

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Design and Implementation of Self-Balancing Robot Using Advanced LQR Controller

AUTHOR	Antony Fawzi Tawfik Farid (20400049)
SUPERVISOR	Dr. David Dewar
MODERATOR	Dr. Rishad Ahmed
DATE	September 2022

Project thesis submitted in part fulfilment of the requirements for the degree of Master of Science **in Electrical Engineering for Sustainable and Renewable Energy**, The University of Nottingham.

Abstract

The objective of this project is to design, simulate and implement a two-wheeled self-balancing robot that is controlled using a Linear Quadratic Regulator (LQR) controller. The first section of the project is the literature review which will focus on control systems in robotics. This section will analyse the different control strategies traditionally used in robotics and the most up to date techniques of tuning control systems including adaptive and iterative methods. Additionally, the literature review will outline the gaps in the research and how they can be improved. The following section is the mathematical model of the system. Using LaGrangian mechanics the physics of the system is translated to non-linear mathematical equations that model the system. The non-linear equations are then linearised and used to obtain the state-space matrix of the system. From the state-space of the system, the LQR controller is designed, and the weighting of the Q and R matrices is decided. The system is then simulated and tested to view the effects of varying the weighting the matrices. Additionally, the system is tested to find the response of the controller when external disturbances are applied to the robot. This is then compared to a PID controller self-balancing robot. The implementation strategy is then discussed which outlines the system design. Moreover, the reasoning behind component selection is outlined, and a design of the circuitry and model of the robot are provided.

Table of Contents

Abstract.....	1
Chapter 1: Introduction	3
1.1 Background	3
1.2 Aims and objectives	3
1.3 Project plan.....	4
Chapter 2: Literature Review	6
Chapter 3: Mathematical Modelling	14
3.1 Description of the System	14
3.2 Non-linear mechanical model.....	17
3.3 Linearisation	21
3.4 State-Space Model	22
3.6 Modified state-space for voltage output	24
Chapter 4: LQR Controller Design and Simulation	27
4.1 MATLAB simulation of the system (Weighting Matrices)	27
4.2 Simulink simulation (Robustness Testing).....	35
Chapter 5: Implementation.....	38
5.1 System Design	39
5.2 Description of components	40
5.3 DC/DC converters design.....	42
5.4 Design of the chassis	46
5.5 Cost.....	46
Chapter 6: Discussion of Results.....	48
Chapter 7: Future work.....	50
Conclusion	51
References.....	52

Chapter 1: Introduction

1.1 Background

The focus of this paper is to design and implement a two-wheeled self-balancing robot (SBR) using a Linear Quadratic Regulator (LQR) controller. Numerous research has been done on different control theories using self-balancing robots, this is because the unstable non-linear dynamics of the inverted pendulum provide a good complex platform to test different control theories. The past decade has seen a rapid increase and development of robotics as they enter the more mainstream consumer market. According to Juang [1], consumer robotics can be classified into two categories, the first being mechanically stable robot such as automated vacuum cleaners and the second being mechanically unstable robots such as the Segway. The popularization of the “Two wheeled self-balanced transportation robot” (Segway) coupled with the potential of increased applications has inspired a plethora of research in the control and robotics fields [2].

1.2 Aims and objectives

Aims:

1. Design and Simulate LQR controller
2. Implementation of the Robot.

Objectives:

1. Literature review, to learn how to design LQR controller.
2. Familiarization with ESP 32.
3. Purchasing gyroscopes, motors, encoders and temperature sensors.
4. Creating layout of the devices on the Veroboard.
5. Use C to write the code for the ESP 32 microcontroller.
6. Implement a functional robot.
7. Test the robot on different terrain.

8. Optimise the code to improve the stability of the controller and the response time.

1.3 Project plan

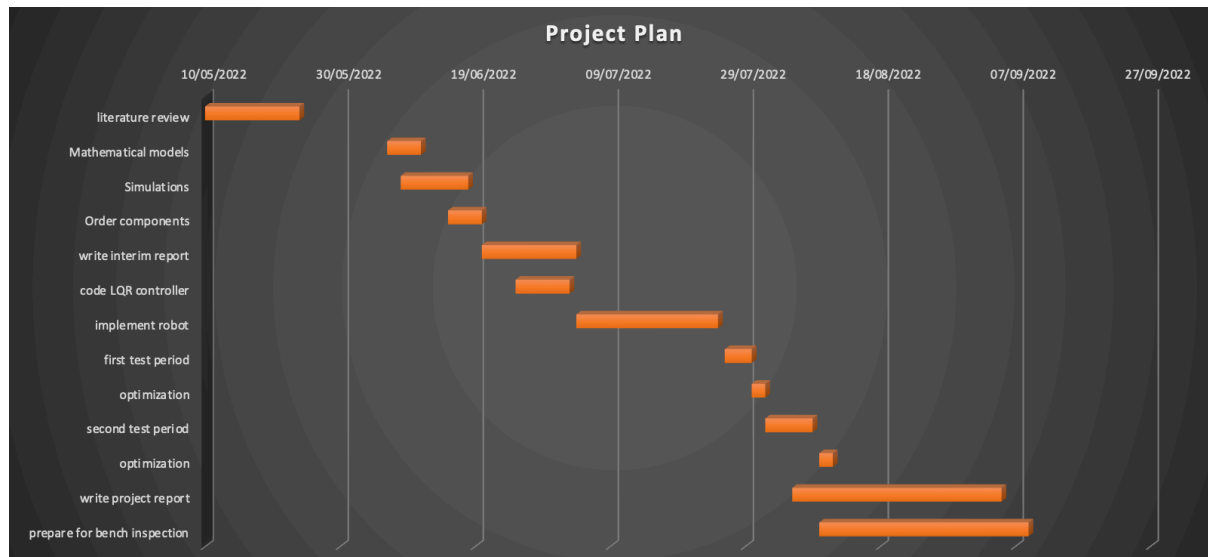


Figure 1.1 Gantt chart

Figure 1.1 shows a Gantt chart which outlines the project plan. The first task in the plan is to carry out a comprehensive literature review. The purpose of this task is to identify and analyse the classical and current techniques of designing control systems. Control strategies such as PID, fuzzy logic, neural network, Model Predictive controls (MPC) and other configurations of cascaded controls are studied based on the research done. From the literature review, a better understanding of control systems and objectives of the project is attained based on the gaps in the research, the duration of this task is two weeks.

The second task is the mathematical model, here, the LaGrange equations are used to produce a non-linear mathematical model of the system, which is then linearised. From the linearised equations, the state-space model of the system is developed, the duration of this task is five days.

Task three is the simulation. The simulation section focuses on obtaining the LQR gain matrix from MATLAB. Additionally, the system will be tested for the effect of tuning the weighting matrices and the effect of external disturbances on the response time of the system.

the duration of this task is ten days. This is then followed by ordering the components required to implement the self-balancing robot and working on the interim presentation.

The next task is to code the ESP32 microcontroller, by interfacing the components such as the gyroscope, motors, and DC-DC converters. Additionally, the LQR controller designed previously is coded onto the microcontroller. The duration of this task is eight days. The following task is to build the whole robot, which was planned to take three weeks.

The final tasks consist of two test and optimization periods, before writing the project report and working on the bench inspection.

Chapter 2: Literature Review

This literature review section will analyse the previous research that has been done on control theories which were tested using SBR, it will also provide the gaps in the research which could be developed further. The mathematical modelling of the robot design carried out in the papers which will be discussed relies on the use of Newtonian mechanics, which use forces in the system to produce the mechanical equations which are then used to find the state equations. Alternatively, LaGrangian mechanics can be utilised to mathematically model the system by using the kinetic energy and potential energy of the system to find the LaGrange equation. From this equation the non-linear mechanical equations of the system are found, the number of which depends on the Degrees of Freedom (DOF) of the system, finally using linearisation techniques such as Taylor sequencing, the mechanical model is linearised to produce the state equations, the process of using LaGrange mechanics to model the system will be outlined in detail in chapter 2. However, the papers do differ in the control theory being researched, the testing of the system, and the implementation of the design.

In 2019, Jayakody and Sucharitharathna published a research paper in which they discussed the design of a PID control unit for an SBR [3]. This paper focuses on comparing PID, PD and P controllers based on the overshoot, settling time and steady state error to determine which provides the best response for a given applications. While this research paper provides a foundational understanding to how the controller of an SBR is designed, it does not provide a mathematical method of tuning the PID controller. Additionally, it does not provide a strategy for testing the robot.

In 2015, another research paper on PID controllers of SBR compared two strategies. A system that cascades the kinematic model with a PI controller and another that uses two cascaded PID controllers [4]. In this paper the robustness of the controller is tested using four experiments. The first is balancing the robot while stationary which both designs passed the

test, and their performances were similar. The second test was rotating the robot about the vertical axis, which again both designs were capable of. The third test was accelerating forward and backward to determine which system can handle the highest acceleration, both performed similarly. Finally, the research carried out an obstacle test in which both systems were able to clear the same obstacles. The paper provides a well thought out testing procedure to determine which control strategy is best. The paper shows that both controllers are suitable for the implementation of the robot, as both proved to be robust controllers. However, the research does not show how the optimal PI and PID settings can be found. Additionally, the paper does not provide reasoning behind the choice of PID controllers over other strategies.

Philip and Golluri in 2020 [5], expanded on the research of cascaded PID controls [4] by combining them with a trajectory tracking controller. To design the PID controls, Philip and Golluri implement separate PID to control each DOF of the system which. Three PIDs were designed to control the translational motion of the robot, the yaw rotation about the z-axis and the tilt of robot with respect to the xy-plane. To implement the trajectory control, the researchers use the position, velocity and heading of the robot, combined with the desired trajectory. This system would allow for making an autonomous robot in which the user must determine the trajectory of motion before operation. To test the system, they simulated each PID to determine that the system will stabilise. The research has successfully shown that a combination of PID and trajectory controllers can be used to design an autonomous system with a pre-determined trajectory. However, the results have shown high levels of noise in the system response which could be attributed to the use of multiple cascaded controllers, to combat the issue of the noise, filters outlined in [3] could be implemented, in which the gyroscope was connected to a low-pass filter and the accelerometer was connected to a high-pass filter. While this research outlines a different application of SBR, it does not test the

balancing aspect of the control system independent of the trajectory tracking. Since the system has knowledge of its trajectory which cannot be altered in real time, the robustness of the controller is in question.

There are multiple methods of designing a PID controller which are based on the placement of poles in desired locations. One of the most popular methods of tuning a PID controller is the Zeigler-Nichols method which is discussed in [6]. This paper researches the method of estimating the PID gains of the controller for the speed control of a DC motor. The research aimed to minimize the rise time, overshoot, and settling time. To do so, the electrical model of the motor was analysed to find the transfer function of the system and the Zeigler-Nichols tuning method was used to calculate the PID parameters. The results of the research showed that the method was successful in achieving the speed control of a DC motor. This method is best for first order systems which can be easily modelled in order to find the transfer function of the system.

Another method of tuning PID, which is more suitable for higher order systems, is the Domain-Decomposition (D-Decomposition) method which was used in [7] to control a two-mass system. The research outlines that this method requires the calculation of the boundaries of the stable regions from the characteristics equation, the number of which is equal to the order of the system. After dividing the complex region into smaller finite portions based on the boundaries, the problem is solved using an iterative method. While the research shows that this method is successful in estimating the gains of PID controller for a higher order system, it is very mathematically challenging to calculate. Additionally, both methods in [6] and [7] are designed based on the transfer function of the system and placing the poles in the within the stable boundaries of the complex region. Thus, they do not allow for a change in parameters of the system under control. For a balancing application it is crucial that the controller can handle changes in the properties of the robot, such a shift in the Center of

Gravity (COG). A more mathematically intuitive control method, which is also robust under changes in the parameters is required.

From the literature review done so far, it is apparent that PID controllers are capable of being applied to SBR. However, the controller cannot be intuitively manipulated to favour certain operational characteristics of the robot such as the performance and effort. Additionally, tuning the PID becomes very mathematically complex for higher order systems containing multiple actuators, which led various researchers to cascade PIDs such that each controller is responsible for only one DOF, when implementing this sort of design higher computational capabilities are required from the microcontroller being used. Furthermore, to ensure that the controller will be capable of achieving the desired output, the measured parameters from the sensors need to be passed through filters to remove the noise. Finally, since the PID relies on accurate transfer functions and operation within a region of stability, it is not robust when changes to the properties of the system occur.

In an attempt to tackle the robustness issue of the PID controller, [8] uses a two-level adaptive control. The ideology of this strategy is to vary the control parameters of the system based on pre-determined set points. In this paper the set points were the PWM signal fed to the converters which supply the voltage to the motors and the tilt angle. This was done by starting off the motors with a high tuning parameter, then once stability was achieved, the parameters were then redefined using the adaptive control levels. When compared to PD controller, this system showed lower overshoots, lower oscillations, and was more robust against disturbances. Due to the robust nature of intelligent adaptive controllers, it is worthwhile to research more adaptive based control strategies.

Fuzzy logic is a form of adaptive control which was studied for an SBR application in [9].

The aim of this research paper is to design and analyse both a pole placement controller and a fuzzy logic controller to work together. The paper outlines the four steps of implementing a

fuzzy logic controller, the first step being the fuzzification, in which the input values are converted into a degree of membership of fuzzy sets. Then, the fuzzy rules step outlines rules that the system will follow using if/else statements. The third step is the fuzzy inference in which the final fuzzy conclusion is found by subjecting the degree of membership values to the fuzzy rules. Finally, the defuzzification step converts the fuzzy conclusion to an output that can be used by the system. The fuzzy logic controller is multiplied with the PID controller and is then fed into the system. Simulations were carried out to compare PD and Fuzzy PID controllers. The results showed that the system response and robustness of the fuzzy control were superior to that of the PID alone. While this method does tackle the robustness issue of the PID, the issues of tuning the pole placement system and concerns of stability are still prevalent.

