **Joystick module**
  A joystick module [38] is included for analogue and digital control of the prototype. The joystick module features a two-axis thumb joystick, six buttons, and a switch. The module is not seen in the figures, as it is used to control the prototype.
- **Motor driver**
  The ODrive v3.6 [39] is chosen as the motor driver as it is compatible with the Arduino and can operate two brushless direct current (BLDC) motors at a maximum of 120 A per motor. The motor driver can receive an input voltage between 12 V and 56 V. The motor driver is seen fitted above the paint tank.
- **BLDC hub motors**
  The robot is actuated by two three-phase BLDC motors [40], the motors are controlled through the ODrive that receives signals from the Arduino Nano. The BLDC motors are taken from a DENVER HBO-6620 Hoverboard [41]. The tires are 168 cm in diameter, and the motor is fitted with an incremental encoder, with a resolution of 90 counts per revolution. The power at maximum torque is 477 W where it draws a current of 13.22 A.

3.1.1 Previous Approach for Modelling the System

Here, the approach by [22] for the model of the system is outlined to gain an understanding of their model and how it is used for the control design.

The mechanical model of the robot is simplified to an inverted pendulum on a cart in the planar case with a single input as seen in Figure 3.2.

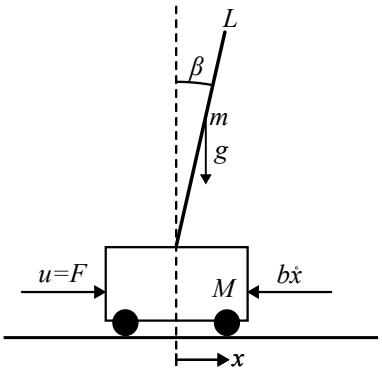


Figure 3.2: Inverted pendulum on a cart (b: viscous friction, F: external force as the system's input u, g: gravitational acceleration, L: length of the pendulum, m: mass of the pendulum, M: mass of the cart, β: pitch angle).

The pendulum represents the frame of the robot and is assumed to have an evenly distributed mass. The system has two degrees of freedom; the pitch angle of the pendulum β, and horizontal displacement of the cart x. u is the input to the system in the form of a force F.

State space representation is used to describe the linear time-invariant (LTI) system. Here, the LTI system is a set of n first-order differential equations which is written in matrix/vector form as given in (3.1).

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.1a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{3.1b}$$

$\mathbf{x}$ is the state vector that contains the n number of state variables. The order of the system is determined by the dimension of the state vector. $\mathbf{u}$ and $\mathbf{y}$ are the input- and output vector of the system. $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are the system matrix, the input matrix, the output matrix, and the direct coupling matrix.

Through a dynamic analysis of the system found in Appendix A, the state space model of the system is given by (3.2).

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \beta \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+mL^2)b}{I(m+M)+MmL^2} & \frac{m^2gL^2}{I(m+M)+MmL^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mLb}{I(m+M)+MmL^2} & \frac{-bmgL(m+M)}{I(m+M)+MmL^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \beta \\ \dot{\beta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+mL^2}{I(m+M)+Mm^2} \\ 0 \\ \frac{mL}{I(m+M)+Mm^2} \end{bmatrix} u \tag{3.2a}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \beta \\ \dot{\beta} \end{bmatrix} \tag{3.2b}$$

Instead of making use of the state space model to derive a transfer function, the previous group realised their model through Matlab Simulink using the Simscape package.

The two BLDC motors' stator voltage is modelled by an equivalent inductance-resistance circuit by neglecting the back emf.

The control system is a three-layer cascade control, controlling the pitch angle of the robot. The inner loop controls the motor current using a PI controller tuned by the ODrive to have a bandwidth of $100\,\mathrm{rad/s}$ based on ODrive's recommendations. The middle loop controls the two BLDC motor's speed using a proportional-integral (PI) controller tuned based on the same recommendations. The outer loop controls the pitch angle of the robot using a PID controller.

A Simulink tool is used to create a transfer function of the two inner loops. Matlab's SISO tool is applied to tune the pitch angle PID controller and ensure a stable response based on the determined transfer function.

3.1.2 Experiments and Model Testing

Initially, the pitch angle PID controller for the system is tested. It does not yield a desirable results as the system becomes unstable during testing. Furthermore, the model of the two inner loops cannot be validated.

To accommodate the issue, a new controller is proposed which is tuned based on an iterative approach and yields a system that is stable with an empty, half full, and full paint bucket. Additionally, a Kalman Filter is applied to filter the sensor noise from the gyroscope readings.

Using the new controller seven experiments with the robot are conducted. Three different surfaces are used in these experiments: a flat surface inside, a grass surface, and a tilted surface of 6° inside. Only the first and last of them are recorded as the grass surface is deemed too uneven and the low clearance between the frame and the ground resulted in the robot getting stuck, thus only the recorded cases are included. For these surfaces the following experiments are conducted, and the results are estimated based on the data given in [22]:

- On a flat surface indoors:
  - Self-balancing tests:
    Make the robot balance itself:
    - With empty tank, the pitch angle oscillates between to -0.17° and 0.15°
    - With half full tank, the pitch angle oscillates between -0.35° and 0.41°
    - With full tank, the pitch angle oscillates between -0.27° and 0.26°
  - Applying a step input of 1.33° pitch angle with varying levels of liquid:
    - With empty tank
      - Overshoot: $\frac{2.04-1.33}{1.33} \cdot 100 = 53.4\,\%$
      - Steady state error: $\frac{1.48-1.33}{1.33} \cdot 100 = 11.3\,\%$
      - Rise time: $0.154\,\mathrm{s}$
    - With half full tank
      - Overshoot: $\frac{2.15-1.33}{1.33} \cdot 100 = 61.7\,\%$

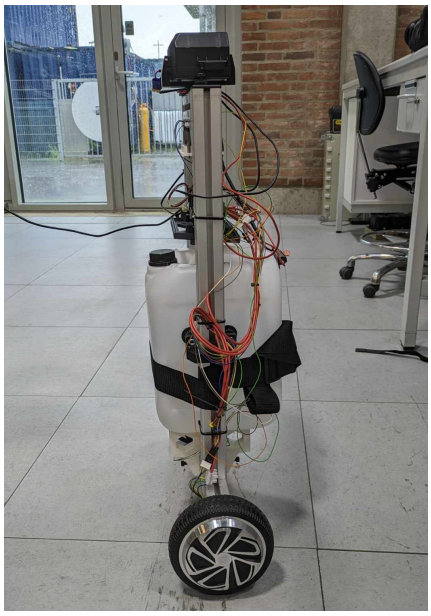
        - Steady state error: $\frac{1.47-1.33}{1.33} \cdot 100 = 10.5\,\%$
        - Rise time: $0.170\,\text{s}$
      - With full tank
        - Overshoot: $\frac{2.13-1.33}{1.33} \cdot 100 = 60.2\,\%$
        - Steady state error: $\frac{1.49-1.33}{1.33} \cdot 100 = 12.0\,\%$
        - Rise time: $0.158\,\text{s}$
- On a tilted surface of 6° indoors:
  - Self-balancing tests:
    Make the robot balance itself:
    - With an empty tank
      - The robot's angle is shown to oscillate between 1.11° and 1.60°, meaning a peak-to-peak value of 0.49°

During the experiments, a number of critical problems are encountered, which are mentioned in the following:

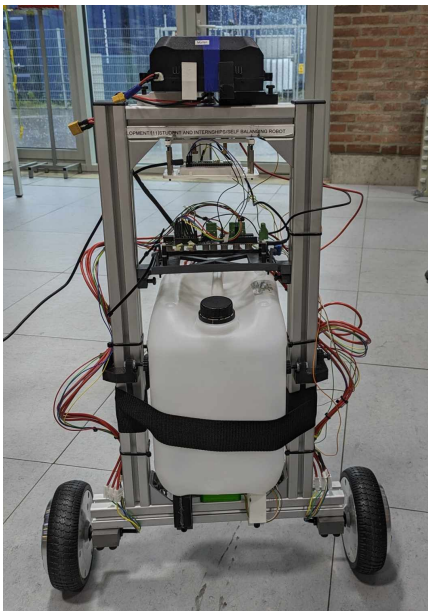
- Inadequate model
  The previous group deemed their model inadequate as it is not validated from the experiments. The previous group suggests that this could be due to the lack of introducing the dynamics of the filter in the system.
- IMU is not fixated properly
  As the IMU is movable, it can potentially cause excess noise in the feedback.
- Ground clearance
  It experiences issues with uneven surfaces as the ground clearance is low and as a result the frame can collide with the ground.
- Inadequate MCU
  The MCU is no longer able to receive or transmit data, and will therefore need to be substituted. The previous group presumes that the MCU lacks processing power. Additionally, it is not possible to implement feedback from the controller and the wireless communication as it does not have a sufficient number of serial ports.
- Breakage of the existing supports for the paint bucket
  Some of the supports for the paint bucket are broken and will have to be replaced.

3.2 New Prototype

Based on the experiences of the previous group, it is chosen to modify the prototype to avoid similar issues and improve the design. The modified prototype with the paint tank is seen in Figure 3.3.



(a) Side view without paint tank.



(b) Front view without paint tank.

Figure 3.3: The modified prototype viewed from the front and side.

3.2.1 Solutions to Identified Problems

To resolve the critical problems the following solutions are chosen:

- Secure the IMU to the frame to moderate the noise.
- Mount the wheels on the lower side of the frame in order to increase the ground clearance resulting in a higher centre of mass.
- Implementation of a new MCU, with increased clock speeds and additional serial communication ports.
- New supports for paint bucket due to breakage of the existing supports.

Inertial Measurement Unit

The IMU is upgraded to a BNO055 Intelligent 9-axis absolute orientation sensor [42]. The switch from the GY-91 to BNO055 is made as a spare BNO055 is available at no additional costs.
The accuracy of the two sensors are comparable with the primary difference being software implementation.
The BNO055 offers an integrated sensor fusion algorithm which calibrates the gyroscope, accelerometer, and magnetometer. After this calibration, the BNO055 calculates the absolute orientation of the IMU based on a fusion of the three sensors' output when it is set in the default mode. The fusion of signals is done by the BNO055's onboard MCU and thereby allows for faster code integration compared to the GY-91 where the signals needs to be fused manually on an external MCU. The signals are fused in order to obtain outputs with greater reliability, as they are less prone to sensor reading outliers and external factors. The working principle of the IMU and a description of why the signals are fused is described in Appendix B.

Inertial Measurement Unit Mount

On the original prototype the IMU is fastened to a perfboard mounted to the frame within an acrylic casing. Therefore a new mount is designed as shown in Figure 3.4 in order to mitigate the noise connected with vibrations of the mount.

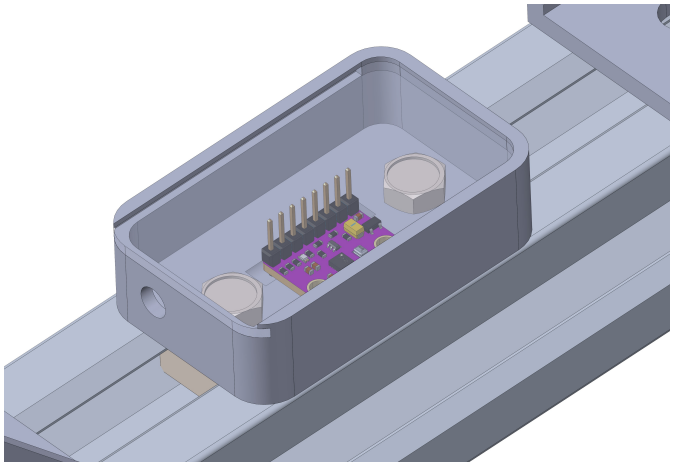


Figure 3.4: Mount and casing to the IMU.

Microcontroller Unit

The MCU is upgraded to the Teensy 4.0 [43] as the original Arduino Nano MCU is unresponsive, and therefore needs to be substituted. The ARM Cortex-M7 600 MHz processor of the Teensy provides a significant increase in processing power compared to the Arduino Nano's ATmega328 16 MHz processor, and it provides seven serial ports as opposed to the single serial port of the Arduino Nano, and therefore allows for more sensor inputs. The operating voltage of the Teensy and Arduino Nano is the same, 5 V, but the Teensy draws a current of 100 mA as opposed to the 19 mA of the Arduino Nano. Furthermore the prices of the Teensy 4.0 and the Arduino Nano are within 1 USD [44] [45].

Microcontroller Unit Mount

Due to the requirements of the IMU mount, it is chosen to move the MCU and other electrical components to the top of the frame below the battery as seen in Figure 3.5.

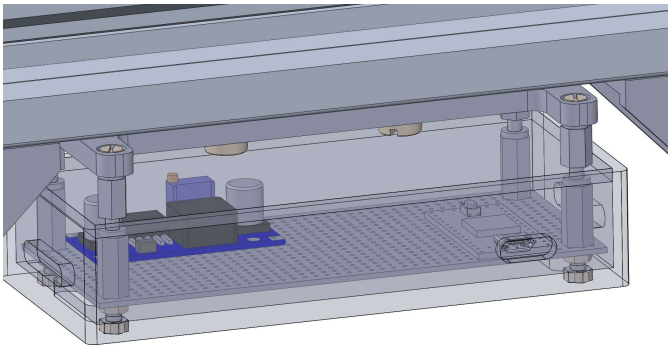


Figure 3.5: Mount and casing to the MCU and buck converter.

Paint Bucket Support

The paint bucket supports broke during the testing of the existing prototype. New supports are not 3D printed, instead, the paint bucket is supported using luggage straps to ensure quick release and exchange of paint buckets while being lightweight and cheap, as shown in Figure 3.3.

3.2.2 System Requirements for Prototype

Before completing the modifications and changes to the robot, the system requirements are needed to get an overview of the wanted capabilities and how to achieve these. The system of requirements are seen in Table 3.1.

Table 3.1: List of system requirements.

Requirement	Description
Able to compete with Turf Tank's current solution	Straight line speed of 1 m/s. Straight line tolerance of ± 5 mm.
Capable of steering	Able to control each wheel independently
Capable of self-balancing	Able to balance itself on flat and tilted surfaces.
Wireless communication	Able to send and receive data from the computer directly to the robot through a wireless communication.
Able to prevent collisions with obstacles	Capable of detecting obstacles and stopping before colliding with them.
Superior control compared to the original one	During the self-balancing test it has to oscillate less than the following under the specified conditions: • On a flat surface indoors: – With empty tank: the peak-to-peak value of the angle needs to be less than 0.32° – With half full tank: the peak-to-peak value of the angle needs to be less than 0.76° – With full tank: the peak-to-peak value of the angle needs to be less than 0.53° • On a tilted surface of 6° indoors: – With empty tank: the peak-to-peak value of the angle needs to be less than 0.49° Applying a step input of 1.33° for the pitch angle, the response demonstrate better performance than: • With empty tank: – Overshoot must be lower than 54.3 % – Steady state error must not exceed 11.3 % – Rise time must be less than 0.154 s • With half full tank: – Overshoot must be lower than 61.7 % – Steady state error must not exceed 10.5 % – Rise time must be less than 0.170 s • With full tank: – Overshoot must be lower than 60.2 % – Steady state error must not exceed 12.0 % – Rise time must be less than 0.158 s

3.2.3 Further Development

To extend the capabilities of the prototype to those listed in Table 3.1, a new set of components are introduced and described below.

Wireless Serial Communication

As an alternative to having the joystick module physically connected to the MCU, two HC-12 transceivers [46] are included. The purpose of the HC-12 transceivers is to allow for wireless communication between the user and the prototype. This is done by connecting one of the HC-12 transceivers to the MCU, and the other to a spare Arduino UNO MCU which is readily available. Furthermore, the joystick module is connected to the Arduino UNO, thereby making it possible to send position inputs from a computer to the prototype while receiving e.g. sensor information from the IMU. Each of the HC-12 transceivers draws a continuous current of 100 mA and is supplied with a voltage of 5 V. According to [46], a 1N4007 diode has to be connected in series with its power supply when it is supplied with a voltage greater than 4.5 V in order to avoid heating of its low-dropout regulator. Furthermore, a 330 µF 25 V decoupling capacitor is connected between the power supply input and ground in order to reduce the signal noise produced by surrounding components.

Proximity Sensors

As an addition to the prototype two HC-SR04 [47] ultrasonic sensors are implemented. These sensors supplies distance measurements to objects in front of and behind the prototype. Based on these measurements the prototype can be programmed to stop if objects appear within a certain distance, and thereby reducing the risk of collision with surrounding objects. The HC-SR04 measures distances between 20 and 400 cm with a resolution of 0.3 cm. The HC-SR04 draws a current of 15 mA and is supplied an operating voltage of 5 V.

In Figure 3.6 the main components and the corresponding connections are depicted, while Figure 3.7 describes the circuit for wireless communication between the users PC and the MCU. Lastly Table 3.2 describes the function of the wires. The ODrive is initialised using the python code shown in Appendix C.

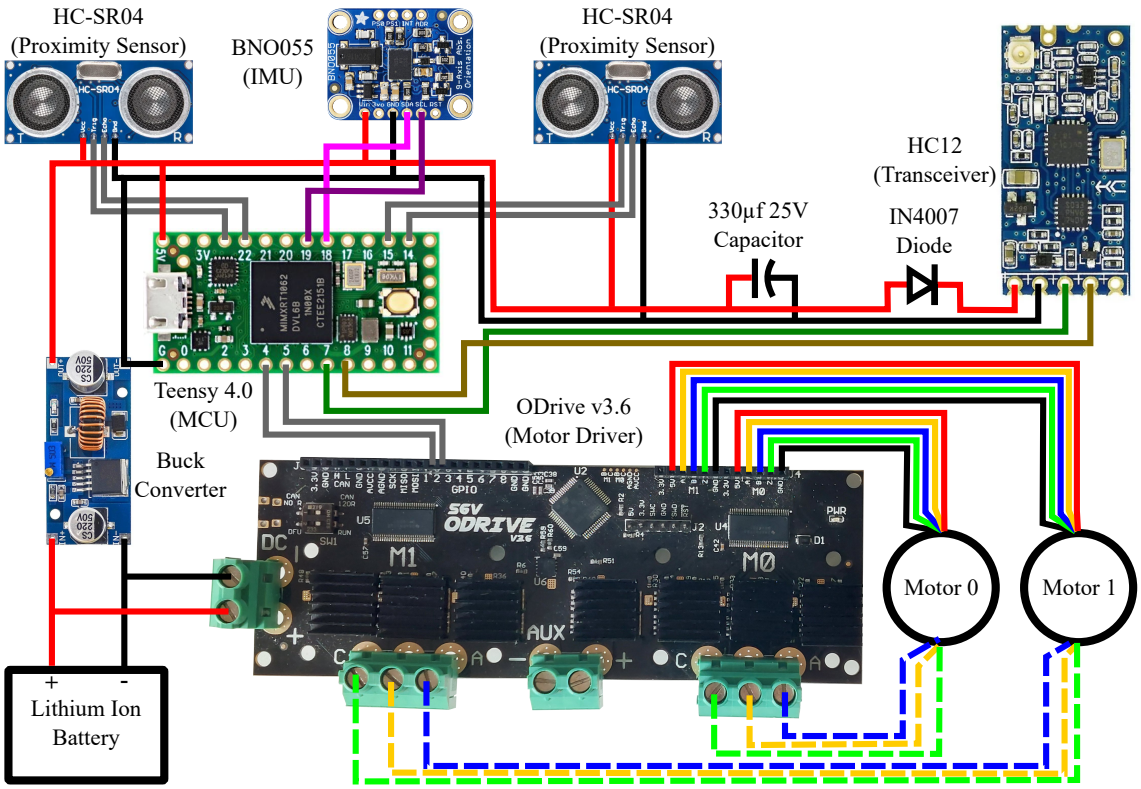


Figure 3.6: Wiring diagram of electrical components on the robot (IMU: inertial measurement unit, MCU: microcontroller unit).

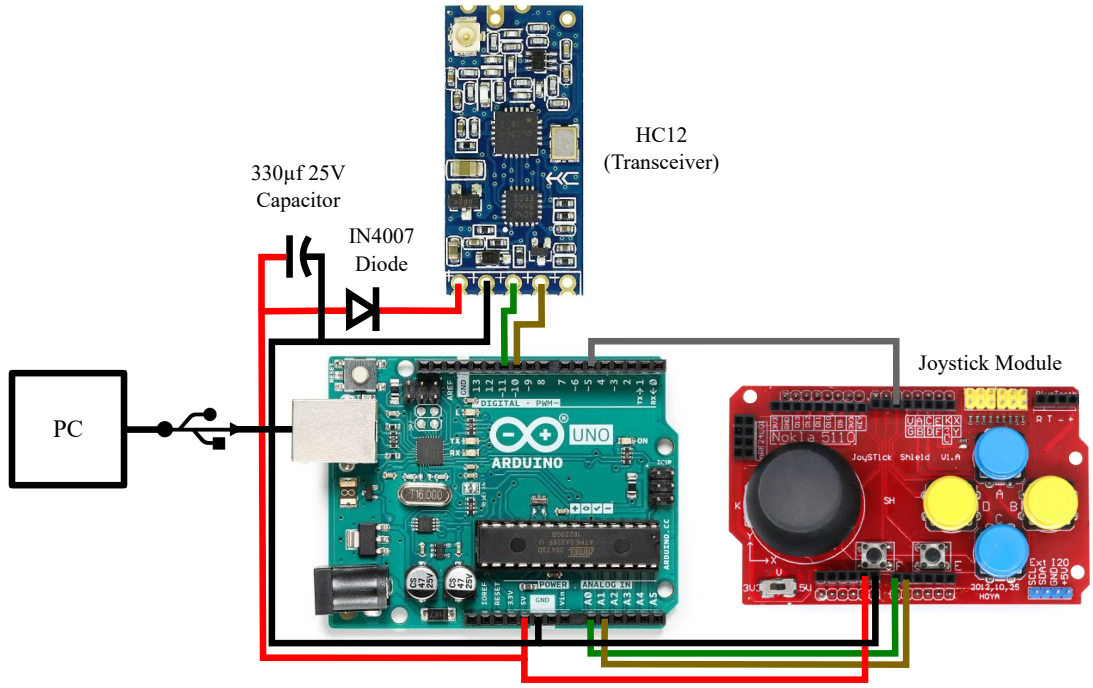


Figure 3.7: Wiring diagram of wireless part.

Table 3.2: Wire colour description.

Colour	Meaning	Colour	Meaning
Red	Power input	Blue Dashed	Motor Coil A Input
Black	Ground	Yellow Dashed	Motor Coil B Input
Pink	Serial Data Line	Light Green Dashed	Motor Coil C Input
Purple	Serial Clock Line	Yellow Full	Hall Sensor A Output
Dark Green	Receive	Blue Full	Hall Sensor B Output
Brown	Transmit	Light Green Full	Hall Sensor C Output
Grey	General-Purpose Input/Output		

3.3 Summary

A description of the original prototype is made, it covers the physical and electrical components and their uses. Furthermore, the planar model of the prototype is presented and used for the system modelling and control. The results obtained from the previous experiments and the encountered issues are documented. These results serves as a baseline for the control of the prototype and the encountered issues are resolved. Subsequently, a set of system requirements are made, and new components which fulfils the requirements are implemented.

Chapter 4

Modelling of the Mobile Wheeled Inverted Pendulum System

As a foundation for analysis and control design, mechanical and electrical models of the self-balancing robot are developed. Furthermore an overview of the system is presented in a block diagram that describes how it functions.

4.1 Mechanical Model

A three-dimensional mechanical model is made in order to analyse the full dynamics of the MWIP. This allows for the design of a controller that controls both the pitch and yaw of the MWIP and to analyse the coupling between them.

The mechanical model is derived using Lagrangian mechanics where the frame, paint, and paint tank are considered as solid masses. The changes in the CM and inertia of the paint tank are considered as a function of the paint level in the tank, while the fluid of the paint is assumed to be at rest. Thus, sloshing of the paint is not included in the mechanical model.

To derive the mechanical model of the self-balancing robot, the following assumptions are made:

- The robot is on a level surface i.e. $\dot{z} = 0$.
- Frame and bucket cannot roll.
- Tank yaws and pitches with frame.
- Wheels cannot slip.
- Sloshing effect is simplified to the standstill case.

The known parameters of the mechanical system are listed in Table 4.1.

The generalised coordinates consist of the yaw angle of the frame α, the pitch angle of the frame β, and distance travelled s as depicted in Figure 4.1. This gives the following generalised coordinate vector $\mathbf{q}$:

$$\mathbf{q} = \begin{bmatrix} \alpha & \beta & s \end{bmatrix}^T \tag{4.1}$$

Using the Euler-Lagrange equation, the joint torques $\boldsymbol{\tau}$ are defined by the Lagrangian L as:

$$\boldsymbol{\tau} = \frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} \tag{4.2}$$

The Lagrangian is defined as:

$$L = K - P \tag{4.3}$$

Table 4.1: Parameters of the mechanical system.

Parameter	Symbol	Value
Distance from the origin to the CM of the frame	L_{ACMF}	16.8 cm
Distance from the origin to the CM of the wheel	L_{ACMW}	24.4 cm
Mass of the frame	m_F	9.00 kg
Mass of the wheel	m_W	1.56 kg
Inertia of the frame	$J_{F,xx}$	1.28 kgm^2
	$J_{F,yy}$	1.01 kgm^2
	$J_{F,zz}$	0.283 kgm^2
Inertia of the wheel	$J_{W,xx}$	$3.00 \cdot 10^{-3}$ kgm^2
	$J_{W,yy}$	$3.50 \cdot 10^{-3}$ kgm^2
	$J_{W,zz}$	$3.50 \cdot 10^{-3}$ kgm^2

This means that the kinetic energy K and potential energy P must be found as functions of the generalised coordinates in order to derive the mechanical model. The kinetic energy is given by the mass and velocity of the CMs while the potential energy is given by the positions of the CMs.

4.1.1 Analysis of Change in Mass

The varying amount of paint in the paint tank yields a change in the tank's mass, and as a consequence its CM and inertia change. The paint container's mass m_c is measured to be 0.45 kg and the paint is assumed to have a density equal to water. The mass of the paint is found using the volume of the container, which is estimated to be 0.28x0.235x0.19 m. The mass of the paint m_P is given as a function of the paint level within the tank:

$$m_P = \rho_P V_P = \rho_P L_P w d \tag{4.4}$$

ρ_P and V_P denote the density and the volume of the paint, respectively. The paint level L_P varies from 0 to 0.28 m, and w_P and d_P denote the width and the depth of the container, respectively. Through SolidWorks the bottom of the container is measured to be 0.114 m above the origin. The distance from the origin to the CM of the tank L_{ACMT} is determined by assuming the container and paint to be two rigid bodies:

$$L_{ACMT} = \frac{m_P \frac{L_P}{2} + m_c \frac{h}{2}}{m_P + m_c} + 0.114 \tag{4.5}$$

Where h is the height of the container.
The inertia tensor of a rigid body is defined as:

$$\mathbf{J} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{yz} & J_{zz} \end{bmatrix}$$

The off diagonal terms are considered negligible, the inertia of the rectangular cuboid about its centre is given as (4.6).

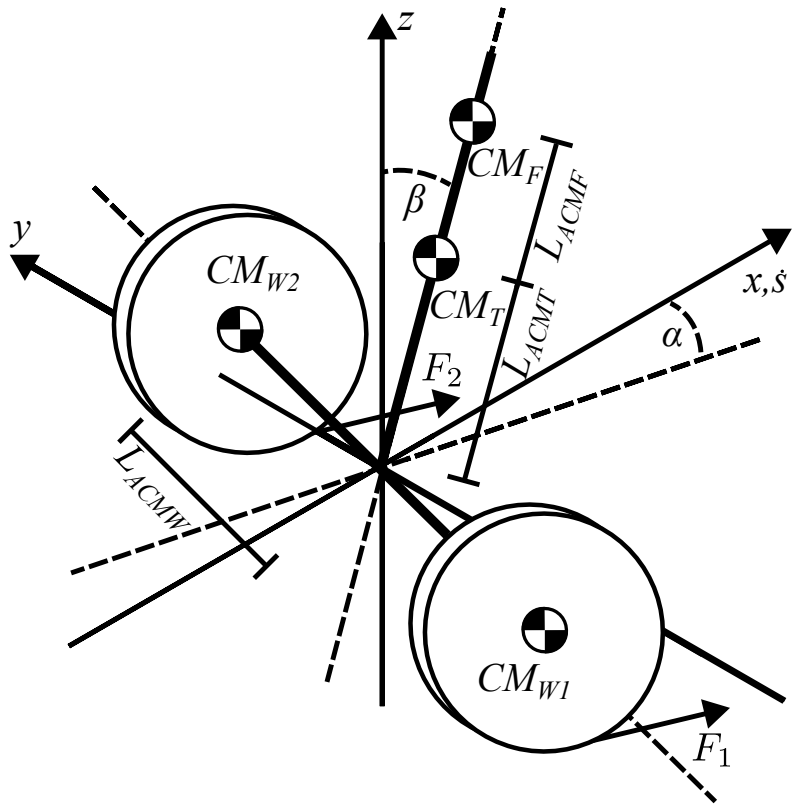


Figure 4.1: Illustration of the system (CM_F: centre of mass of the frame, CM_T: centre of mass of the tank, CM_{Wi}: centre of mass of wheel $i = \{1,2\}$, F_i: force applied by motor $i = \{1,2\}$, L_{ACMF}: distance from the origin to CM of frame, L_{ACMT}: distance from the origin to the CM of the tank, $\dot{s}$: linear velocity of the MWIP system, α: yaw angle, β: pitch angle).

$$\mathbf{J}_c = \begin{bmatrix} \frac{1}{12}m\left(h^2+d^2\right) & 0 & 0 \\ 0 & \frac{1}{12}m\left(h^2+w^2\right) & 0 \\ 0 & 0 & \frac{1}{12}m\left(d^2+w^2\right) \end{bmatrix} \tag{4.6}$$

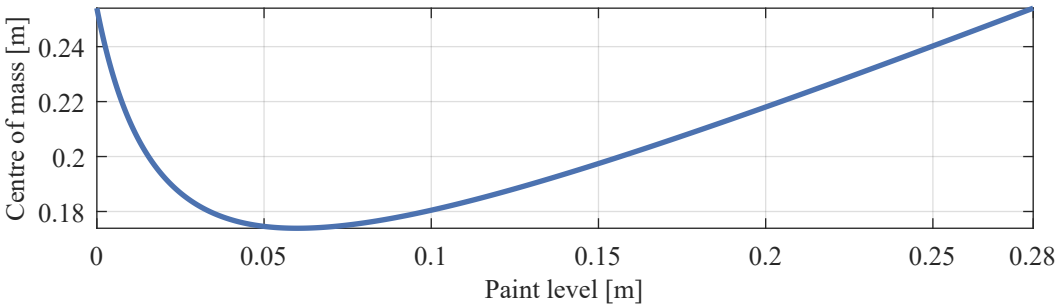
Where $\mathbf{J}_c$ is the inertia tensor of the container, and the mass m is the mass of the container and the paint within it. In order to determine the inertia about the wheel axis (the axis that coincides with the centre of both wheels), the parallel axis theorem given below is used:

$$\mathbf{J}_T = \mathbf{J}_c + m\mathbf{l}^2 \quad \text{where} \quad \mathbf{l} = \begin{bmatrix} L_{ACMT} & L_{ACMT} & 0 \end{bmatrix}^T$$

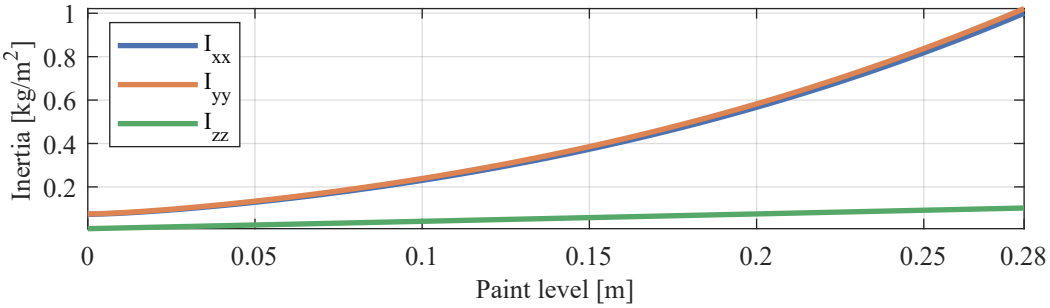
Combining the two above equations the inertia of the tank and paint is given below:

$$\mathbf{J}_T = \begin{bmatrix} \frac{1}{12}\left((m_P+m_c)(h^2+d^2)\right)+(m_P+m_c)L_{ACMT}^2 & 0 & 0 \\ 0 & \frac{1}{12}\left((m_P+m_c)(h^2+w^2)\right)+(m_P+m_c)L_{ACMT}^2 & 0 \\ 0 & 0 & \frac{1}{12}\left((m_P+m_c)(d^2+w^2)\right) \end{bmatrix} \tag{4.7}$$

The CM and the diagonal of the inertia tensor are seen plotted as a function of the paint level in the container in Figure 4.2.



(a) Centre of mass as a function of paint level.



(b) Inertia of tank about the origin.

Figure 4.2: Variation in the tank properties due to the paint level.

A minimum for the CM is observed at 0.055 m of paint level. The minimum is caused by the differing rate of change of the paint's and tank's contribution as seen in Appendix D.

4.1.2 Lagrangian Mechanics

The derivation is in accordance with Figure 4.1. The positions of the CMs in the local coordinate system are defined as:

$$\mathbf{P}_{CMF,local} = \mathbf{A}_1\mathbf{S}_1 \tag{4.8a}$$

$$\mathbf{P}_{CMW1,local} = \mathbf{A}_2\mathbf{S}_2 \tag{4.8b}$$

$$\mathbf{P}_{CMW2,local} = \mathbf{A}_3\mathbf{S}_2 \tag{4.8c}$$

$$\mathbf{P}_{CMT,local} = \mathbf{A}_1\mathbf{S}_3 \tag{4.8d}$$

Here, the distance the frame travels in the global coordinate system is not taken into account.

Where $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ are three-dimensional rotation matrices given as:

$$\mathbf{A}_1 = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta - \frac{\pi}{2}) & 0 & \sin(\beta - \frac{\pi}{2}) \\ 0 & 1 & 0 \\ -\sin(\beta - \frac{\pi}{2}) & 0 & \cos(\beta - \frac{\pi}{2}) \end{bmatrix} \tag{4.9a}$$

$$\mathbf{A}_2 = \begin{bmatrix} \cos(\alpha+\frac{\pi}{2}) & -\sin(\alpha+\frac{\pi}{2}) & 0 \\ \sin(\alpha+\frac{\pi}{2}) & \cos(\alpha+\frac{\pi}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9b}$$

$$\mathbf{A}_3 = \begin{bmatrix} \cos(\alpha-\frac{\pi}{2}) & -\sin(\alpha-\frac{\pi}{2}) & 0 \\ \sin(\alpha-\frac{\pi}{2}) & \cos(\alpha-\frac{\pi}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9c}$$

$\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$ are local lengths given as:

$$\mathbf{S}_1 = \begin{bmatrix} L_{ACMF} \\ 0 \\ 0 \end{bmatrix} \tag{4.10a}$$

$$\mathbf{S}_2 = \begin{bmatrix} L_{ACMW} \\ 0 \\ 0 \end{bmatrix} \tag{4.10b}$$

$$\mathbf{S}_3 = \begin{bmatrix} L_{ACMT} \\ 0 \\ 0 \end{bmatrix} \tag{4.10c}$$

The velocities of the CMs are given as the time derivative of the positions plus the velocity of the frame moving in the global coordinate system:

$$\dot{\mathbf{P}}_{CMF} = \dot{P}_{CMF,local} + \begin{bmatrix} \dot{s}\cos(\alpha) \\ \dot{s}\sin(\alpha) \\ 0 \end{bmatrix} \tag{4.11a}$$

$$\dot{\mathbf{P}}_{CMW1} = \dot{P}_{CMW1,local} + \begin{bmatrix} \dot{s}\cos(\alpha) \\ \dot{s}\sin(\alpha) \\ 0 \end{bmatrix} \tag{4.11b}$$

$$\dot{\mathbf{P}}_{CMW2} = \dot{P}_{CMW2,local} + \begin{bmatrix} \dot{s}\cos(\alpha) \\ \dot{s}\sin(\alpha) \\ 0 \end{bmatrix} \tag{4.11c}$$

$$\dot{\mathbf{P}}_{CMT} = \dot{P}_{CMT,local} + \begin{bmatrix} \dot{s}\cos(\alpha) \\ \dot{s}\sin(\alpha) \\ 0 \end{bmatrix} \tag{4.11d}$$

The kinetic translational energy, K_T is given as:

$$K_T = \frac{1}{2}\left(m_F\dot{\mathbf{P}}_{CMF}^T\dot{\mathbf{P}}_{CMF} + m_W\dot{\mathbf{P}}_{CMW1}^T\dot{\mathbf{P}}_{CMW1} + m_W\dot{\mathbf{P}}_{CMW2}^T\dot{\mathbf{P}}_{CMW2} + m_T\dot{\mathbf{P}}_{CMT}^T\dot{\mathbf{P}}_{CMT}\right) \tag{4.12}$$

The kinetic rotational energy, K_R is given as:

$$K_R = \frac{1}{2}\left(\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ 0 \end{bmatrix}^T \mathbf{J}_F \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} + \dot{s}\, r \\ 0 \end{bmatrix}^T \mathbf{J}_W \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} + \dot{s}\, r \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} - \dot{s}\, r \\ 0 \end{bmatrix}^T \mathbf{J}_W \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} - \dot{s}\, r \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ 0 \end{bmatrix}^T \mathbf{J}_T \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ 0 \end{bmatrix} \right) \tag{4.13}$$

The kinetic energy is given as the sum of the translational and kinetic energy:

$$K = K_T + K_R \tag{4.14}$$

The potential energy is given as:

$$P = \begin{bmatrix} 0 & 0 & g \end{bmatrix} (m_F \mathbf{P}_{CMF} + m_W \mathbf{P}_{CMW1} + m_W \mathbf{P}_{CMW2} + m_T \mathbf{P}_{CMT}) \tag{4.15}$$

The kinetic and potential energy is then used in (4.3) and the Lagrangian is used in (4.2) to derive the joint torques.

The non-conservative externally applied torques, $\mathbf{Q}$ are given as:

$$\mathbf{Q} = \begin{bmatrix} \frac{T_1}{r} L_{ACMW} - \frac{T_2}{r} L_{ACMW} + \frac{2B_v L_{ACMW}}{r} \dot{\alpha} \\ 2B_v \dot{\beta} \\ \frac{T_1}{r} + \frac{T_2}{r} + \frac{2B_v}{r} \dot{s} \end{bmatrix} \tag{4.16}$$

Where T_1 and T_2 denote the torques delivered by the motors at wheel 1 and wheel 2, respectively. All terms in the Euler-Lagrange equation are now known, and the dynamic equations of motion can be represented using the general form given in (4.17).

$$\boldsymbol{\tau} = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{4.17}$$

Where $\mathbf{D}(\mathbf{q})$ is the inertia matrix consisting of masses and moments of inertia terms, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis and centripetal vector consisting of velocity product terms, and $\mathbf{G}(\mathbf{q})$ is the gravity vector. The inertia matrix is given as:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}(1,1) & 0 & 0 \\ 0 & \mathbf{D}(2,2) & \mathbf{D}(2,3) \\ 0 & \mathbf{D}(3,2) & \mathbf{D}(3,3) \end{bmatrix} \tag{4.18}$$

where the entries are:

$$\begin{aligned} \mathbf{D}(1,1) = {} & \mathbf{J}_{F,(x,x)} + \mathbf{J}_{T,(x,x)} + 2\mathbf{J}_{W,(x,x)} + \frac{L_{ACMT}^2 (m_p + m_c)}{2} + \frac{L_{ACMF}^2 m_F}{2} + 2L_{ACMW}^2 m_W \\ & + \frac{L_{ACMT}^2 \cos(2\beta)(m_p + m_c)}{2} + \frac{L_{ACMF}^2 m_F \cos(2\beta)}{2} \end{aligned} \tag{4.19a}$$

$$\mathbf{D}(2,2) = m_F L_{ACMF}^2 + (m_p + m_c) L_{ACMT}^2 + \mathbf{J}_{F,(y,y)} + \mathbf{J}_{T,(y,y)} + 2\mathbf{J}_{W,(y,y)} \tag{4.19b}$$

$$\mathbf{D}(2,3) = \mathbf{D}(3,2) = -\sin(\beta)(L_{ACMF} m_F + L_{ACMT} m_p + L_{ACMT} m_t) \tag{4.19c}$$

$$\mathbf{D}(3,3) = m_F + 2m_W + m_p + m_c + \frac{2\mathbf{J}_{W(y,y)}}{r^2} \tag{4.19d}$$

The Coriolis and centripetal vector is given as (4.20).

$$\mathbf{C} = \begin{bmatrix} \dot{\alpha}\dot{\beta}\sin(2\beta)(L_{ACMF}^2 m_F + L_{ACMT}^2 m_p + L_{ACMT}^2 m_t) \\ -\frac{1}{2}(\dot{\alpha}^2\sin(2\beta)(L_{ACMF}^2 m_F + L_{ACMT}^2 m_p + L_{ACMT}^2 m_t)) \\ \dot{\beta}^2\cos(\beta)(L_{ACMF} m_F + L_{ACMT} m_p + L_{ACMT} m_t) \end{bmatrix} \tag{4.20}$$

The gravity vector is given as (4.21).

$$\mathbf{G} = \begin{bmatrix} 0 \\ -\cos(\beta)(L_{ACMF} m_F + L_{ACMT} m_p + L_{ACMT} m_t) \\ 0 \end{bmatrix} \tag{4.21}$$

The accelerations of the generalised coordinates are found by solving (4.17) for $\ddot{\mathbf{q}}$:

$$\ddot{\mathbf{q}} = -\mathbf{D}^{-1}(\mathbf{C} + \mathbf{G} - \mathbf{Q}) \tag{4.22}$$

The Matlab script used to calculate the Lagrangian and solve for the acceleration are given in Appendix E.2.

