University of Cape Town

Department of Mechanical Engineering

Rondebosch, Cape Town
South Africa

2018

Project No:
27

Single-Axis Inertially Stabilised Platform for Smartphone Videography

Author:
Luke D. Leach

A thesis submitted in fulfilment of the requirements for the degree of Bachelor of Science in Engineering

in the

Faculty of Engineering & the Built Environment

October 23, 2018

Keywords:
stabilisation, control, line of sight, inertial, base motion rejection

Supervisor:

Assoc. Prof. Hennie Mouton

Declaration of Authorship

I, Luke LEACH, declare that this thesis titled, "Single-Axis Inertially Stabilised Platform for Smartphone Videography" and the work presented in it are my own. I confirm that:

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- Each significant contribution to, and quotation in, this assignment from the work(s) of other people has been attributed, and has been cited and referenced.
- This assignment is my own work.
- I have not allowed, and will not allow anyone to copy my work with the intention of passing it off as his or her own work.

Signed:

Date:

UNIVERSITY OF CAPE TOWN

Abstract

Faculty of Engineering & the Built Environment
Mechanical Engineering

Bachelor of Science in Engineering

Single-Axis Inertially Stabilised Platform for Smartphone Videography

by Luke LEACH

This report documents the design and construction of a single-axis inertially stabilised platform for smartphone videography. The platform is designed to track base motion at low frequencies and reject base motion at high frequencies - thus producing smooth pans as the user aims the smartphone's camera. Classical control theory is used to design and tune a closed-loop control system that implements two feedback loops; namely: positional feedback (outer tracking loop) and inertial angular velocity feedback (inner stabilisation loop). The device was driven by a brushless DC motor which was ultimately deemed unsuitable for the application. Despite the built prototype not meeting system specifications, the operating principles of the platform could clearly be observed and analysed. The simulation of the system was moderately accurate, with a maximum and minimum normalised root mean square error of 12.43% and 3.45%.

Acknowledgements

Firstly, I would like to thank my supervisor, A/Prof Hennie Mouton for his guidance throughout the project. His truly unparalleled experience with regards to control systems proved invaluable. His prompt responses to emails was testament to his devotion and interest in the project.

Secondly, I would like to thank Peter Kageler of Horne Tech for is vested interest in the project and sound advice on brushless DC motors.

Finally, a special thanks to my mother for her unwavering enthusiasm and support throughout the year.

Contents

List of Figures

List of Tables

List of Abbreviations

ISP	**I**nertially **S**tabilised **P**latform
LOS	**L**ine **O**f **S**ight
FOV	**F**ield **O**f **V**iew
GPS	**G**lobal **P**ositioning **S**ystem
FGS	**F**ine **G**uidance **S**ensor
BLDC	**B**rushless **D**irect **C**urrent
EC	**E**lectronic **C**ommutation
PWM	**P**ulse **W**idth **M**odulated
MEMS	**M**icro-**e**lectro**m**echanical **S**ensor
CE	**C**haracteristic **E**quation
PM	**P**hase **M**argin
GM	**G**ain **M**argin
PID	**P**roportional **I**ntegral **D**erivative
CAD	**C**omputer **A**ided **D**esign
dps	**d**egrees **p**er **s**econd
IMU	**I**nertial **M**easurement **U**nit
DLPF	**D**igital **L**ow**p**ass **Filter**
ADC	**A**nalogue (to) **D**igital **C**onverter
MSB	**M**ost **S**ignificant **B**it

Chapter 1

Introduction

1.1 Background to Research

Since their introduction about 100 years ago, *inertially stabilised platforms* (ISPs) have played a critical role in the aiming of a vast range of objects; objects ranging from cameras to weapon systems. While these platforms are typically mounted to moving vehicles, an increasing number of handheld and surface-mounted ISPs are emerging [1].

Two of the fundamental parts of almost all ISPs is/are the gimbal(s) and the gyroscope(s). A gimbal is a pivoting frame that allows an object mounted to it to rotate relative to the frame. The word gyroscope (or gyro) refers to any device capable of measuring rotational motion in an inertial space [1].

Multiple gimbals are often connected orthogonally to each other to allow for multiple axes of rotation. ISPs typically offer stabilisation in two or three axes. The three axes of rotation are pitch, roll and yaw and can be seen below in Figure 1.1. Each axis of rotation should have a gyroscope mounted to it.

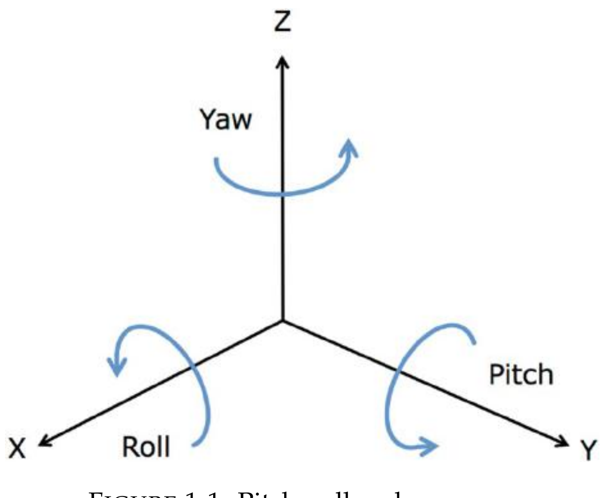


FIGURE 1.1: Pitch, roll and yaw axes

Despite their incredibly broad list of applications, all ISPs have a shared objective: to control the LOS of the object mounted to it relative to another object or inertial space [1]. "Control" includes holding the LOS steady and/or manipulating the LOS.

1.2 Research Objectives

1.2.1 Aims of the Project

The aim of this project is to design, test and build a single-axis ISP for improved smartphone photography and videography. The ISP should control the LOS of the smartphone's camera relative to the inertial space of the earth. The ISP should reduce *jitter* by nullifying any vibration or unintentional movements and allow for smooth pans by tracking intentional rotations.

1.2.2 Performance Requirements

The degree of the project's success will depend on the ISP's adherence to the metrics listed below.
When a step input of $90°$ with a rise time of $0.5s$ is applied to the base:

1. the damping factor of the system should be above 0.707,
2. the settling time of the system should be below $1.5s$, and,
3. at the time of peak base velocity, the velocity of the platform should be below 10% of the base velocity.

The metrics should be met with a payload of up to $200g$ as this is the upper limit of the mass of smartphones[1].
The reasons for these requirements are as follows:

1. It is important that the camera does not oscillate around its target - this stipulates that the damping factor of the system should be as close to one as possible.
2. A delay of longer than $1.5s$ for a $90°$ rotation will make it difficult to keep a moving target in the camera's field of view (FOV).
3. This requirement effectively stipulates that the base motion rejection ratio of the system should be below 10% at high frequencies - meaning that the ratio of the platform velocity to the base velocity should be below 0.1.

1.2.3 Significance of the Study

One of the defining components of a smartphone is its camera. It is a core focus of smartphone developers and is often what gives companies like Apple and Samsung the leading edge. With the increasing ubiquity of smartphones (and therefore handheld cameras), photography has progressively become an integral part of the lives of many people. The rapid advancement of camera sensors and their accompanying software allows inexperienced photographers to take high-quality photographs in a vast array of environments. One of the few aspects of photography that has yet to be automated by smartphones is the stabilisation of the camera's LOS. While high-end smartphones have small-scale translational stabilisation, stabilisation against angular rotations has not been achieved.

A stabilised platform for smartphones is therefore a suitable solution for providing users with a tool to get the most out of their cameras by removing unwanted jitter.

[1]Apple's *iPhone 8 Plus* is one of the largest phones currently on the market and, according to the Apple website, has a mass of $202g$.

The stabilised platform will be particularly useful in applications where the photographer is moving - as is often the case during outdoor activities such as skateboarding, cycling and running.

1.3 Scope and Limitations

The stabilised platform shall provide stabilisation and control in the pitch axis only - not the roll and yaw axes.

The control of the actuator falls outside the scope of this project - therefore an integrated speed controller is used.

This report focuses on the design and tuning of the control loop simulation as well as the construction, testing and improving of the ISP.

1.4 Plan of Development

The report begins by outlining the relevant literature surrounding ISPs. This includes a brief introduction to ISPs and their applications, a look at their basic theory of operation, and finally, a summary of all the components needed for an ISP. The report then moves on to the classical control theory - a fundamental part of ISPs. Following this, the design of the control loop, simulation and physical model is discussed before moving-on to the testing of the system. Lastly, the results from testing are discussed, conclusions are drawn, and recommendations are made for future work.

Chapter 2

Literature Review

The most rudimentary form of an ISP is one that prevents rotation in inertial space around one or more axes. This is something that is common to all ISPs. However, most ISPs have additional functionality depending on their application. These additional functions are discussed further below.

2.1 Industry Applications of ISPs

2.1.1 Target Tracking

ISPs are used to stabilise and manipulate the LOS of targeting sensors that are mounted on platforms that may be subjected to disturbances. The tracking algorithms, along with the control loop, ensure that the sensor's LOS is always pointing at the target despite relative motion between the sensor and the target - and despite disturbances to the sensor's base [2]. Figure 2.1 below illustrates an example of a weapon-pointing system on a ship. In this example, the target (i.e. the jet) remains within the field of view (FOV) of the weapon despite both relative movement between the ship and the jet, and disturbances to the base of the weapon.

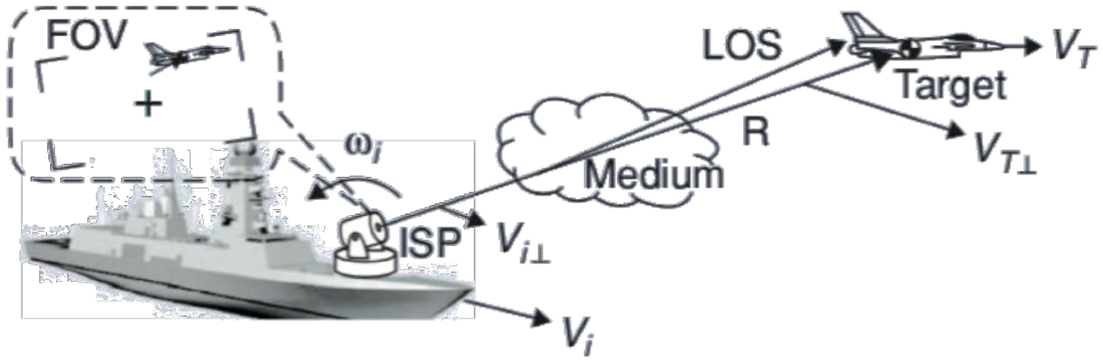


FIGURE 2.1: Weapon-pointing System
Source: [1]

2.1.2 Missile Guidance

ISPs are used to guide missiles to their target without any external communication. These missiles are able to accurately hit their target despite external and internal disturbances such as wind and uneven thrust respectively [3]. To do this, the missile must know its attitude[1] and position. To determine its attitude, a stabilised platform, as shown in Figure 2.2, is used. This platform remains constant in its orientation

[1]The *attitude* of an object is the direction in which it points.

relative to the earth - thus allowing for the missile's attitude to be determined. As for position, accelerometers are used to measure the missile's acceleration in each axis. These accelerations are then integrated with respect to time twice to determine the position of the missile. Modern missile guidance systems now use *global positioning systems* (GPS) to determine position - a more accurate system as each measurement is independent of the previous measurement, and thus errors are not compounded.

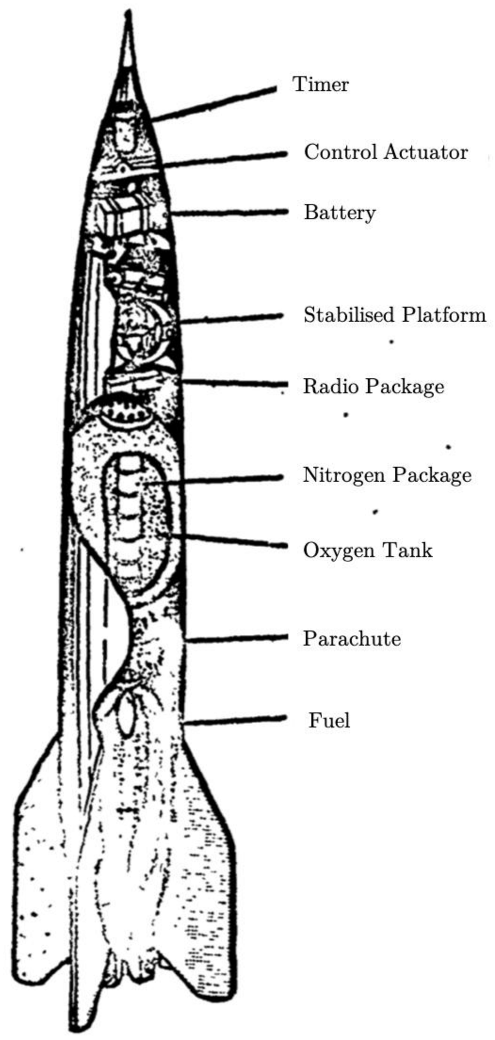


FIGURE 2.2: Early Missile Guidance System: German A-5 Rocket
Adapted from: [3]

2.1.3 Telescopes

Astronomical telescopes depend on ISPs to keep targets in their FOV. Because the distance between the target and the telescope is typically extremely large, it is imperative that the LOS of the telescope can be pointed to within a fraction of a degree of its target. One of the more impressive examples of these pointing control systems is the Hubble Space Telescope. As shown in Figure 2.3 below, it utilises five types of sensors - including gyroscopes and *fine guidance sensors* (FGSs) - to measure the telescope's attitude. These sensors, along with finely-tuned control algorithms, allow for the Hubble telescope to be pointed at its target "with no more than 0.007 arcsecond [1.94×10^{-6} degrees] of deviation over extended periods of time" [4] - a truly incredible feat.

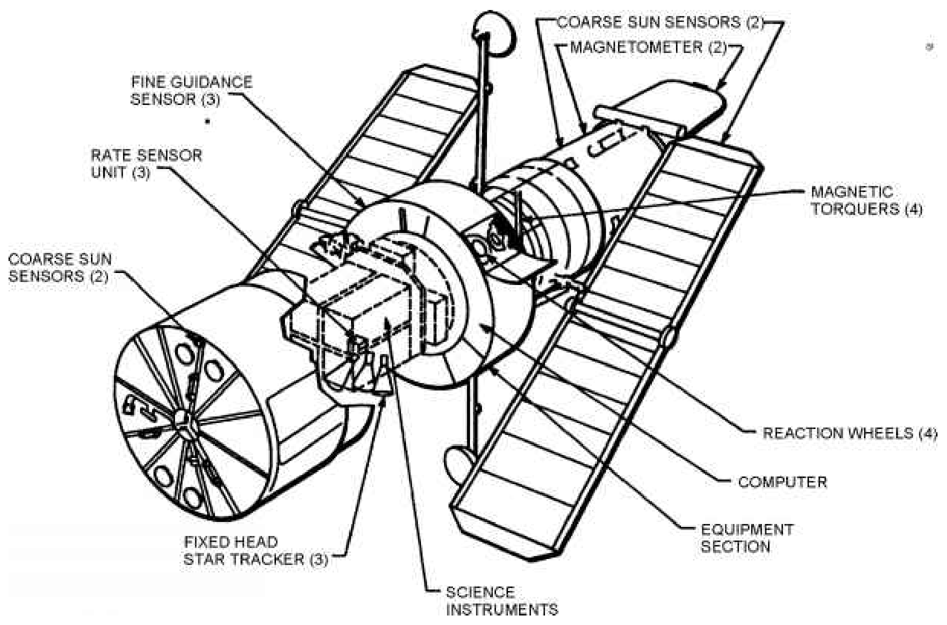


FIGURE 2.3: Hubble Space Telescope: Pointing Control System
[www.aerospacearchives.tk/hubble-space-telescope/pointing-control-subsystem.html]

2.1.4 Camera Systems

More recently, ISPs have been implemented in camera systems. These systems are used to mitigate jitter and produce smooth pans. They were first introduced in larger camera systems used for shooting cinema films but are now increasingly popular in smaller camera systems. They have become especially popular with the recent commercialisation of drones/multi-rotors for aerial photography.

Without the presence of an ISP on a multi-rotor camera system, footage would appear to be out-of-focus and jittery due to the disturbances caused by vibrating motors and propellers as well as the turbulent forces of wind. These cameras systems are often stabilised in all three axes and offer multiple modes of operation. These modes include: a tracking mode, where the camera tracks an object in its FOV; a pan-follow mode where the camera tracks the motion of the multi-rotor body in a smooth manner; and finally, a mode where the LOS of the camera remains inertially fixed.

2.2 Basic Terminology

Figure 2.4 below illustrates the fundamental principles related to ISPs. Despite their wide array of applications, almost all ISPs can described using this model.

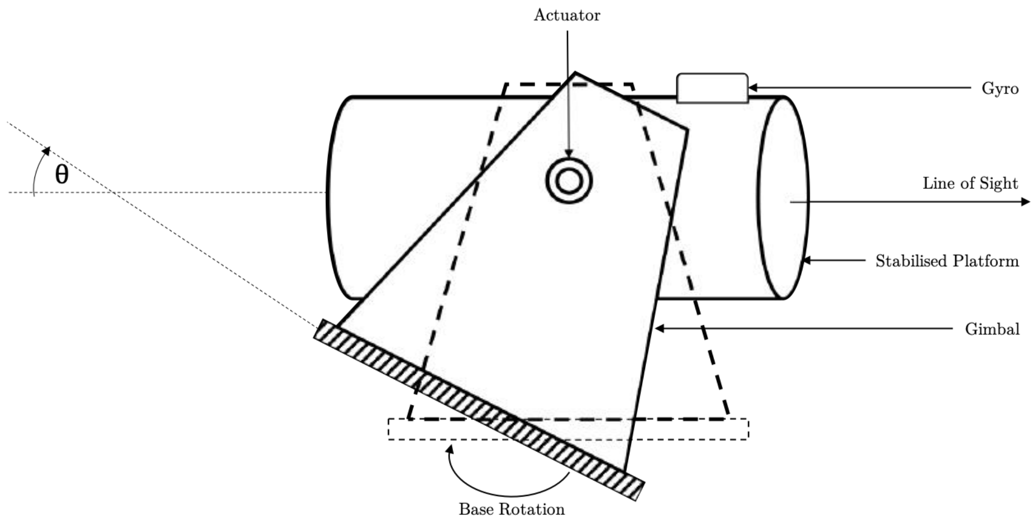


FIGURE 2.4: Generalised ISP
Adapted from: [1]

The terminology associated with this model is described in Table 2.1 below. In addition to this, an example of a multi-rotor system is used to illustrate what each term refers to.

TABLE 2.1: Terminology associated with ISPs

Term	**Description**	**Multirotor Camera System**
Stabilised platform	The platform upon which the object to be stabilised is mounted. This platform's inertial orientation can be controlled.	Camera mount
Line of sight	The direction in which the stabilised object is pointing.	The direction is which the camera is pointing.
Gimbal	The structure that allows for the rotation of the stabilised platform.	3-axis camera gimbal.
Base rotation	Also known as base movement; the rotation of the gimbal frame.	Rotation of the multi-rotor itself.
Gyro	Device capable of measuring angular rate; usually mounted to stabilised platform.	3-axis gyroscope.
Actuator	Device capable of rotating the stabilised platform.	Three orthogonal electric motors.

Table 2.2 describes the more technical terms associated with a typical ISP.

TABLE 2.2: Technical terms associated with ISPs

Term	Symbol	Description
Relative angle	θ	Relative angle between the stabilised platform and the base.
Relative velocity	$\dot{\theta}$	Relative angular velocity between the stabilised platform and the base.
Base velocity	ω_b	Angular velocity of the base.
Platform velocity	ω_p	Angular velocity of the platform.

2.3 Basic Theory of Operation

An ISP that is required to fix the LOS of its payload (i.e. the stabilised object) in inertial space does so by keeping the angular rate of the payload at zero. It does this by measuring the angular rate of the payload (using a gyroscope) and then driving the actuator (usually an electric motor) at the same rate in the opposite direction such that the rotation of the base is counteracted. This occurs at a sufficiently high rate such that the payload appears to be rotationally motionless despite base rotations. This basic theory of operation can be adapted to achieve the aforementioned different functionalities. For example the pan-follow mode of a camera system can be achieved by instructing the relative angle between the base and the stabilised platform to return to zero in a smooth manner - resulting in the LOS of the camera tracking the base motion smoothly.

2.4 Components of a Typical ISP

The following components are required to develop an ISP capable of tracking the base motion.

2.4.1 Actuator

In order to manipulate the LOS of the payload, ISPs use electric motors. It is critical that these motors provide adequate torque and velocity. The torque should be high enough to rotate the entire gimbal and payload structure at a reasonable velocity. Another very important requirement for the motor is its ability to respond to high-frequency noise without reaching saturation. In addition to this, the advent of miniature camera ISPs - such as the ones found on multi-rotors - have driven the demand for more compact and lighter systems. This demand introduces additional requirements on constraints on the actuator. The actuator should be as compact, lightweight, and quiet[2] as possible.

One way in which the required torque of the motor can be reduced is to ensure that the neutral axis of the combined payload and gimbal is aligned with the axis of rotation of the motor. In other words, the center of gravity of the gimbal and payload

[2]High levels of noise interferes with the audio recording of the camera.

combined should be aligned with the output shaft of the motor. This can be proven using the *parallel axis theorem*:

$$I_o = \bar{I} + m \cdot d^2$$

where I_o is the moment of inertia about the rotational axis of the motor, $\bar{I}$ is the moment of inertia of the gimbal and payload about their own centre of gravity, m is the mass of the gimbal and payload, and d is the distance between the centroid of the gimbal-payload structure and the motor axis.

By aligning the centre of gravity of the gimbal-payload structure with the motor axis, the d term goes to zero - therefore reducing the overall moment of inertia. Since the torque required to rotate an object is the product of the object's moment of inertia and angular acceleration, reducing the moment of inertia results in a reduction in the torque required.

Due to the above requirements, a common choice for ISP actuators is the *brushless direct current* (BLDC) motor.

BLDC motor

As the name implies, BLDC motors do not have any brushes and have to be commutated by other means. This is the biggest disadvantage of brushless motors. Their control is significantly more complicated than brushed motors because they have to be commutated manually. While it is possible to commutate a brushless motor using a microcontroller alone, this is a complicated process which falls outside of the scope of the project. Typically, dedicated BLDC motor controllers are used to commutate the motor.
According to [5], the main advantages of brushless motors over similarly sized brushed and induction motors are the following:

- Better speed vs. torque characteristics
- Higher dynamic response
- Higher efficiency
- Longer operating life
- Noiseless operation
- Higher speed ranges
- Higher torque to size ratio

Many of these advantages are highly pertinent to the ISP application. A high dynamic response is critical in ensuring that the motor is able to reject high-frequency disturbances without saturating. Increased efficiency allows for longer flight-time in the case of multi-rotors. Noiseless operation reduces interference with the camera's microphones. Perhaps the most important advantage is the higher torque to size ratio which allows for more compact and lightweight platforms.

BLDC motor controller

As mentioned above, BLDC motors have to be commutated manually through a process known as *electronic commutation* (EC). EC involves manually passing current through each of the electromagnets at the correct time to induce steady rotation [6]. It is therefore important to know the position of the permanent magnet relative to the electromagnets at all times so that the correct coils are energised next. There are two schemes that can be used to drive the motor: (1) where all three coils are energised at once; and (2), where only two coils are energised at any given time [6]. Only scheme (2) will be discussed further as it is the most popular method of commutation.

The majority of brushless controllers use three *hall effect sensors*[3] embedded in the stator to determine the position of the permanent magnet relative to the electromagnets [5]. Depending on the output of these sensors, the controller will activate a pair of transistors which, in turn, energise the next set of coils (or electromagnets). A control block diagram for one of these controllers is shown below in Figure 2.5.

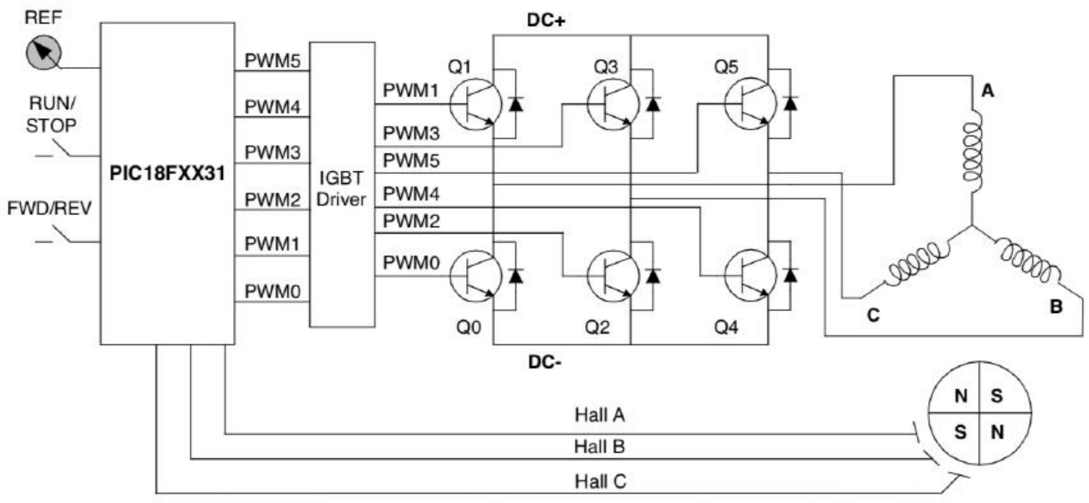
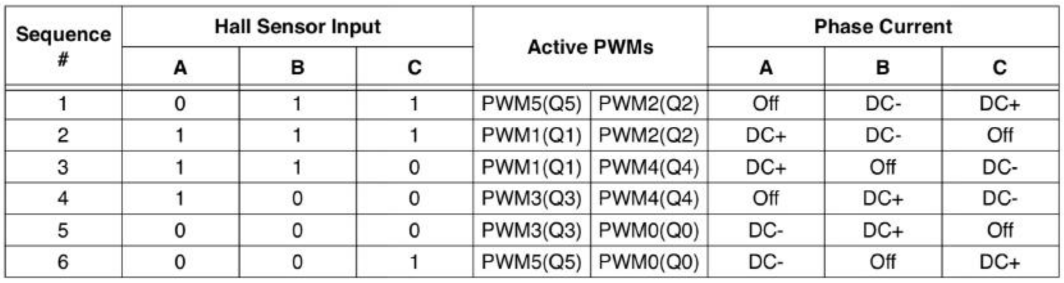


FIGURE 2.5: BLDC control block diagram [5]

Table 2.3 below describes the commutation sequence for counter-clockwise (CCW) rotation of the motor. The number of hall effect sensors is always equal to the number of electromagnets - in this case three. There are therefore six unique permutations of hall effect sensor outputs - resulting in a six-step commutation sequence. Depending on the hall effect sensor feedback, each step activates a different pair of *pulse width modulated*[4] (PWM) signals which consequently drives current through the correct pair of coils.

[3]Sensors that produce a potential difference across them in the presence of a magnetic field.
[4]High frequency modulation of a digital signal to produce a varying effective output voltage.

TABLE 2.3: PWM sequence for CCW rotation of motor [5]



Sequence #	Hall Sensor Input			Active PWMs		Phase Current		
	A	B	C			A	B	C
1	0	1	1	PWM5(Q5)	PWM2(Q2)	Off	DC-	DC+
2	1	1	1	PWM1(Q1)	PWM2(Q2)	DC+	DC-	Off
3	1	1	0	PWM1(Q1)	PWM4(Q4)	DC+	Off	DC-
4	1	0	0	PWM3(Q3)	PWM4(Q4)	Off	DC+	DC-
5	0	0	0	PWM3(Q3)	PWM0(Q0)	DC-	DC+	Off
6	0	0	1	PWM5(Q5)	PWM0(Q0)	DC-	Off	DC+

2.4.2 Angular Rate Sensor

A fundamental requirement of any ISP is the ability to measure the angular velocity of the stabilised platform. This allows the angular velocity of the platform to be controlled using closed-loop control. The sensor used to measure angular rate is known as a gyroscope.

MEMS gyroscope

A MEMS (micro-electromechanical system) gyroscope is a miniature *inertial absolute sensor*[6] because it measures absolute angular velocity relative to an inertial set of axes (i.e. the earth). These sensors output a voltage or digital signal proportional to their angular velocity. They often come in 3-axis packages meaning that one gyroscope package can be used for a 3-axis ISP. The gyroscope is typically attached directly to the stabilised platform - thus measuring the absolute inertial angular velocity of the platform and the payload.

2.4.3 Relative-Motion Transducers

In addition to the gyroscope, some ISPs require sensors that are capable of measuring relative motion between the gimbal/base and the platform/payload. This is critical in applications where the payload needs to be pointed relative to the base - such as in camera systems implementing pan-follow mode.

2.4.4 Tracking Sensor and Algorithm

ISPs that are required to track a certain target - such as the ones found in advanced telescopes and in weapon systems - need a sensor that is able to feedback the position of the target. Because it is a complex task to determine the relative position of an object, the tracking sensor is usually paired with a tracking algorithm that takes the raw data from the sensor and outputs a position vector. This position vector can then be used as feedback in the control loop.

2.4.5 Microcontroller

The microcontroller is responsible for interpreting and manipulating the data received from the sensors. For example, the microcontroller will have to convert the voltage reading from the potentiometer's wiper into an angle. Additionally, the microcontroller is responsible for implementing the entire control loop and generating the PWM drive signals for the motor controller.

Chapter 3

Control Theory Development

Note: The information in this entire chapter, unless otherwise indicated, is adapted from the University of Cape Town's MEC4107S - Fundamentals of Control Systems course notes written by A/Prof Hennie Mouton [7].
All plots (throughout the rest of the report) were generated using MATLAB. Control systems are found in the majority of technological devices used throughout society today. They are responsible for ensuring that these devices function as expected and are thus a fundamental component of engineering design.

Examples of everyday devices/systems that require control are:

- Temperature control of ovens and refrigerators.
- Cruise-control of vehicles.
- Position control of sun-tracking solar-panels.
- Temperature control of a toaster.
- Speed control of a household fan.

To ensure correct functioning, devices implement control loops. These control loops are at the heart of all control systems and are discussed further below.

3.1 Types of Control Loops

There are two types of control loops - *open-loop* and *closed-loop*. The first three examples listed above are considered closed-loop control systems while the last two examples are considered open-loop control systems. The fundamental difference between closed- and open-loop systems is the addition of feedback to a closed-loop system.

3.1.1 Closed-loop

Figure 3.1 below illustrates the general structure of closed-loop control.

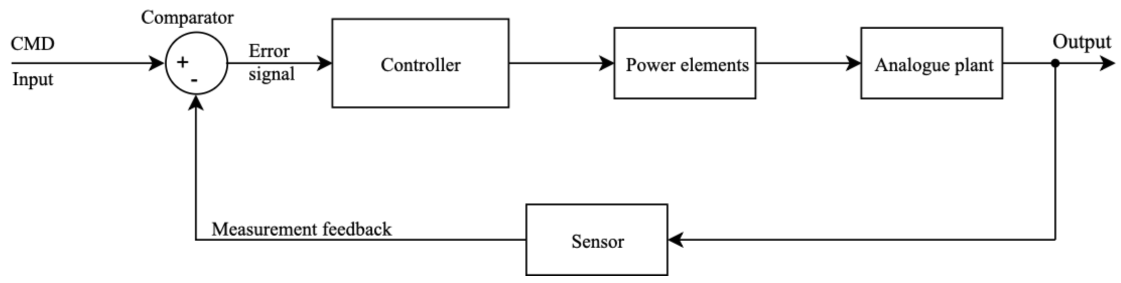


FIGURE 3.1: Closed-loop control block diagram

It consists of the following components:

- Command input:
  The command (CMD) input (sometimes referred to as the setpoint (SP)) is the value that the output attempts to follow.

- Feedback:
  Feedback is what distinguishes a closed-loop control system from an open-loop one. It is a measurement of the output value and is used to ensure that the output follows the input command.

- Comparator:
  A comparator is a term used to describe the process of calculating the difference between two signals. With reference to Figure 3.2, a comparator can be described by the following equation: $c = a - b$.