Another example of fuzzy logic is provided by [10], where a PD controller is used for position tracking and a fuzzy logic controller is used for regulating the balance of the robot by controlling the pitch angle. Additionally, the researchers carried out test of the system that also included an adaptation controller, which would take in the error and the error rate of the system to provide a scaling factor to the fuzzy logic controller unit. When tested for robustness by applying a disturbance, the systems that included the adaptation unit performed better than the one without, with a 25% improvement in the regulation time.

The research done on the application of fuzzy logic to PID shows that this control theory helps with the response of the system to disturbances, as fuzzy logic provides an adaptive approach to the parameter setting process. This allows for external changes to the properties of the controller while maintaining stability. However, since the PID controls are still used, the inherent issues of tuning and ensuring operation within the stability region still exist.

There are multiple ways to assess the stability of a system, one of which is to use Lyapunov's direct method of stability [11]. Research done in [12] states that LQR controllers are designed

to achieved Lyapunov stability, guaranteeing system stability. This means that when using an LQR controller one does not need to be concerned about operating within the stability region, because the LQR controller fulfills Lyapunov's stability criteria. This provides a solution to the stability concerns when using a PID controller.

In 2016 a group of researchers set out to study the combination of LQR and PID controller when applied to an Unmanned Ariel Vehicle (UAV) [13]. The objective of combining the two controller strategies is to benefit from the advantages of both. The LQR controller is very computationally efficient and can provide stable, fast, and robust performance will maintaining a low energy consumption. Additionally, LQR controllers are easier to implement for multiple actuator systems [14]. Meanwhile, [13] outline that PID controls are superior in tuning the gain matrix for optimized transient behaviour. To benefit from the mentioned advantages of both controls, the LQR controller is used in an inner loop to regulate the unstable system, the output of this controller is guaranteed stability based on the proof done in [12]. The PID controller is added in a second/outer loop of the system, which uses the parameters stabilized by the LQR to provide a specific transient response and reference tracking. The results of the simulations showed that the system was indeed capable of achieving stability of the UAV and tracking of motion parameters. To test the robustness of the design, the researchers varied the mass of the plane and found that the controller was capable of handling the changes. While this paper shows that the LQR's guarantee of stability is highly beneficial when controlling a naturally unstable system, it does not provide information as to how the tuning of the LQR controller can be varied to manipulate the performance parameters of the system.

As the benefits of LQR control are discussed more, in terms of system stability, robustness, and computational efficiency, more research has been done to analyse the applications of this controller in the field of robotics. In 2018, Okyere et al. set out to study the application of

LQR control to a quad-rotor helicopter [14]. The focus of this paper is to analyse the response of the LQR controller for different tuning parameters. The performance index of the controller can be found by solving a cost function which uses two weight matrices, Q and R . The Q matrix is a symmetric semi-absolute positive matrix which is responsible for the performance of the system i.e., time required to reach stability, while the R a symmetric absolute positive matrix which is responsible for the effort of the system i.e., energy consumed by the system [15]. In [14], Okyere et al. observed that when the R matrix was increased the effort of the system was prioritized more than the performance, thus the system state variables took a longer time to approach zero. When the Q matrix was increased, the performance of the system was prioritized more than the effort, the outcome was a faster system response. [14] shows that even for a four-actuator model, the LQR control can be easily tuned to deliver on the desired system characteristics by choosing how much the control designer wants to prioritise effort and performance. However, for the majority of applications, one cannot completely prioritise effort over performance or vice-versa, usually controllers are designed to provide a balance of both characteristics.

There are multiple mathematical approaches to find the weighting of the Q and R matrices, one of which is outlined in [15]. In this paper the researchers study a heuristic search algorithm called Bees Algorithm (BA) which relies on learning and remembering to determine the weighting matrices of an LQR controller applied to a flexible robotic arm experiencing disturbances. The testing compared the response of a traditionally designed LQR, without the BA algorithm and found that it was capable of stabilizing the system under external physical disturbances and when noise was added to the sensors. Then the researchers carried out the same test with the addition of the BA algorithm to determine the weighting matrices and found that the response time of the system was four times faster.

Another method of determining Q and R is the particle swarm optimization (PSO) which was applied to a double inverted pendulum system on a cart in [16]. PSO is an iterative approach to finding the optimal controller tuning, it does so by creating an initial state which in the case of [16] was an initial value for the position and velocity of the robot, it then uses a fitness equation to calculate the best position and velocity of the system which is fed back into the system. this process is repeated depending on the pre-determined number of iterations to find the optimal Q and R matrix. The results of this research showed that the PSO was highly effective in determining the tuning parameters of the LQR controller.

Chapter 3: Mathematical Modelling

3.1 Description of the System

To mathematically model the system, this paper will utilize the LaGrangian mechanics method. This method uses the kinetic energy and the potential energy to produce the LaGrange equation. This equation is then applied to the Euler-Lagrange equation to produce the non-linear mechanical equations of the system, the number of mechanical equations is dependent on the Degrees of Freedom (DOF) of the system. Finally, these equations are linearised and used to develop the state-space matrix.

Figure 3.1 below shows a 3D illustration of the robot, the structural parameters of the robot that are used for the modelling of the system are outlined below in table 3.1. The parameters that determine the motion of the robot are defined in table 3.2. Additionally, the motor parameters such as the torque constant and the back emf constant are included as they will be used to produce an output from the state-space model in terms of voltage.

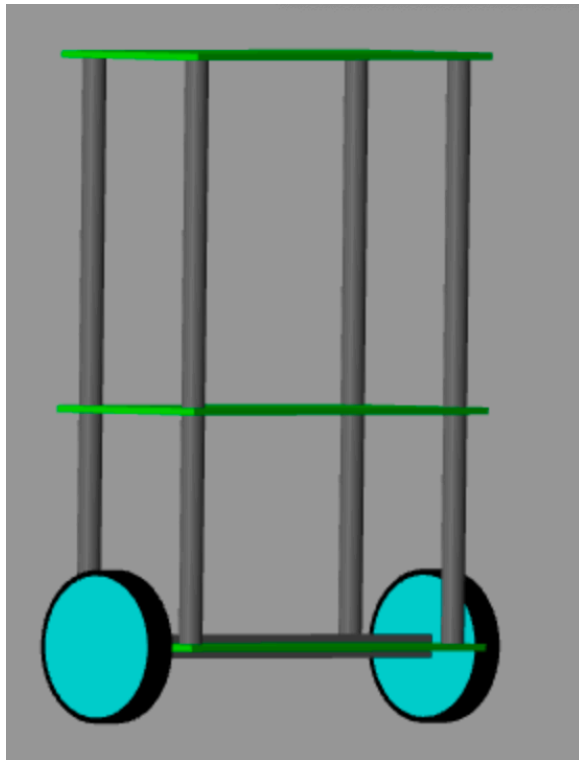


Figure 3.1 3D robot model

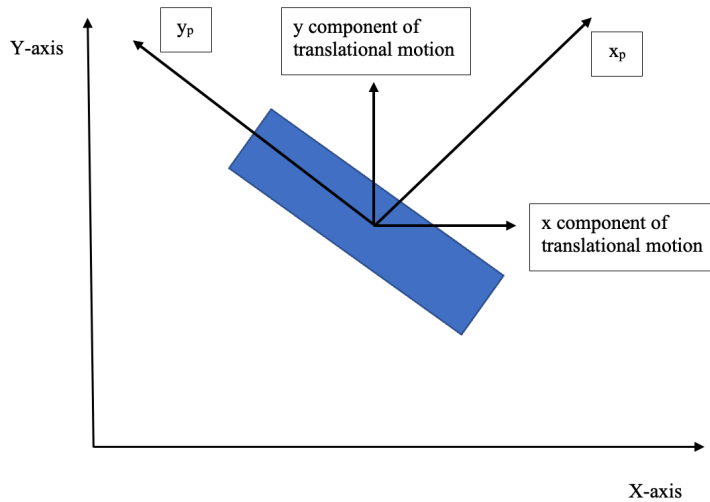


Figure 3.2 Defining motion of the robot

The motion of the robot can be broken down into three Degrees of Freedom (DOF), the first being the translational motion of the robot. The second component is the tilt angle of the robot with respect to the z-axis, this parameter is denoted by θ . The third component is the yaw rotation of the robot, which is measured with respect to the y-axis, denoted by δ .

Figure 3.2 shows the robot viewed from above, on the xy-plane. The x component of the robot is denoted by x_c , and the y-component is denoted by y_c .

Table 3.1 Definition of parameters (1)

Parameter	Symbol
Center of Gravity (COG)	l
Distance between the wheels	d
Radius of the wheels	r
Mass of the robot (Chassis, rods, and shelves)	M
Mass of Wheels	m
Torque constant of motor	K_t
Back emf constant of the motor	K_e
Internal resistance of motor	R_m

Parameters:*Table 3.2 Definition of parameters (2)*

Parameter	Symbol	Value	Units
Mass of body	M	1.5	Kg
Mass of wheels	m	0.046	Kg
Viscous friction coefficient	b	0.125	N/m/s
Gravitational acceleration	g	9.81	m/s ²
Position of COG	l	0.15	m
Radius of wheel	r	0.05	m
Distance between wheels	d	0.3	m
Axial MOI of wheels	I _a	$I_r/2 = 0.000038$	Kg.m ²
Radial MOI of wheels	I _r	$(0.023*0.05^2)/3 = 0.000019$	Kg.m ²
y-axis MOI of body	I _y	$md^2 = 0.00414$	Kg.m ²
z-axis MOI of body	I _z	$MI^2 = 0.0234$	Kg.m ²
Torque constant of motor	k _t	0.658 [21]	Nm/A
Back emf constant of motor	k _e	1 [21]	Krpm/V
Resistance of motor	R	1 [21]	Ω

3.2 Non-linear mechanical model

The translation kinetic energy is made up of four components. The translational kinetic energy of the chassis, denoted by T_c . The translational kinetic energy of the wheels, denoted by T_w . The rotational kinetic energy of the chassis, denoted by T_{rc} . The rotational kinetic energy of the wheels, denoted by T_{rw} .

To calculate the velocity in the x_p direction:

$$\dot{x}_p = \dot{x}_c \cos \delta + \dot{y}_c \sin \delta \quad (3.1)$$

To calculate the velocity in the y_p direction:

$$\dot{y}_p = -\dot{x}_c \sin \delta + \dot{y}_c \cos \delta \quad (3.2)$$

Since the robot cannot move in the y_p direction then y_p is zero. Additionally, the forward velocity is equal to \dot{x}_p .

$$\dot{x}_p = \dot{x}_c \cos \delta + \dot{y}_c \sin \delta = \dot{x} \quad (3.3)$$

$$\dot{y}_p = -\dot{x}_c \sin \delta + \dot{y}_c \cos \delta = 0 \quad (3.4)$$

The position of the COG along the height of the pendulum is denoted. By “ l ”, and is measured with respect to the z-axis.

The position of the robot with respect to the x-axis, y-axis and the z-axis is given by the following matrix.

$$P = \begin{bmatrix} x_c + l \sin \theta \cos \delta \\ y_c + l \sin \theta \sin \delta \\ l \cos \theta \end{bmatrix} \quad (3.5)$$

The derivative of the position matrix provides the velocity of the robot.

$$v = \begin{bmatrix} \dot{x}_c + l \cos \theta \cos \delta \dot{\theta} - l \sin \theta \sin \delta \dot{\delta} \\ \dot{y}_c + l \sin \theta \cos \delta \dot{\theta} + l \cos \theta \sin \delta \dot{\delta} \\ -l \sin \theta \dot{\theta} \end{bmatrix} \quad (3.6)$$

From the velocity equations (3.6), the translational kinetic energy of the chassis is calculated as shown in (3.7).

$$T_c = \frac{1}{2} M \begin{bmatrix} \dot{x}_c + l \cos \theta \cos \delta \dot{\theta} - l \sin \theta \sin \delta \dot{\delta} \\ \dot{y}_c + l \sin \theta \cos \delta \dot{\theta} + l \cos \theta \sin \delta \dot{\delta} \\ -l \sin \theta \dot{\theta} \end{bmatrix}^T \cdot \begin{bmatrix} \dot{x}_c + l \cos \theta \cos \delta \dot{\theta} - l \sin \theta \sin \delta \dot{\delta} \\ \dot{y}_c + l \sin \theta \cos \delta \dot{\theta} + l \cos \theta \sin \delta \dot{\delta} \\ -l \sin \theta \dot{\theta} \end{bmatrix} \quad (3.7)$$

Equation (3.8) shows the translation kinetic energy of the wheels where d is the distance between the wheels, and θ_R and θ_L represents the angles of the right and left wheel respectively.

$$\theta_R = x + d\delta \quad (3.8)$$

$$\theta_L = x - d\delta \quad (3.9)$$

$$T_w = \frac{1}{2} m r^2 (\dot{\theta}_R^2 + \dot{\theta}_L^2) \quad (3.10)$$

The angular velocity vector of the pendulum is given by equation (3.9)

$$\alpha = \begin{bmatrix} -\sin \theta \dot{\delta} \\ \dot{\theta} \\ \cos \theta \dot{\delta} \end{bmatrix} \quad (3.11)$$

To calculate the rotational kinetic energy of the chassis, the moment of inertia is defined as shown below, where I_x , I_y , and I_z are used to denote the MOI with respect to the x, y, and z-axis respectively.