4.2 Electrical Model

To describe the characteristics of the actuation of the system, an electrical model of the three-phase BLDC motor is developed. The notation presented in [48] is used for the derivation, and symmetrical windings are assumed. The BLDC motors' parameters used in the development of the electrical model are listed in Table 4.2. The stator resistances and inductances are measured using the ODrive's onboard software, while the torque constants and the Kv ratings are measured in Appendix F.

Table 4.2: Parameters of the BLDC motors (counts-per-revolution (cpr.)).

Parameter	Symbol	Value		Unit
		Motor 0	Motor 1	
Stator resistance	R_s	0.246	0.244	Ω
Stator inductance	L_s	0.408	0.401	mH
Torque constant	K_T	0.303	0.305	N m/A
Wheel radii	r_w	0.0825	0.0825	m
Number of poles	P	30	30	[–]
Hall effect sensor cpr.	n_{cpr}	90	90	[–]
Motor Kv rating	K_v	2.86	2.84	V s

As a step in the modelling of the BLDC motor, its driving principle is explained. The driving principle of a BLDC motor is to change the active phases depending on the permanent magnet position placed on the rotor in order to produce a continuous torque. Thus emulating the driving principle of direct current (DC) motor with brush commutators. A circuit diagram of the BLDC motor is shown in Figure 4.3.

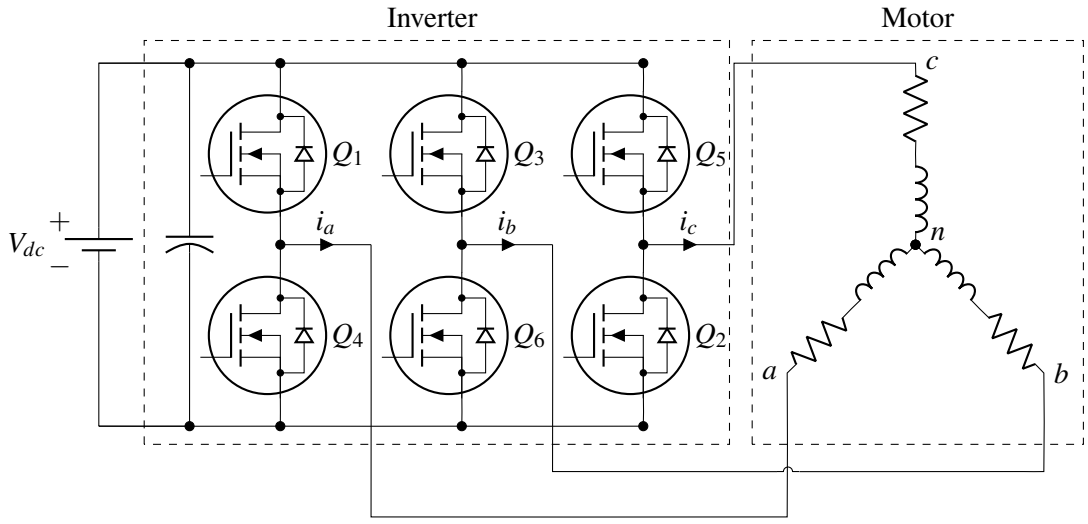


Figure 4.3: Circuit diagram of the BLDC motor (a: phase a, b: phase b, c: phase c, $i_{a,b,c}$: phase currents a, b, and c, n: neutral point, V_{dc}: Direct current voltage, Q_i: MOSFET where $i = \{1,..,6\}$).

In Figure 4.4 a two pole three-phase BLDC motor with Hall effect sensors is illustrated. The BLDC motor operate with electronic commutation by using the magnet position placed on the rotor side. The rotor position is detected by using Hall effect sensors placed within the motors that detects the generated magnetic fields. Based on the rotor position, the inverter activates two of the switches accordingly, allowing the current to flow in two of the phases and thereby producing a continuous torque.

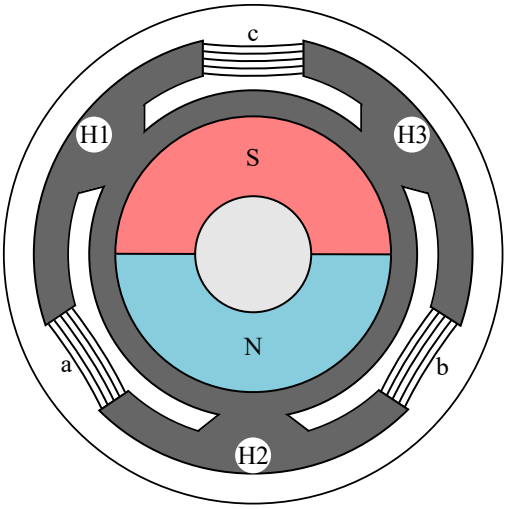


Figure 4.4: Sketch of a BLDC motor (a: phase a, b: phase b, c: phase c, H1: Hall effect sensor 1, H2: Hall effect sensor 2, H3: Hall effect sensor 3, N: north pole of permanent magnet, S: south pole of permanent magnet).

In Figure 4.5, the driving principle of a two-pole three-phase BLDC motor is illustrated including its Hall sensor outputs, back emfs, phase currents, and torque during operation [48].

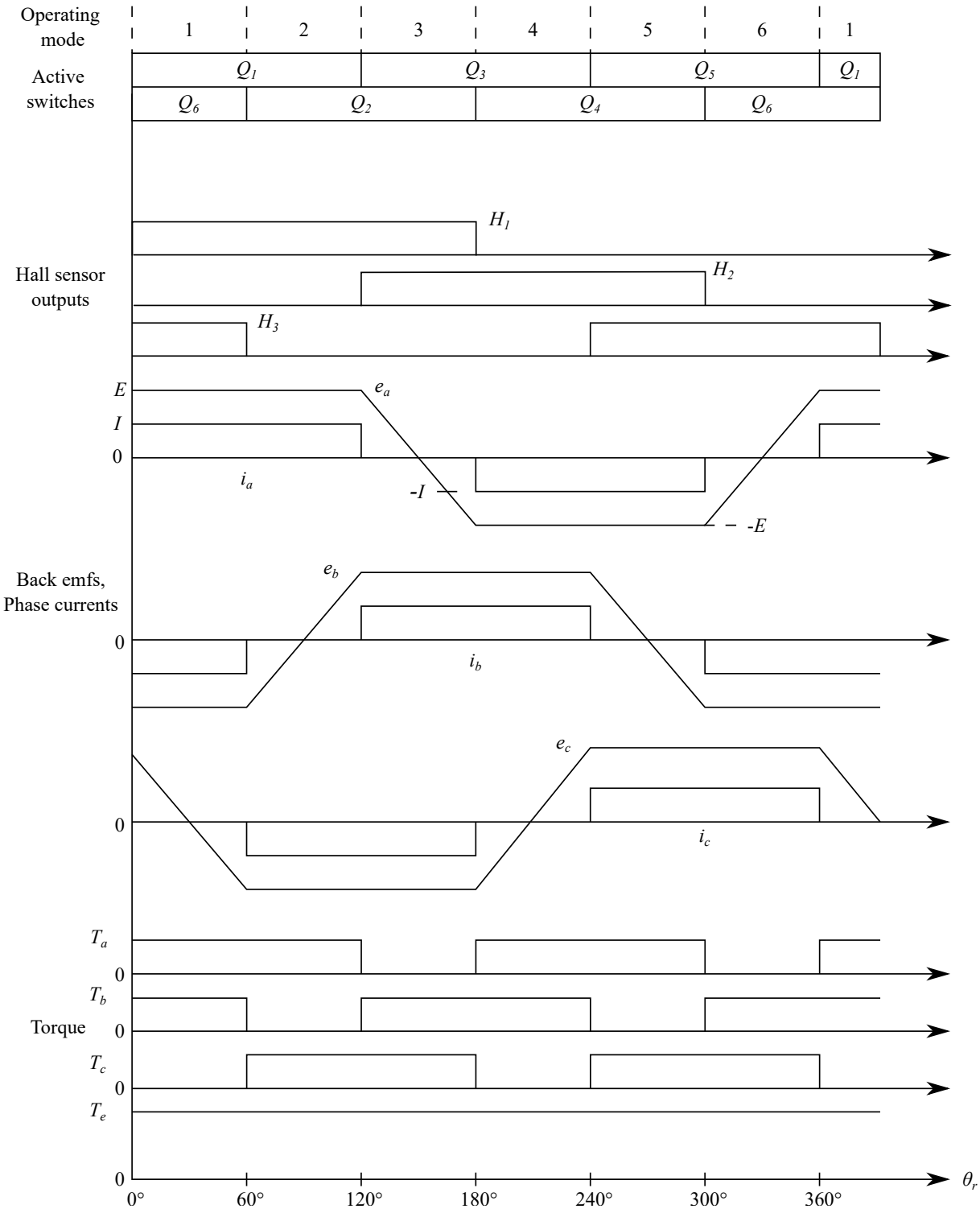


Figure 4.5: Driving principle of the BLDC motor ($e_{a,b,c}$: back emf of phase a, phase b, and phase c, H_i: Hall sensor output $i = 1,2,3$, $i_{a,b,c}$: phase current a, b, and c, $T_{a,b,c}$: torque generated by phase a, b, and c, T_e: output torque, Q_i: switches of the inverter $i = \{1,2,...,6\}$, θ_r: mechanical angle).

In the figure, the electrical cycle consists of six different sections as there is only one pole pair. However, the implemented BLDC motors have fifteen pole pairs, and as there are six states for each pole pair, the number of sections is ninety in each revolution of the motor. The back emfs are

trapezoidal, and the phase currents are constant, resulting in non-sinusoidal flux as it depends on these currents. As a result, the BLDC motors utilise their three-phase quantities for control as opposed to alternating current (AC) motors which use dq axes frame for their control. The dq transformation is used for sinusoidal quantities where the three-phase currents are transformed into the currents of the d- and q-axes, but as BLDC motors' quantities are non-sinusoidal such a transform is nonsensical [48].

To begin the derivation of the model of the three-phase BLDC motor, the voltage equations are considered. The stator voltage equation consists of the voltage drop due to the stator resistance $\mathbf{R}_s$ and the rate of change of the stator flux linkage of the windings $\boldsymbol{\lambda}_{abcs}$:

$$\mathbf{v}_{abcs} = \mathbf{R}_s \mathbf{i}_{abcs} + \frac{d\boldsymbol{\lambda}_{abcs}}{dt} \tag{4.23}$$

The subscript abc denotes the three windings while s denotes the stator. The four vector and matrix quantities are written as:

$$\mathbf{v}_{abcs} = \begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} \tag{4.24a}$$

$$\mathbf{R}_s = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \tag{4.24b}$$

$$\mathbf{i}_{abcs} = \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} \tag{4.24c}$$

$$\boldsymbol{\lambda}_{abcs} = \begin{bmatrix} \lambda_{as} \\ \lambda_{bs} \\ \lambda_{cs} \end{bmatrix} \tag{4.24d}$$

The flux linkage consists of the contribution from the stator currents and the permanent magnet placed in the rotor denoted with subscript (s) and (f) respectively.

$$\boldsymbol{\lambda}_{abcs} = \boldsymbol{\lambda}_{abcs(s)} + \boldsymbol{\lambda}_{abcs(f)} \tag{4.25}$$

Inserting (4.25) into (4.23) results in (4.26).

$$\mathbf{v}_{abcs} = \mathbf{R}_s \mathbf{i}_{abcs} + \frac{d\boldsymbol{\lambda}_{abcs(s)}}{dt} + \mathbf{e}_{abcs} \quad \text{Where} \quad \mathbf{e}_{abcs} = \frac{d\boldsymbol{\lambda}_{abcs(f)}}{dt} \tag{4.26}$$

$\mathbf{e}_{abcs}$ is the back emf due to the magnet flux from the permanent magnet.

The stator flux linkage $\boldsymbol{\lambda}_{abcs(s)}$ produced by the stator current is given as (4.27).

$$\boldsymbol{\lambda}_{abcs(s)} = \mathbf{L}_s \mathbf{i}_{abcs} = \begin{bmatrix} L_{aa} & L_{ab} & L_{ac} \\ L_{ba} & L_{bb} & L_{bc} \\ L_{ca} & L_{cb} & L_{cc} \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} \tag{4.27}$$

The inductances with subscripts containing repeated letters denote the self inductance while the non repeating subscripts denote the mutual inductances. For the mutual inductances the first letter represents the specific winding while the second letter represents the current producing the flux linkage in the other winding.

Due to the assumption of symmetrical windings, the self inductances are equal and the mutual inductances also are equal. Thus, the inductances can be written as:

$$L_{aa} = L_{bb} = L_{cc} = L_s = L_{ls} + L_m \tag{4.28}$$

$$L_{ab} = L_{ac} = L_{ba} = L_{ca} = L_{cb} = -\frac{1}{2}L_m = M \tag{4.29}$$

Where L_{ls} and L_m denote the leakage inductance and magnetising inductance, respectively.

Using (4.27) to (4.29) for rewriting the voltage equation stated in (4.26):

$$\mathbf{v}_{abcs} = \mathbf{R}_s\mathbf{i}_{abcs} + \mathbf{L_{abcs}}\frac{d\mathbf{i}_{abcs}}{dt} + \mathbf{e}_{abcs} \tag{4.30}$$

$$\begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} L_s & M & M \\ M & L_s & M \\ M & M & L_s \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} e_{as} \\ e_{bs} \\ e_{cs} \end{bmatrix} \tag{4.31}$$

Due to the winding symmetry of the stator, the stator currents which flow through the neutral wire has the same quantities, but as they differ by 120° the vector sum of the three currents is: $i_{as} + i_{bs} + i_{cs} = 0$, which can also be seen be applying Kirchhoff's current low to the phase windings of Figure 4.3. Using this relation the second term of the equation is rewritten, e.g. for the first row it is written as:

$$\frac{d}{dt}\left(L_s i_{as} + M(i_{bs} + i_{cs})\right) = \frac{di_{as}}{dt}\left(L_s - M\right) \tag{4.32}$$

Rewriting the other rows of (4.31) as well, yields:

$$\begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} L_s - M & 0 & 0 \\ 0 & L_s - M & 0 \\ 0 & 0 & L_s - M \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix} + \begin{bmatrix} e_{as} \\ e_{bs} \\ e_{cs} \end{bmatrix} \tag{4.33}$$

The output torque equation of the BLDC motor is calculated using the electrical power P_e, which is the product of the output torque T_e and the mechanical angular velocity ω_r:

$$P_e = T_e\omega_r \qquad \text{Where} \quad \omega_r = \omega_e/PP \tag{4.34}$$

Where ω_e is the electrical angular velocity and PP is the number of pole pairs.

The electrical power can also be calculated as the product of the stator current and the voltage generated by the back emf:

$$P_e = i_{as}e_{as} + i_{bs}e_{bs} + i_{cs}e_{cs} \tag{4.35}$$

By using (4.35) and rewriting (4.34), the torque is determined as (4.36).

$$T_e = \frac{i_{as}e_{as} + i_{bs}e_{bs} + i_{cs}e_{cs}}{\omega_r} = 2\frac{EI}{\omega_r} \tag{4.36}$$

4.2.1 Control of the Brushless DC motor

Assuming the motor driver applies proper commutation of the phase currents, the control of the BLDC motor is the similar to that of a DC motor. Under the assumption of proper commutation, the BLDC motor's operating speed is proportional to the applied DC voltage [48]. Therefore, the circuit diagram in Figure 4.3 is simplified to the equivalent circuit model depicted in Figure 4.6.

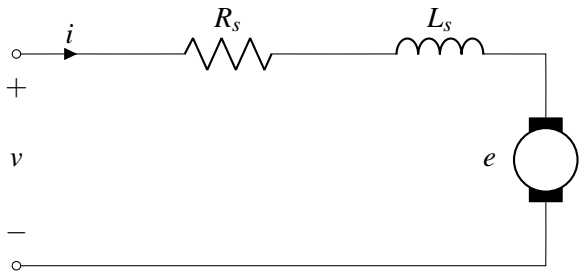


Figure 4.6: Equivalent circuit model of the BLDC motor (e: back emf, i: current, L_s: stator inductance, R_s: stator resistance, v: voltage).

The voltage equation of the equivalent circuit model is given by (4.37).

$$v = R_s i + L_s \frac{di}{dt} + e \tag{4.37}$$

Where v, i, and e are voltage, current, and the back emf induced by the rotor rotating its magnetic field. R_s and L_s are stator resistance and stator inductance, respectively.

The back emf is estimated through (4.38).

$$e = K_v \omega_r \tag{4.38}$$

Where K_v is the Kv rating of the motor.

As is seen the back emf is estimated through (4.38) by measuring the angular velocity, thereby allowing feedforward compensation of the back emf and eliminating it in (4.37). Thereby, a first order transfer function with voltage as the input and current as output is established:

$$\frac{I(s)}{V(s)} = \frac{1}{L_s\, s + R_s} \tag{4.39}$$

The output torque is determined as the product of the current and the torque constant:

$$T_e = K_T i \tag{4.40}$$

4.3 Overview of the System

To provide an overview of the system, a block diagram is made which is shown in Figure 4.7. The MCU is responsible for managing the settings of the IMU and sending the measured data to the PC. Initially, it is decided to use the information generated by the IMU to measure the following states: α,

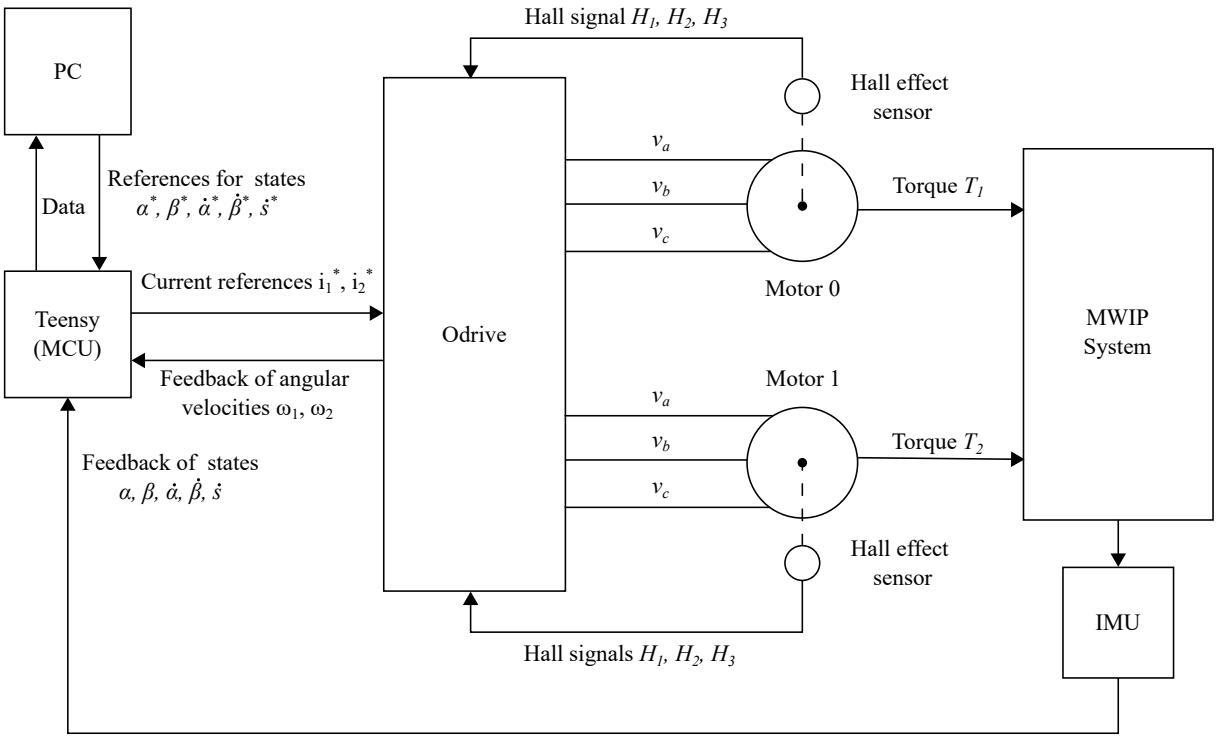


Figure 4.7: Diagram of the system ($v_{a,b,c}$: phase voltages a, b, and c, IMU: inertial measurement unit, MCU: microcontroller unit, MWIP: mobile wheeled inverted pendulum, $\dot{s}$: linear velocity of the MWIP system, α: yaw angle, β: tilt angle).

$\dot{\alpha}$, β, $\dot{\beta}$, and $\dot{s}$. The ODrive is responsible for controlling the motors using the method of field oriented control (FOC). The method is for AC motors and therefore uses sinusoidal phase currents for the motor in order to produce a continuous torque, as explained in Section 4.4. However, as explained in Section 4.2, the BLDC motors have a trapezoidal back emf, therefore the sinusoidal phase currents causes the motors to produce a noncontinuous torque. This is also mentioned in [49] where it is stated that the controllability for motors with trapezoidal back emf may be reduced. Therefore an investigation of the waveform of the motors' back emf is performed to determine if the controllability is affected. The diagram of the ODrive is given in Figure 4.8.

In this thesis, it is chosen to utilise the current loop control mode in the ODrive. This is achieved by setting the ODrive to torque control which ensures only the current control is active. However, the ODrive calculates the reference torque based on a current reference applied to it. Thus, the MCU is set to send a current reference to the ODrive as seen in Figure 4.7. The diagram only includes a single motor as control of the motors are identical. The Clarke and Park transformations are introduced in Section 4.4.

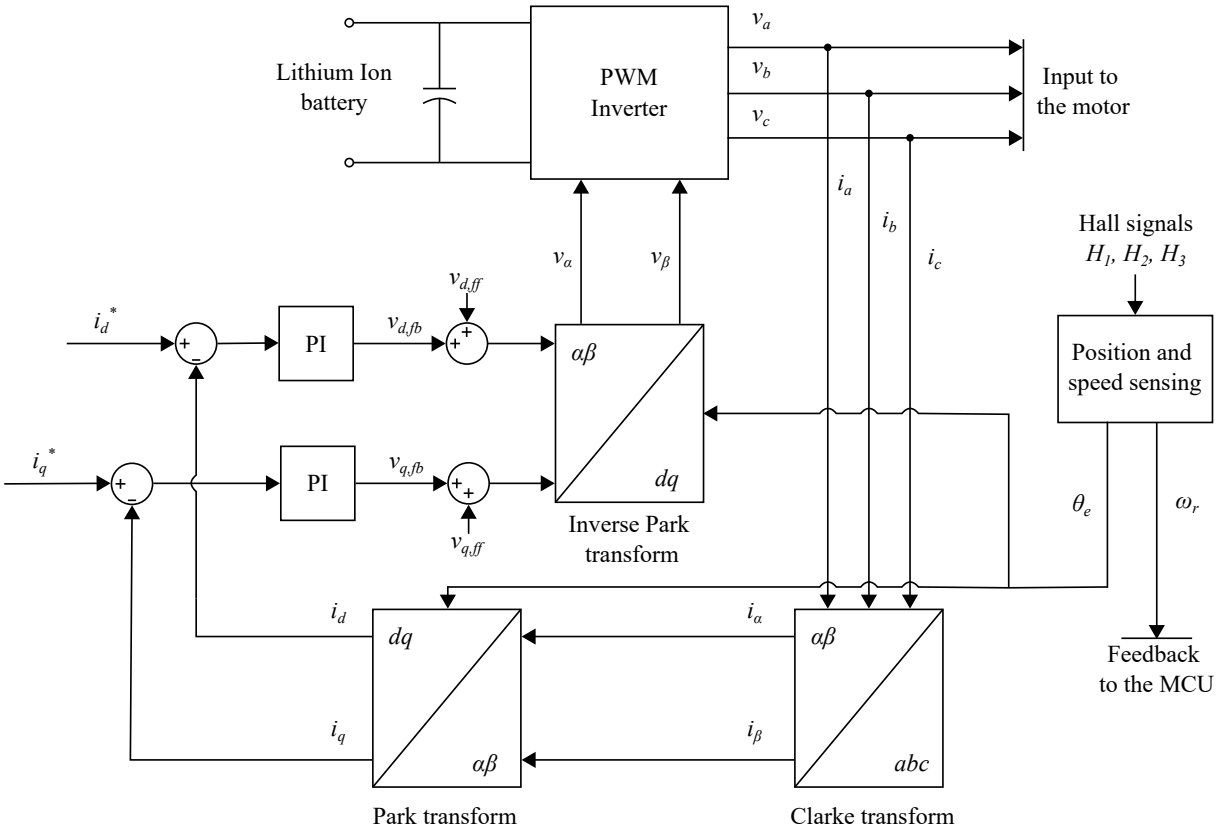


Figure 4.8: Diagram of the current loop with one motor controlled by the ODrive using field oriented control ($\bullet^*$: reference, $i_{a,b,c}$: phase currents a, b, and c, $i_{d,q}$: dq-axes current, $i_{\alpha,\beta}$: α,β-axes current, MCU: microcontroller unit, PI: proportional-integral controller, PWM: pulse width modulation, $v_{a,b,c}$: phase voltages a, b, and c, $v_{d,q}$: dq-axes voltage, $v_{\alpha,\beta}$: α,β-axes voltage, θ_e: electrical angle of the rotor, ω_r: mechanical angular velocity).

4.3.1 Investigation of the Back Emf Waveform

To investigate the back emf waveform of the BLDC motors and see if they are trapezoidal as stated in Figure 4.5, tests are conducted on motor 1. The back emf is measured by using voltages probes to measure the line-to-line voltages. The setup for the test illustrated in Figure 4.9. The conditions for the test are as follows:

- No input is applied to the system
- The motor is allowed to spin freely
- An electric hand drill that spins the motor

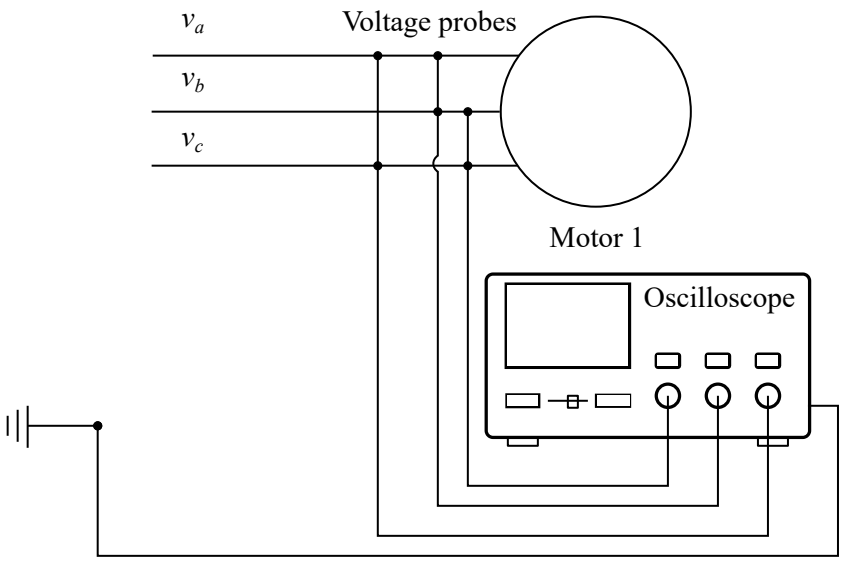


Figure 4.9: Setup for investigation of the back emf waveform ($v_{a,b,c}$: phase voltages a, b, and c).

The test is conducted by spinning the motor using an electric hand drill until the steady state speed is reached. At the steady state speed the voltage is measured on the oscilloscope. The measurement of the signals' amplitude for the test using the faster speed of the drill is shown in Figure 4.10. In Appendix H.1, the measurement of the signals' frequency using the faster speed of the drill is documented, alongside the measurements of the signals' frequency and amplitude using the slower speed of the drill. In Figure 4.10, and the figures seen in the appendix, the waveform of the back emf is

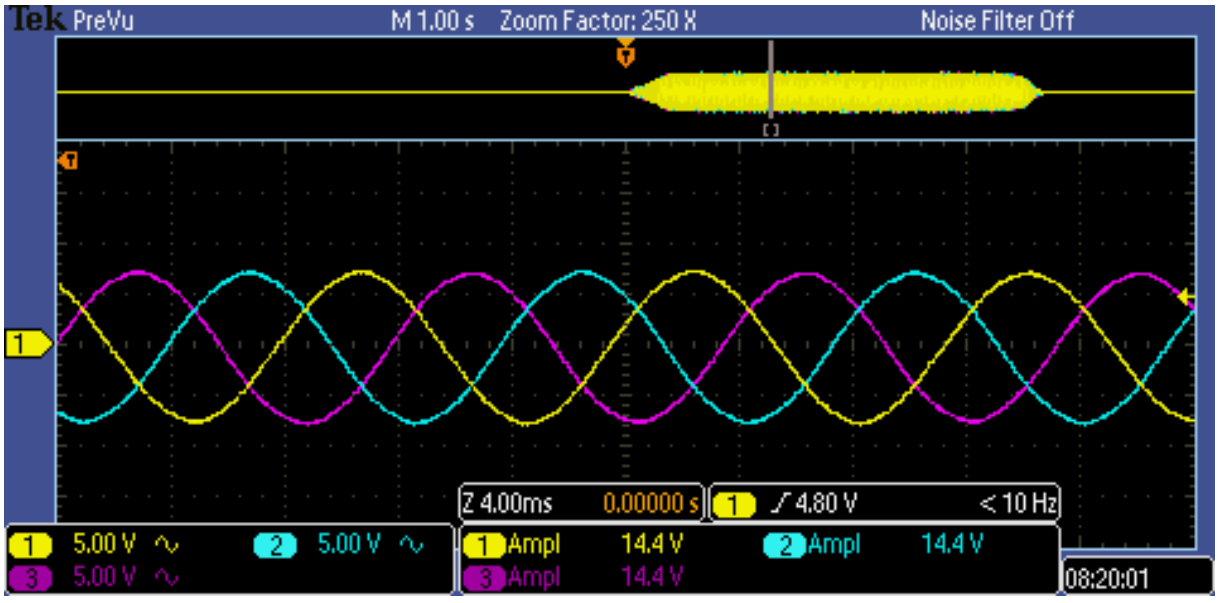


Figure 4.10: Oscilloscope measurement of the back emf waveform with the drill set to the fastest speed, where the oscilloscope measures the amplitude of the signal.

observed to be sinusoidal, which is not the case for a BLDC motor. As described in Figure 4.5, the back emf waveform of a BLDC motor is trapezoidal. However, it also implies the controllability is not affected through the use of the ODrive. The sinusoidal back emf indicates that it is a PMSM. The

BLDC motor and PMSM are often confused due to the structural similarity, and a method to tell them apart is by the shape of the back emf as performed here [48]. Therefore, a revision of the motor model is made in Section 4.4 where a PMSM model is developed to replace the BLDC motor model.

4.4 Permanent Magnet Synchronous Motor Model

In this section the working principle of the PMSM is described. Reference frame transformations are used in order to create a model of the machine and apply FOC. The FOC is analysed and the pulse width modulation (PWM) realisation is documented.

4.4.1 Working Principle

The PMSM works by inducing a magnetic field in the stator windings that align with the magnetic field of the permanent magnets. By rotating the induced magnetic field in the stator windings the permanent magnets also rotates at equal speed.

The stator winding layout of the motors are shown in Figure 4.11. The hub motors are fitted with 30 permanent magnets mounted to the inner surface of the rotor. The stator has 27 slots where each phase is wound in alternate polarity around three consecutive teeth followed by the next phase and the next. This pattern is repeated three times [50, 51, 52].

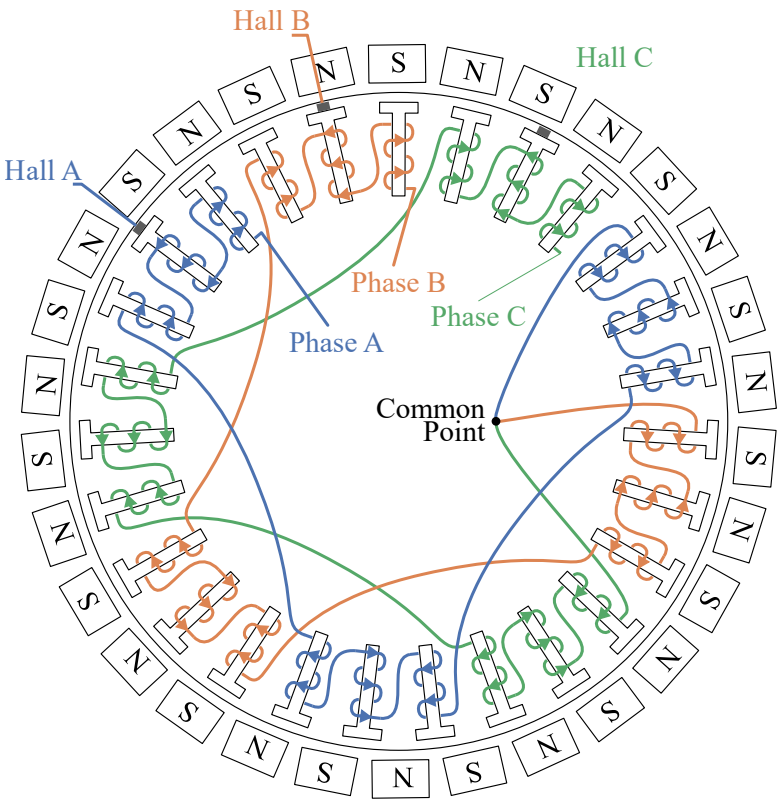


Figure 4.11: Winding layout of the PMSM.

The PMSM is modelled in the rotating dq reference frame in order to simplify the model and apply

FOC in the dq reference frame.

4.4.2 Modelling in Rotating Reference Frame

The transformation from the three-phase abc reference frame to the dq reference frame takes two steps, the first step is the Clarke transform that yields the stationary reference frame, the Park transform is then applied to that reference frame which yields the rotating dq reference frame. The reference frames and the transformations are explained in the following.

The stationary reference frame denoted with $\alpha\beta$ is the reference frame where the α-axis aligns with one of the phases, typically the a-phase. The transformation from three-phase abc reference frame to the $\alpha\beta$ reference frame is the Clarke transformation and is given in (4.41) along with its inverse which transforms from the $\alpha\beta$ reference frame to the three-phase abc reference frame. The Clarke transform shown here is the power invariant version of the Clarke transform, this is seen by the factor $\sqrt{2/3}$ multiplied the transformation matrix. The power invariance is due the fact that the power, which is the product of voltage and current, must be equal in both the three-phase abc reference frame and the two-phase $\alpha\beta$ reference frame. The non power invariant factor is $2/3$ [48, 53].

$$\begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} \qquad \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} \tag{4.41}$$

Another significant reference frame is the rotating dq reference frame. This reference frame has the d-axis aligned with the rotor flux. The transformation from the $\alpha\beta$ reference frame to the dq reference frame is the Park transform and is given as (4.42) along with its inverse.

$$\begin{bmatrix} f_a \\ f_q \end{bmatrix} = \begin{bmatrix} \cos(\theta_e) & \sin(\theta_e) \\ -\sin(\theta_e) & \cos(\theta_e) \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} \qquad \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} = \begin{bmatrix} \sin(\theta_e) & \cos(\theta_e) \\ -\cos(\theta_e) & \sin(\theta_e) \end{bmatrix} \begin{bmatrix} f_a \\ f_q \end{bmatrix} \tag{4.42}$$

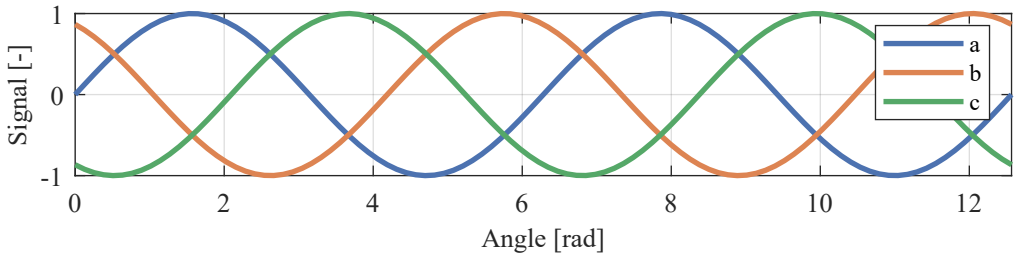
Combining both transformations it is possible to go directly from the abc reference frame to the dq reference as shown in (4.43).

$$\begin{bmatrix} d \\ q \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \sin(\theta_e) & \sin\left(\theta_e - \frac{2\pi}{3}\right) & \sin\left(\theta_e + \frac{2\pi}{3}\right) \\ \cos(\theta_e) & \cos\left(\theta_e - \frac{2\pi}{3}\right) & \cos\left(\theta_e + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

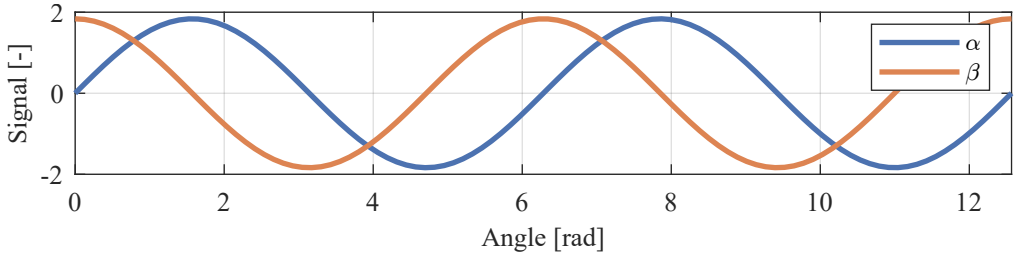
$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \sin(\theta_e) & \cos(\theta_e) \\ \sin\left(\theta_e - \frac{2\pi}{3}\right) & \cos\left(\theta_e - \frac{2\pi}{3}\right) \\ \sin\left(\theta_e + \frac{2\pi}{3}\right) & \cos\left(\theta_e + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix} \tag{4.43}$$

In reality the Clarke and Park transforms produce a third signal, the zero signal, which is always zero for balanced three phases signal, which is the case for most electrical machines and the reason it is not included here.

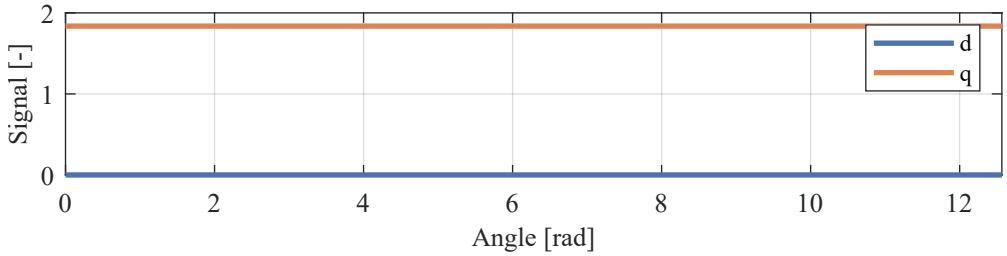
Figure 4.12 shows a three-phase signal 120° out of phase and the signals produced by the Clarke and the Park transforms. The voltage equations in the dq reference frame is given as (4.44), and the torque is given as (4.45) [48, 53].



(a) three-phase signal 120° out of phase



(b) three-phase signal in $\alpha\beta$ reference frame



(c) three-phase signal in dq reference frame

Figure 4.12: Signal transformation using Clarke and Park transform.

The ODrive applies FOC without field weakening assuming surface mounted permanent magnets with feedforward compensation of the coupling terms and the back emf.

$$\frac{d}{dt}i_d = \frac{1}{L_d}(v_d + L_q PP\omega_r i_q - R_s i_d) \tag{4.44a}$$

$$\frac{d}{dt}i_q = \frac{1}{L_q}(v_q - L_d PP\omega_r i_d - R_s i_q - PP\lambda_f \omega_r) \tag{4.44b}$$

$$T_e = \frac{3}{2}PP(\lambda_f i_q + (L_d - L_q)i_d i_q) \tag{4.45}$$

Where i_d and i_q are the d-axis and q-axis currents, v_d and v_q are the d-axis and q-axis voltages, L_d and L_q are the d-axis and q-axis inductances, and λ_f is the flux linkage of the permanents magnet.

Assuming surface mounted permanent magnets the inductances in the dq reference frame L_d and L_q are equal i.e. $L_d = L_q = L_s$, because the permanent magnets are spread evenly around the surface of the rotor. Thus, simplifying the voltage equations to (4.46) and the torque to (4.47), it is seen that only the

q-axis current is the torque producing current.

$$\frac{d}{dt}i_d = \frac{1}{L_s}(v_d + L_s PP\omega_r i_q - R_s i_d) \tag{4.46a}$$

$$\frac{d}{dt}i_q = \frac{1}{L_s}(v_q - L_s PP\omega_r i_d - R_s i_q - PP\lambda_f \omega_r) \tag{4.46b}$$

$$T_e = \frac{3}{2}PP(\lambda_f i_q) \tag{4.47}$$

The q-axis current is the torque producing current. The d-axis current is the flux-producing current and is set to zero as the FOC is without field weakening.