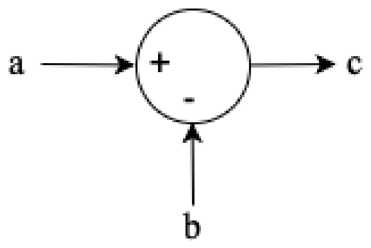


FIGURE 3.2: Comparator block

- Error signal:
  The error signal comes from the output of the comparator. In other words, it is the difference between the command and the measured output.

- Sensor:
  The sensor is the device used to measure the output value (e.g. gyroscope, accelerometer, potentiometer etc.).

- Controller:
  The controller is responsible for converting the error signal into an output signal that will drive the power elements in a such a way that the error signal goes to zero. It is ultimately what determines the performance of the control

loop. While there are many types of controllers, the simplest and most ubiquitous type is the *proportional-integral-derivative* (or PID) controller. Derivatives of the PID controller include the P (proportional) controller, the PI (proportional-integral) controller, and the PD (proportional-derivative) controller. Each of these types of controllers have their advantages and disadvantages which will be discussed in subsequent chapters.

- Output:
  The output represents the actual value of the variable of interest[1].
- Analogue plant:
  The analogue plant (often referred to as the plant) is a model of the real-world system dynamics and kinematics. The plant essentially models the relationship between the power elements output and its resulting real world effect on the variable of interest.
- Power elements:
  The power elements are the components that can be controlled to cause changes to the plant. Typical power elements would be motors, amplifiers, and pumps. The power elements can often be included in the plant.

The functioning of a control loop is best explained using an example. The example of the temperature control of an oven is described below with reference to Figure 3.3.

The user sets the command or desired temperature of the oven using the oven's user interface. The command temperature is then compared with the actual temperature of the oven measured by a temperature sensor. The difference between the command temperature and the actual temperature is called the error. This error goes through a specially designed controller which manipulates the error into a suitable voltage across the heating element of the oven. This voltage causes current to flow which, in turn, causes the heating element to produce heat. This heat causes the temperature of the oven to increase. This increased temperature is measured by the temperature sensor and, once again, compared to the command temperature. Given that the oven has power, this process will repeat itself continuously. Eventually, the actual oven temperature will reach the command temperature resulting in an error of zero. Consequently, the voltage across the heating element will be zero and the temperature of the oven will stop increasing. After time, however, the temperature of the oven will drop. This will, once again, result in an error, causing the heating element to increase the temperature again. In this way, the output follows the input command.

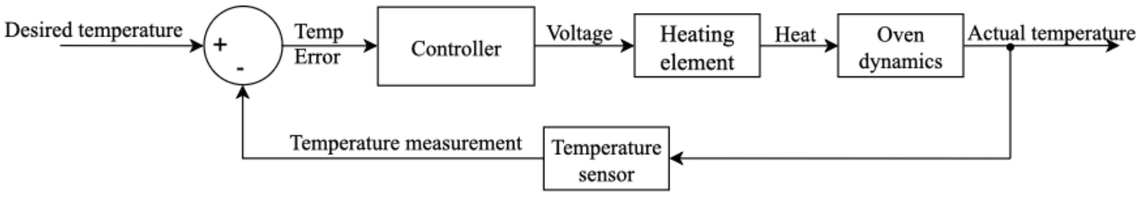


FIGURE 3.3: Temperature control of an oven

[1]The variable of interest refers to what is being controlled (e.g. speed, temperature, position etc.)

3.1.2 Open-loop

Figure 3.4 below illustrates the general structure of open-loop control. Open-loop control is different from closed-loop control in that there is no feedback loop.

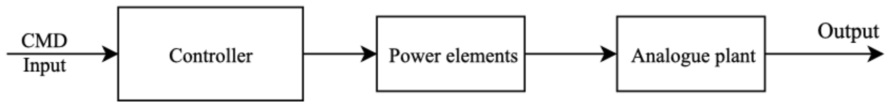


FIGURE 3.4: Open-loop control block diagram

Open-loop control is used in certain applications where the one or both of the following statements is true:

- The accuracy of the output is not particularly important.
- The system is exposed to very few environmental changes/disturbances.

Consider the example of the temperature control of a toaster. A voltage is applied across the heating elements to make them heat-up. This voltage is constant over the entire duration of its operation. There is no indication of what the actual temperature of the elements is. Therefore, the exact same toaster operating in a very cold environment will produced toast that is less cooked than that of a toaster in a very warm environment.

3.1.3 Advantages of Close-Loop Control

Oone of the biggest advantages of closed-loop control over open-loop control is that a closed-loop system will follow the command even if there are external disturbances or environmental changes that affect the system. Consider the example of a car's cruise-control system. Open-loop control would be suitable if the car were to only ever travel on level roads. If, however, the car was required to travel up a hill, the speed of the car would drop. In this situation, closed-loop feedback would be required to maintain the car's speed.

Feedback is therefore useful for the following reasons:

- It reduces the effect of environmental change such as changes in temperature and pressure.

- It reduces the effect of disturbances to the system such as external torques or forces applied to the system.

- It allows the system to correct itself more quickly in response to disturbances.

3.2 Block Diagram Algebra

In order to design and develop control systems, block diagrams are used to illustrate relationships between different components of the control system. Block diagrams are thus a crucial part of control theory.

Each block typically represents a *transfer function* that determines the relationship between the input and output signals of the block. These transfer functions are usually defined in Laplace's s-domain for reasons which will be discussed further in Section 3.3. Again, block diagrams are best understood with the aid of an example.

Figure 3.5 is an example of a block diagram implementing closed loop control. The associated equations are detailed below.

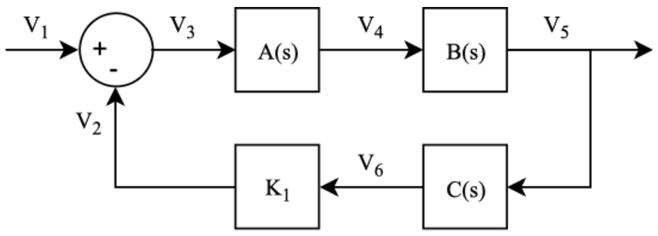


FIGURE 3.5: Example closed-loop block diagram

Associated equations:

$$V_3 = V_1 - V_2$$
$$V_4 = V_3 \times A(s)$$
$$V_5 = V_4 \times B(s)$$
$$V_6 = V_5 \times C(s)$$
$$V_2 = V_6 \times K_1$$

From these equations, or from inspection, one can determine the *closed-loop transfer function* which is defined as the output (V_5) divided by the input (V_1):

$$V_5 = [V_1 - (C(s) \cdot K_1)V_5] \cdot A(s) \cdot B(s)$$

$$\Rightarrow V_5 \cdot (1 + A(s) \cdot B(s) \cdot C(s) \cdot K_1) = (A(s) \cdot B(s))V_1$$

$$\Rightarrow \frac{V_5}{V_1} = \frac{A(s) \cdot B(s)}{1 + A(s) \cdot B(s) \cdot C(s) \cdot K_1}$$

$A(s) \cdot B(s)$ is known as the *forward transfer function* and $A(s) \cdot B(s) \cdot C(s) \cdot K_1$ is known as the *open-loop transfer function*.
Therefore:

$$\text{Closed-loop transfer function} = \frac{\text{Forward transfer function}}{1 + \text{Open-loop transfer function}}$$

This is relationship is derived from *Mason's rule* and it is valid for any single loop block diagram. It is therefore useful to manipulate block diagrams such that there is only a single loop. However, this is not always possible - especially when considering more complicated control systems.

The open-loop transfer function is particularly important because it determines the stability of the system. Therefore, once stability is achieved, it is important that the open-loop transfer function is not changed while manipulating the block diagram.

For example, if a gain[2] of K_2 had to be added to the system (see Figure 3.5) between the transfer functions $A(s)$ and $B(s)$, the open loop transfer function would become: $A(s)K_2B(s)C(s)K_1$. A factor of $\frac{1}{K_2}$ would have to be added in the feedback loop to bring the open loop transfer back to $A(s)B(s)C(s)K_1$.

3.3 Laplace Transforms

Also known as *s-transforms*, the Laplace transform provides an effective way to analyse the dynamics of control systems. It is used when dealing with the continuous signals associated with analogue systems such as motors and analogue electronics. The definition of the Laplace transform is as follows:

$$F(s) = L\{f(t)\} = \int_0^\infty f(t)e^{-st}\mathrm{d}t \tag{3.1}$$

where $f(t)$ is an arbitrary function in the time domain.

By convention, functions in the time domain are written in lower case and the corresponding function in the s-domain is written in upper case. For example: $g(t) \Rightarrow G(s)$.

The purpose of the Laplace transform is to take a signal from the time domain and represent it in the frequency (or s-) domain. Each point in the s-domain corresponds to a waveform in the time domain through the equation e^{st}. The s-transform enables one to calculate and manipulate transfer functions using algebra as opposed to relatively complicated differential equations .

eck this statement

3.3.1 Useful Laplace transforms

Integration

$$\begin{aligned}
L\left[\int f(t)\mathrm{d}t\right] &= \int_0^\infty \left[\int f(t)\mathrm{d}t\right] e^{-st}\mathrm{d}t \\
&= \left(\int f(t)\mathrm{d}t \cdot -\frac{1}{s}e^{-st}\right)\Big|_0^\infty - \int_0^\infty f(t)\left(-\frac{1}{s}\right)e^{-st}\mathrm{d}t \\
&= \frac{1}{s}F(s) + \frac{1}{s}\int f(t)\mathrm{d}t\Big|_{t=0}
\end{aligned}$$

If it is assumed that the initial output of the integrator is zero, the last term can be ignored. Therefore, $\frac{1}{s}$ represents integration in the s-domain.

Differentiation

$$\begin{aligned}
L\left[\dot{f}(t)\right] &= \int_0^\infty \dot{f}(t)e^{-st}\mathrm{d}t \\
&= \left(f(t)e^{-st}\right)\Big|_0^\infty - \int_0^\infty f(t)(-s)e^{-st}\mathrm{d}t \\
&= sF(s) - f(0)
\end{aligned}$$

[2]Gain is defined as the ratio of the magnitude of the output to the magnitude of the input at steady-state.

If it is assumed that the initial output of the differentiator is zero, the last term can be ignored. Therefore, s represents differentiation in the s-domain.

3.3.2 Relationship between s and frequency

A transfer function in the s-domain can be represented as a frequency transfer function by substituting s with $i\omega$. It is known that $\omega = 2\pi f$; where f is frequency in *Hertz* (*Hz*). Consequently, s is directly proportional to frequency. Therefore, s can be replaced by zero at DC and infinity at extremely high frequencies.

3.3.3 Gain and phase

The gain of a transfer function can be found by calculating the modulus (length) of the vector formed by the transfer function in the Gaussian plane.

The phase shift caused by a transfer function can be found by calculating the angle formed between the aforementioned vector and the positive real axis.

Consider the transfer function:

$$\frac{As+1}{s^2+Bs+1}$$

Substitute s with $i\omega$:

$$\frac{1+(A\omega)i}{(1-\omega^2)+(B\omega)i}$$

The gain of this transfer function is:

$$\frac{\sqrt{(1)^2+(A\omega)^2}}{\sqrt{(1-\omega^2)^2+(B\omega)^2}}$$

And the phase is:

$$\arctan\left(\frac{A\omega}{1}\right)-\arctan\left(\frac{B\omega}{1-\omega^2}\right)$$

Gains are often converted to *decibels* (dB). This is done by taking the *log* of the gain and multiplying it by 20.

Therefore, the gain in dB is: The gain of this transfer function is:

$$20\log\left(\frac{\sqrt{(1)^2+(A\omega)^2}}{\sqrt{(1-\omega)^2+(B\omega)^2}}\right)$$

3.4 z-transforms

While the s-transform is extremely useful when dealing with continuous analogue signals, it is not suitable when dealing with discrete digital signals. This is where the z-transform is used. The z-transform is used to represent discrete signals that are found in digital systems such as microcontrollers and other digital components.

From these z-transforms, *difference equations* can be formed and implemented on microcontrollers. These difference equations will be covered in Section 3.9.

3.4.1 Definition of the z-transform

Figure 3.6 below depicts the sampling of the continuous signal, $f(t)$ at discrete time intervals $(T, 2T, 3T, \ldots, nT)$.

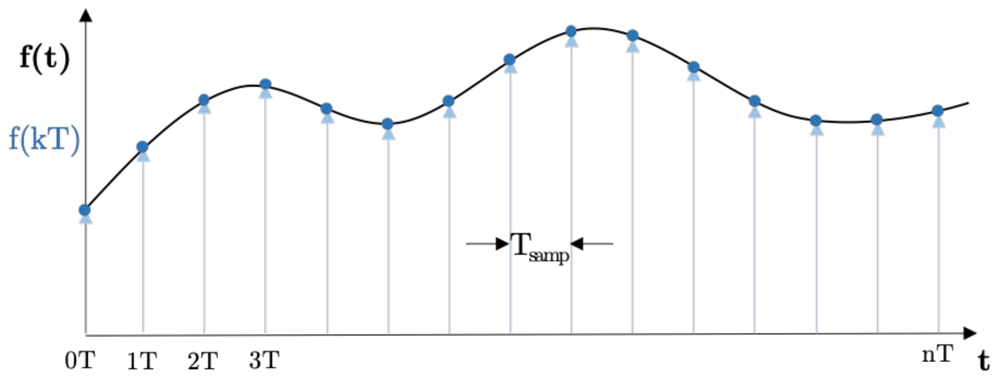


FIGURE 3.6: Sampling of continuous signal.

$f(kT)$ is represents the values of $f(t)$ at at different time intervals and is illustrated by the blue dots. k can be any integer from zero to infinity and T is the sampling period in seconds.
The formal definition of the z-transform is thus:

$$F(z) = Z\left[f(kT)\right] = \sum_{k=0}^{\infty} f(kT) \cdot z^{-k} \tag{3.2}$$

$$= f(0) + f(T)z^{-1} + f(2T)z^{-2} + f(3T)z^{-3} + \ldots \tag{3.3}$$

where $\left|z^{-1}\right| < 1$

Again, by convention, functions in the time domain are written in lower case and the corresponding function in the z-domain is written in upper case. For example: $g(t) \Rightarrow G(z)$.

3.5 Relationship between the s-domain and the z-domain

3.5.1 Accurate relationship

By comparing a delay of T in the s-domain with the same delay in the z-domain, it can be shown that $z^{-1} = e^{-st}$. Although this is an accurate equation, it is rarely used when converting between the s- and z-domains due to its complexity. Therefore, approximate relationships are used instead.

3.5.2 Approximate relationships

Consider the arbitrary time signal shown in Figure 3.7.
Let $g(t) = \int f(t)\mathrm{d}t$.
Therefore, $g((k-1)T)$ is equal to the area of the shaded orange region.

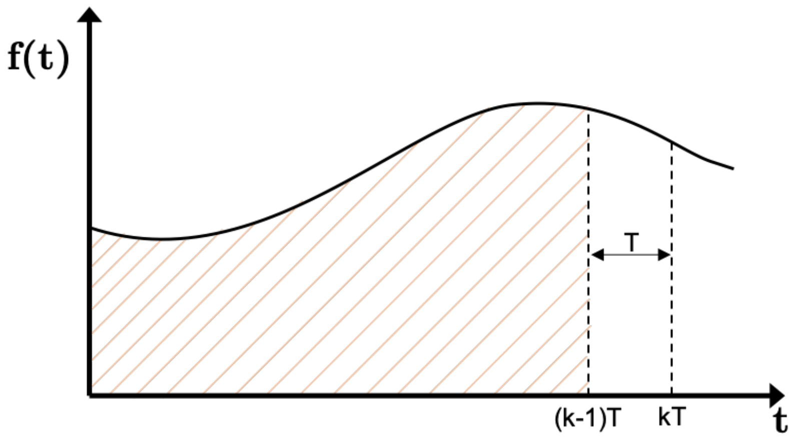


FIGURE 3.7: Arbitrary signal, $f(t)$

The following three methods can be used to approximate $g(k)$ and therefore the relationship between s and z:

Left-hand rectangular rule:

$$\begin{aligned}
&g(k) = g(k-1) + T \cdot f(k-1) \\
&\Rightarrow G(z) = z^{-1} \cdot G(z) + T \cdot z^{-1} \cdot F(z) \\
&\Rightarrow \frac{G}{F}(z) = \frac{T \cdot z^{-1}}{1 - z^{-1}}
\end{aligned}$$

It is known that $\frac{G}{F}(s) = \frac{1}{s}$

Therefore:

$$s = \frac{1}{T} \cdot \frac{1 - z^{-1}}{z^{-1}}$$

Right-hand rectangular rule:

$$\begin{aligned}
&g(k) = g(k-1) + T \cdot f(k) \\
&\Rightarrow G(z) = z^{-1} \cdot G(z) + T \cdot F(z) \\
&\Rightarrow \frac{G}{F}(z) = \frac{T}{1 - z^{-1}}
\end{aligned}$$

It is known that $\frac{G}{F}(s) = \frac{1}{s}$

Therefore:

$$s = \frac{1}{T} \cdot (1 - z^{-1})$$

Trapezium rule (Bilinear transform):

$$\begin{aligned}
g(k) &= g(k-1) + T \cdot \frac{f(k-1)+f(k)}{2} \\
\Rightarrow G(z) &= z^{-1} \cdot G(z) + \frac{T}{2}\left(z^{-1} \cdot F(z) + F(z)\right) \\
\Rightarrow \frac{G}{F}(z) &= \frac{T}{2} \cdot \frac{1+z^{-1}}{1-z^{-1}}
\end{aligned}$$

It is known that $\frac{G}{F}(s) = \frac{1}{s}$

Therefore:

$$s = \frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}}$$

The last rule - the bilinear transform - is the most accurate of the three approximations and is for this reason more commonly used.

3.6 Control Loop Analysis

Once a control system's plant has been modelled, the next step is to analyse the system. There are two approaches to this:

1. Mathematical analysis
2. Simulation analysis

3.6.1 Mathematical Analysis

An important aspect of mathematically analysing a control loop is the establishment of the *poles* and *zeros* of the system.

Consider the transfer function $G(s)$:

$$\begin{aligned}
G(s) &= \frac{A(s)}{B(s)} \\
&= \frac{(s-a_1)(s-a_2)}{(s-b_1)(s-b_2)(s-b_3)}
\end{aligned}$$

The zeros of $G(s)$ are the where $A(s) = 0$ and the poles of $G(s)$ are where $B(s) = 0$.
Therefore the zeros are: $s = a_1,\ a_2$ and a_3,
and the poles are: $s = b_1,\ b_2$ and b_3.

While both zeros and poles are needed to characterise the overall response of the system to an input, the poles are what determine the homogeneous response, and therefore the stability, of the system.
If any poles are located to the right of the imaginary axis the system will be unstable.
If all the poles are located to the left of the imaginary axis then the system will be stable. This can be shown by considering the fact that any point in the s-domain

produces a waveform of e^{st} in the time-domain. Therefore, a pole to the right of the imaginary axis will result in an exponentially growing homogeneous response (unstable); and a pole to the left of the imaginary axis will result in an exponentially decaying homogeneous response (stable)[8].

Root locus plots

One of the most effective ways to visualise the effects of the poles of a system is to construct a *root locus* plot. Root locus plots are made by plotting the values of s that satisfy the *characteristic equation* (CE). While these values are technically poles, they are often referred to as roots too. The CE is defined as follows:

$$1 + \textit{open loop transfer function} = 0$$

This equation comes from the aforementioned closed-loop transfer function formula:

$$\textit{closed loop transfer function} = \frac{\textit{forward transfer function}}{1 + \textit{open loop transfer function}}$$

It is clear that, when the denominator of this equation goes to zero, the closed loop transfer function will potentially run-away to infinity and the system will be unstable.

Root locus plots trace the paths that the poles of the system will follow as the gain is changed. As described above, a system is only stable if all of its poles are left of the imaginary axis.

Consider the block diagram below:

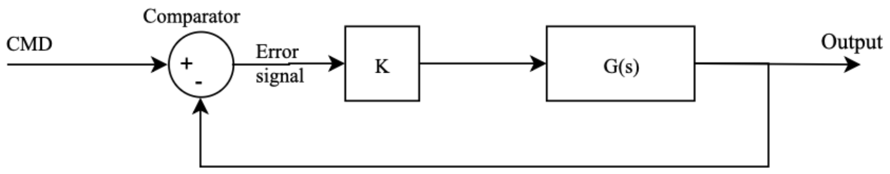


FIGURE 3.8: Example block diagram

The open loop transfer function is $K \cdot G(s)$.
The CE is therefore $1 + K \cdot G(s) = 0$.

$$\text{Let } G(s) = \frac{(s+20)(s+40)}{(s+50)(s^2+10s+100)(s^2+40s+1000)}$$

A root locus plot of $K \cdot G(s)$ yields the following:

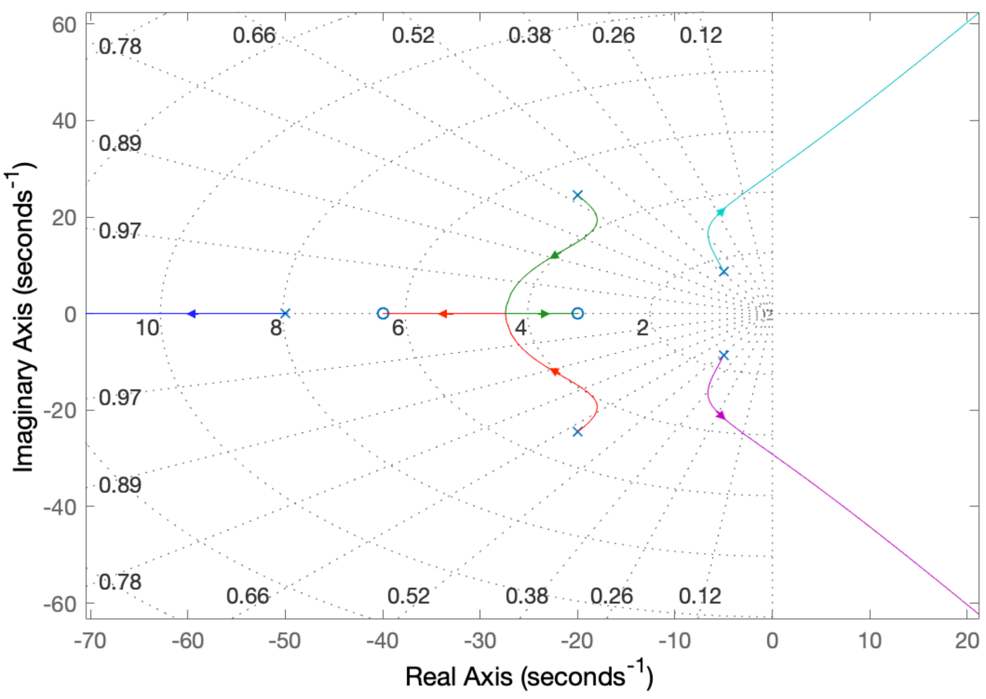


FIGURE 3.9: Root locus plot

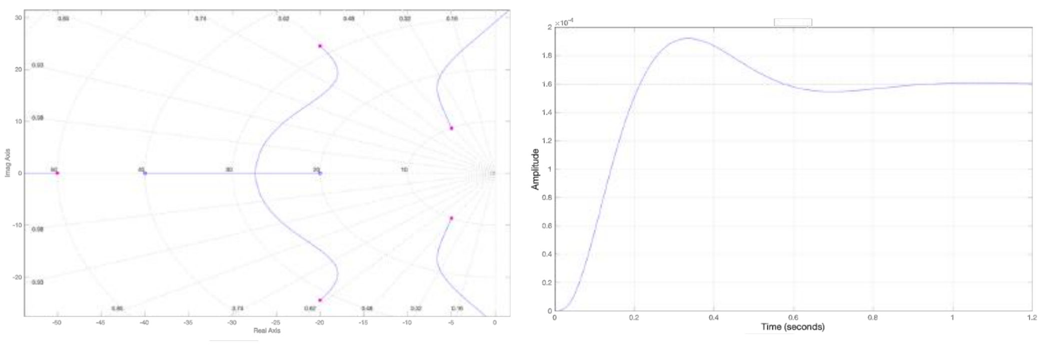
As expected, there are five poles (four of which form conjugate pairs) and two zeros represented by crosses and circles respectively. As the gain K is increased, the positions of the poles follow the lines in the directions indicated by the arrowheads. It is clear that, if the gain is increase too far, the turquoise and magenta poles will cross the imaginary axis and the system will become unstable. In addition to the stability of the system, root locus plots can determine other factors that characterise the response.

The real component of the pole determines the settling time of the response. The larger the magnitude of the real component (i.e. the further away from the imaginary axis), the faster, more aggressive the response will be.

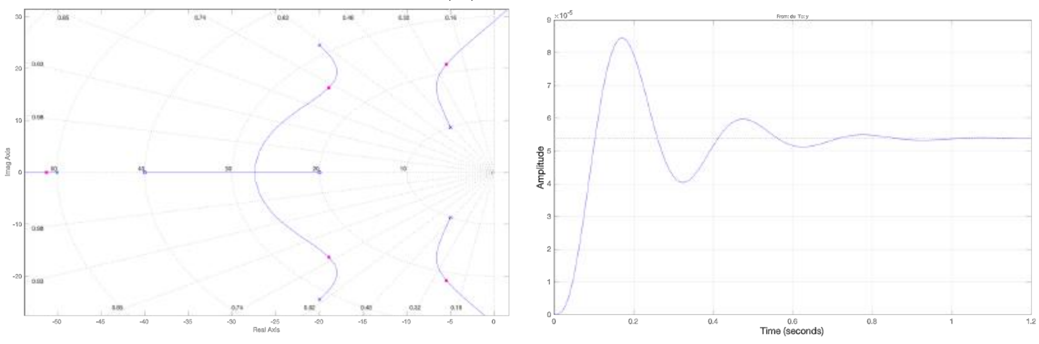
The imaginary component of the pole is what determines the natural frequency of the response. The larger the magnitude of the imaginary component (i.e. the further away from the real axis), the higher the natural frequency.

Furthermore, by considering the maximum angle between the poles and the real axis, one can determine the damping ratio of the response. An angle of zero (i.e. a pole on the real axis) results in a damping ratio of one while an angle of 90 degrees (i.e. a pole on the imaginary axis) results in a damping ratio of zero.

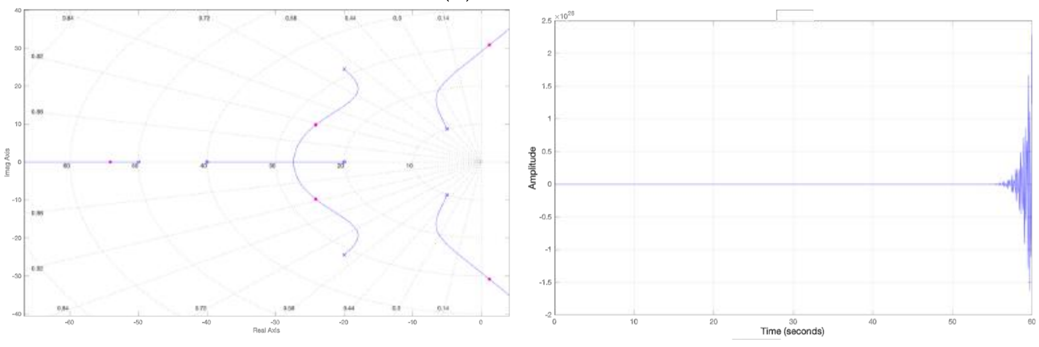
The following three sets of graphs demonstrate the statements above by showing the response of the system to a step input with varying values of K.



(A) $K = 1.027$



(B) $K = 12325$



(C) $K = 37528$

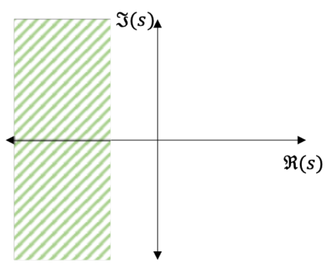
FIGURE 3.10: Root locus and corresponding step response. Positions of poles are indicated by the pink dots.

As expected, the position of the poles in Figure 3.10c corresponds with an unstable response. This is due to two of the poles lying right of the imaginary axis.

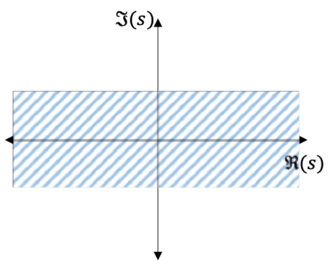
The oscillatory response shown in Figure 3.10b indicates a decrease in damping which is caused by an increase in the maximum angle between the poles and the negative real axis.

The utility of the root locus plot can especially be seen when given system requirements. For example, if constraints are implemented on: (A) the maximum exponential decay time, (B) the maximum natural frequency, and (C) the minimum damping ratio of the system, then one can determine the area of the root locus plot that the poles must fall within.

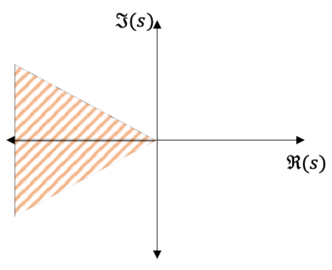
Figure 11.6 below depicts the areas that the poles should fall within to ensure that the requirements stipulated above are met. The poles should fall within the shaded areas.



(A) Exponential decay time



(B) Natural frequency



(C) Damping Ratio

FIGURE 3.11: Shaded areas represent where, given that the poles fall within these areas, the system will meet the stipulated requirements.

When all of these requirements are combined, one can visualise the area within which the poles must lie to ensure that all of the system requirements are met. This area is illustrated in Figure 3.12.

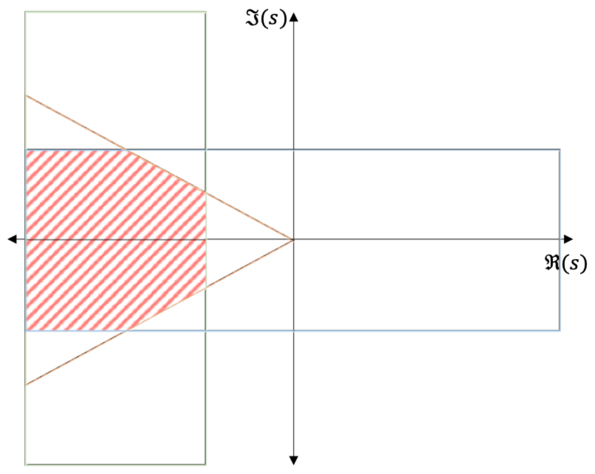


FIGURE 3.12: The red shaded area represents the area that the poles must fall within in order to meet system specifications.

3.6.2 Simulation Analysis

While the mathematical approach of using root locus plots to determine the characteristics of the system's response is very effective, it is sometimes difficult to determine the transfer function that accurately represents the entire system - especially when dealing with complicated systems with multiple loops. Therefore, using simulations is often a more feasible method for designing a control loop.

In order to simulate the system, the block diagram must be implemented in code. There are several simulation packages available that assist in this implementation by offering pre-defined functions such as comparators, integrators and compensators. Some packages, such as MATLAB's *Simulink* allow the user to simulate the system directly from the block diagram. Simulink will be used in all subsequent simulations.

The most important aspect of control loop design through simulation is the ability to generate *bode plots*. When a signal is applied to the input of a transfer function, the output can undergo a change in amplitude and/or a delay in time. The output amplitude divided by the input amplitude is known as *gain* and the difference (in degrees/radians) between the input and output is known as the *phase* shift. A bode plot is a graph that plots the *gain* and *phase* of a transfer function at different frequencies.

Stability margins

Stability margins is a collective term for the *gain margin* and *phase margin* of a system. These two margins indicate how much "protection" the system has from instability. The reason that stability margins are used is to increase the robustness of the system by accounting for process variations. Process variations refer to small changes in the dynamics of the system that cannot be predicted.

In order to properly understand stability margins, it is important to understand exactly what makes a system unstable.