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (3.12)$$

The equation below is used to calculate the rotational kinetic energy of the chassis.

$$T_{rc} = \frac{1}{2} \begin{bmatrix} -\sin \theta \dot{\delta} \\ \dot{\theta} \\ \cos \theta \dot{\delta} \end{bmatrix}^T \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} -\sin \theta \dot{\delta} \\ \dot{\theta} \\ \cos \theta \dot{\delta} \end{bmatrix} \quad (3.13)$$

To calculate the rotational kinetic energy of the wheels, the angular velocity of each wheel is calculated as shown below.

$$\alpha_R = \begin{bmatrix} 0 \\ \dot{\theta}_R \\ \dot{\delta} \end{bmatrix} \quad (3.14)$$

$$\alpha_L = \begin{bmatrix} 0 \\ \dot{\theta}_L \\ \dot{\delta} \end{bmatrix} \quad (3.15)$$

The moment of inertia of the wheels is shown below, where I_r is the radial MOI of the wheel and I_a is the axial MOI of the wheel.

$$I_w = \begin{bmatrix} I_r & 0 & 0 \\ 0 & I_a & 0 \\ 0 & 0 & I_r \end{bmatrix} \quad (3.16)$$

The rotational kinetic energy of the wheels is calculated as shown below.

$$T_{rw} = \frac{1}{2} \begin{bmatrix} 0 \\ \dot{\theta}_R \\ \dot{\delta} \end{bmatrix}^T \begin{bmatrix} I_r & 0 & 0 \\ 0 & I_a & 0 \\ 0 & 0 & I_r \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta}_R \\ \dot{\delta} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ \dot{\theta}_L \\ \dot{\delta} \end{bmatrix}^T \begin{bmatrix} I_r & 0 & 0 \\ 0 & I_a & 0 \\ 0 & 0 & I_r \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta}_L \\ \dot{\delta} \end{bmatrix} \quad (3.17)$$

The total kinetic energy is the sum of all the previously calculated kinetic energies

$$T = T_c + T_w + T_{rc} + T_{rw} \quad (3.18)$$

$$T = \left(\frac{I_a + mr^2}{2} \right) ((\dot{x} + d\dot{\delta})^2 + (\dot{y} - d\dot{\delta})^2) + M \left(\frac{\dot{x}_c^2 + \dot{y}_c^2}{2} \right) + \left(I_r + \frac{I_x \sin^2 \theta}{2} - \frac{Ml^2 (\cos^2 \theta)}{2} \right) \dot{\delta}^2 + \frac{\dot{\theta}}{2} (I_y + Ml^2) + Ml^2 \cos \delta \cos \theta \dot{x}_c \dot{\theta} + Ml \cos \delta \sin \theta \dot{y}_c \dot{\delta} + Ml \sin \delta \cos \theta \dot{y}_c \dot{\theta} - Ml \sin \delta \sin \theta \dot{x}_c \dot{\delta} \quad (3.19)$$

The potential energy of the robot is calculated based on the z-position of the COG.

$$U = mgl \cos \theta \quad (3.20)$$

LaGrangian equation:

$$L = T - U \quad (3.21)$$

$$\begin{aligned}
L = & \left(\frac{I_a + mr^2}{2} \right) (\dot{\theta}_R^2 + \dot{\theta}_L^2) + M \left(\frac{\dot{x}_c + \dot{y}_c}{2} \right) + \left(I_r + \frac{I_x \sin^2 \theta}{2} - \frac{Ml^2 (\cos^2 \theta)}{2} \right) \dot{\delta}^2 + \frac{\dot{\theta}}{2} (I_y + Ml^2) + \\
& Ml^2 \cos \delta \cos \theta \dot{x}_c \dot{\theta} + Ml \cos \delta \sin \theta \dot{y}_c \dot{\delta} + Ml \sin \delta \cos \theta \dot{y}_c \dot{\theta} - Ml \sin \delta \sin \theta \dot{x}_c \dot{\delta} - mgl \cos \theta
\end{aligned} \tag{3.22}$$

Euler-LaGrange, for the 3 DOF:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F_x \tag{3.23}$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = F_\theta \tag{3.24}$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\delta}} - \frac{\partial L}{\partial \delta} = F_\delta \tag{3.25}$$

Finding the Euler LaGrange for the translation motion of the robot.

$$\frac{dL}{dx} = 0$$

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x}_c + Ml^2 \cos \delta \cos \theta \dot{\theta} - Ml \sin \delta \sin \theta \dot{\delta}$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = M\ddot{x}_c + Ml^2 \cos \delta \cos \theta \ddot{\theta} - Ml \sin \delta \sin \theta \ddot{\delta}$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = M\ddot{x}_c + Ml^2 \cos \delta \cos \theta \ddot{\theta} - Ml \sin \delta \sin \theta \ddot{\delta} = F_x \tag{3.26}$$

Finding the Euler LaGrange for the tilt of the robot.

$$\begin{aligned}
\frac{\partial L}{\partial \theta} = & \frac{\dot{\delta}^2}{2} (I_x \sin 2\theta + Ml^2 \sin 2\theta) + Ml (-l \cos \delta \sin \theta \dot{x}_c \dot{\theta} + \cos \theta \cos \delta \dot{y}_c \dot{\delta} - \sin \delta \sin \theta \dot{y}_c \dot{\theta} - \\
& \sin \delta \cos \theta \dot{x}_c \dot{\delta} + g \sin \theta)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L}{\partial \dot{\theta}} &= (I_a + mr^2)\dot{\theta} + (I_y + Ml^2)\dot{\theta} + Ml^2 \cos \delta \cos \theta \dot{x}_c + Ml \sin \delta \sin \theta \dot{y}_c \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} &= (I_a + mr^2)\ddot{\theta} + (I_y + Ml^2)\ddot{\theta} + Ml^2 \cos \delta \cos \theta \ddot{x}_c + Ml \sin \delta \sin \theta \ddot{y}_c \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} &= (I_a + mr^2)\ddot{\theta} + (I_y + Ml^2)\ddot{\theta} + Ml^2 \cos \delta \cos \theta \ddot{x}_c + Ml \sin \delta \sin \theta \ddot{y}_c - \\
&\quad \left(\frac{\delta^2}{2} (I_x \sin 2\theta + Ml^2 \sin 2\theta) + Ml(-l \cos \delta \sin \theta \dot{x}_c \dot{\theta} + \cos \theta \cos \delta \dot{y}_c \dot{\delta} - \sin \delta \sin \theta \dot{y}_c \dot{\theta} - \right. \\
&\quad \left. \sin \delta \cos \theta \dot{\delta} \dot{x}_c + g \sin \theta) \right) = F_\theta
\end{aligned} \tag{3.27}$$

Finding the Euler LaGrange for the tilt of the robot.

$$\begin{aligned}
\frac{\partial L}{\partial \delta} &= -Ml^2 \sin \delta \cos \theta \dot{x}_c \dot{\theta} - Ml \sin \theta \sin \delta \dot{y}_c \dot{\delta} + Ml \cos \delta \cos \theta \dot{y}_c \dot{\theta} - Ml \cos \delta \sin \theta \dot{\delta} \dot{x}_c \\
\frac{\partial L}{\partial \dot{\delta}} &= (2I_r + I_x \sin^2 \theta - Ml^2 \cos^2 \theta) \dot{\delta} + Ml \sin \theta \cos \delta \dot{y}_c - Ml \sin \delta \sin \theta \dot{x}_c \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{\delta}} &= (2I_r + I_x \sin^2 \theta - Ml^2 \cos^2 \theta) \ddot{\delta} + Ml \sin \theta \cos \delta \ddot{y}_c - Ml \sin \delta \sin \theta \ddot{x}_c \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{\delta}} - \frac{\partial L}{\partial \delta} &= (2I_r + I_x \sin^2 \theta - Ml^2 \cos^2 \theta) \ddot{\delta} + Ml \sin \theta \cos \delta \ddot{y}_c - Ml \sin \delta \sin \theta \ddot{x}_c - \\
&\quad (-Ml^2 \sin \delta \cos \theta \dot{x}_c \dot{\theta} - Ml \sin \theta \sin \delta \dot{y}_c \dot{\delta} + Ml \cos \delta \cos \theta \dot{y}_c \dot{\theta} - Ml \cos \delta \sin \theta \dot{\delta} \dot{x}_c) = F_\delta
\end{aligned} \tag{3.28}$$

3.3 Linearisation

To linearise the system we assume that the system is stable when the tilt angle, theta, is 0.

Additionally, we assume that the yaw angle is very small that it can be estimated to be 0.

From the Euler-Lagrange equations, and the assumptions, the following equations linearised equations are derived:

$$\ddot{x} = \frac{-M^2 l^2 g r^2}{(M l^2 + I_y)(M r^2 + 2m r^2 + 2I_a) - M^2 l^2 r^2} \dot{\theta} + \frac{r(M l^2 + I_x + M l r)}{(M l^2 + I_y)(M r^2 + 2m r^2 + 2I_a) - M^2 l^2 r^2} \dot{x} \quad (3.29)$$

$$\ddot{\theta} = \frac{(M r^2 + 2m r^2 + 2I_a) M g l}{(2M m r^2 + 2I_a) I_y + 2M l^2 (m r^2 + I_a)} \dot{\theta} - \frac{(M r^2 + 2I_a) + M l r}{(2M m r^2 + 2I_a) I_y + 2M l^2 (m r^2 + I_a)} \dot{x} \quad (3.30)$$

$$\ddot{\delta} = \frac{d}{2r d^2 \left(M + \frac{I_d}{r^2} \right) + r I_z} \dot{x} \quad (3.31)$$

3.4 State-Space Model

The parameters of the 'a' matrix of the state space are the definitions of the 3 DOF and the rate of change of the parameters with respect to time.

$$[a] = \begin{bmatrix} x \\ \theta \\ \delta \\ \dot{x} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} \quad (3.32)$$

The 'b' matrix is the input matrix of the system.

$$[b] = \begin{bmatrix} C_R \\ C_L \end{bmatrix} \quad (3.33)$$

The overall state-space model is shown below:

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{\delta} \\ \ddot{x} \\ \ddot{\theta} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & a_{42} & 0 & a_{44} & 0 & 0 \\ 0 & a_{52} & 0 & a_{54} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \delta \\ \dot{x} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ b_{41} & b_{42} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{bmatrix} \begin{bmatrix} C_R \\ C_L \end{bmatrix} \quad (3.34)$$

From the linearised equations (3.29) - (3.31) the state-space model in equation (3.34) is developed such that:

$$a_{42} = -\frac{M^2 l^2 g}{M I_y + 2(I_y + M l^2)\left(m + \frac{I_a}{r^2}\right)} \quad (3.35)$$

$$a_{52} = \frac{M^2 g l + 2 M g l \left(m + \frac{I_a}{r^2}\right)}{M I_y + 2(I_y + M l^2)\left(m + \frac{I_a}{r^2}\right)} \quad (3.36)$$

$$a_{44} = \frac{\left(\frac{M l^2}{r^2} + \frac{M l}{r} + \frac{I_y}{r^2}\right)}{\left(M I_y + 2(I_y + M l^2)\left(m + \frac{I_a}{r^2}\right)\right)} \quad (3.37)$$

$$a_{54} = \frac{\left(\frac{2 I_a}{r^2} + (2m + M) + \frac{M l}{r}\right)}{\left(M I_y + 2(I_y + M l^2)\left(m + \frac{I_a}{r^2}\right)\right)} \quad (3.38)$$

$$b_{41} = b_{42} = \frac{\frac{(I_y + M l^2)}{r} + M l}{M I_y + 2(I_y + M l^2)\left(m + \frac{I_a}{r^2}\right)} \quad (3.39)$$

$$b_{51} = b_{52} = \frac{\frac{-M(r+l)}{r} - 2\left(m + \frac{I_a}{r^2}\right)}{M I_y + 2(I_y + M l^2)\left(m + \frac{I_a}{r^2}\right)} \quad (3.40)$$

$$b_{61} = -b_{62} = \frac{\frac{d}{2r}}{I_z + \frac{d^2}{2r}\left(m r + \frac{I_a}{r}\right)} \quad (3.41)$$

3.6 Modified state-space for voltage output

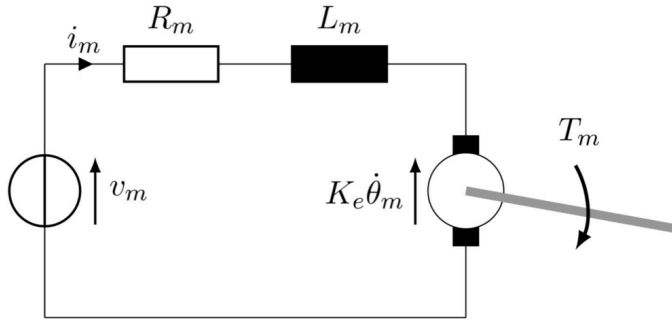


Figure 3.3 Motor circuit [17]

Figure 3.3 shows the motor circuit. electrical inductance of the motor has the symbol L_m . the inductance is taken to be zero, this is because the inductance is 0 when the motor is at rest. This is based on the assumption that the inductor has already discharged its current from the previous motor operation.