In order to simplify the control of the PMSM the coupling terms and the back emf terms are compensated for using feedforward. By measuring the motor speed and the phase current the terms can be cancelled, as shown in the block diagram in Figure 4.13. The feedforward terms are given in (4.48).

$$v_{d,ff} = LPP\omega_r i_q \tag{4.48a}$$

$$v_{q,ff} = \omega_r PP(L_s i_d + \lambda_f) \tag{4.48b}$$

This results in a first order decoupled current dynamics shown in (4.49) assuming perfect compensation, which allows for linear control of the current which is also the case for current control of the BLDC motor.

$$\frac{d}{dt}i_d = \frac{1}{L_s}(v_d - R_s i_d) \tag{4.49a}$$

$$\frac{d}{dt}i_q = \frac{1}{L_s}(v_q - R_s i_q) \tag{4.49b}$$

To design a controller of the torque producing current, a transfer function for the q-axis current is derived using Laplace transform, as given in (4.50).

$$\frac{I_q(s)}{V_q(s)} = \frac{1}{L_s\, s + R_s} \tag{4.50}$$

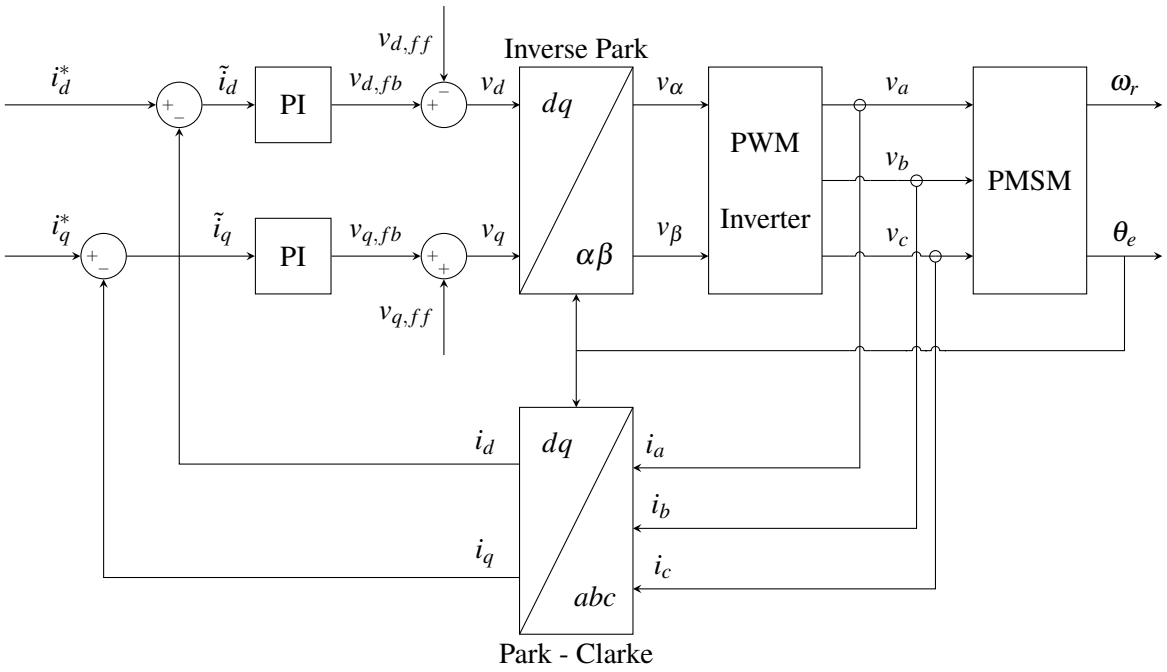


Figure 4.13: Block diagram of field oriented control and feedforward compensation of PMSM ($\bullet^*$: reference, $\tilde{\bullet}$: error between reference and feedback, $i_{a,b,c}$: phase currents a, b, and c, $i_{d,q}$: dq-axes current, $i_{\alpha,\beta}$: α,β-axes current, PI: proportional-integral controller, PWM: pulse width modulation, $v_{a,b,c}$: phase currents a, b, and c, $v_{d,q}$: dq-axes voltage, $v_{\alpha,\beta}$: α,β-axes voltage, θ_e: electrical angle, ω_r: mechanical angular velocity).

4.4.3 Space Vector Pulse Width Modulation

In order to convert the DC voltage source from the battery to three-phase voltages a six-step inverter is used. In Figure 4.14, a circuit of the three-phase inverter is presented. The ODrive uses symmetric space vector modulation to control the switching of the MOSFETs [54].

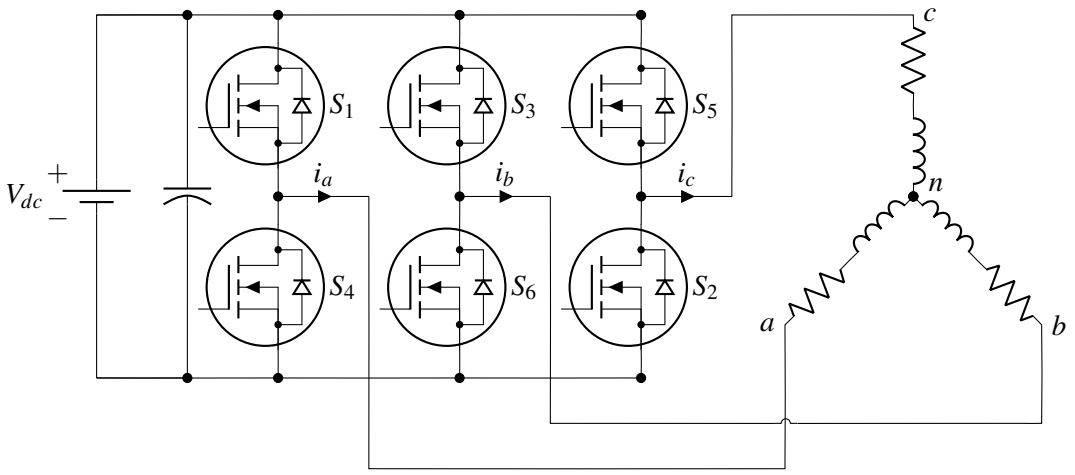


Figure 4.14: Circuit of three-phase inverter (a: phase a, b: phase b, c: phase c, $i_{a,b,c}$: phase currents a, b, and c, n: neutral point, V_{dc}: direct current voltage, S_i: MOSFET where $i = \{1,..,6\}$).

By analysing the phase voltages in the eight switch states a complex space vector can be made. Table 4.3 shows the phase voltages generated by the different switch states of the inverter. The last

column shows the complex space vector resulting from these phase voltages, as given as in (4.51).

$$V = \frac{2}{3}(v_{as} + a v_{bs} + a^2 v_{cs}) \qquad \text{Where:} \quad a = e^{j2\pi/3},\ a^2 = e^{j4\pi/3} \tag{4.51}$$

Figure 4.15 shows the space vector with its real axis aligned with the α-axis in the $\alpha\beta$ reference frame.

Table 4.3: Phase voltage and space voltage vector as a function of switching.

Switch States			Phase Voltages			Space Voltage Vector
S_1	S_3	S_5	v_{an}	v_{bn}	v_{cn}	$V_n, n = \{0,...,7\}$
0	0	0	0	0	0	$V_0 = 0\ \angle 0°$
1	0	0	$\frac{2}{3}V_{dc}$	$-\frac{1}{3}V_{dc}$	$-\frac{1}{3}V_{dc}$	$V_1 = \frac{2}{3}V_{dc}\ \angle 0°$
1	1	0	$\frac{1}{3}V_{dc}$	$\frac{1}{3}V_{dc}$	$-\frac{2}{3}V_{dc}$	$V_2 = \frac{2}{3}V_{dc}\ \angle 60°$
0	1	0	$-\frac{1}{3}V_{dc}$	$\frac{2}{3}V_{dc}$	$-\frac{1}{3}V_{dc}$	$V_3 = \frac{2}{3}V_{dc}\ \angle 120°$
0	1	1	$-\frac{2}{3}V_{dc}$	$\frac{1}{3}V_{dc}$	$\frac{1}{3}V_{dc}$	$V_4 = \frac{2}{3}V_{dc}\ \angle 180°$
0	0	1	$-\frac{1}{3}V_{dc}$	$-\frac{1}{3}V_{dc}$	$\frac{2}{3}V_{dc}$	$V_5 = \frac{2}{3}V_{dc}\ \angle 240°$
1	0	1	$\frac{1}{3}V_{dc}$	$-\frac{2}{3}V_{dc}$	$\frac{1}{3}V_{dc}$	$V_6 = \frac{2}{3}V_{dc}\ \angle 300°$
1	1	1	0	0	0	$V_7 = 0\ \angle 0°$

The reference vector V^* is given by the two inner control loops controlling the d and q-axis currents transformed to the $\alpha\beta$ reference frame.

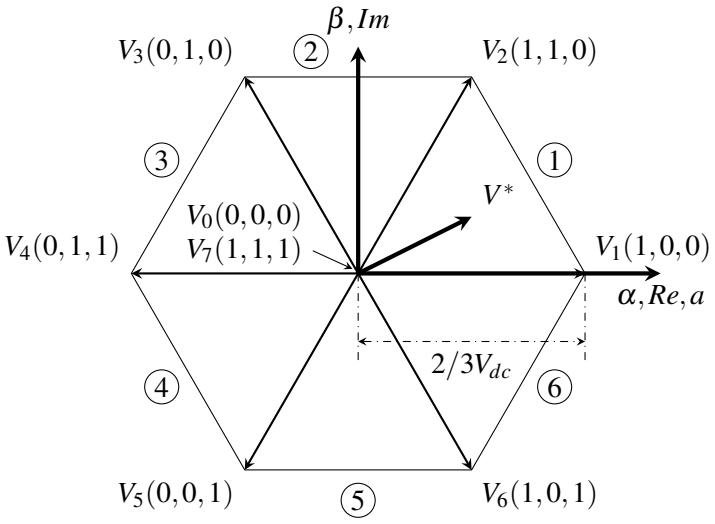


Figure 4.15: Output voltage vector in the complex plane (V_{dc}: DC voltage, V_n: Space voltage vector $n = \{0,2,...,7\}$).

The active switch states are determined by the reference vector as the two adjacent switch states. In the case of the figure where the reference vector is in sector $m = 1$, it is switch state V_1 and V_2 as well as the zero voltage vectors V_0 and V_7 that are active. In order to have the same fundamental volt-second average as the reference vector throughout the modulation period T_s, each switch state is active a certain time. By assuming a constant DC voltage the following expression must be true:

$$V^* T_s = V_n T_1 + V_{n+1} T_2 \tag{4.52}$$

By splitting this expression into its α and β components yields the following:

$$T_s|V^*|\cos(\phi) = \frac{2}{3}T_1V_{dc}\cos\left(m\frac{\pi}{3}\right) + \frac{2}{3}T_2V_{dc}\cos\left((m+1)\frac{\pi}{3}\right) \tag{4.53a}$$

$$T_s|V^*|\sin(\phi) = \frac{2}{3}T_1V_{dc}\sin\left(m\frac{\pi}{3}\right) + \frac{2}{3}T_2V_{dc}\sin\left((m+1)\frac{\pi}{3}\right) \tag{4.53b}$$

where T_1 and T_2 are the time that the switch states are active, and m is the sector. Solving (4.53) for T_1 and T_2 yields:

$$T_1 = \frac{2}{\sqrt{3}}T_sMI\sin\left(\frac{\pi}{3}m - \phi\right) \tag{4.54a}$$

$$T_2 = \frac{2}{\sqrt{3}}T_sMI\cos\left(\frac{\pi}{3}m - \phi + \frac{\pi}{6}\right) \tag{4.54b}$$

where MI is the modulation index, given as (4.55). The ODrive has a maximum modulation index of around 70% [55].

$$MI = \frac{|V^*|}{\frac{2}{3}V_{dc}} \tag{4.55}$$

If $T_1 + T_2 < T_s$ then the zero voltage vectors are applied in the remaining time:

$$T_0 = T_s - (T_1 + T_2) \tag{4.56}$$

To exemplify the symmetrical modulation technique, the symmetrical switching of sector 1 is illustrated in Figure 4.16.

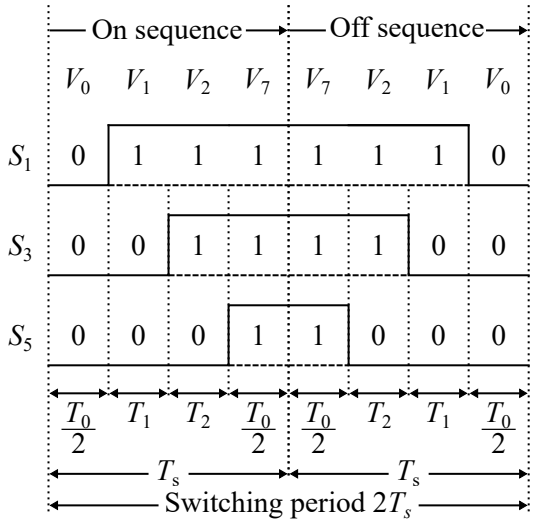


Figure 4.16: Illustration of the symmetric space vector pulse width modulation for sector 1 in Figure 4.15 ($S_{1,3,5}$: switch states, T_s: modulation period, T_0: zero voltage time, $T_{1,2}$: time effective vectors are active, $V_{0,7}$: zero vectors, $V_{1,2}$: effective vectors).

This means that in sector 1, $V_1(1,0,0)$ is on for time T_1 and $V_2(1,1,0)$ is on for time T_2 and either $V_0(0,0,0)$ or $V_7(1,1,1)$ is on for T_0. The sequence in which the voltage vectors are on is symmetric to reduce the high order harmonics and increase the voltage modulation range. The symmetrical SVPWM

technique splits the zero voltage time T_0 into two parts, and applies the active states in the centre of the modulation period. This means that in sector 1, if $S_1 = S_2 = S_3 = 0$ (i.e. all upper switches are off), the sequence is all follows: $V_0(0,0,0)$ followed by $T_0/2 \rightarrow V_1(1,0,0)$ followed by $T_1 \rightarrow V_2(1,1,0)$ followed by $T_2 \rightarrow V_7(1,1,1)$ followed by $T_0/2$. If all switches are on in the beginning of the modulation period, the sequence is reversed. This can be done for all sectors as the zero voltage vector is used in all sectors [48, 53, 56].

4.5 Summary

Firstly, a mechanical model of the robot is developed using the Euler-Lagrange formulation and the changing inertia of the paint tank effect on the system is addressed. Then, an electric model of the BLDC motors is made. This is followed by an overview of the system displayed in a block diagram. The back emf of the motors are investigated to validate the motor model. The investigation shows the back emf wave as sinusoidal, which is the case for PMSMs and not BLDC motors. Therefore, it is chosen to make a PMSM model to replace the BLDC motor model.

Chapter 5

Linear Control Schemes

This chapter describes the approach and reasoning for the chosen controllers. It reveals how the controllers are designed based on the linearised models of the physical system.

In order to control the system, cascade control is implemented with the inner loop controlling the current using the ODrive, and the outer loop controlling the states of the MWIP through the use of the MCU and the IMU, as seen in Figures 4.7 and 4.8. A PI controller is employed to control the current of each motor, while MIMO control is utilised to control the torque of the motors.

5.1 Control Design for the Inner loop

As the ODrive provide the current control, the model developed in the section is used to describe how the ODrive realises the control. However, ODrive does not provide documentation of the method used to realise their current controller. Therefore, it is assumed that the ODrive uses pole-zero cancellation to determine the values of the proportional and integral gains in the PI controller. For the derivation of the pole-zero cancellation method, [48] is used as a reference. As the q-axis current is the current producing torque to make the PI controller for this loop, as the d-axis is set always to be zero. The calculation of the gains, the plant for the q-axis current, G_p (derived in Section 4.4) and the current controller, G_c stated in (5.1) and (5.2) are used.

$$G_p(s) = \frac{1}{L_s\, s + R_s} \tag{5.1}$$

$$G_c(s) = K_p + \frac{K_i}{s} = K_p\left(1 + \frac{K_i}{K_p\, s}\right) = K_p\left(\frac{s + \frac{K_i}{K_p}}{s}\right) \tag{5.2}$$

K_p and K_i are the proportional and integral gain. Using these, the open loop transfer function for the q-axis current, $G_{ol.c}$ is determined as (5.3).

$$G_{ol.c}(s) = G_p(s)G_c(s) = \frac{1}{L_s\, s + R_s}\, K_p\left(\frac{s + \frac{K_i}{K_p}}{s}\right) = \frac{\frac{1}{L_s}}{s + \frac{R_s}{L_s}} K_p\left(\frac{s + \frac{K_i}{K_p}}{s}\right) \tag{5.3}$$

To cancel the pole $-R_s/L_s$ for the system, the term K_i/K_p needs to be equal to it:

$$\frac{K_i}{K_p} = \frac{R_s}{L_s} \tag{5.4}$$

Using the relation given in (5.3), the open current loop transfer function becomes:

$$G_{ol.c}(s) = \frac{\frac{1}{L_s} K_p (s + \frac{R_s}{L_s})}{(s + \frac{R_s}{L_s})s} = \frac{1}{\frac{L_s}{K_p} s} \tag{5.5}$$

By taking the magnitude of the open loop transfer function and inserting $j\omega_c$, it is observed that the crossover frequency, ω_c for the system is determined by K_p/L_s, as stated in (5.6).

$$|G_{ol.c}(s)| = \frac{1}{|\frac{L_a}{K_p}s|} = 1 \Rightarrow \omega_c = \frac{K_p}{L_s} \tag{5.6}$$

If the required bandwidth is set to ω_c, K_p is obtained from (5.6):

$$K_p = L_s\omega_c \tag{5.7}$$

K_i is determined by inserting (5.7) in (5.4).

$$K_i = K_p\,\frac{R_s}{L_s} = R_s\omega_c \tag{5.8}$$

It is desired to transform this first order system to have a bandwidth of 100 rad/s based on the recommendation from [57]. Based on the values stated in Table 4.2, the resulting values for K_p and K_i are 0.04 and 23 respectively.

A block diagram of the closed-loop system of the q-axis current with the PI controller is seen in Figure 5.1.

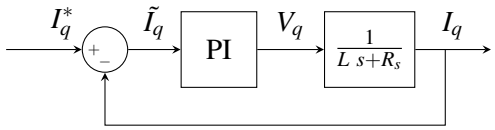


Figure 5.1: Closed-loop block diagram of q-axis current loop for the PMSM ($\bullet^*$: reference, $\tilde{\bullet}$: error between reference and feedback, I_q: q-axis current, L_s: inductance, R_s: stator resistance, s: Laplace operator, PI: proportional-integral controller).

As a controller for the inner loop of the system has been designed, the design of a controller for the outer loop of the system is initiated.

5.2 Control Design for the Outer loop

In this section an LQR is proposed as the controller of the outer loop.

5.2.1 Linear Model of Mechanical System

For the purpose of implementing a linear controller for the multiple-input-multiple-output (MIMO) system, the dynamics are linearised and a state space model is created. The states are given in (5.9).

$$\mathbf{x} = \begin{bmatrix} \alpha & \dot{\alpha} & \beta & \dot{\beta} & \dot{s} \end{bmatrix} \tag{5.9}$$

The distance travelled by the frame s is not included as a state because the other states are independent of it.

To decouple the control and simplify the analysis, the input to the system is changed to represent the difference and sum torque:

$$T_{diff} = T_1 - T_2 \tag{5.10}$$

$$T_{sum} = T_1 + T_2 \tag{5.11}$$

This changes the non-conservative externally applied torque given in (4.16) on page 24 to (5.12).

$$\mathbf{Q} = \begin{bmatrix} \frac{T_{diff}}{r} L_{ACMW} + \frac{2B_v L_{ACMW}}{r}\dot{\alpha} \\ 2B_v\dot{\beta} \\ \frac{T_{sum}}{r} + \frac{2B_v}{r}\dot{s} \end{bmatrix} \tag{5.12}$$

The dynamics of the system is not changed by representing the torques like this, only the analysis of the system changes.

The state space model is derived by linearising the differential equations given by (4.22), at zero, i.e. linearising with the frame at the upright position and a full paint of tank, meaning $L = h$. This gives the following state space model assuming full state feedback.

$$\begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \\ \dot{\beta} \\ \ddot{\beta} \\ \ddot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.0085 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 21.5 & 0.0032 & -0.0071 \\ 0 & 0 & -3.96 & -0.0006 & 0.0046 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\alpha} \\ \beta \\ \dot{\beta} \\ \dot{s} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1.218 & 0 \\ 0 & 0 \\ 0 & -1.019 \\ 0 & 0.654 \end{bmatrix} \begin{bmatrix} T_{diff} \\ T_{sum} \end{bmatrix} \tag{5.13a}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\alpha} \\ \beta \\ \dot{\beta} \\ \dot{s} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_{diff} \\ T_{sum} \end{bmatrix} \tag{5.13b}$$

5.2.2 Linear Quadratic Regulator

The LQR controller provides a systematic method to determine the feedback gains based on comprehensible parameters. The LQR controller is determined by subjective iterative estimation of the importance of the state error and expenditure of energy.

The block diagram depicting the control of the robot is given in Figure 5.2.

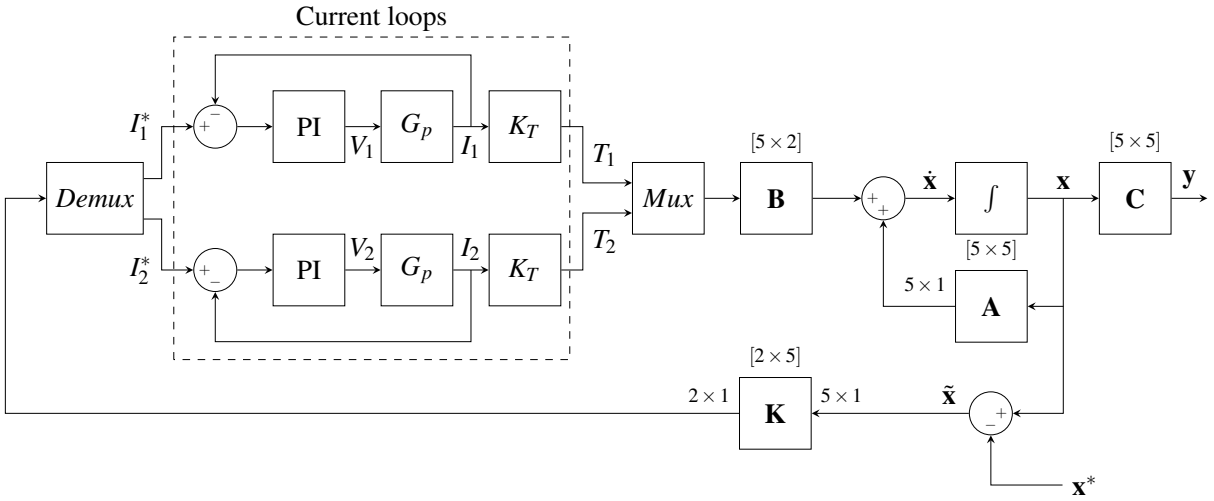


Figure 5.2: The linear closed-loop block diagram ($\bullet^*$: reference, $\tilde{\bullet}$: error, **A**: system matrix, **B**: input matrix, **C**: output matrix, G_p: plant for the q-axis current, I_i: current for motor $i = \{1,2\}$, T_i: torque applied by motor $i = \{1,2\}$, **K**: feedback gain matrix, K_T: torque constant, PI: proportional-integral controller, V_i: voltage for motor $i = \{1,2\}$).

The control law is given as (5.14).

$$\mathbf{u} = -\mathbf{K}\tilde{\mathbf{x}} \tag{5.14}$$

$\tilde{x}$ is the error between the reference and feedback. The feedback gain matrix $\mathbf{K}$ is derived by minimising the performance index given in (5.15).

$$J = \int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt \tag{5.15}$$

Where $\mathbf{Q}$ and $\mathbf{R}$ are positive definite. $\mathbf{Q}$ determines the relative importance of the error, and $\mathbf{R}$ determines the relative importance of the energy expenditure. The theory behind LQR and the method to derive the optimised feedback gain matrix are presented in Appendix G.

The $\mathbf{Q}$ and $\mathbf{R}$ matrices are determined based on the following guidelines:

$$\mathbf{Q}(i,i) = \frac{1}{\mathbf{e}_{i,\max}^2} \tag{5.16}$$

$$\mathbf{R}(i,i) = \frac{1}{\mathbf{u}_{i,\max}^2} \tag{5.17}$$

Where $i = \{1,\ldots,5\}$, $\mathbf{e}_{i,\max}$ represents the maximum allowed deviation of the state, and $\mathbf{u}_{i,\max}$ represents the maximum allowed control signal of each input.

It is wanted for the robot to have less deviation from desired states of α, $\dot{\alpha}$, and $\dot{s}$ compared to β and $\dot{\beta}$. Noted that the robot should be able to be self-balancing, but the referenced tilt angle is less important than maintaining overall stability of the system. Using the parameters given below the optimised feedback gain matrix $\mathbf{K}$ is determined as (5.18).

$$\mathbf{e}_{\max} = \begin{bmatrix} 0.50 & 1 & 10 & 10 & 2 \end{bmatrix}$$

$$\mathbf{u}_{\max} = \begin{bmatrix} 10 & 10 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} 20.0 & 11.5 & 0 & 0 & 0 \\ 0 & 0 & -64.86 & -15.0 & -5.00 \end{bmatrix} \tag{5.18}$$

5.3 Summary

Two linear controllers are proposed for controlling the system. The current loop is designed with a PI controller for each motor using pole-zero cancellation to determine the controller gains. An LQR controller is designed for the outer loop and the control gains are determined based on the maximum allowed deviation of the state and the maximum allowed control signal of each input.

Chapter 6

Experiments

In this chapter, the experiments performed on the system are documented.

6.1 Test of Hall Sensor Outputs

To verify the functionality of the hall effect sensors, the output voltage of the hall effect sensors is measured using three voltage probes. The experimental setup is depicted in Figure 6.1. Afterwards it is compared with the expected waveforms depicted in Figure 6.2. The conditions for the experiment are listed in the following:

- Only the current loop is used in the test along with 'motor 1'.
- The motor is allowed to spin freely.
- A reference of 1 A is given for the q-axis current while the d-axis current reference is to zero.

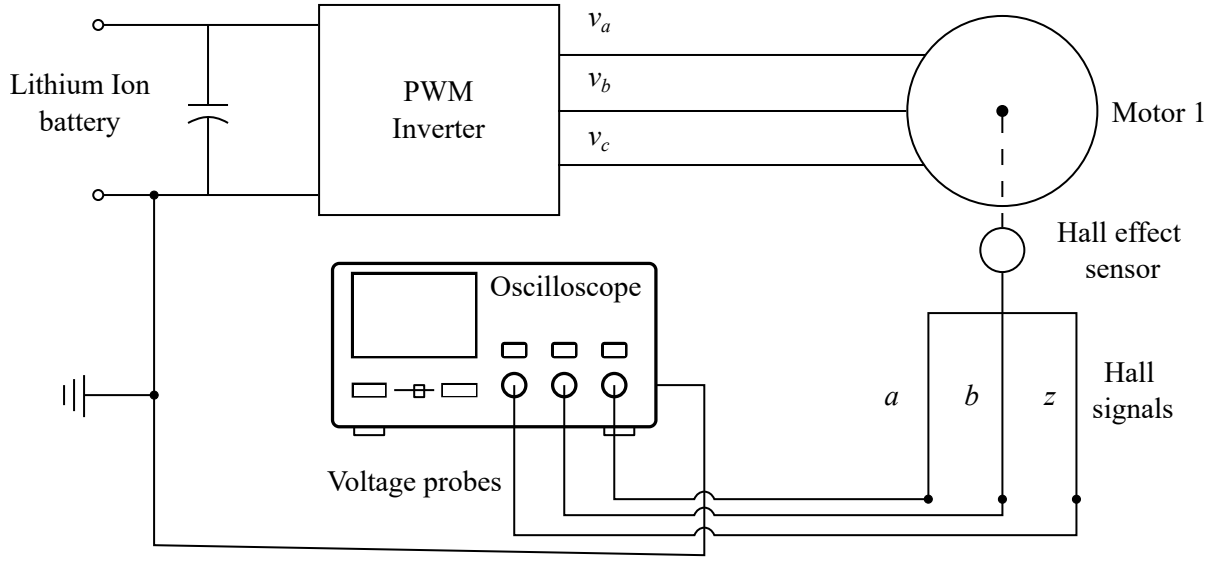


Figure 6.1: Setup for the test of the Hall sensors (PWM: pulse width modulation, $v_{a,b,c}$: phase voltages a, b, and c).

The produced output voltage waveforms from the measurements are depicted in Figure 6.3. The timescale on the oscilloscope is adjusted to increments of 2 ms and the produced waveforms are extracted from the oscilloscope.

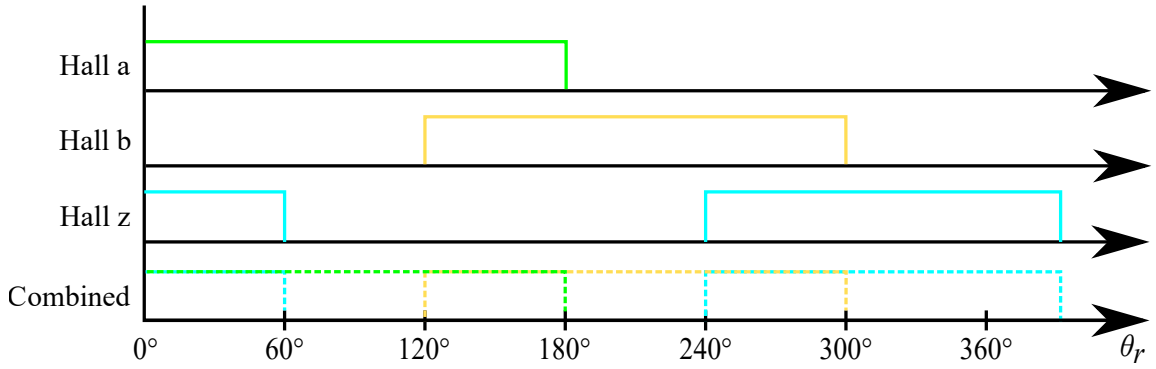


Figure 6.2: Expected Hall sensor output voltage waveforms (Hall a, b, z: Hall sensor output voltage waveform, Combined: Hall a, b, and c on one axis, θ_r: rotor position).

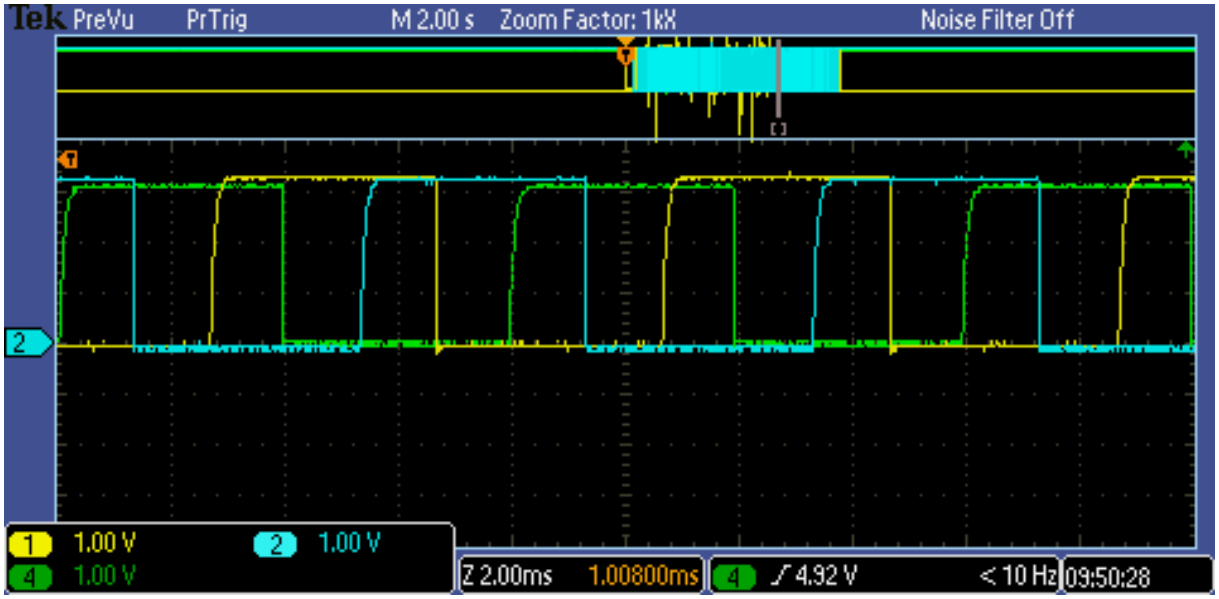


Figure 6.3: Oscilloscope measurements depicting the output voltage of the three Hall sensors in three colours, with each grey horizontal line representing 1 V and each grey vertical line representing 2 ms (green waveform: Hall signal a, yellow waveform: Hall signal b, turquoise waveform: Hall signal z).

As can be derived from the comparison of Figures 6.2 and 6.3, the Hall effect sensors produce 3.3 V logic output voltage waveforms which overlap the preceding hall effect sensor with the first third of its conduction interval, during the middle third of its conduction interval the other hall effect sensors are turned off, and the last third of the conduction interval overlaps the subsequent Hall effect sensor. Thus, it is concluded that the Hall effect sensors trigger at the expected instances from Figure 6.2, but as is seen in Figure 6.3, the output voltage level of the three hall effect sensors differs in magnitude.

6.2 Validation of the q-axis Current Loop

To validate the linear model of the q-axis current loop with a PI controller, a test is conducted using only one of the motors since their controls are identical. It is validated using the closed-loop system as it is not possible to test the current response in open loop.

6.2.1 Analysis of the Modelled Closed q-axis Current Loop

For comparison, the response of the modelled q-axis current loop is analysed, and a pole-zero plot of the closed-loop system is made in Figure 6.4. From the figure, it is seen that there is a dominant pole

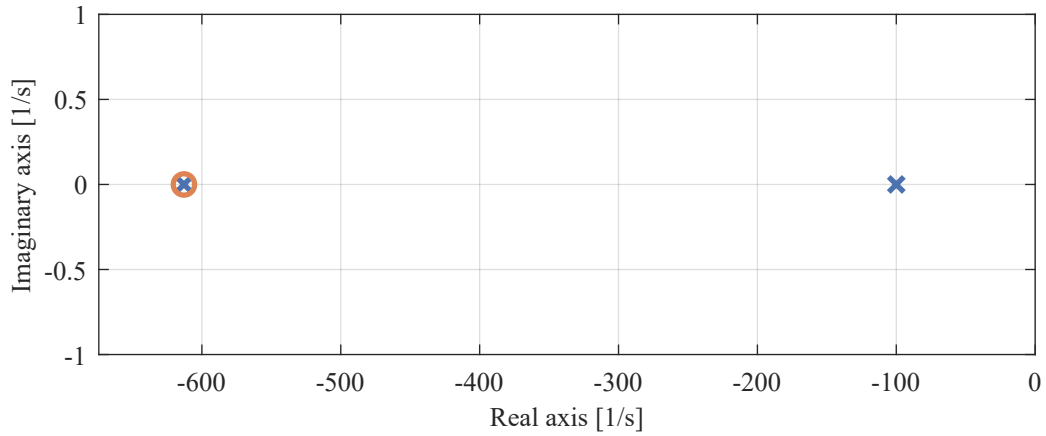


Figure 6.4: Pole-zero plot of closed current loop (×: pole, ○: zero).

placed at $-100\,\mathrm{rad/s}$. As a rule of thumb, if the ratio of the poles placed farther on the left divided with the dominant poles is 5-10 or greater, the poles placed farther away may be neglected [58]. Thus, the dominant pole dictates the system's transient response. Using the dominant pole, the time constant τ of the system is determined as [58]:

$$\tau = \frac{1}{\omega_r} = \frac{1}{100\,\mathrm{s}^{-1}} = 10\,\mathrm{ms} \tag{6.1}$$

Furthermore, as the dominant pole has a zero valued imaginary part, the system's transient response has the characteristics of a first order system as shown in Figure 6.5.

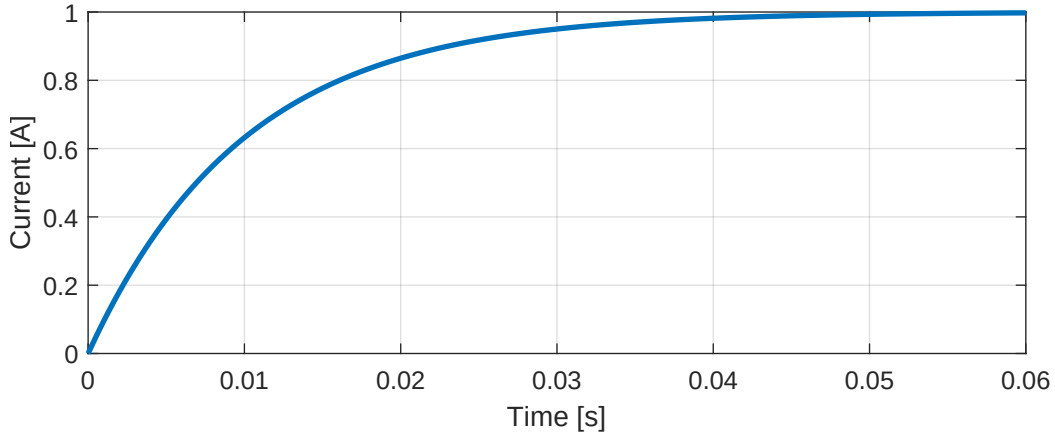


Figure 6.5: Step response of the closed current loop.

6.2.2 Test of the Closed q-axis Current Loop

The test is conducted on 'Motor 1' under the following conditions:

- The motor is blocked making it unable to rotate.

- A reference of 2 A is given for the q-axis current while the d-axis current reference is zero.

The experimental setup for the test is illustrated in Figure 6.6. To measure the current output of the phases, an oscilloscope is connected to the phase wire using a current probe.

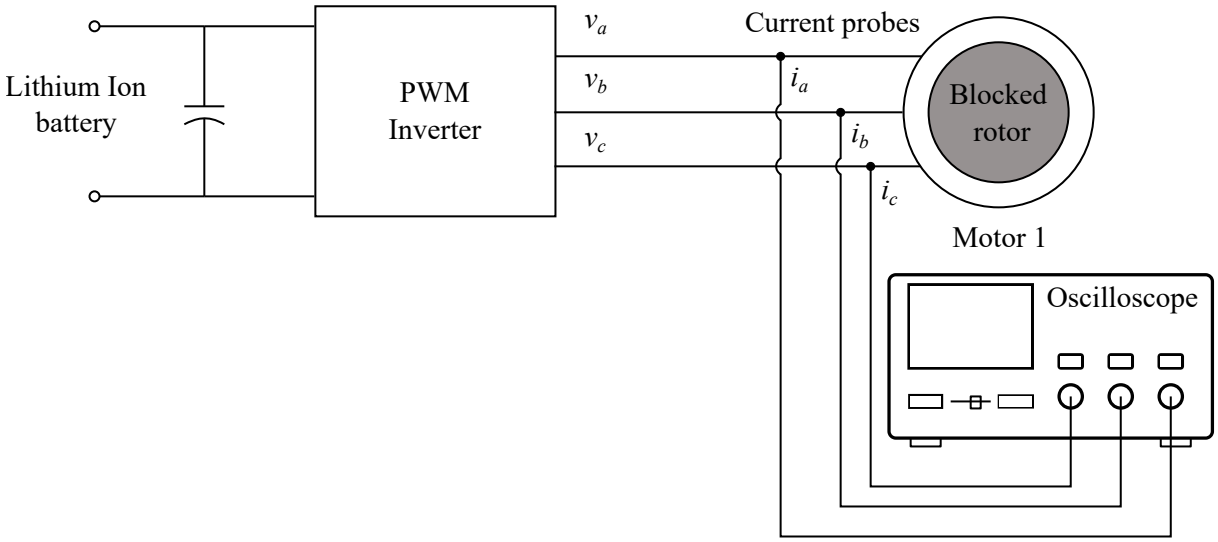
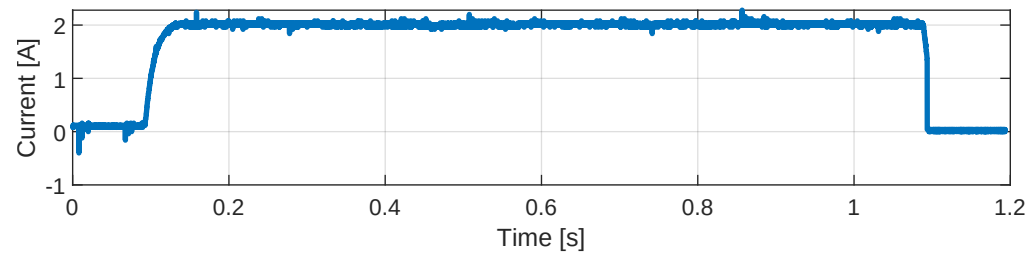


Figure 6.6: Setup for test of the closed q-axis current loop ($i_{a,b,c}$: phase currents a, b, and c, PWM: pulse width modulation, $v_{a,b,c}$: phase voltages a, b, and c).

