UNIVERSITY OF DA NANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

CAPSTONE PROJECT

MAJOR: MECHATRONICS ENGINEERING

PROJECT TITLE:

DESIGN AND CONTROL OF A SELF-BALANCING ROBOT WITH LEG-WHEEL INTEGRATION

Supervisor:	**Dr. Tran Dinh Son**
Reviewer:	**Dr. Pham Anh Duc**
Student:	**Phan Huu Thang**
Student ID:	**101200242**
Class:	**20CDT1**

Da Nang, 06/2025

SUMMARY

Project name: Design and Control of a Self-Balancing Robot with Leg-Wheel Integration.

Studient: Phan Huu Thang

Studient code: 101200242 Class: 20CDT1

1. Demand for a Two-Wheeled Self-Balancing Robot:

In the era of modern technology, the demand for smart, flexible, and space-saving vehicles is increasing. A two-wheeled self-balancing robot offers an optimal solution for movement across various terrains, supporting exploration tasks or serving research purposes in control systems and dynamic balancing. Developing such a model enables students to gain hands-on experience with automatic control systems, sensors, and mechatronic technologies.

2. Scope of the Research Project:

The project focuses on designing, manufacturing, and controlling a small-scale two-wheeled self-balancing robot capable of maintaining stable balance on a flat surface. The research content integrates mechanical design, electronics, and control programming.

3. Project Implementation:

- Study the operating principles and real-world applications of two-wheeled self-balancing robots.
- Perform kinematic and dynamic calculations of the system.
- Design the mechanical structure including the chassis, motor layout, and wheel placement.
- Design the control system using the LQR (Linear Quadratic Regulator) algorithm.
- Select appropriate sensors, microcontrollers, and motors, and design the electrical circuit.
- Program and fine-tune the LQR controller to ensure the robot can maintain balance.

4. Achieved Results:

The project successfully developed a two-wheeled robot capable of self-balancing using sensors and an LQR control system. This serves as a foundational step in the research process, paving the way for future enhancements such as flexible movement, command-based navigation, and AI integration for obstacle detection and avoidance.

GRADUATION PROJECT ASSIGNMENT

No.	Name	ID	Class	Industry
1	Phan Huu Thang	101200242	20CDT1	Mechatronic Engineering

1. Topic: Design and Control of a Self-Balancing Robot with Leg-Wheel Integration

2. The subject is subject to: ☐ Having signed an agreement on intellectual property for the results of implementation.

3. Initial figures and data:

- Maximum speed.
- The height varies.

4. Content of the explanations and caculations:

No.	Name	Content
1	Phan Huu Thang	- Find out about the project - The mathematical dynamics and dynamics of the model. - Hardware design and experimental construction. - Electronic circuitry and component assembly. - Preparation of demonstrations, slides and drawings on demand.

5. Drawing, graphs (specify types and sizes of drawings):

No.	Name	Content
1	Phan Huu Thang	- 1 A0 General Drawing - 1 A0 Detailed Dimension Drawing - 1 A0 Assembly Detail Drawing - 1 A0 Electrical Circuit Diagram - 1 A0 Algorithm Flowchart Diagram

6. Supervisor: Dr. Tran Dinh Son

7. Project assignment date: 20/03/2025

8. Project completion date: 16/6/2025

Da Nang, May 2025

Head of Mechatronics Department **Supervisor**

Dr. Vo Nhu Thanh Dr. Tran Dinh Son

PREFACE

In the context of the rapidly advancing Industry 4.0 era, the research and application of automatic control systems, smart sensors, and mechatronic technologies are becoming increasingly widespread and essential in both daily life and industrial production. One of the practical and foundational applications in this field is the development of a self-balancing two-wheeled robot.

This type of robot not only serves as an effective platform for testing advanced control algorithms such as LQR, PID, and Kalman Filters, but also provides students with hands-on experience of integrating hardware (mechanical and electronic systems) with software (programming and control) into a complete dynamic system.
From that perspective, our student group has carried out the project **"** Design and Control of a Self-Balancing Robot with Leg-Wheel Integration**"**, aiming to develop a compact model capable of maintaining stable balance, easy to control, and with high potential for future smart feature expansion.

This project is expected not only to apply theoretical knowledge into practice, but also to serve as a steppingstone for further research into automatic control systems, embedded systems, and artificial intelligence. This report presents in detail the process of research, design, fabrication, control, and testing of the robot.

We would like to express our sincere gratitude to Dr.Trần Đình Sơn for his dedicated supervision, guidance, and valuable suggestions throughout the implementation of this project. We also extend our thanks to the lecturers of the Faculty of Mechanical Engineering and the Mechatronics Department for their support and for providing a favorable learning and research environment.

Although we have made every effort to complete the assigned tasks, due to limitations in time, knowledge, and practical experience, there may still be shortcomings in our work. We highly appreciate and welcome all sincere feedback and comments from lecturers and peers to help us improve and develop better products in the future.

Best regards,

Implementation team

Phan Huu Thang

ASSURE

We hereby declare that the entire content of the graduation project titled “Design and Control of a Self-Balancing Robot with Leg-Wheel Integration” is the result of our own research and work, carried out under the dedicated supervision of phD. Trần Đình Sơn.

During the implementation of this project, we have referred to various relevant documents and research studies. All contents, images, and data cited from external sources are clearly listed in the References section.

We take full responsibility for the honesty and accuracy of the content presented in this report. In the event of any errors or violations related to intellectual property rights, we accept full responsibility in accordance with the regulations of our university and applicable laws.

Implementation team

Phan Huu Thang

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION LIST

LQR: Linear Quadratic Regulator
PID: Proportional – Integral – Derivative
WBC: Whole-Body Controller
IMU: Inertial Measurement Unit
MPU6050: Microprocessor Unit 6050
ESP32: ESP32 Microcontroller
PWM: Pulse Width Modulation
ADC: Analog to Digital Converter
DAC: Digital to Analog Converter
SPI: Serial Peripheral Interface
I2C: Inter-Integrated Circuit
UART: Universal Asynchronous Receiver-Transmitter
NB: Negative Big
NS: Negative Small
ZE: Zero
PS: Positive Small
PB: Positive Big

INTRODUCTORY

Purpose of the project:

In the trend of modernization and the development of service robots for daily life and industrial applications, self-balancing robotic systems have become a key research area in control and mechatronics. The two-wheeled self-balancing robot is a nonlinear, unstable system that requires precise control algorithms to maintain balance during movement.

In addition, the ability to change height allows the robot to adapt to a wider range of real-world situations, such as moving over uneven terrain or adjusting its center of gravity for better stability.

Therefore, the team has carried out the project **"** Design and Control of a Self-Balancing Robot with Leg-Wheel Integration**"** to develop a flexible robotic platform capable of stable balancing, dynamic height adjustment, and easy expansion for future applications in research, education, or industry.

Objective of the topic:

- Design and fabricate a two-wheeled robot model capable of self-balancing and adjusting its height.
- Apply the LQR (Linear Quadratic Regulator) control algorithm to maintain the robot's balance during movement.
- Develop a control system using the ESP32 microcontroller, interfacing with the MPU6050 sensor.
- Create a simulation environment and tune the control algorithm using MATLAB.
- Test and evaluate the robot’s balancing capability, height adjustment, and system response in real-world scenarios.

Research method:

- Study the theory of two-wheeled robots, the inverted pendulum model, and the LQR control algorithm.
- Survey existing two-wheeled robot models and common height-adjustment mechanisms.
- Simulate the kinematics and implement LQR control in MATLAB/Simulink.
- Design the 3D mechanical model of the robot using SolidWorks.
- Build the actual robot and integrate hardware components (ESP32, MPU6050, motors, and lifting mechanism).

- Program the control system, collect sensor data, and fine-tune the LQR controller.
- Conduct experiments to evaluate the robot's balancing performance and height-adjustment capability.

CHAPTER 1 : INTRODUCTION TO THE TOPIC

1.1. Overview of the topic

1.1.1. Problem Statement

- Nowadays, with the breakthrough of science and technology, together with the support of computers that have enabled robots to respond with high accuracy, rapid signal processing time increases labor productivity and reduces damage to humans...However, industrial robots show limitations in terms of flexibility, working space, ability to connect with workers, structures, etc. So, to overcome these shortcomings, researchers around the world have introduced and defined mobile robots…

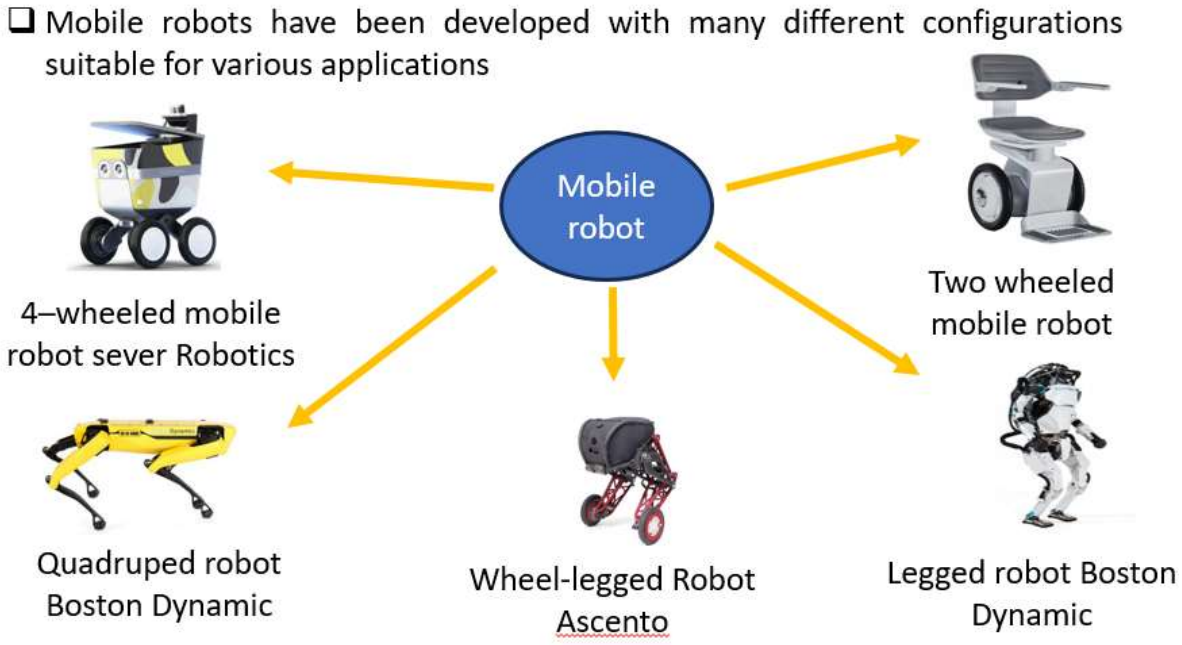


Figure 1.1 Introduction to Mobile Robots

- Mobile robots are one of the fast-growing fields of scientific research. Thanks to their features, mobile robots can support humans in many fields. Applications of mobile robots include patrol, field exploration, military emergency, etc. In the sky, unmanned aircraft systems have demonstrated great engine capabilities, fast but limited in terms of system maintenance power and the amount of loads they can carry, environmental factors are always a challenge. Underground, scientists are still looking for and developing robots capable of moving on complex, rough, fast-navigating terrain. As for

mobile robots moving on the ground, we have the tags divided into two main branches: Robots moving by wheels and robots moving by legs [6]. The advantages of the cane robot are its flexibility in movement, smoothness and efficiency. However, when operating in harsh environments, uneven terrain such as sliding, roughness and rigidity cannot be handled, especially in the case of obstacles larger than the wheel radius. On the other hand, the robot that moves the bladder can move almost the ground, can change the height to suit the task. Besides, these robots require enormous amounts of resources to operate. In order to solve this limitation, hybrid robotics between legs and wheels have been studied around the world. Two-wheeled robots with leg structures that can stretch and shrink. Each leg can be expanded and narrowed smoothly independently, allowing the height to be adjusted to suit the task, or avoid collisions with obstacles... [6]

1.1.1.1. *Research abroad*

- Specifically, in 2019, the research team at the University of Zürich (Switzerland) introduced the LQR controller for forward orbital balance in conjunction with the decentralized whole-body controller (WBC) [7]. In 2022, a team in China successfully applied the PID controller to synchronize the two legs of the robot so that height can be changed [8]. Designing the SR600 robot model and applying a PID controller layering the balance control when the robot's height changes. Subsequently, the team designed the WBC set to allow the robot to keep balance in the state of jumping and flying, while reducing external interference from impact. A stratified PID controller to balance the robot combined with enhanced learning to consolidate the most appropriate rules of control [9]. An optimally adaptive controller-based solution to control parameter conditions can be learned directly from input data and simulated on the Gazebo [10]. The team presented a combination of Feed forward and LQR controller to stabilize the system and apply it to the Hebi Robot. Next, an intelligent adaptive slide controller was proposed to adhere to the orbit for the front, while stable for the robot, which uses a fuzzy base to be able to estimate the dynamics of the robot in real time in Figure 1.1 showing the configurations of the two-wheeled robot with variable heights around the world.

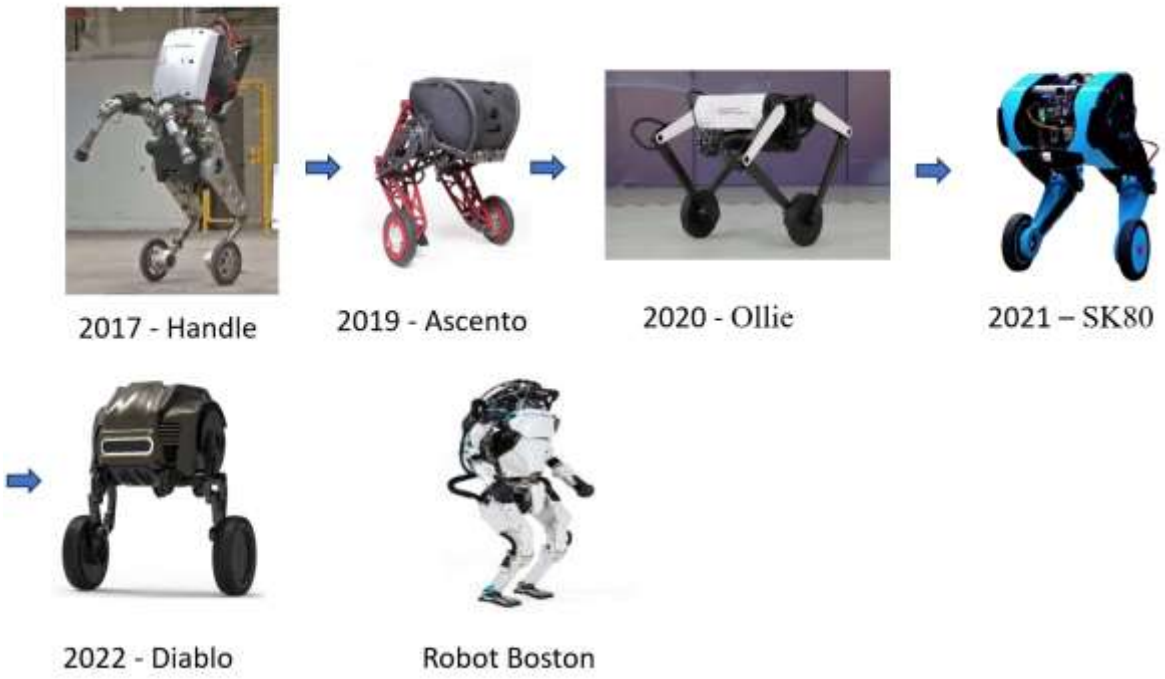


Figure 1.2 Two-wheeled robot model with variable height in the world

- With regard to the configuration of the two-wheeled leg matching robots, today in the world have been and are developing a lot of different kinds of robotic configuration with two wheels-legs can be said to be the Boston Dynamic Robot Handle developed with flexible movements, performing difficult movements such as jumping, squat and carrying heavy loads. The Handle can move at speeds of up to 14.5 km/h and is very well balanced. Next, the Ascento Robot developed by ETH Zurich has a foot configuration that is a type of four-bar bond, improving system strength and reducing mass through an optimized design that can quickly navigate on flat surfaces and overcome obstacles by changing altitude [12]. Next, the hydraulic wheel foot robots WLR and WLLRII developed by the HIT research team have demonstrated the efficiency of hydraulics in changing the height of the robot. WLR robots can sit around, move on complex terrain and carry heavy loads. Next, Robot Ollie of the Tencent Robotics team introduced a two-wheeled robot with a parallel leg structure and a balanced tail right at the body. With this new configuration, the robot is relatively compact, flexible, improved jumping and overcoming obstacles flexibly [13]. Next, the research team at Zhejiang University researched and developed a new structure of a two-legged robot. Each leg of the robot will be designed in the form of four Ascento-like led beams, each leg can be shrunk and stretched by a spiral link, which allows the leverage structure to vary from smooth movement to rotary movement at the joints, while using additional spirals to withstand the gravity affecting the leg joints [14]. The research has

been and is developing structures of two-wheeled robots as well as developing more sustainable control algorithms so that robots can move on complex terrain and change the height of robots in more cases than females.

1.1.1.2. *In-country studies*

- Vietnam has done a lot of research on two-wheel balancing robots using optimum, sustainable controls, including: In 2005, Mai Tan Dao of Ho Chi Minh City Polytechnic University worked on the theme "Self-balancing two-wheeled vehicles moving on flat terrain" by KS Võ Tường Military Guidance [13], the subject had succeeded in approaching from the reverse shake model to the real model of the vehicle, realistic model design, electric circuit board and microcontrol programming, setting up Kalman filter module for angular measurement sensors. In 2010, a team of authors at Ho Chi Minh City Polytechnic University successfully simulated and tested a PID Backstepping controller for a balanced two-wheel system with balance and stability against external impacts [14].

- So far, in 2020, the author of Huang Anh Wu and author of the International University of Hong Bang's Training Creed published an article entitled "Online auto-tuning PID controllers using neural networks for self-balancing two-wheeled vehicles" in the Journal of Science, which provides an explanation of self-adjusting PID parameters based on a neural network for balancing two wheels. Along with the year 2020, the author, Võ Anh Khoa, presented the theme "Trajectory Tracking Pid-Sliding Mode Control for Two-Wheeled Self-Balancing Robot" at the International Conference "Intelligent Computing in Engineering", a paper that presents orbital adhesion using a combined PID sliding controller that has proven its effectiveness when the system moves in orbit number 8.