From the schematic in figure 3.3, the following formula is derived [17]:

$$L_m \frac{di_m}{dt} + Ri_m = Ri_m = v_m - e \quad (3.42)$$

Where is the electromotive force which has the following equation [17]:

$$e = K_e \left(\frac{x_w}{I_a} - \dot{\theta} \right) \quad (3.43)$$

From equations (3.40) and (3.41), the current is found below [17].

$$i_m = \frac{v_m}{R} - \frac{K_e}{R} \left(\frac{x_w}{I_a} - \dot{\theta} \right) \quad (3.44)$$

To convert from current torque, the current is multiplied by the torque constant of the motors being used, the final equation in terms of torque is shown below [17].

$$\tau_m = \frac{K_t}{R} v_m - \frac{k_e k_t}{R} \left(\frac{x_w}{I_a} - \dot{\theta}_b \right) \quad (3.45)$$

By using equation (3.45), the state-space model is modified such that the output is in terms of voltage instead of torque, the finalized State-Space model is shown below.

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{\delta} \\ \ddot{x} \\ \ddot{\theta} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & a_{42} & 0 & a_{44} & 0 & 0 \\ 0 & a_{52} & 0 & a_{54} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \delta \\ \dot{x} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ b_{41} & b_{42} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \quad (3.46)$$

$$a_{42} = -\frac{M^2 l^2 g}{M I_y + 2(I_y + M l^2) \left(m + \frac{l a}{r^2}\right)} \quad (3.47)$$

$$a_{52} = \frac{M^2 g l + 2 M g l \left(m + \frac{l a}{r^2}\right)}{M I_y + 2(I_y + M l^2) \left(m + \frac{l a}{r^2}\right)} \quad (3.48)$$

$$a_{44} = \frac{2 k_t k_e \left(\frac{M l^2}{r^2} + \frac{M l}{r} + \frac{l y}{r^2}\right)}{R \left(M I_y + 2(I_y + M l^2) \left(m + \frac{l a}{r^2}\right)\right)} \quad (3.49)$$

$$a_{54} = \frac{2 k_t k_e \left(\frac{2 l a}{r^2} + (2 m + M) + \frac{M l}{r}\right)}{R \left(M I_y + 2(I_y + M l^2) \left(m + \frac{l a}{r^2}\right)\right)} \quad (3.50)$$

$$b_{41} = b_{42} = \frac{k_t \left(\frac{(I_y + M l^2)}{r} + M l\right)}{R \left(M I_y + 2(I_y + M l^2) \left(m + \frac{l a}{r^2}\right)\right)} \quad (3.51)$$

$$b_{51} = b_{52} = \frac{k_t \left(\frac{-M(r+l)}{r} - 2 \left(m + \frac{l a}{r^2}\right)\right)}{R \left(M I_y + 2(I_y + M l^2) \left(m + \frac{l a}{r^2}\right)\right)} \quad (3.52)$$

$$b_{61} = -b_{62} = \frac{k_t \left(\frac{d}{2r}\right)}{R \left(I_z + \frac{d^2}{2r} \left(m r + \frac{l a}{r}\right)\right)} \quad (3.53)$$

Using the finalized state-space, and the parameters specific to the robot being designed given previously in table 3.2, the calculate state-space is shown below.

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{\delta} \\ \ddot{x} \\ \ddot{\theta} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -35.952 & 0 & 2027.7 & 0 & 0 \\ 0 & 311.14 & 0 & 14741 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \delta \\ \dot{x} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 50.694 & 50.694 \\ -368.53 & -368.53 \\ 387.06 & -387.06 \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \quad (3.53)$$

Chapter 4: LQR Controller Design and Simulation

4.1 MATLAB simulation of the system (Weighting Matrices)

An LQR controller is a full-state feedback controller which is tuned based weighting matrices which are set based on the desired output of the system. To code the self-balancing robot on MATLAB, the following steps were followed:

1. Define the parameters of the SBR as shown in table 3.2.
2. Write the equations of the A, B, C, and D matrices which were derived in Chapter 3.
3. Using the “ss(A,B,C,D)” function on MATLAB, the state-space model is constructed from the matrices in step 2.
4. Define the Q and R weighting matrices.
5. Use the “lqr(A,B,Q,R)” to obtain the k-gain matrix of the LQR controller.
6. Find the step response of the system.
7. Vary the weighting matrices and analyse the effects on the step response.

An example of the results of the steps outlined above is shown below.

Setting the State-Space Matrices:

A =

0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	-35.952	0	2027.7	0	0
0	311.14	0	14741	0	0
0	0	0	0	0	0

Figure 4.1 A-matrix

B =

0	0
0	0
0	0
50.694	50.694
-368.53	-368.53
387.06	-387.06

Figure 4.2 B-matrix

C =

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

Figure 4.3 C-matrix

D =

0	0
0	0
0	0
0	0
0	0
0	0

Figure 4.4 D-matrix

Setting the Q and R weighting matrices. For the first run, these matrices are set randomly to provide a starting point for the analysis.

Q =

100	0	0	0	0	0
0	100	0	0	0	0
0	0	100	0	0	0
0	0	0	100	0	0
0	0	0	0	100	0
0	0	0	0	0	100

R =

1	0
0	1

Figure 4.4 Q & R weighting matrices

From step 5, the gain matrix is calculated.

K =

$$\begin{bmatrix} -7.0711 & 6.9692 & 7.0711 & 156.33 & 7.0761 & 7.0724 \\ -7.0711 & 6.9692 & -7.0711 & 156.33 & 7.0761 & -7.0724 \end{bmatrix}$$

Figure 4.5 gain matrix

Finally, the step response of the system, based on the six states of the system, which are used in the A-matrix of the system. The step response is shown in figure 4.6.

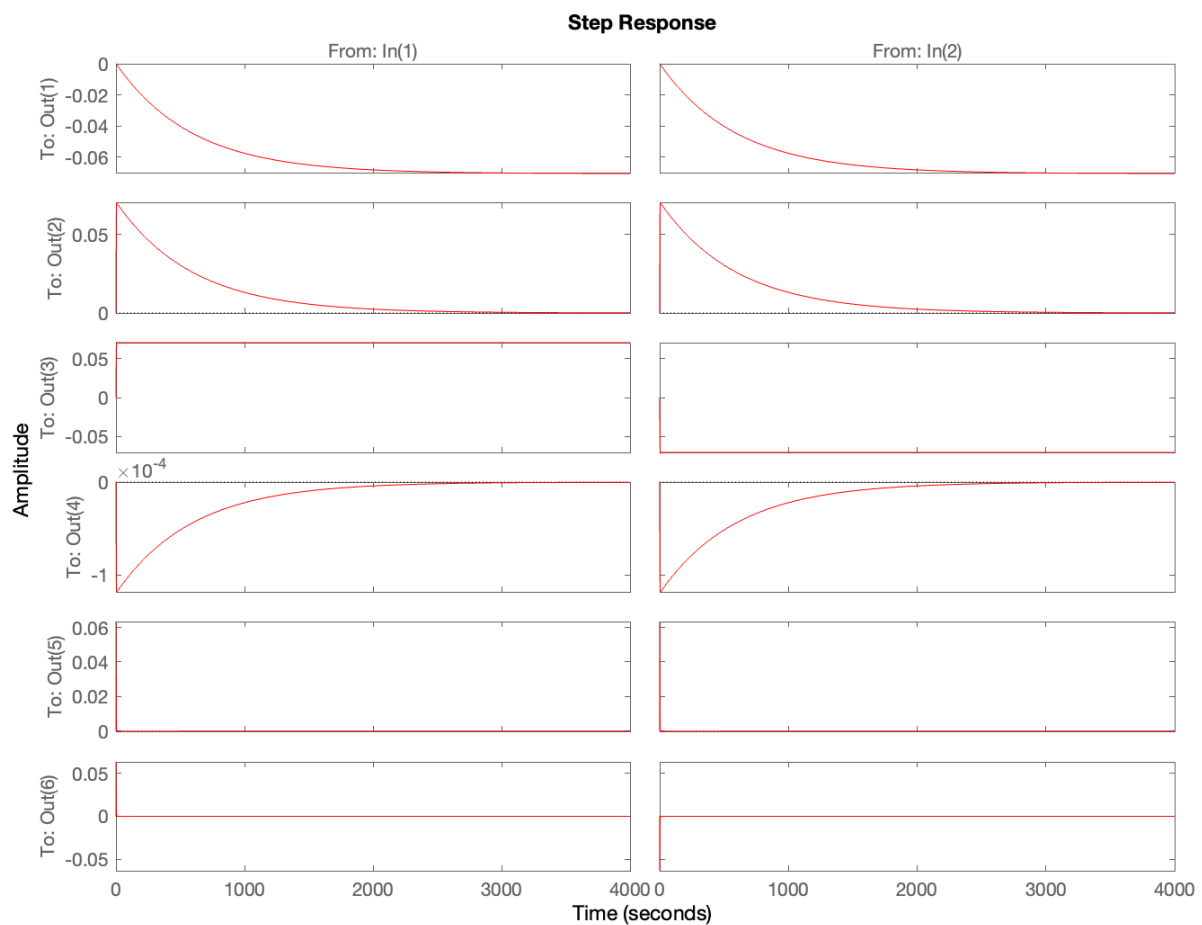


Figure 4.6 initial Step response

The first row of graphs shows the positional response of the system, it can be seen from these graphs that the response time is 2500s. The second row shows the tilt angle response which again settles after 2500s. The third row is the yaw angle response, the step response of this state is nearly 0, as the settling criteria is based on the tilt angle reaching zero, which is controlled by the translational speed of the robot, not the yaw angle. The following three

rows of graphs shows the response of the translational velocity, tilt angular velocity and yaw angular velocity respectively.

- Varying the Q matrix.

Since the first row of the Q matrix is used for controlling the positional response of the robot, the weighting of that parameter will be increased. The second row is for the tilt angle, which is the basis of deeming the system stable, this value will be increased significantly.

Meanwhile, the yaw parameters already provided an acceptable response, they will not be varied. Moreover, the translational velocity of the robot, how fast it can move to adjust the base of the robot to be directly under COG, it the method by which the tilt angle is adjusted, thus this value will also be increased. To balance the robot, the desires tilt angular velocity should be minimized. The modified weighting matrices and gain matrix are shown below.

Q =

$$\begin{bmatrix}
 1000 & 0 & 0 & 0 & 0 & 0 \\
 0 & 5000 & 0 & 0 & 0 & 0 \\
 0 & 0 & 100 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1000 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

R =

$$\begin{bmatrix}
 1 & 0 \\
 0 & 1
 \end{bmatrix}$$

K =

$$\begin{bmatrix}
 -22.361 & 49.72 & 7.0711 & 63.095 & 0.805 & 0.13516 \\
 -22.361 & 49.72 & -7.0711 & 63.095 & 0.805 & -0.13516
 \end{bmatrix}$$

Figure 4.7 modified Q matrix

The step responses are shown below.

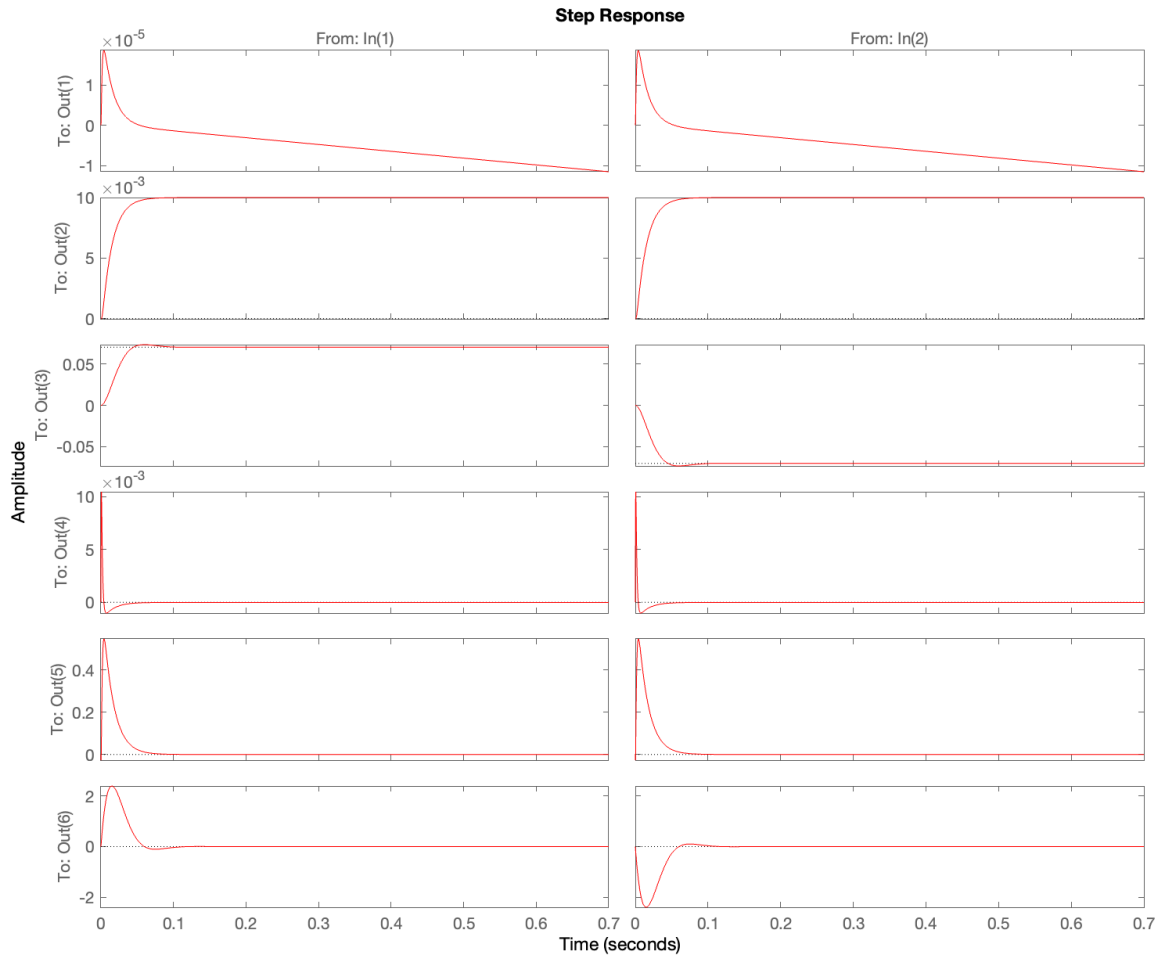


Figure 4.8 step response of modified Q matrix

From the gain matrix it can be observed that the gain of the desired parameters (position and tilt angle) has increased independently of the yaw states. Additionally, the response time of the system improved significantly, with the tilt angle parameter, stabilizing within 0.1s.