As discussed above, a closed loop system will become unstable when the denominator (i.e. $1 + \textit{open loop transfer function}$) is equal to zero and the pole is right of the imaginary axis. Another way of describing this is: the system will be unstable if the open-loop gain is equal to one and the phase is equal to 180 degrees (or -180 degrees[3]) at a given frequency. Therefore, in order for the closed-loop system to be stable, the gain of the open loop must never equal 1 (or 0 dB[4]) when the phase is equal to -180 degrees over the whole range of expected operating frequencies. In fact, in order to account for the previously mentioned process variations, the phase at the zero dB crossover frequency should be well above -180 degrees and the gain at the -180 degree crossover frequency should be well below zero dB. This leads to the definitions of the stability margins.

The phase margin (PM) and gain margin (GM) are defined as follows:

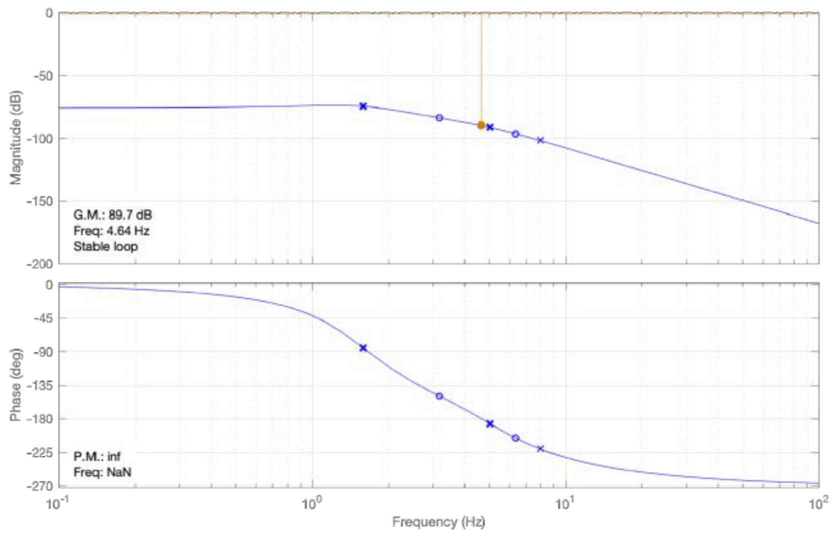
$$\begin{aligned} PM &= \textit{Phase @ 0 dB crossover frequency} - (-180^\circ) \\ GM &= \textit{Gain @ } -180^\circ \textit{ crossover frequency} \times -1 \end{aligned}$$

While stability margins are application specific, generally one should aim to make the margins as big as possible. Doing this, however, often comes at the cost of other performance metrics such as bandwidth.

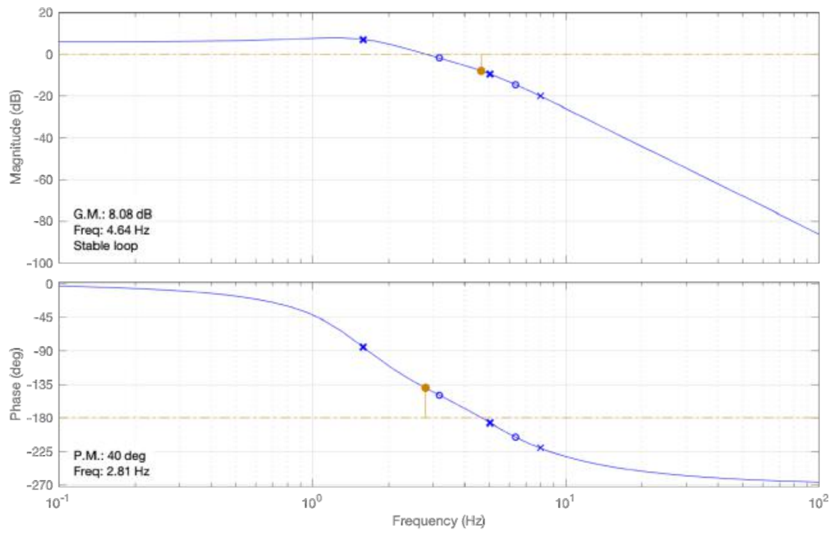
Again, consider the example block diagram shown in Figure 3.8 and its associated open-loop transfer function $K \cdot G(s)$. Figure 3.13 below shows the bode plots for the three previously used values of K.

[3] A phase shift of 180 degrees is equivalent to a phase shift of -180 degrees.

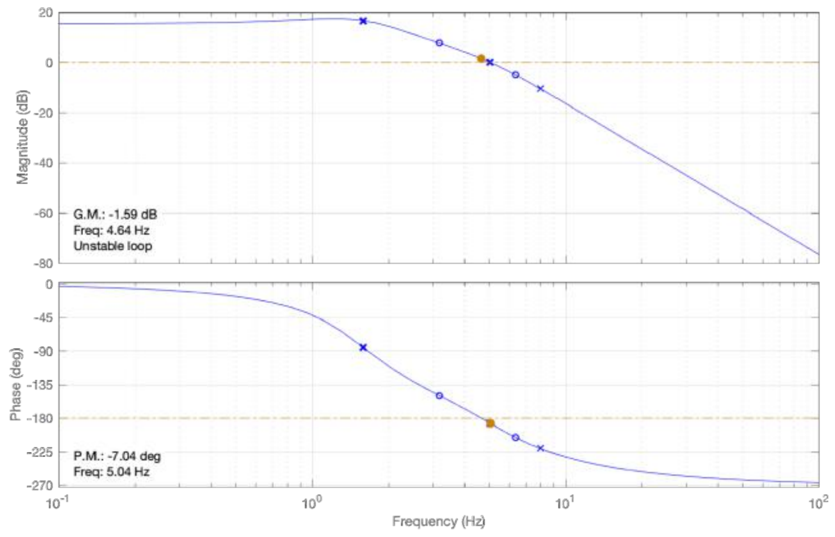
[4] $20log(1) = 0$



(A) $K = 1.027$



(B) $K = 12325$



(C) $K = 37528$

FIGURE 3.13: Bode plots at varying gains.

Figures 3.13a and 3.13b both have sufficiently large stability margins and are thus stable. Figure 3.13c, however, has negative gain and phase margins and is therefore unstable.

3.7 Controller Design

The next step in the control loop design process is to design the controller(s). The controller design phase goes hand-in-hand with the analysis phase; when testing new configurations for controllers, one should analyse the loop constantly to verify its stability. The type of controller that will form the basis of this section is the *proportional-integral* controller as it is the type of controller most commonly used in position control loops.

3.7.1 The proportional integral controller

The biggest advantage of a PI controller is its ability to produce zero error at DC steady state and remove any offset error. This is because, at DC, $s = 0$ which causes the gain to shoot to infinity. Another, more practical way of thinking about it is, due to the integral component, even the smallest of errors will be summed over time to produce an increasingly large error. The only time that the error term will not grow is when the error is zero. A disadvantage of a PI controller compared to a P controller is that it decreases the bandwidth of the system.

The form of a PI controller is shown below:

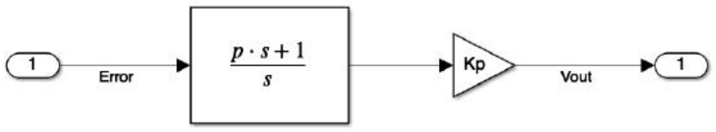


FIGURE 3.14: General form of a PI controller.

The following algebraic manipulation shows how this is a PI controller:

$$\begin{aligned}
V_{out} &= Error \times \frac{p \cdot s + 1}{s} \times K_p \\
&= Error \times \left(p + \frac{1}{s}\right) \times K_p \\
&= Error \times \left(K_p \cdot p + K_p \cdot \frac{1}{s}\right) \\
&= Error \times K_p \cdot p + Error \times K_p \cdot \frac{1}{s}
\end{aligned}$$

Considering that $\frac{1}{s}$ represents integration, it is clear that, in the last line of the above equation, the first term forms the proportional component while the second term forms the integral component. $K_p \cdot p$ is often referred to as the proportional gain and K_p is often referred to as the integral gain. The reason for using the form shown in Figure 3.14 - instead of combining the two blocks into one - is that it is useful to keep the blocks separate when analysing the bode plots. When the blocks are separated, the p can be considered a time constant and manipulating its value will cause a known effect on the phase plot while manipulating the K_p value will cause a known effect on the gain plot. A change in the p value can have an effect on the gain plot too; but a change in the value of K_p only has an effect on the gain plot - not the phase plot. Therefore, the value of p should be decided before determining the value of K_p.

The effect that p and K_p have can be shown by generating bode plots from the *error* signal to V_{out} in Figure 3.14.

First, p will be set to zero and K_p to one. The resulting bode plot is shown below:

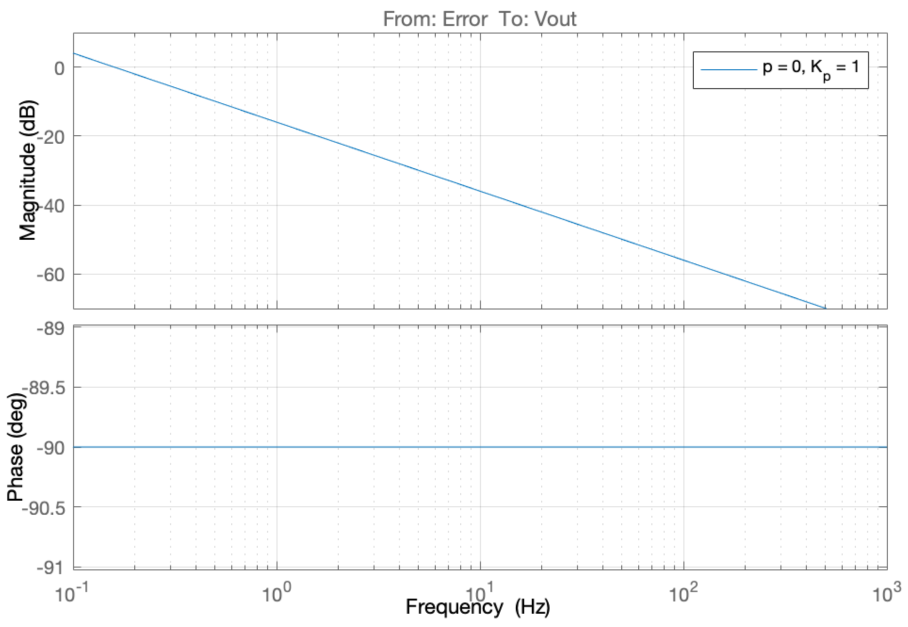


FIGURE 3.15: PI controller bode plot

After changing the value of p to $\frac{1}{2\pi \cdot 10}$ s, the bode plot changed as follows:

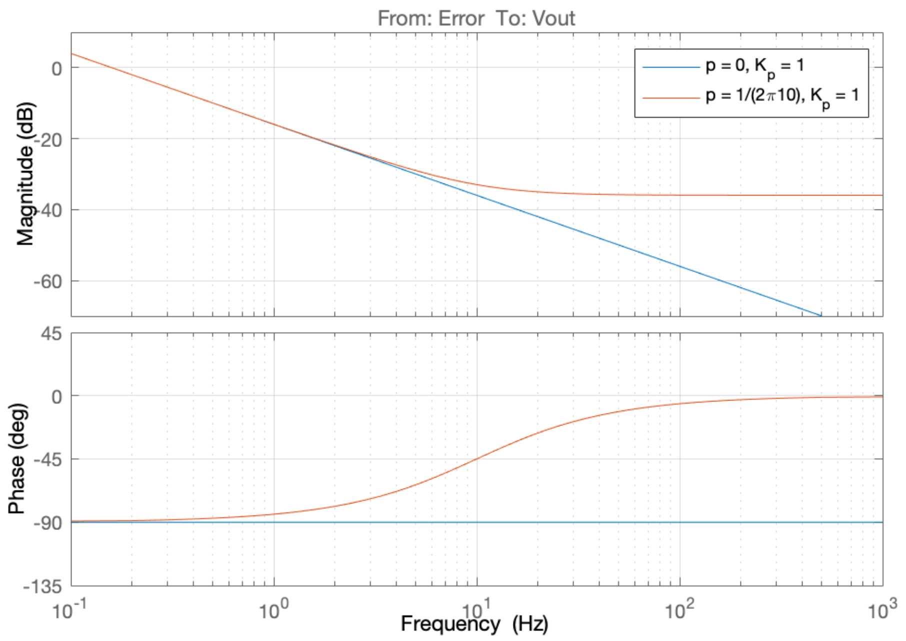


FIGURE 3.16: PI controller bode plot

This demonstrates the effect of the time constant p. In general, if p is changed to $\frac{1}{2\pi f_p}s$, the phase at f_p will increase by 45° and continue to rise to a maximum increase of 90° at a frequency of approximately $10 \times f_p$. The increase in phase will start at a frequency of approximately $0.1 \times f_p$.

The value of K_p was changed to 44.668 with the following effect:

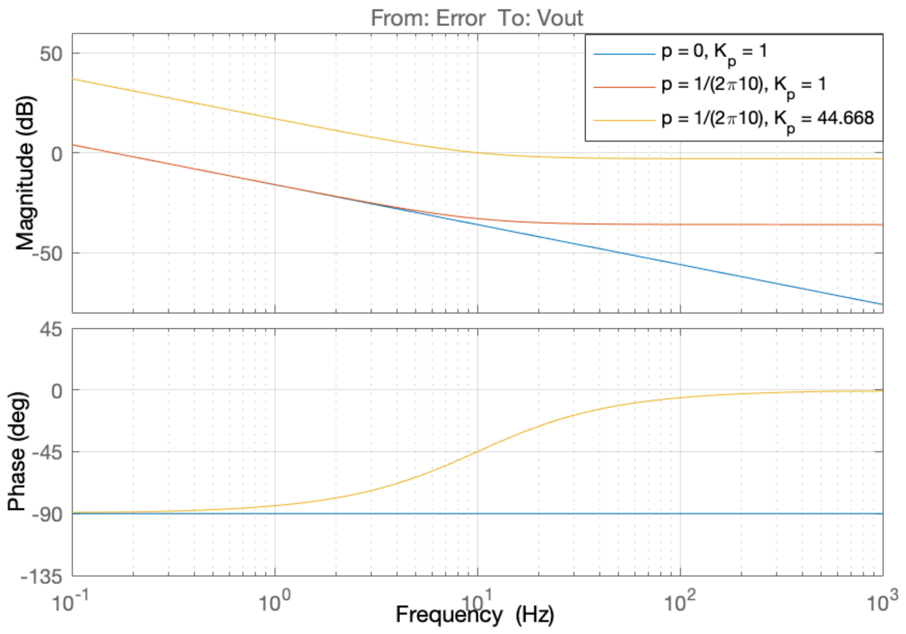


FIGURE 3.17: PI controller bode plot

It is clear that changing the value of K_p has no effect on the phase plot. It merely shifts the gain plot up or down. Upon closer inspection, one can see that the gain plot has been shifted up by approximately $33dB$. This is because $20log(44.668) = 33dB$.

A proportional controller has the same effect that K_p has - it merely scales the gain.

3.8 Mixed Digital and Analogue Loops

An important aspect of control loop design is accounting for digital components in a control loop. An inherent flaw in all digital components is their sample time. The sample time of a component is effectively a delay and therefore causes a phase shift. These delays can be mathematically modelled by converting the relevant blocks from the s-domain to the z-domain. This is done using the approximate relationships between s and z discussed in Section 3.5.2. The *bilinear transform* will be used as it is the most accurate[5].

Remembering that the bilinear transform is defined as:

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} = \frac{2}{T} \cdot \frac{z - 1}{z + 1}$$

[5]In special cases where the order of the numerator exceeds the order of the denominator, like in differentiation, the right-hand rectangular relationship should be used.

one can substitute this equation into a transfer function in the s-domain to produce the equivalent z-domain transfer function. For example, the conversion of the PI controller, $C(s)$ to the z-domain is as follows:

$$\begin{aligned}C(s) &= \frac{p \cdot s + 1}{s}\\ \Rightarrow C(z) &= \frac{p \cdot \left(\frac{2}{T} \cdot \frac{z-1}{z+1}\right) + 1}{\frac{2}{T} \cdot \frac{z-1}{z+1}}\\ \Rightarrow C(z) &= \frac{(T+2p)z + (T-2p)}{2z-2}\end{aligned}$$

In addition to converting the relevant transfer functions to the z-domain, one should also ensure that the digital signals are quantized and only sampled at a certain rate - thus modelling the effect of an analogue to digital converter (ADC).

3.9 Difference Equations

In order to implement a control system on a microcontroller, the z-transforms must first be converted to *difference equations*. When forming a difference equation, it is crucial to understand that z^{-1} represents a delay of one sample period. Once this is understood, the derivation of difference equations is straightforward. As an example, the difference equation of a generic 2^{nd} order transfer function will be derived.

Consider the block diagram shown below:

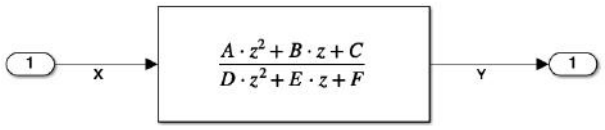

1
X
$\frac{A \cdot z^2 + B \cdot z + C}{D \cdot z^2 + E \cdot z + F}$
Y
1

FIGURE 3.18: 2^{nd} order transfer function block diagram

This can be mathematically represented as:

$$\begin{aligned}\frac{Y}{X}(z) &= \frac{Az^2 + Bz + C}{Dz^2 + Ez + F}\\ &= \frac{A + Bz^{-1} + Cz^{-2}}{D + Ez^{-1} + Fz^{-2}}\end{aligned}$$

$$\Rightarrow DY(z) + Ez^{-1}Y(z) + Fz^{-2}Y(z) = AX(z) + Bz^{-1}X(z) + Cz^{-2}X(z)$$

Considering that z^{-1} represents a delay of one sample period, the following deductions can be made:

$$\begin{aligned}Dy(k) + Ey(k-1) + Fy(k-2) &= Ax(k) + Bx(k-1) + Cx(k-2)\\ \Rightarrow y(k) &= \frac{Ax(k) + Bx(k-1) + Cx(k-2) - Ey(k-1) - Fy(k-2)}{D}\end{aligned}$$

where:
k refers to the current value,

$k-1$ refers to previous value, and
$k-2$ refers to the value two samples prior.
This equation can now be implemented in code by using arrays or extra variables to store previous values of the same variable.

Chapter 4

Methodology

The principle steps that were followed throughout the project are briefly outlined in this chapter. A detailed analysis of each step can be found in subsequent chapters.

4.1 Conceptual Design

Before designing any control system, it is important to know the approximate dynamics and kinematics of the system. This allows one to make informed decisions about the controller selection and block diagram design. In order to formulate the approximate dynamics and kinematics, one should have a rough idea of the physical design of the system. This conceptual design process is closely related to the component selection process described in the following section. This is because the structural design and the components used both influence each other in various ways. However, considering that the final product is a stabilised platform for smartphones, one can make certain deductions and assumptions about the components.

As described in Section 2.4, the following components can be assumed to form part of the physical design:

- DC motor
- MEMS gyroscope
- Relative motion transducer
- Microcontroller

While the exact specifications of these components is not known at this stage, a conceptual design built around these components can be made.

4.2 Component Selection

As discussed above, the process of selecting components is done in conjunction with the conceptual design. The biggest factors in selecting components, however, are the requirements of the control system. For example, the stabilised platform should be able to accommodate smartphones of various dimensions and weights; the system should stabilise the platform up until a minimum frequency (known as the bandwidth of the system); and the system should be battery-powered and handheld.

Once the components have been selected and a conceptual design has been formed, the control loop design can begin.

4.3 Control Loop Formulation

As mentioned above, before the control loop for a system can be designed, one must familiarise oneself with the dynamics and kinematics of the system in order to select a suitable controller type. An understanding of what property of the system needs to be controlled and how this property can be measured/calculated is essential before designing the loop. Next, a mathematical model of the plant needs to be formulated. This initial plant model need not be entirely accurate - its accuracy can be improved further along in the design process. Lastly, it is important to add any potential disturbances to the system to the control loop.

4.4 Controller Design and Simulation

Once the rudimentary control loop has been formulated, the controllers can be designed and simulations can be performed to see how the system responds. The simulation phase includes analysing root locus, bode, and time plots, as described in Section 3.6, in order to improve the system response. At this point, improvements to the plant model can be made and the digital portions of the loop can be converted to the z-domain. These changes will further increase the accuracy of the simulations relative to real system.

4.5 Detailed Design

During the detailed design process, the conceptual design must be carefully analysed and specified. This involves the modelling of the mechanical design in CAD so that the dimensions, kinematics and dynamics of the system can be more accurately estimated and implemented in the control loop and simulation.

4.6 Physical Construction

Once the detailed design is complete, the mechanical system needs to be physically constructed. This involves the ordering/purchasing of components such as the actuator and gyroscope, as well as the manufacturing of the necessary parts. Once the components have been acquired and the parts have been produced, they can be brought together to form the final product. This is a lengthy process and must be performed concurrently with other phases of the project.

4.7 Plant Verification

Once the final mathematical model of the plant has been developed, and the real-life plant is available, it is important to verify the mathematical model. In this project in particular, it is important to verify the model of the power element - the DC motor. An accurate motor model is imperative in producing an accurate - and therefore useful - simulation.

There are multiple ways in which the motor model can be verified. One method is to apply a step input (or ramp input) to the motor and record its response in terms of current draw, torque production or angular velocity - whichever can be most easily and most accurately measured. The same input can be applied to the simulation and the same output can be measured. The two outputs - real and simulated - can then be compared. The more similar the outputs, the more accurate the model. Another verification method - one that is easier to conduct but less informative - is to apply the maximum voltage across the motor terminals and record the motor's angular velocity. This angular velocity multiplied by the torque constant[1] of the motor should yield a voltage that is approximately equal to the voltage supplied to the motor.

4.8 Digital Implementation

Once the physical model has been built, the plant verified and the controllers designed, the controllers can be implemented on the microcontroller. This is a relatively uncomplicated process that can be achieved using the difference equations described in Section 3.9. In addition to implementing the controllers, code that handles the reading of the sensor(s) and the outputting of the signals to the power elements needs to be generated.

Once the digital implementation of the control loop is finished and the physical model is ready, the testing phase can begin.

4.9 Testing and Results

During the testing phase, the performance of the control system is analysed. Ideally, one would use a *testing rig* to assess whether the system requirements are met. In this project, the ideal testing rig would be a single-axis *rate table*. Rate tables are devices capable of rotating at very accurate command speeds and are thus commonly used in industry to test the performance of inertial devices such as ISPs, accelerometers and gyroscopes. The rate table can be programmed to follow a certain speed profile that will test the requirements of the system. The data from the sensors of the system can be recorded during the test and analysed afterwards to evaluate its performance.

However, without access to a rate table, one must use alternative methods to test the performance of the ISP. One such alternative method is to use an additional gyroscope to measure and record the angular velocity of the base motion applied to the system. This base motion can be compared to the platform motion (relative and inertial) to analyse the performance of the system.

4.9.1 Comparison between Simulation and Actual

Once testing has been completed, it is good practice to compare the *actual* performance of the system to the *simulated* performance of the system. Even if the system's performance does not meet the specifications, the comparison is still useful. The comparison can reveal how accurate the simulation is. An accurate simulation will produce results that are almost identical to the actual results. If the simulation is known to be accurate, then one can confidently tune the controller to produce more desirable results while knowing that these adaptations will produce the same results when implemented in the actual system. This is of great benefit because it is significantly easier and less time-consuming to improve the response of a simulation than to improve the response of the physical system through trial-and-error.

[1]The torque constant of a DC motor (units: Nm/A) is approximately equal to the *back-EMF* constant (also known as the *voltage* constant) with units: $V/(rad/s)$.

4.10 Summary

The process of designing, analysing, constructing, verifying, implementing and testing an ISP is an iterative one. There is great overlap between each of the sub-processes and one sub-process is often influenced by multiple others. It is best to start with a simplified model of the system's dynamics and to improve, tweak, and tune the model as the actual dynamics are more accurately known.

Chapter 5

Conceptual Design

A potential design for the gimbal is shown below in Figure 5.1. The main components of this design include:

- the handle/base,
- the BLDC motor,
- the potentiometer,
- the platform, and,
- the gyroscope

The base of the BLDC motor and the base of the potentiometer are fixed to the handle as shown. The output shafts of the motor and potentiometer are then coupled to the mount along the same axis of rotation. The smartphone will be held in place by the mount. The gyroscope will be attached directly to the mount.

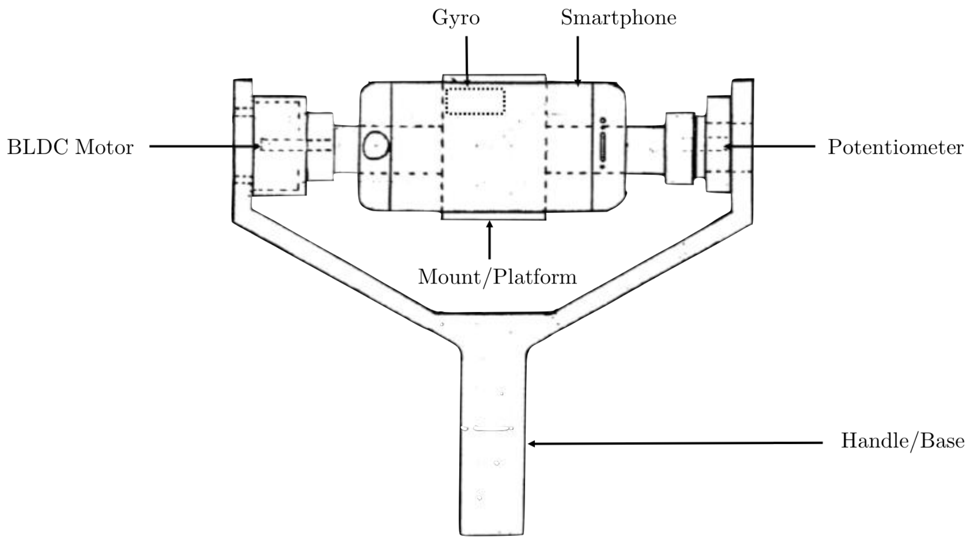


FIGURE 5.1: Basic Concept Design

In order for different sized smartphones to be accommodated by the gimbal, the mount needs to be adjustable. Figure 5.2 below shows a potential design for the mounting feature. The top and bottom grips can both translate vertically along the slots. When the desired width is achieved, the grips can be secured using nuts and bolts.

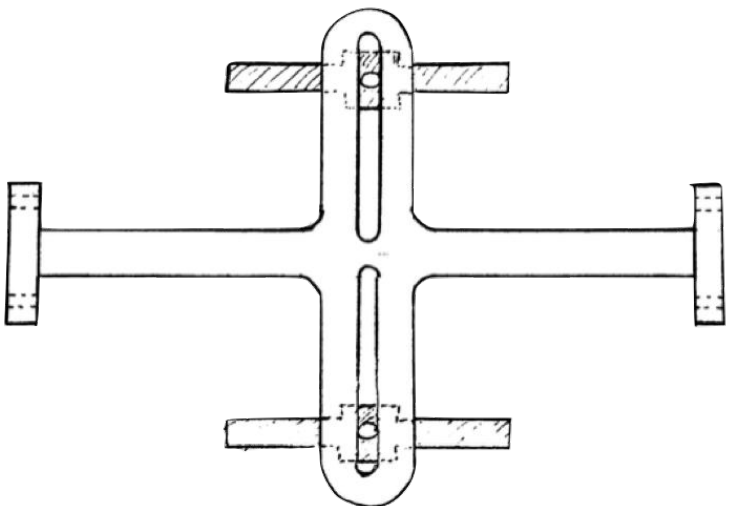


FIGURE 5.2: Mounting Feature Concept

A crucial part of the mechanical design is the ability to adjust the position of the smartphone and/or mount perpendicularly relative to the rotational axis of the motor. This is so that the torque required by the motor is minimized - as discussed in Section 2.4.1.

Chapter 6

Component Selection

In order to produce an ISP capable of tracking base motion, the following components are essential:

- Electric motor
- Gyroscope
- Relative motion transducer
- Microcontroller

Table 6.1 below describes what each component does and why it is needed:

TABLE 6.1: Purpose of, and reason for, base-tracking ISP components.

Component	Purpose	Reason
Electric motor	The electric motor is responsible for rotating the payload in a controlled manner.	Rotation is needed to negate disturbance rotations and to track intentional base motion.
Gyroscope	The gyroscope is used to measure the absolute inertial angular velocity of the payload.	The angular velocity of the payload is the feedback signal for the velocity control loop.
Relative motion transducer	The relative motion transducer is used to determine the absolute relative angle between the base and the platform.	The angle between the base and platform is the feedback signal for the position control loop.
Microcontroller	The microcontroller is responsible for reading the sensor measurements, implementing the control system, and outputting command signals to the power elements.	The entire control loop must be digitally implemented in an embedded system.

An in-depth analysis of each of these components is covered in the following sections.

6.1 Electric Motor

As discussed in Section 2.4.1, BLDC motors are ideal for stabilised platforms. A rough torque calculation to estimate the torque required for the system is done below:

Assume the dimensions of the smartphone to be $140 \times 70 \times 5mm$ and the mass to be $150g$. The moment of inertia of the smartphone can be approximated using the formula for the moment of inertia of a rectangular prism about its longest axis:

$$\begin{aligned} I_{phone} &= \frac{m(h^2 + w^2)}{12} \\ &= \frac{0.150(0.005^2 + 0.070^2)}{12} \\ &= 6.2 \times 10^{-5} kg \cdot m^2 \end{aligned}$$

Assume the moment of inertia of the gimbal arm to be the same. The total moment of inertia of the gimbal arm and smartphone is therefore:

$$I_{total} = 2 \times 6.2 \times 10^{-5} = 1.2 \times 10^{-4} kg \cdot m^2$$

Multiply this by a safety factor of 1.4 to account for any potential misalignments between the motor's axis of rotation and the centroid of the gimbal and smartphone:

$$\Rightarrow I_{total} = 1.4 \times 1.2 \times 10^{-4} = 1.68 \times 10^{-4} kg \cdot m^2$$

The torque required can be calculated using the rotational adaptation of Newton's second law:

$$\begin{aligned} T &= I_{total} \times \ddot{\theta} \\ \Rightarrow \dot{\theta} &= \frac{T}{I_{total}} \cdot t \\ \Rightarrow \theta &= 0.5 \cdot \frac{T}{I_{total}} \cdot t^2 \\ \Rightarrow T &= \frac{2\theta I_{total}}{t^2} \end{aligned}$$

For a disturbance that would require the platform to rotate 2° in 0.02 seconds, the following amount of torque would be required:

$$\begin{aligned} T &= \frac{2\left(2 \cdot \frac{\pi}{180}\right) \cdot 1.68 \times 10^{-4}}{0.02^2} \\ &= 0.0293Nm \end{aligned}$$

6.1.1 Final Selection

The BLDC motor and controller that was eventually used is the *Faulhaber 2232S024BX4SC* shown in Figure 6.1 below. This is a 4-pole brushless motor with an integrated speed controller. It is a $10.5W$ motor with a nominal torque of $17.5mN$. The integrated speed controller allows the output speed of the motor to be controlled with a PWM signal (0 to $10V$) and the direction to be controlled using a digital high or low signal.

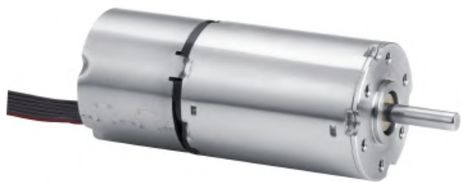


FIGURE 6.1: Faulhaber BLDC motor
[www.faulhaber.com/en/products/series/2232bx4-sc/]

Unfortunately, the slowest speed at which the controller can drive the motor is approximately 18 rad/s. This is too high a speed - therefore, a gearhead had to be fitted to the motor. The gearhead used was the *Faulhaber Series 22/7* planetary gearhead (shown below) with a reduction ration of 66 : 1.