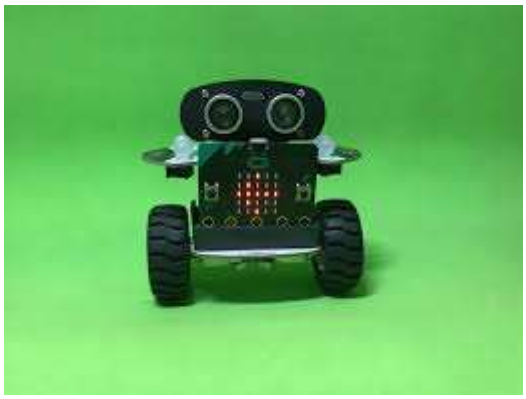


Figure 1.3 Two-wheel balancing robot using PID

- However, we can see that domestic topics so far have stopped in the design of controls for two-wheeled balanced vehicles moving on flat terrain, taking into account that at the moment there is no research team and no company that has designed, manufactured and controlled two wheeled vehicles with legs, the topics above have not been able to solve problems such as moving on uneven terrain or changing the height of robots to pass obstacles larger than the radius of the robot's wheels. Therefore, the opening up of new directions of research is extremely important in order to promote and contribute to the domestic technology. That's why this is the motivation for the team to implement the theme "Model design and balance controller for two-wheeled focus shifting robots".

1.1.2. Research objectives

1.1.2.1. Purpose

- Purpose: Design a two-wheeled robot model with a variable height, building a balancing controller for a robot whose height varies with the change of focus time.

1.1.1.2. *Goal*

- The goal of talent is as follows:
  + Analysis, calculation of the dynamics of the two-wheeled robot.
  + Two-wheeled robot model design with variable height
  + Build balancing control tactics and advanced controls for two-wheeled variable height bobots, applied on simulations and experimental models.
  + Build a user interface and wireless data collection system to monitor, evaluate the quality and performance of the controller.

1.1.3. Scope of study

+ Within the framework of this topic only consider and study
+ The hardware of the robot uses non-specialized electrical equipment, so the responsiveness exists many limitations.
+ The robot is only able to keep balance and move according to the set signal, not taking into account the load factor.
+ Maximum speed of movement of the robot is 0.2m/s, variable height of robot is from 100mm to 220mm.
+ The selection of parameters for the controller is based on objective assessment and test method. As a result, the system has not yet achieved optimal quality.

1.1.4. Research content

This chapter presents the urgency, subject and scope of the study, together with the purpose of the topic. Next, the theoretical foundations of the two-wheeled robot with variable height and the knowledge used to implement the topic are presented in chapter 2. Chapter 4 will present the detailed design work of the robot model using Solidworks 2020 software, as well as the modeling work, the design of the PCB circuit to control the robot and the execution of the robotic hardware modeling, the configuration of the transmission data chain in the system and the user interface design to monitor the system. Next, the dynamic and dynamic model of the robot is calculated, the work on the design of the controller is presented in detail in chapter 3. In chapter 4, the simulation results, the practical experiments will be presented in detail. Finally, chapter 5 will go to the conclusion and direction of the topic.

This topic combines the study of theoretical and experimental methods: Theroreticad rerearch: The subject is to undertake a comprehensive study of two-wheeled mobile robots, balancing algorithms, advanced systems, data transmission and influencing factors. At the same time the subject also learns about how to design the user interface so that it is best suited, research and learn the types of magnetic field acceleration sensors...

Experimental research: After successfully re-simulating the system, the team will carry out modeling, construct the operating environment of the robot, carry out data collection via wireless communication, observe the data returned through the user interface, from there to adjust the controller parameters to suit different circumstances. Evaluate the quality of the controller by putting different cases and experiments on the robot.

1.2. Theoretical basis

This chapter presents the theoretical background relevant to the research and implementation of the project. The theories covered include an overview of mobile robots, the structure of mobile robots with differential drive, the algorithms used in the project, the theory behind the sensors employed in the model, and the theory regarding communication protocols between microcontrollers and wireless communication.

1.2.1. Overview of mobile robots

1.2.1.1. The concept of mobile robots

- A mobile robot is a robot that is capable of moving in a working environment without being fixed in any position like many other robots (Figure 1.4). Mobile robots can move automatically, which means they have the ability to navigate in an undefined environment without the need for control devices (Figure 1.5). In addition, mobile robots can also rely on control devices to move controls according to the operator in a defined environment.

Figure 1.4 a) Fanuc arm b) Omro mobile robot

1.2.1.2. The concept of a two-wheeled mobile robot

- The two-wheeled mobile robot model is a two wheeled vehicle with axles vertically placed together (unlike a regular bicycle). On the model, it uses sensors to measure the inclination of the body, the speed of the two wheels and the velocity of the vehicle relative to the ground, the angle of the whole vehicle. In order for the vehicle to be balanced, the focus of the vehicle must be kept between the wheels. We can determine the angle between the plane moving with the wheel and the dimension of gravity. So, instead of trying to determine the focus of the vehicle, we can determine the angle of inclination of the car. If the angle of the car is forward, we can keep it moving forward to keep the car in balance and vice versa.

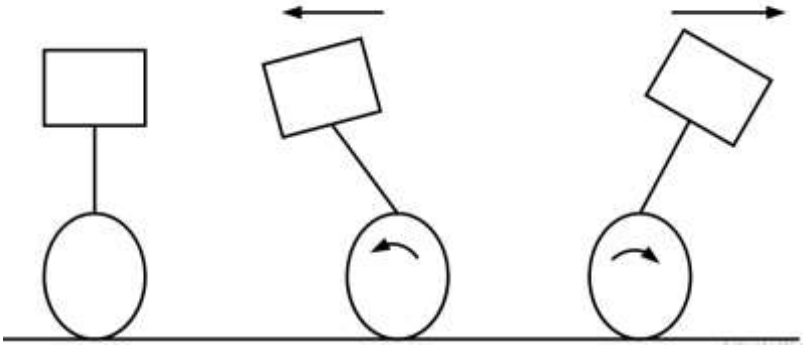


Figure 1.5 Describes how to balance the two-wheeled robot

1.2.2. Two-wheel mobile robotic dynamics

- Robot dynamics deals with the problem of the configuration of a robot in a working space, which is the relationship between the geometric parameters of the robot and the constraints applied to the robot's moving orbit. These dynamic equations always depend on the geometric structure of the robot. The study of kinetics is a prerequisite for the study of mathematics related to the dynamics of mobile robots. The development of new and dedicated dynamic structures remains a subject that attracts a lot of researchers and people's interest in building robots that can perform many tasks that require a high degree of precision and complexity in modern times.

1.2.2.1. Forward Kinematics

- Robotic mathematics tasks: Determine the position and direction of movement of the robot based on information on the velocity and the direction of motion of the wheel or leg of the mobile robot.

+ In mobile robots, variables $q_1, q_2, \dots\dots\dots, q_n$ are general variables in matching spaces $x_1, x_2, \dots\dots\dots, x_m$ are general variables in the workspace. From there we define vectors [24]:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}, p = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{1.1}$$

+ Mathematical dynamics is a mathematical question that finds p when the q variables are pre-known. For the general formula $P \in \Re^m$ và $q \in \Re^n$. We have a connection by nonlinear functions according to the formula below:

$$p = f(q), f(q) = \begin{bmatrix} f_1(q) \\ f_2(q) \\ \vdots \\ f_m(q) \end{bmatrix} \quad (1.2)$$

+ In mobile robots, the change in the movements of the robot in the working space is adjusted through the movement of the joint variables, expressed by $\dot{q} = [\dot{q_1}\ \dot{q_2}\ ...\ \dot{q_n}]$. So we have to find the differential relationship between q and p. This is called polar dynamics and is represented by:

$$dp = Jdq \quad (1.3)$$

In there:

$$dq = \begin{bmatrix} dq_1 \\ dq_2 \\ \vdots \\ dq_n \end{bmatrix}, dp = \begin{bmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_m \end{bmatrix} \quad (1.4)$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \frac{\partial x_1}{\partial q_2} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial x_m}{\partial q_1} & \frac{\partial x_m}{\partial q_2} & \cdots & \frac{\partial x_m}{\partial q_n} \end{bmatrix} \quad (1.5)$$

$\mathbf{J} \in \Re^{m \times n}$ known as the Jacobian matrix, is a matrix that represents the relationship between the velocity of variables in matching space and the speed of variable in working space.

1.2.2.2. Reverse dynamics

Robot reverse mathematics tasks: to calculate the parameters of a robot's mobile structures, including the speed and direction of the wheels or legs of the robot, based on information about the position and direction in which the robot moves. This method calculates these parameters by solving a complex mathematical task, in which the input parameters include the position and direction of the robot's movement, and the output parameters involve the speed and directions of the robotic wheels or legs.

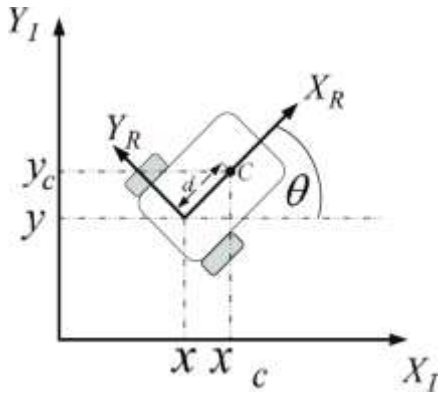


Figure 1.6 Examples of nonholonomic bonding

- Nonholonomic binding: A system or robot with an executive structure smaller than the free level will have nonholonomic constraints. For example, a drone, which has a smaller executive structure than a free ladder, will also be bound in its workspace. An example of nonhonolomic bonding is the wheel. (wheel rolling). Suppose a bicycle wheel is above a certain location (on the ground). Initially, the wheel valve is in a certain position on the wheels. If the wheel moves around and returns to its original position, the valve will not be in its previous position, but in a different position. The new location depends on the distance. If the wheel is honolomic, then the valve must always be right above its original position regardless of how it rolls, so this system is called nonholonomic. Or for an airplane in flight that can't immediately stop in the air or move backwards, if there's an additional enforcement structure, the aircraft can either be fixed up or moving backwards; a plate, a coin rolling but not slipping out.
- Holonomic constraints are structurally and locally bound, and nonhonolomic is velocity constraint, so nonholonomic will relate to functional expressions and are more commonly used in robotic research.

1.2.3. LQR Controller Theory

- LQR is an all-linear optimized controller that is applied to nonlinear system queries. To be applicable, we need to model the system and linearise it around the workpoint.

We have a system after linear:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + D \end{cases} \quad (1.6)$$

The query is set to find the control u(t) so that it satisfies the minimum qualitative criteria of the quality criteria function:

$$J(u)=\frac{1}{2}\int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \quad (1.7)$$

+ In which: Q is the semi-positive matrix and R is the positive matrix Optimal control signal:

$$u(t) = -Kx(t) \tag{1.8}$$

- In there:

$$K(t) = R^{-1}B^{T}P(t) \tag{1.9}$$

In which: P(t) is the positive determining matrix of the Ricatti equation:

$$-\dot{P} = PA + A^{T}P + Q - PBR^{-1}B^{T}P = 0 \tag{1.10}$$

Maximum value of the quality index:

$$J_{min} = x^{T}(0)P_{x}(0) \tag{1.11}$$

We can interpret the equation (1.10) with the tool on MATLAB that is lqr (A, B, Q, R) or dare (A, B, Q, R) for the differential (1.10) equation. For A, B is the linearized matrix from the equation (1.6) and Q, R is the matrix we choose from the Equation (1.10) to find K for the U control signal. (1.8).

1.2.4. Theory of PID Controller

The PID controller is a classical and simple control method, commonly used for nonlinear systems or systems that can be represented by transfer functions. This controller adjusts the control signal based on the error between the desired value and the measured value.

❖ Control Formula:

$$u(t) = K_{p}e(t) + K_{i}\int e(t)dt + K_{d}\frac{de(t)}{dt}$$

In this formula:

$e(t)$ l is the error between the setpoint value $r(t)$ and the measured value $y(t)$.

K_{p}, K_{i}, K_{d} are the coefficients for the proportional, integral, and derivative components, respectively.

Applications:

- PID control is widely applied in closed-loop control systems, such as temperature control, motor speed control, or pressure control, where a detailed model of the system is not required..
- The PID controller can easily adjust its parameters to match the characteristics of the system.

Advantages:

- Easy to implement and adjust.
- Can be applied to many systems without the need for linearization.

Disadvantages:

- Not an optimal controller.
- Performance can be affected in complex nonlinear systems or systems with significant delays.

1.2.5. Magnetic Field Acceleration Sensor Theory

1.2.5.1. Next acceleration

+ The accelerometer functions as a measure of acceleration caused by gravity or motion of an object. In general, the IMU has angles and linear accelerometers. An accelerometer is a suitable accelerometre. For example, an accelerometer still on Earth's surface would measure an appropriate accelerator. Accelerate the angular measurement of objects rotating in space. And the linear accelerometer measures the unattractive acceleration of an object.

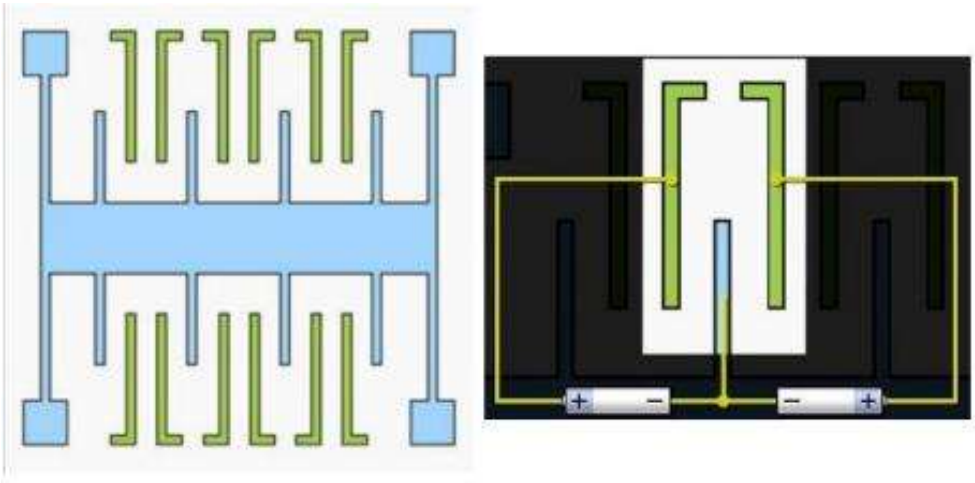


Figure 1.7 How the accelerometer works

+ Let's take an example of how the activity of the retrospectives is shown in Figure 1.7 The moving object measured in the sensor will be a pale blue "flat" in the drawing. The main container is the entire white space, and the "ball" is a light blue object that looks a little bit like a barrel.The torque in the sensor will be the silicon layer that goes along this barrel, sir.If you measure the movement of this vertical silicon layer, you can measure that movement of the sensor.We're looking at a corner of the sensor.Here, you can see that one leg of the central jig and two legs mounted on the container are three parts of a variable condenser.Therefore, when the central bracket moves, electricity is generated.h. By identifying these currents, we can identify the movement of the senses. When measuring the strength of the current, engineers can measure the level of movement of the sensors

1.2.5.2. I'm turning back

- The retrospective is a device used to measure or maintain orientation. When the disk rotates at very high speeds, the external moment shift is minimized, allowing the gyroscope to almost maintain its inclination. This phenomenon is applied to monitoring inclination. The accelerometer can only measure the acceleration of the device, while the retractor can detect the device's direction, the system can easily record both horizontal and vertical movements. The operating sensor is based on the Coriolis effect.

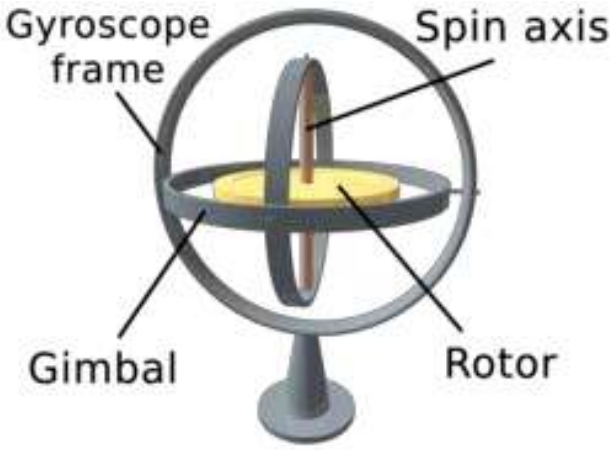


Figure 1.8 I'm turning back.

- The retrospective rotor used in measurement controls are divided into three main groups:
    + Spinning mass gyros or ordinary mechanical gyros.
    + Optical gyros: ring laser, optical fiber
    + Systems Gyro), voltage (piezoelectric)

1.2.5.3. Roll, Pitch, Yaw

- Roll is the angle made by the bot with X-axis
- Pitch is the angle made by the bot with Y-axis
- Yaw is the angle made by the bot with Z-axis

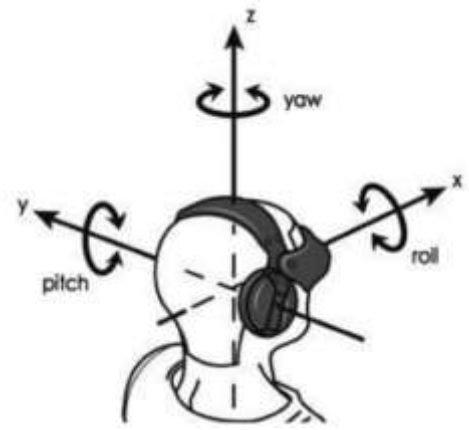


Figure 1.9 Axis Rotation

1.2.5.4. *Understanding Complementary Filter*

- The Mpu6050 for roll, pitch, yaw values. Euler angles are complementary roll, pitch, yaw values. We get the readings of angle from both Gyroscope and Accelerometer. Both sensors can give readings directly. Gyroscope has a problem of Gyroscopic Drift which accumulates over time, so it gives reliable readings for a short period of time. Whereas Accelerometer is jittery and sensitive, but the readings are based on g-vector which has an absolute value, the readings can be relied on for long term.