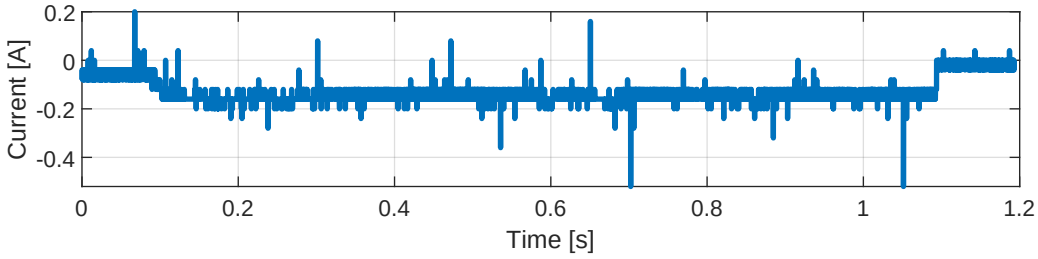
In a blocked rotor test the coupling terms and the back emf in (4.46) on page 37 are zero, because they are proportional to the motor speed, which in turn also suppresses the feedforward control. The current response is DC because the electrical angle θ_e is constant throughout the test. By analysing (4.42) on page 35 it is seen that if θ_e is zero, the transformation matrix becomes the identity matrix thus eliminating AC currents. Therefore, the expected phase current response is a first order response on all three phases, where one of the phases converges to 2 A while the other two phases converges to -1 A as Kirchhoff's current law dictates that the sum of the current phases must equal zero and assuming that the the phase resistance is equal on all phases. This is also seen by evaluating the inverse Clarke-Park transform given by (4.43) on page 35 in steady state, represented in the power variant version below:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sin(\theta_e) & \cos(\theta_e) \\ \sin\left(\theta_e - \frac{2\pi}{3}\right) & \cos\left(\theta_e - \frac{2\pi}{3}\right) \\ \sin\left(\theta_e + \frac{2\pi}{3}\right) & \cos\left(\theta_e + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} \sin(0) & \cos(0) \\ \sin\left(0 - \frac{2\pi}{3}\right) & \cos\left(0 - \frac{2\pi}{3}\right) \\ \sin\left(0 + \frac{2\pi}{3}\right) & \cos\left(0 + \frac{2\pi}{3}\right) \end{bmatrix} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}$$

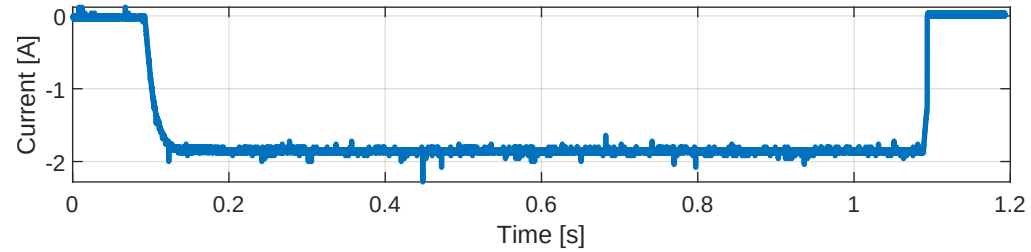
The test results of the three phases' current response is shown in Figure 6.7. The phase currents are fluctuating throughout the tests. Contrary to expectations, the results show that the currents are not evenly distributed between phase b and phase c. The current flowing through phase c is significantly larger than the one flowing through phase b. Furthermore, before the input is given there is an offset in the current value. This is evident from the figures, which zoom in on the the beginning of the test, as shown in Appendix H.2 where, for example, the current of phase a has an offset on the y-axis of 80 mA.



(a) Phase *a* current.



(b) Phase *b* current.



(c) Phase *c* current.

Figure 6.7: Phase current response in blocked rotor test.

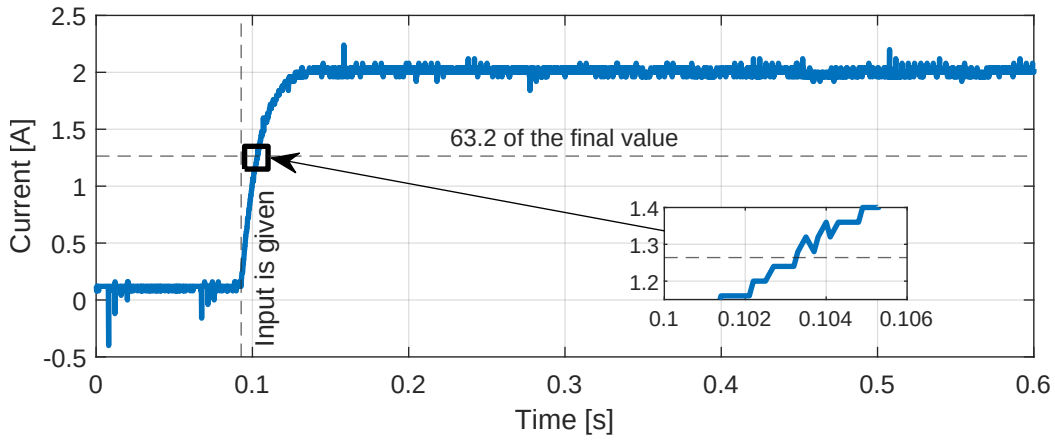


Figure 6.8: Current loop validation using the step response of phase *a* to determine the time constant.

The system is validated by analysing the time constant measured using the step response. The time constant is defined as $1 - 1/e \approx 0.632$. This means that the time constant can be measured as the time it takes the current to reach 63.2 % of its final value. In Figure 6.8, step response of the phase *a* is used to determine the time constant. The input time is 0.0928 s, and the time at 63.2 % is measured to be 0.103 s. By subtracting the first value with the second one, the time constant is determined to be 10.2 ms. In the figure, the transient response of the system is displaying characteristics of a first order system as expected.

Table 6.1: Comparison of the model with the physical system for the closed current loop.

System	Time constant
Physical system	10.2 ms
Modelled system	10.0 ms
Difference	1.96 %

The measured results compared with the modelled system are stated in Table 6.1. It results in a difference of 1.96 % for the time constant when comparing the model with the physical system.

6.3 Validation of Motor Model in Open Loop

In this section the motor model's transient behaviour is validated in order to ensure that the model accurately represents the physical system, which is critical for the effective design, analysis, and implementation of the control systems. Firstly, a model without the SVPWM is compared to the model with SVPWM to simplify the model for further analysis, the Simulink models are given in Appendix E.2. Then, the maximum modulation index is implemented in the simplified model. Lastly, an optimisation algorithm is used to find the unknown motor parameters and the friction coefficient.

Figure 6.9 shows the simulated motor current under the same conditions as Section 6.2. The blue curve shows the current using SVPWM and switch conditions as described in Section 4.4.3, while the orange curve shows the current simulated without SVPWM. As seen in the figure the current response is similar for both simulations, but the SVPWM simulation shows some noise in the response, which occurs because of the switching. Based on this, it is concluded that the simplified model is

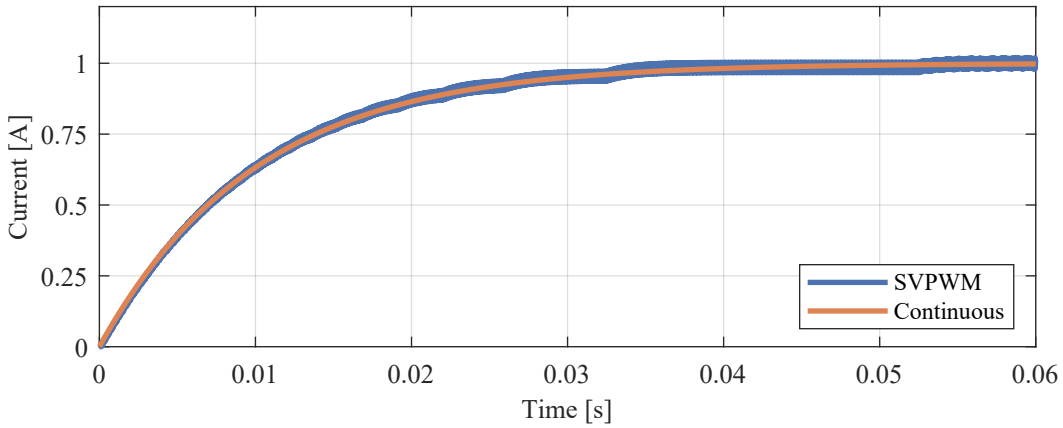


Figure 6.9: Comparing between SVPWM model and simplified model.

representative of the SVPWM model, and it is used in the following validation. The block diagram of the simplified model is shown in Figure 6.10.

The friction characteristics are unknown for the motors, so it is assumed that the friction is dominantly viscous. An initial guess is made using a steady state analysis of the speed response with a current of 1 A as seen in Figure 6.11. In steady state the dynamics of the motor is simplified using:

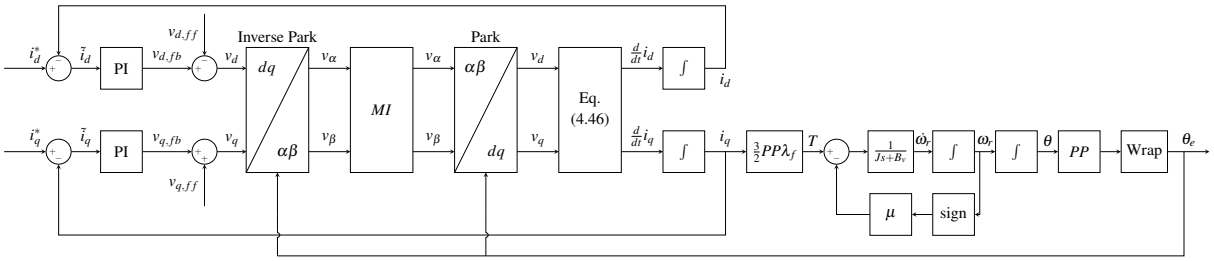


Figure 6.10: Block diagram of simplified motor model ($\bullet^*$: reference, $\tilde{\bullet}$: error between reference and feedback, B_v: viscous friction, $i_{a,b,c}$: phase currents a, b, and c, $i_{d,q}$: dq-axes current, $i_{\alpha,\beta}$: α,β-axes current, J: inertia of the motor, MI: modulation index, PI: proportional-integral controller, PP: pole pair, s: Laplace operator, $v_{a,b,c}$: phase currents a, b, and c, $v_{d,q}$: dq-axes voltage, $v_{\alpha,\beta}$: α,β-axes voltage, θ_e: electrical angle, μ: friction coefficient, ω_r: mechanical angular velocity).

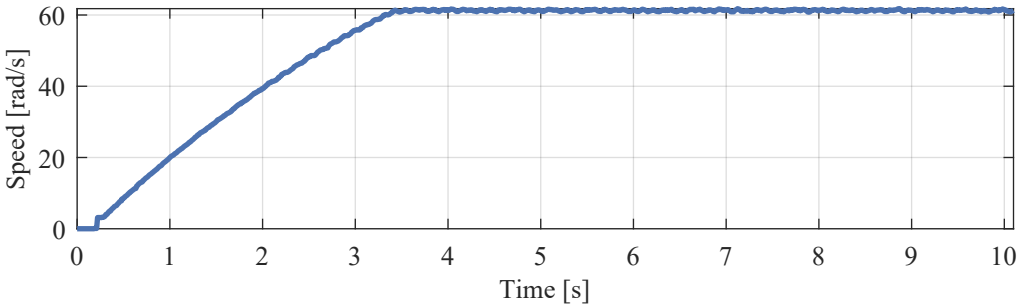


Figure 6.11: Open loop motor velocity response with a current of 1 A applied at $t = 0.1$.

$$\dot{\omega}_r = \frac{1}{J}(T_e - B_v\omega_r) = 0$$

This means that either the sum within the parentheses is zero or the inertia is infinite which is trivial:

$$T_e - B_v\omega_r = 0 \quad \Rightarrow \quad B_v = \frac{T_e}{\omega_r} = \frac{\frac{2}{3}PP\lambda_f i_q}{\omega_r} \tag{6.2}$$

As seen in Figure 6.11, the speed at steady state is approximately 61 rad/s and the current is 1 A, which results in:

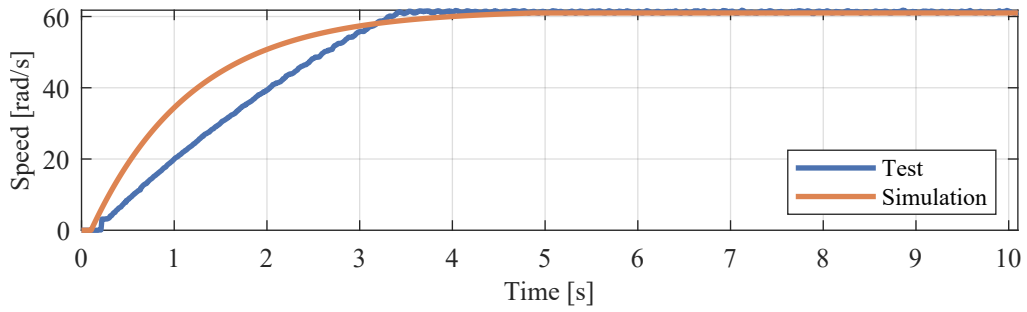
$$B_v = \frac{2}{3}\frac{PP\lambda_f i_q}{\omega_r} = 0.0022\,\mathrm{N\,m\,s} \tag{6.3}$$

Letting the motor spin freely during a step input in q-axis current to 1 A gives the speed and voltage response shown in Figure 6.12, where the test response is plotted alongside the simulation response. These show different transient behaviour.

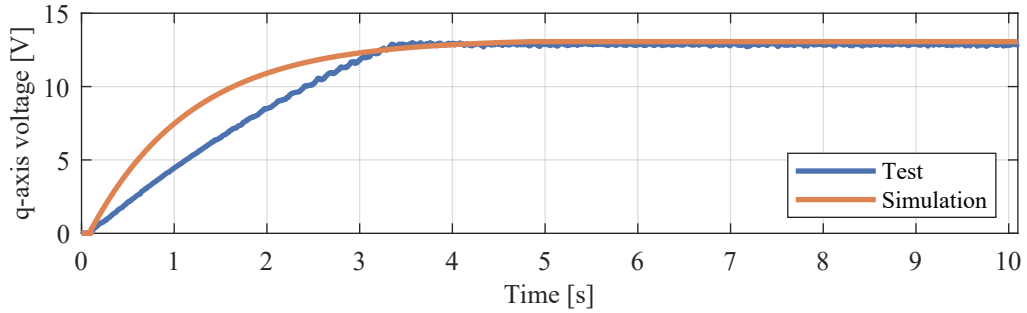
Assuming that the inertia, pole pairs, resistance and inductance are the known parameters, leaves the friction and the flux linkage as variables. By introducing Coulomb friction to the dynamic equation of motion:

$$\dot{\omega}_r = \frac{1}{J}(T_e - B_v\omega_r - \mu\ \mathrm{sign}(\omega_r)) \tag{6.4}$$

where μ is the Coulomb friction coefficient.



(a) Motor speed.



(b) q-axis voltage.

Figure 6.12: Response with a q-axis current of 1 A applied at $t = 0.1$.

6.3.1 Parameter Identification Using Optimisation

To identify the true friction coefficients and the true flux linkage, an optimisation algorithm is employed. The objective function is given as (6.5). In each iteration ten simulations are run to simulate the response from a q-axis current reference $i_q^* = 1\,\text{A}$ to $i_q^* = 2\,\text{A}$ in 0.1 A increments, and compared to ten tests with the same current reference. The total error between the simulations and tests are squared and summed to calculate the objective function.

$$J(x) = \sum(\omega_{sim} - \omega_{test})^2 + \sum(v_{q,sim} - v_{q,test})^2 \tag{6.5}$$

where

$$x = \begin{bmatrix} B_v & \mu & \lambda_f \end{bmatrix}$$

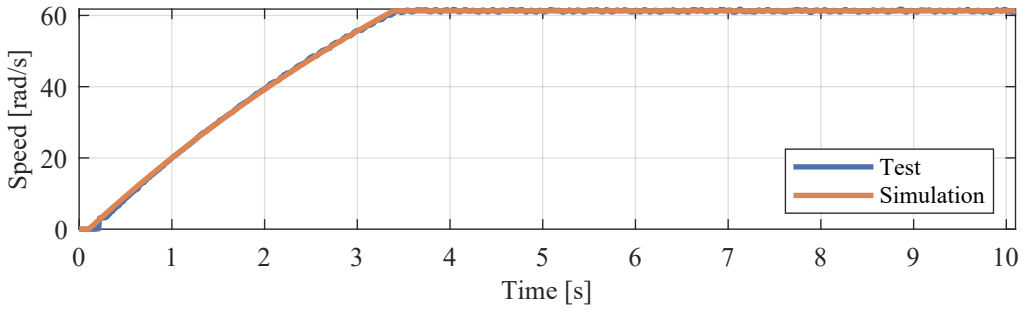
The evaluation of the objective function is summarised in the following steps:

1. The optimisation algorithm is given an initial guess for design variables (B_v,μ and λ_f).
2. The Simulink model is run ten times using the design variables.
3. The error between the Simulink model's speed and voltage, and the speed and voltage measured in the test is calculated.
4. The objective function is evaluated.

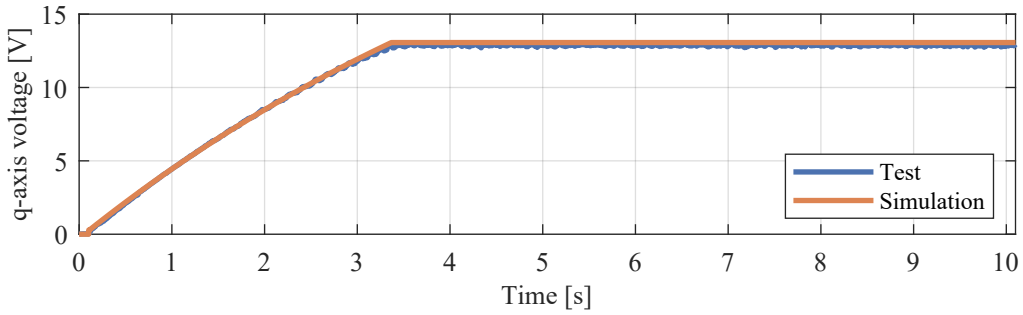
This is run iteratively until the optimisation algorithm find a local minimum. The optimised friction parameters and flux linkage is found as:

$$B_v = 887 \cdot 10^{-6}\,\mathrm{N\,m\,s}$$
$$\mu = 0.181\,\mathrm{N\,m} \tag{6.6}$$
$$\lambda_f = 0.014\,\mathrm{Wb}$$

The speed and voltage response using the optimised parameters are shown in Figure 6.13.



(a) Motor speed.



(b) q-axis voltage.

Figure 6.13: Response with a q-axis current of 1 A applied at $t = 0.1$.

Implementing the Coulomb friction changes the non-conservative externally applied torque used in the Euler-Lagrangian derivation of the dynamics (see (4.16) on page 24) to (6.7).

$$\mathbf{Q} = \begin{bmatrix} \frac{T_{diff}}{r} L_{ACMW} + B_v \frac{2L_{ACMW}}{r}\dot{\alpha} + \mu \frac{2L_{ACMW}}{r}\mathrm{sign}(\dot{\alpha}) \\ 2B_v\dot{\beta} + 2\mu\,\mathrm{sign}(\dot{\beta}) \\ \frac{T_{sum}}{r} + \frac{2B_v}{r}\dot{s} + \frac{2\mu}{r}\mathrm{sign}(\dot{s}) \end{bmatrix} \tag{6.7}$$

6.4 Code Utilised in the Experiments

This section explains the code used in the control of the MWIP system.
In Figure 6.14, a flow chart describing the Arduino code for wireless communication and obstacle avoidance is presented. These code blocks are the first to be executed after predetermination of variables, and a setup code which enables the different pins. The point of these code blocks is to receive input from the user, and ensure that the MWIP does not collide with surrounding objects. Secondly, the

code block described in Figure 6.15 is executed, this code block is used to extract signals from the IMU and convert them into a current reference for the two motors. Here, various control schemes can be implemented in the "Torque calculation" block based on both the IMU signals and the motor velocities and positions. The full Arduino code is found in Appendix I.

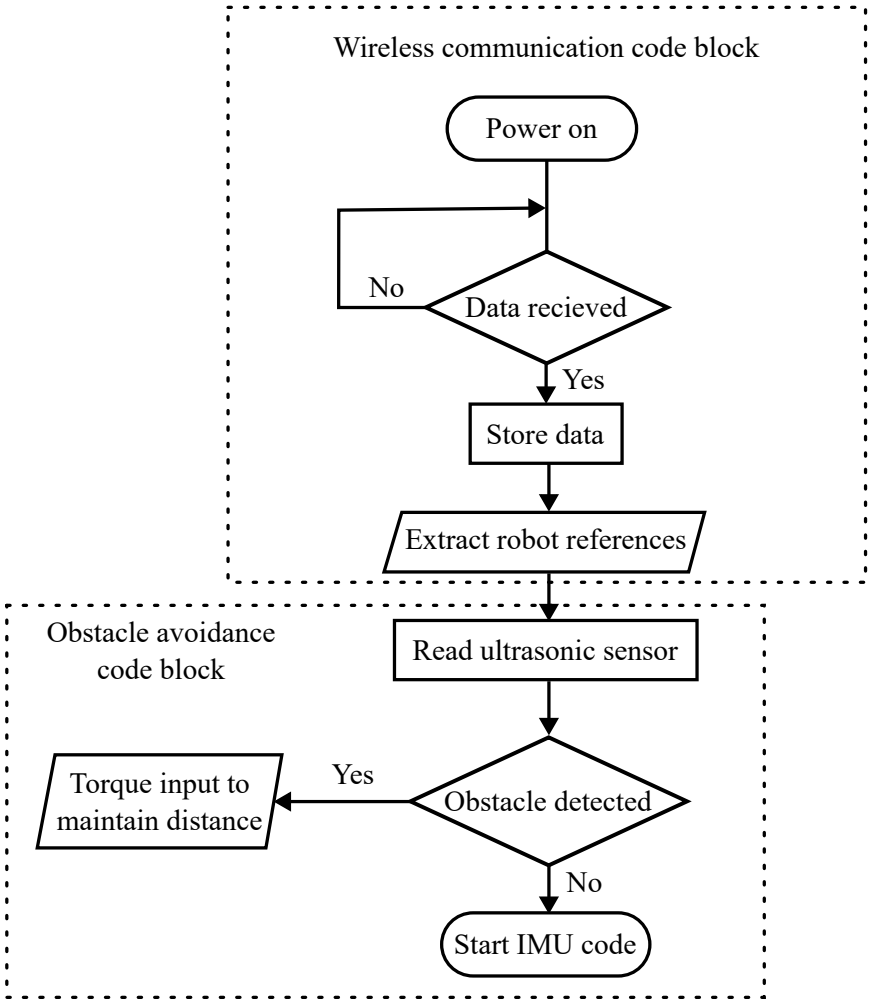


Figure 6.14: Flow chart describing the working principle of the wireless communication and obstacle avoidance Arduino code blocks.

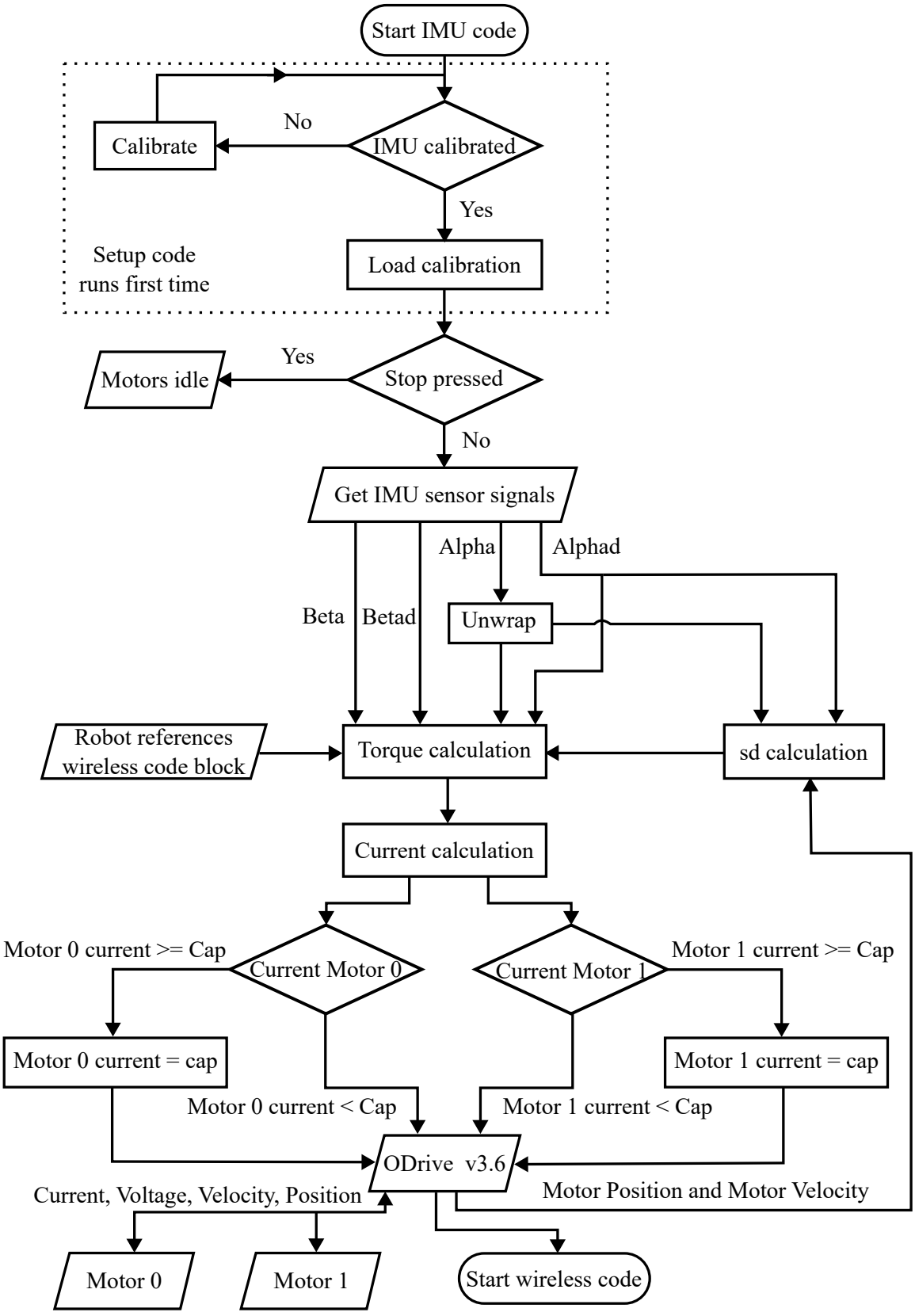


Figure 6.15: Flow chart describing the working principle of the IMU Arduino code block.

6.5 Validation of the Mobile Wheeled Inverted Pendulum Model

For the validation of the MWIP system, an open loop test of the system is not viable because the open loop system is unstable. Therefore, it is chosen to use the closed-loop system with the LQR controller as seen in Figure 5.2 where the ODrive is controlling the current loops.

6.5.1 Analysis of the Modelled Closed-Loop System

To test the modelled closed-loop system, a square wave reference is given to either the yaw angle or linear speed, respectively, while the rest of the states are given a zero reference throughout the simulation.

Figure 6.16 shows the closed-loop yaw and speed step response using the LQR controller on both the linear and nonlinear model with both a full and empty paint tank. Where it is seen that the linear model sufficiently models the dynamics of the system. Figure 6.16d shows the speed step response with an empty tank where it is seen that the controller has sufficient disturbance rejection to control the robot with a varying mass. The Matlab code used to run the simulation is given in Appendix E.1.

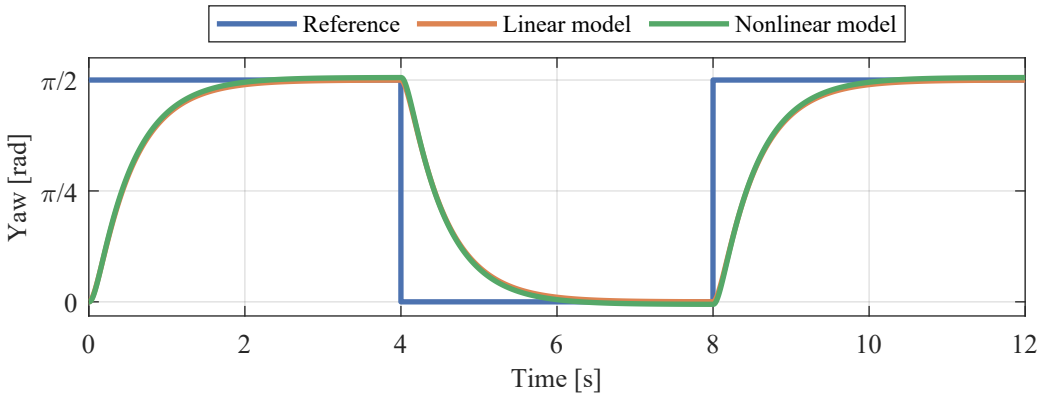
6.5.2 Test of the Modelled Closed-Loop System

Initial testing of the MWIP shows that the robot is able to stay upright but converges to a pitch angle just above zero, which results in the robot drifting slightly and not standing still. During the iterative process of determining the best LQR gains, the Teensy and the radio communication device failed, which limited further testing of the prototype. The Teensy still turns on, but the communication between the PC and the Teensy is nonexistent. It is assumed that the micro-USB port failed on the board.

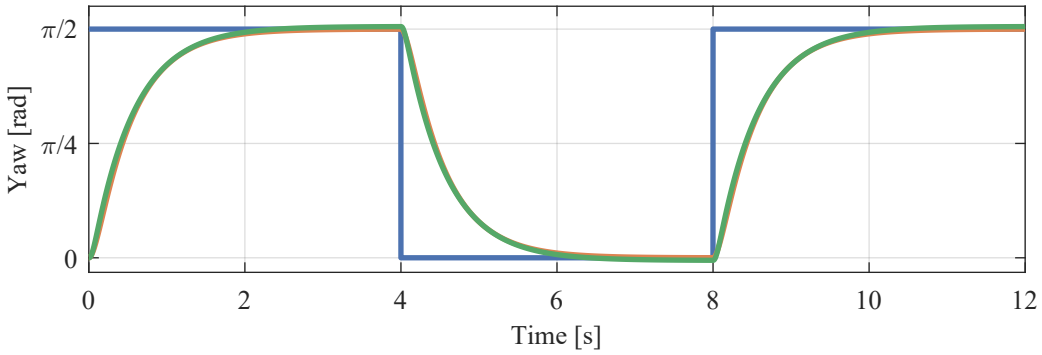
A new Teensy is acquired and testing of the prototype continues, with no change to the other electrical components and their layout, in the hope of the failure being a fluke. The only difference is that the new Teensy is configured to only receive power through its 5 V pin and not through the PC via the micro-USB.
This however resulted in the new Teensy also failing, this time likely caused by a short between the 3.3 V pin and the 5 V pin, which is fatal for this type of MCU. Continuity is measured between several of the I/O pins, ground and the 3.3 V pin, which is typical for this type of failure [59, 60].

6.5.3 Changes to the Prototype

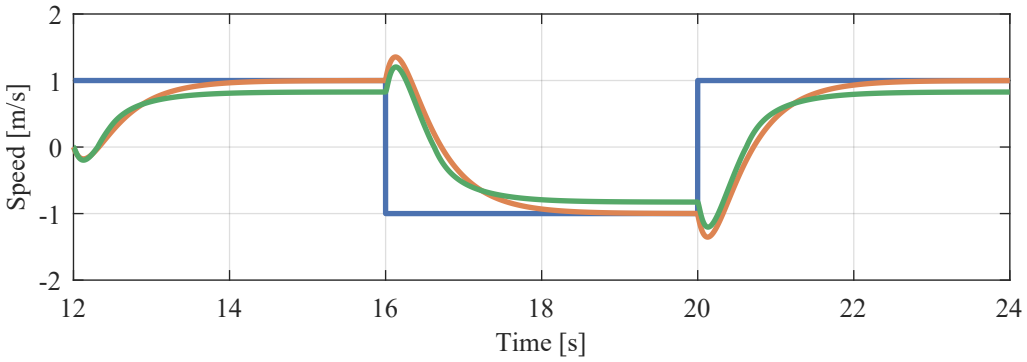
To resolve the issue new units are ordered, however the Teensy 4.0 model is unavailable, thus two Teensy 4.1 MCUs are ordered instead. As a new MCU has to be incorporated into the system, it is chosen to make modifications to the system. Instead of powering the Teensy through the buck-converter from the battery, it is chosen to power the Teensy from the PC during troubleshooting and by the ODrive's 5 V output pin during testing when it is not connected to a PC. As a consequence, the distance sensors are removed and the perfboard is changed to a breadboard for easy assembly and testing.



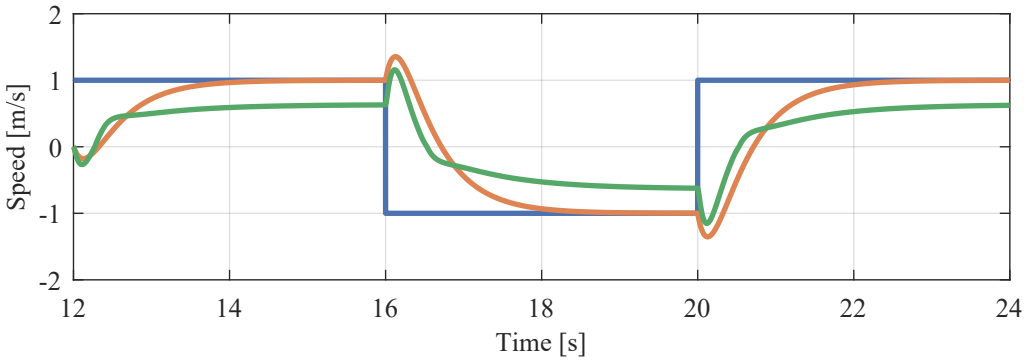
(a) Yaw step response with full tank



(b) Yaw step response with empty tank



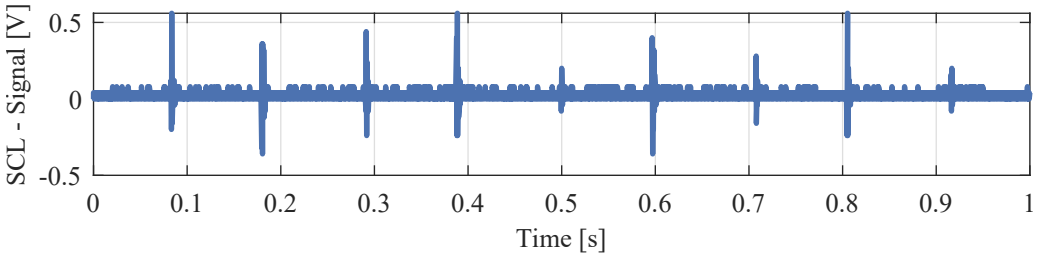
(c) Speed step response with full tank



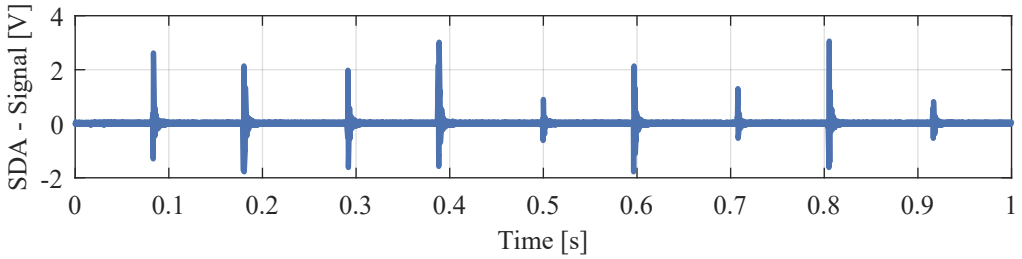
(d) Speed step response with empty tank

Figure 6.16: Closed-loop step response of linear and nonlinear model with full and empty paint tank.

After implementing the Teensy 4.1, the signals sent by the IMU are examined before initiating further tests. While examining the IMU signals, an abrupt stop of the signals is observed between the MCU and the IMU. Therefore, it is chosen to measure the input pin from the MCU to the IMU, the results are presented in Figure 6.17. Activating the pull-up resistor yields the signals given in Figure 6.18. As is seen in both figures, the I2C clock and data signals do not have clear high and low signals, which corrupts the communication, between the Teensy and the IMU.

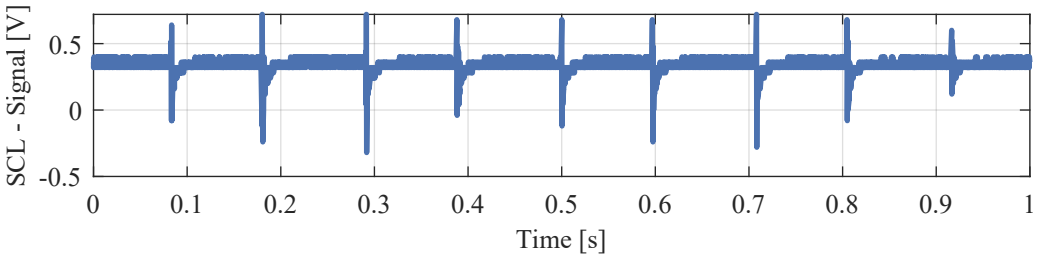


(a) I2C clock pin voltage with the pull-up resistor off.

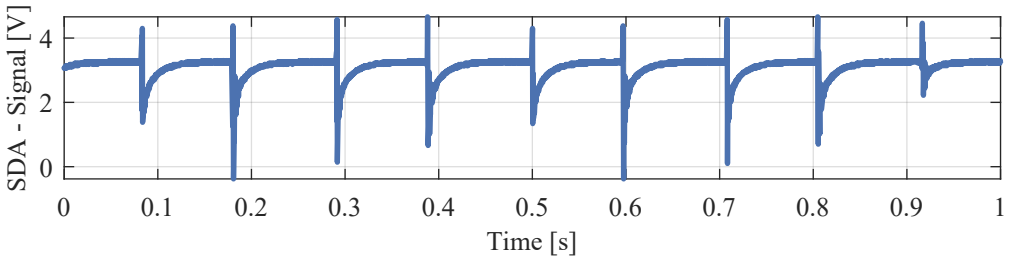


(b) I2C data pin voltage with the pull-up resistor off.

Figure 6.17: BNO055 I2C clock and data pin voltage with the pull-up resistor off.



(a) I2C clock pin voltage with the pull-up resistor on.

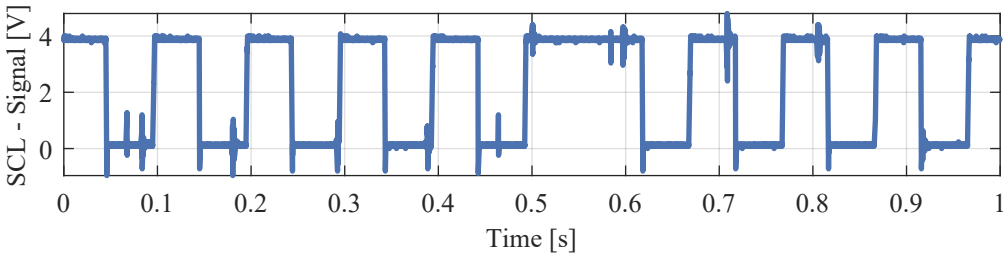


(b) I2C data pin voltage with the pull-up resistor on.

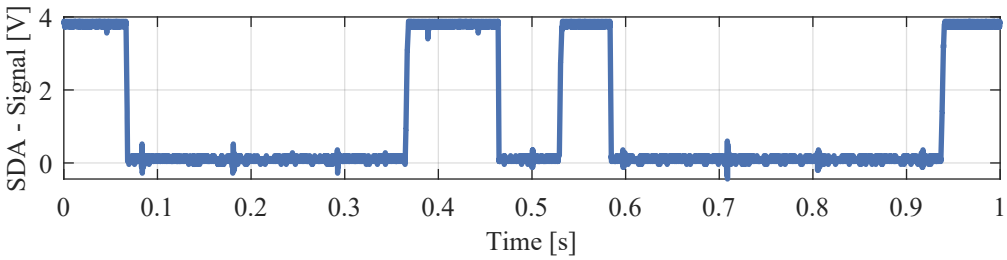
Figure 6.18: BNO055 I2C clock and data pin voltage with the pull-up resistor on.

To determine if the new MCU is causing the issue, the MCU is connected to the old IMU GY-91. It is observed that the connection between the MCU and the GY-91 is not disrupted over an extended period

of time. Then, the signals received from the GY-91 are examined. The examined signals with the pull-up resistors off are shown in Figure 6.19, where high and low signals are seen with some noise, possibly due the pull-up resistors on the IMU, enabling the pull-up resistors on the Teensy yields the signal given in Figure 6.20 with less noise than with the pull-up resistors off.

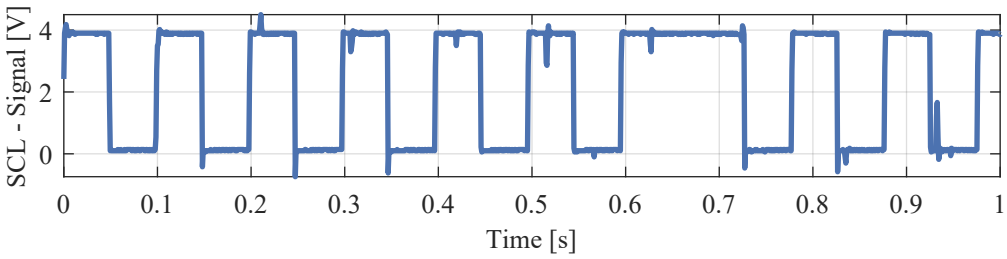


(a) I2C clock pin voltage with the pull-up resistor off.

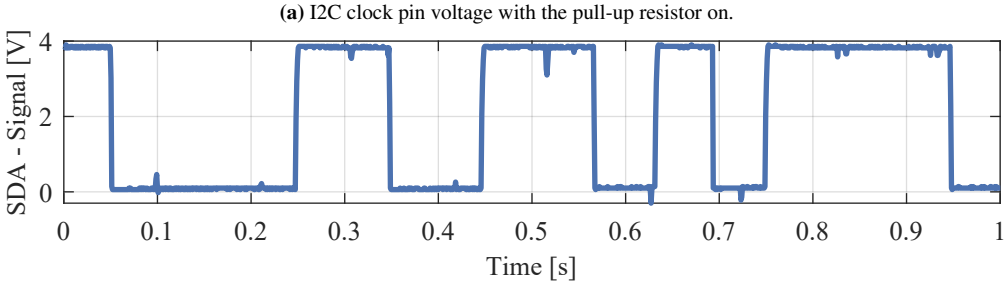


(b) I2C data pin voltage with the pull-up resistor off.

Figure 6.19: GY-91 I2C clock and data pin voltage with the pull-up resistor off.



(a) I2C clock pin voltage with the pull-up resistor on.



(b) I2C data pin voltage with the pull-up resistor on.

Figure 6.20: GY-91 I2C clock and data pin voltage with the pull-up resistor on.

Based on the results, it is chosen to proceed using the old GY-91 10-axis orientation sensor. In Appendix H.4 the three accelerometer signals, gyroscope signals, and magnetometer signals are shown. It is seen that the IMU is not capable of measuring magnetic fields which limits its yaw measurement to the gyroscope signal and removes its ability to tilt compensate the roll and pitch angles.