FIGURE 6.2: Faulhaber planetary gearhead
[www.faulhaber.com/en/products/series/2232bx4-sc/]

When possible, it is best to implement direct drive[1] when building an ISP. This is because, without the presence of gears, (i) only the frictional forces present during rotation will have to be overcome when keeping the payload inertially stable, and (ii) the motor will only have to rotate through the same angle that the base is rotated. When gears are introduced, the motor has to pro-actively drive the entire geartrain, platform and payload in order to keep the platform inertially stable despite base rotation. Furthermore, due to the reduction ratio, the motor will have to rotate at a far higher velocity than the applied base motion - 66 times higher in this case.

The ideal BLDC motor for this application would be one with a high number of poles - 12 or 14 poles. These motors are capable of rapid acceleration but can only be driven at relatively low speeds - perfect for the ISP application [9]. Unfortunately, the availability of these motors, especially in South Africa, is sparse. Furthermore, controllers for the motors are even more difficult to source.

6.2 Gyroscope

As described in Section 2.4.2, a MEMS gyroscope is used to measure the angular velocity of the payload. This measured velocity provides feedback that is used to ensure that the payload is moving at the desired speed.

In this application the base shall be subjected to angular velocities of no more than 11 rad/s (the maximum speed of the motor fitted with the gearhead). Therefore a gyroscope with a full scale range of over 630 dps[2] will suffice. Seeing that it is a single-axis ISP, a single-axis gyroscope is sufficient.

[1]The payload is rotated directly by the motor - there is no geartrain involved.
[2]Degrees per second

6.2.1 Final Selection

The MEMS gyroscope that was finally selected actually forms part of an inertial measurement unit[3] (IMU) called the *MPU6050*. This is an extremely popular IMU that is simple to use despite being highly configurable.

The device communicates using the *inter-integrated circuit* (I^2C) protocol. On-board registers allow for the device to be configured to suit its application. An on-board *digital lowpass filter* (DLPF) is also available to the user.

Table 6.2 below shows the possible full scale range configurations and the associated sensitivity.

TABLE 6.2: MPU6050 - Full Scale Range and Sensitivity [10]

Full Scale Range (dps)	**Sensitivity (bits/dps)**
±250	131
±500	62.5
±1000	32.8
±2000	16.4

Table 6.3 below shows the possible configurations of the DLPF and their corresponding sampling rates.

TABLE 6.3: MPU6050 - Bandwidth and Sample Rate [10]

Bandwidth (Hz)	**Sample Rate (kHz)**
256	8
188	1
98	1
42	1
20	1
10	1
5	1

For this project, the gyroscope was configured to have a full-scale range of 1000 dps and the DLPF was configured to a bandwidth of 256 Hz.

6.3 Relative Motion Transducer

There are multiple sensors that can be used to determine the relative angle between to objects. One method that is popular in commercial camera stabilisation platforms is to use two IMUs. One mounted on the platform/payload and the other mounted on the base. The three-axis accelerometers of the IMUs can be used to detect absolute inertial angle through a process known as *gravimetric tilt sensing* [11]. This is essentially the process of determining tilt by measuring the component of gravitational acceleration acting in each axis and using basic trigonometry to determine the tilt. One problem with this is, if the accelerometer is subject to linear accelerations,

[3]Integrated device containing a gyroscope, accelerometer and magnetometer.

the gravitational vector will be distorted. To overcome this a *Kalman filter* is used. In its most primitive form, a Kalman filter effectively fuses the accelerometer data with the gyroscopic data (which is immune to linear accelerations) [11]. It does this by integrating each gyroscopic reading with respect to time to measure the change in angle between readings. This change in angle is then added to the previous angle measured by the accelerometer to give the current angle of the device. This current angle is then fused with the current angle measured by the accelerometer alone by assigning a weighting to the gyroscope's measurement. The weighting represents how much 'trust' is placed in the gyroscope's reading.

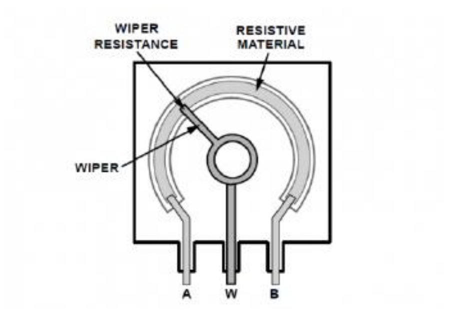
One can determine the absolute relative angle between the base and the platform by simply calculating the difference between the two. This method is particularly popular in platforms that are stabilised across two or three axes. With just two IMUs (each with a three-axis accelerometer and three-axis gyroscope), one can determine the relative angle between the base and platform in all three axes. This significantly reduces complexities associated with wiring and gimbal construction due to the fact that a separate relative angle sensor is not needed for every axis. Because, however, this project is only concerned with a single axis, this method is unnecessarily complicated.

Another method is use a single sensor that is capable of measuring relative angles. This can be done using an encoder or a potentiometer. While encoders have a longer operating life than potentiometers, they are more expensive, more difficult to communicate with, and offer a lower resolution than potentiometers. Therefore a potentiometer was chosen to measure the relative angle between the platform and the base.

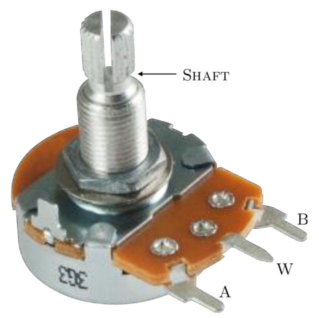
6.3.1 Potentiometer

A potentiometer is a device that can be used as a *relative absolute sensor* to measure relative angles. With reference to Figure 6.3 below, the rotation of the potentiometer's shaft will cause the wiper to rotate - consequently causing the wiper voltage to vary. This variation is linear and can therefore be used to determine the angular position of the shaft relative to the base.

Therefore, the relative angle between the base and platform of the ISP can be determined by connecting the base of the potentiometer to the base of the ISP and coupling the shaft of the potentiometer to the stabilised platform.



(A) Schematic
[https://www.robomart.com/blog/variable-resistors-types/]



(B) Photograph
[www.amplifiedparts.com/products/potentiometer]

FIGURE 6.3: Potentiometer

6.3.2 Final Selection

The potentiometer used this project is the one shown in Figure 6.3b. It is a 10 $k\Omega$ potentiometer with a rotational range of 310°.

6.4 Microcontroller

The following specifications are required of the microcontroller for a project such as this one:

- Clock speed of at least 8 kHz.
- At least one PWM output pin to control the speed of the motor. The switching frequency of this signal should be at least 1.5 kHz.
- At least one analogue to digital converter (ADC) with a resolution greater than 6 $bits$ to read the potentiometer output voltage..
- General purpose input/output (GPIO) capabilities to control the direction of the motor and communicate with IMU.

The above specifications are met by the majority of microprocessors.

6.4.1 Final Selection

Considering that the focus of this project was on the control theory behind ISPs and not necessarily the configuration of an embedded system upon which the control system could be run, an *Arduino* microcontroller was used; this helped expedite the process of digital implementation.

More specifically, the *Arduino Duemilanove* was used. According to the *Arduino* website, it has the following specifications:

TABLE 6.4: Arduino Duemilanove specifications

Chip	**Operating Voltage (V)**	**GPIO Pins**	**PWM Pins**	**ADC**	**Clock Speed (MHz)**
ATmega168	5	14	6	10 bit	16

Chapter 7

Control Loop Formulation

This chapter discusses the formulation of the control loop for a base-tracking ISP that improves the footage quality of a camera mounted to it.

7.1 Control System Requirements

A successful system would be one that nullifies base motion due to vibration and other sharp, jerky movements - the sort of movements that would cause the camera to lose focus. In addition to this, the system should allow the user to point the camera by rotating the base. This requires that the platform tracks the intentional base motion while simultaneously rejecting jerky movements. This sort of motion can be achieved by designing an outer positional loop with a relatively low bandwidth compared to the inner stabilisation loop. The bandwidth required to reject vibrational movements is higher than can be achieved with a gear reduction ratio of $66:1$, however, the aforementioned sharp, jerky movements can still be rejected.

In order to achieve the desired control, two variables need to be controlled. Namely, the angular velocity of the platform (ω_p) and the relative angle between the base and the platform (θ). These two variables are measured by the gyroscope and the potentiometer respectively. Since it is ultimately the relative position that is being controlled, the positional loop forms the outer loop while the velocity loop forms the inner loop.

The command signal for the loop will constantly be zero. The reason for this is, in order for the platform to track the base, the angle between them must be driven to zero.

7.2 Kinematics of the System

Due to the simple geometry, the kinematics of the system are uncomplicated. The inertial rotation of the output shaft of the motor results in an equal inertial rotation of the platform and payload in the same direction.

The relative velocity between the platform and base ($\dot{\theta}$) is simply the inertial velocity of the platform (ω_p) minus the inertial velocity of the base (ω_b). This relative velocity can the be integrated with respect to time to give the relative position between the platform and base (θ).

7.3 Initial Control Loop

The initial control loop is a simplified model of the system. It helps provide a foundation upon which a more complex and realistic system can be built.

The block diagram shown below was provided by the supervisor of this project, Associate Professor Hennie Mouton.

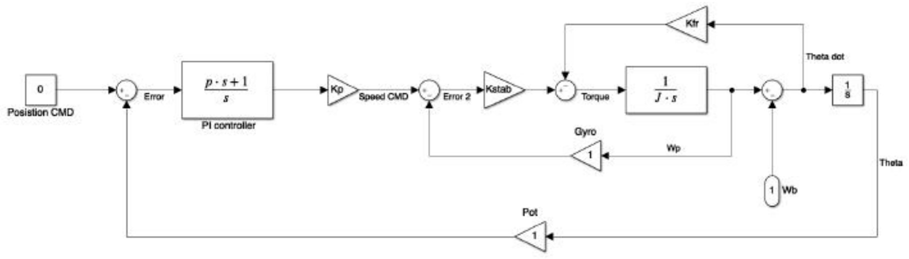


FIGURE 7.1: Initial control loop block diagram

The position command of 0 *rad* is compared with the relative angle measured by the potentiometer (pot) to produce an error signal. This error signal then goes through the PI controller and gain K_p to produce the speed command. The speed command is compared to the inertial speed measured by the gyroscope (gyro) to produce the second error term. This error term goes through the P controller K_{stab} to produce a torque value. The effective torque is then found by subtracting the disturbance torque due to friction from this value. The torque is then divided by the moment of inertia of the rotor, payload and gimbal structure, and simultaneously integrated to produce the inertial velocity of the platform (ω_P). The inertial velocity of the base is subtracted from this value to produce the relative velocity ($\dot{\theta}$). $\dot{\theta}$ is then integrated to produce the relative angle between the platform and the base (θ).

After setting the values of K_{stab} and J to 0.1 and 0.0001 $kg \cdot m^2$ respectively, the above block diagram was simulated with different combinations of p and K_p to produce the following results:

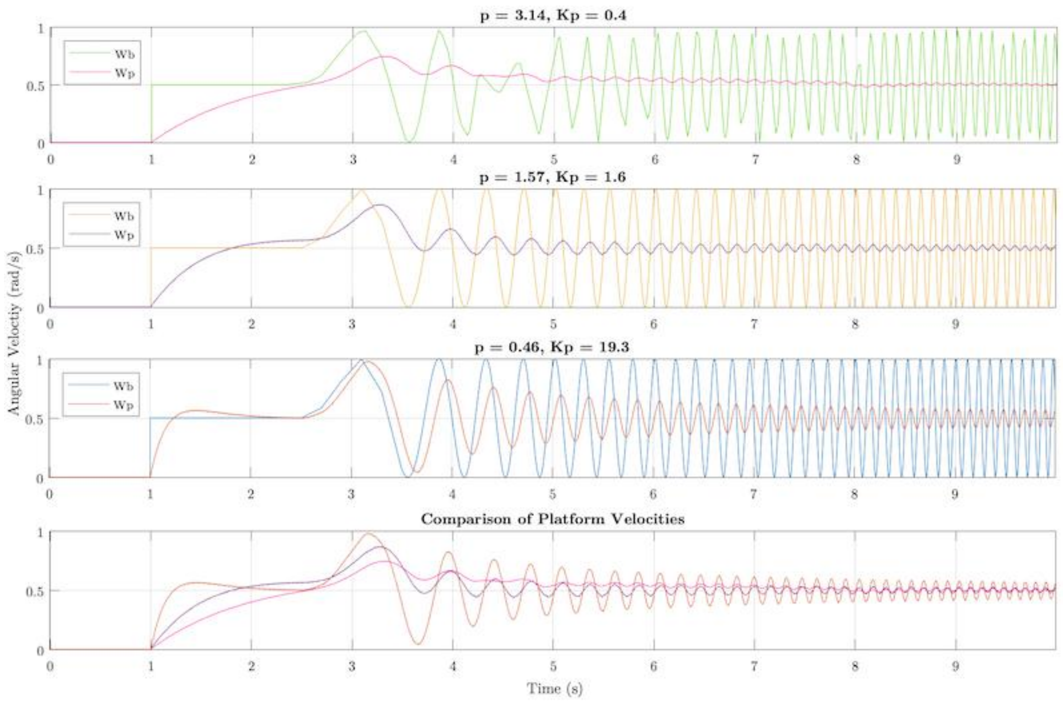


FIGURE 7.2: Initial control loop simulation with varying combinations of p and K_p.

In the simulation above, a step input was applied to the base at 1 second and then a sine wave of increasing frequency was added at 2.5 seconds. This base motion signal is plotted against the inertial velocity of the platform. One can see how the platform velocity tracks the base motion velocity at low frequencies. At higher frequencies, the base motion is rejected and the angular velocity of the platform remains close to constant.

The three sets of results show how the values of p and K_p affect the position response and stability response of the platform. Increasing the value of p results in a slower position response but better base motion rejection (see top graph); while increasing the value of K_p results in a faster position response but lower stability.

The simulation confirms that the loop works, in theory at least. Next, a more accurate representation of the plant must be modelled.

7.4 Dynamics of a DC Motor

In this project, the plant is the electric motor and the gimbal structure and payload which it drives. The plant is therefore effectively a DC motor driving a load - the load in this case being the gimbal structure and the payload.

While the operation of a BLDC motor is vastly different to that of a brushed one, the same governing equations can be applied to both. The following equations represent the dynamics of a DC motor [6]

$$V_{eff} = V_m - \dot{\theta} \times K_v \times N_{gear}$$

where V_{eff} is the effective voltage, V_m is the applied voltage, $\dot{\theta}$ is the relative angular velocity, K_v is the back-EMF constant, and N_{gear} is the gear ratio.

$$I_m = \frac{V_{eff}}{R_m + sL_m}$$

where I_m is the motor current, R_m is the resistance of the motor, and I_m is the inductance of the motor.

$$T_m = I_m \times K_m$$

where T_m is the motor torque, and K_m is the torque constant.

$$T_{eff} = T_m \times N_{gear} \times \eta_{gear} - \dot{\theta} \times K_{fr}$$

where T_{eff} is the effective torque, η_{gear} is the efficiency of the geartrain, and K_{fr} is the friction constant.

$$\dot{\omega} = \frac{T_{eff}}{J}$$

where $\dot{\omega}$ is the inertial angular acceleration, and J is the total moment of inertia of the load (incl. the rotor's inertia).

$$\omega = \frac{1}{s} \times \dot{\omega}$$

where ω is the inertial angular velocity, and $\frac{1}{s}$ represents integration.

$$\dot{\theta} = \omega - \omega_b$$

where ω_b is the inertial angular velocity of the base.

$$\theta = \frac{1}{s} \times \dot{\theta}$$

where θ is the relative angle.

From these equations, a block diagram representing the mathematical model of a DC motor can be made:

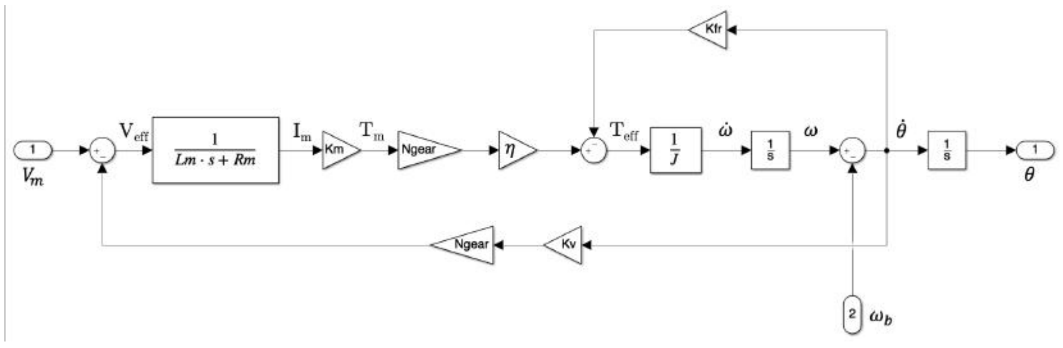


FIGURE 7.3: DC motor block diagram

The motor used in this project has the following known characteristics from the datasheets:

$$
\begin{aligned}
V_m &= 24V(max) \\
R_m &= 54.86\Omega \\
L_m &= 270\mu H \\
K_m &= 31.4mNm/A \\
N_{gear} &= 66 \\
\eta_{gear} &= 0.7 \\
K_v &\approx 31.4mV/(rad/s)
\end{aligned}
$$

The values of J and K_{fr} were estimated to be 0.0001 $kg \cdot m^2$ and 0.001 respectively.

The initial control loop (Figure 7.2) and the motor model block diagram where then brought together to from the final loop.

7.5 Dynamics of a MEMS Gyroscope

A MEMS gyroscope uses a small vibrating mass and the effects of rotation on Coriolis force to detect angular velocity. While the physics behind this effect are not pertinent to this project, it is important that the dynamics of this spring-mass system are taken into account. The dynamics of a MEMS gyroscope are governed by a second-order equation. This means that if the gyroscope is rotated back and forth at a frequency higher than the natural frequency, the gyroscope will experience a drop in gain and a lag in phase. Fortunately, the natural frequencies of gyroscopes are usually in the *kHz* range and are therefore never reached.

The dynamics can therefore be modelled as shown below:

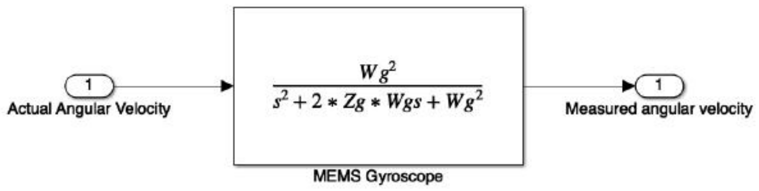

1
Actual Angular Velocity
$\frac{Wg^2}{s^2 + 2 * Zg * Wgs + Wg^2}$
MEMS Gyroscope
1
Measured angular velocity

FIGURE 7.4: Transfer function modelling the dynamics of a MEMS gyroscope.

where W_g is the natural frequency of the gyroscope and and Z_g is the damping factor of the system. Typical values for W_g and Z_g are $2\pi \cdot 1000 rad/s$ and 0.707 respectively.

7.6 Digital Components

As described in Section 3.8, the digital components of a control system need to be taken into account in the simulation. In this project, the potentiometer produces an analogue signal which is then digitised by the ADC on the *Arduino*. The gyroscope module has an on-board ADC that digitises the readings. Therefore, the two sensor readings must be digitised before reaching their respective comparators. For now, these two signals can simply be quantised at the sample rate of the respective ADCs.

In Section 8.3 on *scaling*, the ADCs will be modelled properly.

In addition to adding the quantisers, the PI controller must be converted to the z-domain as shown in Section 3.8. The sampling times of the ADCs and the PI controller can be set in *Simulink* by double-clicking the block. Once these sample times are set, *Simulink* will take of converting the signal back to an analogue one before entering the plant.

7.7 Updated Control Loop

The updated control loop block diagram is shown in Figure 7.5 below:

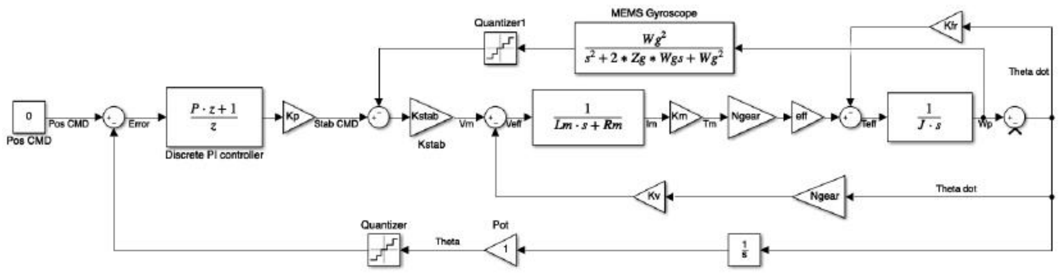


FIGURE 7.5: Updated control loop block diagram

Chapter 8

Controller Design and Simulation

Now that the control loop has been formulated, the controllers need to be designed to produce the desired response. The general process followed when designing a controller is as follows:

1. Start by designing the innermost controller - in this case the proportional controller K_{stab}.
2. Set any time constants to zero and gains to one.
3. Generate a bode plot of the open loop response.
4. Adjust the time constants to get an acceptable phase plot.
5. Adjust the gain to get acceptable stability margins.
6. Verify stability with a closed loop bode plot.
7. Verify system response with a time plot.

8.1 Stabilisation Loop

The stabilisation loop forms the inner loop and must therefore be designed first. The controller in this case is a proportional controller named K_{stab}.

With K_{stab} set to 1.0, an open loop bode plot from the inner loop error signal to the gyroscope output is generated:

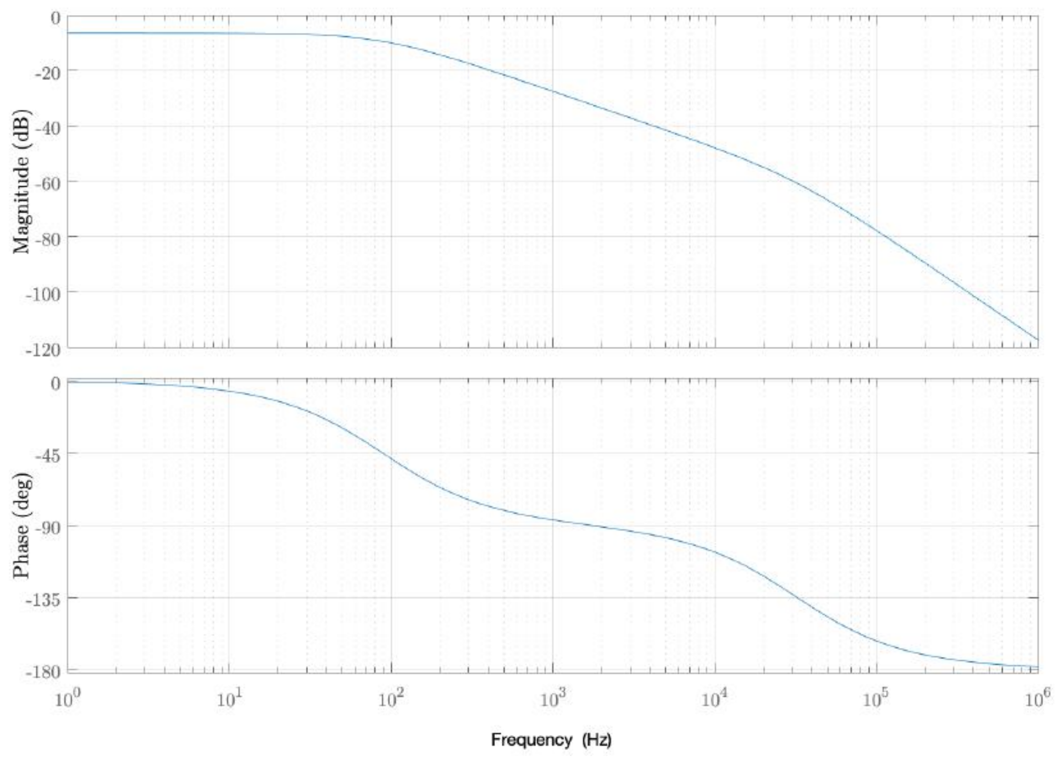


FIGURE 8.1: Open loop bode plot.
$K_{stab} = 1.0$

Because the gain is never above 0 dB and the phase never crosses $-180°$, the phase and gain margins are both infinite. This is supported by the root locus plot below:

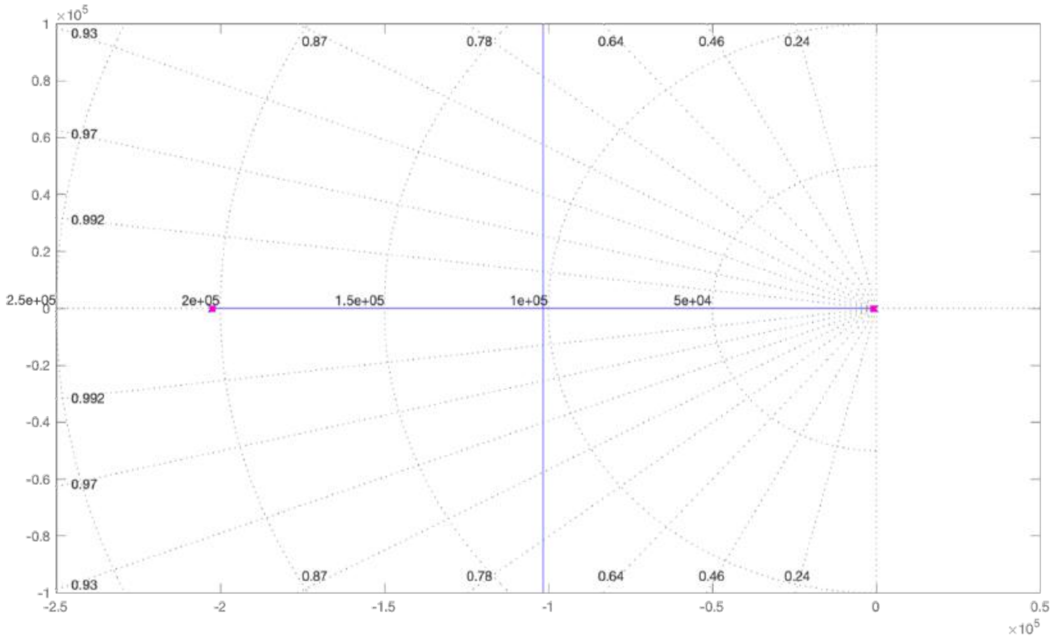


FIGURE 8.2: Root locus plot

The root locus plot shows that all the poles of the system lie to the left of the imaginary axis for all values of K_{stab} - meaning that the system will never go unstable. A K_{stab} of 5 will be used for now.

8.2 Position Loop

Now that the inner stabilisation loop has been designed, the outer position loop can be designed.

Figure 8.3 below is a bode plot of the open loop system with $p = 0$ and $K_p = 1.0$.

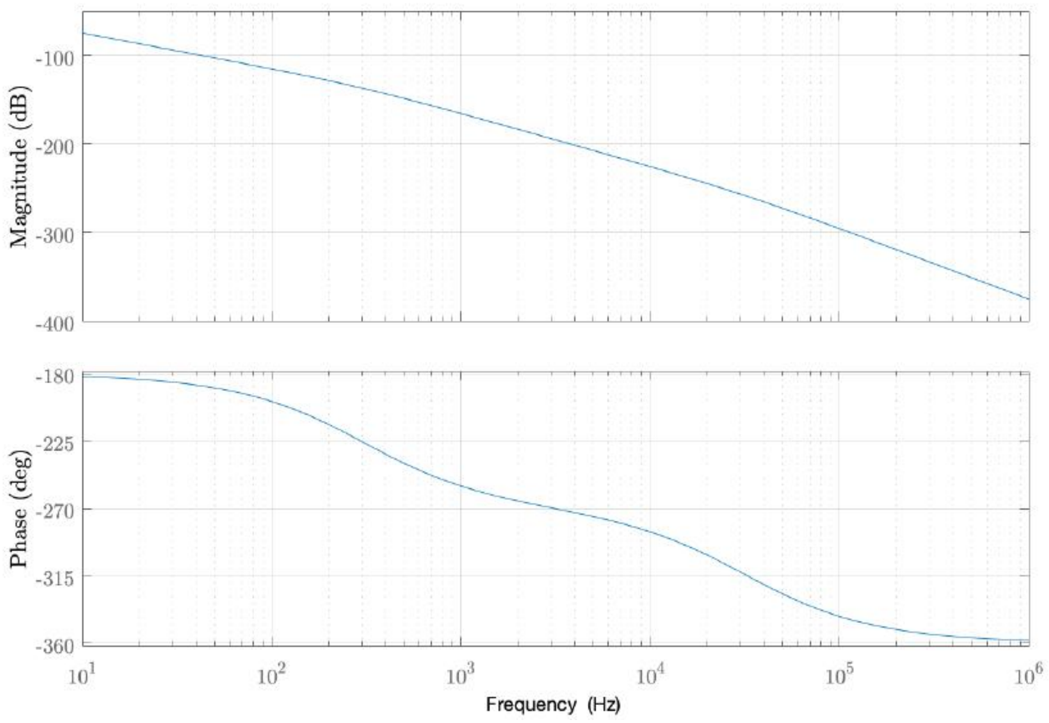


FIGURE 8.3: Open loop bode plot.
$p = 0$, $K_p = 1.0$

It is clear that the phase needs to be increased at low frequencies. This can be done by setting the time-constant p to a non-zero value. By setting p to $\frac{1}{2\pi \cdot 0.1}s$[1], 45° will be added at 0.1 Hz as shown by the orange plot below:

[1]This *s* refers to the unit *seconds* - not the Laplace variable.