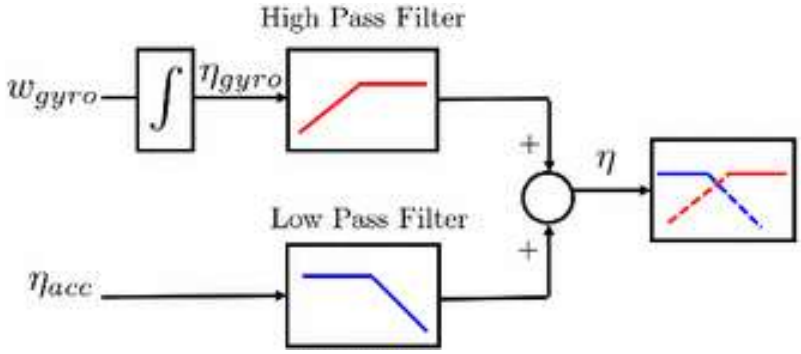


Figure 1.10 Complementary Filter

- To use the benefits of each sensors we do sensor fusion. Idea behind the complementary filter is to take slow moving signals from gyroscope and fast moving signals from an accelerometer and combine them. Accelerometer data is reliable for long term so we apply a "Low pass" filter to it whereas Gyroscope data is reliable in short term so we use "High pass" filter

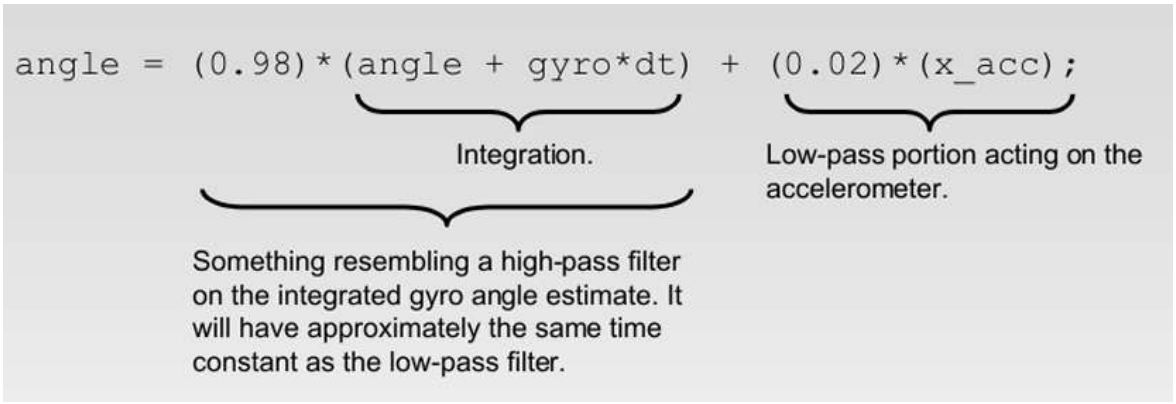


Figure 1.11 Low & High pass filter

CHAPTER 2 : MECHANICAL MODEL DESIGN

❖ **Theoretical basis**

Design Requirements

- Precisely control the posture, balance, and movement of the wheeled robot
- LQR & Fuzzy Controller Design

2.1. Building the Forward Kinematics Problem for TLWR Robot

- Mathematical model of the two-legged self-balancing wheeled robot TL [1].

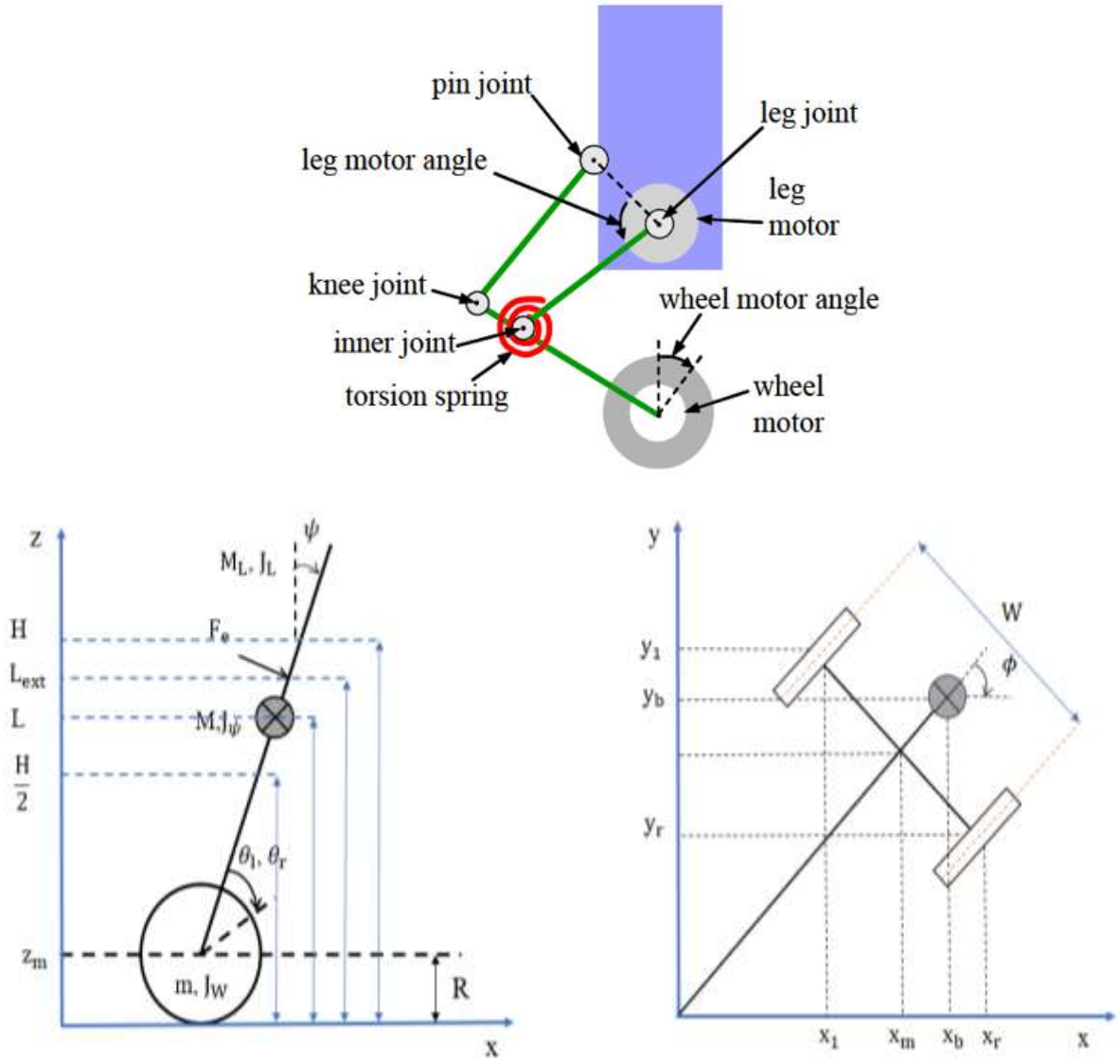


Figure 2.1 Robot components

Table 2.1 Symbols and model parameters for TLWR robots

Paramater	Symbols	Value	Meaning	Units
Robot	W	0.232	Width of the robot	m
	D	0.12	Depth of the robot	m
	H	0,1 → 0,22	Height of the robot	m
	L	0,7*H	Distance from the robot's center of mass to the wheel axis (determined by SolidWorks software)	m
	M	2	Mass of the robot body	kg
	θ		Average angle of the left and right wheels	rad
	ϕ		Rotation angle of the robot	rad
	ψ		Tilt angle of the robot body	rad
	g	9,81	Gravitational acceleration	m/s^2
	J_ψ	$\frac{ML^2}{3}$	Moment of inertia of the robot in pitch	$kg.m^2$
	J_ϕ	$\frac{M(W^2+D^2)}{12}$	Moment of inertia of the robot in yaw	$kg.m^2$
Wheels	R	0.0325	Radius of the robot wheel	m
	m	0.04	Mass of one wheel	kg
	θ_R, θ_L		Rotation angle of the left and right wheels	m
	$V_{L,R}$	12	Power supply for the left and right wheels	V
Motor	n	30	Gear ratio	

	J_m	0.001	Moment of inertia of the DC motor	$kg.m^2$
	J_w	$\frac{mR^2}{2}$	Moment of inertia of the wheel	$kg.m^2$
	K_b	0.22244	EMF constant of the DC motor	Vs/rad
	K_t	0.30896	Torque constant of the DC motor	Nm/A
	$i_{L,R}$		Current through the left and right wheels	A
	R_m	8.884	Resistance of the DC motor	Ω
Friction	f_m	0.01	Friction coefficient between the robot and the DC motor	
	f_w	0.01	Friction coefficient between the wheels and the ground	
	$F_{L,R}$	0.01	Driving torque of the left and right wheel motors	$N.m$

2.1.1. Forward dynamics of a wheeled robot

- We used the Euler-Lagrange method to construct a dynamic model. Suppose at the time of t = 0, the robot moves in the x-axis direction, we have the average straight forward angle of the two wheels and the robot's angle of rotation is defined as follows:

$$\begin{bmatrix}\theta \\ \phi\end{bmatrix} = \begin{bmatrix}\frac{1}{2}(\theta_R + \theta_L) \\ \frac{R}{W}(\theta_R - \theta_L)\end{bmatrix} \qquad \text{Formula -TL [1](2.1)}$$

- There's:

$$v_R = v_m + \frac{W}{2}\dot{\phi} \tag{2.2}$$

$$v_L = v_m - \frac{W}{2}\dot{\phi} \tag{2.3}$$

⇨ From 2.2 and 2.3, together, we get the following formula:

$$v_m = \frac{1}{2}(v_L + v_R) \tag{2.4}$$

❖ Average coordinates of the robot in the projection system

$$\begin{bmatrix}x_m \\ y_m \\ z_m\end{bmatrix} = \begin{bmatrix}\int \dot{x_m} \\ \int \dot{y_m} \\ R\end{bmatrix} \tag{2.5}$$

With: $$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix} = \begin{bmatrix} R\dot{\theta}cos\phi \\ R\dot{\theta}sin\phi \end{bmatrix} = \begin{bmatrix} \frac{R}{2}cos\phi(\dot{\theta}_R + \dot{\theta}_L) \\ \frac{R}{2}sin\phi(\dot{\theta}_R + \dot{\theta}_L) \end{bmatrix}$$

- The robot's left wheel coordinates in the projection system

$$\begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} = \begin{bmatrix} x_m - \frac{W}{2}\sin(\phi) \\ y_m + \frac{W}{2}\cos(\phi) \\ z_m \end{bmatrix} \quad (2.6)$$

- The robot's right wheel coordinates in the projection system

$$\begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} = \begin{bmatrix} x_m + \frac{W}{2}\sin(\phi) \\ y_m - \frac{W}{2}\cos(\phi) \\ z_m \end{bmatrix} \quad (2.7)$$

- Focus coordinates of the robot in the projection system

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} x_m + Lsin\psi cos\phi \\ y_m + Lsin\psi\cos\phi \\ z_m + Lcos\psi \end{bmatrix} \quad (2.8)$$

  - From the formula (2.1) and (2.5) we get the following formula:

$$\begin{aligned} \dot{x}_m &= \frac{R}{2}cos\phi(\dot{\theta}_R + \dot{\theta}_L) \\ \dot{y}_m &= \frac{R}{2}sin\phi(\dot{\theta}_R + \dot{\theta}_L) \\ \dot{\phi} &= \frac{R}{W}(\dot{\theta}_R - \dot{\theta}_L) \end{aligned} \quad (2.9)$$

❖ From the above equations (2.9) we get the following matrix:

$$\begin{bmatrix} \dot{x_m} \\ \dot{y}_m \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}cos\phi & \frac{R}{2}cos\phi \\ \frac{R}{2}sin\phi & \frac{R}{2}sin\phi \\ \frac{R}{W} & -\frac{R}{W} \end{bmatrix} \begin{bmatrix} \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix} \quad (2.10)$$

- In it: R is the radius of the wheel, θ_R, θ_L is the angle of rotation of the left and right cakes, $\dot{\theta}_R$, $\dot{\theta}_L$ is the rotation speed of the right and left cakes, W is the width of the body of the Robot, x_m, y_m, z_m is called the middle position holding the position of the fruit and right cake. Coordinates like x_R, y_R, z_R x_L, y_L, z_L are the coordinates of the robot's left and right wheels.

2.1.2. Reverse Dynamics of the Wheel Part Robot

- Based on the formula (2.5) we infer the average speed of the robot:

$$v_m = \dot{x}_m \cos(\phi) + \dot{y}_m \sin(\phi) \quad (2.11)$$

❖ Replace (2.11) with (2.2), (2.3) we get:

$$\dot{\theta}_R = \frac{\dot{x}_m \cos(\phi) + \dot{y}_m \sin(\phi) + \frac{W}{2}\dot{\phi}}{R}$$

$$\dot{\theta}_L = \frac{\dot{x}_m \cos(\phi) + \dot{y}_m \sin(\phi) - \frac{W}{2}\dot{\phi}}{R}$$

- At the speed of an object v = w*r = $\dot{\theta} * r$

❖ From the equation above we get the following:

$$\begin{bmatrix} \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix} = \frac{1}{R} \begin{bmatrix} cos\phi & sin\phi & \frac{W}{2} \\ cos\phi & sin\phi & -\frac{W}{2} \end{bmatrix} \begin{bmatrix} \dot{x_m} \\ \dot{y_m} \\ \dot{\phi} \end{bmatrix} \quad (2.12)$$

❖ The TLWR two-wheeled robot is a non-homonomic mobile robot which means it has unintegrated motion constraints. This means that not all movements in 3D space are feasible. Specifically, the two-wheeled robot can only move in the direction it is heading, and it can rotate around its axis. It can't move horizontally without turning.

2.1.3. Kinematic of robot leg segment

❖ The TLWR robot was developed with two adjustable components, the body and leg of the robot, which is a promising method to improve the performance of the two-legged robot in terms of its flexibility, transmission speed and energy usage. By adding wheels at the ends of the legs, each leg can be stretched and withdrawn independently by driving the hip motor on the body.

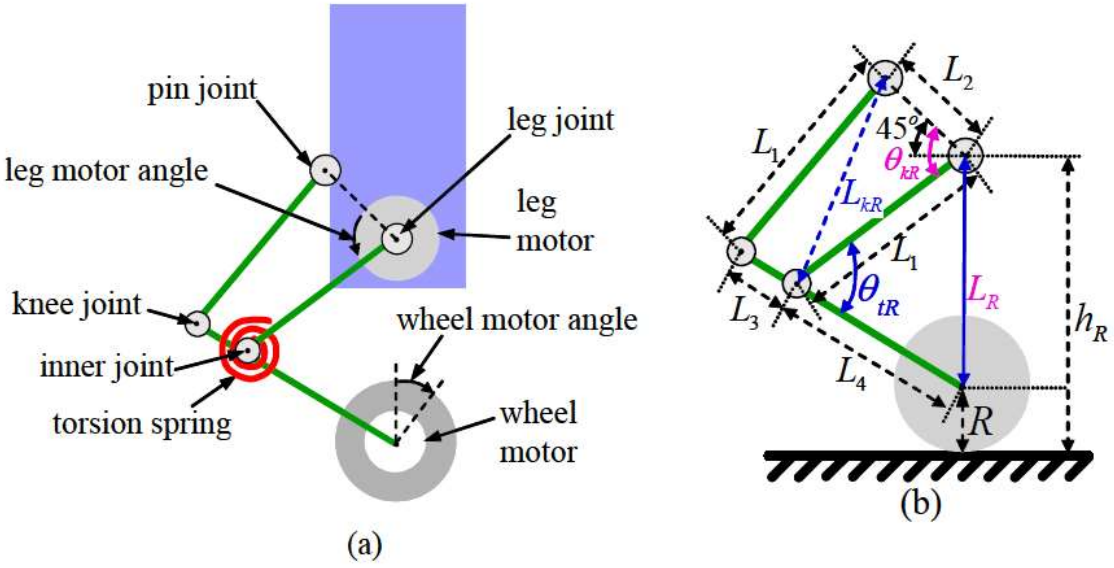


Figure 2.2 (a) Robot components b) Dynamic model of the right leg

Based on Figure 2.2b, we have the following geometric parameters:L_1 is the length between the side joint and the cushion joint, L_2 is the longitude between the sides and

the hips joints, L_3 is the Length between side joints and the torque joints, L_4 is the distance from the junction to the axis of the wheel. The spiral furnace mounted inside the 4 seam has an effect against gravity acting vertically down the robot, while improving the performance of the robot as height changes. The total height that the TLWR can vary from 0.1 m - 0.22 m through the implementation of four-bar linkage.

❖ The robot's leg movements

From Figure 2.2, we can calculate the length of the robot's legs in angles. Here's a mathematical calculation of the right leg of the robot and the left leg.

Step 1: Calculate the length between the side joint and the twist joint L_{kR}:

Based on the geometry of the robot's leg, Cosin's theorem finds the length of the L_{kR}:

$$L_{kR} = \sqrt{({L_1}^2 + {L_2}^2 - 2L_1L_2\cos(\theta_{kR}))} \quad (2.13)$$

Step 2: Calculation of the robot's foot opening angle θ_{tR}:

Applying Cosin's principle, we have:

$$\theta_{tR} = 180^o - \left(cos^{-1}\left(\frac{{L_1}^2 + {L_{kR}}^2 - {L_2}^2}{2L_1L_{kR}}\right) + cos^{-1}\left(\frac{{L_3}^2 + {L_{kR}}^2 - {L_1}^2}{2L_3L_{kR}}\right)\right) \quad (2.14)$$

Leg length L_R:

$$L_R = \sqrt{L_1^2 + L_4^2 - 2L_1L_4\cos(\theta_{tR})}$$

Step 3: Calculate the robot's altitude change:

$$h_R = R + L_R = R + \sqrt{L_1^2 + L_4^2 - 2L_1L_4\cos(\theta_{tR})} \quad (2.15)$$

❖ Robot foot reverse dynamics

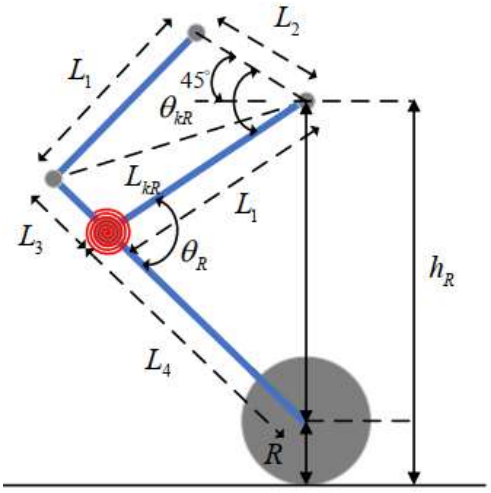


Figure 2.3 Geometric structure of the robot's leg

+ From Figure 2.3, we can calculate the length of the robot's legs in angles. Here's the reverse dynamics of the robot's leg.