6.6 IMU Measurement Noise

Figure 6.21 shows the raw data from the gyroscope measuring the pitch angle β of the robot. Through time 0 s to 4.5 s the robot is manipulated manually and the sensor measures a change in angle as shown. At a time of 6 s the battery is connected and the ODrive receives power, some time after this the IMU measurements are corrupted and spike up to 34.9 rad/s.

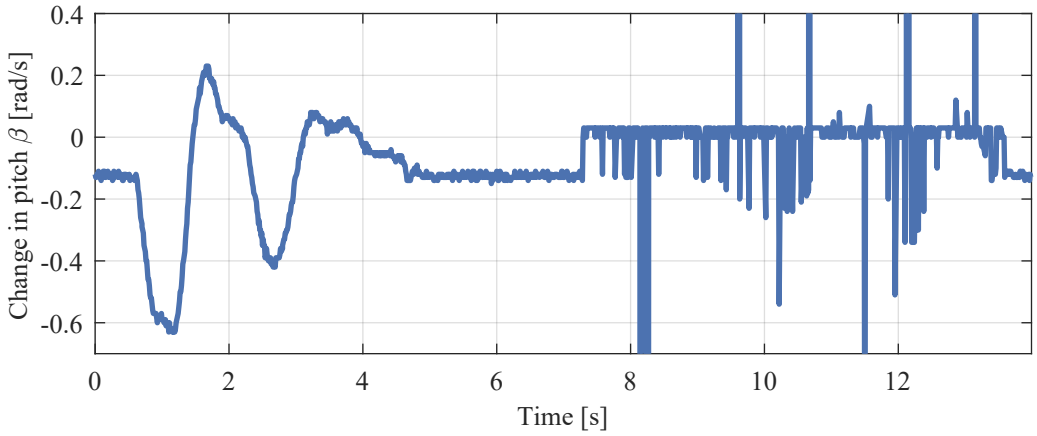


Figure 6.21: IMU pitch angle measurements when connecting ODrive to power.

6.7 Summary

Initially, the Hall effect sensor outputs are measured to ensure their integrity. Subsequently, the inner current loop is validated and is found to behave similarly to the modelled system. With the current loop validated the motors are validated from input torque to speed, and motor parameters are estimated. With the models validated and the parameters found an Arduino script is made to control the MWIP with LQR control. Initial testing is performed before the MCU and transceiver communication fails, which leads to troubleshooting faults in the design of the prototype.

Chapter 7

Discussion

The chapter entails the observations made, with the purpose of discussing their causes.

7.1 Differing Hall Sensor Voltage Magnitudes

To explore the cause of differing voltage magnitudes the structure of the Hall effect sensor is investigated. In Figure 7.1, a sketch of a Hall effect sensor is depicted. The relationship between the output voltage and the components of the Hall sensor is described by (7.1).

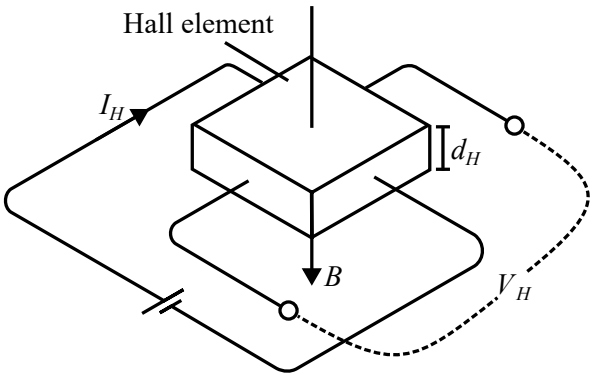


Figure 7.1: Sketch of a Hall effect sensor (B: magnetic field density, d_H: width of Hall sensor element, I_H: Hall sensor current, V_H: Hall sensor voltage).

$$V_H = \frac{R_H}{d_H} I_H B \tag{7.1}$$

Where V_H is the voltage output of the Hall effect sensor, R_H is the Hall constant, d_H is the width of the Hall element, I_H is the current flowing through the Hall element, and B is the magnetic field density [48]. The cause of the difference in voltage level is suspected to originate from a variance in either R_H or d_H between the three Hall effect sensors, which would explain the difference in voltage level. This is suspected as the difference in magnitude is constant where a difference in B or I_H would depend on surrounding magnetic fields and thereby be varying as described by the following.

A surrounding magnetic field induces an electromotive force in a current carrying wire in said magnetic field according to Faraday's law [61]:

$$emf = -\frac{d\Phi_M}{dt} = \int \vec{B_{mf}} \cdot \vec{dA} \tag{7.2}$$

Where emf is the electromotive force, Φ_m is the magnetic flux, t is the time, B_{mf} is the magnetic field, and A is the surface area which the magnetic field lines passes. The magnetic field can be calculated

through Ampere's law [61]:

$$B_{mf} = \frac{\mu_0 I}{2\pi r_{dw}} \tag{7.3}$$

Where μ_0 is the permeability of air, I is the current flowing through the wire, and r_{dw} is the distance to the surrounding wire which produce the magnetic field. The induced current can be calculated through Ohm's law [61]:

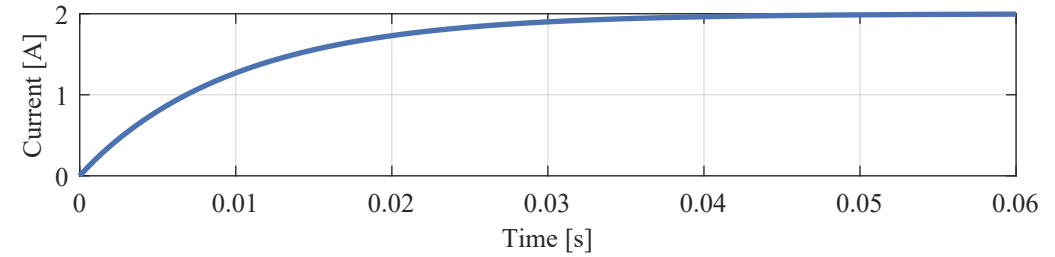
$$I = \frac{emf}{R} \tag{7.4}$$

Where I is the induced current, and R is the resistance in the wire. As the induced electromotive force is produced by a change in magnetic flux, the induced current in the Hall effect sensor would be time varying, which is not observed during the testing.

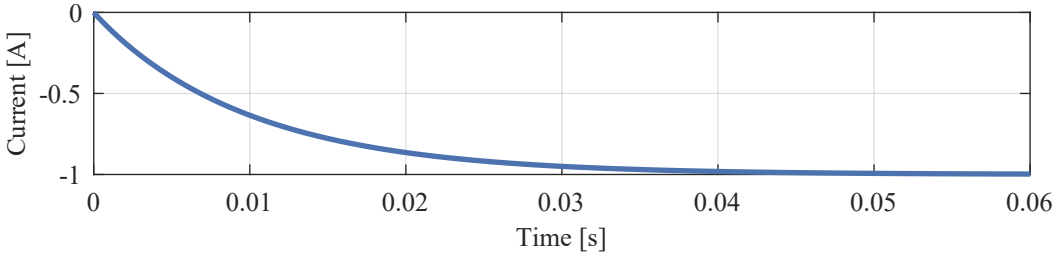
7.2 Expected Phase Currents for Blocked Rotor Test

The expectation is that phase b and phase c both have a current of -1 A, but the test result rejected that hypothesis as described in Section 6.2.2.

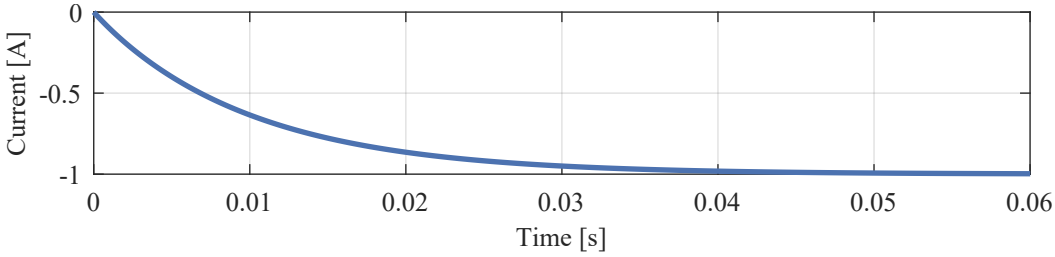
A simulated current response of the three phases, displaying the expected theoretical responses, is shown in Figure 6.7.



(a) Phase a current.



(b) Phase b current.



(c) Phase c current.

Figure 7.2: Simulated phase current response in blocked rotor test.

It is seen the b and c phases are of equal magnitude as opposed to the observed results from the physical system. During a blocked rotor test the electrical angle θ_e is constant, and if the reference current i^*_{dq} is constant the reference vector V^* is constant, and the sector is constant. By assuming that the reference vector is in sector 1 shown in Figure 4.15 on page 39, the adjacent switch states are V_1 and V_2, where the active switches are $(1,0,0)$ and $(1,1,0)$. That means that during one switching sequence phase a only sees positive currents, phase c only sees negative currents, and phase b sees both negative and positive currents. Assuming that both switching states are on at an equal amount of time, phase b would have an average current of 0 A. Then the steady state current of phase a and c is determined by the duration the switch states are active.

This could be investigated by measuring the emitter pins of the MOSFETs while running the blocked rotor test to measure the signal. However, it has not been possible to locate the emitter pins on the ODrive, thus the investigation is abandoned.

7.3 Offset in Current Value for Blocked Rotor Test

In the tests of the closed current loop, there is an offset in the current value. To investigate this, two additional tests are made: one only measuring noise and one to enter closed-loop mode without a torque reference.

The noise measurement of the phase current is given in Figure 7.3 where the system is turned off and the oscilloscope measures phase currents. There is observed a fluctuating current with a maximum amplitude of 2 mA. Therefore it verifies the system is subject to a certain amount of noise, but it is of negligible size. The test of the system entering closed-loop control mode is shown in Figure 7.4. It is

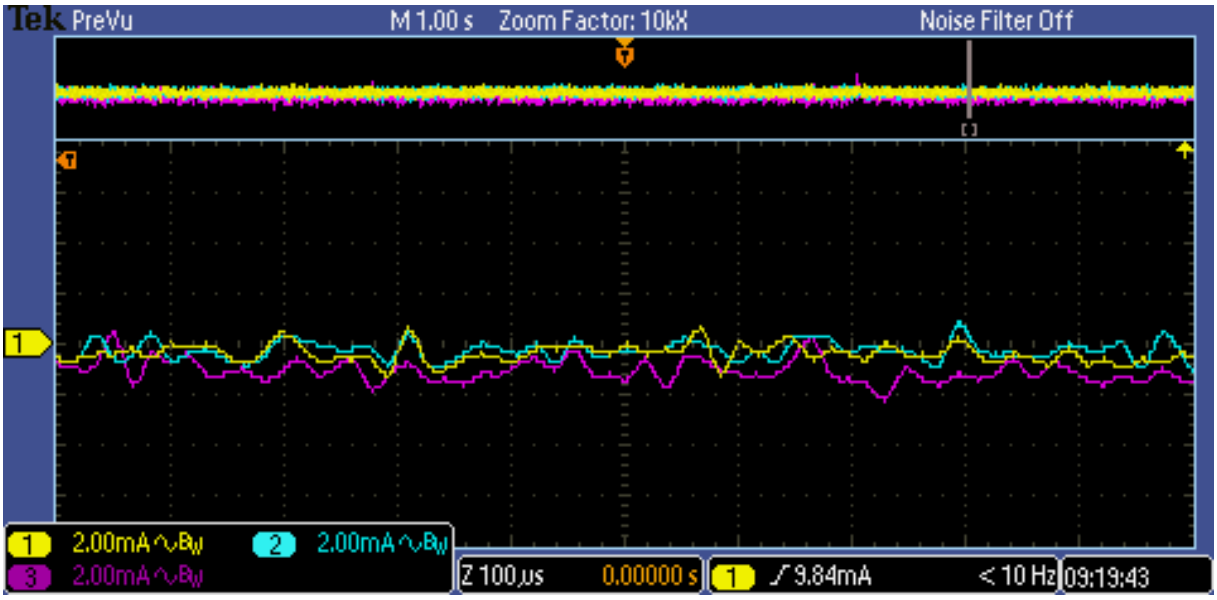


Figure 7.3: Measurement of noise for the phase currents.

observed that the phase currents draw currents with an amplitude of approximately 25 mA, which verifies that the system requires a certain amount of current upon entering closed-loop control mode. The phase currents are fluctuating similarly to what is observed in the experiment (See Appendix H.2),

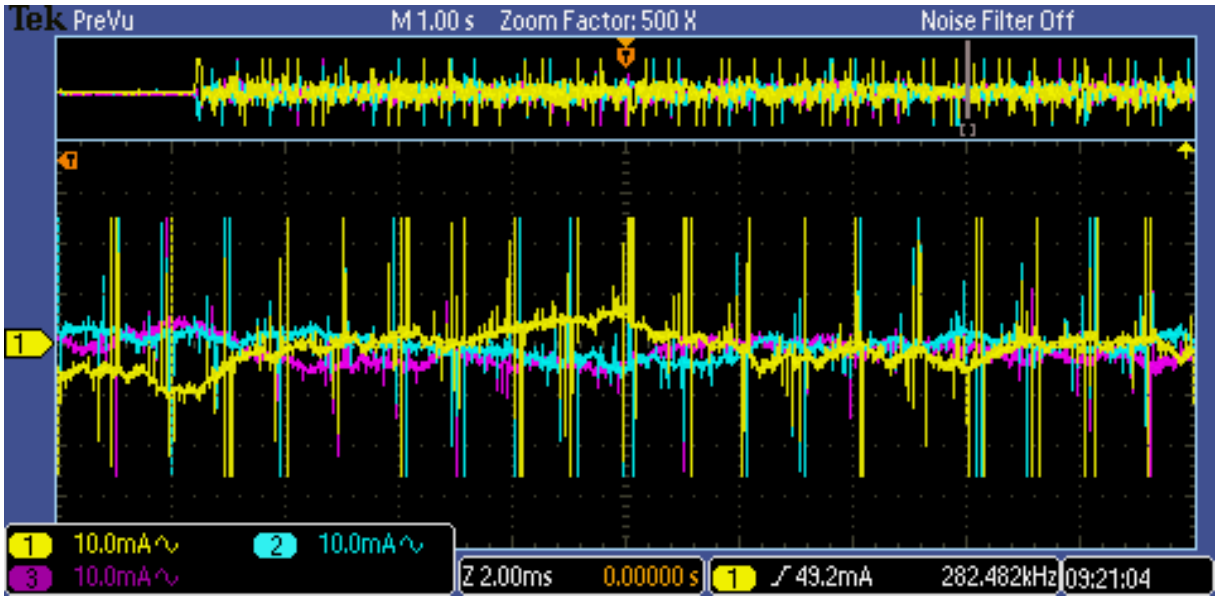


Figure 7.4: Measurement of phase current when commanding the system to enter closed-loop mode without a torque reference.

but with lower magnitude. However, even combined with the noise, it cannot reach the same magnitude as the offset as observed in the experiments, which indicates that an additional unknown cause might be affecting the current readings.

7.4 Hardware

In this section the hardware and its implementation is discussed.

Motor and Motor Driver

The motor and Hall effect sensors should be able to perform the required control, as it is close to its usual functionality of hover board control, which is also in the low speed high torque domain. The motor driver however is a third party motor driver which is normally use for hobby grade BLDC motors and PMSMs operating at high speed and low torque.

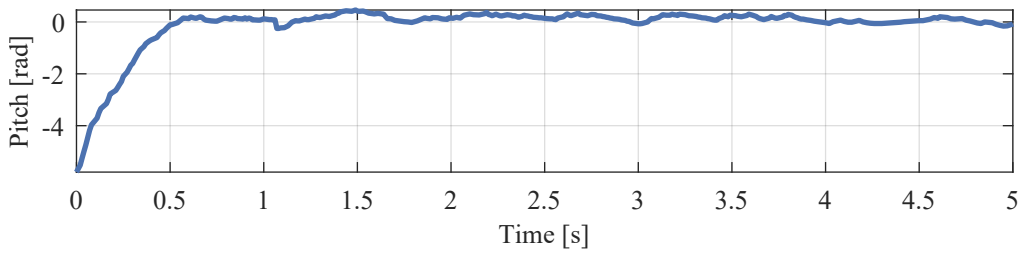
Communication

The serial communication between the Teensy and ODrive both support baud rates up to and beyond the standard 115 200 bit/s, but during testing the communication between the two components would suddenly terminate after a few seconds of operation. This problem is solved by lowering the baud rate to 57 600 bit/s where the communication is stable.

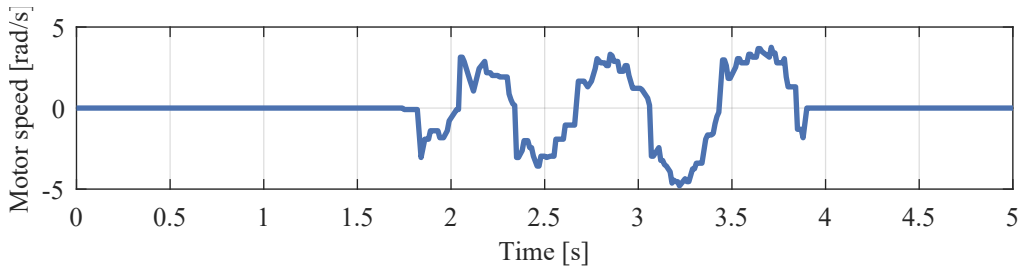
This approach is unsatisfactory as the speed of the control loop is essential for fast control of the robot. The control of the robot is, throughout this thesis, only studied in the continuous case which is less accurate with fewer samples per second. An analysis of the control in the discrete domain needs to be made in order to ascertain its impact on the control.

During testing of the MWIP issues with communication between the MCU, ODrive, and IMU are encountered. To investigate these communication problems, a test is performed where the MCU circuit

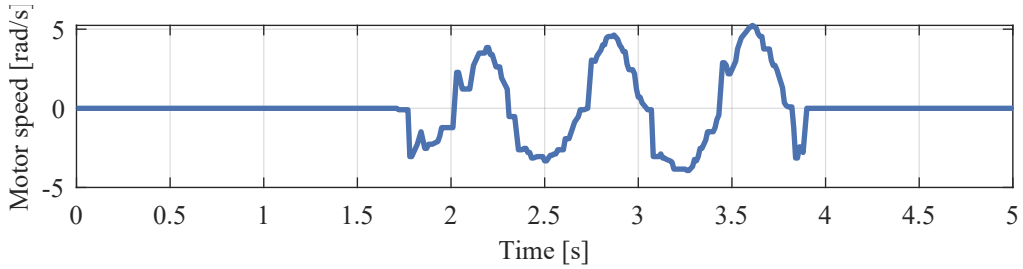
is isolated from the ODrive circuit as illustrated in Figure 7.8. During the test the pitch angle and motor velocity from both motors are logged, while the system is moved back and forth. The data output from this test is seen in Figure 7.5. From these results, it is seen that the IMU provides a pitch angle to the MCU, but the ODrive does not provide velocity feedback from the motors. According to [62] a common ground has to be present between the MCU and the ODrive to communicate via UART. Therefore the test is repeated with a common ground between the MCU and ODrive as illustrated in Figure 7.7. The data obtained from the test is given in Figure 7.6. It is seen that the ODrive sends motor velocity values to the MCU, but the IMU no longer provides a pitch angle. A possible cause for this problem is ground looping [63]. Ground loops result in a difference in potential between the ground wires, and thereby, the current flowing in the ground wire can not be pulled to zero. Ground loops form when a system has multiple paths to ground as in Figure 7.7 where the ground for the system can either be the battery in the PC or the battery powering the ODrive. When ground loops are present in a system, the GPIO signals can vary significantly from the intended 3.3 V logic level, and current flows through the pins into the MCU if the induced voltage from the wire inductance and current is sufficiently large [63]. It could explain why two of the MCUs have stopped responding when the Teensy ports are only 3.3 V tolerant. Furthermore, ground loops introduce noise to surrounding electrical signals, which could result in increased noise measured by the IMU.
A potential solution to the problem is to include an isolator between the MCU and the ODrive, thereby allowing transfer of signals without a physical ground between the components.



(a) Pitch.

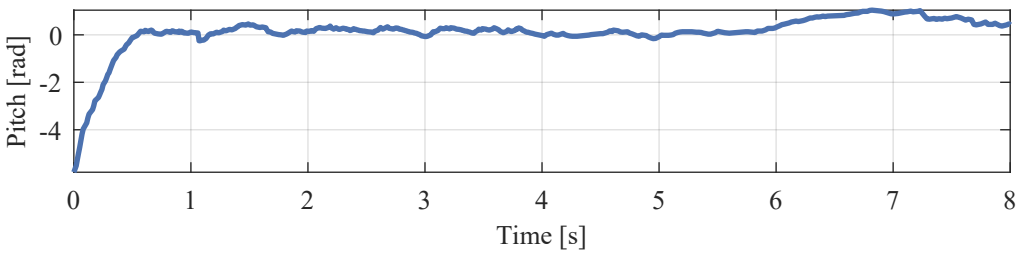


(b) Motor 0 velocity.

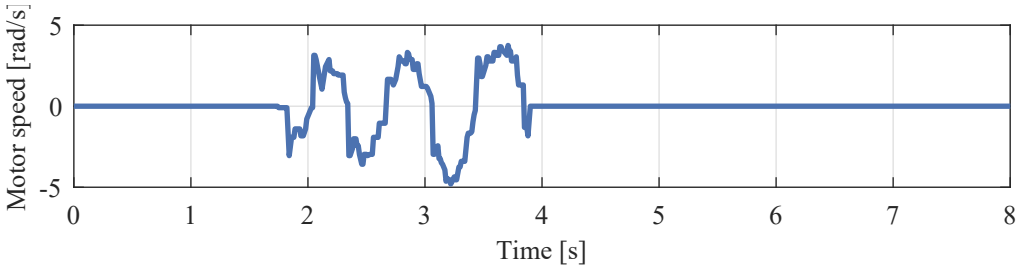


(c) Motor 1 velocity.

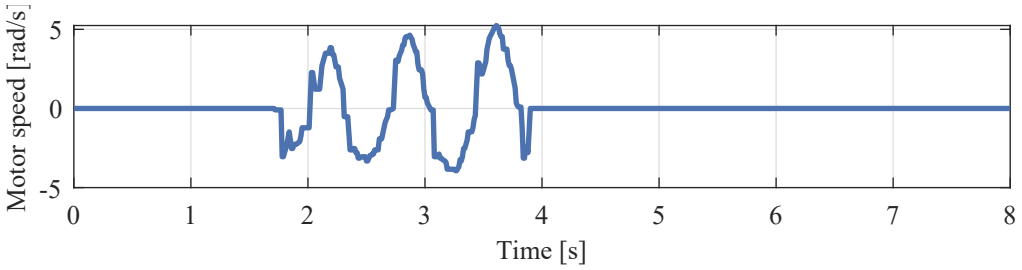
Figure 7.5: Pitch and motor speed readings with no common ground.



(a) Pitch.



(b) Motor 0 velocity.



(c) Motor 1 velocity.

Figure 7.6: Pitch and motor speed readings with common ground.

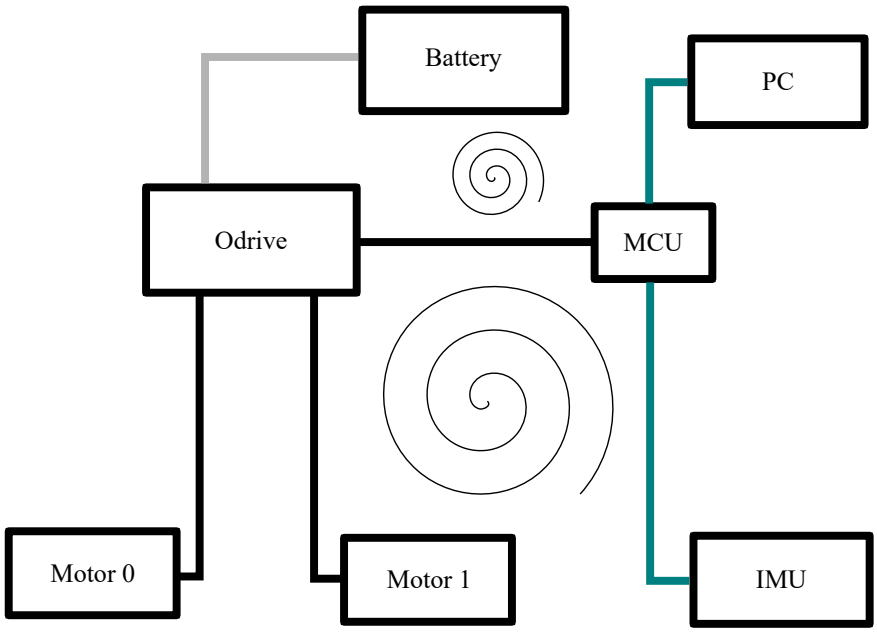


Figure 7.7: Illustration of possible ground loops in the system (Sprials represents the ground loops, each colour represents a ground reference at different voltage references, IMU: inertia measurement unit, MCU: microcontroller unit).

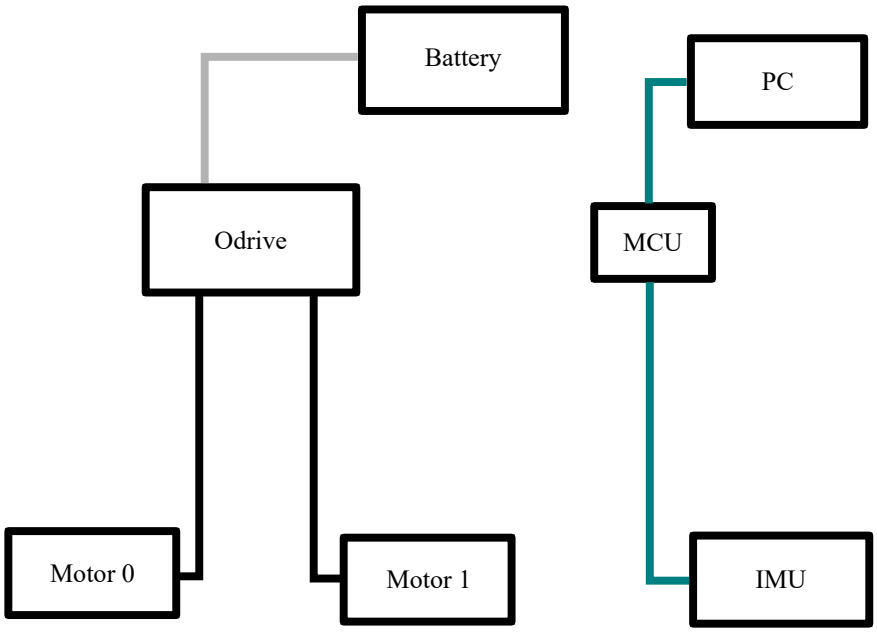


Figure 7.8: Illustration of separated ground connections in the system (Each colour represents a ground reference at different voltage references, IMU: inertia measurement unit, MCU: microcontroller unit).

7.5 Modelling of System

The dynamic model of the robot lacks two major disturbances, the robot travelling on uneven or tilted surfaces, and the paint sloshing with in the paint tank. In real world application the robot could be travelling on uneven or tilted surfaces, and the paint sloshing could be a significant disturbance to the system. The ability to model such events could give further understanding of the dynamic system behaviour in these conditions and allow for quick testing of control strategies in repeatable simulation conditions.

Chapter 8

Conclusion

The former version of the prototype is analysed, and a revision of the design to extend the robot's capabilities is made in accordance with the requirements stated in Table 3.1.

In the modelling of the system, a mechanical model of the MWIP and an electrical model of the BLDC motors are developed. Subsequently, an investigation of the motors' back emf is made, where it is shown that the back emf waveform is sinusoidal, which implies the motors are PMSMs. Therefore, an electrical model of the PMSMs is made. Field Oriented Control is used to control the torque of the PMSMs.

A transfer function for the q-axis current is developed based on the linearised electrical model. Using the transfer function, a PI controller is derived using the method of zero-pole cancellation to determine the gains. The transfer function is validated using the closed q-axis current loop with the PI controller. The results showed a difference of 1.96 % in the time constant between the model and the system.

For control of the torque of the PMSMs, an LQR controller is developed. The controller is derived using the linearised mechanical model in state-space representation. The LQR controller is tested to confirm that the controller ensures a stable closed-loop response.

Having developed the linear controllers to be used in the experiments, a complete nonlinear simulation model is derived with the intention of comparison with the experimental results. However, as mentioned in Chapter 6, the physical system experienced unresolved issues. Consequently, data collection using LQR and the outer loop is unobtainable. Therefore, it is not possible to evaluate the model against the physical system or to test the performance of the LQR controller on the physical system.

Chapter 9

Future Work

This chapter presents topics that can be used in the further development of the prototype, specifically, regarding the control of the system, the incorporation of the paint tank dynamics, and the removal of ground loops.

9.1 Nonlinear Control

As described in the introduction, MWIP systems are prone to model uncertainties, varying work conditions, external disturbances, characteristics of underactuated robot, and the nonlinear instability. Furthermore, in this case, it is intended to incorporate a tank containing varying levels of paint, which affects the system's CM and the inertia as described in Section 4.1. Additionally, the sloshing of the paint acts as a disturbance to the system. Therefore, it is beneficial to use nonlinear controller design which is more robust than a linear controller design. Below, two control design are mentioned that could be of interest for further development of the MWIP system.

9.1.1 Sliding Mode Control

An example of a control design would be the sliding mode controller. This controller design is based on the discrepancy between the actual plant and its mathematical model due to external disturbances, plant parameters, and unmodelled dynamics. According to [64], it might possibly be the best approach for handling bounded uncertainties or disturbances and unmodelled dynamics.

9.1.2 Model Predictive Control

Model predictive control (MPC) is a control scheme controlling the system by predicting future behaviour based on a dynamic model of the system. MPC can be implemented using either the linear or nonlinear model. The input to the system is determined by predicting the future behaviour of the system under certain control inputs, and optimising in order to minimise the state error. The input to the system is determined by predicting the future behaviour of the system under certain control inputs, and optimising in order to minimise the state error.

9.2 Investigation of the Dynamical Effects of Sloshing

As of this point, the tank has not been included physical system during the experiments. However, as pointed out in the introduction, the sloshing effects affect the control of the system. The impact of the sloshing effects can have on the system is unknown, it might result in a large enough disturbance on the

control to cause the system to become unstable. Therefore, an investigation of the dynamical impact of the sloshing should be conducted.

9.2.1 Prevention of Sloshing

In case of the sloshing effects are too large to be compensated solely by the control, the solution would be to reduce the amount of liquid sloshing. Different methods, such as baffles or floating foams, are developed for prevention of liquid sloshing by reducing the amplitude through energy dissipation. It is unclear which of these methods would result in the greatest reduction, consequently the optimal choice remains speculative. Furthermore, both of the options would require some volume, leaving less space for the paint in the paint tank.

9.3 Removal of Ground Loops

In order to resolve the communication issues between the MCU, ODrive, and IMU an isolator can be placed between the ODrive and MCU. The function of the isolator is to allow data transfer between the components without having a physical ground connection between them. For choice of isolator [63] recommends a ISO7762F signal isolator from Texas Instruments, which could resolve the communication issues.

9.4 Validation of the Mobile Wheeled Inverted Pendulum Model

As described in the Experiments chapter, the model of the MWIP is not validated due to hardware issues. It is essential to determine the validity of the model and verify it describes the system, as the control schemes are derived based on the model. The validation process can be performed using the approach mentioned in the Experiments chapter when the hardware issues have been resolved.

Bibliography

[1] International Federation of Robotics. (2023) World robotics 2023 report: Asia ahead of europe and the americas. Accessed: 12-03-2024. [Online]. Available: https://ifr.org/ifr-press-releases/news/world-robotics-2023-report-asia-ahead-of-europe-and-the-americas

[2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed., ser. Springer Handbooks. Cham: Springer International Publishing AG, 2017.

[3] J. Faigl, P. Váňa, R. Pěnička, and M. Saska, "Unsupervised learning-based flexible framework for surveillance planning with aerial vehicles," *Journal of field robotics*, vol. 36, no. 1, pp. 270–301, 2019.

[4] J. Saunders, S. Saeedi, and W. Li, "Autonomous aerial robotics for package delivery: A technical review," *Journal of Field Robotics*, vol. 41, no. 1, pp. 3–49, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22231

[5] B. Oliveira, P. Morais, H. R. Torres, J. C. Fonseca, and J. L. Vilaça, "Design optimization of medical robotic systems based on task performance metrics: A feasibility study for robotic guided vascular laser treatments," *Journal of Field Robotics*, vol. 41, no. 1, pp. 144–161, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22250

[6] H. M. La, N. Gucunski, K. Dana, and S. Kee, "Development of an autonomous bridge deck inspection robotic system," *Journal of field robotics*, vol. 34, no. 8, pp. 1489–1504, 2017.

[7] H. Shi, R. Li, X. Bai, Y. Zhang, L. Min, D. Wang, X. Lu, Y. Yan, and Y. Lei, "A review for control theory and condition monitoring on construction robots," *Journal of field robotics*, vol. 40, no. 4, pp. 934–954, 2023.

[8] R. Fan, Y. Li, L. Yang, T. Qin, Y. Zhang, and Z. Wang, "Coordinated trajectory tracking control for mining excavator with oscillating external disturbance," *Journal of Field Robotics*, vol. 41, no. 2, pp. 258–272, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22257

[9] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of field robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[10] B. H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, and B. Cooper, "Athlete: A cargo handling and manipulation robot for the moon," *Journal of field robotics*, vol. 24, no. 5, pp. 421–434, 2007.

[11] P. La Hera, O. Mendoza-Trejo, O. Lindroos, H. Lideskog, T. Lindbäck, S. Latif, S. Li, and M. Karlberg, "Exploring the feasibility of autonomous forestry operations: Results from the first experimental unmanned machine," *Journal of field robotics*, 2024.

[12] M. Li, K. Imou, K. Wakabayashi, and S. Yokoyama, "Review of research on agricultural vehicle autonomous guidance," *International journal of agricultural and biological engineering*, vol. 2, no. 3, pp. 1–16, 2009.

[13] Y. Hua, N. Zhang, X. Yuan, L. Quan, J. Yang, K. Nagasaka, and X. G. Zhou, "Recent advances in intelligent automated fruit harvesting robots," *The open agriculture journal*, vol. 13, no. 1, pp. 101–106, 2019.

[14] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, ser. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009, vol. 56.

[15] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert systems with applications*, vol. 165, 2021.

[16] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Hoboken: Wiley, 2011.

[17] M. Karkee and Q. Zhang, *Fundamentals of agricultural and field robotics*, ser. Agriculture automation and control. Cham, Switzerland: Springer, 2021.

[18] T. Fukao, "Field robotics: Applications and fundamentals," *Journal of robotics and mechatronics*, vol. 33, no. 6, pp. 1216–1222, 2021.

[19] M. Latifinavid and A. Azizi, "Development of a vision-based unmanned ground vehicle for mapping and tennis ball collection: A fuzzy logic approach," *Future internet*, vol. 15, no. 2, 2023.

[20] I. A. Hameed, C. G. Sorrenson, D. Bochtis, and O. Green, "Field robotics in sports: Automatic generation of guidance lines for automatic grass cutting, striping and pitch marking of football playing fields," *International journal of advanced robotic systems*, vol. 8, no. 1, pp. 113–121, 2011.

[21] TurfTank. (2024) Turf tank two. Accessed: 12-04-2024. [Online]. Available: https://turftank.com/us/turf-tank-two/

[22] A. K. Dolmer, G. O. Rodriguez, N. S. Christensen, P. L. Rossell, R. Rose-Hansen, and S. W. V. Nielsen, "Self-balancing robot for painting," Aalborg University, Tech. Rep., 2023.

[23] K. Yamafuji, Y. Miyakawa, and T. Kawamura, "Synchronous steering control of a parallel bicycle," *Nihon Kikai Gakkai ronbunshū. C*, vol. 55, no. 513, pp. 1229–1234, 1989.

[24] N. M. A. Ghani, F. Naim, and T. P. Yon, "Two wheels balancing robot with line following capability," *World Academy of Science, Engineering and Technology*, vol. 55, no. 7, pp. 634–638, 2011.

[25] G. H. Lee and S. Jung, "Line tracking control of a two-wheeled mobile robot using visual feedback," *International journal of advanced robotic systems*, vol. 10, no. 3, pp. 177–, 2013.

[26] K. M. Ibrahim and M. N. Noaman, "Optimal control approach for robot system using lqg technique," *Journal Europeen des Systemes Automatises*, vol. 55, no. 5, pp. 671–677, 10 2022. [Online]. Available: https://www.proquest.com/scholarly-journals/optimal-control-approach-robot-system-using-lqg/docview/2807002194/se-2

[27] W.-F. Kao, C.-F. Hsu, and T.-T. Lee, "Motion control of mobile-wheeled inverted pendulum robot with center-of-mass offset," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 2128–2133.

[28] J. Huang, M. Zhang, S. Ri, C. Xiong, Z. Li, and Y. Kang, "High-order disturbance-observer-based sliding mode control for mobile wheeled inverted pendulum systems," *IEEE transactions on industrial electronics (1982)*, vol. 67, no. 3, pp. 2030–2041, 2020.

[29] M. Zhang, J. Huang, and Y. Cao, "Adaptive super-twisting control for mobile wheeled inverted pendulum systems," *Applied Sciences*, vol. 9, no. 12, 2019. [Online]. Available: https://www.proquest.com/scholarly-journals/adaptive-super-twisting-control-mobile-wheeled/docview/2331354136/se-2

[30] J. Shen and D. Hong, "Model predictive control using dynamic model decomposition applied to two-wheeled inverted pendulum mobile robot," in *2022 19th International Conference on Ubiquitous Robots (UR)*, 2022, pp. 332–337.

[31] A. Unluturk and O. Aydogdu, "Machine learning based self-balancing and motion control of the underactuated mobile inverted pendulum with variable load," *IEEE Access*, vol. 10, pp. 104 706–104 718, 2022.

[32] J. Huang, M. Zhang, and T. Fukuda, *Robust and intelligent control of a typical underactuated robot : mobile wheeled inverted pendulum*, ser. Research on Intelligent Manufacturing. Singapore: Springer, 2023.

[33] N. Mahdavi Tabatabaei, E. Kabalci, and N. Bizon, *Microgrid Architectures, Control and Protection Methods*, ser. Power Systems. Cham: Springer International Publishing, 2020.

[34] Samsung SDI Co.,Ltd. (2009) Specification of product for lithium-ion rechargeable cell - model : Icr18650-26f. Accessed: 12-04-2024. [Online]. Available: https://www.batteryspace.com/prod-specs/ICR18650-26F.pdf

[35] Arduino S.r.l. (n.d.) 10DOF Gyro + G-Sensor + kompas + Barometer GY-87 GY-88 GY-91. Accessed: 15-04-2024. [Online]. Available: https://arduinotech.dk/shop/10dof-gyro-g-sensor-kompas-barometer-gy-87-gy-88-gy-91/

[36] ——. (n.d.) Nano v3.0 atmega328 16m 5v micro-controller ch340g board. Accessed: 15-04-2024. [Online]. Available: https://arduinotech.dk/shop/nano-v3-0-atmega328-16m-5v-micro-controller-ch340g-board/

[37] Elektronik Lavpris Aps. (n.d.) Product name: Dc-dc step down module power supply module power converter 5a xl4015. Accessed: 12-04-2024. [Online]. Available:

https://elektronik-lavpris.dk/files/sup216/151525_1661504246.pdf

[38] Handson Technology. (n.d.) Joystick shield for arduino uno/mega. Accessed: 12-04-2024. [Online]. Available: https://handsontec.com/dataspecs/module/Arduino%20Shield/Joystick%20Shield.pdf

[39] ODrive Robotics. (2021) Specifications. Accessed: 12-04-2024. [Online]. Available: https://docs.odriverobotics.com/v/0.5.4/specifications.html#electrical-specifications

[40] Parallax Inc. (2020) 6.5 inches hub motor with encoder. Accessed: 24-04-2024. [Online]. Available: https://www.parallax.com/product/6-5-hub-motor-with-encoder/

[41] Denver A/S. (n.d.) Denver hbo-6620black mk2. Accessed: 12-04-2024. [Online]. Available: https://denver.eu/productpdf/4291

[42] Bosch Sensortec. (2014) BNO055 Intelligent 9-axis absolute orientation sensor. Accessed: 16-04-2024. [Online]. Available: http://www.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf

[43] NXP Semiconductors. (2019) i.mx rt1060 crossover processors for consumer products. Accessed: 12-04-2024. [Online]. Available: https://www.pjrc.com/teensy/IMXRT1060CEC_rev0_1.pdf

[44] PJRC. Teensy 4.0 development board. Accessed: 29-05-2024. [Online]. Available: https://www.pjrc.com/store/teensy40.html

[45] Arduino S.r.l. Arduino nano. Accessed: 29-05-2024. [Online]. Available: https://store.arduino.cc/products/arduino-nano

[46] Seeed Technology Co., Ltd. (n.d.) Ultrasonic ranging module hc - sr04. Accessed: 12-04-2024. [Online]. Available: https://elektroshoppen.dk/help/HC-12_User_Manual.pdf

[47] Elecfreaks. (n.d.) Ultrasonic ranging module hc - sr04. Accessed: 22-05-2024. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[48] K. Sang-Hoon, *Electric motor control : DC, AC, and BLDC motors*. Amsterdam, Netherlands: Elsevier, 2017.

[49] ODrive Robotics. (2021) Odrive reference. Accessed: 17-05-2024. [Online]. Available: https://docs.odriverobotics.com/v/0.5.6/fibre_types/com_odriverobotics_ODrive.html

[50] O. Weigl, "Project hoverarm," https://discourse.odriverobotics.com/t/project-hoverarm/441, February 2018, accessed 28-05-2024.