- Varying the R matrix.

The R matrix is used to adjust the effort consumed by the system to reach stability. By reducing the effort, the response time will increase. An example of reducing R matrix is shown below.

Q =

1000	0	0	0	0	0
0	5000	0	0	0	0
0	0	100	0	0	0
0	0	0	1000	0	0
0	0	0	0	1	0
0	0	0	0	0	0

R =

3	0
0	3

K =

-12.91	28.571	4.0825	52.671	0.48129	0.1027
-12.91	28.571	-4.0825	52.671	0.48129	-0.1027

Figure 4.9 increased R matrix

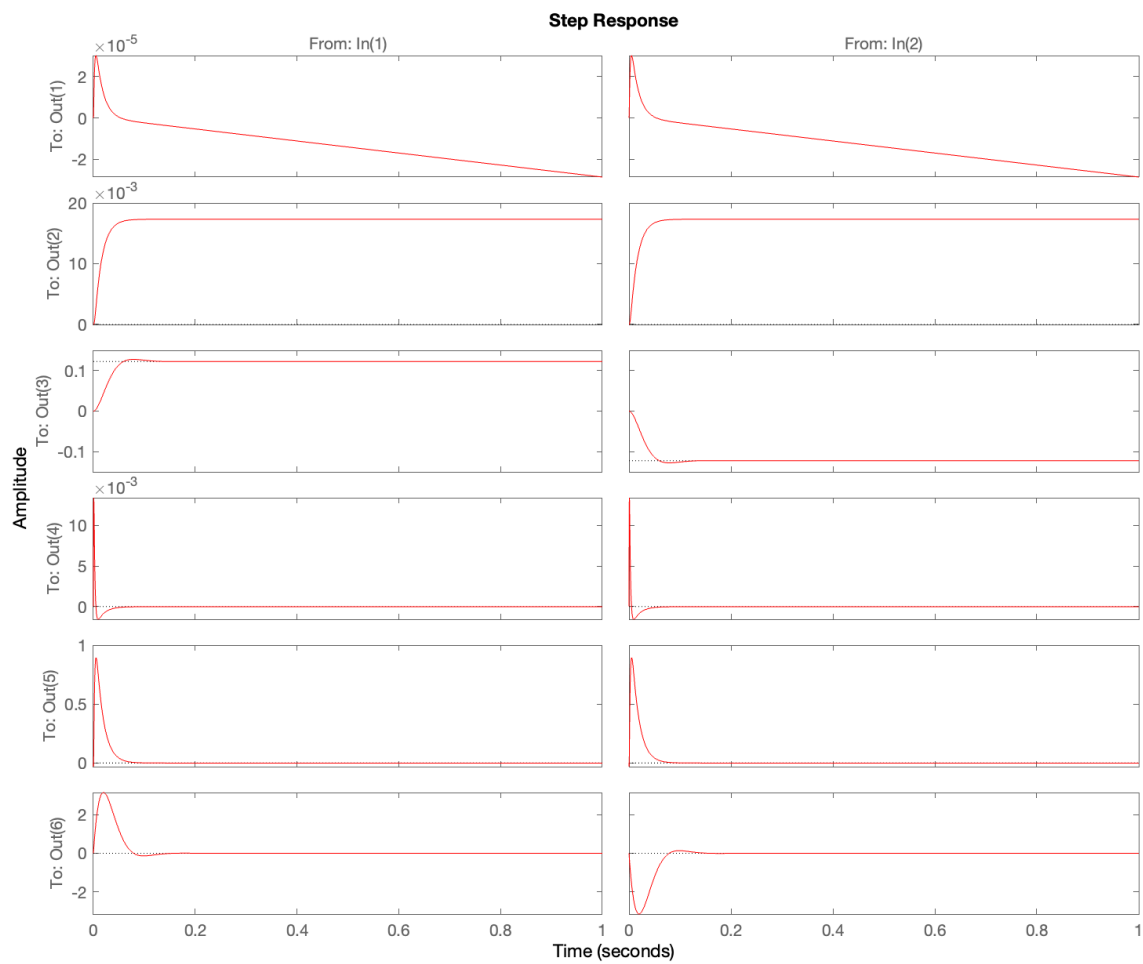


Figure 4.10 Step response for increased R-matrix

The gain matrix shown in figure 4.9 has decreased from the previous configuration, and the step response shown in figure 4.10 has increased to 0.2s. To test minimizing the step response, the R matrix values are decreased as shown below.

Q =

1000	0	0	0	0	0
0	5000	0	0	0	0
0	0	100	0	0	0
0	0	0	1000	0	0
0	0	0	0	1	0
0	0	0	0	0	0

R =

0.01	0
0	0.01

K =

-223.61	500.15	70.711	316.13	7.7569	0.42742
-223.61	500.15	-70.711	316.13	7.7569	-0.42742

Figure 4.11 decreased R-matrix

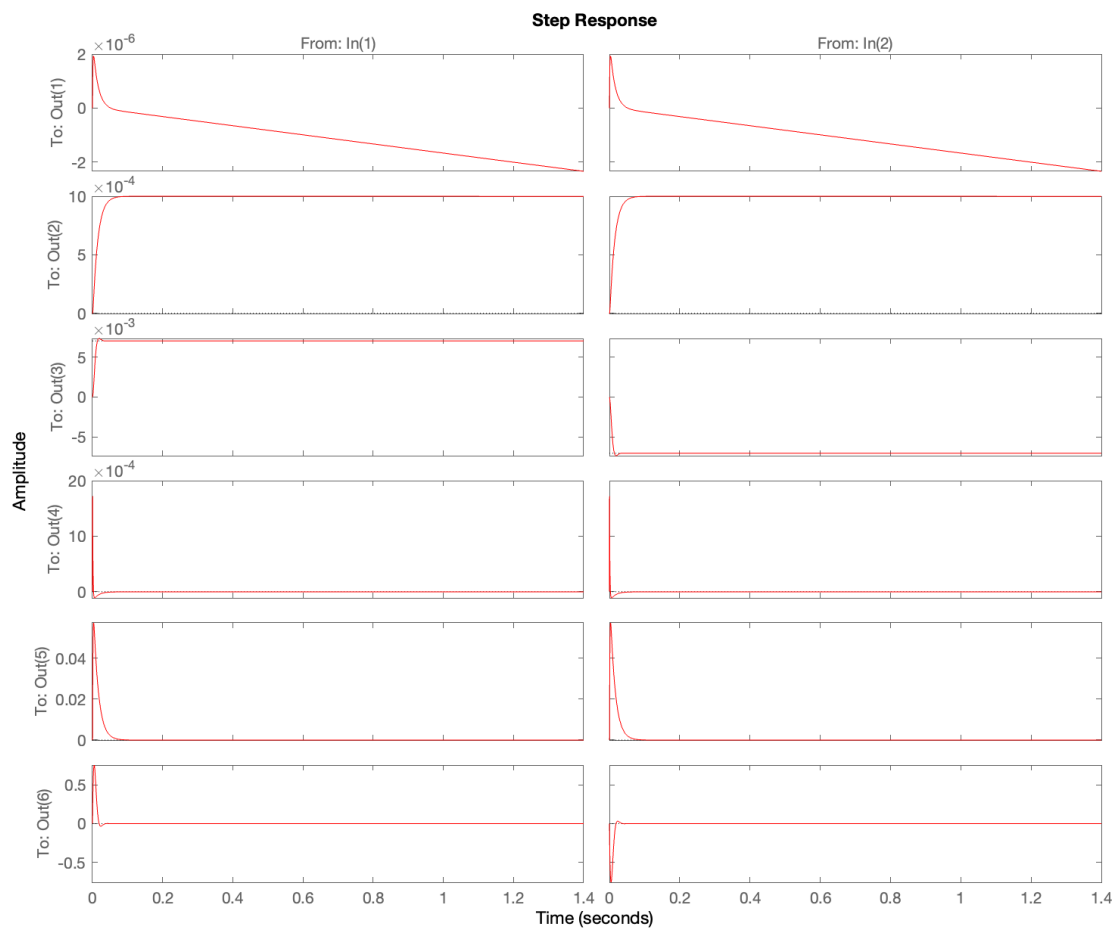


Figure 4.12 step response for decreased R-matrix

The gain matrix shown in figure 4.11 has the highest values tested, as this has the highest Q values and the lowest R values. This is reflected in the response time as shown in figure 4.12 which dropped to 0.06 seconds. This response time may be challenging to achieve in the physical system; however, it was done to reflect the ease of tuning the LQR controller to achieve the most desirable results.

From the literature review, [15] and [16] provide methods of obtaining the most suitable weighting matrices values using the Bees Algorithm (BA) and the Particle Swarm Optimisation (PSO). Both approaches are based on an iterative approach which uses the response of the system to adjust the weighting matrices either until the preset number of iterations or the desired response is achieved. While these methods are effective, for the purposes of this project, the approach used above was effective in producing an acceptable response.

4.2 Simulink simulation (Robustness Testing)

To test the robustness of the controller, a Simulink model of the robot was created which included a step function input to the robot, the schematic of the system is shown in the figure 4.13 below. The step function labelled initial force is used to act as external disturbance to the system, which is varied to test the force that the system can handle.

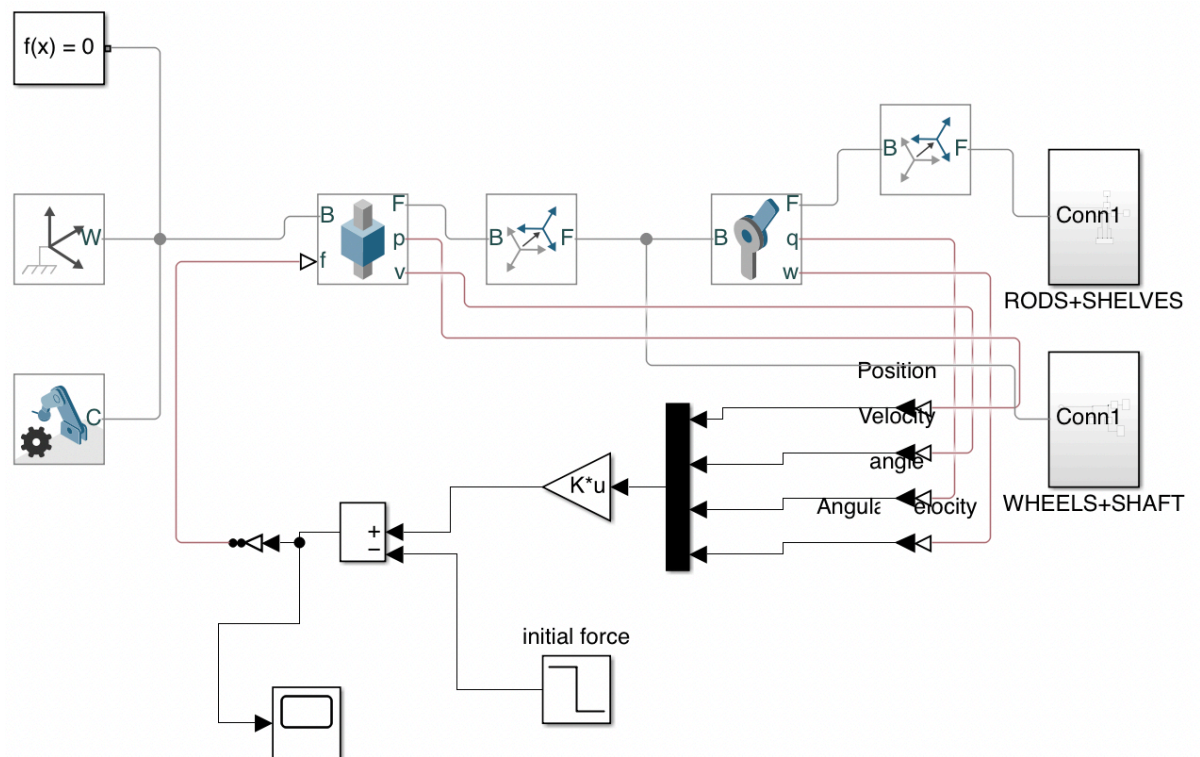


Figure 4.13 Simulink model for LQR control

For this simulation only 2DOF were considered, which are the translational motion of the robot and the tilt angle. This is done because the external force would not affect the yaw motion of the SBR. Additionally, a MATLAB code was written to derive the gain matrix, as done previously in section 4.1. The gain matrix is shown below.