Step 1: Angle Calculation θ_R

Apply Cosin's theorem, find the angle between the bars L_1 and L_4.

$$\theta_R = \cos^{-1}\left(\frac{L_1^2+L_4^2-(h_R-R)^2}{2L_1 L_4}\right) \quad (2.16)$$

Step 2: Calculate length L_{kR}

Continuing to apply Cosin's theorem, we draw the length of L_{kR} in L_1 and L_3.

$$L_{kR} = \sqrt{L_1^2 + L_3^2 - 2L_1L_3\cos(180-\theta_R)} \quad (2.17)$$

Step 3: Calculate the robot's hip angle

$$\theta_{kR} = \cos^{-1}\left(\frac{{L_2}^2+{L_{kR}}^2-{L_1}^2}{2L_2L_{kR}}\right) + \cos^{-1}\left(\frac{{L_1}^2+{L_{kR}}^2-{L_3}^2}{2L_1L_{kR}}\right) \quad (2.18)$$

❖ From the above analysis we can determine the height of the robot when it moves on an uneven surface and when it jumps down the floor [1]:

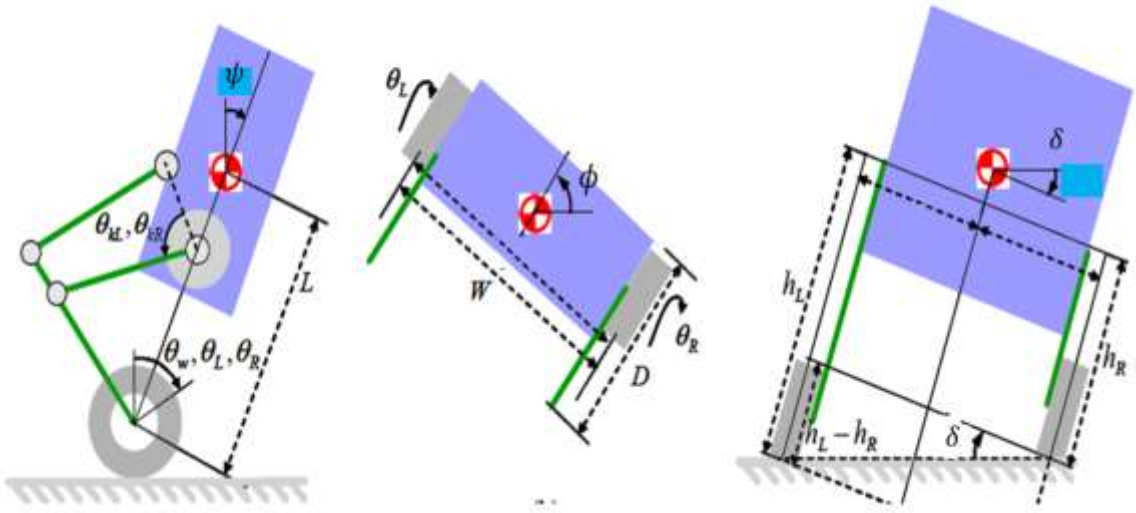


Figure 2.4 Robot dynamics when two legs move unevenly

Note: δ: $pitch, \psi$: $roll, \phi$: yaw

- We have H height defined.:

$$H = 1/2(h_L + h_R)$$

2.2. Dynamic calculations for robot

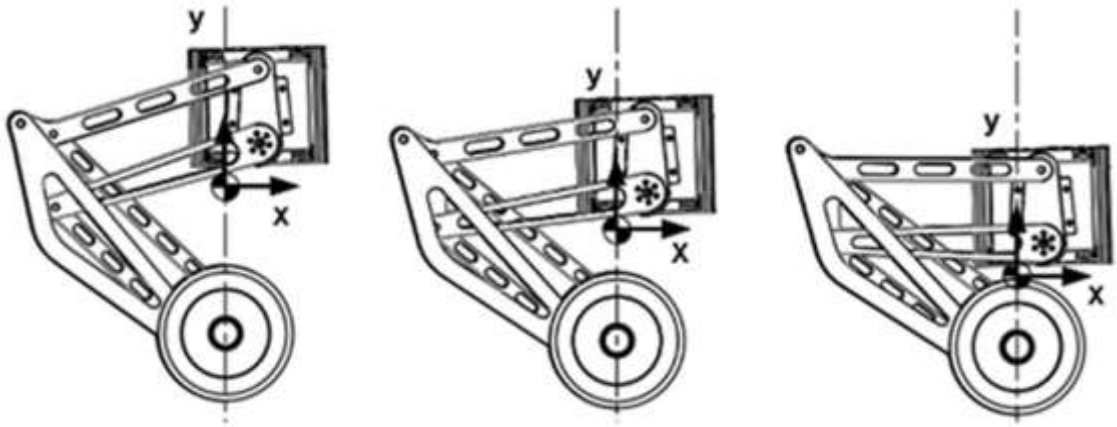


Figure 2.5 Description of robotic dynamics when changing height and focus

- As the height of the robot changes, the robot's dynamics dramatically change the focus. Here, the dynamics are at fixed heights such as the robot's lowest, average and highest height. The purpose is to make the focus of the robot pass through the middle axis of the wheel.

- Steps to analyze the dynamics of two-wheel balancing robots

Step 1: Find dynamic energy and power of the system (formula TL [5])

Dynamic Energy (TL)

$$K(\theta,\dot{\theta}) = \sum_{i=1}^{n}(\frac{1}{2}m_i v_i^2 + \frac{1}{2}\dot{\theta}_i^2 I_i)$$

There's: m_i is the mass of the solid.

v_i is the speed at the center of the object.

$\dot{\theta}_i$ is the angle speed of the solid.

I_i is the inertial moment of the solid to the center block.

The progressive dynamic equation of the system is described as follows

$$K_1 = \frac{1}{2}m\left(\dot{x_R}^2 + \dot{y_R}^2 + \dot{z_R}^2 + \dot{x_L}^2 + \dot{y_L}^2 + \dot{z_L}^2\right) + \frac{1}{2}M\left(\dot{x_b}^2 + \dot{y_b}^2 + \dot{z_b}^2\right) \quad (2.19)$$

System rotary dynamics equation:

$$K_2 = \frac{1}{2}(J_w\left(\dot{\theta_R}^2 + \dot{\theta_L}^2\right) + J_\psi\dot{\psi}^2 + J_\phi\dot{\phi}^2 + n^2J_m\left(\left(\dot{\theta_R} - \dot{\psi}\right)^2 + \left(\dot{\theta_L} - \dot{\psi}\right)^2\right) \quad (2.20)$$

⇨ From equations 2.13 and 2.14 we get the total dynamics of the system:

$$K = K_1 + K_2$$

Potential energy

$$U = \sum_{i=1}^{n}(m_i\, g.z_i + U_{refi})$$

The power equation of the system consists of the two wheels and the focus of the robot.

$$U = mg(z_R + z_L) + Mgz_b$$

Step 2: Apply the Lagrange function to the total dynamics and power of the system is represented as follows:

$$L = K - U$$

Then the differential equation of the corresponding angles:

$$\tau_\theta = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} \quad (2.21)$$

$$\tau_\phi = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} \quad (2.22)$$

$$\tau_\psi = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} \quad (2.23)$$

Step 3: The robot's differential equation in the formula (2.15,16,17) is translated into the general form as follows:

$$M(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right) + G(\theta) + \mathrm{J}F_e = \tau \quad (2.24)$$

There's: $\tau = \left[\tau_\theta\ \tau_\psi\ \tau_\phi\right]^T$ is the input moment of the system

$M(\theta)$ It's an inertial matrix.

$C\left(\theta, \dot{\theta}\right)$ is the Coriolis effect matrix.

$G(\theta)$ is the vector of the gravity field.

J is the Jacobia matrix of the system at this point.

$\mathrm{F_e} = \left[F_x\ F_y\ F_z\right]^T$ is an external force that affects the system.

- Identify inertial matrix $M(\theta)$

$$M(\theta) = \frac{\partial \tau}{\partial \ddot{\theta}} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

$$M(\theta) = \begin{bmatrix} 2J_w + 2J_m n^2 + R^2(M+2m) & LMRcos\psi - 2J_m n^2 & 0 \\ LMRcos\psi - 2J_m n^2 & J_\psi + L^2 M + 2J_m n^2 & 0 \\ 0 & 0 & J_\phi + \frac{W^2 m}{2} + L^2 M sin(\psi)^2 + \frac{W^2(J_w + J_m n^2)}{2R^2} \end{bmatrix}$$

- Identify Coriolis effect matrix

$$C\left(\theta, \dot{\theta}\right) = \sum_{k=1}^{n} \Gamma_{ijk}(\theta)\ \dot{\theta}_k$$

There's:

$$\Gamma_{ijk}(\theta) = \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial \theta_k} + \frac{\partial M_{ik}}{\partial \theta_j} - \frac{\partial M_{jk}}{\partial \theta_i}\right)$$

With: corresponds to the variables of the system i, j, k

$$C(\theta,\dot{\theta}) = \begin{bmatrix} 0 & -LMR\dot{\psi}\sin(\psi) & 0 \\ 0 & 0 & -\dfrac{L^2M\dot{\phi}\sin(2\psi)}{2} \\ 0 & \dfrac{L^2M\dot{\phi}\sin(2\psi)}{2} & \dfrac{L^2M\dot{\psi}\sin(2\psi)}{2} \end{bmatrix}$$

+ To test the properties of the $C(\theta,\dot{\theta})$ matrix, we use the property of $\dot{M}(\theta) - 2C(\theta,\dot{\theta})$ as a symmetrical matrix. TL [4].

With $$\dot{M}(\theta) = \begin{bmatrix} 0 & -MLRsin(\psi)\dot{\psi} & 0 \\ -MLRsin(\psi)\dot{\psi} & 0 & 0 \\ 0 & 0 & ML^2\sin(2\psi)\,\dot{\psi} \end{bmatrix}$$

$$-(\dot{M} - 2C) = \begin{bmatrix} 0 & MLR\dot{\psi}sin\psi & 0 \\ -\mathrm{MLR}\dot{\psi}\sin\psi & 0 & ML^2\,\phi\sin(2\psi) \\ 0 & -ML^2\,\phi\sin(2\psi) & 0 \end{bmatrix}$$

- So, the Criolis/Centrifugal matrix herbs magnetic properties of the symmetrical matrix.
- Define the gravitational vector G (θ).

$$G(\theta) = \begin{bmatrix} 0 \\ -LMgsin(\psi) \\ 0 \end{bmatrix}$$

- Jacobian matrix J

+ We have $J^T F_{ext}$ (2.17) based on the principle that the force F_e on the upper point of the robot's symmetrical focal coordinates under the following conditions:

$$0 < L_{ext} \leq H$$

So the robot's focal coordinate equation is rewritten as follows:

$$\begin{bmatrix} x_{ext} \\ y_{ext} \\ z_{ext} \end{bmatrix} = \begin{bmatrix} \int R\dot{\theta}cos\phi + L_{ext}sin\psi cos\phi \\ \int R\dot{\theta}sin\phi + L_{ext}sin\psi\cos\phi \\ R + L_{ext}cos\psi \end{bmatrix} \quad (2.25)$$

Taking a separate function from the formula (4.18) we get

$$J = \begin{bmatrix} Rcos\phi & L_{ext}cos\psi cos\phi & -L_{ext}sin\psi sin\phi \\ Rsin\phi & L_{ext}cos\psi sin\phi & L_{ext}sin\psi cos\phi \\ 0 & -L_{ext}sin\psi & 0 \end{bmatrix}$$

Shift matrix J^T

$$J^T = \begin{bmatrix} Rcos\phi & Rsin\phi & 0 \\ L_{ext}cos\psi cos\phi & L_{ext}cos\psi sin\phi & -L_{ext}sin\psi \\ -L_{ext}sin\psi sin\phi & L_{ext}sin\psi cos\phi & 0 \end{bmatrix}$$

So $J^T F_e$ has the following formula:

$$J^T F_e = \begin{bmatrix} R(F_x cos\phi + F_y sin\phi) \\ L_{ext}(F_x cos\phi cos\psi + F_y sin\phi cos\psi - F_z sin\psi) \\ L_{ext}sin\psi(F_y cos\phi - F_x sin\phi) \end{bmatrix} \quad (2.26)$$

- Dynamic momentum generated by the engine TL [3]

$$\begin{bmatrix} \tau_\theta \\ \tau_\psi \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} F_L + F_R \\ F_\psi \\ \frac{W}{2R}(F_L - F_R) \end{bmatrix} \quad (2.27)$$

$There's$:

$$F_L = nK_t i_L + f_m(\dot{\psi} - \dot{\theta}_L) - f_w\dot{\theta}_L \quad (2.28)$$

$$F_R = nK_t i_R + f_m(\dot{\psi} - \dot{\theta}_R) - f_w\dot{\theta}_R \quad (2.29)$$

$$F_\psi = -nK_t i_L - nK_t i_R - f_m(\dot{\psi} - \dot{\theta}_L) - f_m(\dot{\psi} - \dot{\theta}_R) \quad (2.30)$$

Using the PWM method to control the engine should switch from current to engine voltage:

$$L_m i_{L,R} = v_{L,R} + K_b(\dot{\psi} - \dot{\theta}_{L,R}) - R_m i_{L,R}$$

- This is a relatively small response (by zero), which can be ignored, and the following formula can be deduced:

$$i_{L,R} = \frac{v_{L,R} + K_b(\dot{\psi} - \dot{\theta}_{L,R}) - R_m}{R_m} \quad (2.31)$$

=> From the equation (2.21-2.24) and instead (2.20) we get the following equation:

$$\begin{bmatrix} \tau_\theta \\ \tau_\psi \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} \alpha(v_L - v_R) - 2(\beta + f_w)\dot{\theta} + 2\beta\dot{\psi} \\ -\alpha(v_L + v_R) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \\ \frac{W}{2R}\alpha(v_L - v_R) - \frac{W^2}{2R^2}(\beta + f_w)\dot{\phi} \end{bmatrix} \quad (2.32)$$

. With $\alpha = \frac{nK_t}{R_m}$, $\beta = \frac{nK_t K_b}{R_m} + f_m$

⇨ From the equations instead (2.24) without considering the force of influence we have (2.33) the following:

$$\begin{cases} \left(2J_w + 2J_m n^2 + R^2(M+2m)\right)\ddot{\theta} + (LMRcos\psi - 2J_m n^2)\ddot{\psi} - MLR\dot{\psi}^2 sin\psi = \\ \alpha(v_L - v_R) - 2(\beta + f_w)\dot{\theta} + 2\beta\dot{\psi} \quad (*) \\ (LMRcos\psi - 2J_m n^2)\ddot{\theta} + \left(J_\psi + L^2M + 2J_m n^2\right)\ddot{\psi} - MgLsin\psi - ML^2\dot{\phi}^2 sin\psi cos\psi = \\ -\alpha(v_L + v_R) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \quad (**) \\ \left(J_\phi + \frac{W^2 m}{2} + \frac{W^2(J_w + J_m n^2)}{2R^2} + ML^2 \sin^2\psi\right)\ddot{\phi} + 2ML^2\dot{\psi}\dot{\phi} sin\psi cos\psi = \\ \frac{W}{2R}\alpha(v_L - v_R) - \frac{W^2}{2R^2}(\beta + f_w)\dot{\phi} \quad (****) \end{cases}$$

2.3. Engine selection

2.3.1. Selection of wheel drive motor

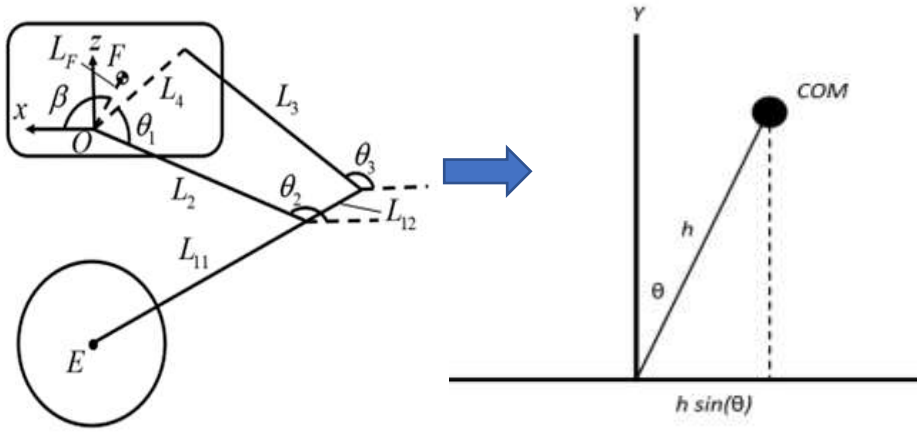


Figure 2.6 Model for the center of mass analysis of the robot.

According to the theory: $L_4 = L_{12} = 0.05\ m,\ L_2 = L_3 = L_{11} =$ 0.1 m

- The load mass applied to the robot: 1.5kg
- The robot's load capacity is 2 kg.
- The maximum opening angle of the robot is: $\theta_{max} = 5^{\circ}$

The height from the wheel axis to the center of mass of the robot:

$$L_{Com} = \sqrt{(x_E - x_F)^2 + (y_E - y_F)^2}$$

Since the robot's height changes, the center of mass height will also vary with the robot's height. In this case, the center of mass height will fluctuate. To calculate and select the motor, the maximum center of mass height, which is approximately equal to the robot's actual height, will be used for easier calculations and motor selection

$$\tau_{max} = (m+M) * g * L_{com} * \sin(\theta_{max}) * \varepsilon_{hs} = 3.5 * 9.8 * 0.2 * \sin(5^0) * 1.5$$
$$= 0.896 \text{ N.m}$$

- Since the load applied to the robot is evenly distributed, the torque of the two motors will be determined as follows:

$$\tau_{left} = \tau_{right} = \frac{\tau_{max}}{2} = 0.4484\ N.m$$

Requirements for selecting motors for the legs:

+ Operating speed of 0.2 m/s
+ Motor integrated with an encoder for pulse reading.
+ Provides a suitable velocity for the robot

⇨ To meet the above requirements and theoretical calculations, the team decided to select DC motors for the system. Specifically, two JGB37 333RPM motors were chosen and mounted on the robot's legs (0.4484 N·m < 0.5 N·m).

o Technical Specifications:

Table 2.2: Engine specifications JGB37-520

Paramater	Value	Units
Supply voltage	12	V
Rotation speed	333	RPM
Encoder	11	pluses
Resolution	330	pluses/rev
Torque	3.5	Kg.cm
Maximun torque	5	Kg.cm
Weight	0.1	kg
Rated current	250	mA

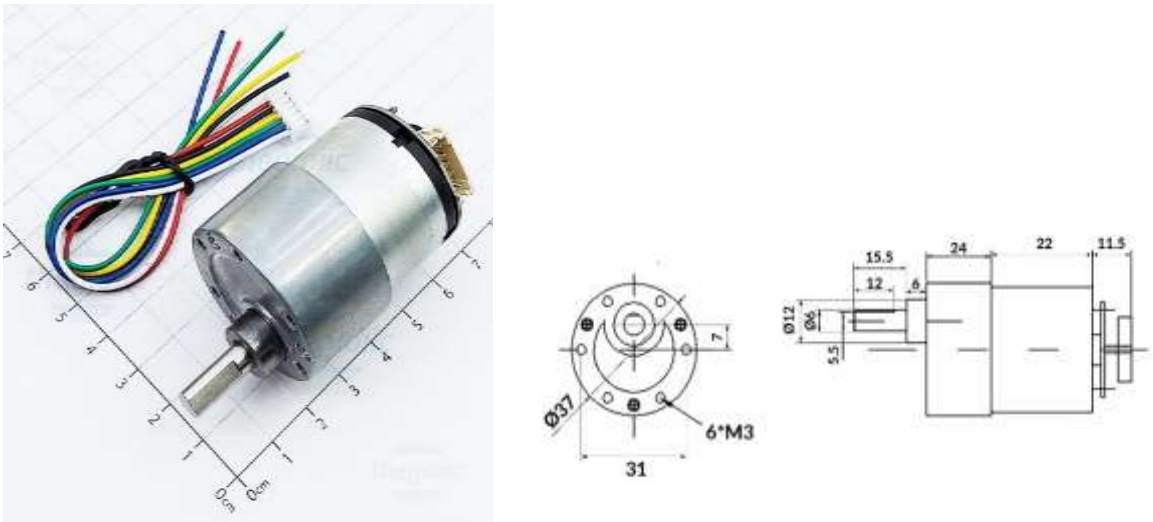


Figure 2.7 JGB Engine 37-520

2.3.2. Selection of hip joint motor

- According to theory:

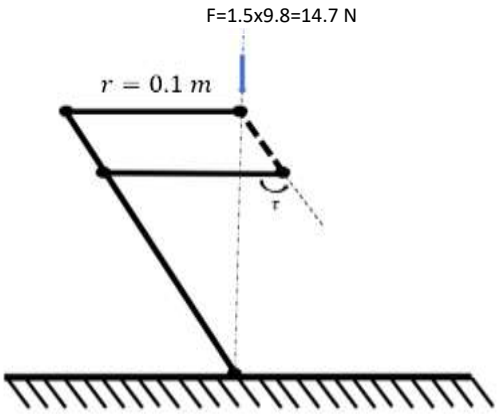


Figure 2.8 Model of the force acting on the servo joint.