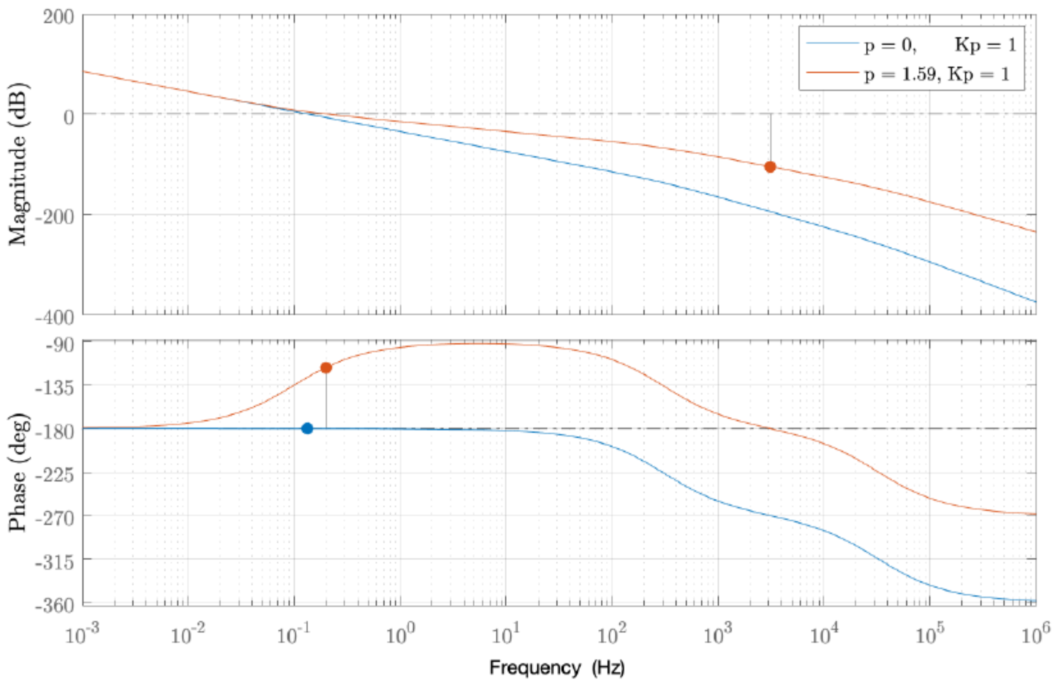
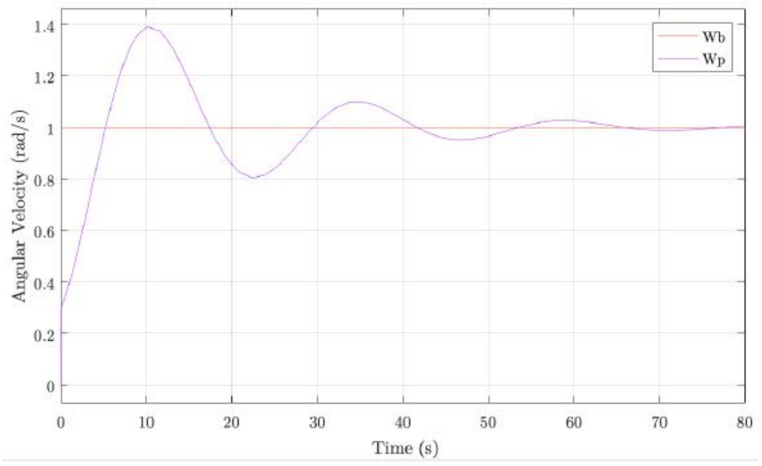
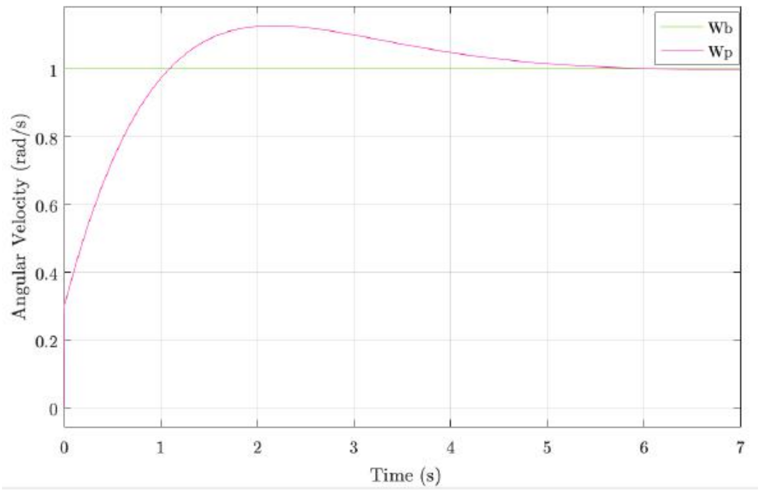


FIGURE 8.4: Open loop bode plot.
$p = \frac{1}{2\pi \cdot 0.1}s = 1.59s$, $K_p = 1.0$

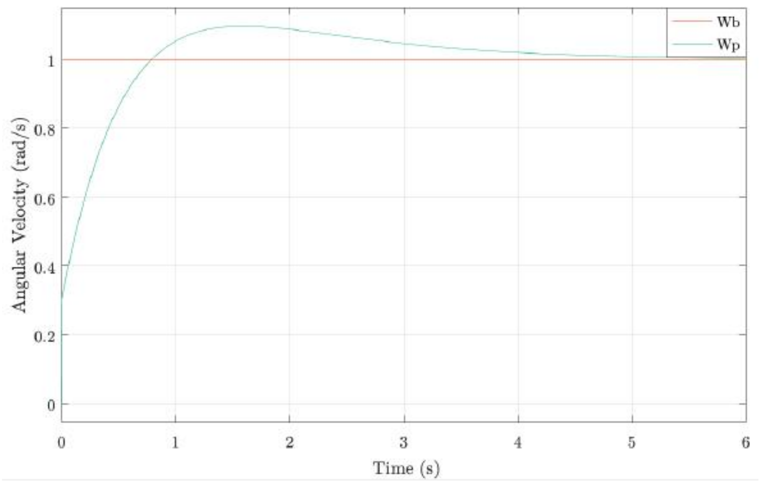
The resulting gain and phase margins are 105 dB and 63.3° respectively - therefore the loop is stable. By varying the value of K_p, the response time and damping of the position loop can be varied. The plots below show the response of the platform velocity at varying values of K_p when a unit step input is applied to the base:



(A) $K_p = 0.1$



(B) $K_p = 1.4$



(C) $K_p = 2.24$

FIGURE 8.5: Step responses at varying values of K_p. (Note the different time scales.)

Figure 8.5a illustrates an extremely slow and underdamped response. Figures 8.5b and Figure 8.5c show faster responses with fewer oscillations. The response of 8.5b is slightly to slow however, therefore a K_p value of 2.24 will be used. Figure 8.6 below shows the response of the system given the same base motion input as Figure 7.2.

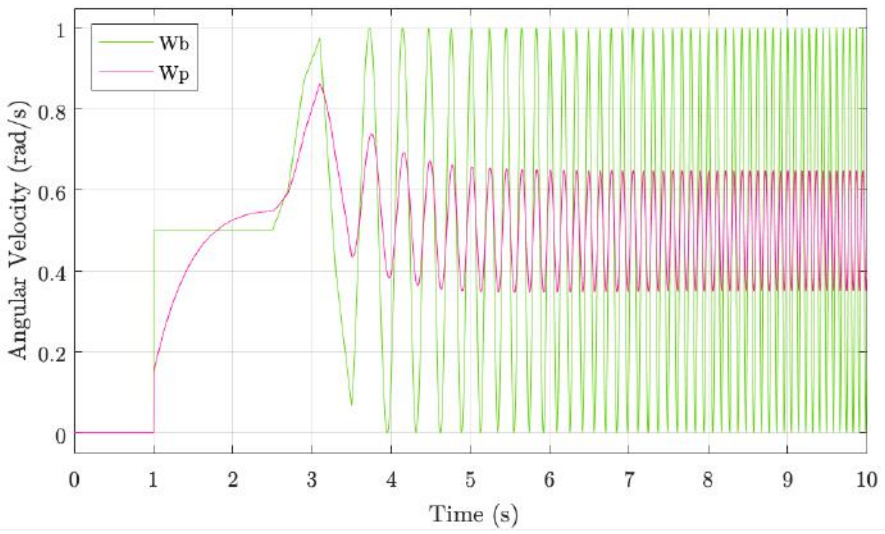


FIGURE 8.6: Response of system to base motion disturbance.
$p = \frac{1}{2\pi \cdot 0.1}s = 1.59s$, $K_p = 2.24$, $K_{stab} = 5$

The base motion can be further rejected by increasing the value of K_{stab} to 10:

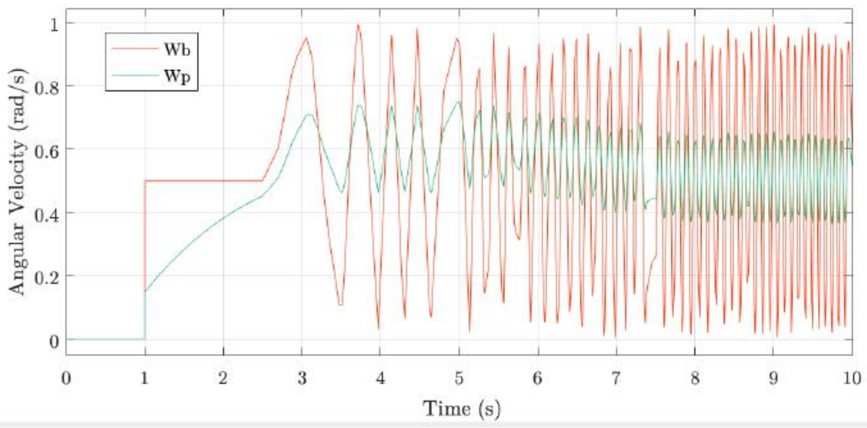


FIGURE 8.7: Response of system to base motion disturbance.
$p = \frac{1}{2\pi \cdot 0.1}s = 1.59s$, $K_p = 2.24$, $K_{stab} = 10$

8.3 Scaling

The majority of sensors output readings that are not in the desired units. These readings need to be scaled such that the output is in the correct units.

8.3.1 Gyroscope

Before scaling the output of the gyroscope, it is important to improve the accuracy of the simulation by introducing a more in-depth model of the gyroscope. Usually this would be done by adding a *static bias* and a *sensor noise* block. However, the static bias block will be omitted as the on-board *offset registers* of the MPU6050[2] remove

[2]The gyroscope of the MPU6050 will be discussed further in Section 12.2

this bias.

The noise will be modelled using the *band-limited white noise* block in *Simulink* - a block that produces noise of equal power at all frequencies. To determine the noise power, one can use the *gyro rate noise* value from the datasheet. The noise power is simply the gyro rate noise value squared. According to the MPU6050 datasheet, the gyro rate noise is $0.005 dps/\sqrt{Hz}$. The noise power is therefore $2.5 \times 10^{-5}(dps)^2/Hz$.

The MPU6050 also has an on-board digital lowpass filter with a cut-off frequency of $256Hz$. This filter will also be modelled in the gyroscope model. The model of a lowpass filter has already featured in this report; it was used to model the dynamics of the gyroscope. The same model will be used for the DLPF except the value of ω_g will change to $2\pi \cdot 256\ rad/s$ and will be called ω_f. This filter should technically be converted to the z-domain as it is a digital filter - this conversion, however, will produce negligible changes and is thus omitted.

Lastly, the output the lowpass filter should go through an ADC. The MPU6050 has a 16 bit ADC that outputs a signed integer. The minimum and maximum outputs of the ADC will be the full scale range of the gyroscope - $\pm 1000dps$.
The final gyroscope model is as shown below:

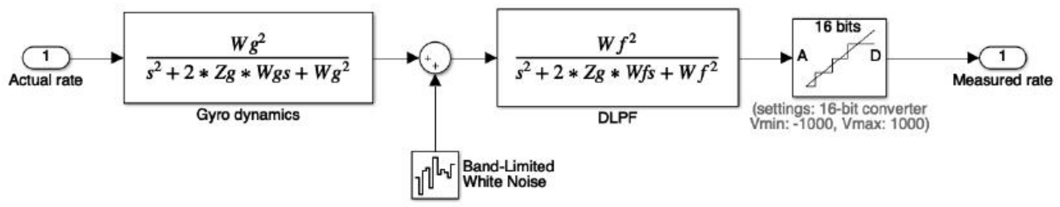

1
Actual rate
$\frac{Wg^2}{s^2 + 2 * Zg * Wgs + Wg^2}$
Gyro dynamics
+
+
$\frac{Wf^2}{s^2 + 2 * Zg * Wfs + Wf^2}$
DLPF
16 bits
A
D
(settings: 16-bit converter
Vmin: -1000, Vmax: 1000)
1
Measured rate
Band-Limited
White Noise

FIGURE 8.8: Final gyroscope model

In the interest of space, the gyroscope model will henceforth be depicted as:

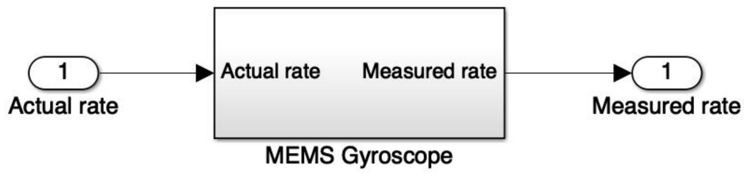


FIGURE 8.9: Gyroscope block

Because the MSB[3] of the gyroscope output represents the sign (positive or negative), the magnitude of the angular rate is given in 15 bits. Therefore, the maximum output of the ADC is $2^{15} = 32768$. This number represents an angular rate of $1000dps$. The output of the gyroscope therefore needs to be scaled by:

$$\frac{1000}{32768} = 0.03052 = \frac{1}{32.8} dps/unit$$

[3]Most significant bit

8.3.2 Potentiometer

The full range of the potentiometer circuit is $0 \rightarrow 4.5454V$[4]. This is over a range of $310°$ or $5.411rad$. The sensitivity of the potentiometer output is therefore:

$$\frac{4.5454}{5.411} = 0.8401V/rad$$

The gain of 1 in the positional feedback loop can be replaced by this scaling factor. When theta is a negative value, the above scaling factor will output a negative voltage. This is not possible for the potentiometer. Therefore an offset must be introduced. The offset must be half of the maximum output of the potentiometer; this means that an angle of $-155°$ will result in a voltage of $0V$ which is correct. The model of the potentiometer is shown below:

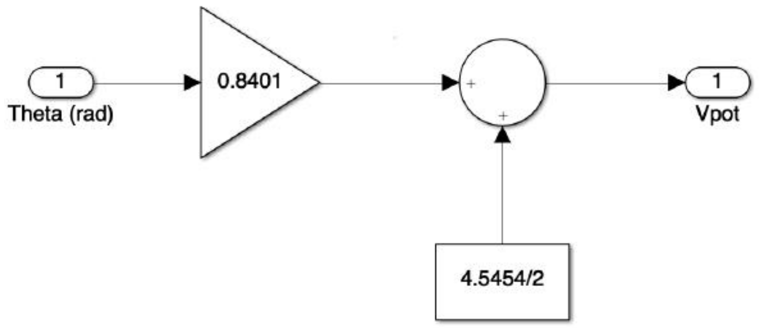


FIGURE 8.10: Potentiometer block

The above model will subsequently be depicted as:

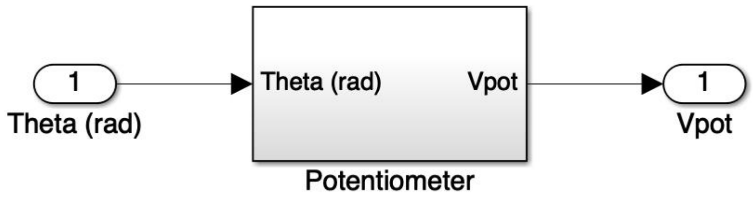


FIGURE 8.11: Potentiometer model

The output of the potentiometer is read-in by the *Arduino's* 10 bit ADC. The range of this ADC is $0 \rightarrow 5V$. Using the same logic applied above, the output of the ADC needs to be scaled by:

$$\frac{5}{2^{10}} = 4.9 \times 10^{-3}V/unit$$

The resulting voltage must then be offset by $-\frac{4.5454}{2}$ and then finally scaled by $\frac{310\times\frac{\pi}{180}}{4.5454} = 1.1903rad/V$ to bring the units back to radians before entering the positional loop comparator.

[4]This value is derived in Section 9.2.1.

8.3.3 P controller

The P controller of the stabilisation loop also needs to be scaled. This is because, instead of directly driving the $\pm 24V$ of the motor, the controller's output is first converted to a PWM signal, which is then amplified by two by the operational amplifier[5], which is then amplified by 2.4 in the speed controller. The new value of K_{stab} should be divided by all of these gains in order to keep the open loop transfer function the same.

8.4 Updated Control Loop Block Diagram

The figure on the next page shows the updated block diagram of the control loop.

[5]Discussed in Section 9.2.2

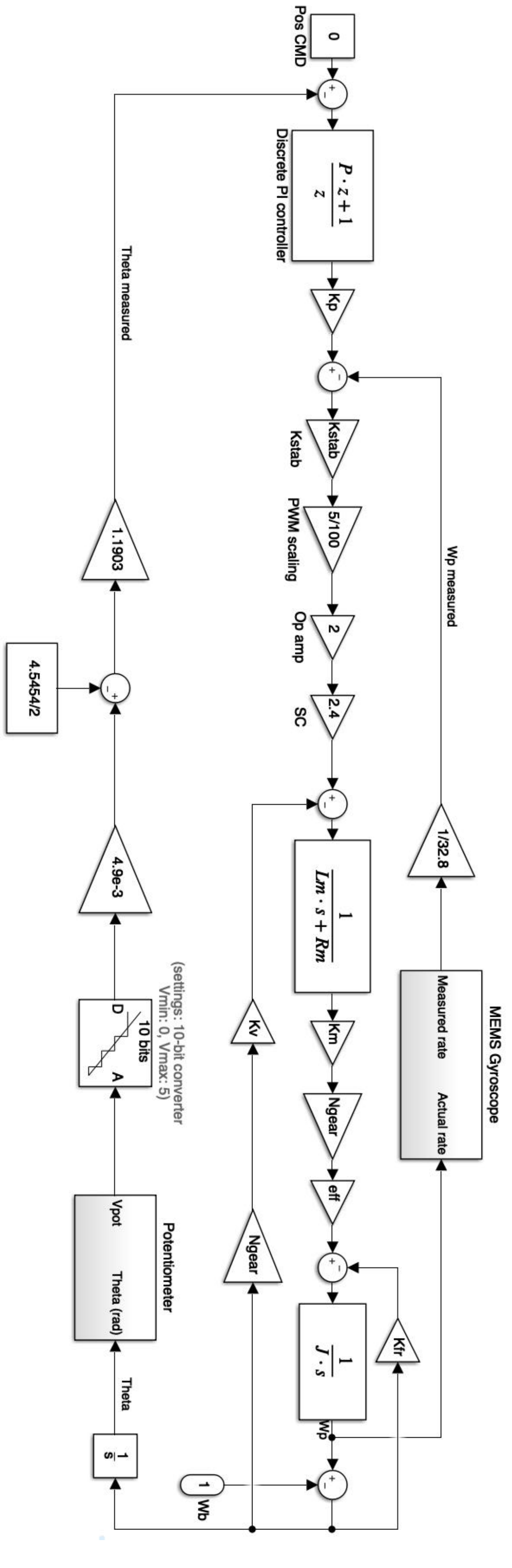


FIGURE 8.12: Updated block diagram

Chapter 9

Detailed Design

9.1 Mechanical Design

9.1.1 Parts

The mechanical design was made up of the following parts:

1. The handle
2. The gimbal frame
3. The motor mounting plate
4. The potentiometer mounting plate
5. The motor shaft connector
6. The potentiometer shaft connector
7. The gimbal arm
8. The smartphone mount

Handle

The handle serves as a grip for the user to hold while using the device. It is simply a cylinder that can be attached to the gimbal frame.

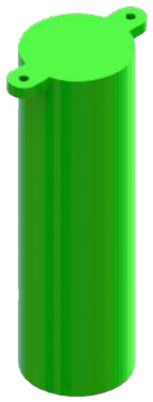


FIGURE 9.1: Part 1: Handle

Gimbal frame

The gimbal frame forms the outer structure upon which the motor and potentiometer sit, and to which the gimbal arm is coupled.

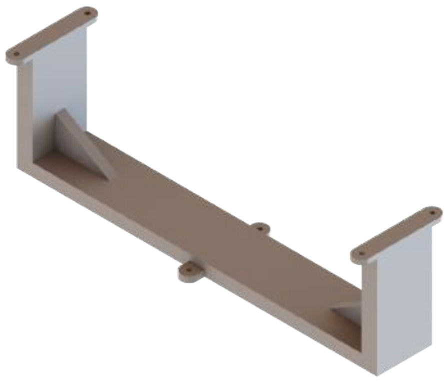


FIGURE 9.2: Part 2: Gimbal frame

Motor mounting plate

The motor mounting plate is what keeps the motor in place. The three $M2$ holes allow for the motor to be secured to the plate using bolts. The extruded slots allow for the plate to be connected to the gimbal frame at varying positions.

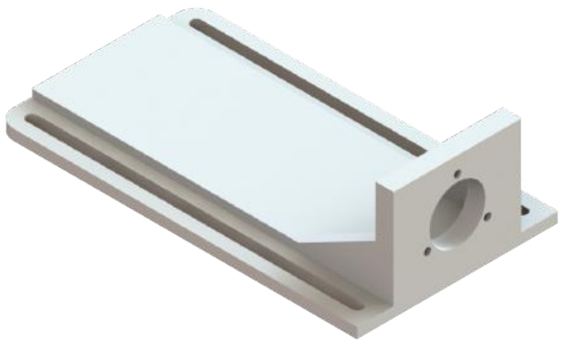


FIGURE 9.3: Part 3: Motor mounting plate

Potentiometer mounting plate

The potentiometer mounting plate is what keeps the potentiometer in place. Again, the extruded slots allow for the plate to be connected to the gimbal frame at varying positions.

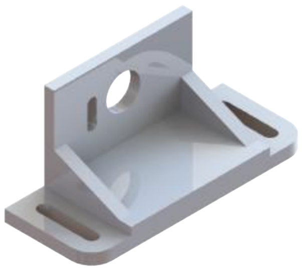
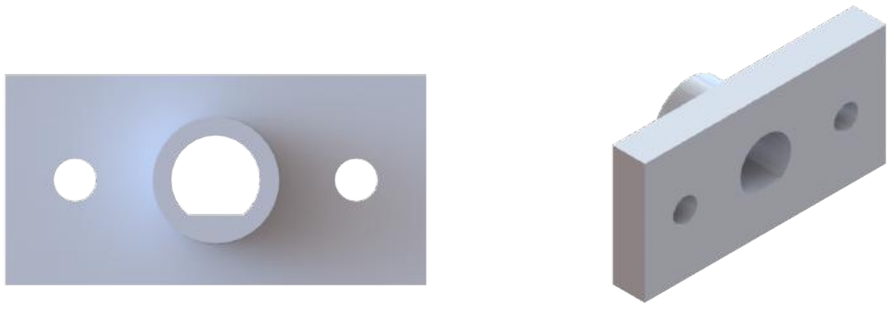


FIGURE 9.4: Part 4: Potentiometer mounting plate

Motor shaft connector

The motor shaft connector is used to couple the motor shaft to the gimbal arm.



(A) Front view

(B) Isometric View

FIGURE 9.5: Part 5: Motor shaft connector

Potentiometer shaft connector

The potentiometer shaft connector is used to couple the potentiometer shaft to the gimbal arm.

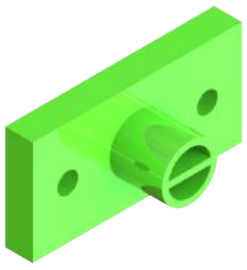


FIGURE 9.6: Part 6: Potentiometer shaft connector

Gimbal arm

The gimbal arm is connected to the motor and potentiometer through their respective connectors. The smartphone mount is then connected to the gimbal arm.

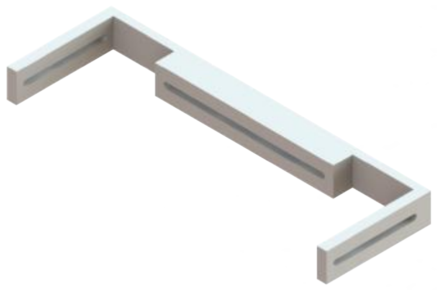
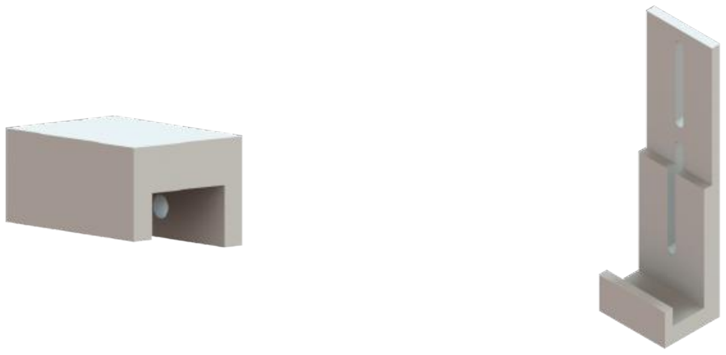


FIGURE 9.7: Part 7: Gimbal arm

Smartphone mount

The smartphone mount is used to hold the smartphone in place. It consists of two parts - the upper mount and the lower mount. The upper mount is connected to the lower mount using an *M*3 nut and bolt. The extruded slots in the lower mount allow for the upper mount to be secured at different positions - thus allowing for different sized smartphones to be held.



(A) Upper (B) Lower

FIGURE 9.8: Smartphone mount

9.1.2 Final Renders

All of the above parts were designed and assembled in *SolidWorks*. Figure 9.9 below shows the final renders of the mechanical design.

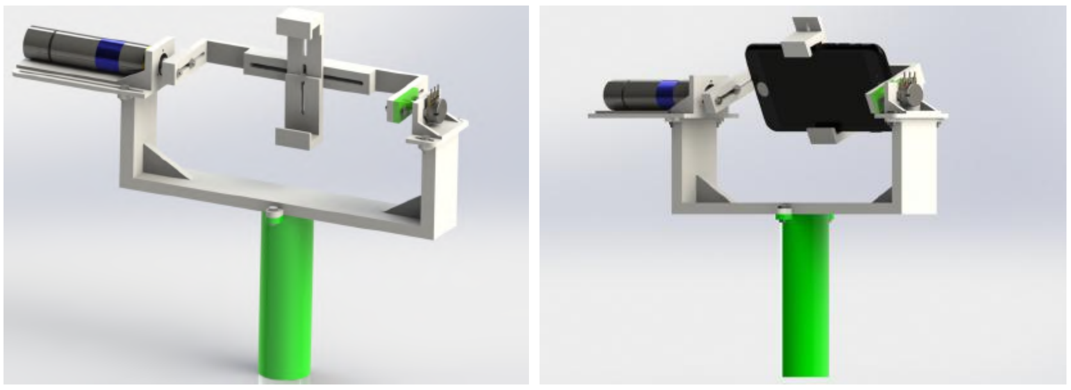

(A) (B)

FIGURE 9.9: Final renders

9.2 Electronics Design

A circuit was designed and built to act as an interface between the *Arduino* and the three sensors (two IMUs and one potentiometer). In addition to this, circuitry was required to connect the motor's speed controller to the *Arduino*.

9.2.1 Sensors

The following tables detail the pin configurations of all the sensors as well as the *Arduino* pin to which they should be connected.

Potentiometer

TABLE 9.1: Potentiometer pin description

Pin #	Pin Name	Description	Arduino Pin
1	V+	Positive input	5V
2	W	Wiper voltage	A3
3	GND	Ground	GND

A 1 $k\Omega$ resistor was connected to pin 1 of the potentiometer to avoid high current draw when the resistance of the potentiometer is set to zero. The following circuit diagram represents the potentiometer circuitry:

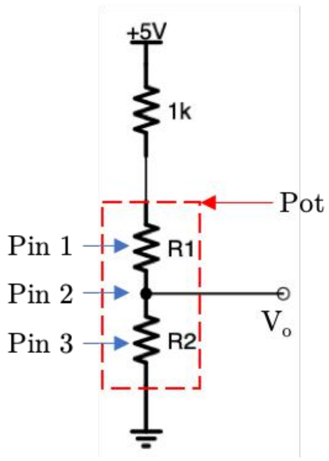


FIGURE 9.10: Representation of the potentiometer circuit.

Rotation of the pot will cause the values of $R1$ and $R2$ to change - however, the sum of $R1$ and $R2$ will always be 10 $k\Omega$. Using the voltage divider formula, the following equation can be derived:

$$V_o = 5\left(\frac{R2}{R1 + R2 + 1k}\right)$$

From the above equation, the minimum and maximum outputs of the circuit can be calculated. The minimum output is of course 0 V when $R2$ is set to zero. The maximum output occurs when $R2$ is set to its maximum of 10 $k\Omega$:

$$V_o = 5\left(\frac{10k}{0 + 10k + 1k}\right) = 4.5454V$$

The following graph plots the output voltage of the circuit against the rotation of the potentiometer:

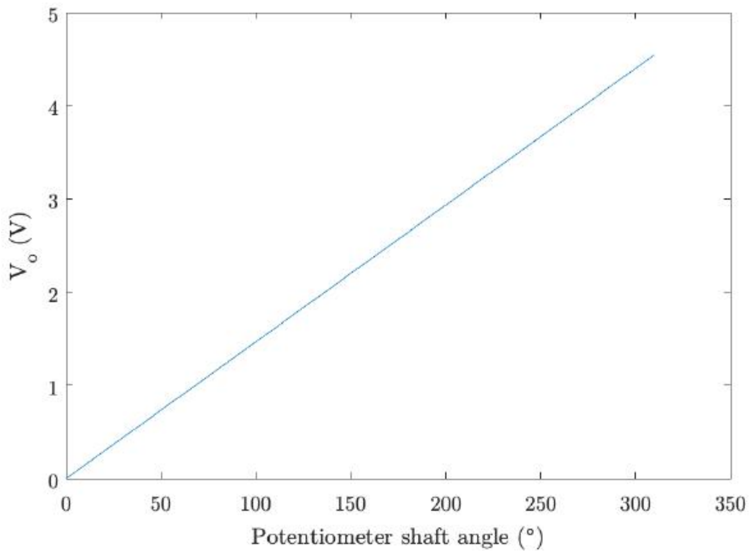


FIGURE 9.11: Output voltage of potentiometer circuit vs the angle of the potentiometer shaft.

MPU6050

TABLE 9.2: MPU6050 pin description

Pin #	Pin Name	Description	Arduino Pin
1	VCC	Positive input	5V
2	GND	Ground	GND
3	SCL	Serial clock	A5
4	SDA	Serial data	A4
~~5~~	~~XDA~~	~~Aux serial clock~~	-
~~6~~	~~XCL~~	~~Aux serial data~~	-
7	AD0	Address select	GND/5V
8	INT	Interrupt	2

Pins 6 and 7 have been crossed out because they were not needed. The *address select* pin (pin 7) is used to differentiate between devices. Connecting this pin to ground sets the I^2C address of the device to 0x68 while connecting it to 5 V sets the address to 0x69.

9.2.2 Speed Controller

With reference to Figure 9.12, the pins of the speed controller have the following functions:

TABLE 9.3: Speed controller pin description

Pin #	Pin Name	Function	Description
1	U_P	Power supply electronic	$5 \rightarrow 28V$
2	U_{mot}	Power supply motor	$6 \rightarrow 28V$
3	GND	Ground	-
4	U_{in}	Set speed value	$0 \rightarrow 10V$
5	DIR	Direction of rotation	$CCW < 0.5V$, $CW > 3V$
6	FG	Frequency output	6 lines per revolution

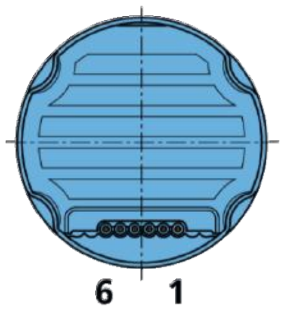


FIGURE 9.12: Pin configuration of integrated speed controller.
[$www.faulhaber.com/fileadmin/Import/Media/EN_2232_BX4_SC_DFF.pdf$]

Ref datasheet

The frequency output can be used to estimate the speed of the motor at high speeds. This, however, is not useful for this project's application and was therefore not used.

Because the *set speed value* pin has a range of zero to ten volts, the *LM358* operational amplifier was used to step up the voltage output of the *Arduino* (which has a maximum of five volts) by a factor of two - thus achieving the full voltage range.

Figure 9.13 below shows the circuit used to achieve a non-inverting amplification of two.

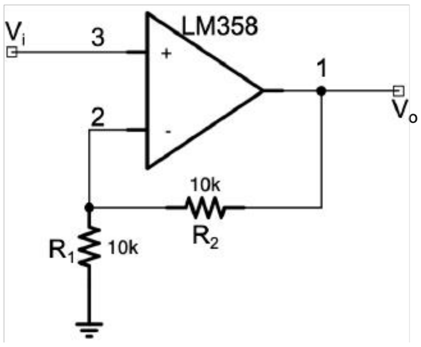


FIGURE 9.13: *LM358* operational amplifier circuitry. Configured for a non-inverting gain of 2. Note: The supply rails of the op-amp are not shown; these were $+24\ V$ and ground.

The resistor combination was calculated using the following equation [6]

$$\begin{aligned}\frac{V_o}{V_i} &= 1 + \frac{R_2}{R_1} = 2\\ &\Rightarrow \frac{R_2}{R_1} = 1\\ &\Rightarrow R_1 = R_2\end{aligned}$$

9.2.3 Final Circuit Board

Figure 9.14 belows shows the layout of the final circuit board. The circuitry was built on a 20×20 hole piece of veroboard. This board was designed to slot straight into the *Arduino* - this is shown by the *Arduino* pin-labels on either side. The blue pin-labels illustrate where the motor power supply should be connected.