Kr =

-31.6228 -29.4271 148.5711 42.5850

Figure 1 Gain matrix LQR robustness test

Table 1 Robustness test LQR vs PID

Force	LQR controller		PID controller	
	x-position	Tilt angle	x-position	Tilt angle
10	yes	yes	yes	yes
100	yes	yes	yes	yes
150	yes	yes	yes	yes
200	yes	yes	no	yes
250	yes	yes	no	yes
300	yes	yes	no	yes
350	yes	yes	no	yes
400	yes	yes	no	yes
450	yes	yes	no	yes
500	yes	yes	no	yes
550	yes	yes	no	yes
600	yes	yes	no	yes
650	yes	yes	no	yes
700	yes	yes	no	yes
750	yes	yes	no	yes
800	yes	yes	no	no
850	yes	yes	no	no
900	yes	yes	no	no
2550	yes	yes	no	no
2600	no	no	no	no

Chapter 5: Implementation

Unfortunately, I experienced delays in ordering the components required for my project and delays in manufacturing. I submitted the order form for my components on the 6th of July and Mark Birkin submitted my order form on the 8th of July. However, the order was delayed by the finance department due to entering the new financial year, and my motors did not arrive until the 4th of August (almost a month after ordering the motor even though I ordered them from Amazon which have next day delivery) and the wheels did not arrive until the 12th of August.

I had already made CAD models of the chassis of the robot and spoke to Andrew Plummer in manufacturing about manufacturing the chassis, however we needed the motors to make sure we got the layout of the screws on the chassis right, since the motors came with mounting brackets. Since the motors were delayed then the work on the chassis was also delayed. I sat down with Andrew Plummer on the 10th of August to get the measurements from the brackets and to start working on the chassis. However, the chassis manufacturing also got delayed for reasons unknown to me. I was told that it is a simple design, and that the department already had all the metal that was required to get it done. The chassis is still not done as of writing this thesis.

Due to delays in the components and delays in the manufacturing of the chassis, I had to change the objectives of my project from design and implementation to design and simulation. However, the implementation strategy is discussed in this section.

5.1 System Design

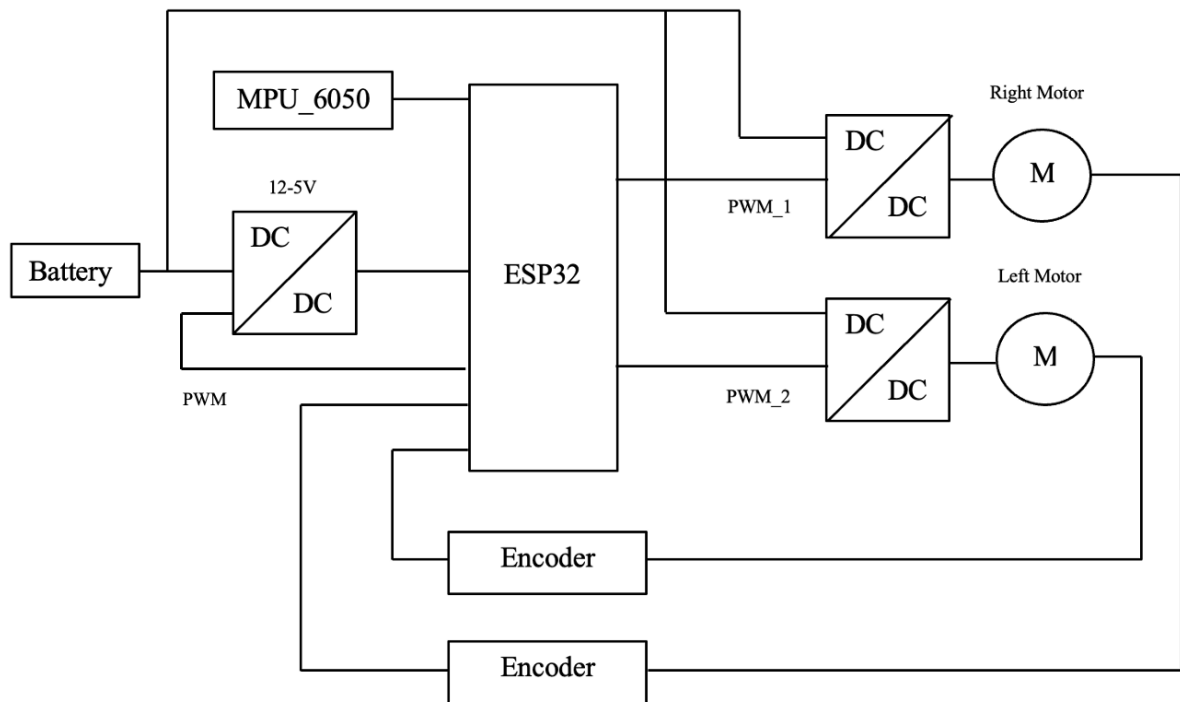


Figure 5.1 System Design

The design of the system is shown above in figure 5.1. The battery will supply 12V to the system, however the input voltage of the microcontroller to be used is only 5V, and so, the battery is connected to a DC-DC buck converter which will convert the voltage from 12V to 5V. The gyroscope which is the MPU 6050 is used to find the changes in the angle of the robot with respect to the x, y, and z-axis. This is interfaced with the ESP32 to extract the tilt angle, and the yaw angle. Additionally, the encoders input the speed of the motors to the ESP32. The LQR controller design in chapter 4 is coded onto the ESP32 using Arduino IDE. Using the inputs to the ESP32 and the LQR controller, the code will output the desired voltage to be sent to each motor. These voltages are converted to duty ratios based on the design of the DC-DC buck-boost converter which feeds the motors, the design of all three converters is discussed below in this chapter.

5.2 Description of components

- Batteries

The batteries used for this project are 4 lithium ion 18650 batteries [18]. The 18650s were selected because they manufactured with built in PCB protection against overcurrent. This makes them much safer to use, as lithium-ion batteries are limited in their charge and discharge current. If the maximum rated current is exceeded, and no protection is implemented, there is a risk of an electrical fire which is challenging to put out. While testing and building the robot a bench supply is used at the same voltage level that will be supplied by the batteries, this is done to ensure that the current drawn by the motors does not exceed the battery limit. The output of each battery from [18] is 3.7V, the output of the batter pack is 14.8V.

- ESP32

The microcontroller used in this project is the ESP32-WROOM-32D [19] from Esspressif systems. Figure 5.2 shows the pin layout of the microcontroller.

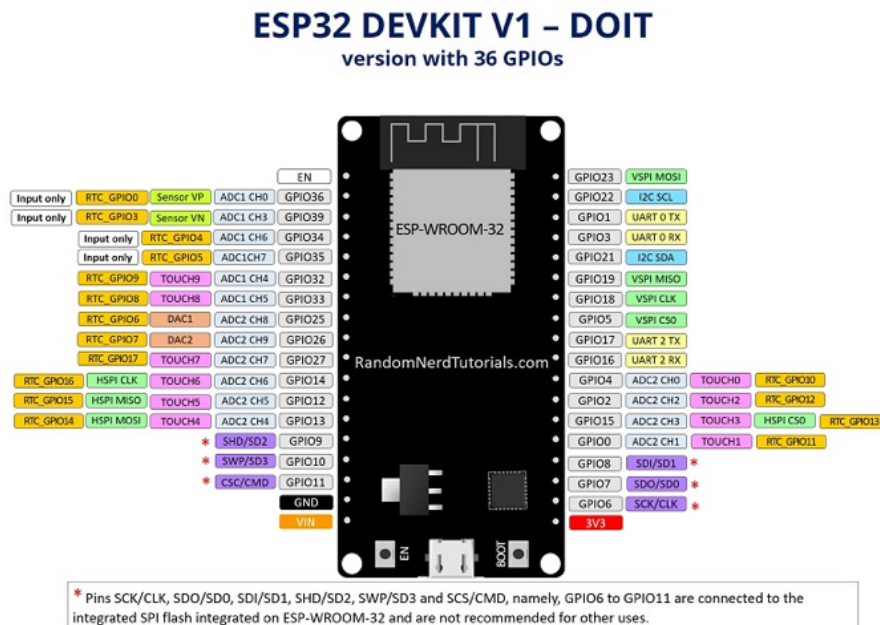
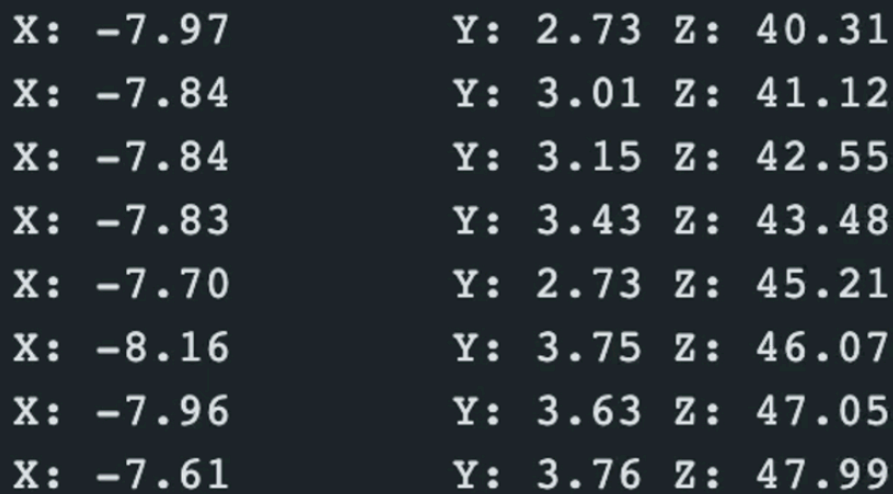


Figure 5.2 ESP32 Pin layout

- Gyroscope

The gyroscope selected for this project is the MPU6050 [20]. This gyro provides the changes in the angles for the 3 axis of rotation. To extract angles from the gyroscope, it is first connected to the ESP32. The pins used from the MPU6050 are the VCC, GND, SCL, and SDA, which are connected to the 3v3, GND, GPIO22, and GPIO21 shown in figure 5.2 respectively. After the connections are made, code is written on Arduino IDE, this is provided in the submitted “Master’s file”, to interface the MPU6050 with the ESP32. Figure 5.3 below, shows the extracted angles, where X represents the angle with respect to the x-axis, Y represents the angle with respect to the y-axis, referred to throughout this report as the yaw angle. Z is the angle with respect to the z-axis, the tilt angle. These angles are fed back into the system and used for the control process.



X: -7.97	Y: 2.73	Z: 40.31
X: -7.84	Y: 3.01	Z: 41.12
X: -7.84	Y: 3.15	Z: 42.55
X: -7.83	Y: 3.43	Z: 43.48
X: -7.70	Y: 2.73	Z: 45.21
X: -8.16	Y: 3.75	Z: 46.07
X: -7.96	Y: 3.63	Z: 47.05
X: -7.61	Y: 3.76	Z: 47.99

Figure 5.3 Extracting angles from gyroscopes

- Motors

Two DC775 motors will be used, the datasheet is given in [21]. These motors have an rpm of 10000rpm for an input voltage of 12V, and 20000rpm at 24V. This motor was selected due to the high torque, as it will not limit the performance of the system and will be able to provide the require rpm. The motors are connected to encoders, which are fed back into the system to provide the translational velocity.

5.3 DC/DC converters design

Since the input voltage of the ESP32 is 5V and the output voltage of the battery pack is 12V, a buck converter will be used to step down the voltage from the battery so it can be supplied to the ESP32.

The topology of a DC-DC buck converter is shown below in figure 5.4, where V_s is the supply voltage of the converter, in this case it is 12V. V_o is the output voltage, which is fed to the ESP, 5V.

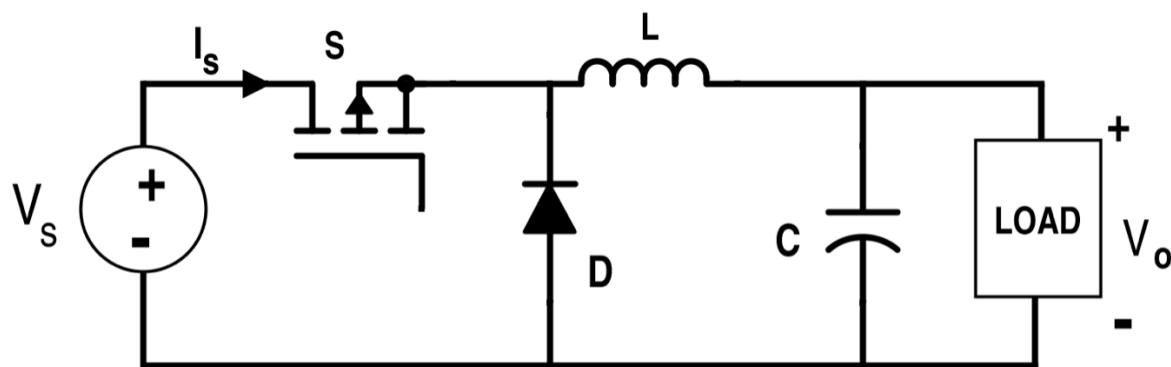


Figure 5.4 buck converter topology

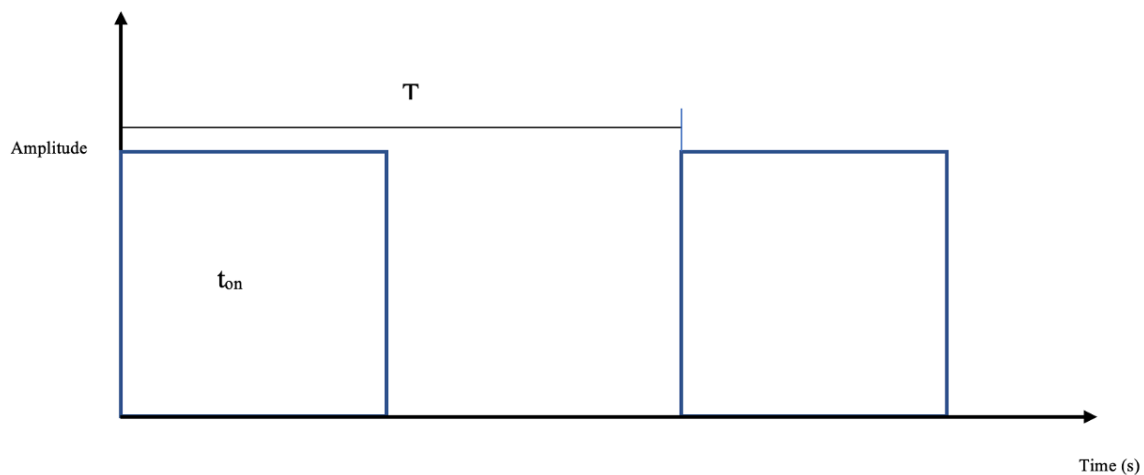


Figure 5.5 PWM signal

To design the buck converter the following design considerations are taken into account. The inductor, L , is used to smooth the current ripple at the output and the capacitor, C , is used to regulate the voltage ripple at the output, the load in this case is the internal resistance of the motor which was from the datasheet to be 1 ohm. The switch, S , is fed a PWM signal as

shown above in figure 5.5 which controls the on/off states of the switch. T is the period of the signal in seconds, f_s is the frequency in Hz, t_{on} is the duration of each pulse, and d is the duty cycle.