$\tau_{servo} = F * \mathrm{r} * 1.5 = 14.7 * 0.1 * 1.5 = 2.205$ N.m

⇨ Based on the above calculations, we choose two servo motors for the robot's hips with a torque greater than 2.205 N.m

- Servo Specification 35 KG

Table 2.3: Engine Specifications Servo SPT5435LV

Parameter	Value	Units
Supply voltage	5-7	V
Rotational speed	27	RPM
Gear ratio	290	
Torque	35	Kg.cm
Maximum torque	70	Kg.cm
Rated current	3	A

Figure 2.9 Servo engine SPT5435LV

2.4. Design a 3D model.

2.4.1. Selecting a design option

❖ Option 1: Design a 5-bar linkage mechanism model.

Hình 2.10 The shape of the robot with Option 1, a 5-bar linkage design

Advantages:

- The vehicle easily finds the center of gravity to achieve balance
- Easier to control.

Disadvantages:

- Uses more motors, so the cost is higher.
- Difficult to jump.

❖ Option 2: Design a 4-bar mechanism model.

Figure 2.11 The shape of the robot with Option 2, a 4-bar mechanism type

Advantages:

- Low cost
- Can jump
- High practical applicability

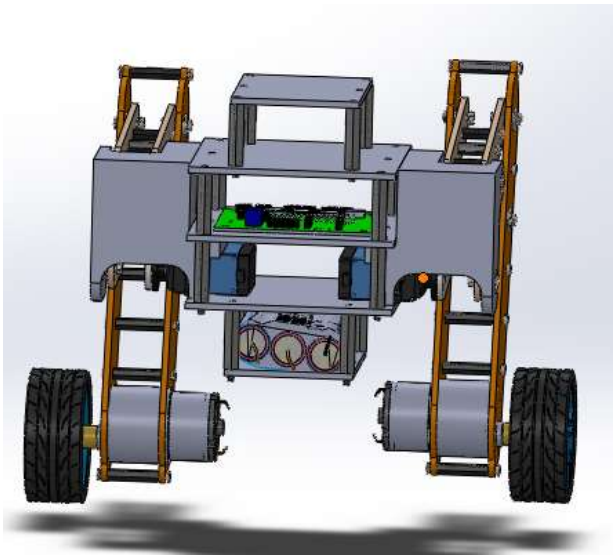
Disadvantages:

- Difficult to maintain balance control

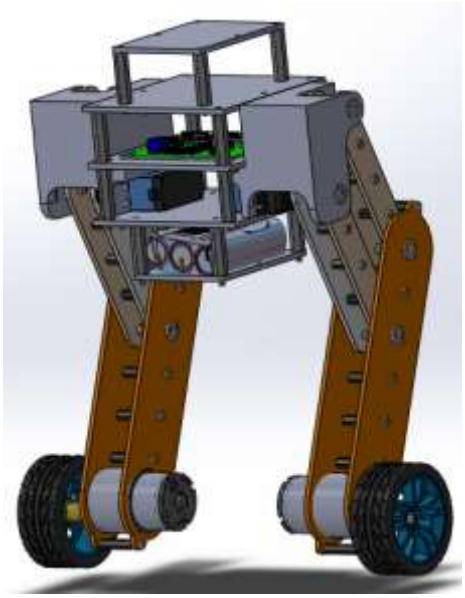
⇨ Based on the analysis above, it can be concluded that Option 2 will be chosen, as it offers advantages in cost, the ability to jump flexibly, and move according to the established criteria and requirements.

2.4.2. Design the 3D model.

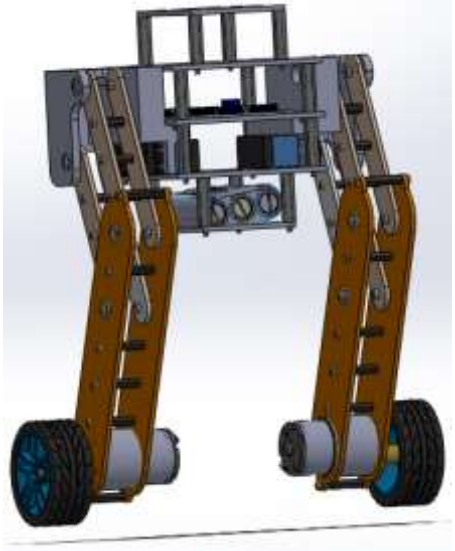
The 3D model of the two-wheeled balancing robot with adjustable height is designed using SolidWorks 2020. To better visualize and observe the robot model, the team presents various perspectives of the robot.



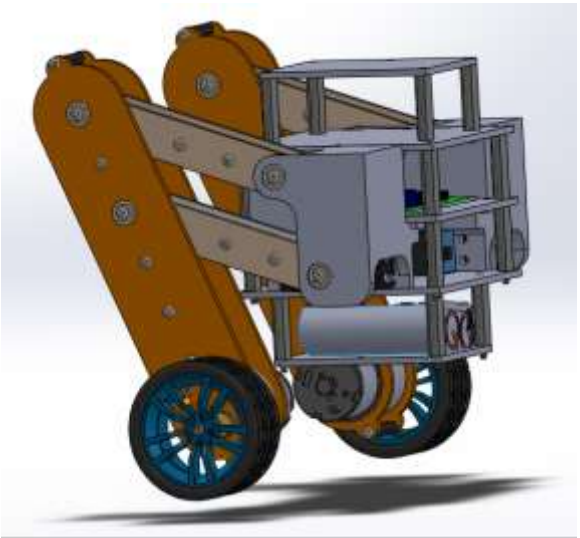
a) Height at low position



b) Maximum reversal of the robot



c) The back of the robot



d) The side of the robot

Figure 2.12 3D model of mobile two-wheeled robot with variable height

2.4.2.1. Robot head frame

- The head frame of the robot is used to fix the details of the body and the position of the hips engine. Thus, the frame design requirement must ensure that the element is sustainable, balanced and stable, while at the same time being light enough throughout the operation of the robot. That's why the team designed robotic frames from microscopic materials. The 3D drawing is presented below.

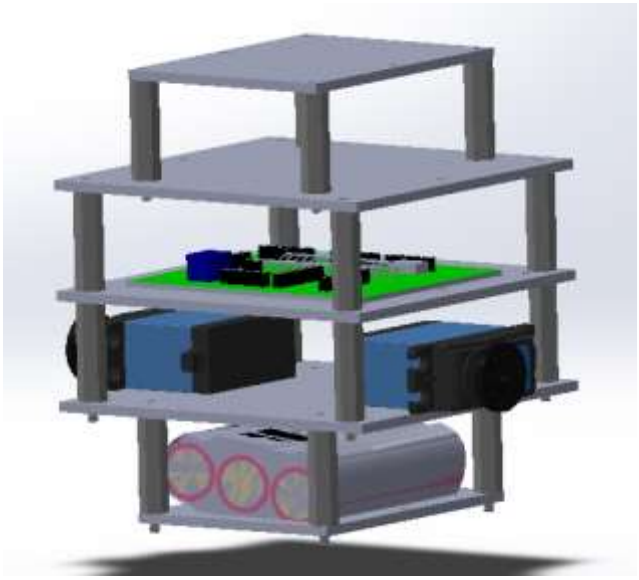


Figure 2.13 3D model of the robot head

2.4.2.2. Robot body parts

- The body part of the robot is used to fix the body part and form a complete block of the body. When the robot is active, the body will function as a link to the frame, while the robot's thighs and legs will move so that the robot can change height. So the design part of the team will be using 3D printed plastics.

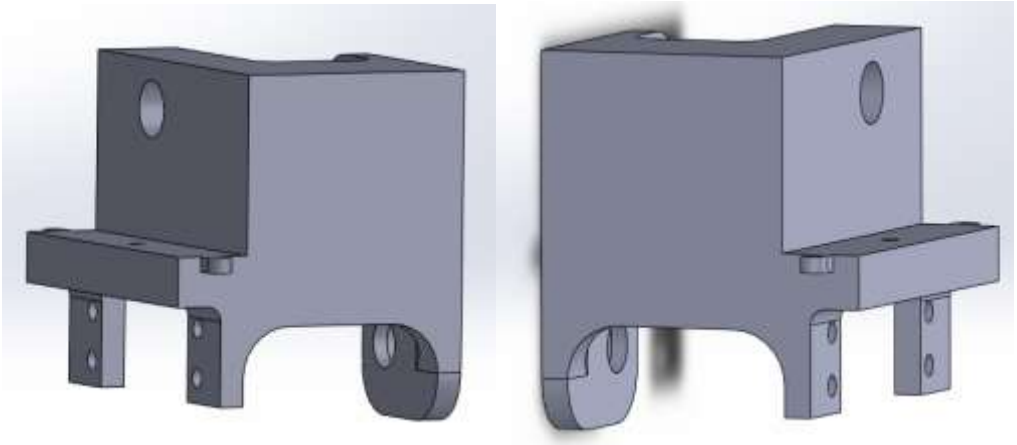


Figure 2.14 Robot body 3D model

2.4.2.3. Robot leg selection

- The robot's leg consists of several bonding bars, including secondary bars and main legs, which are linked to the body of the robot and are made of mica material that increases hardness and durability during operation. The detailed 3D drawing is shown in the picture below.

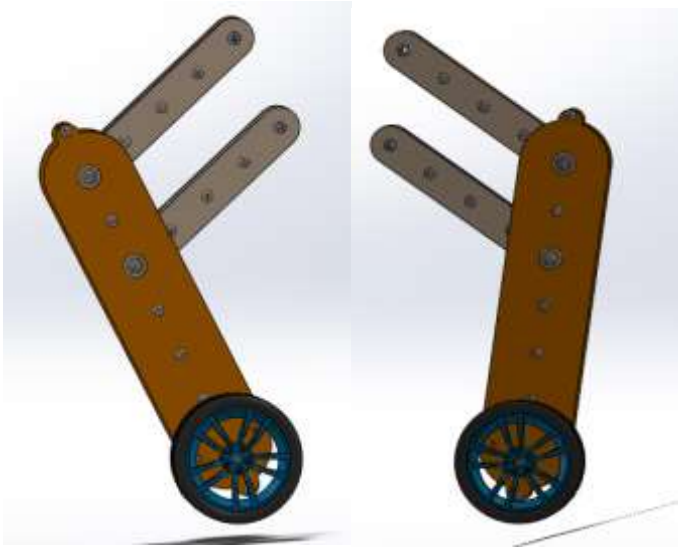


Figure 2.15 Model of robotic legs

2.4.3. Laser cutting machining parts

+ After we test the design of a 3D model, calculation and design, we will convert the 3D file to 2D in CAD format and then edit it and cut it with a laser with high precision with the Mica plastic thickness of a 5 mm bar.

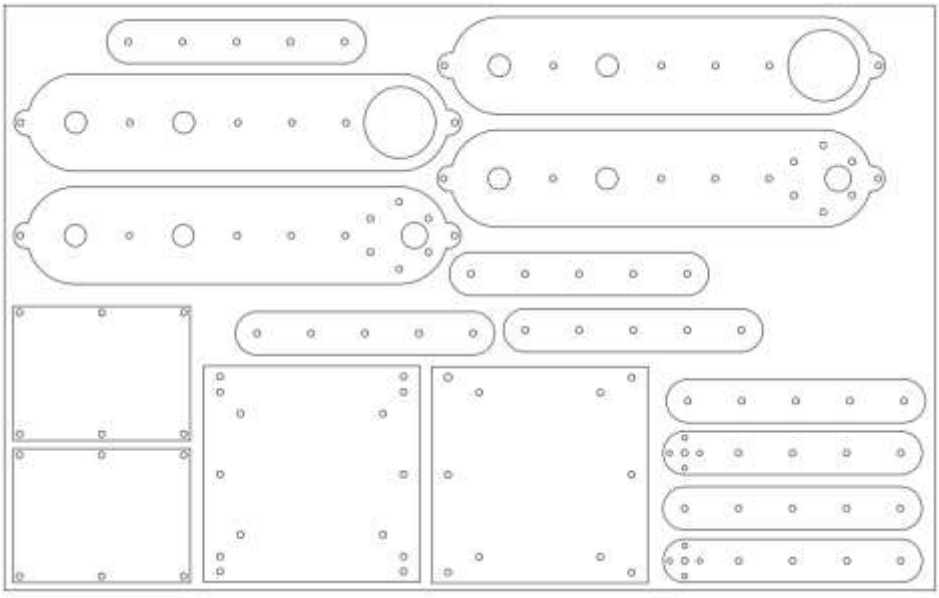


Figure 2.16 2D Drawing of detailed components of the robot

2.4.4. Natural frequency analysis:

2.4.4.1. Theory of natural frequency:

Definition: The natural frequency of a system is the frequency at which the system tends to oscillate in the absence of any external excitation. During free vibration, structures vibrate at their natural frequencies.

Properties: Natural frequency depends on factors such as mass, stiffness, and the shape of the structure. Each system can have multiple natural frequencies corresponding to different vibration modes.

2.4.4.2. Purpose and importance:

Natural frequency analysis helps to identify the frequencies at which the robot system is most susceptible to resonance. Resonance can lead to intense vibrations, causing damage or reducing the lifespan of the robot. Ensuring that the robot's operating frequency does not coincide or come close to its natural frequencies is crucial.

Using the results from the natural frequency analysis allows our team to assess the stability of the design and make modifications to ensure the durability and performance of the robot's arm.

2.4.4.3. Natural frequency of the Model:

- SAfter applying the materials to each component, we obtained the following results for the model's natural frequencies:

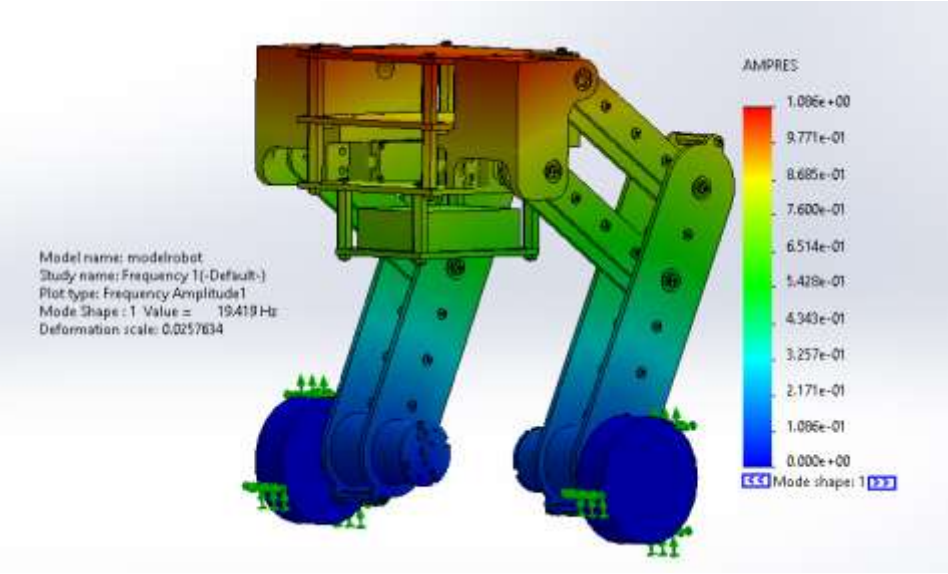


Figure 2.17 Mode Shape 1

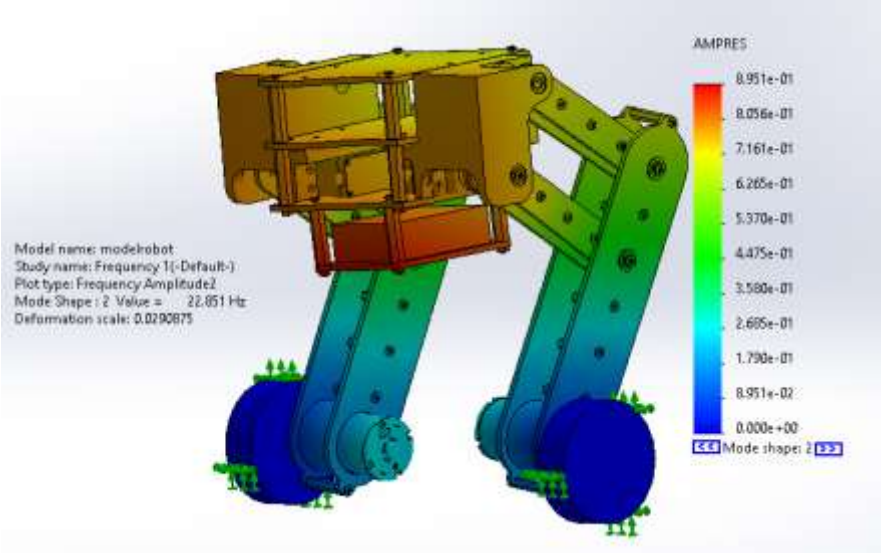


Figure 2.18 Mode Shape 2

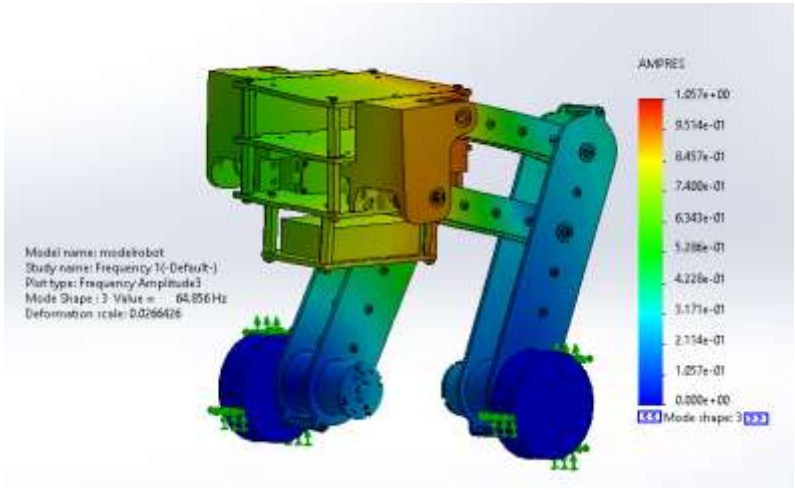


Figure 2.19 Mode Shape 3

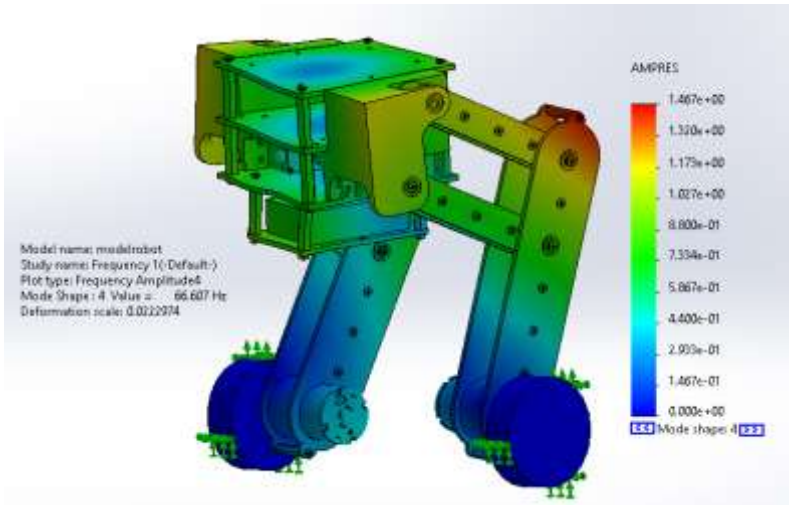


Figure 2.20 Mode Shape 4

Table 2.4: Natural frequencies of the first five mode shapes.