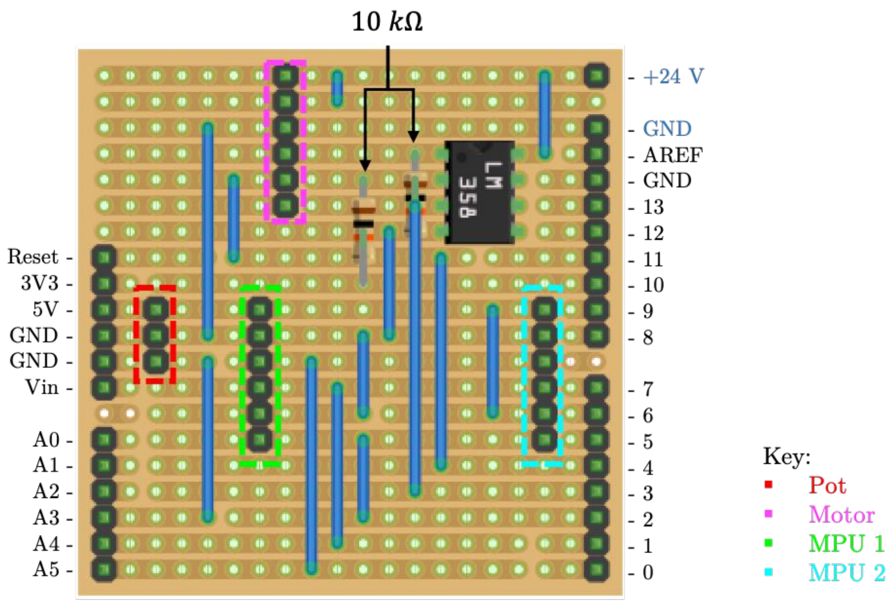


FIGURE 9.14: Final circuit board. The key illustrates where the motor and each sensor is connected.
[Made using Fritzing]

9.3 System Architecture

The following diagram illustrates the integration of the mechanical design and the electronics:

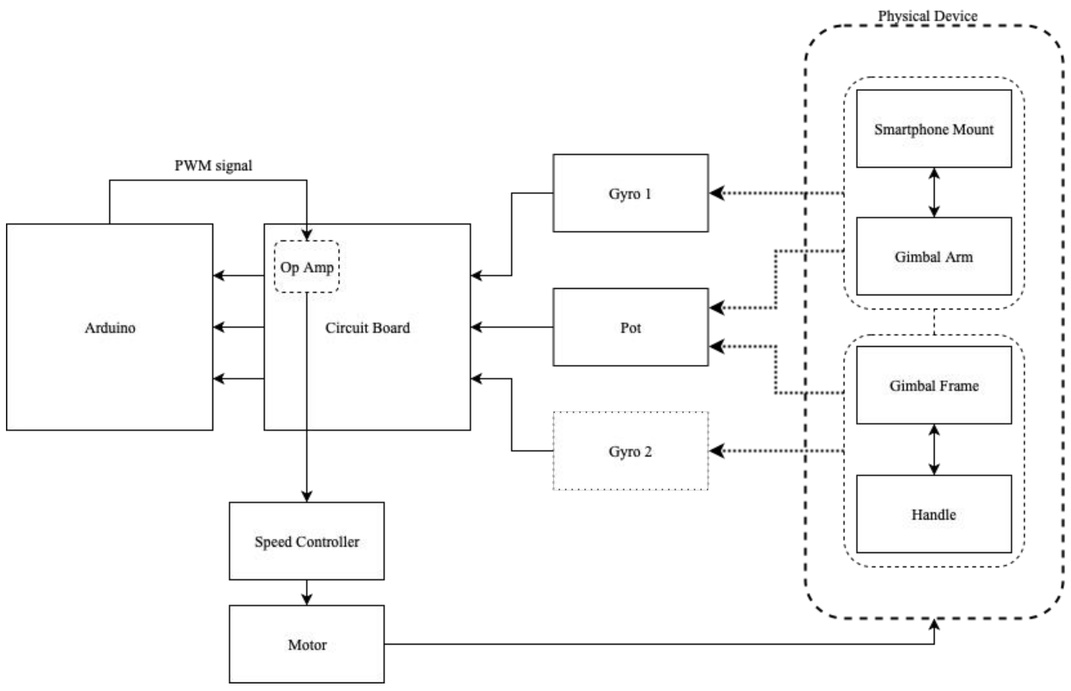


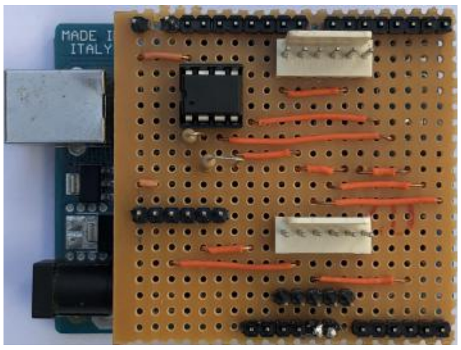
FIGURE 9.15: System architecture. Note: *Gyro 2* is not an essential part of the system; it is used for data capture.

Chapter 10

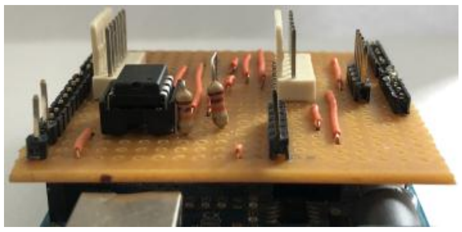
Physical Construction

10.1 Circuit Board Construction

The circuit board was constructed using components freely available at the University of Cape Town. Figure 10.1 below shows two photographs of the circuit board.



(A) Top view



(B) Side view

FIGURE 10.1: Assembled circuit board

10.2 Mechanical Construction

All the parts discussed in the previous chapter were 3D printed using the *Ultimaker 2+* 3D printers available at the University of Cape Town. The printing material used was *polylactic acid*, more commonly known as PLA - a thermoplastic regularly used for 3D printing.

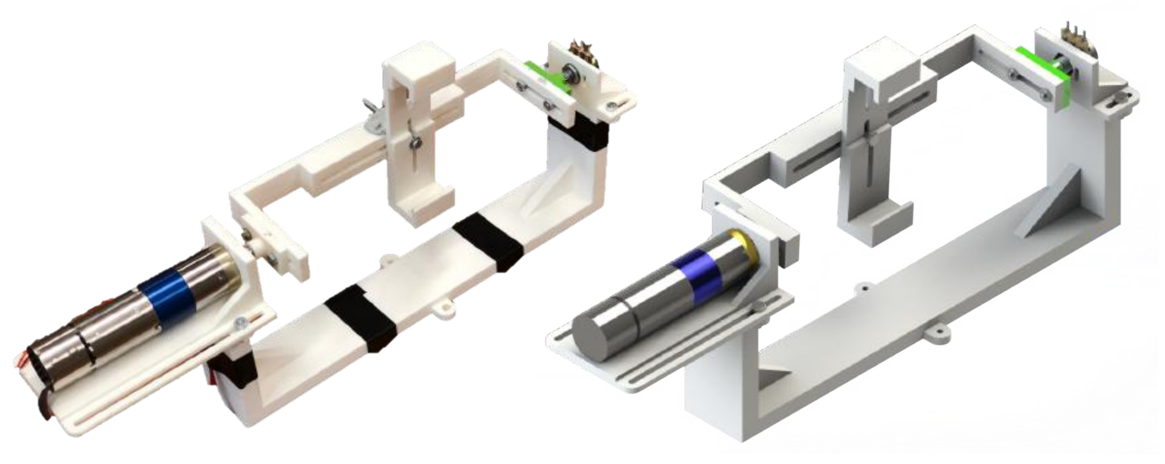
The process of converting the CAD model to a format understandable by the 3D printers is handled by a programme called *Cura*. Cura essentially converts a SolidWorks file to an adapted version of *g-code*[1] used in 3D printers.

Once all of the parts had been printed, the assembly process began.

The assembly process was a simple one. It merely involved securing the various parts to their respective positions using nuts and bolts. The entire device can be assembled in approximately five minutes.

The assembled model is shown below alongside the SolidWorks model:

[1]Programming language developed for the control of CNC machines.



(A) Assembled model (B) SolidWorks model

FIGURE 10.2: Assembled model vs Solidworks model

Note: The handle is not part of the photograph as it is difficult to balance the model while the handle is attached.
Note: The potentiometer model was downloaded from *GrabCad*.

Chapter 11

Plant Verification

This chapter concerns itself with verifying and improving the plant model to allow for more accurate simulations.

11.1 Motor Model Verification

First, the response of the motor will be analysed by applying various voltages to the speed controller and measuring the output speed with a gyroscope. Bear in mind that the speed controller has a range of 0 to 10 V which produces a voltage across the motor from 0 to 24 V. Therefore, the speed controller is initially represented by a gain of 2.4. The responses of the motor are illustrated by Figure 11.1 below:

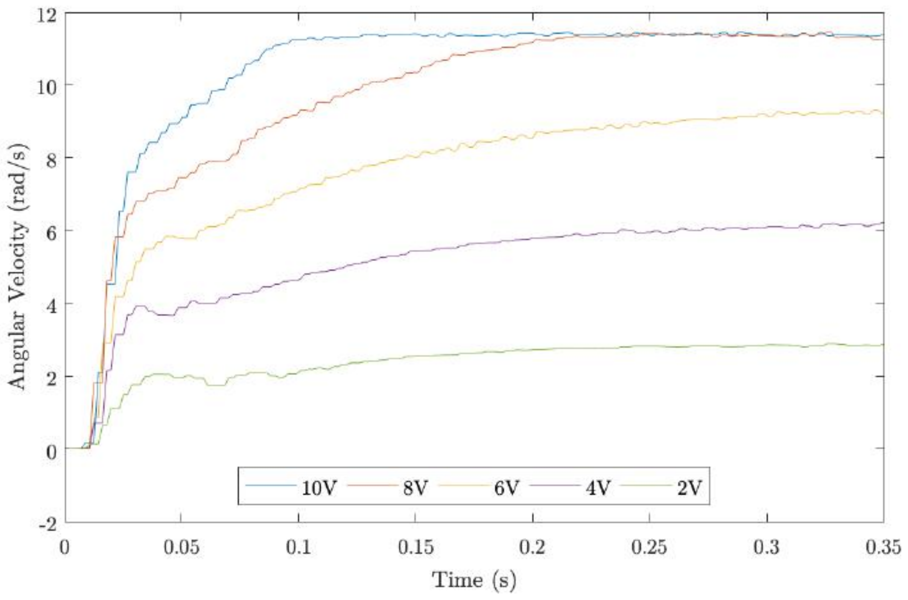


FIGURE 11.1: Response of the motor at different input voltages.

The response of the motor to a ramp input of 0 to 10 V is shown below:

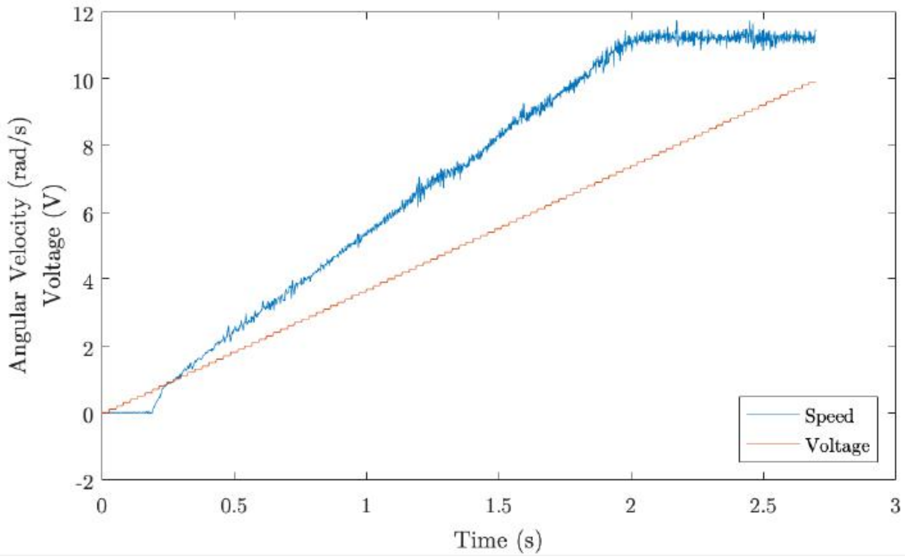


FIGURE 11.2: Response of the motor to a ramp input ($0 \to 10\ V$).

11.1.1 Actual vs Simulation

A simulation of the DC motor block diagram shown in Figure 7.3 was run. A gain of 2.4 was added to the input (V_m) of the block diagram to represent the effect of the speed controller.

A step input of 10 V at the input yielded the following results:

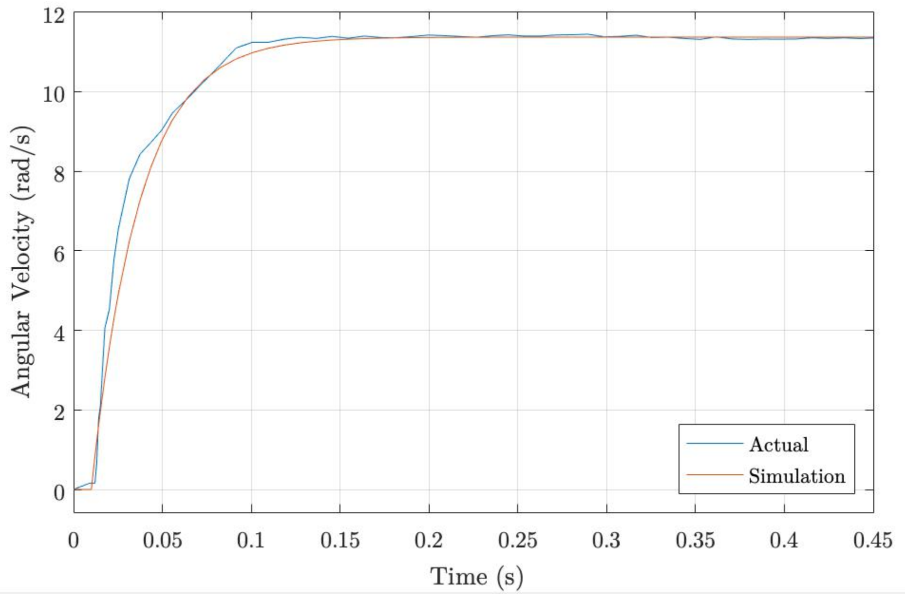


FIGURE 11.3: Comparison of actual and simulated motor response to a step input of 10 V.

This shows that the motor model accurately represents the actual dynamics of the motor. Unfortunately, this is only the case for the maximum input of 10 V. Lower step inputs yield the following results:

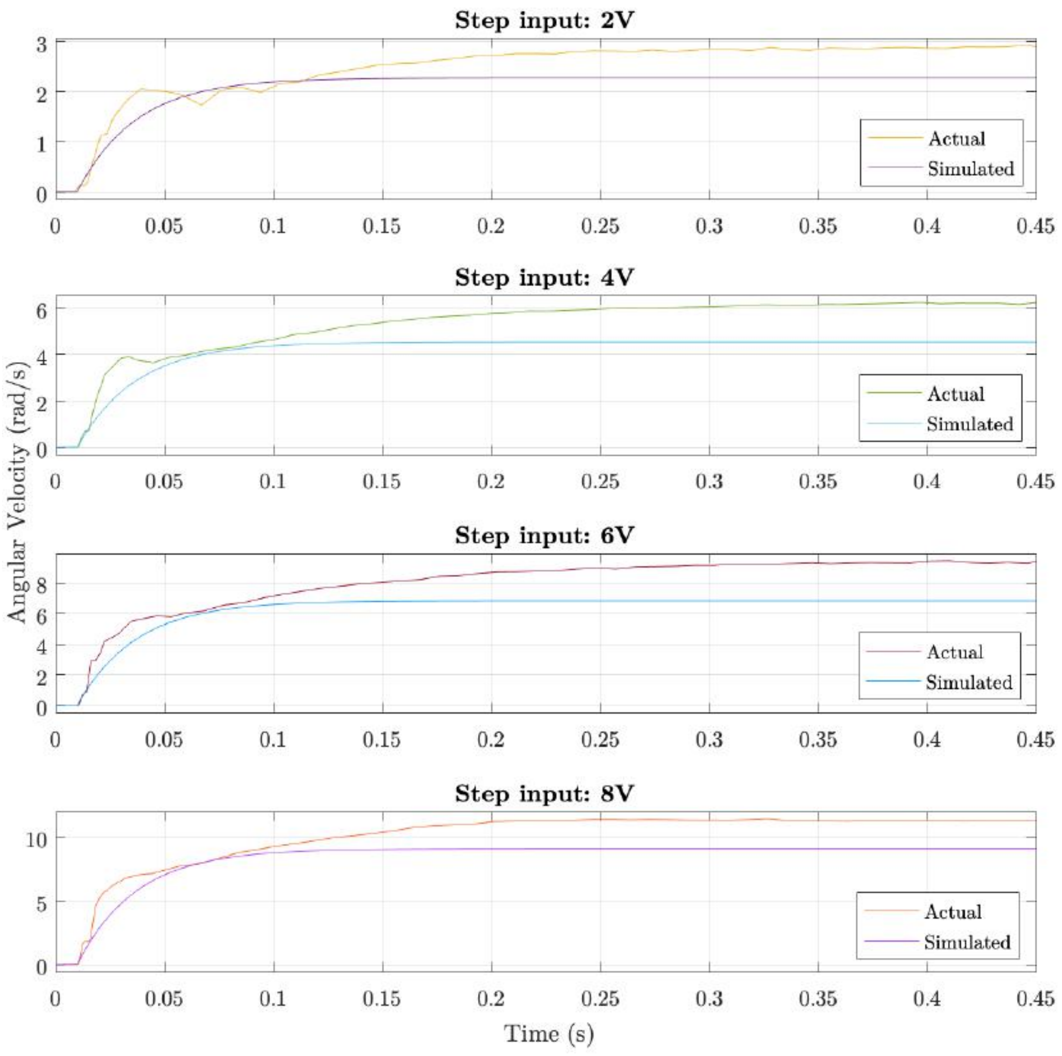


FIGURE 11.4: Comparison of actual and simulated motor response to varying step inputs.

The above graphs show that the motor model is not entirely accurate. This is due to the unknown gains of the integrated speed controller. It is evident from Figure 11.2 that the motor reaches its maximum speed before the voltage reaches its maximum of 10 V. This suggests that the speed controller has its own gain. Furthermore, it is evident from Figure 11.2 that there the controller has a *deadband*. According to the motor's datasheet, the motor will only start to rotate when the voltage input to the speed controller exceeds 0.3 V. By adding an additional gain of 1.43 and a deadzone from -0.3 to 0.3 V to the motor input and then limiting the input of the motor to 24 V, the effect of the speed controller can be modelled. Figure 11.5 below shows the response of the updated model to a ramp input:

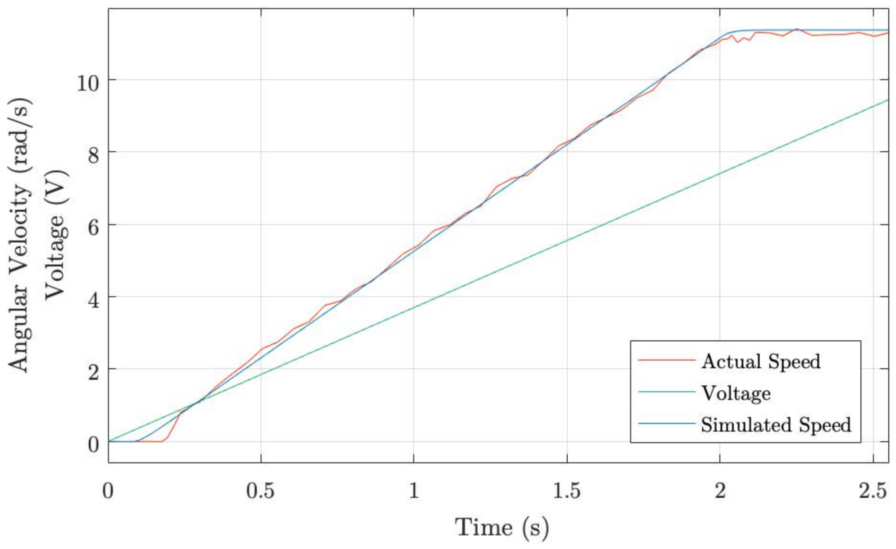
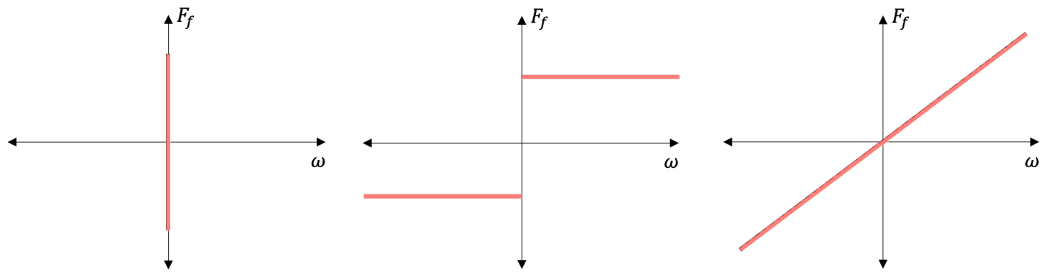


FIGURE 11.5: Comparison of actual and simulated motor response to a ramp input ($0 \rightarrow 10\ V$).

This graph shows that the model closely follows the actual motor response when given a ramp input. Upon closer inspection, one can see that the motor only starts moving at a voltage of about $0.7V$. This is caused by friction in the motor. In reality, friction is non-linear and therefore cannot be modelled as a gain. Non-linearities in control systems are undesirable as they decrease the validity of the control design techniques described in Chapter 3. There are three types of friction present in a motor: static, Coulomb, and viscous. Static friction is a constant value and is present when, and only when, the motor is not moving. Coulomb friction, also known as dynamic friction, is also a constant value (slightly lower than static friction) which is only present when the motor is rotating. Viscous friction is linearly dependent on the angular velocity of the motor; as the motor speed increases, so does the viscous friction. The three types of friction are illustrated below by plotting the frictional force against the angular velocity of the motor:



(A) Static friction (B) Coulomb friction (C) Viscous friction

FIGURE 11.6: Types of friction present in DC motors.

These three types of friction can be combine to produce the friction model for a DC motor:

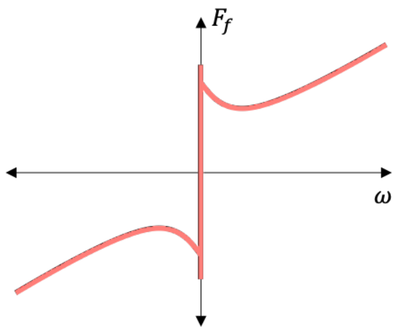


FIGURE 11.7: Combined friction model of DC motor

This model replaces the K_{fr} block in the motor model. It is modelled using *Simulink*'s *Coulomb and Viscous Friction* block. This block automatically sets the static friction to the same value as the Coulomb friction; this has a negligible effect as the Coulomb and static frictions are very similar. The Coulomb friction value was set to $0.01Nm$ and the viscous friction was left as $K_{fr} = 0.001Nm/(rad/s)$.
The response of the updated model to varying step inputs is shown below:

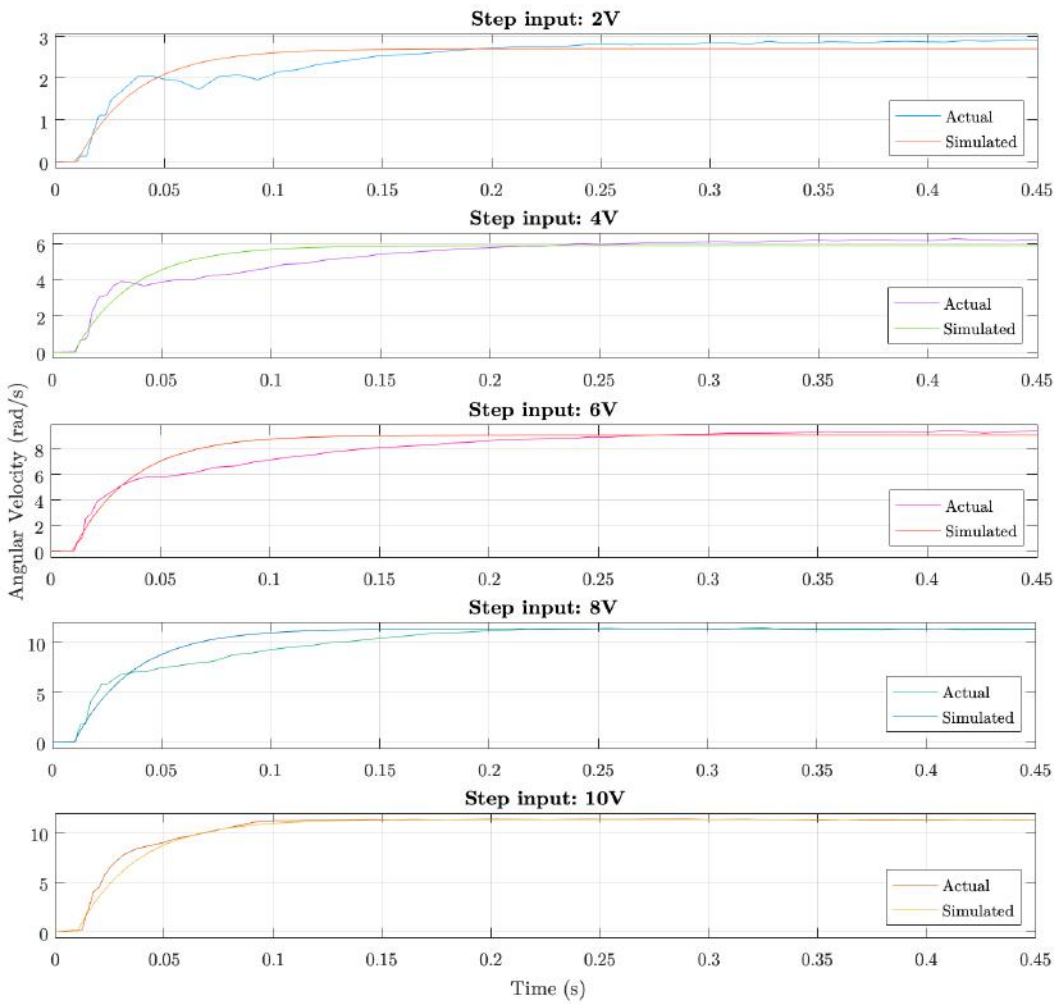


FIGURE 11.8: Comparison of actual and simulated motor response to varying step inputs.

These responses are significantly more accurate than the previous ones (Figure 11.4).

11.2 Moment of Inertia

The last way in which the plant model was improved was by determining the combined moment of inertia of the rotor, gimbal structure and smartphone.

To empirically determine the moment of inertia of a motor and its load, the following steps can be followed:

1. Apply a step command to the motor to achieve its maximum velocity.
2. Record the velocity profile using a gyroscope.
3. Take note of the max current draw (I_{pk}).
4. Differentiate the velocity profile to generate the acceleration profile.
5. The peak torque output is calculated as the peak current drawn (I_{pk}) times the torque constant of the motor (K_m).
6. The moment of inertia of the entire load is the peak torque divided by the maximum acceleration (α_{pk}).

After following these steps, the subsequent results were found:

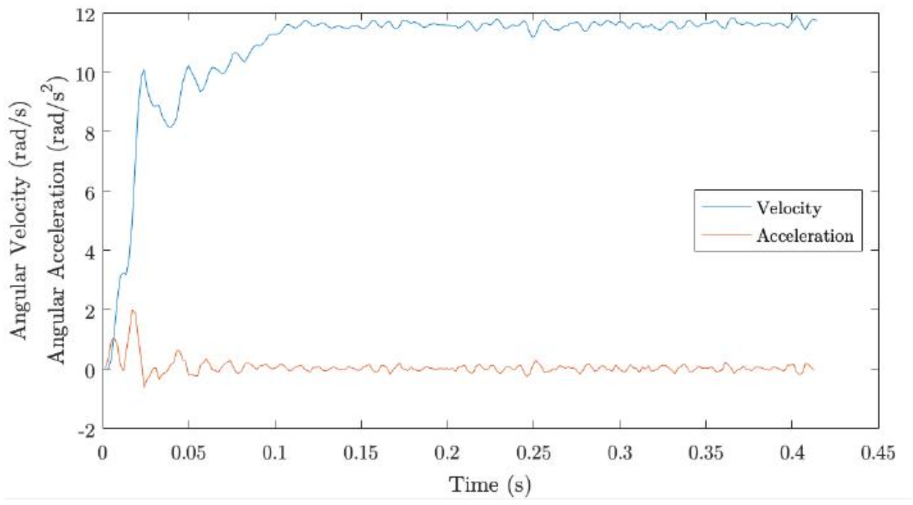


FIGURE 11.9: Speed and acceleration of motor and load given 10 V input.

The maximum acceleration was 1.948 rad/s^2 and the peak current draw was 0.13 A. The peak torque was therefore:

$$\begin{aligned}T_{pk} &= I_{pk} \times K_m \\ &= 0.13 \times 0.0314 \\ &= 4.082 mNm\end{aligned}$$

The empirically calculated moment of inertia is therefore:

$$\begin{aligned} J &= \frac{T_{pk}}{\alpha_{pk}} \\ &= \frac{4.082 \times 10^{-3}}{1.948} \\ &= 2.095 \times 10^{-3} kg \cdot m^2 \end{aligned}$$

Chapter 12

Digital Implementation

This chapter discusses the process of implementing the simulation on a microcontroller. The full code used in the project can be found in Appendix B.

12.1 Controller Implementation

This section draws from the difference equations discussed in Section 3.9.

The inner stabilisation loop uses a P controller of the form K_{stab}. The digital implementation of this P controller is simply a gain of K_{stab} in the code.

The outer loop, however, uses a PI controller of the form $\frac{p \cdot s+1}{s}$. First, this must be converted to the z-domain, as done in Section 3.8. The resulting z-transform of the PI controller, $C(s)$ is:

$$C(z) = \frac{(T+2p)z + (T-2p)}{2z-2}$$

Which can then be converted to a difference equation:

$$\begin{aligned}
&\frac{C_{out}}{C_{in}}(z) = \frac{(T+2p) + (T-2p)z^{-1}}{2-2z^{-1}} \\
&\Rightarrow 2C_{out}(z) - 2z^{-1}C_{out}(z) = (T+2p)C_{in}(z) + (T-2p)z^{-1}C_{in}(z) \\
&\Rightarrow 2C_{out}(k) - 2C_{out}(k-1) = (T+2p)C_{in}(k) + (T-2p)C_{in}(k) \\
&\Rightarrow C_{out}(k) = \frac{(T+2p)C_{in}(k) + (T-2p)C_{in}(k) + 2C_{out}(k-1)}{2}
\end{aligned}$$

This difference equation can be implemented in code.

12.2 MPU6050 Communication

The MPU6050 IMU uses the I^2C protocol to communicate. The *Arduino Wire.h* library is an open-source library that provides high-level functionality such as reading from, and writing to I^2C devices. This library was used to expedite the process of digital implementation.

Functions specific to the MPU6050 were taken from *Jeff Rowberg's* open-source *Github* page. These functions allowed one to do the following:

- initialise the device,
- set the full scale gyroscope range,

- configure the digital low pass filter,
- configure offsets for each axis, and,
- read the gyroscopic output for each axis.

The full scale range of the gyroscope was set to 1000 dps - a value well above the expected operating range of the ISP.

The low pass filter was configured to the maximum bandwidth of $256Hz$ as this allowed for the fastest sampling frequency (8 kHz).

Gyroscopes are subject to *sensor bias* (or offset) which results in the gyroscope outputting a non-zero reading when completely still. Fortunately, this bias is mostly constant and can therefore be accounted for using an offset. There is a dedicated offset register on the MPU6050 for each axis - the value in this register is subtracted from each gyroscope reading before the reading is sent to the microcontroller. The bias of a gyroscope can easily be calculated by taking a large number (thousands) of readings while the gyroscope is completely still. These readings can be summed and then divided by the total number of readings to give the offset value.

12.3 Potentiometer Interface

As shown in Section 9.2.1, the potentiometer circuit has a range of $0 \rightarrow 4.5454V$. The analogue to digital converter (ADC) of the *Arduino* has a voltage range of $0 \rightarrow 5V$ and a resolution of 10 $bits$. Therefore the ADC has a sensitivity of:

$$\frac{5}{2^{10}-1} = 4.9 \times 10^{-3} V/unit = 204 units/V$$

Therefore, the maximum output of the potentiometer circuit (4.545 V) will result in an ADC reading of $204 \times 4.545 = 927 units$. Therefore, the sensitivity of the ADC reading is:

$$\frac{310^\circ}{927\ units} = 0.334\ ^\circ/unit$$

The output voltage of the potentiometer can be read into the *Arduino* by using the *analogRead*(*<pin number>*) function available in the *Arduino* IDE[1].