$$f_s = \frac{1}{T} \quad (6.1)$$

$$d = \frac{t_{on}}{T} \quad (6.2)$$

For the buck converter the duty cycle is calculated as,

$$d = \frac{V_o}{V_s} = \frac{5}{12} = 0.4167 \quad (6.3)$$

The maximum current ripple allowed is 2% and the maximum voltage ripple is 3%. This is used to size the inductor and capacitor as shown in the equations below.

$$L = \frac{(V_s - V_o) \times d}{f_s \times \Delta I_L} = 7.29 \text{ mH} \quad (6.4)$$

$$C = \frac{\Delta I_o}{8 f_s \Delta V_o} = 4.167 \mu\text{F} \quad (6.5)$$

Buck converter simulation

To test verify that the designed buck converter will output the desired signal, the following simulation was down in which the values of V_s , d , L , C and R matched the calculated values.

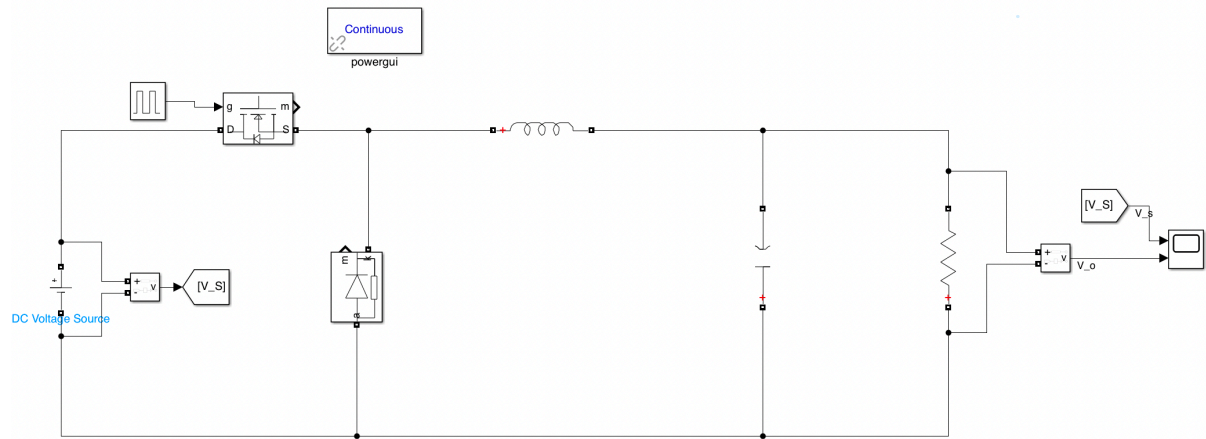


Figure 5.6 supply voltage to ESP32

The input and output voltages of the system are shown in figure 5.7, it can be observed that the designed buck converter can convert the 12V to 5V which will be supplied to the ESP32.

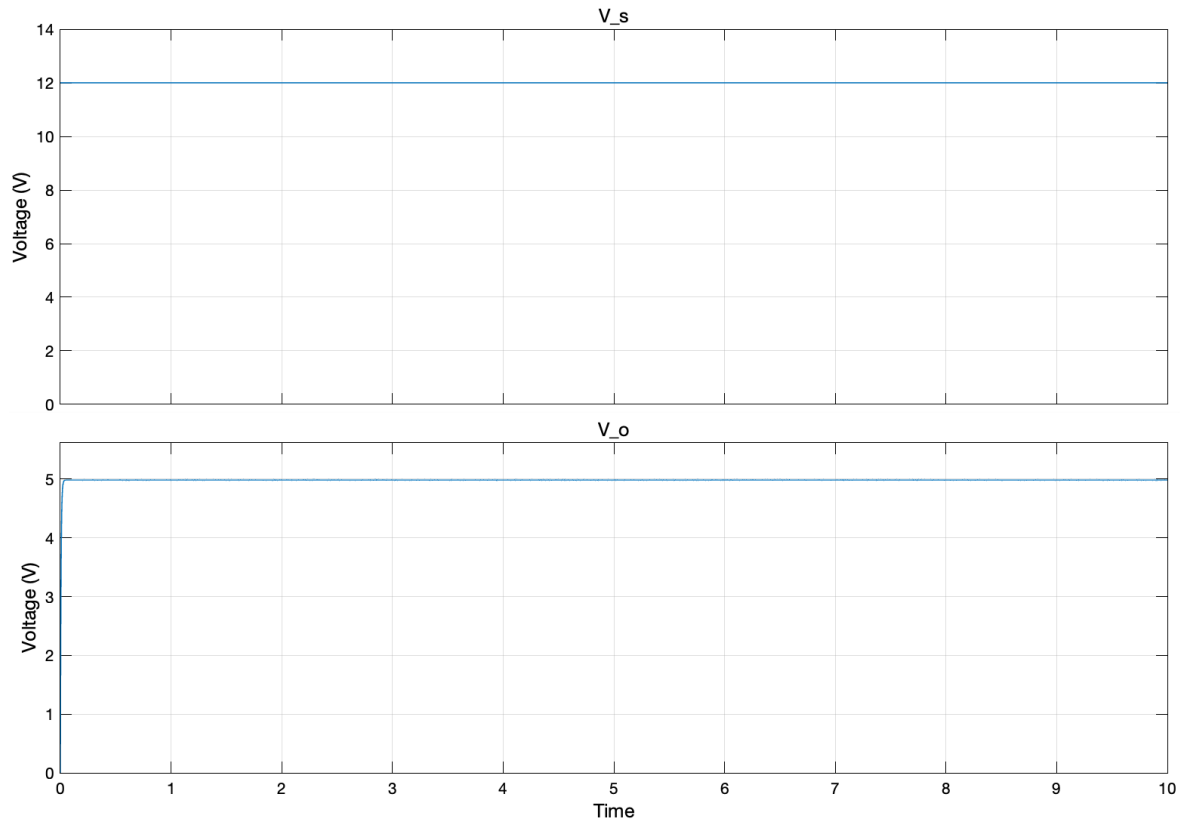


Figure 5.7 Output voltage of Converter fed to ESP32

For the DC-DC converters that will feed the motors, buck converters are used, however, the voltage required by the motors is constantly changing.

Given that the motors used operate at 10krpm at 12V and 20krpm at 24V. the output voltage is limited to 12V as to avoid the risks of operating at such high rotational speeds and to limit the current drawn from the batteries, as Lithium-ion batteries are very dangerous at high currents and could cause an electrical fire.

The output of the controller discussed in Chapter 3 is the desired input voltages to the motors, this was done by using the torque constant and the back emf constant of the motors to convert the output to a voltage that will produce the desired rotational speed of the motors. Since the voltage required by the motors is constantly changing, a function is used to convert the desired voltage and the supply voltage from the battery to a duty cycle using equation 6.3 which is used to control the switch of the buck converter.

To simulate this type of controller the same topology of the controller used in figure 5.6 with exception of using a function block instead of a pulse generator to generate a PWM signal as shown below in figure 5.8. The graph in figure 5.9 shows that the converter was able to deliver the desired voltage which is shown in the constant block fed into the function block.