Mode No.	Frequency(Rad/sec)	Frequency(Hertz)	Period(Seconds)
1	122.02	19.419	0.051495
2	143.58	22.851	1.051495
3	407.5	64.856	2.051495
4	418.5	66.607	3.051495
5	474.6	75.535	4.051495

2.4.4.4. Mode shape of the Model

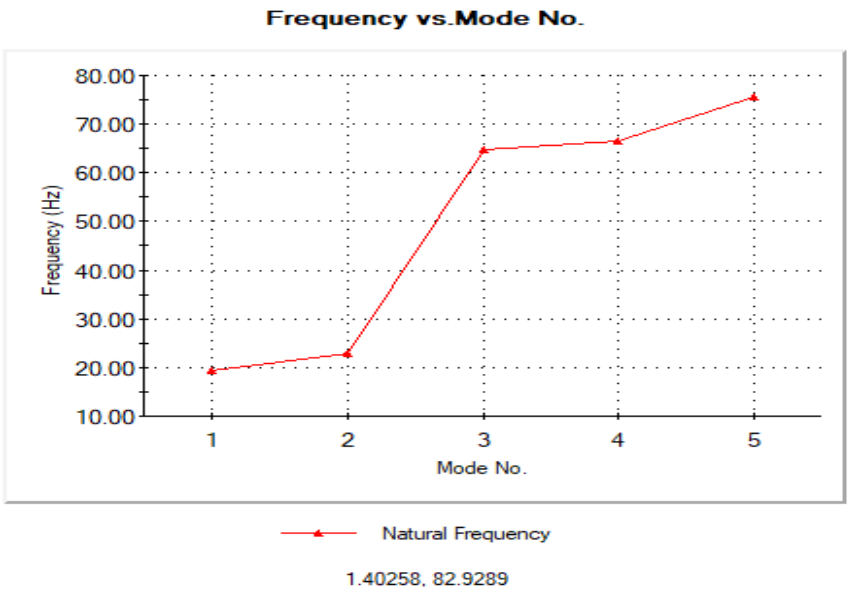


Figure 2.21 Frequency graph at the mode shape positions.

⇨ Conclusion: Based on the graph above, it can be observed that at higher mode shape positions, the vibration frequency increases, indicating that the head portion of the robot experiences more oscillation compared to the other parts.

CHAPTER 3 : CONTROL SYSTEM DESIGN

3.1. Selection of equiment

3.1.1. Execution Requirements

❖ In order for the robot to operate properly, it is necessary to select the equipment that meets the following requirements:

+ The processor is strong enough to verify the control logic.
+ Supply compatible with the operating system.
+ Signal reader sensors must be suitable and restrict interference.
+ Robots vary in height from 100mm to 220mm. The fixed heights of the robot include lowest height: 100mm, average altitude: 160mm, highest heights: 220mm.
+ The speed of the robot is: 0.2 m/s.

3.1.2. Device selection

❖ A block diagram to control the system is shown in Figure 3.1 with the blocks assuming the following functions:

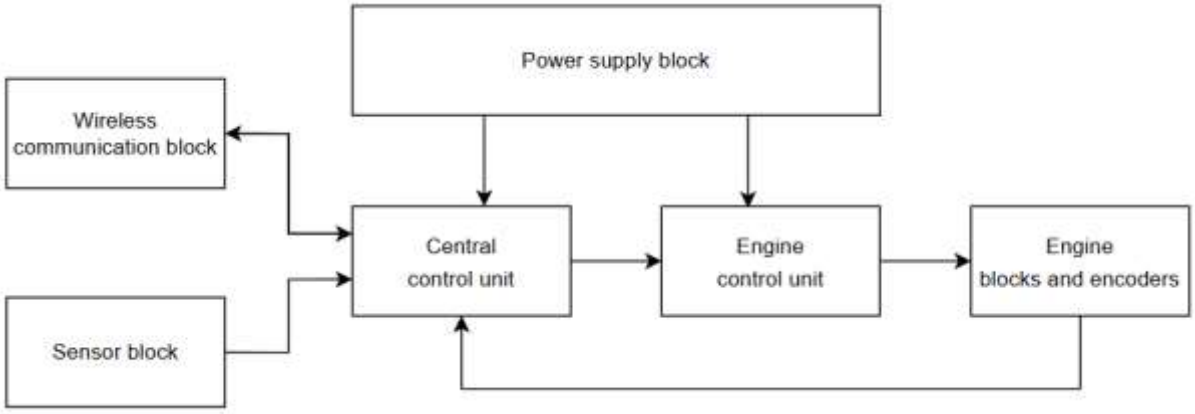


Figure 3.1 System hardware block chart

- System Power Supply: A stable power supply for the Engine Controller to control the engine and the Central Controller for handling each task.
- Central control unit: Calculates controller processing, outputs control signals, reads pulses returned from the encoder.

- Engine control unit: The task is to receive signals from the central control unit to supply the appropriate voltage to control the engine.
- Engine blocks and encoders: Receive the voltage emitted from the engine control blocks.
- Sensor blocks: The primary task is to read and return values from the inclination angle sensor, allowing the central control unit to process the positioning of the system's incline angle speed and angle of rotation.

3.1.2.1. Power supply

The task is to provide a stable supply to the blocks to keep the system operational for a long time. The power supply in the system consists of two 12.6V one-way power sources. The power supply option is 3s 2200mAh 4A battery.

Table 3.1: Specification table of battery 3s 2200mAh 12.6V

Paramater	Value	Units
Voltage	12.6	V
Capacity	2200	mAh
Number of cells	3S	
Discharge rate	25C	
Max discharge rate	50C	
Dimentions	70x70x15	mm
Weight	143	gram

Figure 3.2 Battery 3s 2200mAh 12.6V

3.1.2.2. Engine control unit

- The engine control unit has the task of receiving control signals from the central control unit to convert the control signal to the control voltage of the engine unit and the encoder, controlling the engine to rotate as desired. The requirement for the engine control unit is:
  + The module must match the type of engine used in the system.
  + Provides enough current so that the engine can operate at the right capacity.

⇨ From the above requirements, the team decided to choose the L298N engine control module.

- Technical specifications

Table 3.2: Specifications of the L298N engine control circuit

Paramater	Value	Units
Main IC	L298N	
Maximun power	25	W
Maximun current per brige	2	A
Input voltage	12	VDC
Dimentions	60x55x30	mm
Weight	48	gram

Figure 3.3 Engine controller L298N

3.1.2.3. Engine blocks and encoders

- Based on the calculations in Chapter 2, the JGB37-520 motor was selected

1. M1: Dây nguồn cấp cho động cơ
2. GDN: Nguồn cấp encoder, 0VDC
3. C2: Kênh trả xung A
4. C1: Kênh trả xung B
5. VCC: Nguồn encoder 3.3V
6. M2: Dây nguồn động cơ

Figure 3.4 Foot diagram of the DC Encoder engine

- Servo engine SPT5435LV
  - + perating Voltage: 4.8V – 6.0V
  - + Maximum Torque:
    - 29 kg·cm at 4.8V
    - 35 kg·cm at 6.0V
  - + Dimensions: 40.5 x 20 x 40.5 mm
  - + Weight: 70 g
  - + Gear Material: Full metal gears
  - + Bearings: 2 ball bearings
  - + Motor Type: Iron core motor
  - + Control Type: PWM (Pulse Width Modulation)
  - + PWM Signal Frequency: 330 Hz
  - + PWM Signal Voltage: 3.3V – 5.0V
  - + Control Angle Range: 90° (max 180°)
  - + No-load Current: 100 mA
  - + Rated Current: 1.4 A

Figure 3.5 Servo engine SPT5435LV

3.1.2.4. Sensor block

- Sensor blocks mainly read and process data from magnetic field acceleration sensors, returning angle and speed values to the axes of the system. Here are some requirements for the sensor block:
  + The raw read data must be continuous, not interrupted and not over-interrupted, and receive the measurement value at the required deviation angle.
  + Works well in long-term and experimental environments.
  + Integrated communication standards, possible signal processors.
- From the above requirements, the team proposed and selected the type of MPU6050 sensor to undertake this task.

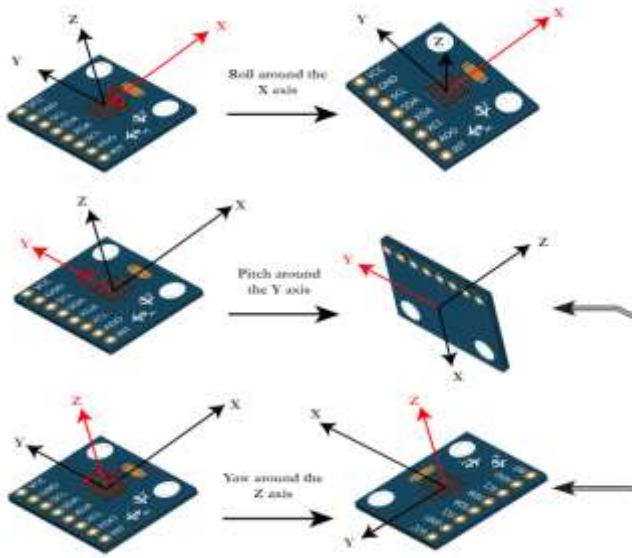


Figure 3.6 Illustration of Roll, Pitch, and Yaw Movements of an IMU Sensor Module

- Specifications of MPU6050:

Table 3.3: Specifications of MPU6050 magnetic field acceleration sensor

Parameter	Description	Units
Chip type	MPU6050	
Power	3.3-5	V
Dimensions	22x17.64	mm
Interface	I2C, SPI	
Gyroscope range	±250, ±500, ±1000, ±2000	
Accelerometer range	±2, ±4, ±6, ±8, ±16	g
Magnetometer range	±4800	

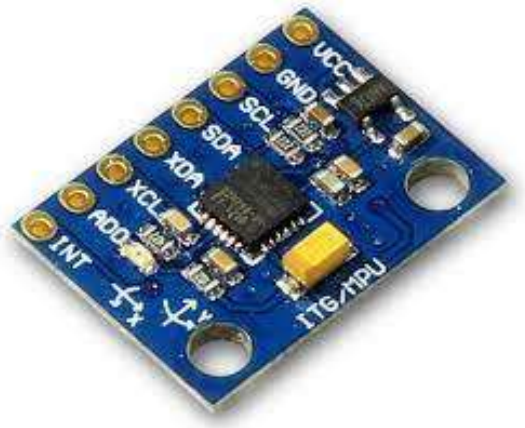


Figure 3.7 MPU6050 Magnetic Field Acceleration Sensor

3.1.2.5. Central control unit

ESP32-WROOM-32 is a versatile, powerfull and widely used MCU module in Wifi-Bluetooth PCB circuit design, BLE is a popular application for many IoT applications today. The range of applications ranges from energy-saving sensor networks to the most complex tasks.

Figure 3.8 ESP32-WROM Foot Chart

- Technical specifications:
  + 2 8-bit Digital to Analog Converters (DAC)
  + 18 12-bit Analog to Digital Converter (ADC) channels
  + 2 I²C communication ports
  + 3 UART communication ports
  + 3 SPI communication ports (1 for FLASH chip)
  + 2 I²S communication ports
  + 10 Pulse Width Modulation (PWM) output channels

+ SD card/SDIO/MMC host
+ Ethernet MAC supporting DMA and IEEE 1588 standards
+ CAN bus 2.0
+ IR (TX/RX)38 chân GPIO
+ 38 feet GPIO

Application: The module is widely used in applications that collect data and control devices via WiFi, Bluetooth....