[51] K. Bartholomew, "Project hoverarm," http://blog.kyleb.me/2018/01/hoverboard-motor-teardown.html, January 2018, accessed 28-05-2024.

[52] Anonymous, "Hoverboard foc motor settings vesc," https://forum.esk8.news/t/hoverboard-foc-motor-settings-vesc/25343/3, April 2020, accessed

28-05-2024.

[53] P. C. Krause, O. Wasynczuk, S. Sudhoff, S. Pekarek, and S. D. Sudhoff, *Analysis of Electric Machinery and Drive Systems.*, 3rd ed., ser. IEEE Press series on power engineering. Hoboken, N.J: Wiley, 2013.

[54] O. Weigl and Bockwurst, "Discussion on odrive driving principles," https://discord.com/channels/369667319280173067/562767743473156138/582558180777656337, May 2019, accessed 28-05-2024.

[55] S. Greenberg, "Discussion on odrive modulation index limit," https://discord.com/channels/369667319280173067/369678934985408524/1091109882578604123, March 2023, accessed 28-05-2024.

[56] D.-W. Chung, J.-S. Kim, and S.-K. Sul, "Unified voltage modulation technique for real-time three-phase power conversion," *IEEE transactions on industry applications*, vol. 34, no. 2, pp. 374–380, 1998.

[57] ODrive Robotics. (2021) Hoverboard motor and remote control setup guide. Accessed: 19-03-2024. [Online]. Available: https://docs.odriverobotics.com/v/0.5.6/hoverboard.html

[58] J. M. Philips and C. L. Parr, *Feedback Control Systems*, 5th ed. Pearson, 2013.

[59] P. Stoffregen, "Technical support of shorted teensy 4.0," https://forum.pjrc.com/index.php?threads/teensy-4-0-shorted-permanently-with-gnd.60185/, March 2020, accessed 30-05-2024.

[60] ——, "Technical support of shorted teensy 4.0," https://forum.pjrc.com/index.php?threads/3-3v-line-shorted-to-gnd.26506/, August 2014, accessed 30-05-2024.

[61] P. Scherz and S. Monk, *PRACTICAL ELECTRONICS FOR INVENTORS*, 4th ed. New York, NY: McGraw-Hill Education, 2016.

[62] ODrive Robotics. Uart interface. Accessed: 30-05-2024. [Online]. Available: https://docs.odriverobotics.com/v/0.5.6/uart.html

[63] ——. Ground loops. Accessed: 30-05-2024. [Online]. Available: https://docs.odriverobotics.com/v/latest/articles/ground-loops.html

[64] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, *Sliding Mode Control and Observation*, ser. Control Engineering. New York, NY: Springer New York, 2014.

[65] Advanced Navigation. Inertial measurement unit (imu) – an introduction. Accessed: 30-05-2024. [Online]. Available: https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/

[66] Analog Devices, Inc. Mems gyroscope provides precision inertial sensing in harsh, high temperature environments. Accessed: 30-05-2024. [Online]. Available: https://www.analog.com/

en/resources/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html

[67] S. S. Ahmadi, "Vibrations of micro-gyroscopes, fundamentals and review," *Multidisciplinary Engineering Science and Technology*, vol. 7, no. 9, 2020.

[68] V. Grygorenko. (n.d.) Sensing – magnetic compass with tilt compensation. Accessed: 30-05-2024. [Online]. Available: https://www.infineon.com/dgdl/Infineon-AN2272_PSoC_1_Sensing_Magnetic_Compass_with_Tilt_Compensation-ApplicationNotes-v04_00-EN.pdf?fileId=8ac78c8c7cdc391c017d0731b0d05573

[69] S. Tomazic, "Commutative rotations in 3d euclidean space and gimbal spatial angles," *IPSI BGD TRANSACTIONS ON INTERNET RESEARCH*, vol. 14, no. 1, 2018.

[70] Movella Technologies B.V. (2023) Understanding gimbal lock and how to prevent it. Accessed: 30-05-2024. [Online]. Available: https://base.movella.com/s/article/Understanding-Gimbal-Lock-and-how-to-prevent-it?language=en_US

[71] J. B. Kuipers, *QUATERNIONS and ROTATION SEQUENCES A Primer with Applications to Orbits, Aerospace and Virtual Reality*, 1st ed. Princeton, NJ: Princeton University Press, 2002.

[72] K. Ogata, *Modern Control Engineering*, 5th ed. Upper Saddle River, N.J: Prentice Hall, 2010.

Appendix A

Previous Model of the Prototype

Initially, a free body diagram of the system shown in Figure A.1 is drawn. The system has two degrees of freedom, one for the translational motion of the cart and one for the rotation of the pendulum.

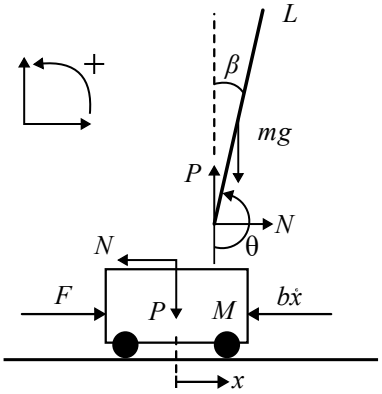


Figure A.1: Free body diagram of the cart with an inverted pendulum

Where F, b, g, l are the external force, viscous friction, gravitational force, and length of pin connection to the COM of the pendulum. θ is the pendulum angle from the vertical downwards position. *beta* is the pitch angle. The horizontal displacement of the cart is described by x. The constraint of the pin connecting the two bodies creates reaction forces between them; these are denoted as N and P for the x- and y-direction respectively.

Using Newton's second law the forces for the x- and y-direction of the cart are derived as (A.1).

$$\sum F_x = F - N - b\dot{x} = M\ddot{x} \tag{A.1a}$$

$$\sum F_y = P = 0 \tag{A.1b}$$

Determining the forces for the x- and y-direction, and the moment for the pendulum results in (A.2).

$$\sum F_x = N = m\ddot{x}_p \tag{A.2a}$$

$$\sum F_y = P - mg = m\ddot{y}_p \tag{A.2b}$$

$$\sum \tau = -Nl\cos(\theta) - Pl\sin(\theta) = \ddot{\theta}I \tag{A.2c}$$

Where subscript p denotes pendulum.

Using algebra N and P are solved for. Initially, the position of the pendulum's centre of mass is

determined:

$$x_p = x + l\sin(\theta) \tag{A.3}$$

$$y_p = -l\cos(\theta) \tag{A.4}$$

To determine the translational velocity and acceleration, the position is differentiated w.r.t. time:

$$\dot{x}_p = \dot{x} + l\cos(\theta)\dot{\theta} \tag{A.5a}$$

$$\dot{y}_p = l\sin(\theta)\dot{\theta} \tag{A.5b}$$

$$\ddot{x}_p = \ddot{x} + l\cos(\theta)\ddot{\theta} - l\sin(\theta)\dot{\theta}^2 \tag{A.5c}$$

$$\ddot{y}_p = l\sin(\theta)\ddot{\theta} + l\cos(\theta)\dot{\theta}^2 \tag{A.5d}$$

The reaction force N of the pendulum is determined by using (A.2) and (A.5):

$$N = m\ddot{x}_p \tag{A.6}$$

$$N = m\left(\ddot{x} + l\cos(\theta)\ddot{\theta} - l\sin(\theta)\dot{\theta}^2\right) \tag{A.7}$$

$$N = m\ddot{x} + ml\cos(\theta)\ddot{\theta} - ml\sin(\theta)\dot{\theta}^2 \tag{A.8}$$

The same procedure is performed for P:

$$P = m\ddot{y}_p + mg \tag{A.9}$$

$$P = m\left(l\sin(\theta)\ddot{\theta} + l\cos(\theta)\dot{\theta}^2\right) + mg \tag{A.10}$$

$$P = ml\sin(\theta)\ddot{\theta} + ml\cos(\theta)\dot{\theta}^2 + mg \tag{A.11}$$

Using (A.2) the equations of motion for the cart and pendulum are determined:

$$M\ddot{x} + b\dot{x} + N = F \tag{A.12}$$

$$\ddot{\theta}I = -Nl\cos(\theta) - Pl\sin(\theta) \tag{A.13}$$

By inserting the expressions for N and P, the equations of motion are rewritten. For the cart the equation of motion becomes:

$$M\ddot{x} + b\dot{x} + m\ddot{x} + ml\cos(\theta)\ddot{\theta} - ml\sin(\theta)\dot{\theta}^2 = F \tag{A.14a}$$

$$(M+m)\ddot{x} + b\dot{x} + ml\cos(\theta)\ddot{\theta} - ml\sin(\theta)\dot{\theta}^2 = F \tag{A.14b}$$

For the pendulum the equation of motion becomes:

$$\ddot{\theta}I = -\left(m\ddot{x} + ml\cos(\theta)\ddot{\theta} - ml\sin(\theta)\dot{\theta}^2\right) l\cos(\theta) - \left(ml\sin(\theta)\ddot{\theta} + ml\cos(\theta)\dot{\theta}^2 + mg\right) l\sin(\theta) \tag{A.15a}$$

$$\ddot{\theta}I = -m\ddot{x}l\cos(\theta) - \left(ml^2\cos^2(\theta) + ml^2 sin^2(\theta)\right)\ddot{\theta} + ml^2\sin(\theta)\cos(\theta)\dot{\theta}^2 \tag{A.15b}$$

$$- ml^2\cos(\theta)\sin(\theta)\dot{\theta}^2 - mgl\sin(\theta) \tag{A.15c}$$

$$\ddot{\theta}I = -m\ddot{x}l\cos(\theta) - \left(ml^2\cos^2(\theta) + ml^2 sin^2(\theta)\right)\ddot{\theta} - mgl\sin(\theta) \tag{A.15d}$$

By using the identity $\sin^2(\theta) + \cos^2(\theta) = 1$ and isolating $\ddot{x}$ on the right side of the equation results in:

$$(I + ml^2)\ddot{\theta} + mgl\sin(\theta) = -m\ddot{x}l\cos(\theta) \tag{A.16}$$

Then (A.14) and (A.16) are linearisered to get a linear system of equations. The equilibrium of interest is when the pendulum is in upward vertical position, therefore the system is linearised at $\theta = \pi$. Furthermore, it is desired to let the pitch angle represent the pendulum's position from the equilibrium, resulting in $\theta = \pi + \beta$. Using the small angle approximation, the nonlinearities in the system becomes:

$$\cos(\theta) = \cos(\beta + \pi) \approx -1 \qquad \sin(\theta) = \sin(\beta + \pi) \approx -\beta \qquad \dot{\theta}^2 = \dot{\beta}^2 \approx 0$$

Substituting these into the equations results in:

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\beta} = F \tag{A.17}$$

$$(I + ml^2)\ddot{\beta} - mgl\beta = m\ddot{x}l \tag{A.18}$$

Then the system can be arranged in the state space representation where F is assumed to be a control input u:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \beta \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+mL^2)b}{I(m+M)+MmL^2} & \frac{m^2gL^2}{I(m+M)+MmL^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mLb}{I(m+M)+MmL^2} & \frac{-bmgL(m+M)}{I(m+M)+MmL^2} & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \beta \\ \dot{\beta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+mL^2}{I(m+M)+Mm^2} \\ 0 \\ \frac{mL}{I(m+M)+Mm^2} \end{bmatrix} u \tag{A.19a}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \beta \\ \dot{\beta} \end{bmatrix} \tag{A.19b}$$

where only the position of the cart and pitch angle is feedback. The transfer functions of the system is then determined using the following expression [58]:

$$G(s) = C(sI - A)^{-1}B \tag{A.20}$$

Appendix B

Inertial Measurement Unit

In this appendix, the theory behind an Inertial Measurement Unit (IMU) is presented. The idea behind an IMU is to convert the sensors' inertia into a dataset which describes the motion or orientation of the sensor [65]. For a 9-axis IMU, three different sensors are available: an accelerometer, a gyroscope, and a magnetometer is available, with each providing information in three dimensions. The information provided by each sensor can then be used individually or fused to determine the units orientation in three dimensions. Below the workings of the three different sensors and their contribution to three dimensional orientation is described.

In Figure B.1 a sketch of an accelerometer is illustrated. The purpose of an accelerometer is to obtain the sensor's accelerations, but these are not measured directly by the sensor. The sensor works by suspending a mass in a set of springs presented as the "Movable Part" and the springs in Figure B.1. Secondly, a set of conductive parts are fixed around the mass, these parts are depicted as "Fixed Part for X&Y" and "Fixed Part for Z".

Based on the distance between the movable and fixed part, it is possible to calculate the capacitance, given by the formula:

$$C = \frac{\varepsilon A}{d} \tag{B.1}$$

Where C is capacitance, ε is the dielectric permittivity of the material, A is the parallel plate area, and d is the distance between two conductive plates.
As the sensor moves, the fixed parts move with it, but due to inertia the movable part opposes the movement, and thereby cause a displacement of the springs, and change the distance between the movable and fixed parts. Thereby a change in capacitance is related to a change in spring deflection, given by:

$$F = kdx \tag{B.2}$$

Where F is force, k is the spring constant, and dx is the change in spring length. This is related to accelerations through the formula:

$$F = ma \tag{B.3}$$

Where m is the mass of the movable part and a is acceleration. This relation is then utilised in the x, y and z direction to obtain accelerations in three directions. Furthermore, the gravitational acceleration of the accelerometer has to be factored into the calculations, as it influences the values in the direction that points upwards. Based on the calculated accelerations it is possible to obtain the orientation of the sensor. In Figure B.2 illustrates how the pitch β is approximated based on the calculated accelerations.

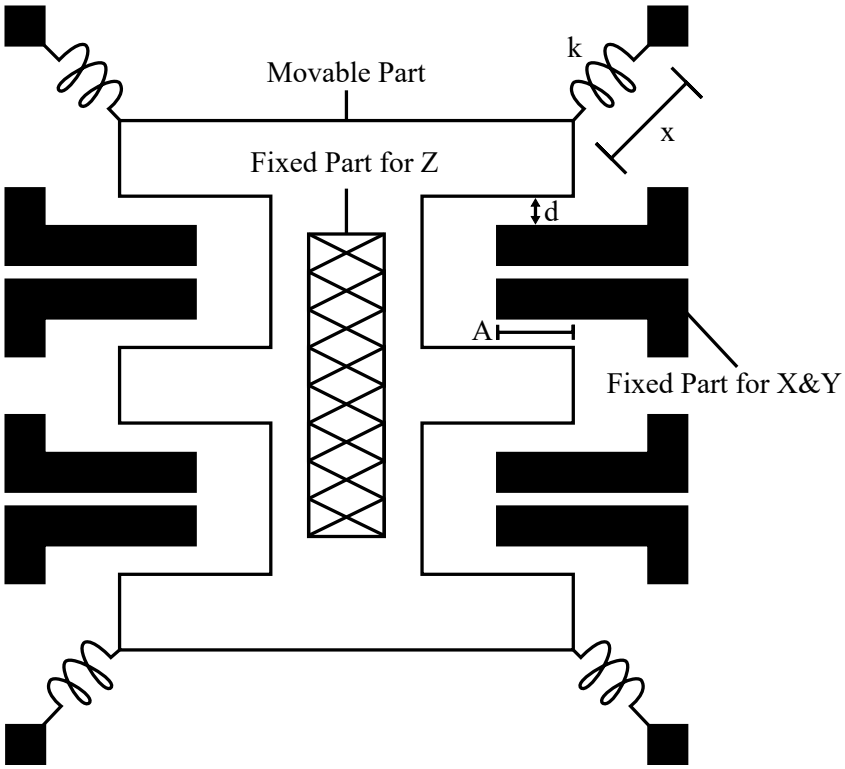


Figure B.1: Illustration of the working principle of an accelerometer, (k: Spring constant, d: distance between movable part and fixed part, A: Overlapping surface area of movable part and fixed part).

From the situation depicted in Figure B.2 the pitch of the IMU is approximated with a tangent function:

$$\beta = \text{atan}\left(\frac{a_x}{a_z}\right) \tag{B.4}$$

The tangent approximation works well within +/- 45 degrees as shown in Figure B.3, but other methods are needed beyond this point as is explained later. Similar calculations can be made for roll, but not for yaw as it is measured perpendicular to the gravitational acceleration, which makes this approach unusable for calculation of the yaw.

The problem with calculating the orientation of the sensor based on the accelerometer alone is that high frequency vibrations in the system results in a force on the accelerometer and thereby a change in angle. In order to circumvent this problem, a low pass filter is implemented, as it filters out the high frequency vibrations, which is present due to e.g. surrounding motors. The discrete low pass filter is calculated using the following:

$$\beta_{filtered} = \beta_{measured} p_1 + \beta_{old} p_2 \tag{B.5}$$

$$1 = p_1 + p_2 \tag{B.6}$$

Where $\beta_{filtered}$ represents the pitch angle with the low pass filter applied, $\beta_{measured}$ is the measured pitch angle, β_{old} is the previously measured pitch angle, and $p_{1,2}$ represent filter constants which determine the amount of filtration.

The downside to the low pass filter is a reduced response time of the system, which is undesirable when

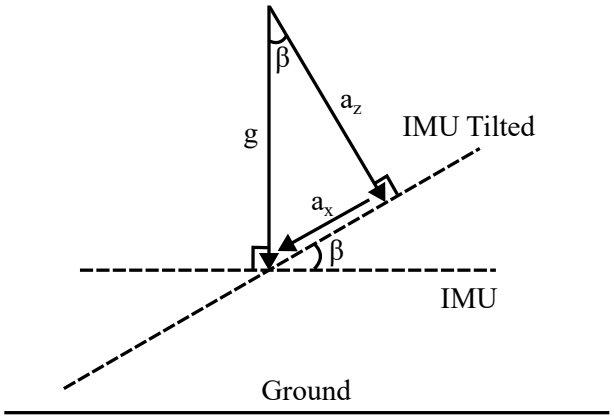


Figure B.2: Illustration of IMU pitch approximation calculation, (a_z: Acceleration in z direction, a_x: Acceleration in x direction, g: Gravitational acceleration, β: pitch angle).

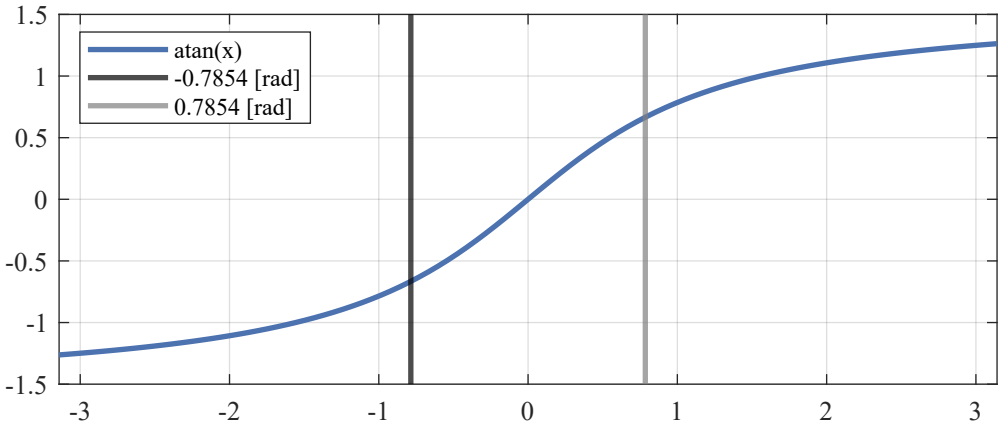


Figure B.3: Graph of atan function from -π to π, with vertical lines at +/- 45°

dealing with MWIPs as they rely on fast response times in order to keep upright. In order to reduce the effects of vibrations without sacrificing response time, the accelerometer data is fused with the data from the gyroscope.

In Figure B.4 a sketch of a gyroscope is illustrated. A gyroscope works by suspending three discs (Rotor, Gimbal, and Spin Axis in Figure B.4), from a frame. These discs are then allowed to spin based on gravitational forces, and surrounding rotations. The Gimbal is the outer ring which when turned rotates the inner rings alongside it. The intermediate axis is the Spin axis which can rotate the Rotor, but not the Gimbal, and lastly the Rotor only rotates itself. From this configuration a phenomenon known as Gimbal lock can occur, where the Spin axis orientates the Rotor and Gimbal in the same plane which in turn reduces the gyroscopes degrees of freedom from three to two. This phenomenon is discussed later. For the gyroscope to measure angular velocities a set of mechanisms are placed within the discs. These mechanisms can be e.g. micro-electromechanical systems. Micro-electromechanical systems works by resonating a small mass proportionally to the angular velocity using the Coriolis effect as illustrated in Figure B.5 [66]. When the small masses resonate an electrical signal proportional to the angular velocity is produced. As a result of this setup gyroscopes are less susceptible to vibrations compared to accelerometers [67].

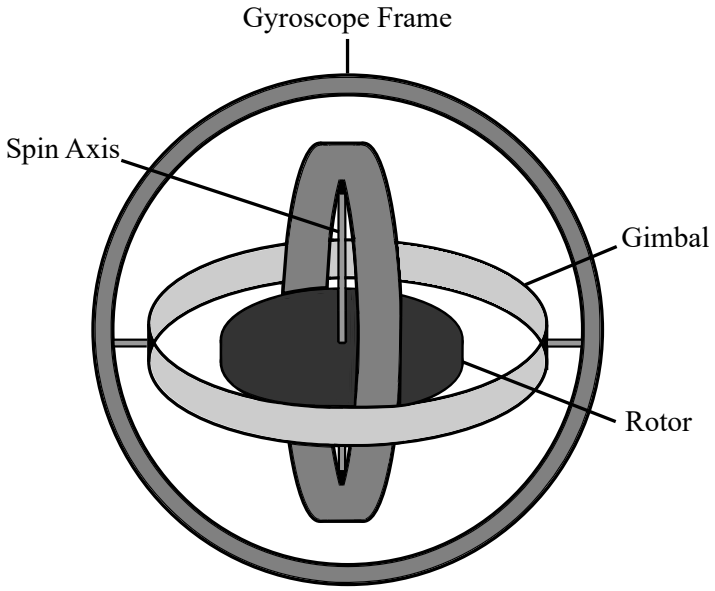


Figure B.4: Illustration of a gyroscope with name convention of different components.

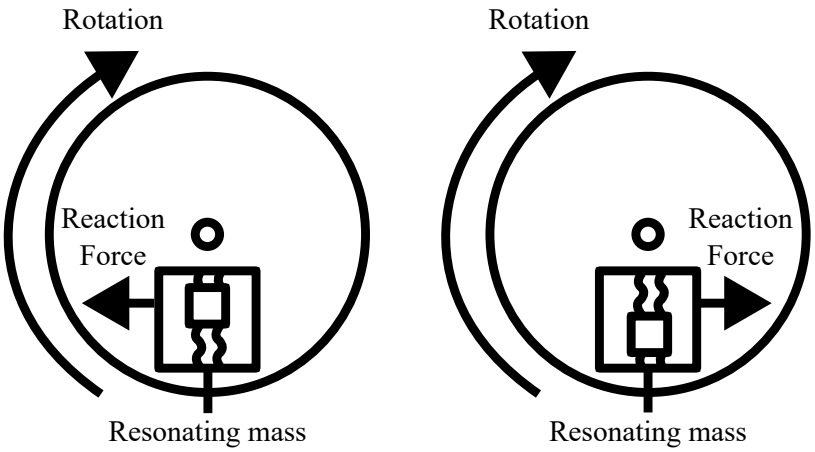


Figure B.5: Sketch of the working principle of a micro-electromechanical system gyroscope.

From the gyroscope the pitch is calculated as:

$$\beta_G = \beta_{Gold} + \omega_{Gy} dt \tag{B.7}$$

Where β_G is the pitch calculated by the gyroscope, β_{Gold} is the previously calculated pitch, dt is an increment in time, and ω_{Gy} is the angular velocity measured by the gyroscope.

The problem with the gyroscope measurement is that it is prone to drifting as any non zero value, either from noise or DC offset, contributes to a continuous drift of the calculated angle. As a result it is preferable to remove the low frequency components of the gyroscope, in order to filter out the drift. Thereby the short term gyroscope measurements is more accurate than the accelerometer measurements, but the accelerometer performs better on long term measurements. In order to fuse these signals to obtain a measurement which is trustworthy at both low and high frequencies, a

complimentary filter is implemented:

$$\beta = (\beta_{old} + \omega_{Gy})\, p_1 + \beta_A p_2 \tag{B.8}$$

$$1 = p_1 + p_2 \tag{B.9}$$

$$p_1 > p_2 \tag{B.10}$$

The complimentary filter works by including a high pass filter on the gyroscope readings, and a low pass filter on the accelerometer readings. This hereby solves the vibration and drift problem.

The next problem is that yaw needs to be calculated. The gyroscope can measure yaw, but the reading is prone to drifting, and as the accelerometer cannot measure yaw, it cannot be fused as is the case for pitch and roll.
To solve this issue a magnetometer is included. In Figure B.6, the working principle of a magnetometer is illustrated. A magnetometer works by calculating the angle between the magnetic field of the

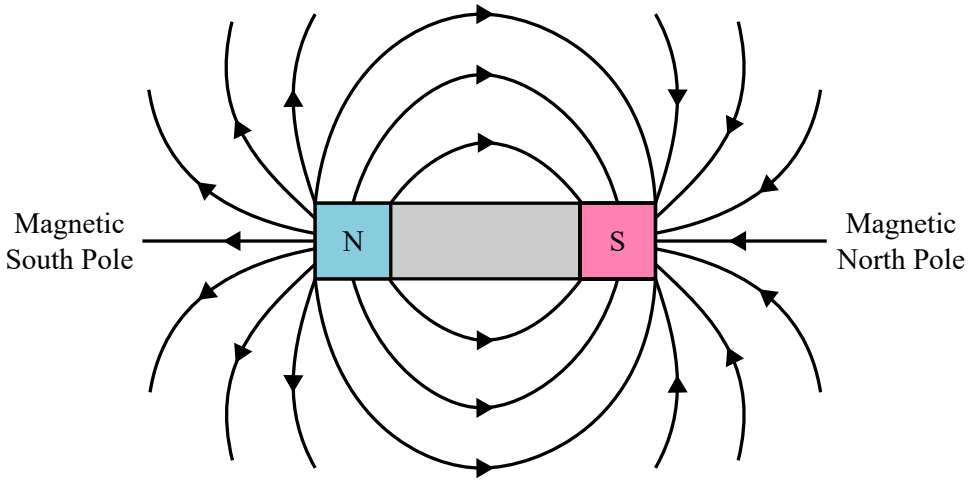


Figure B.6: Illustration of the working principle of a magnetometer, (*N*: north pole of the magnet, *S*: south pole of the magnet).

magnetometer and the earth's magnetic north pole. From this consideration the yaw of the IMU is calculated as depicted in Figure B.7. This calculation is done with the tangents approximation as is done for both the accelerometer and the gyroscope. Thereby roll, pitch and yaw can be calculated, but it is advantageous to expand the calculations and make the IMU tilt compensated. This is advantageous as when the IMU is pitched and then yawed compared to the ground the output of the IMU is a combination of roll, pitch, and yaw as it is in a local coordinate frame. Therefore by making it tilt compensated, the IMU output is related to the global reference frame.

This is done with Euler angles as explained in [68]:

$$C_\gamma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{B.11}$$

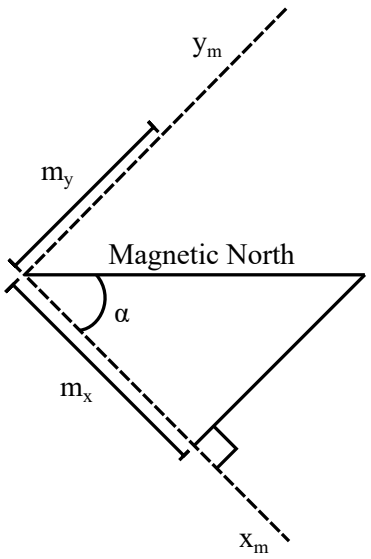


Figure B.7: Illustration of IMU yaw approximation calculation, (m_x: magnetic value in x direction, m_y: magnetic value in y direction, α: yaw angle in relation to magnetic north).

Where C_γ represents rotation matrix for roll, and γ is the roll angle.

$$C_\beta = \begin{bmatrix} \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{B.12}$$

Where C_β is the rotation matrix for pitch, and β is the pitch angle.

These rotation matrices can be combined to obtain:

$$x_m = m_x \cos(\gamma) + m_y \sin(\beta)\sin(\gamma) + m_z \cos(\beta)\sin(\gamma) \tag{B.13}$$

$$y_m = m_y \cos(\beta) - m_z \sin(\beta) \tag{B.14}$$

Where x_m is the orientation of the x-axis for the magnetometer, y_m is the orientation of the y-axis for the magnetometer, and $m_{x,y,z}$ is the measured magnetic value in the x, y, and z direction.

When rotating with Euler angles the order of rotation is of great significance as the rotations are not commutative [69]. As a result of this three operations have to be carried out depending on the chosen scheme. Here either proper Euler angles with an a-b-a sequence or Tait-Bryan Euler angles with an a-b-c sequence can be used. With a-b-a sequence referring to rotation sequences where the first and last rotation is done around the same axis with the middle being about a second axis. When using Euler angles Gimbal lock can be encountered as explained in the gyroscope section. In order to avoid Gimbal lock it is necessary to change the rotation sequence or use quaternions [70].

The use of quaternions furthermore circumvents the use of tangent approximations [71]. [71] is used to introduce the concept of quaternions. According to [71], a quaternion is defined as:

$$q = q_0 + \mathbf{q} \tag{B.15}$$

Where q_0 is a real number (scalar), and $\mathbf{q}$ is an ordinary vector in R^3, given by:

$$\mathbf{q} = iq_1 + jq_2 + kq_3 \tag{B.16}$$

With i, j, k denoting the orthonormal basis of R^3 given as:

$$i = (1,0,0) \tag{B.17}$$

$$j = (0,1,0) \tag{B.18}$$

$$k = (0,0,1) \tag{B.19}$$

As a result of this a quaternion is a sum of a scalar and a vector and an element of R^4, which is not defined in ordinary linear algebra.

In order to define quaternions William Rowan Hamilton defined a set of fundamental products which must be satisfied when taking the product of two quaternions:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{B.20}$$

$$ij = k = -ji \tag{B.21}$$

$$jk = i = -kj \tag{B.22}$$

$$ki = j = -ik \tag{B.23}$$

As a result of this the product of two quaternions is not commutative.

For any unit quaternion:

$$q = q_0 + \mathbf{q} = \cos(\theta) + \mathbf{u}\sin(\theta) \tag{B.24}$$

With $\mathbf{u}$ representing the unit vector which represents the direction of the quaternion q, calculated by:

$$\mathbf{u} = \frac{q}{|q|} = \frac{q}{\sin(\theta)} \tag{B.25}$$

The quaternion rotation operator, $L_q(v)$, is used to describe rotations of a vector, v, in R^3 by using a associated quaternion q. This is done by defining the vector, v, as:

$$v = 0 + \mathbf{v} \tag{B.26}$$

With this definition the quaternion rotation operator is calculated as:

$$L_q(\mathbf{v}) = q\mathbf{v}q^* \tag{B.27}$$

With the complex conjugate q^* given as:

$$q^* = q_0 - \mathbf{q} = \cos(\theta) - u\sin(\theta) \tag{B.28}$$

Which is expanded to:

$$L_q(v) = (q_0^2 - |\mathbf{q}|^2)\mathbf{v} + 2(\mathbf{q}\cdot\mathbf{v})\mathbf{q} + 2q_0(\mathbf{q}\times\mathbf{v}) \tag{B.29}$$

This rotation operator represents a rotation in R^3 with the axis of rotation given by **q**, and the angle of rotation is twice the angle associated with q. To illustrate this rotation the cases in Figures B.8 and B.9 is considered. Figure B.8 represents the rotation operators components, and Figure B.9 represents the rotation operators geometry.

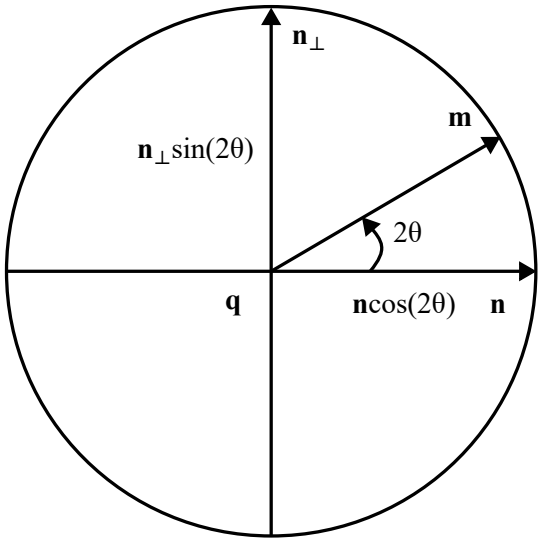


Figure B.8: Rotated vector components.

Where the vector v is given by:

$$\mathbf{v} = \mathbf{a} + \mathbf{n} \tag{B.30}$$

With **a** describing the component of v along the vector part of q, and **n** the component of v normal to the vector part of q.

Where as the vector **a** lies along the vector **q** its rotation operator is simply a scalar multiple of **q**:

$$L_q(a) = k\mathbf{q} = a \tag{B.31}$$

Using **n** the rotation operator is described as:

$$L_q(\mathbf{n}) = (q_0^2 - |\mathbf{q}|^2)\mathbf{n} + 2(\mathbf{q}\cdot\mathbf{n})\mathbf{q} + 2q_0(\mathbf{q}\times\mathbf{n}) \tag{B.32}$$

Using the relation in (B.25) it is simplified to:

$$L_q(\mathbf{n}) = \cos(2\theta\mathbf{n}) + \sin(2\theta\mathbf{n}_\perp) \tag{B.33}$$

Thereby L_q(**a+n**) describes a rotation of **n** by 2θ up to the vector **m** as shown in Figure B.9.

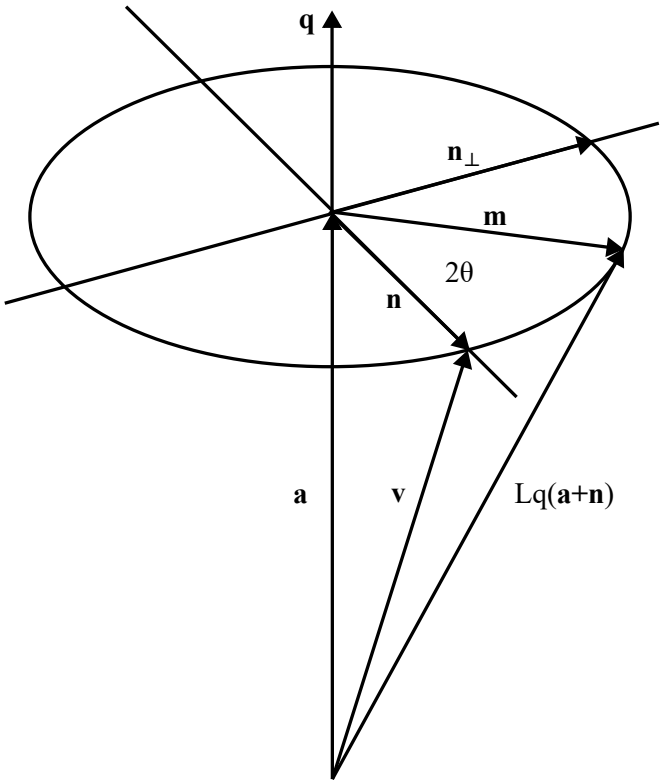


Figure B.9: Rotation operator geometry.

From this, quaternions is used to describe either vector rotations or frame rotations, where:

$$qvq^* \tag{B.34}$$

Represents a rotation of the vector v with respect to a fixed coordinate frame, and:

$$q^*vq \tag{B.35}$$

Represents a rotation of the coordinate frame with respect to the vector v.

Based on these quaternions it is possible to circumvent the problems encountered with Euler angles, and obtain a tilt compensated IMU.

Appendix C

Python Code to Communicate with ODrive

```
import odrive
import time
import sys
import math

odrv0 = odrive.find_any()
print("Found Odrive")

try:
    print("Erasing configurations")
    odrv0.erase_configuration()
except:
    pass
odrv0 = odrive.find_any()

odrv0.config.enable_brake_resistor = True
print("Enabled brake resistor")

odrv0.config.dc_max_negative_current = -0.001
print("Set DC max negative current")

odrv0.axis0.motor.config.pole_pairs = 15
odrv0.axis1.motor.config.pole_pairs = 15
print("Configured number of pole pairs")

odrv0.axis0.motor.config.resistance_calib_max_voltage = 4
odrv0.axis1.motor.config.resistance_calib_max_voltage = 4
print("Set resistance calibrated max voltage")

odrv0.axis0.motor.config.requested_current_range = 25
odrv0.axis1.motor.config.requested_current_range = 25
print("Set current range")

odrv0.axis0.motor.config.current_control_bandwidth = 100
odrv0.axis1.motor.config.current_control_bandwidth = 100
print("Set current control bandwidth")

odrv0.axis0.motor.config.torque_constant = 0.31 # 8.27 / 13.63
odrv0.axis1.motor.config.torque_constant = 0.31 # 8.27 / 13.63
print("Defined torque constant")

odrv0.axis0.encoder.config.mode = 1 # ENCODER_MODE_HALL
odrv0.axis1.encoder.config.mode = 1 # ENCODER_MODE_HALL
print("Entered config mode")

odrv0.axis0.encoder.config.cpr = 90
odrv0.axis1.encoder.config.cpr = 90
print("Set CPR - count pr rotation")

odrv0.axis0.encoder.config.calib_scan_distance = 150
```

```
odrv0.axis1.encoder.config.calib_scan_distance = 150
print("Set scan distance")

odrv0.config.gpio9_mode  = 0 # GPIO_MODE_DIGITAL
odrv0.config.gpio10_mode = 0 # GPIO_MODE_DIGITAL
odrv0.config.gpio11_mode = 0 # GPIO_MODE_DIGITAL
print("Set GPIO mode")

odrv0.axis0.encoder.config.bandwidth = 100
odrv0.axis1.encoder.config.bandwidth = 100
print("Set config bandwidth")

odrv0.axis0.controller.config.enable_torque_mode_vel_limit = True
odrv0.axis1.controller.config.enable_torque_mode_vel_limit = True
odrv0.axis0.controller.config.enable_vel_limit = True
odrv0.axis1.controller.config.enable_vel_limit = True
odrv0.axis0.controller.config.vel_limit = 80/(2*math.pi)
odrv0.axis1.controller.config.vel_limit = 80/(2*math.pi)
print("Set velocity limit")

odrv0.axis0.controller.config.control_mode = 1 # TORQUE_CONTROL
odrv0.axis1.controller.config.control_mode = 1 # TORQUE_CONTROL
print("Set control mode - Torque control")

# Save configuration and proceed
try:
    odrv0.save_configuration()
except:
    pass

odrv0 = odrive.find_any()
try:
    odrv0.reboot()
except:
    pass

odrv0 = odrive.find_any()

print("Starting motor calibration of axis0")
odrv0.axis0.requested_state = 3 # AXIS_STATE_MOTOR_CALIBRATION
print("Sleeping for 45 seconds")
time.sleep(45)

print("Starting motor calibration of axis1")
odrv0.axis1.requested_state = 3 # AXIS_STATE_MOTOR_CALIBRATION
time.sleep(45)
print("Sleeping for 45 seconds")

odrv0.axis0.motor.config.pre_calibrated = True
odrv0.axis1.motor.config.pre_calibrated = True
print("Set motor to pre calibrated")

print("Starting encoder hall polarity calibration of axis0")
odrv0.axis0.requested_state = 12 # AXIS_STATE_ENCODER_HALL_POLARITY_CALIBRATION
print("Sleeping for 15 seconds")
time.sleep(15)
print("Starting encoder hall polarity calibration of axis1")
odrv0.axis1.requested_state = 12 # AXIS_STATE_ENCODER_HALL_POLARITY_CALIBRATION
print("Sleeping for 15 seconds")
time.sleep(15)
```

```
111
112 odrv0.axis0.encoder.config.pre_calibrated = True
113 odrv0.axis1.encoder.config.pre_calibrated = True
114 print("Set encoder to pre calibrated")
115
116 odrv0.config.uart_a_baudrate = 115200
117 print("Set baudrate")
118
119 odrv0.config.enable_uart_a = True
120 print("Enable uart-a")
121
122 odrv0.config.gpio1_mode = 4 # GpioMode.UART_A
123 odrv0.config.gpio2_mode = 4 # GpioMode.UART_A
124 print("Set GPIO 1 and GPIO 2 mode")
125
126 odrv0.axis0.config.startup_closed_loop_control = True
127 odrv0.axis1.config.startup_closed_loop_control = True
128 print("Set startup closed loop control")
129
130 # Save configuration and proceed
131 try:
132     odrv0.save_configuration()
133 except:
134     pass
135
136 odrv0 = odrive.find_any()
137
138 print("Phase inductance")
139 print(odrv0.axis0.motor.config.phase_inductance)
140 print(odrv0.axis1.motor.config.phase_inductance)
141
142 print("Phase resistance")
143 print(odrv0.axis0.motor.config.phase_resistance)
144 print(odrv0.axis1.motor.config.phase_resistance)
```

Appendix D

Investigation of Minimum

To investigate the behaviour of the CM, (4.4) and (4.5) are combined resulting in (D.1).

$$L_{ACMT} = \frac{\rho_P L_P^2 wd}{2(\rho_P L_P wd + m_c)} + \frac{m_c h}{2(\rho_P L_P wd + m_c)} + 0.114 \tag{D.1}$$

Only the first two terms are of interest, as the last term simply offsets the y-axis. The first term is the contribution from the paint while the second term is from the tank. These are seen plotted in Figure D.1. The first term is increasing with respect to the paint level nearly at a linear rate, only offset by the mass

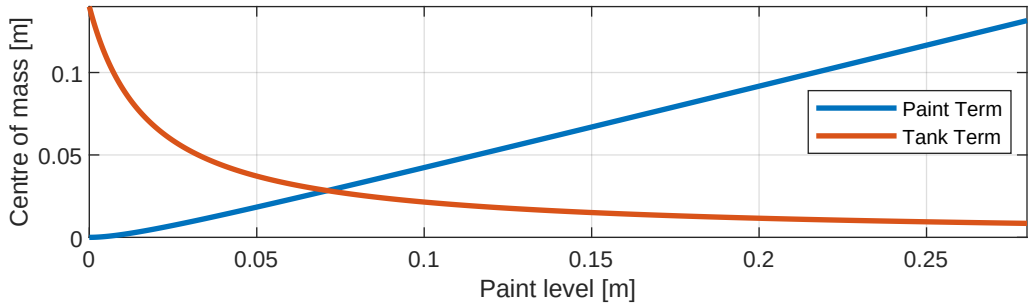


Figure D.1: Contribution of the two first terms in the calculation of CM of the tank stated in (D.1).

of the container. The second term has an inverse relation to the paint level. Their rate of change differ, resulting in the CM achieving a minimum at 0.055 m.