12.4 Speed Controller Interface

The speed controller is controlled using the *set speed value* pin and the *direction* pin.

The direction pin expects a digital high or low value which determines the direction of rotation. This pin can be toggled using any of the GPIO pins on the *Arduino* and the function *digitalWrite*(*<pin number>, <HIGH or LOW>*).

The *set speed value* pin expects a PWM signal ($0 \rightarrow 10V$) which determines the speed of the motor. This pin must therefore be connected to one of the six PWM pins

[1]Integrated development environment

available on the *Arduino*. The effective voltage of the PWM signal can be set using the $analogWrite($*<pin number>, <0 ... 225>*$)$[2] function. The second argument of this function should be an integer between 0 and 255; this scales the effective PWM between 0 and $5V$. Unfortunately, the frequency of this PWM signal is too low - about $500Hz$. At such a low frequency, the motor heats up and makes vibrational noises. The frequency of the PWM signal had to be increased to $1.5kHz$ in order for the motor to operate as expected. This was done by directly accessing the timer registers of the *Arduino* - specifically the *output compare* registers of the timer. This was done by following an online tutorial [12].

[2]Contrary to the function name, this is a digital signal. It is, in fact, a square wave signal.

Chapter 13

Testing and Results

This chapter presents the results of the testing phase. As discussed in Section 4.9, the results were captured using a second gyroscope mounted to the base. The data from the two gyroscopes and the potentiometer were then recorded using the Arduino serial monitor and plotted using MATLAB. In addition to this, the base motion recorded by the base gyroscope was fed into the Simulink simulation - thus allowing for the actual results to be compared to the actual results.

13.1 Actual Results

In the following tests, the constants listed above the plots are representative of the time-constants and gains in the s-domain; these values were of course converted to their z-domain equivalents in the code.

13.1.1 Step Tests

The following tests were conducted by applying a step input to the base angle. Three graphs have been produced for each step test:

1. The first graph plots the measured base velocity and the measured platform velocity.
2. The second graph plots the base angle and the LOS of the platform. These two curves are obtained by integrating base velocity and platform velocity respectively.
3. The third graph plots the base angle and the measured relative angle between the base and the platform.

Test 1

Graph 1:

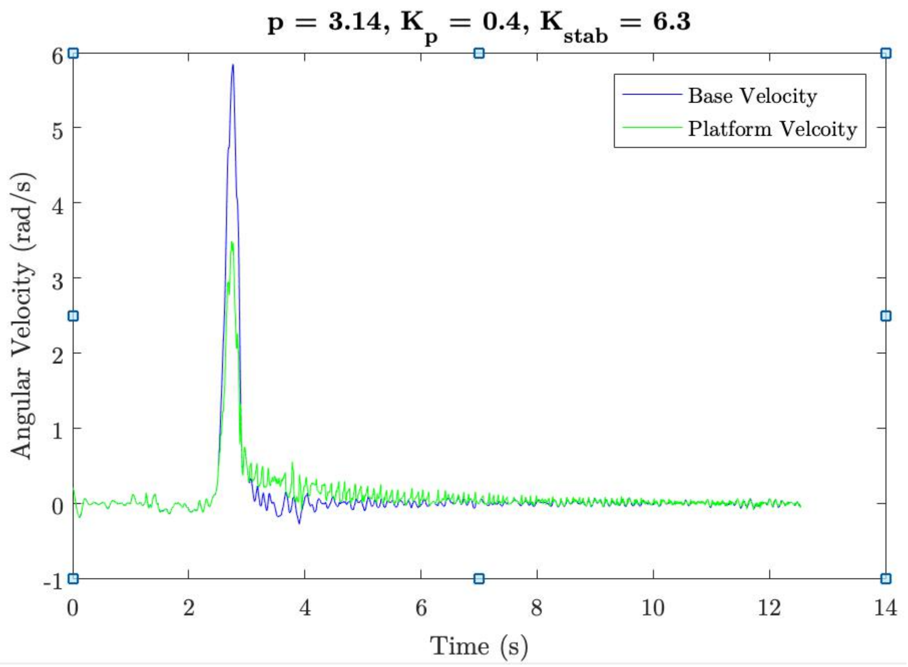


FIGURE 13.1: Base velocity vs. platform velocity

Graph 2:

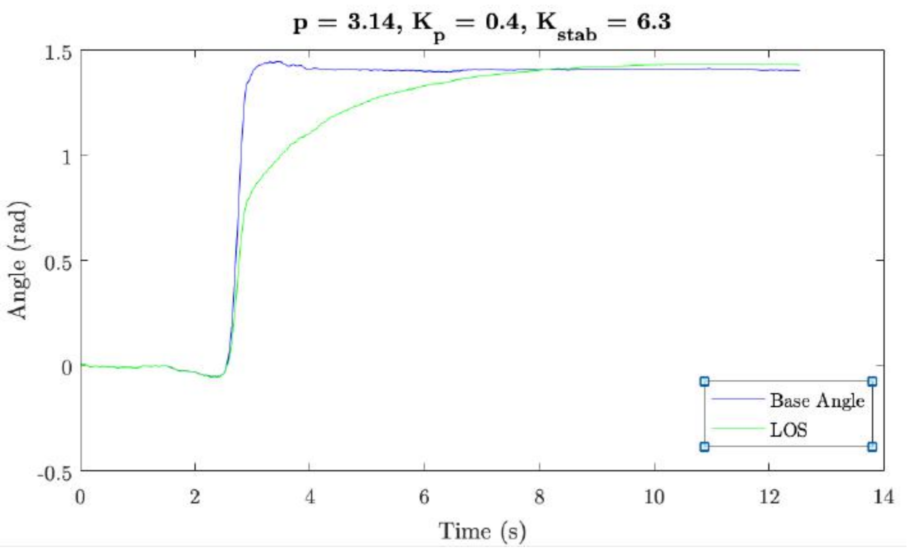


FIGURE 13.2: Base angle vs. platform LOS

Graph 3:

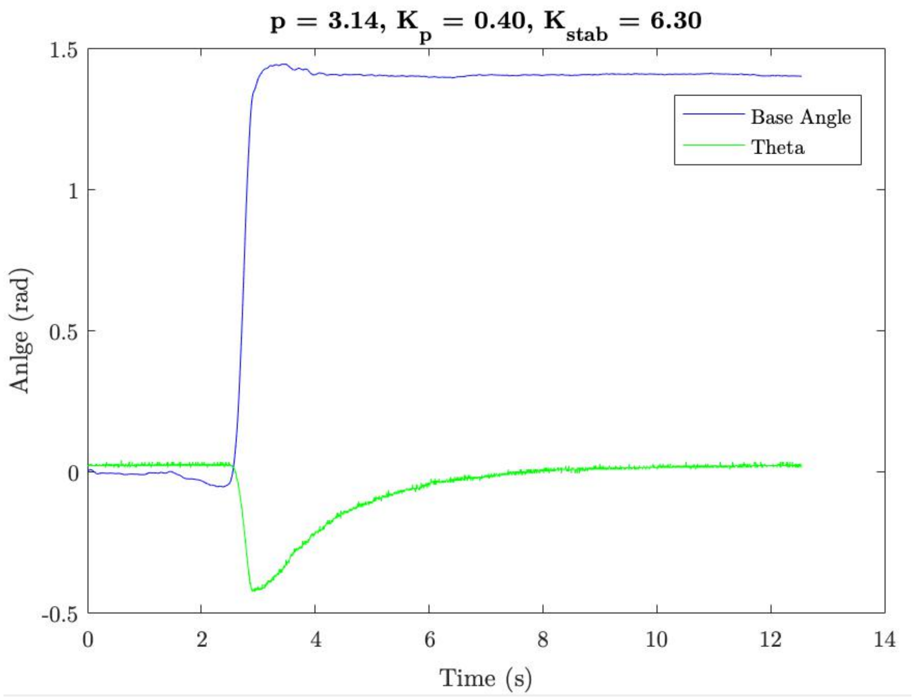


FIGURE 13.3: Base angle vs. Theta

Test 2

<u>Graph 1:</u>

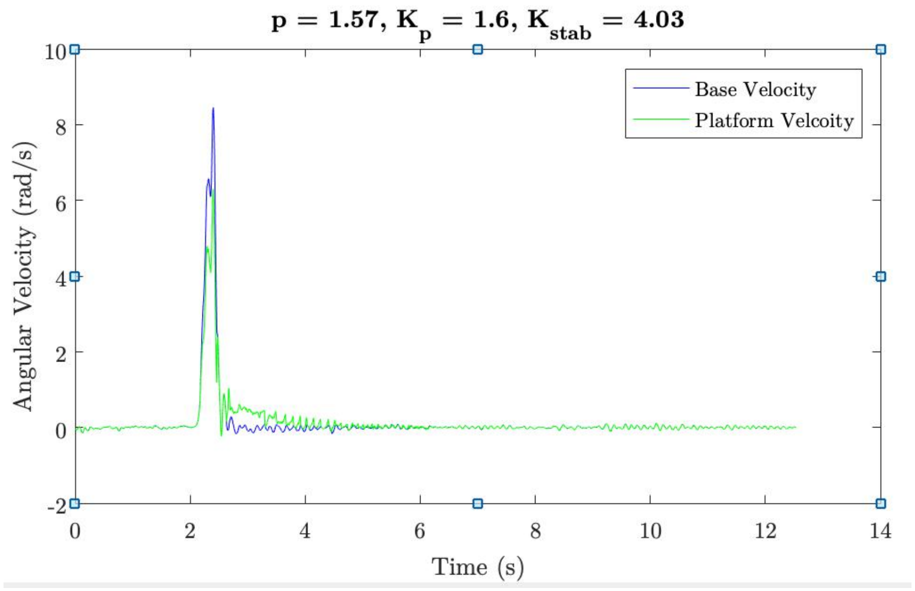


FIGURE 13.4: Base velocity vs. platform velocity

<u>Graph 2:</u>

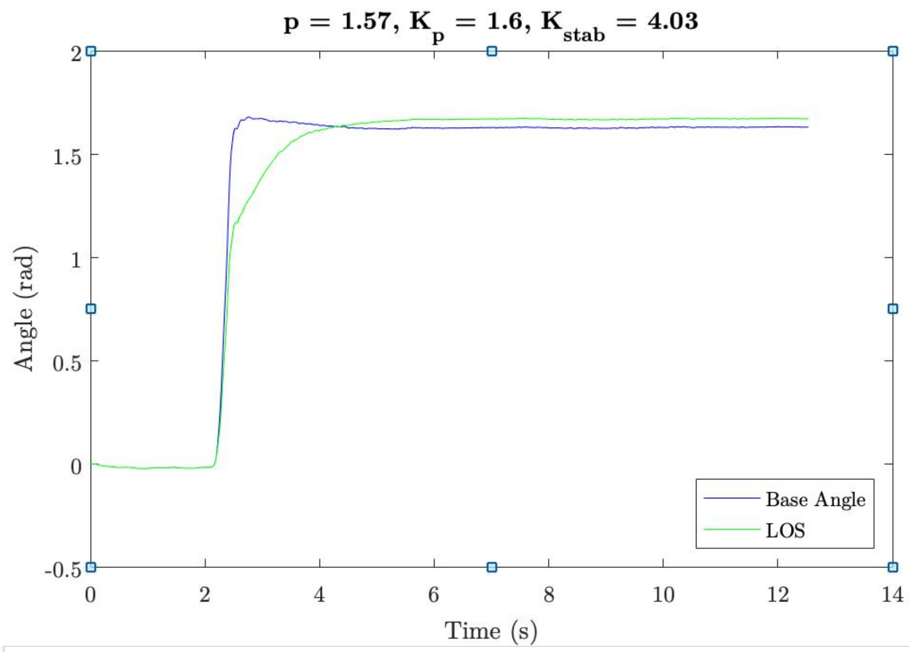


FIGURE 13.5: Base angle vs. platform LOS

Graph 3:

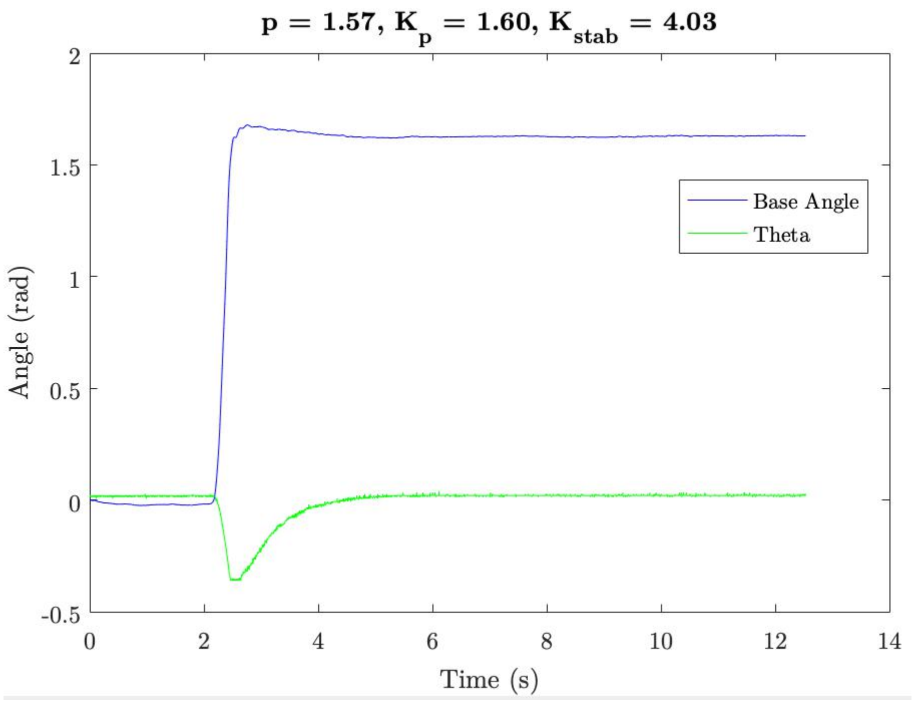


FIGURE 13.6: Base angle vs. Theta

13.1.2 Sinusoidal Tests

The following tests were conducted by applying sinusoidal motion to the base. Again, these tests were performed by hand. The same three graphs were produced for the these tests.

Test 1

Graph 1:

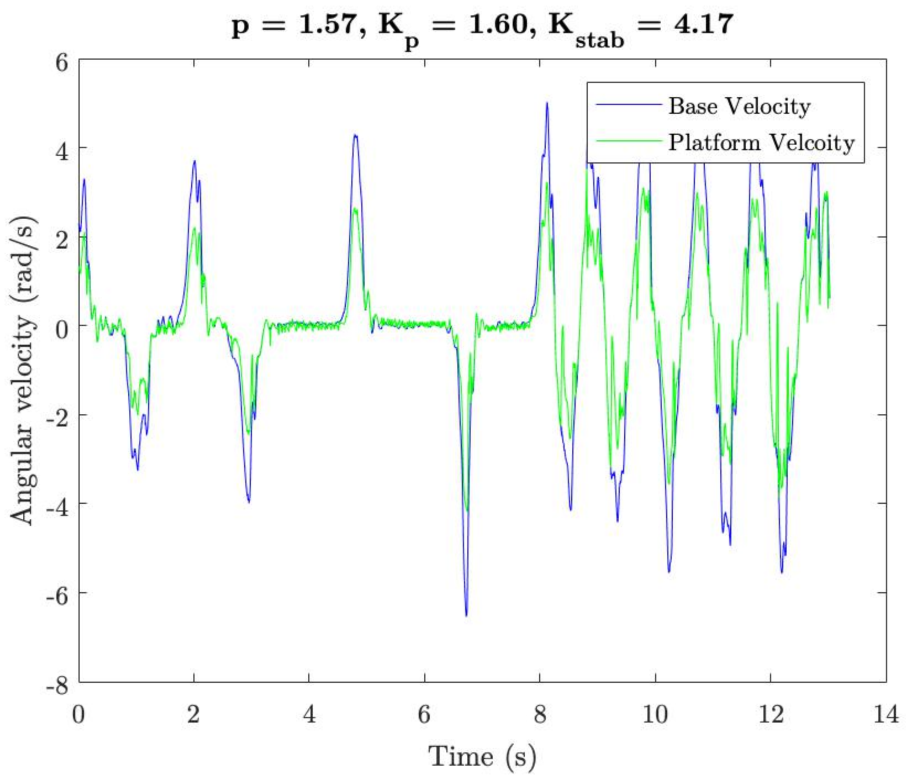


FIGURE 13.7: Base velocity vs. platform velocity

Graph 2:

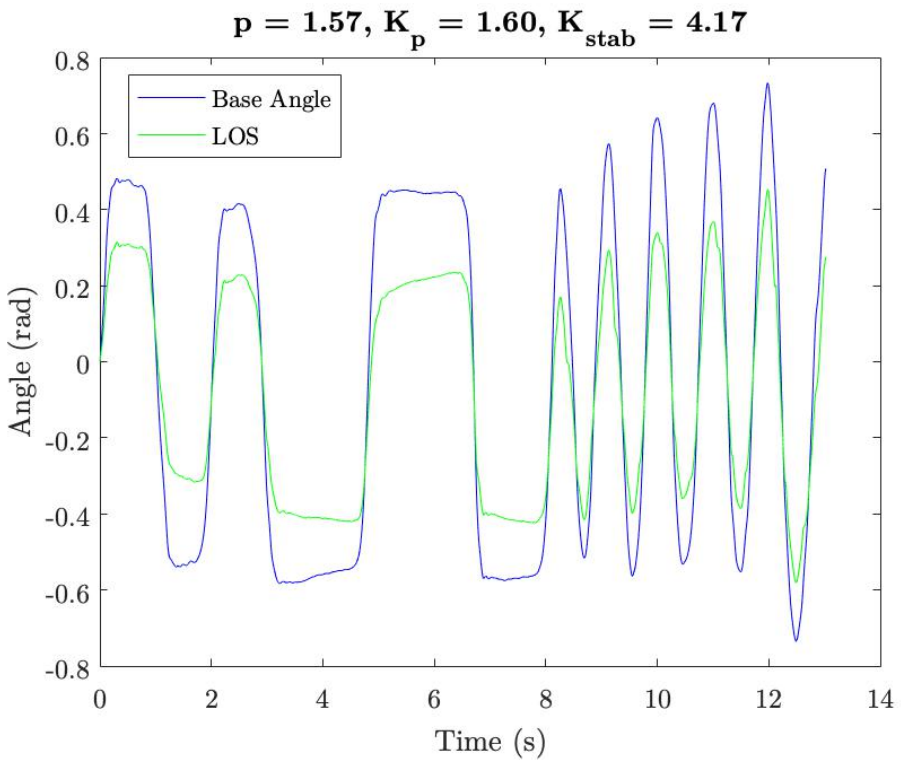


FIGURE 13.8: Base angle vs. platform LOS

Graph 3:

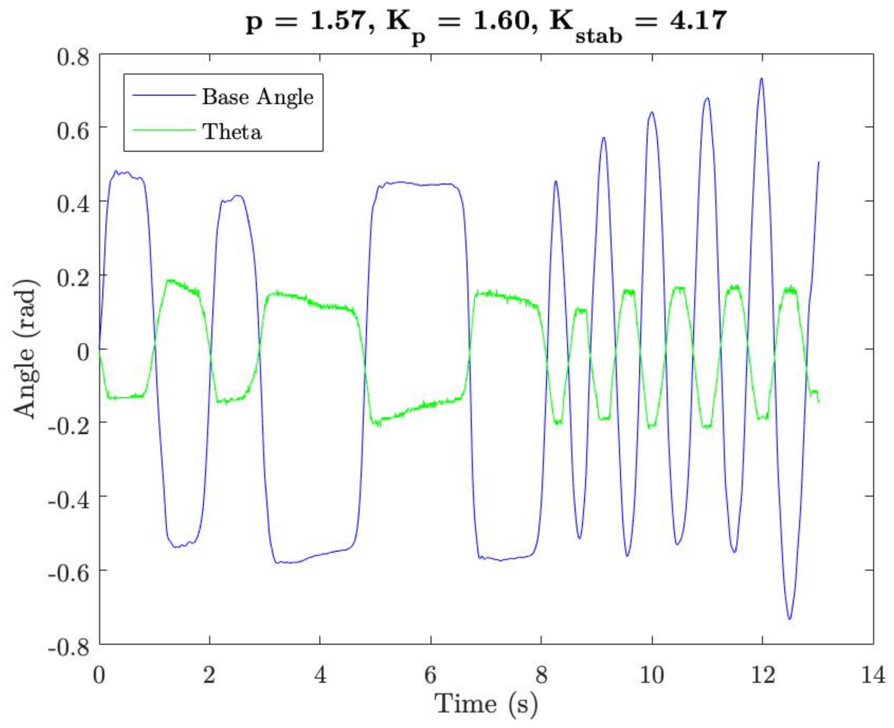


FIGURE 13.9: Base angle vs. Theta

Test 2

Graph 1:

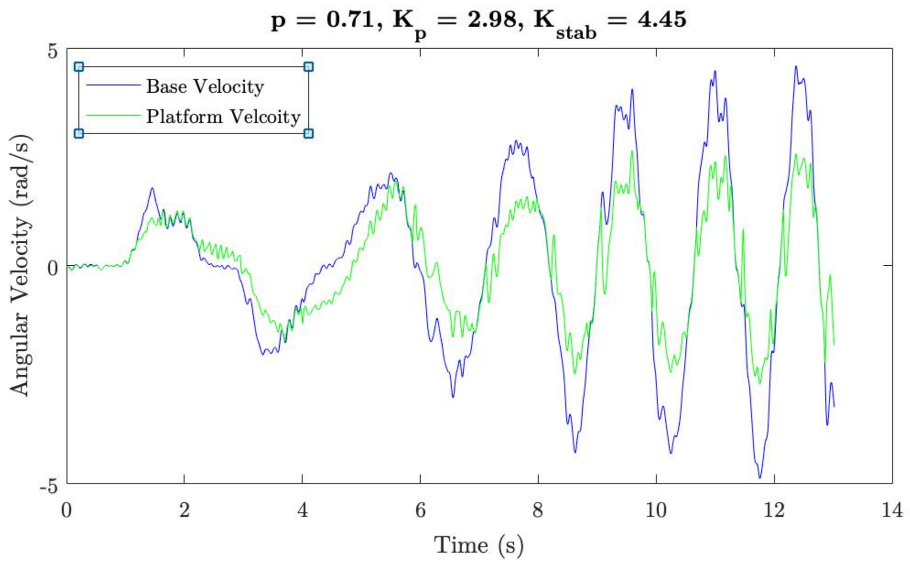


FIGURE 13.10: Base velocity vs. platform velocity

Graph 2:

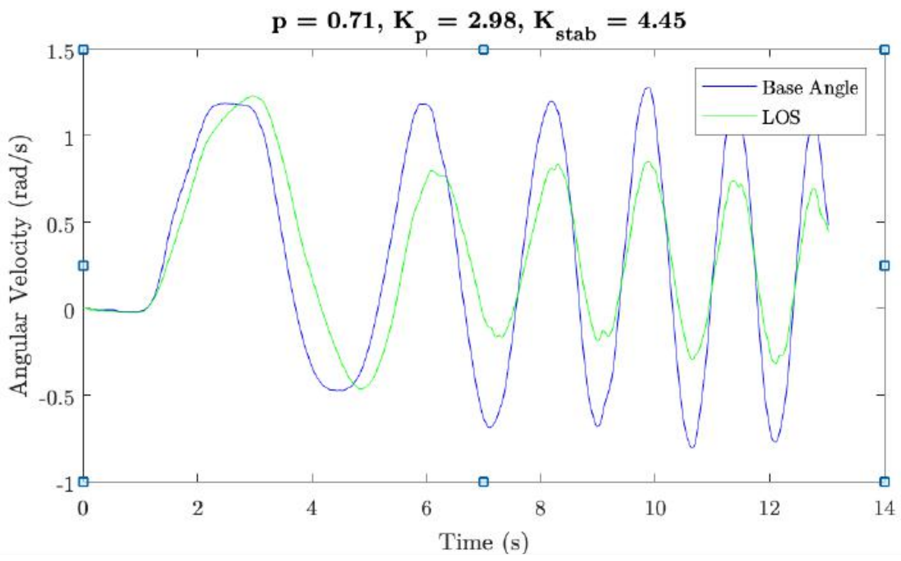


FIGURE 13.11: Base angle vs. platform LOS

Graph 3:

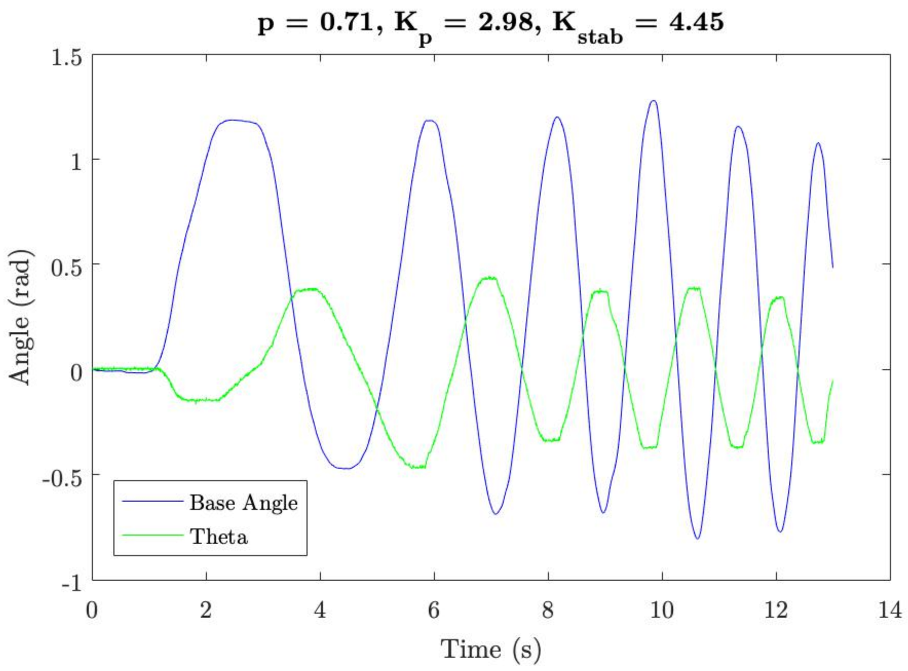


FIGURE 13.12: Base angle vs. Theta

13.2 Comparison between Simulation and Actual

The following plots compare the simulation results to the actual results by plotting the same signals on the same set of axes. The first subplot of each figure is the base motion applied to the system; because this is the input signal to the simulation, the actual and simulated base motion signals are exactly the same. The test type and number (e.g. *Step Test: Test 1*) corresponds to the same test type and number in the previous section.

13.2.1 Step Tests

Test 1

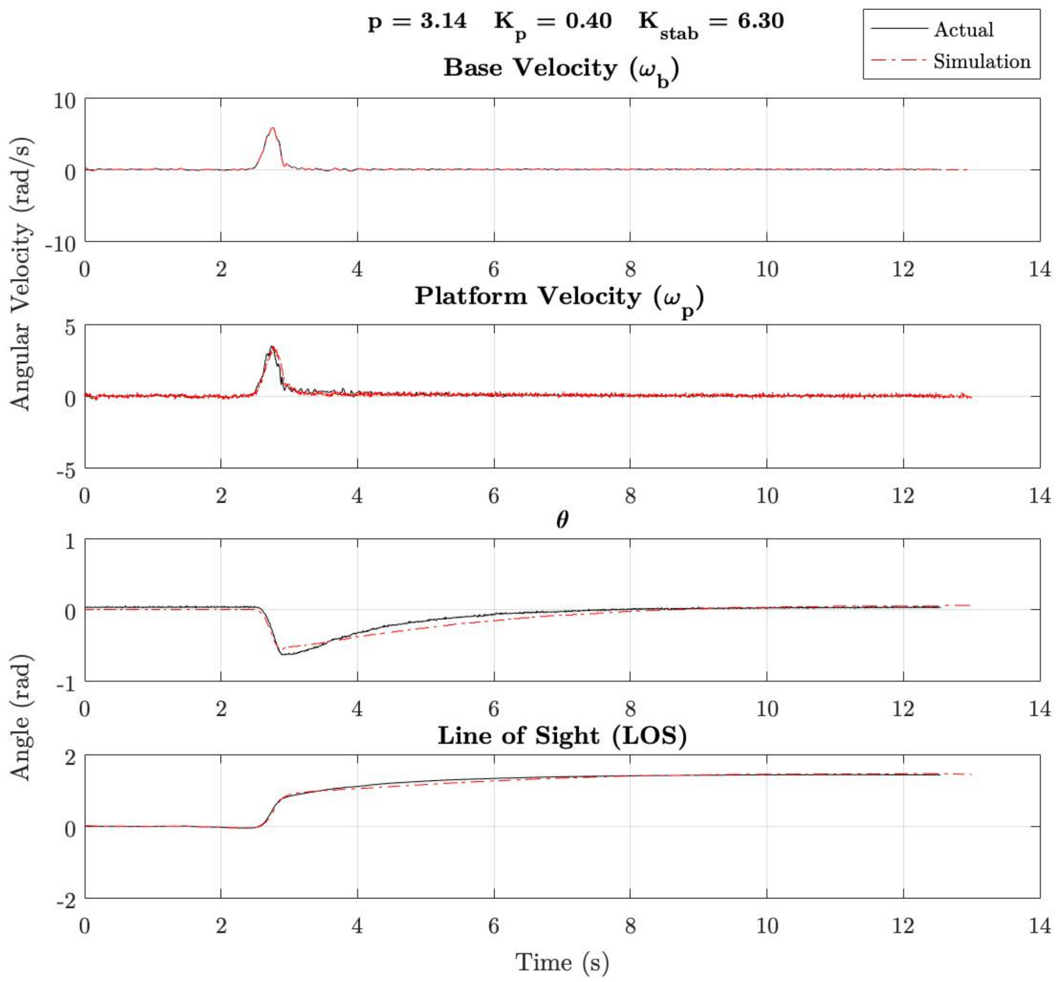


FIGURE 13.13: Comparison between simulation and actual results.

Test 2

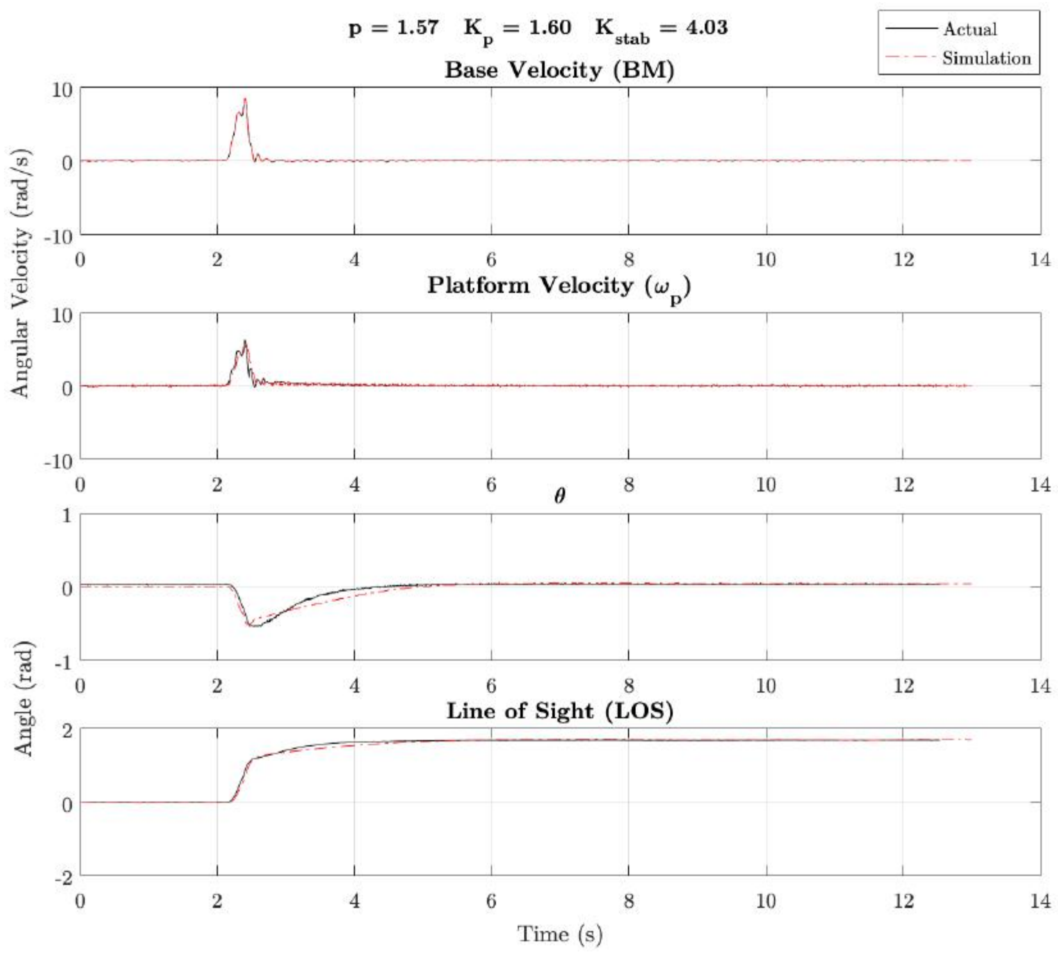


FIGURE 13.14: Comparison between simulation and actual results.