3.2. Electronic circuit design

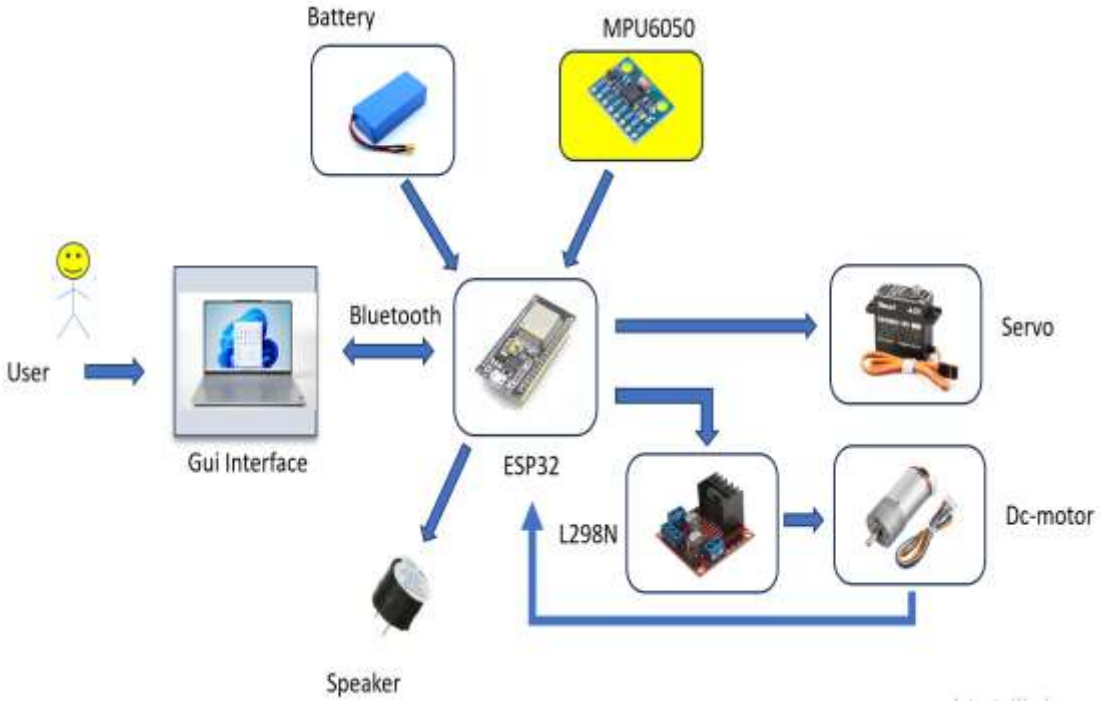


Figure 3.9 Structural circuit system diagram

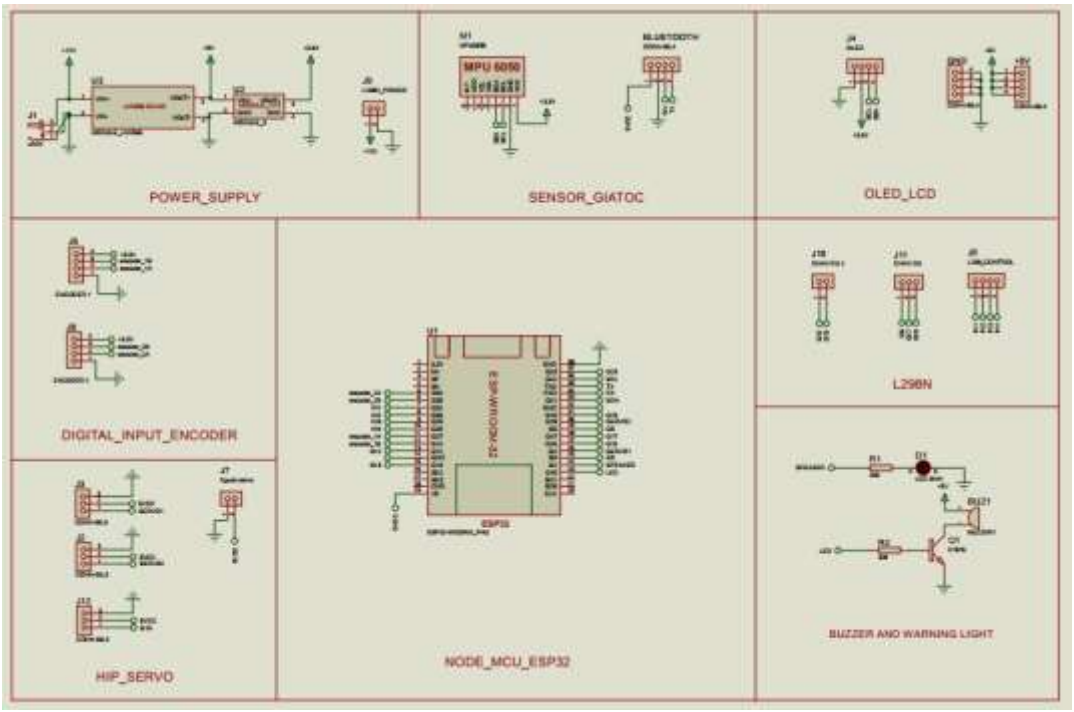


Figure 3.10 Wiring diagram of the system

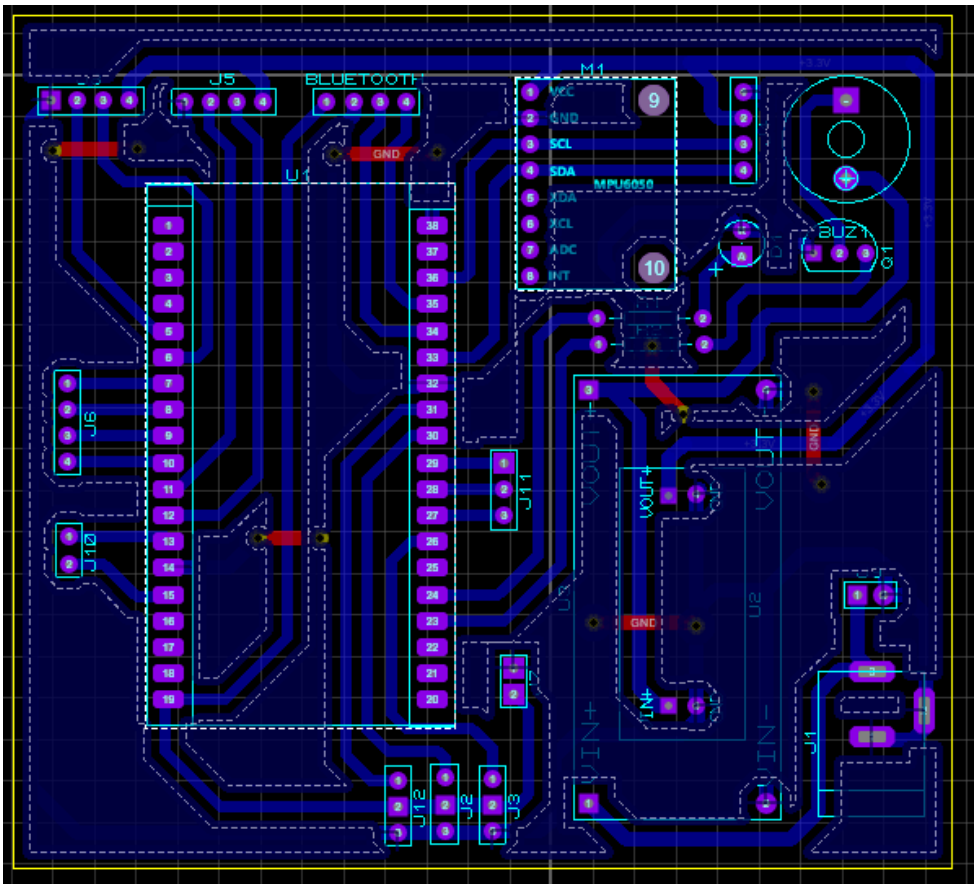


Figure 3.11 Frame PCB layout

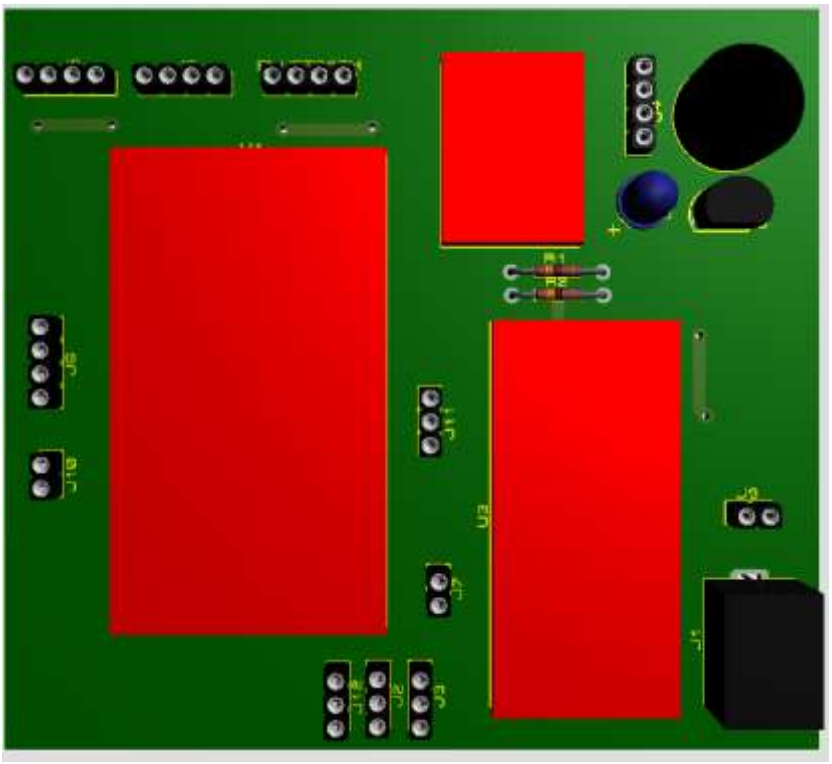


Figure 3.12 3D Proteus

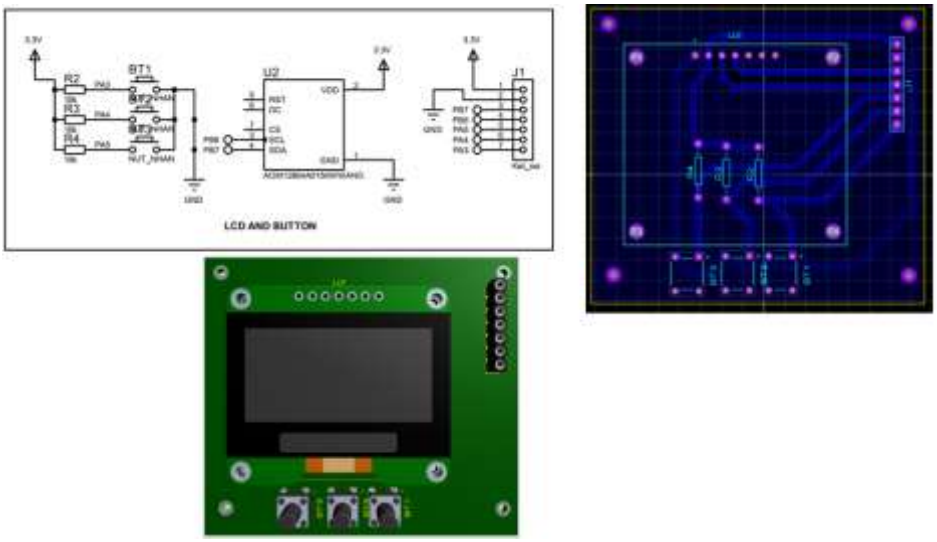


Figure 3.13 Functional keys and 0.96-inch OLED screen

3.3. Controller design LQR

3.3.1. Controller design

- The LQR controller is a commonly used controller in automated systems and systems that require work around the balance position. With the advantages of simple structure, easy to implement, balanced control for the system. That's why the team decided to use

this controller in the system. Figure 3.5 is a block diagram of the LQR controller applied to the system.

❖ According to the document [6], the LQR state feedback controller is an algorithm for finding the optimal K controller for negative state feedback, first of all defining the positive symmetric P matrix as the experiment of the Riccati equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

Where A, B, Q, R are the matrices

- A is the state matrix of the system.
- B is the input control matrix.
- R is the matrix that measures the impact of the input control.
- Q is the matrix that measures the influence of the state.
- P is the positive symmetric matrix)

- Optimal control signal: $u^*(t) = -Kx(t)$
- Elastic deformation: $K = R^{-1}B^T P$

We convert the equation (2.26) of the state variable equation to the following:

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$$

There's:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, B = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, C = \begin{pmatrix} c_1 & \ldots & c_n \end{pmatrix}$$

x is the state variable of the system

u: is the input signal

y: is output signal

A: is state matrix B, is input matrix, C output matrix.

– We set the state variables as follows:

$$\begin{aligned} x_1 = \theta \; ; x_4 = \psi \; ; x_7 = \phi \\ x_2 = \dot{\theta} \; ; x_5 = \dot{\psi} \; ; x_8 = \dot{\phi} \\ x_3 = \ddot{\theta} \; ; x_6 = \ddot{\psi} \; ; x_9 = \ddot{\phi} \end{aligned} \tag{3.1}$$

+ We take the equation (*) as follows:

$$\begin{aligned} x_3 = f_2(x_1, x_2, x_4, x_5, x_7, x_8, v_L, v_R) \\ x_6 = f_4(x_1, x_2, x_4, x_5, x_7, x_8, v_L, v_R) \\ x_9 = f_6(x_1, x_2, x_4, x_5, x_7, x_8, v_L, v_R) \end{aligned} \tag{3.2}$$

⇨ We lower the level of the system and we can take it into the form of:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = f_2 \\ \dot{x_4} = x_5 \\ \dot{x_5} = f_4 \\ \dot{x_7} = x_8 \\ \dot{x_8} = f_6 \end{cases}$$ Then we linearise the system to the equilibrium position so that everything turns to 0.

+ If we choose work point $x_0 = [0\ 0\ 0\ 0\ 0\ 0]^T\ and\ u_0 = [0\ 0]^T$ We can linearize the system as $\dot{x} = Ax + Bu$
+ We got the state variable x = $[\theta, \dot{\theta}, \psi, \dot{\psi}, \phi, \dot{\phi}]^T$ nd output signal u = $[v_L, v_R]^T$

We find matrix A as follows:

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_4} & \frac{\partial f_1}{\partial x_5} & \frac{\partial f_1}{\partial x_7} & \frac{\partial f_1}{\partial x_8} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_4} & \frac{\partial f_2}{\partial x_5} & \frac{\partial f_2}{\partial x_7} & \frac{\partial f_2}{\partial x_8} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_4} & \frac{\partial f_3}{\partial x_5} & \frac{\partial f_3}{\partial x_7} & \frac{\partial f_3}{\partial x_8} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} & \frac{\partial f_4}{\partial x_7} & \frac{\partial f_4}{\partial x_8} \\ \frac{\partial f_5}{\partial x_1} & \frac{\partial f_5}{\partial x_2} & \frac{\partial f_5}{\partial x_4} & \frac{\partial f_5}{\partial x_5} & \frac{\partial f_5}{\partial x_7} & \frac{\partial f_5}{\partial x_8} \\ \frac{\partial f_6}{\partial x_1} & \frac{\partial f_6}{\partial x_2} & \frac{\partial f_6}{\partial x_4} & \frac{\partial f_6}{\partial x_5} & \frac{\partial f_6}{\partial x_7} & \frac{\partial f_6}{\partial x_8} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & A_1 & A_2 & A_3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & A_4 & A_5 & A_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & A_7 \end{bmatrix}$$

In which $A_1, A_2, A_3, A_4, A_5, A_6, A_7$ is calculated in Matlab

$$A_m = 2J_\psi J_w + 2J_w L^2 M + J_\psi M R^2 + 2J_\psi J_m n^2 + 4J_m J_w n^2 + 2J_\psi R^2 m + 2J_m L^2 M n^2 + 2J_m M R^2 n^2 + 2L^2 M R^2 m + 4J_m R^2 m n^2 + 4J_m L M R n^2$$

$$A_1 = -\frac{2J_\psi \beta + 2J_\psi f_w + 2L^2 M\beta + 2L^2 M f_w + 4J_m f_w n^2 + 2LMR\beta}{A_m}$$

$$A_2 = -\frac{RgL^2M^2 - 2J_m gLMn^2}{A_m}$$

$$A_3 = \frac{2M\beta L^2 + 2MR\beta L + 2J_\psi \beta}{A_m}$$

$$A_4 = \frac{4J_w\beta + 2MR^2\beta - 4J_m f_w n^2 + 4R^2\beta m + 2LMR\beta + 2LMRf_w}{A_m}$$

$$A_5 = \frac{LgM^2R^2\beta + 2LgmMR^2 + 2J_m LgMn^2 + 2J_w LgM}{A_m}$$

$$A_6 = -\frac{4J_w\beta + 2MR^2\beta + 4R^2\beta m + 2LMR\beta}{A_m}$$

$$A_7 = -\frac{W^2\beta + W^2 f_w}{mR^2W^2 + 2J_\phi R^2 + J_m W^2 n^2 + J_m W^2}$$

+ We found matrix B as follows:

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial v_L} & \frac{\partial f_1}{\partial v_R} \\ \frac{\partial f_2}{\partial v_L} & \frac{\partial f_2}{\partial v_R} \\ \frac{\partial f_3}{\partial v_L} & \frac{\partial f_3}{\partial v_R} \\ \frac{\partial f_4}{\partial v_L} & \frac{\partial f_4}{\partial v_R} \\ \frac{\partial f_5}{\partial v_L} & \frac{\partial f_5}{\partial v_R} \\ \frac{\partial f_6}{\partial v_L} & \frac{\partial f_6}{\partial v_R} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ B_1 & B_1 \\ 0 & 0 \\ B_2 & B_2 \\ 0 & 0 \\ -B_3 & B_3 \end{bmatrix}$$

In which B_1, B_2, B_3 are calculated in Matlab

$$B_m = 2J_\psi J_w + 2J_w L^2 M + J_\psi M R^2 + 2J_\psi J_m n^2 + 4J_m J_w n^2 + 2J_\psi R^2 m + 2J_m L^2 M n^2 + 2J_m M R^2 n^2 + 2L^2 M R^2 m + 4J_m R^2 m n^2 + 4J_m L M R n^2$$

$$B_1 = \frac{\alpha\left(ML^2 + MRL + J_\psi\right)}{B_m}$$

$$B_2 = -\frac{\alpha(2J_w + MR^2 + 2R^2 m + LMR)}{B_m}$$

$$B_3 = \frac{RW\alpha}{mR^2W^2 + 2J_\phi R^2 + J_m W^2 n^2 + J_w W^2}$$

- After identifying matrix A and matrix B, we choose matrix Q, R. We change the parameters of the system into Matrix A and Matrix B. We get Matrix.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & A_1 & A_2 & A_3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & A_4 & A_5 & A_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & A_7 \end{bmatrix}, \mathrm{B} = \begin{bmatrix} 0 & 0 \\ B_1 & B_1 \\ 0 & 0 \\ B_2 & B_2 \\ 0 & 0 \\ -B_3 & B_3 \end{bmatrix}$$

❖ Model structure

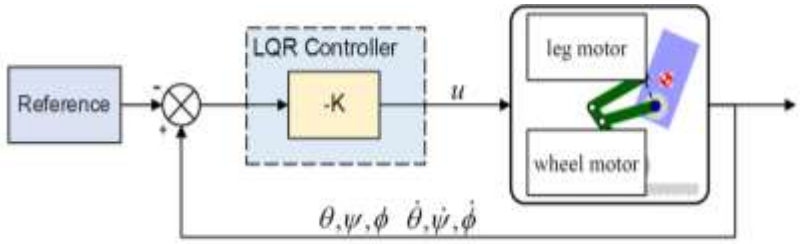


Figure 3.14 LQR controller block chart for the system

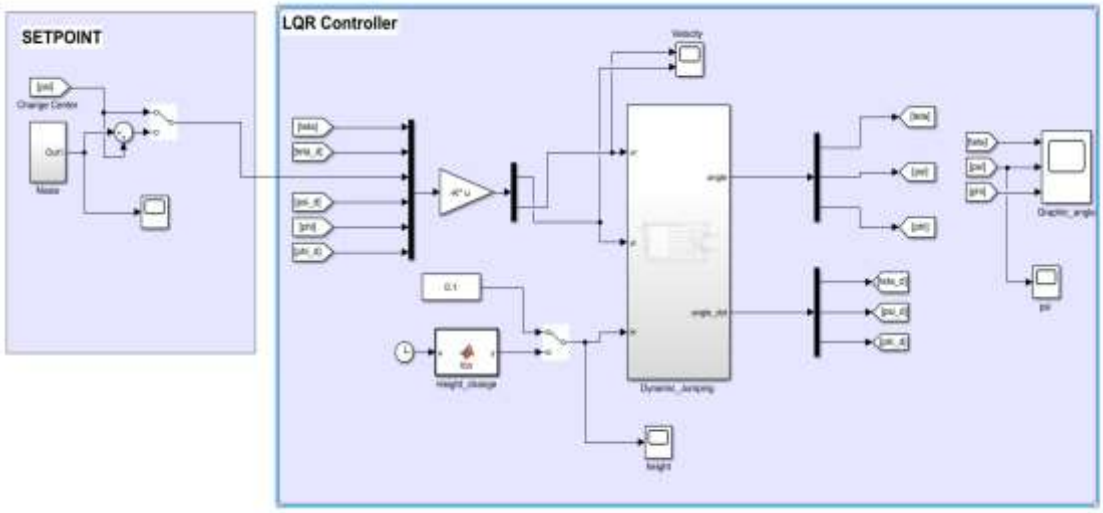


Figure 3.15 LQR Control Model

- We initialized the robot's initial values at the values: x1_init = 0.01; x2_init = -0.012, x4_init = 0.03, x5_init = - 0.02, x7_init = 0.04, x8_init= - 0.03
  - For the height of the robot in low position H = 100 mm corresponding to matrix A_1, B_1, Q_L, R_L
  - We choose the Q_L *and* R_L matrix
    - TH1: we choose

$$Q_L = diag([1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1]),\ R_L = diag([1 \;\; 1])$$

- Use the command available in matlab to determine the K-value matrix

$$K1 = lqr(A_1, B_1, Q_L, R_L)$$

❖ Results

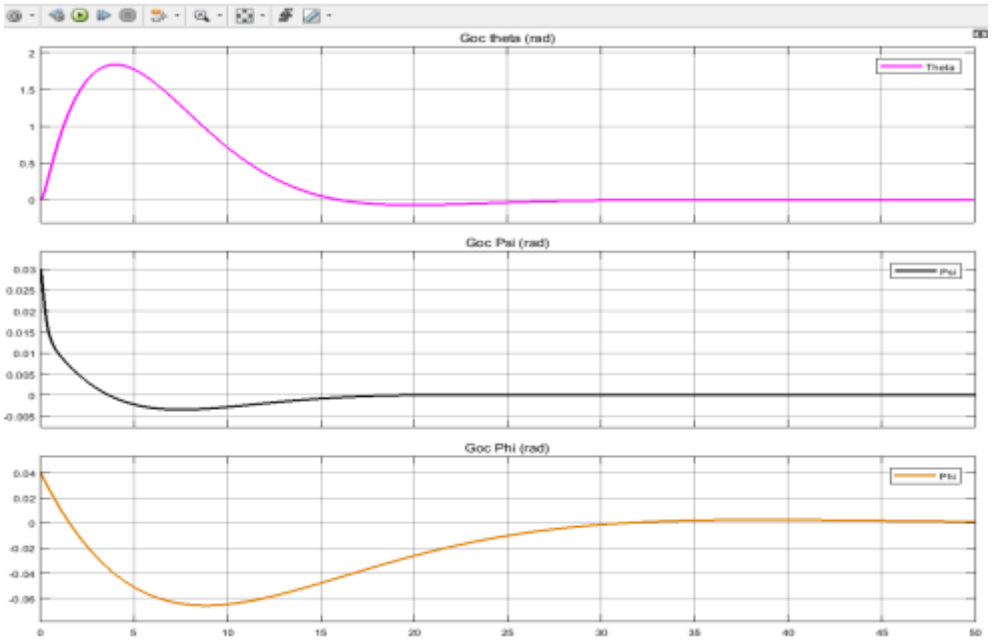


Figure 3.16 The diagram corresponds to the rotation angle of the robot

⇨ Comments: Based on the graph, we find that the output angle response is still quite long and the time to get to the equilibrium position needs to be improved so we need to re-select the K-value matrix by changing the parameters in the Q L matrix to improve the system's response time.