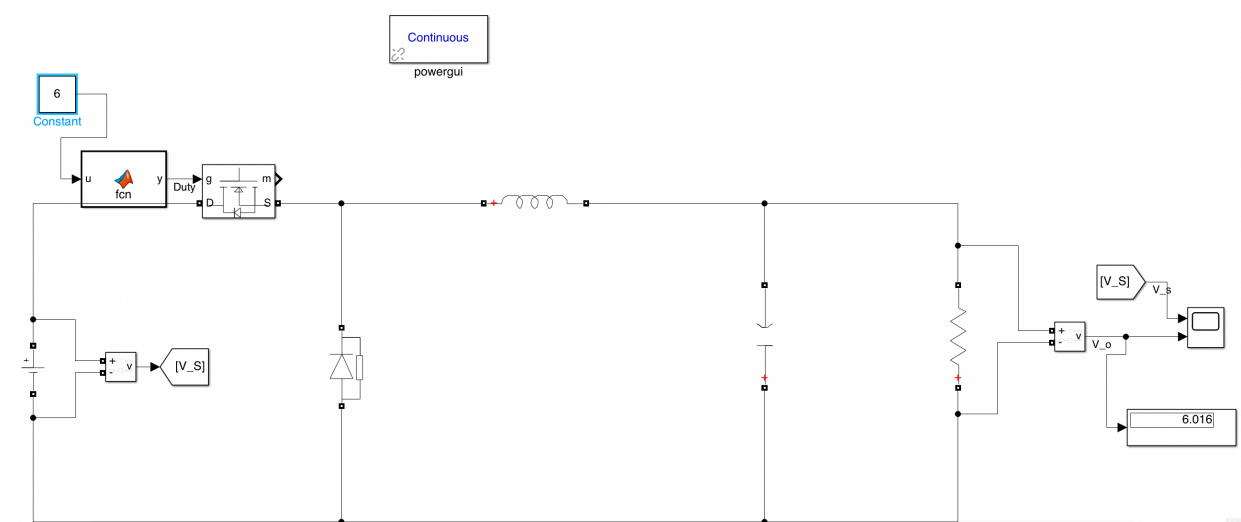


Figure 5.8 Varying PWM signal converter fed to motors

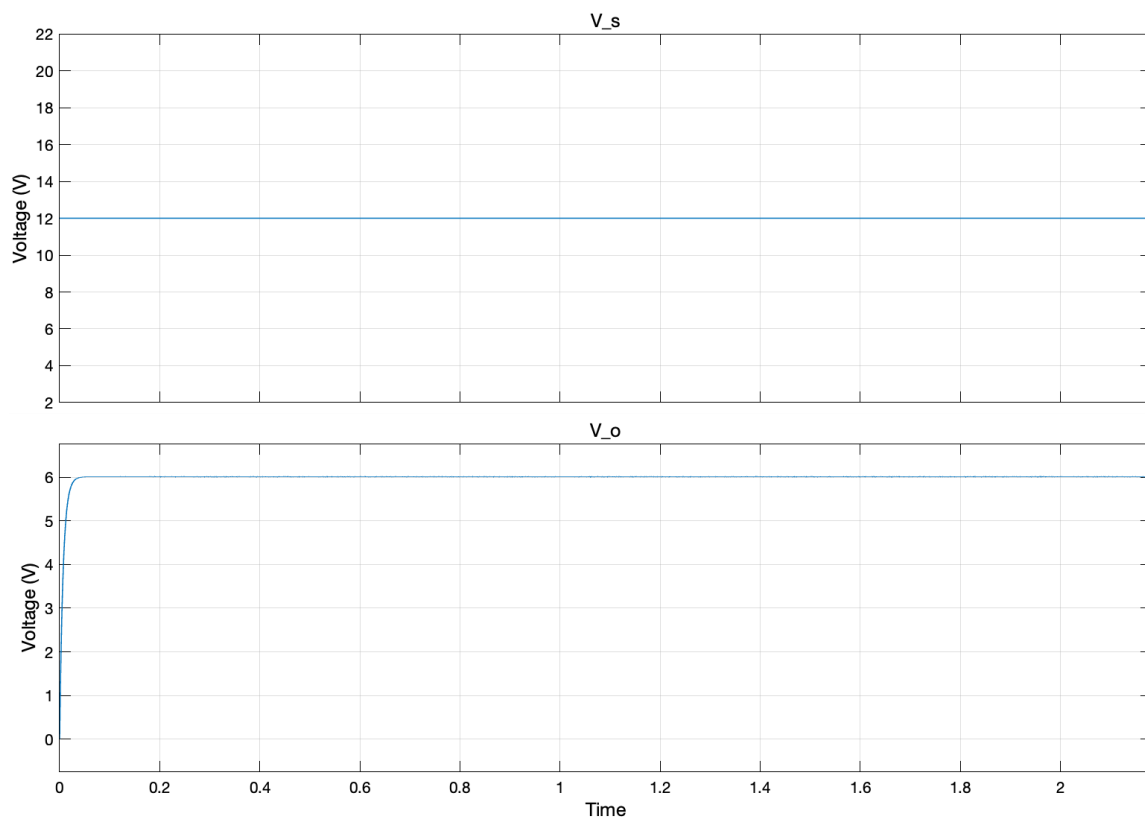


Figure 5.9 Output voltage of converters fed to motors

5.4 Design of the chassis

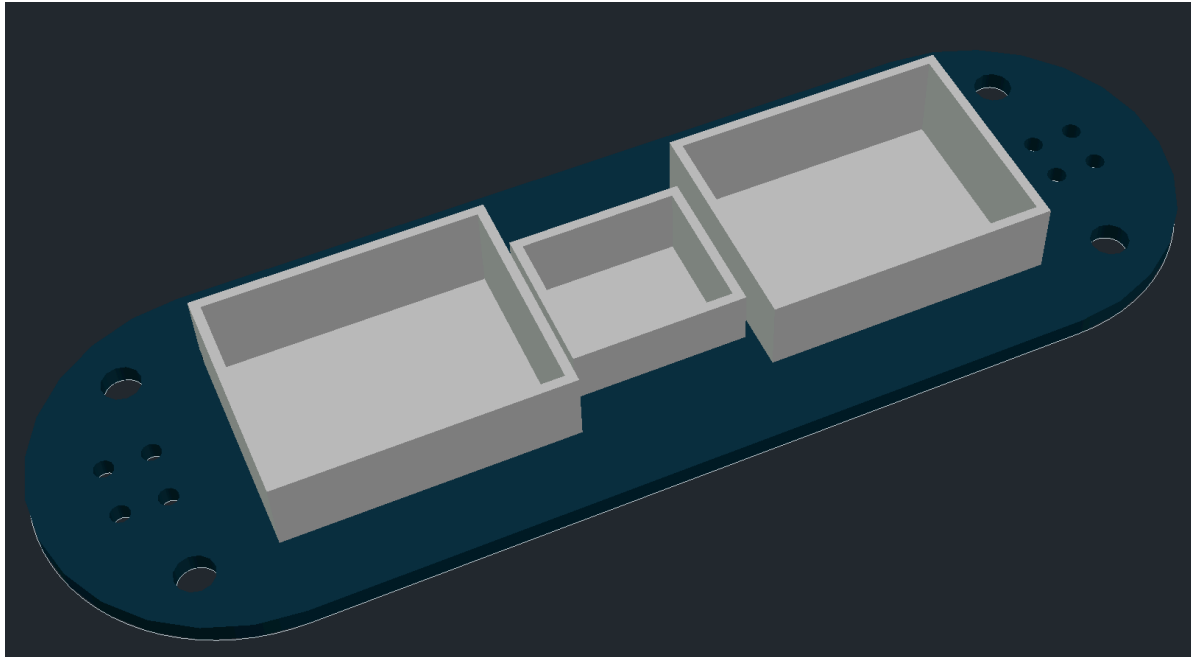


Figure 5.10 AutoCAD model of chassis

Figure 5.10 shows an AutoCAD model of the chassis. The white compartments to the left and right side of the chassis are designed to house the motor controls, while the middle compartment is designed to house the battery pack. The screw holes on both sides of the chassis are designed to match those of the mounting bracket of the motor. Finally, the larger circular cutouts or the corners of the chassis are used to erect rods to add a shelf. The circuitry discussed previously in this section is attached to the upper level. The bulk of the weight of the components is placed directly on the chassis of the robot as to lower the center of mass of the robot making it easier to balance.

5.5 Cost

The cost breakdown of the robot is shown below, this table does not include the cost of the switches, resistors, inductors, and capacitor required to implement the DC-DC converters. This is since they were supplied by the university and the converters were not bought from an external supplier. The budget of the project is £200. The total cost of the project is calculated to be £146.86.

Table 5.1 Cost Breakdown

Product	Units	Cost per Unit	Cost (before tax)	Cost (after tax)
775 DC motor	2	19.9	33.17	39.8
Motor Control	2	20.00	33.32	40.00
ESP 32	1	12.00	10.00	12.00
18650 Lithium- ion Battery	4	8.56	28.52	34.24
MPU 6050	1	10.00	8.33	10
Pololu Servo wheels (pair)	1	5.32	4.43	5.32
Mounting Hub (Pair)	1	5.50	4.58	5.50
Total				£146.86

Chapter 6: Discussion of Results

The research done in the literature review outlined that there is a complexity in tuning the PID gains, where [7] used a mathematically complex D-decomposition technique to calculate the gains of the controller. Additionally, due to the complexity of controller multiple actuators for multiple states, which is the case for two-wheel self-balancing robots, multiple research papers designed a system which cascaded multiple controllers, each controlling a certain parameter of the system.

To tackle the tuning issues of PID controls, the first part of chapter 4, focuses on weighting of the Q and R matrices based on the system response. An initial Q matrix and R matrix are applied to the system as a starting point of the simulation, these parameters have a response time of 2500s. The Q matrix was then modified based on the states of the system, by increasing the weight of the parameters which are desirable to achieve stability quickly. After the Q matrix was adjusted, the response time of the system dropped to 0.1s. Then the effects of varying the R matrix were studied, first by increasing the values of the R matrix, this slowed down the response time to 0.2s as now the energy consumed by the system is prioritized more. Finally, the values of the R-matrix were decreased significantly, and the response time was simulated to be 0.06s. This simulation demonstrated the versatility of an LQR controller as it is very easy to manipulate the tuning of the controller to achieve the desired results. When the values of the Q-matrix were increased, the response time of the system is prioritized, and the values of the k gain matrix increase, this causes the response time of the system to decrease. For applications in which the energy of the system is a priority, the values of the R-matrix are increased, this reduces the values of the gain matrix values and the response time increases. However, due to the fact that the LQR control computation follow Lyapunov's stability criteria discussed in [11], the controller always achieves stability.

The second simulation focuses on testing and comparing the robustness of both a PID and LQR controllers. In the literature review section, it was pointed out that since the PID gains are designed based on the transfer function of the system, PID controlled systems are not robust, researcher experimented multiple strategies to improve the robustness of PIDs such as adding adaptive layers to the controller or utilizing fuzzy logic controllers to adapt the gains based on the system.

In the robustness test simulation, the external disturbance for both control strategies was incremented to find the maximum force that the system can withstand. Both controls had to control the position and tilt angle of the robot. The PID failed in controlling the position after a force of 200 was applied and failed in controlling the tilt angle after a force of 800 was applied. As for the LQR, the system failed in controlling both the position and tilt angle after a force of 2550 was applied. This simulation demonstrated that LQR is much more superior in terms of robustness to a PID controller.

Chapter 7: Future work

Upon reflection of the original plan of the project, the implementation section was not completed due to delays in the components ordered and the manufacturing of the chassis as stated in chapter 5. The plan for the future work is to acquire the chassis of the robot and to build it based on the system design provided in figure 5.1. Additionally, I would implement the DC-DC converters designed in the implementation section, and translate the code used for the MATLAB simulation to C++ so it can be used to program the ESP32.

To test the robot, I would implement the same ideology carried out in chapter 4, in which I would vary the weighting matrix to find the most desirable Q and R matrices. I would then add weights to the top shelf of the robot which can be seen in the 3D model of the robot in figure 3.1 to change the COG, this would be done to test the system's robustness to external interference.

Conclusion

The objectives of this project were to design, simulate and implement a two-wheeled self-balancing robot. Chapter 2 of this paper was the literature review section which investigated different control strategies applied to robotics. This was followed by the mathematical modelling of the system, in which the non-linear mechanical equations were produced, these equations were then linearised to produce the state-space matrix. Using the motor circuit, the torques from the motors were written in terms of voltage, so that the output of the state-space matrix is the voltage which is fed to the motor. Chapter 4 discussed the simulation of the system. In the first part of the simulation chapter, robot was tested on MATLAB to define the weighting matrix parameters, based on the desired system response, which demonstrated the ease with which LQR controllers can be tuned. The second section of chapter 4 tested the robustness of both LQR and PID controller, the results from this simulation showed that the LQR controller was still able to stabilise the system even for a high external disturbance, whereas a PID controller failed for a significantly smaller disturbance, based on the literature review PID controller need to be cascaded with an adaptive element in order to improve the robustness of the system. Chapter 5 discusses the implementation of the robot, in this chapter the system design is provided, the components selected are described, the interfacing of the gyroscope with the ESP32 is explained, the DC-DC converters used to connect the battery to the ESP32 and the motors are designed, and the cost breakdown is provided and shown to be lower than the project budget. However, the complete robot could not be implemented due to delays in components and manufacturing. Finally, chapter 7 provides the plan work, which outlines how the complete robot will be implemented and tested.

References

- [1] Juang, H.-S., & Lurr, K.-Y. (2013). Design and control of a two-wheel self-balancing robot using the Arduino microcontroller board. *2013 10th IEEE International Conference on Control and Automation (ICCA)*.
<https://doi.org/10.1109/icca.2013.6565146>
- [2] Zhang, J., Li, G., Liu, F., & Liu, Y. (2016). Design of a two-wheeled self-balance personal transportation robot. *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*. <https://doi.org/10.1109/iciea.2016.7603583>
- [3] Jayakody, D. P. V. J., & Sucharitharathna, K. P. G. C. (2019). Control Unit for a two-wheel self-balancing robot. *Global Journal of Researches in Engineering*, 7–12.
<https://doi.org/10.34257/gjrevol19is1pg7>
- [4] Majczak, M., & Wawrzynski, P. (2015). Comparison of two efficient control strategies for two-wheeled balancing robot. *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*.
<https://doi.org/10.1109/mmar.2015.7283968>
- [5] E. Philip and S. Golluri, "Implementation of an autonomous self-balancing robot using cascaded PID strategy," *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, 2020.
- [6] P. M. Meshram and R. G. Kanojiya, "Tuning of PID controller using Ziegler-Nichols method for speed control of DC motor," *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, 2012, pp. 117-122.
- [7] R. Nalepa, K. Najdek, K. Wróbel, and K. Szabat, "Application of D-decomposition technique to selection of controller parameters for a two-mass drive system," *Energies*, vol. 13, no. 24, p. 6614, 2020.
- [8] A. Wasif, D. Raza, W. Rasheed, Z. Farooq and S. Q. Ali, "Design and implementation of a two wheel self balancing robot with a two level adaptive control," *Eighth International Conference on Digital Information Management (ICDIM 2013)*, 2013, pp. 187-193, doi: 10.1109/ICDIM.2013.6694021.
- [9] Junfeng Wu and Wanying Zhang, "Design of fuzzy logic controller for two-wheeled self-balancing robot," *Proceedings of 2011 6th International Forum on Strategic Technology*, 2011.
- [10] T. A. Mai, D. N. Anisimov, T. S. Dang, and V. N. Dinh, "Development of a microcontroller-based adaptive fuzzy controller for a two-wheeled self-balancing robot," *Microsystem Technologies*, vol. 24, no. 9, pp. 3677–3687, 2018.
- [11] Y. Li, Y. Q. Chen, and I. Podlubny, "Stability of fractional-order nonlinear dynamic systems: Lyapunov direct method and generalized Mittag–Leffler stability," *Computers & Mathematics with Applications*, vol. 59, no. 5, pp. 1810–1821, 2010.

- [12] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, “Lyapunov-stable neural-network control,” *Robotics: Science and Systems XVII*, 2021.
- [13] S. Bagheri, T. Jafarov, L. Freidovich, and N. Sepehri, “Beneficially combining LQR and PID to control longitudinal dynamics of a SmartFly UAV,” *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016.
- [14] E. Okyere, A. Bousbaine, G. T. Poyi, A. K. Joseph, and J. M. Andrade, “LQR controller design for quad-rotor helicopters,” *The Journal of Engineering*, vol. 2019, no. 17, pp. 4003–4007, 2019.
- [15] H. H. Bilgic, M. A. Sen, A. Yapici, H. Yavuz, and M. Kalyoncu, “Meta-heuristic tuning of the LQR weighting matrices using various objective functions on an experimental flexible arm under the effects of disturbance,” *Arabian Journal for Science and Engineering*, vol. 46, no. 8, pp. 7323–7336, 2021.
- [16] Ü. ÖNEN, A. ÇAKAN, and İ. İLHAN, “Performance comparison of optimization algorithms in LQR Controller Design For a Nonlinear System,” *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, vol. 27, no. 3, pp. 1938–1953, 2019.
- [17] Varagnolo, D. and Lucchese, R. R7003E labs: state-space control of a balancing robot. Version 1.2.1. Luleå University of Technology, 2016. [Online]. Available: <https://staff.www.ltu.se/~damvar/Courses/R7003E-2016-LP2/LabManual.pdf>
- [18] “RS Pro, 3.7V, 18650, lithium-ion rechargeable battery, 2.6Ah,” RS. [Online]. Available: <https://uk.rs-online.com/web/p/speciality-size-rechargeable-batteries/8801558>. [Accessed: 05-Sep-2022].
- [19] “Modules: Espressif Systems,” *Modules | Espressif Systems*. [Online]. Available: <https://www.espressif.com/en/products/modules>. [Accessed: 05-Sep-2022].
- [20] “MPU-6000 and MPU-6050 product Specification Revision 3 - TDK.” [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>. [Accessed: 05-Sep-2022].
- [21] “Handson Technology 775 Ball Bearing DC Motor.” [Online]. Available: https://www.handsontec.com/dataspecs/motor_fan/775-Motor-SB.pdf. [Accessed: 05-Sep-2022].