13.2.2 Sinusoidal Tests

Test 1

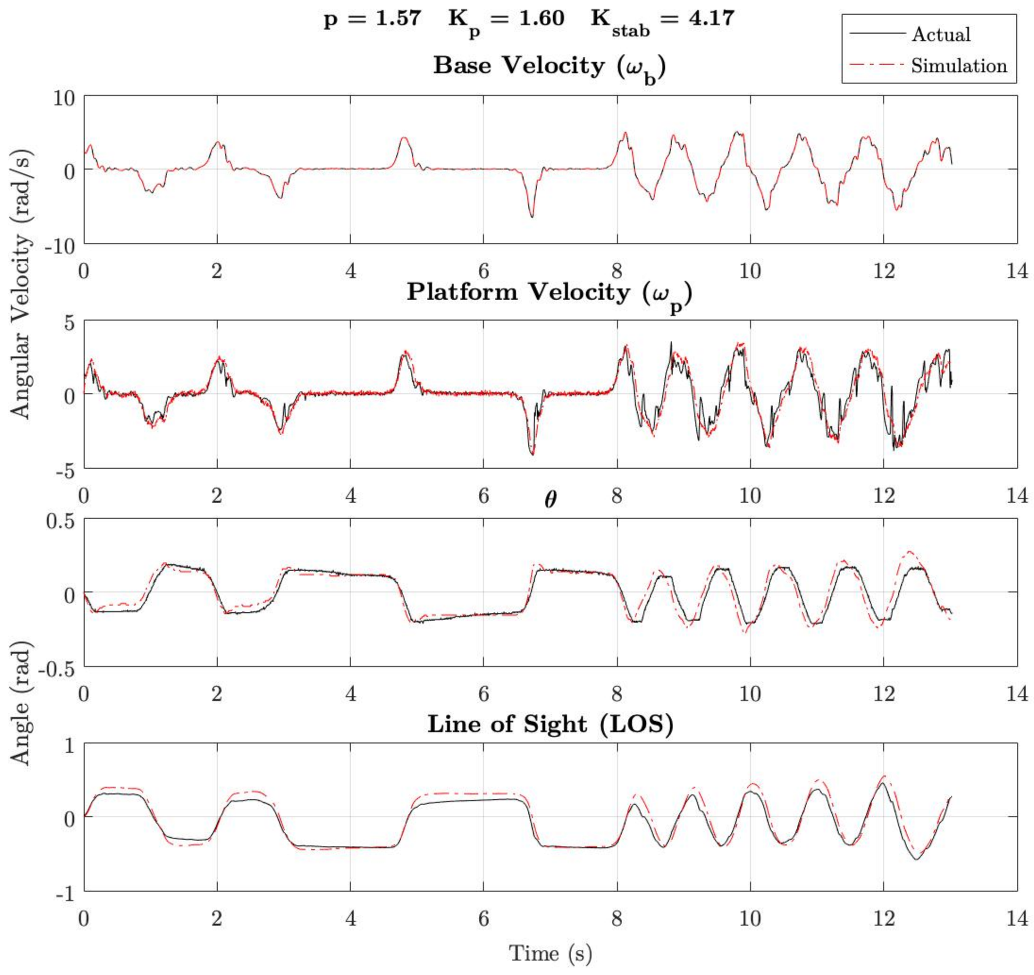


FIGURE 13.15: Comparison between simulation and actual results.

Test 2

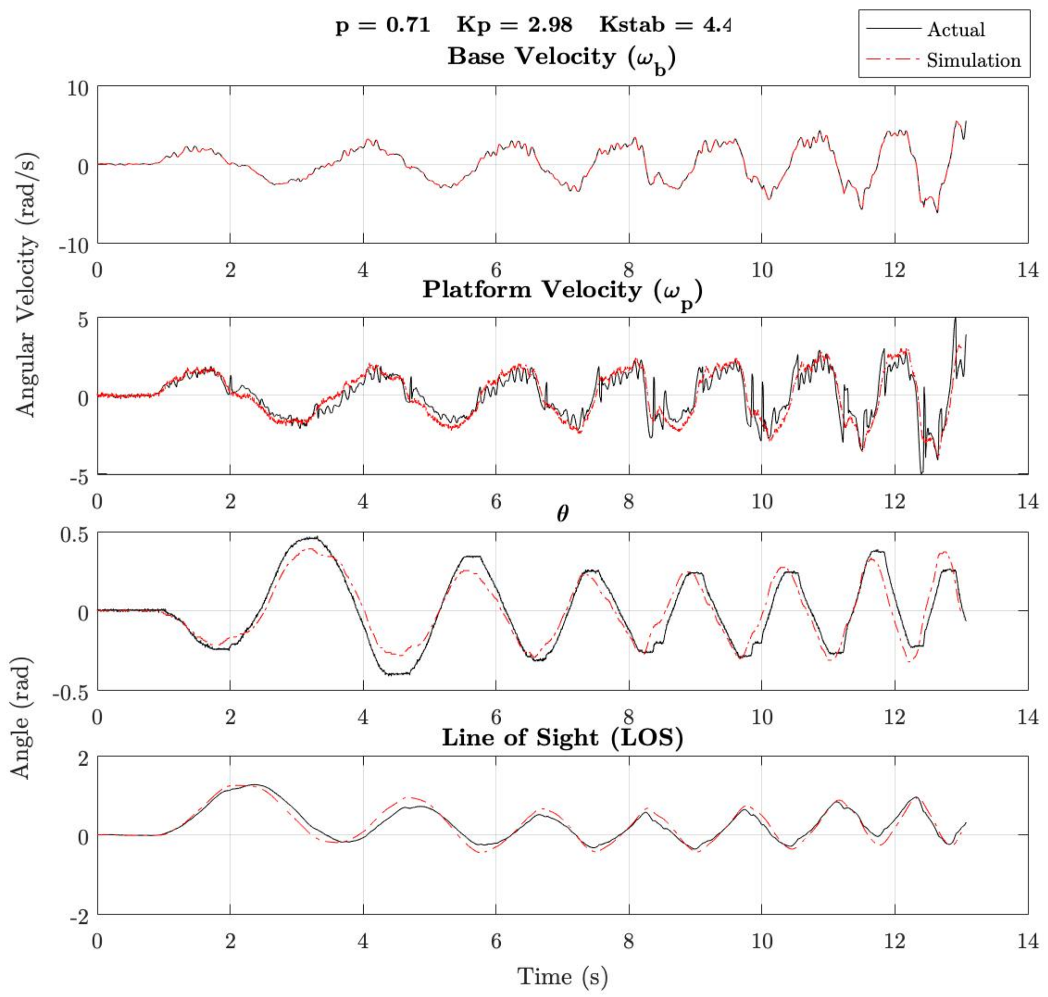


FIGURE 13.16: Comparison between simulation and actual results.

Chapter 14

Discussion

This chapter aims to analyse the results of the project and will thus frequently refer back to figures from in the previous chapter. For ease of analysis, the section headings of this chapter correspond to the same section headings in the previous chapter.

Note: The values of the controller constants (p, K_p, and K_{stab}) shall hereinafter be collectively referred to as the system 'parameters'.

14.1 Actual Results

Because all of the tests were conducted by hand, it was difficult to replicate base motion inputs. Consequently, this made it difficult to accurately analyse the effect that changing the system parameters had on the system. If the exact same motion could be applied to the base for successive tests, the response of the system at different parameter configurations could be plotted on the same set of axes and compared.

14.1.1 Step Tests

Test 1

Stabilisation:

A step of approximately $80°$ was applied to the base over approximately $0.5s$. The peak velocity of the base during this motion was $335°/s$ at a time of $2.76s$. The velocity of the platform at this point in time was $198°/s$ and relative angle was $-24.4°$. The base motion rejection ratio was therefore:

$$\frac{\omega_p}{\omega_b} = \frac{198}{335} = 0.591 = 59.1\%$$

According to the performance requirements stated at the beginning of the report, the rejection ratio should be below 10%.

Position:

The position loop in this test worked very well; the response was almost perfectly damped with an overshoot of less than 2.5%. However, with a settling time of approximately $7.5s$, the response may be considered too slow.

Note: The settling time of the positional response cannot be compared to the performance requirements due to the lack in performance of the stabilisation loop. Instead of having to rotate $80°$ to track the base, the platform only had to rotate $24.4°$

Test 2

For the second test, the parameters of the system were changed as follows:

- $p : 3.14s \rightarrow 1.57s$
- $K_p : 0.40 \rightarrow 1.6$
- $K_{stab} : 6.30 \rightarrow 4.03$

As shown by Figure 7.2 in Section 7.3, these changes should lower the stabilisation of the system and produce a more aggressive positional response.

Stabilisation:

A step of approximately 93° was applied to the base over approximately $0.51s$. The peak velocity of the base during this motion was $484^\circ/s$ at a time of $2.402s$. The velocity of the platform at this point in time was $361^\circ/s$ and the relative angle was -20.6°. The base motion rejection ratio was therefore:

$$\frac{\omega_p}{\omega_b} = \frac{361}{484} = 0.746 = 74.6\%$$

Position:

Once again, the positional response of the system was strong with a damping factor close to 1. The settling time of the system was approximately 3 seconds - a significant improvement from the first test.

Note: Again, the platform only had to rotate 20.6° *as opposed to* 93°. *Therefore the settling time is not a valid metric.*

14.1.2 Sinusoidal Tests

The sporadic nature of the base motion in these tests makes in difficult to analyse the performance of the system numerically - as can be done with a step test. These tests can, however, be helpful in illustrating how the system responds to base motion of differing frequencies. In theory, the motion of the platform should decrease as the frequency of the base motion increases. This phenomenon is shown by Figure 7.2 in Section 7.3.

Test 1

The base motion applied in Test 1 is better described as a combination of steps and sinusoidal motion. The frequency of the sinusoidal motion stays roughly constant throughout the test. The phenomenon described above is therefore not demonstrated by the test. This test is therefore not a useful one.

Test 2

The base motion in this test reveals more about the functioning of the system. The frequency of the base motion is initially low but increases as time goes on.

Graph 1 shows that the velocity of the platform and the base are essentially the same at low frequencies. As the base motion frequency increases, so too does the difference between the base motion velocity and the platform velocity.

Graph 2 shows that, as the frequency of the base motion increases, the variation in the LOS decreases.

Graph 3 shows that, as the frequency of the base motion increases, the peak relative angles between the base and the platform increase.

These three graphs essentially show the same concept using different metrics. This concept aligns with the phenomenon described above.

14.2 Comparison between Simulation and Actual

This section discusses the accuracy of the simulation. This accuracy is determined by calculating the *root mean square error*[1] (RMSE) and then normalising it so that RMSE values can be compared for different signals. The RMSE error is calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(S_{1,t} - S_{2,t})^2}{T}}$$

where T is the total number of data points and $S_{1,t}$ and $S_{2,t}$ are the two signals being compared (simulated and actual) at time t.

The RMSE is then normalised as follows:

$$NRMSE = \frac{RMSE}{S_{max} - S_{min}}$$

where S_{max} and S_{min} are the maximum and minimum values of the actual signal. The RMSE and NRMSE values for each of the tests is summarised by the table below[2]

TABLE 14.1: RMSE and NRMSE values for all tests.

	Variable	Step Tests		Sinusoidal Tests	
		Test 1	Test 2	Test 1	Test 2
RMSE	ω_p	$0.1433 rad/s$	$0.2253 rad/s$	$0.5595 rad/s$	$0.7007 rad/s$
	θ	$0.0511 rad$	$0.0389 rad$	$0.0505 rad$	$0.0703 rad$
NRMSE	ω_p	0.0389	0.0345	0.0727	0.0702
	θ	0.0730	0.0648	0.1243	0.0793

It is clear from this table that the normalised RMS errors for the sinusoidal tests are larger than that of the step tests. This is due to the increased motion of the sinusoidal tests.

It is also clear that the NRMSE for θ is higher than for ω_p. A possible reason for this is a slight inaccuracy in the scaling of the potentiometer signal which can come

[1] Also known as the *standard deviation of residuals* or *standard root mean square deviation*

[2] The MATLAB code written to generate these numbers can be found in Appendix **??**.

about as a result of the tolerances in the actual resistance of both the potentiometer and the $1k\Omega$ resistor to which it is connected.

The table shows that the maximum normalised error occurred in the first sinusoidal test where there was a deviation of 12.43% from the actual results. The minimum normalised error occurred during the second step test where there was a deviation of 3.45% from the actual results. Without the results from other similar studies, it is difficult to qualitatively analyse what these metrics mean. In other words, it is difficult to conclude whether or not these RMS errors are acceptable without knowing the typical RMS errors of similar control systems.

Chapter 15

Conclusions

Based on the above results and discussion, the following conclusions can be made:

15.1 Actual ISP Performance

According to the performance requirements stipulated in Section 1.2.2, with the exception of the damping factor requirement, the platform did not meet its specifications.

While the positional response of the system was good, the stabilisation loop did not perform well.

In other words, the base motion rejection of the platform was particularly poor. This is largely due to the fact that a geartrain had to be used. There are two reasons why geartrains should be avoided in all ISPs:

1. Without gears, the motor merely has to overcome the frictional forces present in the motor and gimbal frame in order to keep the platform inertially still. This requires very little torque. It is helpful to think of a gimbal with frictionless bearings; the object mounted to the gimbal would remain completely still despite rotations to the gimbal frame.

2. Consider a geartrain with a reduction ratio of n. If the base of the ISP is rotated 10°, the motor will have to rotate $n \times 10^\circ$ in the same period of time. This requires the motor to rotate n times faster than the base. Reaching this high speed takes time; thus reducing the rejection of base motion.

15.2 Simulation Accuracy

As stated in the previous chapter, it is difficult to comment on the accuracy of the simulation without access to comparisons between simulated and actual results for other base-tracking ISPs. One can conclude, however, that the accuracy of the simulation was sufficient to warrant controller tuning using the simulation as opposed to tuning by trial-and-error in the actual system.

Chapter 16

Recommendations

Based on the personal experience and insight gained throughout the duration of the project, the following recommendations are made:

The most crucial (and difficult) decisions to make when building an ISP is the selection of the motor. In order to get this decision right, one should spend a great deal of time determining the performance requirements/specifications of the system. This helps in deciding the whether the motor can provide enough torque, be driven at low speeds, and whether it can respond fast enough. Another recommendation with regards to the motor was discussed at length in the previous chapter - avoid using gears.
Once the motor has been selected, ensure that the motor **and its controller** are available in your region. For example, a more suitable motor was found for the project; this motor, however, was only available in the United States of America and its controller was not suitable for the project.

While the simulation is an incredibly important and useful component of control systems, it is important that one does not spend too much time tuning it. It is very easy to spend hours trying to tweak one component of the simulation in an attempt to improve the accuracy. The resulting improvements are more often than not negligible.

The following recommendations are made for future work on the project:

Once the ISP meets its specifications, there are multiple ways in which the device can be improved.

The first of which is to expand the ISP to stabilise all three axes. While this expansion would be mechanically complex and potentially expensive, the same theory that was applied to the single axis device can be applied to all three axes.

Additional features can be added to the device to increase functionality. A joystick that is used to manually aim the camera could be added; this would simply involve changing the positional command from zero to the joystick reading[1]. Another potential feature would be to implement a tracking loop. The input command to the loop could be obtained using an application on the smartphone to calculate the relative coordinates of the object to be tracked. Both of these features exist in commercially available smartphone ISPs.

[1]This reading would have to be converted to radians.

Appendix A

Budget and Variance Report

The following costs were incurred during this project:

A.1 3D Printing

The filament used for printing was quoted as $R700/kg$. The aforementioned *Cura* software produces a mass estimate of each part. This mass estimate was used to cost each part.

TABLE A.1: 3D printing budget

Part	Time Est (h:mm)	Mass Est (g)	QTY	Material Cost (R)
Potentiometer shaft connector	0:14	1	1	0,70
Handle	3:33	42	1	29,40
Phone mount - lower	1:07	9	1	6,30
Phone mount - upper	0:20	2	1	1,40
Gimbal arm	2:18	20	1	14,00
Potentiometer mounting plate	0:48	6	1	4,20
Gimbal frame	7:39	64	1	44,80
Motor shaft connector	0:15	1	1	0,70
Motor mounting plate	2:47	20	1	14,00
Potentiometer mounting plate (REV B)	0:47	6	1	4,20
Gimbal frame reprint	7:35	62	1	43,40
Motor shaft connector (REV B)	0:26	3	1	2,10
			Total	**165,20**

A.2 Electronics

The bulk of the components used for the circuit board were of negligible costs and were supplied by the university free of charge. The following components had to be purchased from *microrobotics* - an electronics shop in Stellenbosch.

TABLE A.2: Electronic components budget

Component	Unit Price (R)	QTY	Total Cost (R)
MPU6050	89,00	2	178,00
Slip ring - 6 channel	295,00	1	295,00
10k Potentiometer (4 pack)	25,00	1	25,00
		Subtotal	**498,00**
		VAT 15%	**74,70**
		Total	**572,70**

A.3 Additional Costs

The following components were borrowed for the duration of the project and will be return upon completion:

TABLE A.3: Borrowed items budget

		Unit		Total Cost
Component	**Owner**	**Price (R)**	**QTY**	**(R)**
Arduino Duemilanove	James Teversham	127,18	1	127,18
Faulhaber Motor 2232S024BX4 SC	A/Prof Mouton	3 282,00	1	3 282,00
Faulhaber gearhead 22/7 66:1 reduction	A/Prof Mouton	2 400,00	1	2 400,00
8067.00110 (Standard) Mounting Kit (G22m74)	A/Prof Mouton	361,00	1	361,00
			Subtotal	**6 170,18**
			VAT 15%	**906,45**
			Total	**7 076,63**

Note: The Arduino Duemilanove price includes VAT and was therefore not included in the VAT subtotal.

A.4 Total Cost

The grand total cost of the project is therefore:

$$Total = R7\,814.53$$

If the borrowed components are removed from this, the total becomes:

$$Total = R737.90$$

Appendix B

Arduino Code

The following code extensively uses a library compiled by *Jeff Rowberg*.

```
/* ============================================
Luke Leach
LCHLUK002
BSc Eng, MMT Eng (2018)
October 2018

I2Cdev device library code is placed under the MIT license
Copyright (c) 2011 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
===============================================
*/

#include "I2Cdev.h"
#include "MPU6050.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

MPU6050 accelgyro(0x69);        // Gyro1

//====================VARIABLES============================

//====================CONSTANTS============================
const int CMD = 0;              // Position command

const int LOWER_LIMIT = 4.45;  // Ensure rotation does not exceed physical limits
const int UPPER_LIMIT = 0.1;
const float potMidpointVoltage = 4.5454/2.0;
//=========================================================

//====================SENSOR READINGS======================
float potVal;    // Pot ADC output
float theta;     // Pot scaled (rad)
float gy;        // Gyro1 output - Y-axis
float Wy;        // Gyro1 scaled (rad/s)
float gy2;       // Gyro2 output - Y-axis
float Wy2;       // Gyro2 scaled (rad/s)
//=========================================================

//=====================PWM Setup===========================
const int A = 99;                                  // OCR2A
//=========================================================

//=====================PIN ALLOCATION=====================
int directionPin = 4;
int pwmPin = 3;
int potPin = A3;
//=========================================================

//====================SIGNALS==============================

// Position Loop
double error = 0;                     // Current Position error
double error_p = 0;                   // Previous Position error
double controller_out = 0;            // Current PI output
double controller_out_p = 0;          // Previous PI output

```

```
// Stab Loop
double Kp_out = 0;
double error2 = 0;

// Motor Input
int pwmIn;                                  // PWM sent to op amp
//===========================================================

//===================TIMING==================================
unsigned long oldTime;
unsigned long currentTime = 0;
unsigned long deltaT = 0.0015;
//===========================================================

//===================CONTROL PARAMETERS======================
float P = 1.57;                // PI controller time constant
float Kp = 1.2;                // PI controller gain
float Kstab = 4.2;             // Scaled P controller
float PWM_Offset = 4;
float Tsamp = deltaT;
float A_1 = Tsamp + 2.0*P;   //Z-domain equivalent
float B = Tsamp -2*P;
float C = 2.0;
float D = -2.0;
//===========================================================

//==========================OTHER============================
int direc;
double windupLim = (1.1*(A+1.0))/(Kp*Kstab); // Prevent
   integrator windup
//===========================================================

//========================FUNCTIONS==========================
void setPWM(int pwmVal);


void setup() {
    // join I2C bus (I2Cdev library doesn't do this
       automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000);
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // Configure pins
    pinMode(potPin, INPUT);
    pinMode(directionPin, OUTPUT);
    pinMode(pwmPin, OUTPUT);

    // Increase PWM frequency to 2.5kHz
    TCCR2A = _BV(COM2A0) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
    TCCR2B = _BV(WGM22) | _BV(CS22);
    OCR2A = A;
    OCR2B = 0;

    // initialize serial communication
    Serial.begin(115200);

    // initialize devices
    Serial.println("Initializing I2C device");
    accelgyro.initialize();
    accelgyro.setFullScaleGyroRange(MPU6050_GYRO_FS_1000); //FSR of 1000dps
    accelgyro.setDLPFMode(MPU6050_DLPF_BW_256); //DLPF bandwidth of 256Hz

    // verify connection
    Serial.println("Testing device connection:");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

    // change accel/gyro offset values
    accelgyro.setXGyroOffset(-21);
    accelgyro.setYGyroOffset(-12);
    accelgyro.setZGyroOffset(37);

    pwmIn = 0;
    setPWM(pwmIn);

    currentTime = micros();
}

void loop() {

      oldTime = currentTime;
      currentTime = micros();
      deltaT = currentTime - oldTime;

      potVal = analogRead(potPin)*0.0049; // V
      theta = -(potVal - potMidpointVoltage)*1.1903;  // rad

      gy = -accelgyro.getRotationY()/32.8;  // dps
      Wy = (gy/57.3);    // rad/s


      error = CMD - theta;

      controller_out = (A_1*error + B*error_p - D*controller_out_p)/C; //PI

      if(controller_out > windupLim){
        controller_out = windupLim;
      }
      else if(controller_out < -windupLim){
        controller_out = -windupLim;
      }

      Kp_out = controller_out*Kp;

      error2 = Kp_out - Wy;
      pwmIn = error2*Kstab;

      controller_out_p = controller_out; // Update values
      error_p = error;

      while(potVal >= UPPER_LIMIT){
        digitalWrite(directionPin, HIGH);
        potVal = analogRead(potPin);
        pwmIn = 10;
        setPWM(pwmIn);
        direc = 1;
      }
      while(potVal <= LOWER_LIMIT){
        digitalWrite(directionPin, LOW);
        potVal = analogRead(potPin);
        pwmIn = 10;
        setPWM(pwmIn);
        direc = -1;
      }

        if(pwmIn >= 0)
        {
          digitalWrite(directionPin, HIGH);
          direc = 1;
          if(pwmIn > (A - PWM_Offset)){
            pwmIn = A - PWM_Offset;
          }
          setPWM(pwmIn + PWM_Offset);
        }

        else if(pwmIn < 0)
        {
          digitalWrite(directionPin, LOW);
          direc = -1;
          pwmIn *= -1;
          if(pwmIn > (A - PWM_Offset)){
            pwmIn = A - PWM_Offset;
          }
          setPWM(pwmIn + PWM_Offset);
        }
```

```
}

void setPWM(int pwmVal){
  OCR2B = pwmVal;
}
```

Appendix C

MATLAB Graph Generation Code

The following code was used to generate all the graphs used in the report.

```
%------------------------------READ DATA IN------------------------------
%Step Tests------------------------
%Test 1----------------------------
data=dlmread('StepTest006.txt');

%Test 2----------------------------
%data=dlmread('StepTest004.txt');

%Sinusoidal Tests------------------
%Test 1----------------------------
%data=dlmread('Test005a.txt');

%Test 2----------------------------
%data=dlmread('Final002.txt');
%--------------------------------------------------------------------------------


%-------------------------ASSIGN DATA TO VARIABLES-------------------------

P = data(1,1)
Kp = data(2,1)
Kstab = data(3,1)
Wp = data(8:end,1);
BM = data(8:end,2);
Theta = data(8:end,3);
DeltaT = data(8:end,4);
%--------------------------------------------------------------------------------


%-------------------------GENERATE TIME VECTOR-------------------------------

n=size(Wp);
time = 1:n;
t3 = transpose(time*mean(DeltaT)*1.0e-6);
```

```
%-----------------------------------------------------------------------------------------------

%--------------------DEFINE PARAMETERS--------------------
Km = 31.4e-3;
Kv = Km;
eff = 0.7;
J = 2.095e-3;
Ngear = 66;
Kfr = 0.001;
Lm = 270e-6;
Rm = 54.86;
Wg = 2*pi*1000;
Zg = 0.707;
Wf = 2*pi*256
%-----------------------------------------------------------------------------------------------

%--------------------SETUP INTEGRATION TF--------------------
s = tf('s');
sys = 1/s;
%-----------------------------------------------------------------------------------------------

%--------------------INTEGRATE SIGNALS--------------------
clf
LOS = lsim(sys, Wp, t3);
BaseAngle = lsim(sys, BM, t3);
%-----------------------------------------------------------------------------------------------

%--------------------GENERATE SIM BM SIGNAL--------------------
sig1 = [t3, BM];
%-----------------------------------------------------------------------------------------------

%--------------------GENERATE PLOTS--------------------
figure(1);
d = subplot(4,1,1);
plot(d, t3, BM, 'k', BM_sim.time, BM_sim.signals.values, 'r-.
    ');
```

```
title(d, 'Base Velocity (BM)');
ylabel(d, 'Angular Velocity (rad/s)');
grid('on');

a = subplot(4,1,2);
plot(a, t3, Wp, 'k', Wp_sim.time, Wp_sim.signals.values, 'r-.
    ');
title(a, 'Platform Velocity (Wp)')
ylabel(a, 'Angular velocity (rad/s)');
grid('on');

b = subplot(4,1,3);
plot(b, t3, Theta, 'k', Theta_sim.time, Theta_sim.signals.
    values, 'r-.');
title(b, 'Theta');
ylabel(b, 'Angle (rad)');
grid('on');

c = subplot(4,1,4);
plot(c, t3, LOS, 'k', LOS_sim.time, LOS_sim.signals.values, '
    r-.');
title(c, 'Line of Sight (LOS)');
ylabel(c, 'Angle (rad)');
grid('on');
xlabel('Time (s)');

suptitle({'Actual Response vs Simulated Response',
     sprintf('p = %0.2f \t Kp = %0.2f \t Kstab = %0.2f', P, Kp
        , Kstab)});
hl = legend({'Actual', 'Simulation'});
newPosition = [0.915 0.02 0.065 0.065];
newUnits = 'normalized';
set(hl,'Position', newPosition,'Units', newUnits);

figure(2);
plot(t3, BaseAngle, 'b', t3, LOS, 'g');
legend({'Base Angle', 'LOS'});
xlabel('Time (s)');
ylabel('Angle (rad)');
title(sprintf('p = %0.2f, K_p = %0.2f, K_{stab} = %0.2f', P,
    Kp, Kstab));

figure(3);
plot(t3, BM, 'b', t3, Wp,'g');
legend({'Base Velocity', 'Platform Velcoity'});
xlabel('Time (s)');
ylabel('Angular velocity (rad/s)');
title(sprintf('p = %0.2f, K_p = %0.2f, K_{stab} = %0.2f', P,
    Kp, Kstab));

figure(4);
```

```
plot(t3, BaseAngle, 'b', t3, Theta,'g');
legend({'Base Angle', 'Theta'});
xlabel('Time (s)');
ylabel('Angle (rad)');
title(sprintf('p = %0.2f, K_p = %0.2f, K_{stab} = %0.2f', P,
    Kp, Kstab));
%
```

Appendix D

MATLAB Root Mean Square Calculation Code

The following code was used to calculate the RMSE and NRMSE values referenced in Section 14.2.

```
Wp_sim1 = zeros(length(Wp), 1);
error = zeros(length(Wp),1);
error_sq = zeros(length(Wp),1);
sum = 0;
for i = 1:length(Wp)
    ind = find(abs(Wp_sim.time - t3(i)) < 0.0005);
    Wp_sim1(i) = Wp_sim.signals.values(ind(1));
    error(i) = Wp_sim1(i) - Wp(i);
    error_sq(i) = error(i)^2;
    sum = sum + error_sq(i);
end
plot(t3, error)
RMSE = sqrt(sum/length(Wp))
NRMSE = RMSE/(max(Wp) - min(Wp))

%%
Theta_sim1 = zeros(length(Theta), 1);
error = zeros(length(Theta),1);
error_sq = zeros(length(Theta),1);
sum = 0;
for i = 1:4635
    ind = find(abs(Theta_sim.time - t3(i)) < 0.0005);
    Theta_sim1(i) = Theta_sim.signals.values(ind(1));
    error(i) = Theta_sim1(i) - Theta(i);
    error_sq(i) = error(i)^2;
    sum = sum + error_sq(i);
end
plot(t3, error)
RMSE = sqrt(sum/length(Theta))
NRMSE = RMSE/(max(Theta) - min(Theta))
```

Bibliography

[1] J. Hilkert, "Inertially stabilized platform technology concepts and principles", IEEE Control Systems, vol. 28, no. 1, pp. 26–46, 2008.

[2] M. K. Masten, "Inertially stabilized platforms for optical imaging systems", IEEE Control Systems, vol. 28, no. 1, pp. 47–64, 2008.

[3] F. K. Mueller, "A history of inertial guidance", VITRO CORP OF AMERICA NEW YORK, Tech. Rep., 1963.

[4] R. Garner, Hubble space telescope pointing control system, 2017. [Online]. Available: `https://www.nasa.gov/content/goddard/hubble-space-telescope-pointing-control-system`.

[5] P. Yedamale, "Brushless dc (bldc) motor fundamentals", Microchip Technology Inc, vol. 20, pp. 3–15, 2003.

[6] H. Mouton, Mec4053z - measurement and control, chapter 3: Actuators, 2018.

[7] ——, Mec4107s - fundamentals of control systems, 2018.

[8] Analysis and design of feedback control systems - understanding poles and zeors. [Online]. Available: `http://web.mit.edu/2.14/www/Handouts/PoleZero.pdf`.

[9] J. R. Mevey, "Sensorless field oriented control of brushless permanent magnet synchronous motors", 2009.

[10] Mpu-6000 and mpu-6050 product specification revision 3.4, 2013. [Online]. Available: `www.invensense.com`.

[11] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing", IEEE Transactions on Biomedical Engineering, vol. 53, no. 7, pp. 1346–1356, 2006.

[12] K. Shirriff and P. Badger, Secrets of arduino pwm. [Online]. Available: `https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM`.