Appendix E

Models of the System

E.1 Simulation of Mechanics and Electronics

```
clc; clear; close all
addpath("Functions")

% Dimensions [m]
L_ACMW = 0.243620;

% Wheel radius [m]
r = 0.1651/2;

% Pole pairs
PP = 15;

% Stator resistance [Ohm]
Rs = 0.2463109940290451;

% Inductance [Henry]
La = 0.000408201856771484;

% Permanent magnet flux linkage
lambda = 0.014009204;

% Current controller bandwidth
w_cc = 100;

% Current controller gains
K_P = La*w_cc;
K_I = Rs*w_cc;

% DC-voltage source
Vdc = 28;

% Maximum modulation index
M = 0.70;

%% Simulation

% Time
t0 = 0;
dt = 1e-3;
t1 = 12;

t = t0:dt:t1-dt;

% Data size
N = length(t);

% Preallocating memory
```

```
48    alpha_1dd = zeros(size(t));
49    alpha_1d  = zeros(size(t));
50    alpha_1   = zeros(size(t));
51     beta_1dd = zeros(size(t));
52     beta_1d  = zeros(size(t));
53     beta_1   = zeros(size(t));
54         s    = zeros(size(t));
55         sd   = zeros(size(t));
56         sdd  = zeros(size(t));
57          x   = zeros(size(t));
58          xd  = zeros(size(t));
59          y   = zeros(size(t));
60          yd  = zeros(size(t));
61            K = zeros(size(t));
62            P = zeros(size(t));
63            T = zeros(size(t));
64          w_1 = zeros(size(t));
65          w_2 = zeros(size(t));
66      theta_1 = zeros(size(t));
67      theta_2 = zeros(size(t));
68     theta_1e = zeros(size(t));
69     theta_2e = zeros(size(t));
70          T_T = zeros(2,length(t));
71        V_dq1 = zeros(2,length(t));
72        V_dq2 = zeros(2,length(t));
73    Valphbet1 = zeros(2,length(t));
74    Valphbet2 = zeros(2,length(t));
75           a1 = zeros(size(t));
76           a2 = zeros(size(t));
77       Id_dq1 = zeros(2,length(t));
78       Id_dq2 = zeros(2,length(t));
79        I_dq1 = zeros(2,length(t));
80        I_dq2 = zeros(2,length(t));
81       I_abc1 = zeros(3,length(t));
82       I_abc2 = zeros(3,length(t));
83       V_abc1 = zeros(3,length(t));
84       V_abc2 = zeros(3,length(t));
85 I_error_int1 = zeros(2,length(t));
86 I_error_int2 = zeros(2,length(t));
87       I_ref1 = zeros(2,length(t));
88       I_ref2 = zeros(2,length(t));
89     I_error1 = zeros(2,length(t));
90     I_error2 = zeros(2,length(t));
91          T_1 = zeros(size(t));
92          T_2 = zeros(size(t));
93        xdhat = zeros(5,N);
94         xhat = zeros(5,N);
95
96 % Paint height
97 L = 0.28;
98
99 % Initial conditions
100 beta_1(1) = 0;
101 alpha_1(1) = 0;
102
103 x(1) = 0;
104 y(1) = 0;
105
106 beta_1d(1) = 0;
107 alpha_1d(1) = 0;
```

```
sd(1) = 0;

K(1) = Kf(L,alpha_1(1),alpha_1d(1),beta_1(1),beta_1d(1),sd(1));
P(1) = Pf(L,beta_1(1));
T(1) = K(1) + P(1);

% Controller
K_LQR = [20.0000   11.5336   -0.0000   -0.0000   -0.0000
          0.0000    0.0000  -64.8638  -14.9927   -4.9930];

% Reference
p = 1/4;
% alpha_1_ref = pi/4*square(2*pi*p/2*t) + pi/4;
%      sd_ref = zeros(size(t));
alpha_1_ref = zeros(size(t));
     sd_ref = square(2*pi*p/2*t);

% System matrix
A = [0,1.0000,      0,      0,      0;
     0,0.0022,      0,      0,      0;
     0,     0,      0, 1.0000,      0;
     0,     0,21.4652, 0.0008,-0.0018;
     0,     0,-3.9643,-0.0001, 0.0012];

% Input matrix
B = [     0,      0;
     1.2172,      0;
          0,      0;
          0,-1.0182;
          0, 0.6538];

% Output matrix
C = eye(5);


startLoop = tic;
for n = 2:N

    if mod(n,N/10) == 0
        clc
        fprintf(' Runtime: %.1f s \n     ETA: %.1f s \nProgress: %.0f %% \n', toc(
    startLoop), toc(startLoop)/(n/N) - toc(startLoop) , n/N*100)
    end

    %%% CONTROLLER %%%

    % Reference virtual input - Linear Quadratic regulator
    T_T(:,n) =  -K_LQR*[alpha_1(n-1) - alpha_1_ref(n);alpha_1d(n-1);beta_1(n-1);beta_1d(n
    -1);sd(n-1) - sd_ref(n)];

    % Current references
    I_ref1(:,n) = [0;(T_T(1,n)/2 + T_T(2,n)/2)/(2/3*PP*lambda)];
    I_ref2(:,n) = [0;(T_T(2,n)/2 - T_T(1,n)/2)/(2/3*PP*lambda)];

    % Current errors
    I_error1(:,n) = I_ref1(:,n) - I_dq1(:,n-1);
    I_error2(:,n) = I_ref2(:,n) - I_dq2(:,n-1);

    % Integral of error
    I_error_int1(:,n) = [RK4(I_error_int1(1,n-1),I_error1(1,n),dt);
```

```
                        RK4(I_error_int1(2,n-1),I_error1(2,n),dt)];
    I_error_int2(:,n) = [RK4(I_error_int2(1,n-1),I_error2(1,n),dt);
                        RK4(I_error_int2(2,n-1),I_error2(2,n),dt)];

    % PID Controller
    V_dq1(:,n) = K_P*I_error1(:,n) + K_I*I_error_int1(:,n) + [-PP*w_1(n-1)*I_dq1(2,n-1)*
    La;PP*w_1(n-1)*I_dq1(1,n-1)*La + PP*lambda*w_1(n-1)];
    V_dq2(:,n) = K_P*I_error2(:,n) + K_I*I_error_int2(:,n) + [-PP*w_2(n-1)*I_dq2(2,n-1)*
    La;PP*w_2(n-1)*I_dq2(1,n-1)*La + PP*lambda*w_2(n-1)];

    % Inverse Park transform
    Valphbet1(:,n) = rot(theta_1e(n-1))*V_dq1(:,n);
    Valphbet2(:,n) = rot(theta_2e(n-1))*V_dq2(:,n);

    % Limiting
    a1(n) = sqrt(Valphbet1(1,n)^2 + Valphbet1(2,n)^2)/(2/3*Vdc);
    a2(n) = sqrt(Valphbet2(1,n)^2 + Valphbet2(2,n)^2)/(2/3*Vdc);

    if a1(n) > M
        a1(n) = M;
        Valphbet1(:,n) = Valphbet1(:,n)*M/a1(n);
    end
    if a2(n) > M
        a2(n) = M;
        Valphbet2(:,n) = Valphbet2(:,n)*M/a2(n);
    end

    % Returning to dq reference frame
    V_dq1(:,n) = [ cos(theta_1e(n-1)),sin(theta_1e(n-1));
                  -sin(theta_1e(n-1)),cos(theta_1e(n-1))]*Valphbet1(:,n);
    V_dq2(:,n) = [ cos(theta_2e(n-1)),sin(theta_2e(n-1));
                  -sin(theta_2e(n-1)),cos(theta_2e(n-1))]*Valphbet2(:,n);

    %%% PLANT %%%

    % Change in current dq-reference frame
    Id_dq1(:,n) = [V_dq1(1,n)/La + PP*w_1(n-1)*I_dq1(2,n-1) - Rs*I_dq1(1,n-1)/La;
                   V_dq1(2,n)/La - PP*w_1(n-1)*I_dq1(1,n-1) - Rs*I_dq1(2,n-1)/La  - PP*
    w_1(n-1)*lambda/La];
    Id_dq2(:,n) = [V_dq2(1,n)/La + PP*w_2(n-1)*I_dq2(2,n-1) - Rs*I_dq2(1,n-1)/La;
                   V_dq2(2,n)/La - PP*w_2(n-1)*I_dq2(1,n-1) - Rs*I_dq2(2,n-1)/La  - PP*
    w_2(n-1)*lambda/La];

    % Current dq-reference frame
    I_dq1(:,n) = [RK4(I_dq1(1,n-1),Id_dq1(1,n),dt);
                  RK4(I_dq1(2,n-1),Id_dq1(2,n),dt)];
    % Current dq-reference frame
    I_dq2(:,n) = [RK4(I_dq2(1,n-1),Id_dq2(1,n),dt);
                  RK4(I_dq2(2,n-1),Id_dq2(2,n),dt)];

    % Torque
    T_1(n) = 2/3*PP*lambda*I_dq1(2,n);
    T_2(n) = 2/3*PP*lambda*I_dq2(2,n);

    % Acceleration
    beta_1dd(n) = beta_1ddf(L,T_1(n),T_2(n),alpha_1d(n-1),beta_1(n-1),beta_1d(n-1),sd(n
    -1));

    % Numerical Integration
    beta_1d(n) = RK4(beta_1d(n-1),beta_1dd(n),dt);
```

```
221     beta_1 (n) = RK4(beta_1(n-1) ,beta_1d(n) ,dt);
222
223     % Acceleration
224     alpha_1dd(n) = alpha_1ddf(L,T_1(n),T_2(n),alpha_1d(n-1),beta_1(n),beta_1d(n));
225
226     % Numerical Integration
227     alpha_1d(n) = RK4(alpha_1d(n-1),alpha_1dd(n),dt);
228     alpha_1 (n) = RK4(alpha_1(n-1) ,alpha_1d(n) ,dt);
229
230     % Acceleration
231     sdd(n) = sddf(L,T_1(n),T_2(n),alpha_1d(n),beta_1(n),beta_1d(n),sd(n-1));
232
233     % Numerical Integration
234     sd(n) = RK4(sd(n-1),sdd(n),dt);
235     s(n)  = RK4( s(n-1), sd(n),dt);
236
237     % Global velocity
238     xd(n) = cos(alpha_1(n))*sd(n);
239     yd(n) = sin(alpha_1(n))*sd(n);
240
241     % Global position
242     x(n) = RK4(x(n-1), xd(n),dt);
243     y(n) = RK4(y(n-1), yd(n),dt);
244
245     % Wheel speed
246     w_1(n) = sd(n)/r - alpha_1d(n)*L_ACMW/r;
247     w_2(n) = sd(n)/r + alpha_1d(n)*L_ACMW/r;
248
249     % Wheel position
250     theta_1(n) = RK4(theta_1(n-1), w_1(n),dt);
251     theta_2(n) = RK4(theta_2(n-1), w_2(n),dt);
252
253     % Eletrical angle
254     theta_1e(n) = wrapTo2Pi(theta_1(n)*PP);
255     theta_2e(n) = wrapTo2Pi(theta_2(n)*PP);
256
257     % Kinetic energy
258     K(n) = Kf(L,alpha_1(n),alpha_1d(n),beta_1(n),beta_1d(n),sd(n));
259
260     % Potential energy
261     P(n) = Pf(L,beta_1(n));
262
263     % Total energy
264     T(n) = K(n) + P(n);
265
266 end
267 endLoop = toc(startLoop);
268
269 %% Linear model simulation
270
271 % Preallocating memory
272 X = zeros(5,length(t));
273 Xd = zeros(5,length(t));
274 Y = zeros(5,length(t));
275 U = zeros(2,length(t));
276
277 startLoop = tic;
278 for n = 2:N
279
280     if mod(n,N/10) == 0
```

```
        clc
        fprintf(' Runtime: %.1f s \n     ETA: %.1f s \nProgress: %.0f %% \n', toc(
    startLoop), toc(startLoop)/(n/N) - toc(startLoop) , n/N*100)
    end

    % Linear Quadratic Regulator
    U(:,n) = -K_LQR*(X(:,n-1) - [alpha_1_ref(n);0;0;0;sd_ref(n)]);

    % Change in states
    Xd(:,n) = A*X(:,n-1) + B*U(:,n);

    % States
    X(:,n) = [RK4(X(1,n-1),Xd(1,n),dt);
              RK4(X(2,n-1),Xd(2,n),dt);
              RK4(X(3,n-1),Xd(3,n),dt);
              RK4(X(4,n-1),Xd(4,n),dt);
              RK4(X(5,n-1),Xd(5,n),dt)];

    % Output
    Y(:,n) = C*X(:,n);
end

%% States
close all

figure()
tiledlayout(3,2)

nexttile
SetFigure(0,0)
hold on
plot(t,alpha_1d)
plot(t,X(2,:))
ylabel('Change in yaw [rad/s]')
xlabel('Time [s]')
legend('Nonlinear model','Linear model')
hold off

nexttile
SetFigure(0,0)
hold on
plot(t,alpha_1)
plot(t,X(1,:))
ylabel('Yaw [rad]')
xlabel('Time [s]')
hold off

nexttile
SetFigure(0,0)
hold on
plot(t,beta_1d)
plot(t,X(4,:))
ylabel('Change in pitch [rad/s]')
xlabel('Time [s]')
hold off

nexttile
SetFigure(0,0)
hold on
plot(t,beta_1)
```

```
plot(t,X(3,:))
ylabel('Pitch [rad]')
xlabel('Time [s]')
hold off

nexttile
SetFigure(0,0)
hold on
plot(t,sd)
plot(t,X(5,:))
ylabel('Speed [m/s]')
xlabel('Time [s]')
hold off

nexttile
SetFigure(0,0)
hold on
plot(t,cumtrapz(sd)*dt)
plot(t,cumtrapz(X(5,:))*dt)
ylabel('Distance [m]')
xlabel('Time [s]')
hold off

return

%% Step response
close all

StepYawLQR = figure;
hold on
SetFigure(1,1.5)
plot(t,alpha_1_ref,'DisplayName','Reference')
plot(t,X(1,:),'DisplayName','Linear model')
plot(t,alpha_1,'DisplayName','Nonlinear model')
xlabel('Time [s]')
ylabel('Yaw [rad]')
set(gca,'YTick',0:pi/4:pi/2)
set(gca,'YTickLabel',{'0','\pi/4','\pi/2'})
ylim([-0.1 pi/2*1.1])
% legend('Location','northoutside','Orientation','horizontal')
hold off

% exportgraphics(StepYawLQR,'StepYawLQR_no_paint.pdf','ContentType','vector')
% exportgraphics(StepYawLQR,'StepYawLQR.pdf','ContentType','vector')

StepSpeedLQR = figure;
hold on
SetFigure(1,1.5)
plot(12+t,sd_ref,'DisplayName','Reference')
plot(12+t,X(5,:),'DisplayName','Linear model')
plot(12+t,sd,'DisplayName','Nonlinear model')
xlabel('Time [s]')
ylabel('Speed [m/s]')
hold off

% exportgraphics(StepSpeedLQR,'StepSpeedLQR_no_paint.pdf','ContentType','vector')
% exportgraphics(StepSpeedLQR,'StepSpeedLQR.pdf','ContentType','vector')

return
```

```
%% Functions

% Fourth order Runge-Kutta - Numerical integration
function y = RK4(x,xd,dt)

k1_2 = dt *  xd;
k2_2 = dt * (xd + k1_2/2);
k3_2 = dt * (xd + k2_2/2);
k4_2 = dt * (xd + k3_2);

y  = x  + (k1_2 + 2*k2_2 + 2*k3_2 + k4_2)/6;

end
```

E.2 Lagrange Calculations

```
clc; clear; close all

% Dimensions [m]
L_ACMF = 0.168196;
L_ACMW = 0.243620;

% Masses [kg]
m_F = 8.997397;
m_W = 1.516;

% Wheel size [m]
r = 0.1651/2;

% Inertia [kg*m^2]
J_F = diag([1.277718, 1.007450, 0.282509]);
J_W = diag([0.003898, 0.003495, 0.003495]);

% Gravitational acceleration [m/s^2]
% g = 9.82;
syms g

% Viscous friction
Bv = 0.00088710922374912487987225206964581;

% Coulumb friction
mu = 0.18084925969874254825242587685352;

%% Positions of COM
disp('Progress: Positions of COM')
tic

% Generalised coordinates
syms alpha_1 alpha_1d alpha_1dd beta_1 beta_1d beta_1dd s sd sdd real

clear A1 S1 PCMF A2 S2 PCMW1 A3 PCMW2

% Main frame COM
A1 = rot3D(alpha_1,beta_1 - pi/2,0);
S1 = [L_ACMF;0;0];

PCMF = simplify(A1*S1);

```

```
% Wheel 1 COM
A2 = rot3D(alpha_1 + pi/2,0,0);
S2 = [L_ACMW;0;0];

PCMW1 = simplify(A2*S2);

% Wheel 2 COM
A3 = rot3D(alpha_1 - pi/2,0,0);

PCMW2 = simplify(A3*S2);

% Paint level
syms L real positive

% Density
rho_pain = 1000;

% Tank height
h = 0.28;

% Tank width
w = 0.235;
d = 0.190;

% Distance from origin to bottom of tank
d_bT = 0.114;

% Mass of tank
m_tank = 0.45;

% Mass of paint
m_pain = rho_pain*(L*w*d);

% Inertia
J_T = diag([1/12*((m_pain + m_tank)*(h^2 + d^2)) + (m_pain + m_tank)*((m_pain*L/2 +
    m_tank*h/2)/(m_pain + m_tank) + d_bT)^2;...
            1/12*((m_pain + m_tank)*(h^2 + w^2)) + (m_pain + m_tank)*((m_pain*L/2 +
    m_tank*h/2)/(m_pain + m_tank) + d_bT)^2;...
            1/12*((m_pain + m_tank)*(d^2 + w^2))]);

% Paint tank COM
S3 = [(m_pain*L/2 + m_tank*h/2)/(m_pain + m_tank) + d_bT;0;0];

PCMT = simplify(A1*S3);

toc

%% Velocities of COM
disp('Progress: Velocities of COM')
tic

clear PCMFd PCM2d

% Time derivative
 PCMFd = difft(PCMF , [alpha_1,beta_1]) + rot3D(alpha_1,0,0)*[sd;0;0];
PCMW1d = difft(PCMW1, [alpha_1,beta_1]) + rot3D(alpha_1,0,0)*[sd;0;0];
PCMW2d = difft(PCMW2, [alpha_1,beta_1]) + rot3D(alpha_1,0,0)*[sd;0;0];
 PCMTd = difft(PCMT , [alpha_1,beta_1]) + rot3D(alpha_1,0,0)*[sd;0;0];

toc
```

```
101
102 %% Lagrangian
103 disp('Progress: Lagrangian')
104 tic
105
106 clear KT KR K P Lag dL_dqd d_dt_dL_dqd dL_dq tau
107
108 % Translational energy
109 KT = 1/2*(m_F*(PCMFd')*PCMFd + m_W*(PCMW1d')*PCMW1d + m_W*(PCMW2d')*PCMW2d + (m_pain +
        m_tank)*(PCMTd')*(PCMTd));
110
111 % Rotational energy
112 KR = 1/2*([alpha_1d;beta_1d       ;0]'*J_F*[alpha_1d;beta_1d       ;0] ...
113        +  [alpha_1d;beta_1d + sd/r;0]'*J_W*[alpha_1d;beta_1d + sd/r;0] ...
114        +  [alpha_1d;beta_1d - sd/r;0]'*J_W*[alpha_1d;beta_1d - sd/r;0] ...
115        +  [alpha_1d;beta_1d       ;0]'*J_T*[alpha_1d;beta_1d       ;0]);
116
117 % Total kinetic energy
118 K = KT + KR;
119
120 % Potential energy
121 P = [0,0,g]*(m_F*PCMF + m_W*PCMW1 + m_W*PCMW2 + (m_pain + m_tank)*PCMT);
122
123 % Lagrange equation
124 Lag = K - P;
125
126 % Derivative w.r.t. qd
127 dL_dqd = [diff(Lag,alpha_1d)
128           diff(Lag, beta_1d);
129           diff(Lag,      sd)];
130
131 % Derivative w.r.t. time
132 d_dt_dL_dqd = difft(dL_dqd,[alpha_1d,alpha_1,beta_1d,beta_1,sd,s]);
133
134 % Derivative w.r.t. q
135 dL_dq = [diff(Lag,alpha_1);
136          diff(Lag, beta_1);
137          diff(Lag,      s)];
138
139 % Joint torque
140 tau = d_dt_dL_dqd - dL_dq;
141
142 toc
143
144 %% Robotics notation
145 disp('Progress: Robotics notation')
146 tic
147
148 % Acceleration and gravity terms in D, and q qd terms in C
149 [D,C] = equationsToMatrix(tau,[alpha_1dd,beta_1dd,sdd,g]);
150
151 % Gravity terms
152 G = simplify(combine(D(:,4),'sincos'));
153
154 % Accelerations terms
155 D = simplify(combine(D(:,1:3),'sincos'));
156
157 C = simplify(combine(-C,'sincos'));
158
159 % Difference and sum torques
```

```
syms T_diff T_sum
% syms T_1 T_2
% T_diff = T_1 - T_2;
% T_sum = T_1 + T_2;

Q = [T_diff/r*L_ACMW + 2*Bv*L_ACMW/r*alpha_1d + 2*mu*L_ACMW/r*sign(alpha_1d);
     2*Bv*beta_1d + 2*mu*sign(beta_1d);
     T_sum/r + 2*Bv/r*sd + 2*mu/r*sign(sd)];

qdd = simplify(combine(D\(Q - C - G*9.82),'sincos'));

alpha_1dd = simplify(combine(qdd(1),'sincos'));
 beta_1dd = simplify(combine(qdd(2),'sincos'));
      sdd = simplify(combine(qdd(3),'sincos'));

toc

return

%% Creating matlab functions
disp('Progress: Creating Matlab functions')
tic

% Matlab functions
addpath("Functions")
matlabFunction(     alpha_1dd,"File","Functions/alpha_1ddf");
matlabFunction(      beta_1dd,"File","Functions/beta_1ddf");
matlabFunction(           sdd,"File","Functions/sddf");
matlabFunction(subs(K,g,9.82),"File","Functions/Kf");
matlabFunction(subs(P,g,9.82),"File","Functions/Pf");

toc
%% Linear Model
clc;

L = 0.28;
alpha_1dd = subs(alpha_1dd,[str2sym('L'),str2sym('sign(alpha_1d)'),str2sym('sign(beta_1d)
    '),str2sym('sign(sd)')],[L,0,0,0]);
 beta_1dd = subs( beta_1dd,[str2sym('L'),str2sym('sign(alpha_1d)'),str2sym('sign(beta_1d)
    '),str2sym('sign(sd)')],[L,0,0,0]);
   sdd = subs(      sdd,[str2sym('L'),str2sym('sign(alpha_1d)'),str2sym('sign(beta_1d)
    '),str2sym('sign(sd)')],[L,0,0,0]);

T = jacobian([alpha_1dd;beta_1dd;sdd],[alpha_1;alpha_1d;beta_1;beta_1d;sd;T_diff;T_sum]);
T = double(subs(T,[alpha_1;alpha_1d;beta_1;beta_1d;sd;],...
                  [      0;       0;     0;      0; 0;]));

B = double(T(:,[end-1,end]));

B = double([0,0;
            B(1,:);
            0,0;
            B(2,:);
            B(3,:)])

T = double(T(:,1:end-2));

A = [0,1,0,0,0;
     T(1,:);
     0,0,0,1,0;
```

```
217       T(2,:);
218       T(3,:)]
219
220 C = diag([1,1,1,1,1]);
221
222 D = 0;
223
224 sys = ss(A,B,C,D);
```

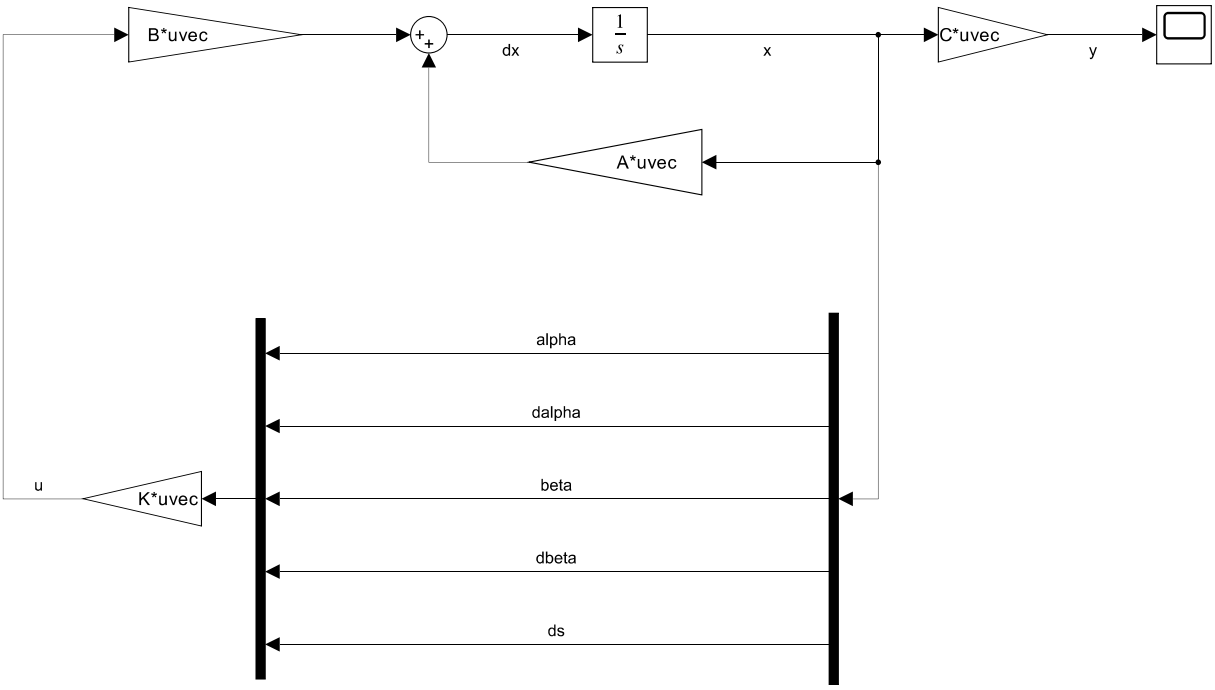


Figure E.1: State space model of the mechanical system with the linear quadratic regulator.

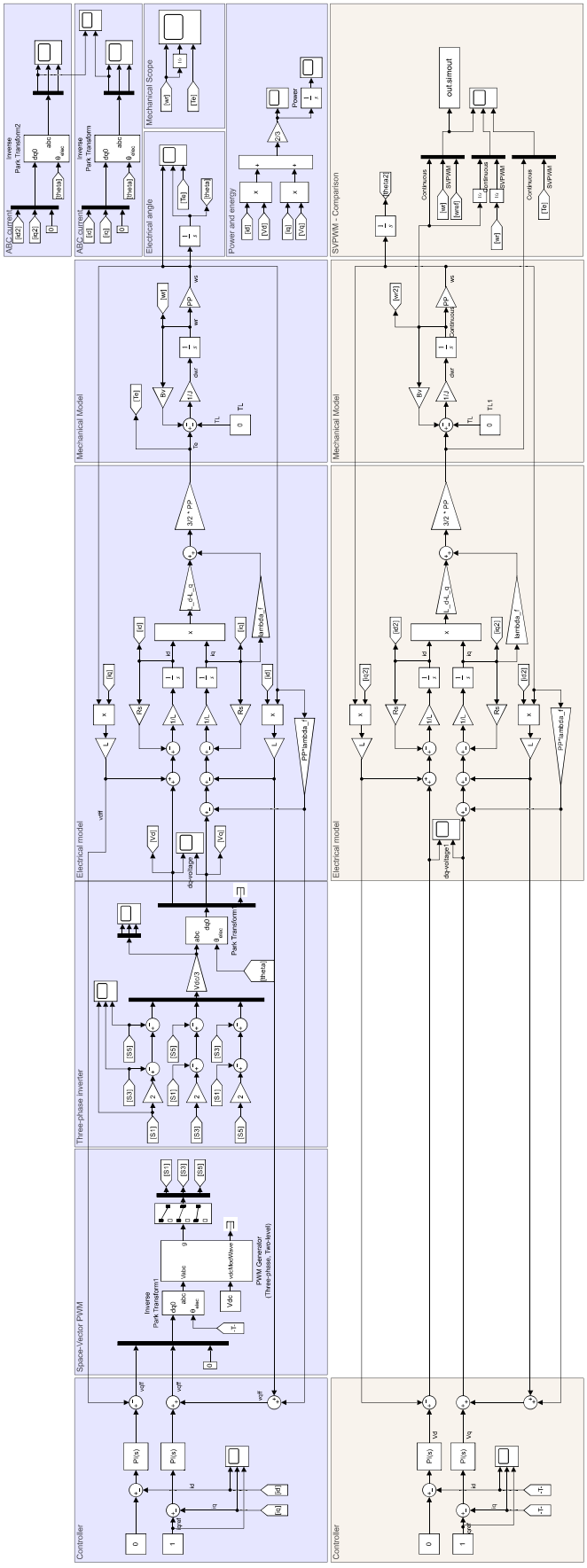


Figure E.2: Simulink model for Nonlinear PMSM with and without SVPWM.

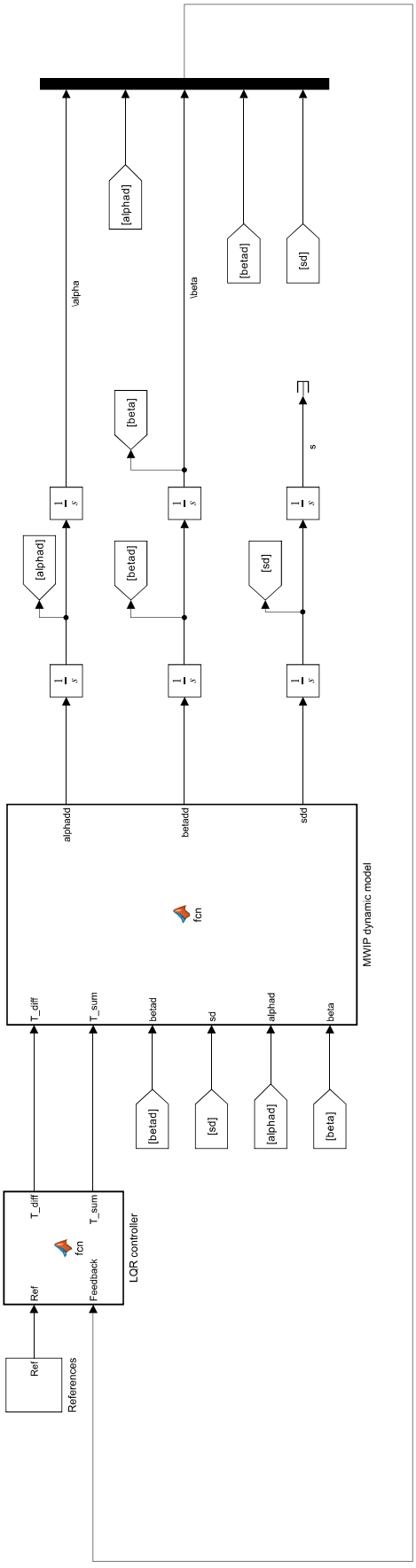


Figure E.3: Simulink model of the nonlinear mechanical model with the linear quadratic regulator.

Appendix F

Determining K_v rating

To determine the K_v rating of the motors a tests are performed on both motors. The test are conducted using the following conditions:

- No input is applied to the system
- The motor is allowed to spin freely
- An electric hand drill is used to spin the motor

In Figure F.1, the setup for the tests is depicted. The tests are conducted by spinning the motor using an electric hand drill until the steady state speed is reached. At the steady state speed the voltage is measured on the oscilloscope. The oscilloscope measures the voltage amplitude and frequency of the sinusoidal signals.

The K_v rating is given as half the measured peak to peak back emf measured across two phases:

$$K_v = \frac{\omega}{\frac{1}{2}V_A} \tag{F.1}$$

where ω is the mechanical speed of the motor, and V_A is the peak to peak value, or amplitude, of the sinusoidal back emf.

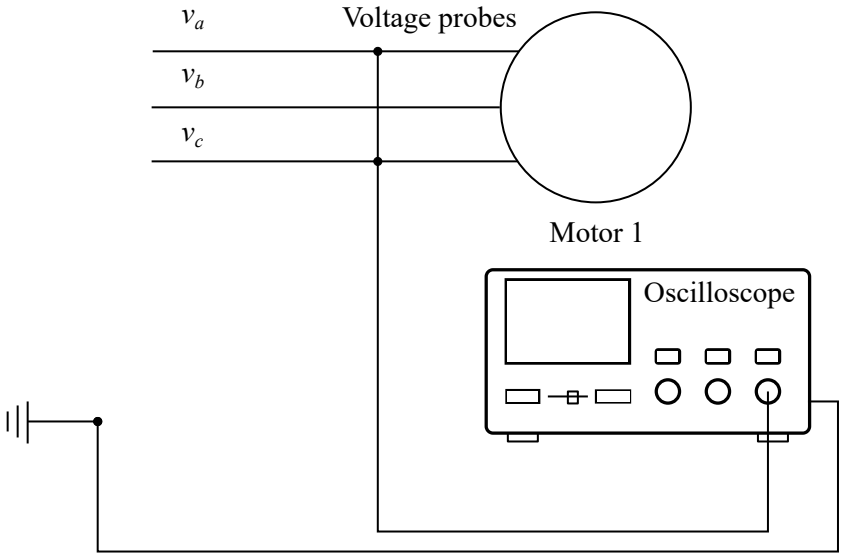


Figure F.1: Setup for determining Kv rating of the motor ($v_{a,b,c}$: phase voltages a, b, and c).

The test is repeated three times for each motor, and the results are shown in Table F.1.

Table F.1: Data from Kv rating tests

Motor 0	Test 1	Test 2	Test 3	Unit
Voltage amplitude	6.96	6.88	7.04	V
Electrical frequency	23.92	23.28	23.99	Hz
Electrical speed	150.3	146.3	150.7	1/s
Mechanical speed	10.02	9.752	10.05	1/s
Kv rating	2.880	2.835	2.855	V s
Motor 1	Test 1	Test 2	Test 3	
Voltage amplitude	6.88	7.04	7.04	V
Electrical frequency	23.18	23.75	24.03	Hz
Electrical speed	145.6	149.2	151.0	1/s
Mechanical speed	9.710	9.949	10.07	1/s
Kv rating	2.823	2.826	2.860	V s

Based on the six total tests the Kv rating is calculated as the average of the results obtained for each motor, this yields the Kv ratings listed below:

$$\begin{aligned} K_{v,0} &= 2.856\,\mathrm{V\,s} \\ K_{v,1} &= 2.836\,\mathrm{V\,s} \end{aligned} \tag{F.2}$$

With $K_{v,0}$ and $K_{v,1}$ representing the Kv rating of motor 0 and motor 1, respectively.

Relationship between Velocity Constant, Torque Constant and Flux linkage

By analysing the power of a surface mounted PMSM the flux linkage of the permanent magnet can be found using the measured Kv rating. The power is given as:

$$P = \frac{3}{2} i_q v_q = T\omega \tag{F.3}$$

Dividing q-axis current and motor speed ω on both sides gives:

$$\frac{3}{2}\frac{v_q}{\omega} = \frac{T}{i_q} \tag{F.4}$$

The left hand side of this equation is simplified by denoting v_q/ω as the phase velocity constant K_v^*, and the expression for torque is inserted:

$$\frac{3}{2}\frac{1}{K_v^*} = \frac{\frac{3}{2}PP\lambda_f i_q}{i_q} = \frac{3}{2}PP\lambda_f \tag{F.5}$$

Solving for the flux linkage:

$$\lambda_f = \frac{1}{K_v^* PP} \tag{F.6}$$

The transformation from phase voltage to line-to-line voltage in a balanced three-phase system is $\sqrt{3}$ which gives:

$$\lambda_{f,0} = \frac{1}{K_{v,0}\sqrt{3}PP} = 0.0135\,\mathrm{Wb} \tag{F.7}$$

$$\lambda_{f,1} = \frac{1}{K_{v,1}\sqrt{3}PP} = 0.0136\,\mathrm{Wb} \tag{F.8}$$

The torque constant is defined as the relationship between the torque generating current and the torque i.e.:

$$K_{T,0} = \frac{3}{2}PP\lambda = 0.303\,\mathrm{N\,m/A} \tag{F.9}$$

$$K_{T,1} = \frac{3}{2}PP\lambda = 0.305\,\mathrm{N\,m/A} \tag{F.10}$$

Appendix G

Linear Quadratic Regulator

In this appendix the underlying theory behind the linear quadratic regulator is presented.

The following theory is based on the derivation given in [72]. Given the linear system equations:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{G.1}$$

with the control law given as:

$$\mathbf{u} = -\mathbf{K}\mathbf{x} \tag{G.2}$$

The control vector $\mathbf{K}$ is chosen such that the performance index J is minimised:

$$J = \int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt \tag{G.3}$$

Inserting the control law (G.2) in the linear system equations (G.1) yields:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}$$

Henceforward it is assumed that the matrix $\mathbf{A} - \mathbf{B}\mathbf{K}$ is stable i.e. all eigenvalues have negative real parts.

Inserting the control law (G.2) into the performance index (G.3) yields:

$$\begin{aligned} J &= \int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{x}^T\mathbf{K}^T\mathbf{R}\mathbf{x}\mathbf{K}\right) dt \\ &= \int_0^\infty \left(\mathbf{x}^T(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}\right) dt \end{aligned}$$

A positive definite matrix $\mathbf{P}$ is introduced as:

$$\mathbf{x}^T(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x} = -\frac{d}{dt}(\mathbf{x}^T\mathbf{P}\mathbf{x})$$

Which yields:

$$\mathbf{x}^T(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x} = -\dot{\mathbf{x}}^T\mathbf{P}\mathbf{x} - \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}} = -\mathbf{x}^T\left((\mathbf{A} - \mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{B}\mathbf{K})\right)\mathbf{x}$$

Eliminating $\mathbf{x}$ from the equation above yields:

$$(\mathbf{A} - \mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{B}\mathbf{K}) = -(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}) \tag{G.4}$$

As the matrix $\mathbf{A} - \mathbf{B}\mathbf{K}$ is stable there exists a positive-definite matrix $\mathbf{P}$ that satisfies (G.4).

Evaluating the performance index using the **P** matrix yields:

$$J = \int_0^\infty \left(\mathbf{x}^T(\mathbf{Q}+\mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}\right) dt = -\mathbf{x}^T\mathbf{P}\mathbf{x}|_0^\infty = -\mathbf{x}(\infty)\mathbf{P}\mathbf{x}(\infty) + \mathbf{x}(0)^T\mathbf{P}\mathbf{x}(0)$$

And as the system is assumed stable $\mathbf{x}(\infty) \to \mathbf{0}$ which yields the performance index in terms of the initial condition and **P**:

$$J = \mathbf{x}(0)^T\mathbf{P}\mathbf{x}(0)$$

To solve this equation another matrix **T** is introduced:

$$\mathbf{R} = \mathbf{T}^T\mathbf{T}$$

This is permissible as **R** is positive definite. Rewriting (G.4) by introducing **T**:

$$(\mathbf{A}-\mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K}) + (\mathbf{Q}+\mathbf{K}^T\mathbf{T}^T\mathbf{T}\mathbf{K}) = \mathbf{0}$$

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

Minimising J with respect to **K**, simplifies the minimisation to:

$$\mathbf{x}^T(\mathbf{T}\mathbf{K} - (\mathbf{T}^T)^{-1}\mathbf{B}^T\mathbf{P})^T(\mathbf{T}\mathbf{K} - (\mathbf{T}^T)^{-1}\mathbf{B}^T\mathbf{P})\mathbf{x}$$

As this expression is non negative, it is minimised when it is zero or when:

$$\mathbf{T}\mathbf{K} = (\mathbf{T}^T)^{-1}\mathbf{B}^T\mathbf{P}$$

Isolating **K** in the above equation yields the optimised feedback gain matrix:

$$\mathbf{K} = \mathbf{T}^{-1}(\mathbf{T}^T)^{-1}\mathbf{B}^T\mathbf{P} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{G.5}$$

Reducing (G.4) by inserting **K** yields the reduced-matrix Riccati equation:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{G.6}$$

which must also be satisfied by the matrix **P**.

The procedure to obtain the optimal feedback gain matrix **K** is to solve (G.6) for **P** and if **P** is positive definite the system $\mathbf{A}-\mathbf{B}\mathbf{K}$ is stable. Then substituting **P** into (G.5) yields the optimised feedback gain matrix **K**.

Appendix H

Data from Experiments

H.1 Data from Investigation of the Back emf waveform

Figure H.1 shows the measurement of the signals' frequency using the faster speed of the drill. The measurements of the signals' frequency and amplitude using the slower speed of the drill are seen in Figures H.2 and H.3.

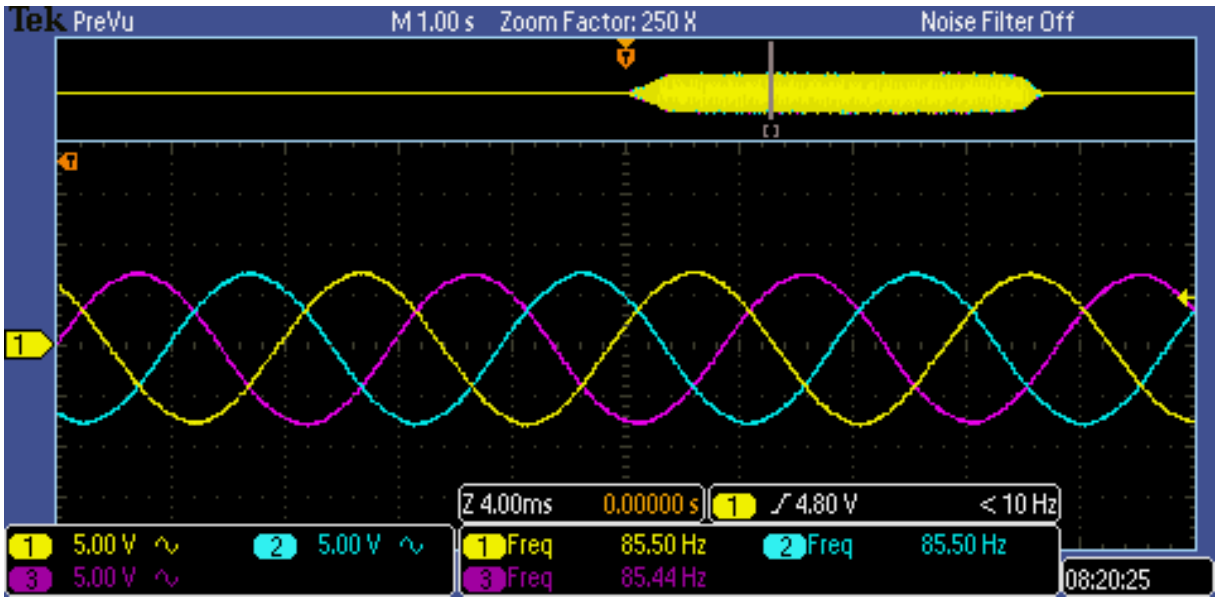


Figure H.1: Oscilloscope measurement of the back emf waveform with the drill set to the fastest speed, where the oscilloscope measures the frequency of the signal.

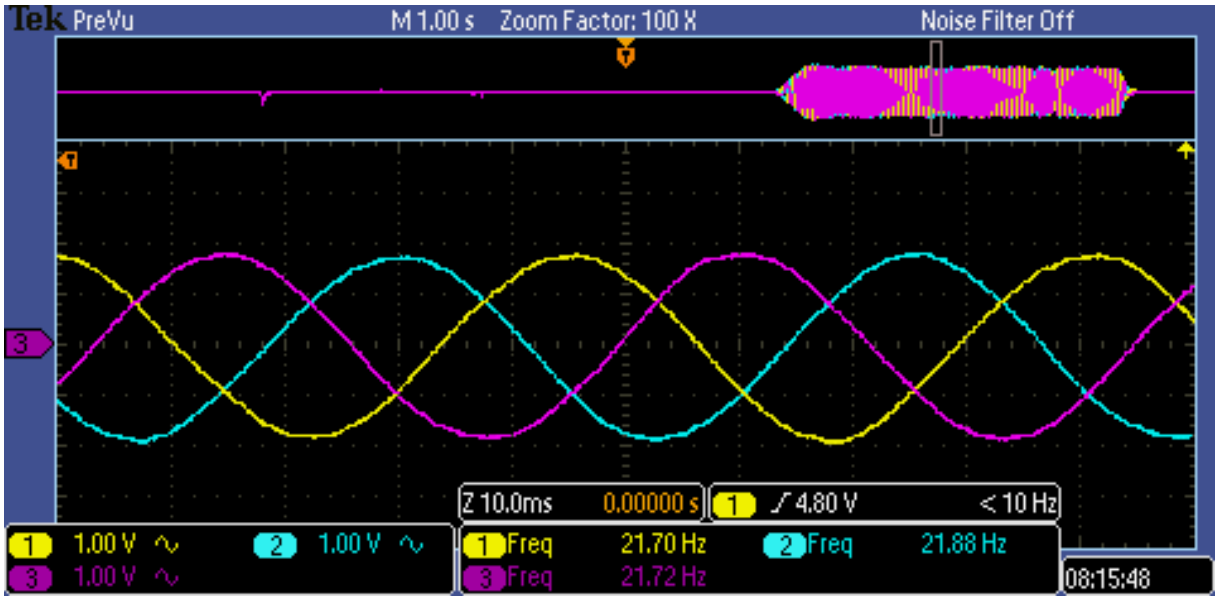


Figure H.2: Oscilloscope measurement of the back emf waveform with the drill set to the slowest speed, where the oscilloscope measures the frequency of the signal

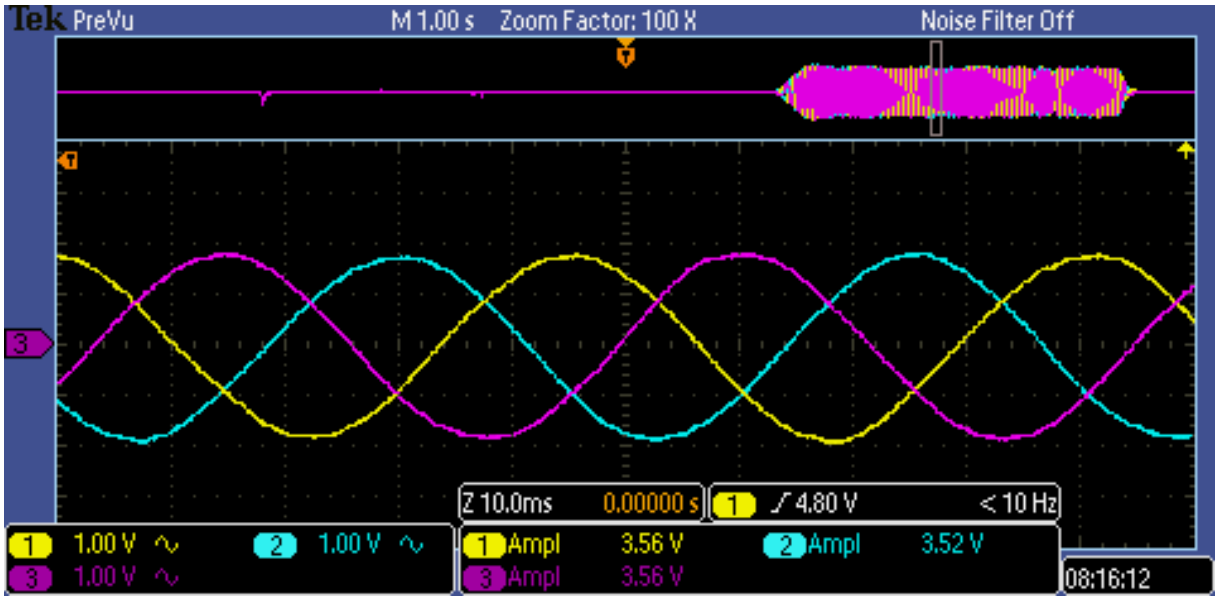
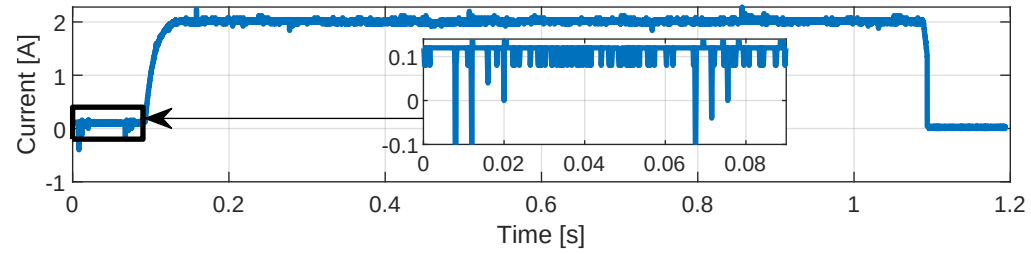


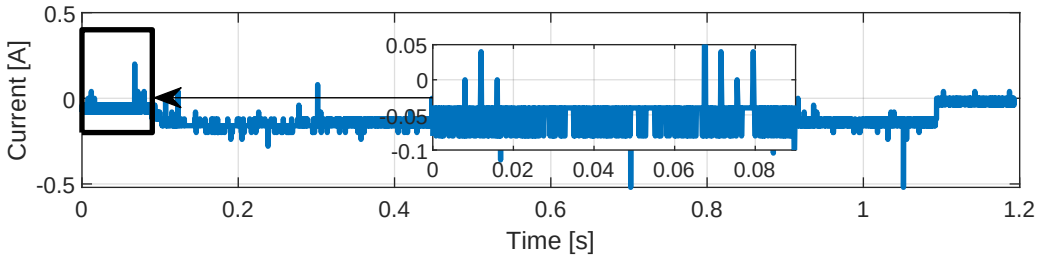
Figure H.3: Oscilloscope measurement of the back emf waveform with the drill set to the slowest speed, where the oscilloscope measures the amplitude of the signal

H.2 Test of the Closed q-axis Current Loop

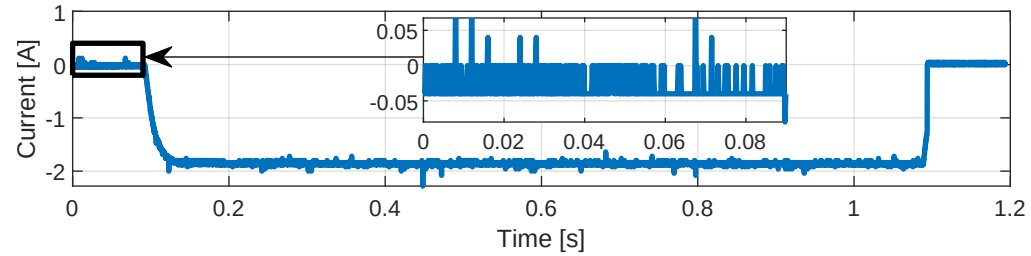
Figure H.4 displays the offset in the current value in the beginning of the tests.



(a) Phase *a* current.



(b) Phase *b* current.

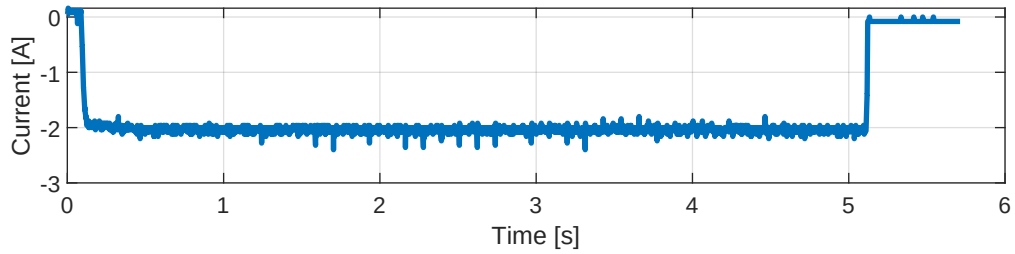


(c) Phase *c* current.

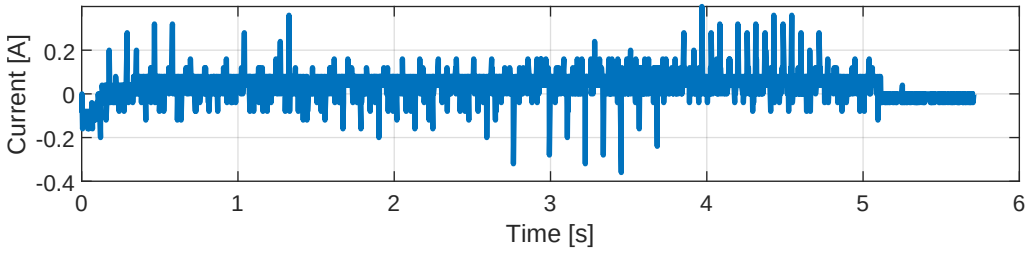
Figure H.4: Phase current response in blocked rotor test with zoomed windows.

H.3 Additional Test of the Closed q-axis Current Loop

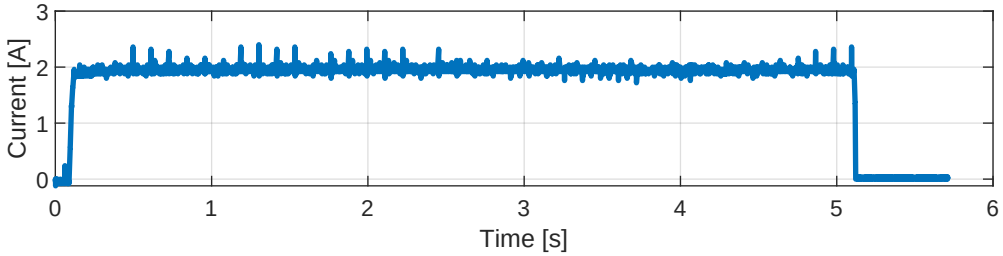
Figure H.5 shows an additional test of the Closed q-axis Current Loop where the rotor is in a different position.



(a) Phase *a* current.



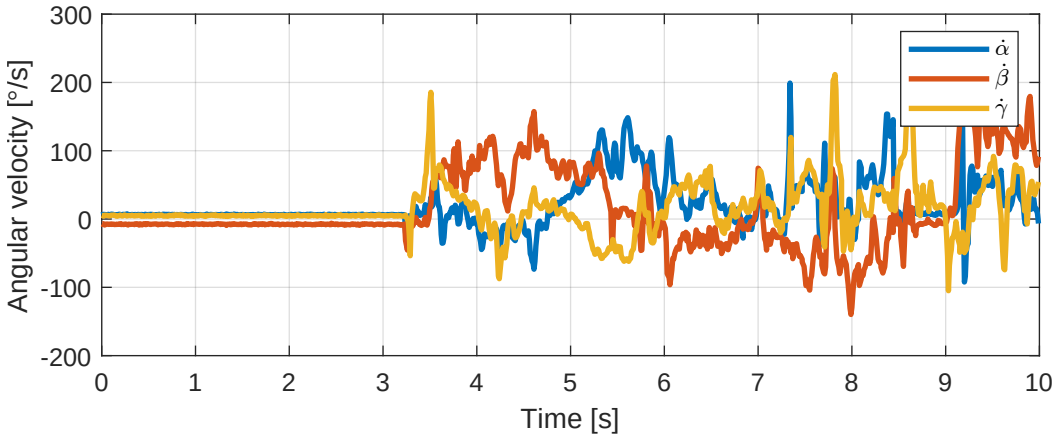
(b) Phase *b* current.



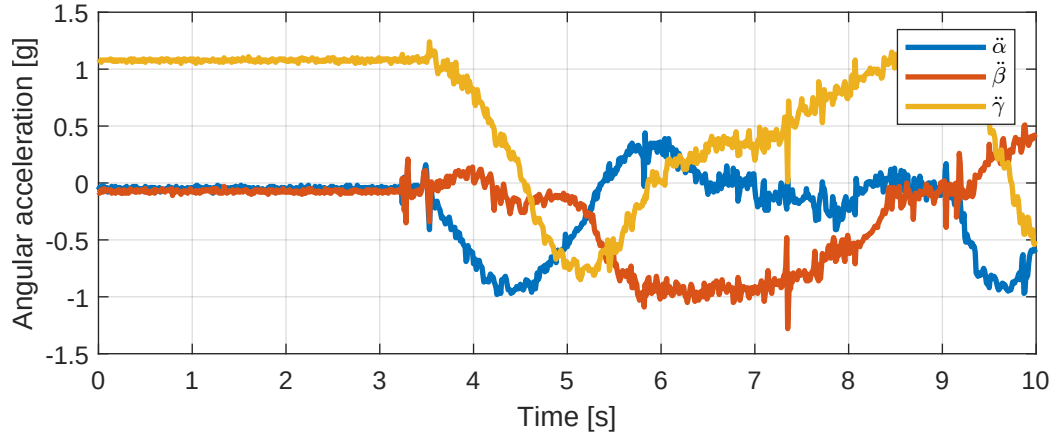
(c) Phase *c* current.

Figure H.5: Additional test of phase current response in blocked rotor test.

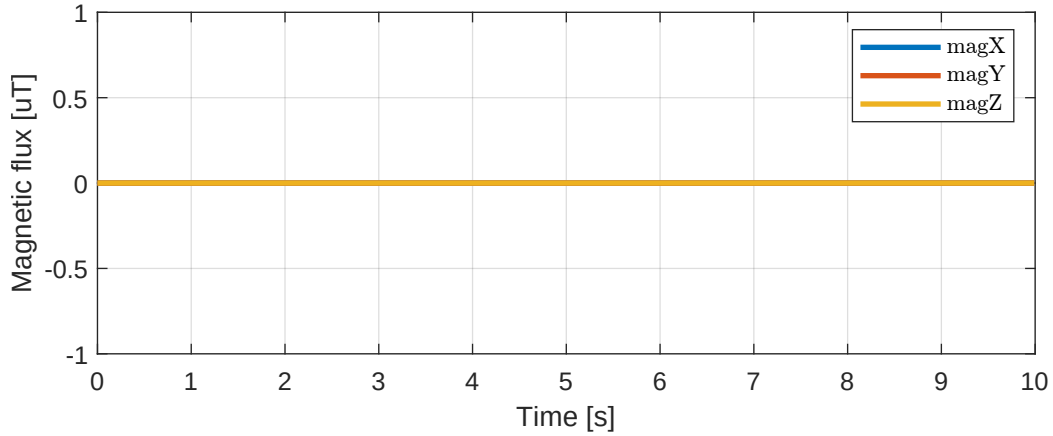
H.4 Test of the IMU GY-91



(a) Gyroscope measurements.



(b) Accelerometer measurements.



(c) Magnetometer measurements.

Figure H.6: Measurements received from the GY-91.

Appendix I

Arduino Code

I.1 Arduino code using BNO055

Below the Arduino code including wireless communication, obstacle avoidance, and the BNO055 IMU is included

```
    // IMU Libs
#include <Wire.h> // I2C communication
#include <Adafruit_Sensor.h> // Common for all adafruit sensors to allow for reading values
#include <Adafruit_BNO055.h> // Sensor specific to allow for reading values
#include <utility/imumaths.h> // math package for IMU
#include <EEPROM.h> // Allows for saving IMU calibration on MCU

// IMU calibration variables
bool zero = true;
bool calibrate = true;
int zeroTime = 50;

// Set sample time of the IMU
#define BNO055_SAMPLERATE_DELAY_MS (100)


// Check I2C device address
//                                   id, address
Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28);

// IMU reading variables
double alpha,beta,sd,beta_meas,betad,beta_old,betai,s,s_gain;
// Variable for integration
unsigned long start;
// Variable for unwrapping of yaw angle
int unwrapFlag = 0;


// ODrive lib
#include <ODriveArduino.h>
// Printing with stream operator
template<class T> inline Print& operator <<(Print &obj,     T arg) { obj.print(arg); return obj; }
template<>        inline Print& operator <<(Print &obj, float arg) { obj.print(arg, 4); return obj; }

// Declare serial port for ODrive
#define odrive_serial1 Serial1
ODriveArduino odrive1(odrive_serial1);


// Wireless comunication
#define HC12 Serial2 // Set wireless comunication to use Serial 2 (7 and 8 pin)
```

```
byte incomingByte; // Variable to store incoming data
String readBuffer = ""; // Variable to store incoming data
# include "ctype.h" // Formatting lib

// Ultrasonic sensor
const int trigPin = 23; // HC-SR04 trigger pin
const int echoPin = 22; // HC-SR04 echo pin
float duration, distance; // For calculation of distance
bool obstacle = 0;         // Variable to signal the presence of a obstacle

// LED
const int ledPin = 13; // On board LED

// Variable for emergency stop
int stop = 0;

// Variables for storing motor currents
float motor0_current;
float motor1_current;

// Codeblock for Sensor details
void displaySensorDetails(void)
{
    sensor_t sensor;
    bno.getSensor(&sensor);
    Serial.println("------------------------------------");
    Serial.print("Sensor:       "); Serial.println(sensor.name);
    Serial.print("Driver Ver:   "); Serial.println(sensor.version);
    Serial.print("Unique ID:    "); Serial.println(sensor.sensor_id);
    Serial.print("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" xxx"
    );
    Serial.print("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" xxx"
    );
    Serial.print("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" xxx
    ");
    Serial.println("------------------------------------");
    Serial.println("");
    delay(500);
}

    // Display some basic info about the sensor status

void displaySensorStatus(void)
{
    // Get the system status values (mostly for debugging purposes)
    uint8_t system_status, self_test_results, system_error;
    system_status = self_test_results = system_error = 0;
    bno.getSystemStatus(&system_status, &self_test_results, &system_error);

    // Display the results in the Serial Monitor
    Serial.println("");
    Serial.print("System Status: 0x");
    Serial.println(system_status, HEX);
    Serial.print("Self Test:     0x");
    Serial.println(self_test_results, HEX);
    Serial.print("System Error:  0x");
    Serial.println(system_error, HEX);
    Serial.println("");
    delay(500);
}
```

```
99
100     // Display sensor calibration status
101
102 void displayCalStatus(void)
103 {
104     // Get the four calibration values (0..3)
105     // Any sensor data reporting 0 should be ignored,
106     // 3 means 'fully calibrated"
107     uint8_t system, gyro, accel, mag;
108     system = gyro = accel = mag = 0;
109     bno.getCalibration(&system, &gyro, &accel, &mag);
110
111     /* The data should be ignored until the system calibration is > 0 */
112     Serial.print("\t");
113     if (!system)
114     {
115         Serial.print("! ");
116     }
117
118     /* Display the individual values */
119     Serial.print("Sys:");
120     Serial.print(system, DEC);
121     Serial.print(" G:");
122     Serial.print(gyro, DEC);
123     Serial.print(" A:");
124     Serial.print(accel, DEC);
125     Serial.print(" M:");
126     Serial.print(mag, DEC);
127 }
128
129
130     // Display the raw calibration offset and radius data
131
132 void displaySensorOffsets(const adafruit_bno055_offsets_t &calibData)
133 {
134     Serial.print("Accelerometer: ");
135     Serial.print(calibData.accel_offset_x); Serial.print(" ");
136     Serial.print(calibData.accel_offset_y); Serial.print(" ");
137     Serial.print(calibData.accel_offset_z); Serial.print(" ");
138
139     Serial.print("\nGyro: ");
140     Serial.print(calibData.gyro_offset_x); Serial.print(" ");
141     Serial.print(calibData.gyro_offset_y); Serial.print(" ");
142     Serial.print(calibData.gyro_offset_z); Serial.print(" ");
143
144     Serial.print("\nMag: ");
145     Serial.print(calibData.mag_offset_x); Serial.print(" ");
146     Serial.print(calibData.mag_offset_y); Serial.print(" ");
147     Serial.print(calibData.mag_offset_z); Serial.print(" ");
148
149     Serial.print("\nAccel Radius: ");
150     Serial.print(calibData.accel_radius);
151
152     Serial.print("\nMag Radius: ");
153     Serial.print(calibData.mag_radius);
154 }
155
156 // Ultrasonic distance measuring code block
157 void distance_meas() {
158
```

```
  // Turn off trigger
  digitalWrite(trigPin, LOW);
  // Allow time for pulling the pin low
  delayMicroseconds(2);
  // Trigger ultrasonic sensor
  digitalWrite(trigPin, HIGH);
  // 10 microsecond echo
  delayMicroseconds(10);
  // Turn off trigger
  digitalWrite(trigPin, LOW);

  // Read distance as pulse on echo pin
  duration = pulseIn(echoPin, HIGH);
  // Convert to cm using speed of sound, /2 because it travels out and back
  distance = (duration*.0343)/2;


  // Trigger onboard LED if distance is less than 10 cm
  if (distance < 5){
    digitalWrite(ledPin, HIGH);
    obstacle = 1;
    // Halt the motors
    odrive1.SetCurrent(1, 0);
    odrive1.SetCurrent(0, 0);
  }
  else {
    digitalWrite(ledPin, LOW);
    obstacle = 0;
  }
}

// Code block for wireless communication
void Wireless(){
 while (HC12.available()) {        // If HC-12 has data
    readBuffer = (HC12.readStringUntil(char(10)));      // Send the data to Serial
    monitor

    int str_len = readBuffer.length() + 1;
    char char_array[str_len];
    readBuffer.toCharArray(char_array,str_len);
    // Check incoming data for stop variable
    sscanf(char_array,"%d",&stop);
}

}

// IMU code block
void IMU_code(){

  //Save old yaw value for integration
  double alpha_old = alpha;
  //Update IMU values
  sensors_event_t orientationData, angVelocityData , linearAccelData;

  // Get quaternion values and convert to euler
  imu::Vector<3> euler = bno.getQuat().toEuler();
  bno.getEvent(&angVelocityData, Adafruit_BNO055::VECTOR_GYROSCOPE);
  bno.getEvent(&linearAccelData, Adafruit_BNO055::VECTOR_LINEARACCEL);

  // Save values from gyroscope and accelerometer
```

```
218  float alphad = -angVelocityData.gyro.z;
219  betad = -angVelocityData.gyro.y;
220  alpha = euler.x();
221  beta_meas = euler.y();
222
223  // Unwrap the yaw angle
224  if(alpha-alpha_old < -3.1415){
225    unwrapFlag = unwrapFlag + round(abs((alpha-alpha_old)/(2*3.1415)));
226  }
227  else if(alpha-alpha_old > 3.1415){
228    unwrapFlag = unwrapFlag - round(abs((alpha-alpha_old)/(2*3.1415)));
229  }
230
231  float alphaU = alpha + unwrapFlag*6.283;
232
233
234  // Start time for dt
235  start = millis() * 1e-3;
236
237  // Get motor velocities
238  float w0 = odrive1.GetVelocity(0)*(2*3.1415);
239  float w1 = -odrive1.GetVelocity(1)*(2*3.1415);
240
241  // Torque calculations
242  sd = (0.2436*alphad - 0.0825*w0 - 0.0825*w1 - 0.2436*alphad)/2;
243  float T_diff = (-20*(alphaU) - 11.5*(alphad) + 0*(beta_meas) + 0*(betad) + 0*(sd));
244  float T_sum = (0*(alphaU) + 0*(alphad) + 65*(beta_meas) + 15*(betad) + 5*(sd));
245  float kt = 0.31;
246
247  // Convertion to current
248  motor0_current = (T_diff/2 + T_sum/2)/(kt);
249  motor1_current = (T_sum/2 - T_diff/2)/(kt);
250
251  // Insert cap on current reference
252  float cap = 4;
253  if(abs(motor1_current) < cap){
254    if(abs(motor1_current) < 0){
255      odrive1.SetCurrent(1,0 * ((abs(motor1_current))/(motor1_current)));
256    }
257    else {
258      odrive1.SetCurrent(1,motor1_current);
259    }
260  }
261  else{
262    odrive1.SetCurrent(1,cap * ((abs(motor1_current))/(motor1_current)));
263  }
264
265  if(abs(motor0_current) < cap){
266    if(abs(motor0_current) < 0){
267      odrive1.SetCurrent(0,-0 * ((abs(motor0_current))/(motor0_current)));
268    }
269    else {
270      odrive1.SetCurrent(0,-motor0_current);
271    }
272  }
273  else{
274    odrive1.SetCurrent(0,-cap * ((abs(motor0_current))/(motor0_current)));
275  }
276
277
```

```
}



void setup() {
  pinMode(PIN_A4, INPUT_PULLUP);            // Enable pullup resistors for I2C
    communication with BNO055
  pinMode(PIN_A5, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);                  // Set inbuilt led as output
  pinMode(trigPin, OUTPUT);                 // Set ultrasonic trigger pin as output
  pinMode(echoPin, INPUT);                  // Set ultrasonic echo pin as input
  Serial.begin(115200);                     // Open serial port for result printing
  HC12.begin(9600);                        // Open serial port to HC12 (for wireless)
  odrive_serial1.begin(57600);              // Open serial port to odrive (for motor
    control)

  Wire.begin(); // Begin I2C
  Wire.setClock(400000); //400khz clock

  delay(1000);

  while (!Serial) delay(10);  // wait for serial port to open!

  // Set motor current to zero during setup
      odrive1.SetCurrent(1, 0);
      odrive1.SetCurrent(0, 0);

  Serial.println("Orientation Sensor Test"); Serial.println("");

  // Initialise the sensor
  if (!bno.begin())
  {
    // There was a problem detecting the BNO055 ... check your connections
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while (1);
  }
  // Calibration code
  if(calibrate) {
      int eeAddress = 0;
      long bnoID;
      bool foundCalib = false;

      EEPROM.get(eeAddress, bnoID);

      adafruit_bno055_offsets_t calibrationData;
      sensor_t sensor;

      /*
      *  Look for the sensor's unique ID at the beginning oF EEPROM.
      */
      bno.getSensor(&sensor);
      if (bnoID != sensor.sensor_id)
      {
          Serial.println("\nNo Calibration Data for this sensor exists in EEPROM");
          delay(500);
      }
      else
      {
          Serial.println("\nFound Calibration for this sensor in EEPROM.");
          eeAddress += sizeof(long);
```

```
336         EEPROM.get(eeAddress, calibrationData);
337
338         displaySensorOffsets(calibrationData);
339
340         Serial.println("\n\nRestoring Calibration data to the BNO055...");
341         bno.setSensorOffsets(calibrationData);
342
343         Serial.println("\n\nCalibration data loaded into BNO055");
344         foundCalib = true;
345     }
346
347     delay(1000);
348
349     // Display some basic information on this sensor
350     displaySensorDetails();
351
352     // Optional: Display current status
353     displaySensorStatus();
354
355    //Use external time crystal for the BNO to give better time readings
356     bno.setExtCrystalUse(true);
357
358     //Calibrate sensor
359     sensors_event_t event;
360     bno.getEvent(&event);
361     if (foundCalib){
362         Serial.println("Move sensor slightly to calibrate magnetometers");
363         while (!bno.isFullyCalibrated())
364         {
365             bno.getEvent(&event);
366             delay(BNO055_SAMPLERATE_DELAY_MS);
367         }
368     }
369     else
370     {
371         Serial.println("Please Calibrate Sensor: ");
372         while (!bno.isFullyCalibrated())
373         {
374             bno.getEvent(&event);
375
376             imu::Vector<3> euler = bno.getQuat().toEuler();
377
378             double x = euler.y() * (180/3.1415);
379             double y = euler.z() * (180/3.1415);
380             double z = euler.x() * (180/3.1415);
381
382             Serial.print("X: ");
383             Serial.print(x, 4);
384             Serial.print(" Y: ");
385             Serial.print(y, 4);
386             Serial.print(" Z: ");
387             Serial.print(z, 4);
388             Serial.print("\t\t");
389
390             // Display calibration status
391             displayCalStatus();
392
393             // New line for the next sample
394             Serial.println("");
395
```

```
396               // Wait the specified delay before requesting new data
397               delay(BNO055_SAMPLERATE_DELAY_MS);
398           }
399       }
400
401       Serial.println("\nFully calibrated!");
402       Serial.println("--------------------------------");
403       Serial.println("Calibration Results: ");
404       adafruit_bno055_offsets_t newCalib;
405       bno.getSensorOffsets(newCalib);
406       displaySensorOffsets(newCalib);
407
408       Serial.println("\n\nStoring calibration data to EEPROM...");
409
410       eeAddress = 0;
411       bno.getSensor(&sensor);
412       bnoID = sensor.sensor_id;
413
414       EEPROM.put(eeAddress, bnoID);
415
416       eeAddress += sizeof(long);
417       EEPROM.put(eeAddress, newCalib);
418       Serial.println("Data stored to EEPROM.");
419     }
420     else {
421       bno.setExtCrystalUse(true);
422     }
423
424     if(zero) {
425       Serial.println("Zeroing... Please do not move the device");
426       delay(1000);
427     }
428     // Set BNO to sensor fusion mode
429     bno.setMode(OPERATION_MODE_NDOF);
430     delay(500);
431
432 }
433
434 void loop() {
435
436   Wireless(); // Check wireless
437   distance_meas(); // Check for obstacles
438
439   // Check for emergency stop
440     if (stop == 1){
441       Serial.println("Stop");
442       // Set motor current to 0
443       odrive1.SetCurrent(1, 0);
444       odrive1.SetCurrent(0, 0);
445       // Check wireless again
446       Wireless();
447     }
448     else{
449         // If no obstacle is prescent, then proceed
450       if (obstacle == 0){
451     IMU_code();
452     }
453 }
454
455
```

```
}
```

I.2 Arduino code using the GY-91

Below the Arduino code using sensor fusion and the GY-91 is included

```
#include "FastIMU.h"
#include <Wire.h>

#define IMU_ADDRESS 0x68    // Change to the address of the IMU
#define PERFORM_CALIBRATION // Comment to disable startup calibration
MPU6500 IMU;                // Change to the name of any supported IMU!

// Currently supported IMUS: MPU9255 MPU9250 MPU6886 MPU6500 MPU6050 ICM20689 ICM20690 BMI055 BMX055 BMI160 LSM6DS3 LSM6DSL QMI8658

calData calib = { 0 };  //Calibration data
AccelData accelData;    //Sensor data
GyroData gyroData;
MagData magData;

#include <ODriveArduino.h>
// Printing with stream operator
template<class T> inline Print& operator <<(Print &obj,     T arg) { obj.print(arg); return obj; }
template<>        inline Print& operator <<(Print &obj, float arg) { obj.print(arg, 4); return obj; }
#define odrive_serial1 Serial1
ODriveArduino odrive1(odrive_serial1);

// Torque constant
float K_T = 0.3152;

// Define pi
const float pi = 3.14159265358979323846;

// Degrees to radians and radians to degree convertation
float deg2rad = pi/180;
float rad2deg = 180/pi;

// Bias in manual calibration
float X_bias;
float Y_bias;
float Z_bias;

// Times
float start_IMU;
float start_control;

// Gyroscope variables
float gyroX, gyroXd, accelX, velX, posX, magX;
float gyroY, gyroYd, accelY, velY, posY, magY;
float gyroZ, gyroZd, accelZ, velZ, posZ, magZ;

// Angles and change in angles
float alpha, alphad;
float beta, betad;
float Gamma, Gammad;
```

```
50
51 // Motor variables
52 float i_1,i_2;
53 float w0,w1;
54 float theta0,theta1;
55 float sd;
56 float T_diff, T_sum;
57
58 // Initial time
59 float t0;
60
61 void setup() {
62
63   // Enable pull up resistors to IMU
64   pinMode(PIN_A4,INPUT_PULLUP);
65   pinMode(PIN_A5,INPUT_PULLUP);
66
67   // Initialise I2C Comminucation
68   Wire.begin();
69   Wire.setClock(400000); //400khz clock
70
71   // Allows gyro reading up to 2000 degree/second
72   IMU.setGyroRange(2000);
73
74   // Initialise serial communication with PC
75   Serial.begin(115200);
76   delay(100);
77
78   // Initialise serial communication with ODrive
79   odrive_serial1.begin(115200);
80   delay(100);
81
82   // Checking for faults with IMU
83   int err = IMU.init(calib, IMU_ADDRESS);
84   if (err != 0) {
85     Serial.print("Error initializing IMU: ");
86     Serial.println(err);
87     while (true) {
88       ;
89     }
90   }
91   delay(100);
92
93   // Performing automatic calibration
94   #ifdef PERFORM_CALIBRATION
95     Serial.println("FastIMU calibration - Keep IMU level.");
96     delay(500);
97     IMU.calibrateAccelGyro(&calib);
98     IMU.init(calib, IMU_ADDRESS);
99   #endif
100
101  // Performing automatic calibration
102  int I = 50;
103  for (int i = 0; i <= I; i++) {
104    // Read gyroscope
105    IMU.update();
106    IMU.getGyro(&gyroData);
107
108    // Define bias
109    X_bias += gyroData.gyroX;
```

```
    Y_bias += gyroData.gyroY;
    Z_bias += gyroData.gyroZ;

    // Delay
    delay(10);
  }

  X_bias = X_bias/I;
  Y_bias = Y_bias/I;
  Z_bias = Z_bias/I;

  Serial.println("Calibration done!");
  delay(100);

  t0 = millis();
}

void IMU_code() {

  // Creating change in time dt since last calculation
  float end = start_IMU;
  start_IMU = 1E-3 * millis();
  float dt = start_IMU - end;

  // Reading IMU data
  IMU.update();
  IMU.getGyro(&gyroData);
  IMU.getAccel(&accelData);
  IMU.getMag(&magData);

  // Gyroscope data
  gyroXd = (gyroData.gyroX - X_bias)*deg2rad;
  gyroYd = (gyroData.gyroY - Y_bias)*deg2rad;
  gyroZd = (gyroData.gyroZ - Z_bias)*deg2rad;

  // Change in states
  alphad = gyroZd;
   betad = gyroYd;
  Gammad = gyroXd;

  // Integration
  gyroX = RK4(gyroX,gyroXd,dt);
  gyroY = RK4(gyroY,gyroYd,dt);
  gyroZ = RK4(gyroZ,gyroZd,dt);

  // Accelerometer data
  accelX = accelData.accelX;
  accelY = accelData.accelY;
  accelZ = accelData.accelZ;

  // Magnetometer data
  magX = magData.magX;
  magY = magData.magY;
  magZ = magData.magZ;

  // Pitch from accelerometer
  float beta_accel = atan2(-accelX,sqrt(pow(accelY,2) + pow(accelZ,2)));

  // Roll from accelerometer
  float gamma_accel = atan2(accelY,accelZ);
```

```
170
171   // Yaw from gyroscope
172   alpha = gyroZ;
173
174   // Complimentary filter on pitch
175   beta = (beta + gyroYd*dt)*0.95 + beta_accel*(1 - 0.95);
176
177   // Complimentary filter on roll
178   Gamma = (Gamma + gyroXd*dt)*0.95 + gamma_accel*(1 - 0.95);
179
180 }
181
182 void LQR_Control(){
183
184   // Creating change in time dt since last calculation
185   float end = start_control;
186   start_control = 1E-3 * millis();
187   float dt = start_control - end;
188
189   // Read motor speeds
190   w0 = odrive1.GetVelocity(0)*(2*pi);
191   w1 = odrive1.GetVelocity(1)*(2*pi);
192
193   // Integrate
194   theta0 = RK4(theta0,w0,dt);
195   theta1 = RK4(theta1,w1,dt);
196
197   // Linear velocity
198   sd = (0.2436*gyroZd - 0.0825*w0 - 0.0825*w1 - 0.2436*gyroZd)/2;
199
200   // LQR - Gain matrix
201   T_diff = (-20.0*(alpha - 0) - 11.53*(alphad - 0) +  0.00*(beta - 0) +  0.00*(betad - 0)
        + 0.00*(sd - 0));
202    T_sum = (  0.0*(alpha - 0) +  0.00*(alphad - 0) + 64.86*(beta - 0) + 14.99*(betad - 0)
        + 4.99*(sd - 0));
203
204   // Motor current references
205   i_2 = (T_sum + T_diff)/(2*K_T);
206   i_1 = (T_sum - T_diff)/(2*K_T);
207
208   // Saturate current
209   i_1 = limit_value(i_1, -10, 10);
210   i_2 = limit_value(i_2, -10, 10);
211
212   // Request current on ODrive
213   odrive1.SetCurrent(0,-i_1);
214   odrive1.SetCurrent(1,i_2);
215
216 }
217
218 void loop() {
219
220   // Read gyroscope
221   IMU_code();
222
223   // Controller
224   LQR_Control();
225
226 }
227
```

```
// Fourth order Runge-Kutta numerical intergration
float RK4(float x, float xd, float dt) {
    float k1_2, k2_2, k3_2, k4_2;

    k1_2 = dt *  xd;
    k2_2 = dt * (xd + k1_2/2);
    k3_2 = dt * (xd + k2_2/2);
    k4_2 = dt * (xd + k3_2);

    return x + (k1_2 + 2*k2_2 + 2*k3_2 + k4_2)/6;
}

// Saturation function
float limit_value(float value, float min_limit, float max_limit) {

    if (value < min_limit) {
        return min_limit;
    } else if (value > max_limit) {
        return max_limit;
    } else {
        return value;
    }
}
```

I.3 Arduino code for the Arduino UNO

Below the Arduino code including wireless communication and joystick measurements from the Arduino UNO is included.

```
    // Lib for ekstra serial port as 0 and 1 are reserved for serial monitor
#include <SoftwareSerial.h>
// Serial port for HC12
SoftwareSerial HC12(10, 11); // HC-12 TX Pin, HC-12 RX Pin


// Joystick variables
#define VRX_PIN  A0 // Pin connected to VRX pin
#define VRY_PIN  A1 // Pin connected to VRY pin
int sender = 0;
bool stop = 0;

}

void setup() {
  HC12.begin(9600); // Open serial port to HC12 (for wireless)
  Serial.begin(9600); // Start serial monitor
  pinMode(but_PIN, INPUT); // Define button as input
}

void loop() {
  // Read stop button
  stop = digitalRead(2);
  // Read joystick potentiometer, convert value into a set reference for the Teensy MCU
  if (analogRead(VRX_PIN) > 500){
      sender = sender+1;
  }
  else if (analogRead(VRX_PIN) < 400){
      sender = sender-1;
```

```
30   }
31   if (sender > 1023){
32     sender = 1023;
33   }
34   else if (sender < 0){
35     sender = 0;
36   }
37   if (stop == 0){
38     sender = 0;
39   }
40   // Send the stop and joystick value to the Teensy
41   HC12.println(String(stop) +  "," + String(sender) );
42   // Print values in the Arduino serial monitor
43   Serial.println(String(stop) + "," + String(sender) + "," + String(analogRead(VRX_PIN))
       );
44 }
```