TH2: We tried several fields and we picked the following weight:

$$Q_L = diag([705\ \ 210\ \ 1200\ \ 268\ 100\ \ 68]), \quad R_L = diag([1 \quad 1])$$

- Use the command available in matlab to determine the K-value matrix

$$K2 = lqr(A_1, B_1, Q_L, R_L)$$

❖ Results

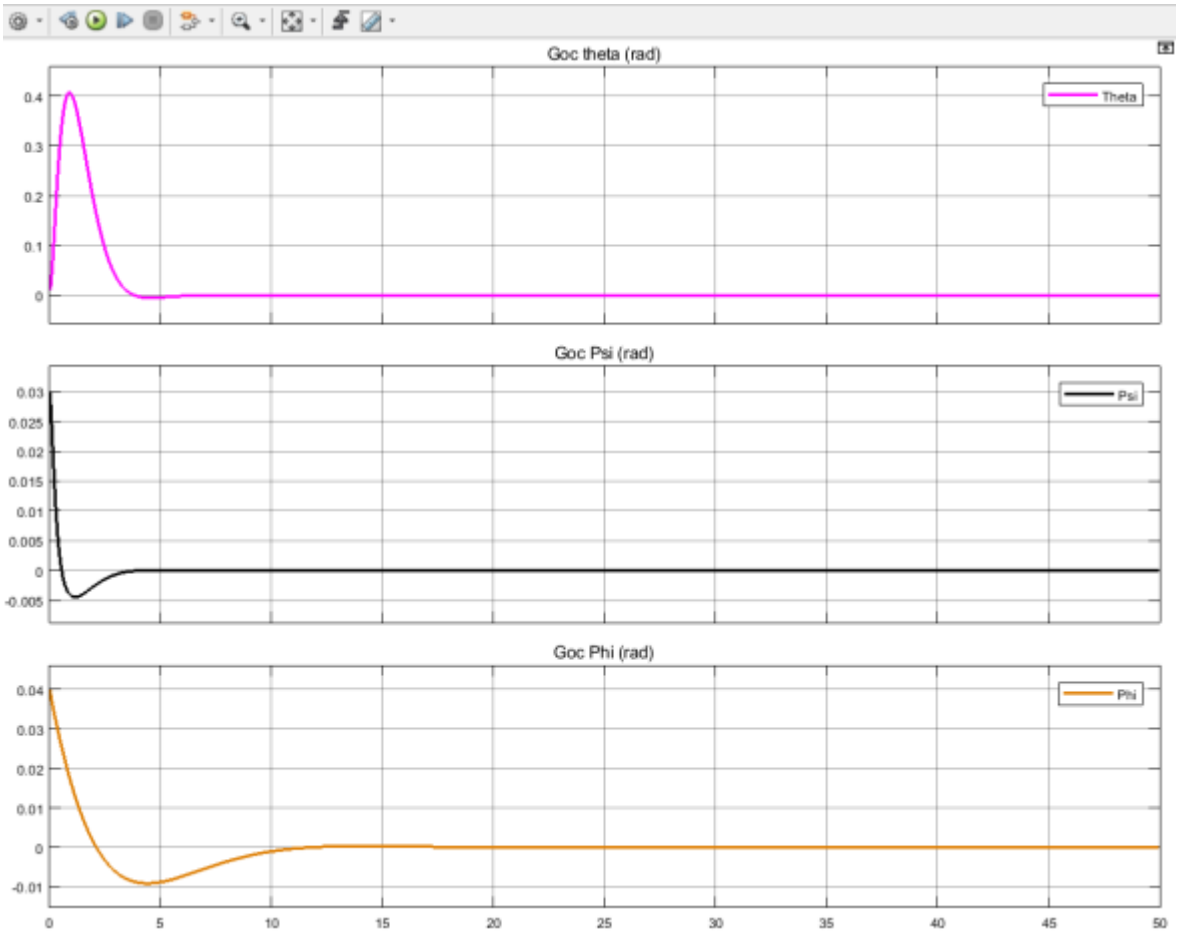


Figure 3.17 Output Response Graph

⇨ Comments: The output response was significantly improved compared to TH1 when we re-selected the K-time matrix for the equilibrium position and returned rapidly and the fluidity decreased significantly, besides we could test the time increase by testing multiple cases to select the K matrix better.

❖ For the height of the robot in the average position H = 160 mm corresponding to the matrix A_2, B_2, Q_M, R_M we perform the same as in the case of the lowest position we test and find the optimal matrix.

. We choose Q_M *and* R_M matrix

- TH1: We choose

$$Q_M = diag([1\ \ 1\ \ 1\ \ 1\ 1\ 1]), \qquad R_M = diag([1\ 1]$$

- TH2 : We choose

$$Q_M = diag([980\ \ 340\ \ 1200\ 348\ \ 100\ \ 68]), \quad R_M = diag([1\ 1])$$

  - Use the command available in matlab to determine the K-value matrix

$$K2 = lqr(A_2, B_2, Q_M, R_M)$$

⇨ Results obtained: Based on graphical review, we found that the output angle response has returned to a balanced state and the response is relatively stable.

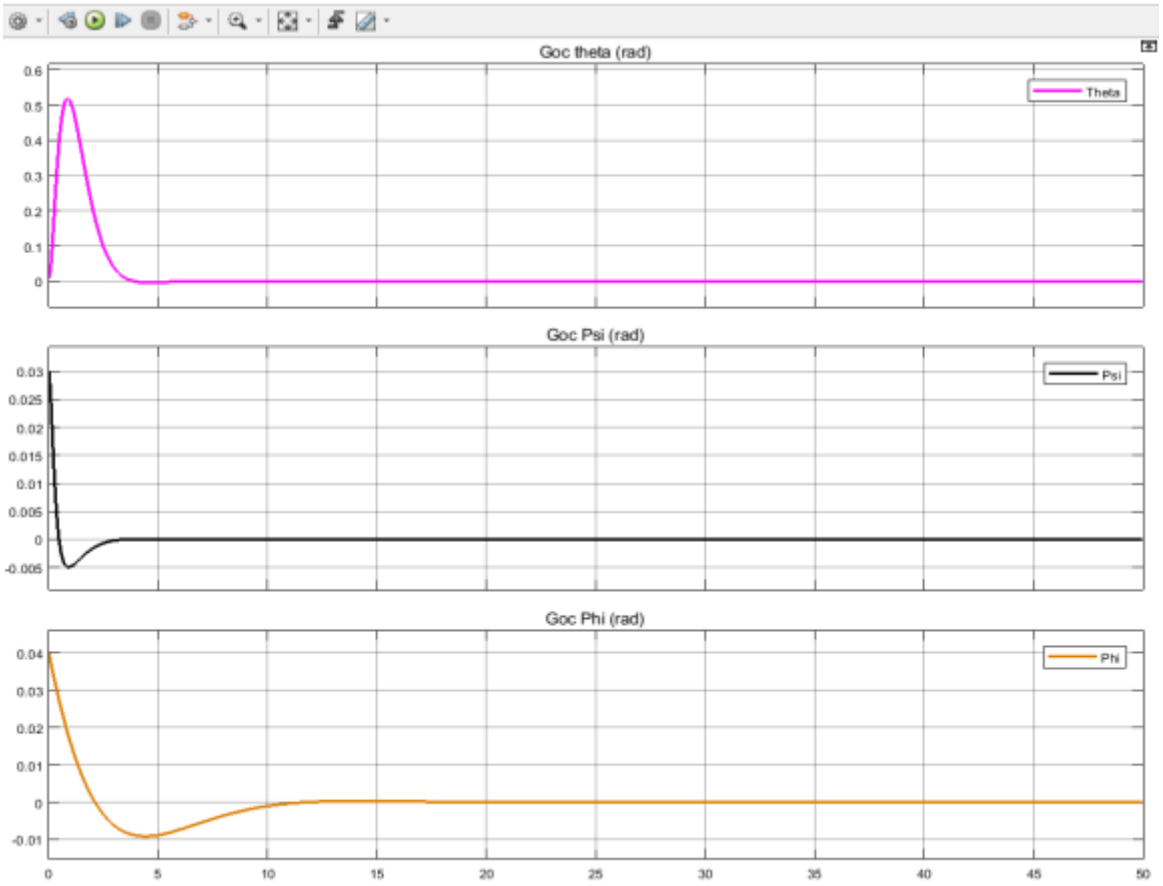


Figure 3.18 Output Response Graph

❖ For the height of the robot in the highest position H = 220 mm corresponding to the matrix A_3, B_3, Q_H, R_H we perform the same test as for the low and medium position:

\+ We choose the following advanced matrix:

$$Q_H = diag([705 \;\; 310 \;\; 1200 \;\; 368 \;\; 100 \;\; 98]), \quad R_H = diag([1 \;\; 1])$$

- Use the command available in matlab to determine the K-value matrix

$$K3 = lqr(A_3, B_3, Q_H, R_H)$$

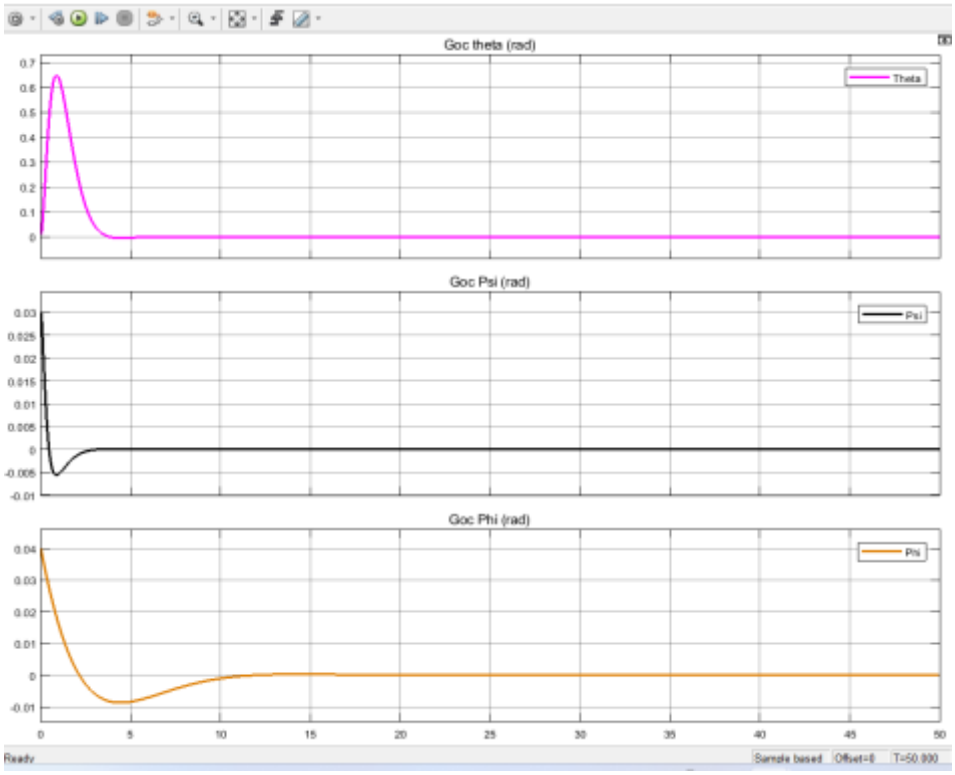


Figure 3.19 Output Response Graph

⇨ Comment: Based on the graph we see that the output angle response has returned to a stable state of equilibrium and response, to look at the stability of the system we will then affect the interference signal and the altitude change on the system to see how the system's response is.

❖ Impact of a 0.1 rad amplitude interference signal on the focus of the psi angle robot at 18-19 s and change the height of the robot from 0.1m to 0.22m in the period after 23s in the highest position to after 35s at the lowest position.

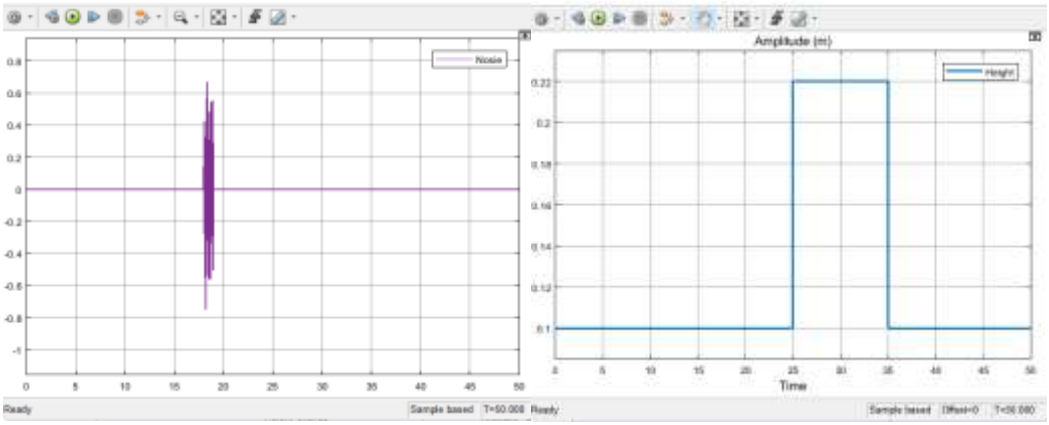


Figure 3.20 a) Interference signal impact on psi angle b) Changing height of the robot

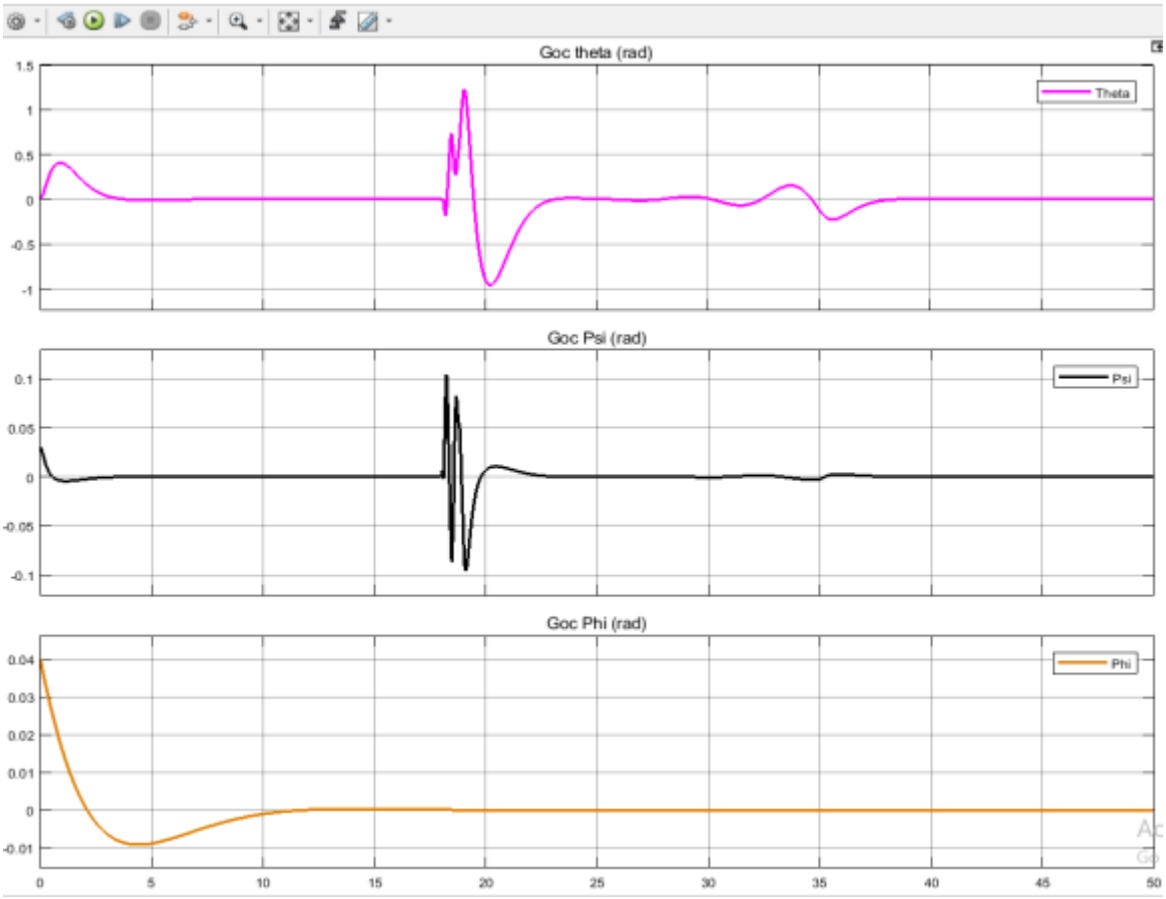


Figure 3.21 Chart responds to interference output and changes in height

⇨ Comments: Based on the above graph, we found that when the interference signal affects T = 18 -19s up the PSI angle of the system, a change occurs, but the system still takes a rapid PSI deviation to the equilibrium position, indicating that the controller has met the required requirement. At the same time, after a period of 25 seconds of height change, the mind of the robot will change the control for the robot to have a state of oscillation that disbalances the robot as we can see on the graph. That's why we need to design a more advanced control to control Psi's inclination to keep the robot in balance when height changes.

3.4. Construction of controller LQR – FUZZY

3.4.1. Controller Structure LQG

- The LQR controller is designed to balance the system. But when the two-wheeled robot changes height, the dynamics of the robot change, namely, the focal position of the shifted system affects the system's balance. If only one LQR controller is used for the entire operation, the system will not meet the set control requirements. So, the design and application of advanced algorithms is essential. In this section, the

team presents the combined control methods between FUZZY and LQR with the aim of improving the quality of the system, allowing the system to balance when the robot is in operation by adjusting the K3 and K4 coefficients of the LKR using FAZZY.

- The rules of height control are presented as follows:

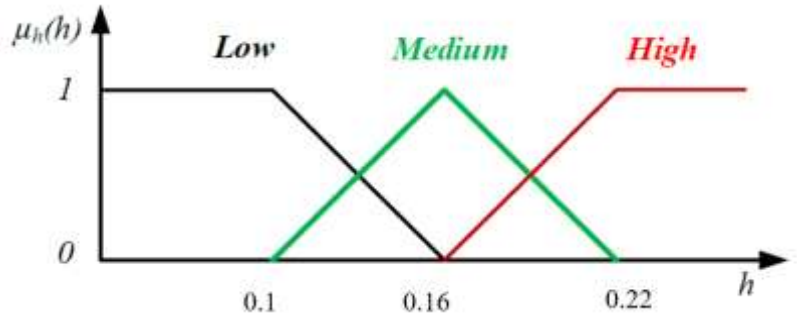


Figure 3.22 Robot height control rules h (m)

$$u = \begin{cases} K_1 x, h < h_{Low} \\ K_2 x, \ h \geq h_{medium} \text{ và } h < h_{high} \\ K_3 x \ \ , h \geq high \end{cases} \quad (3.3)$$

There's:

+ K_1 ∈ Matrix (2x6) is the matrix of the LQR1 controller in low position.
+ K_2 ∈ Matrix (2x6) is the reversibility matrix of the LQR2 controller in the middle position.
+ K_3 ∈ Matrix (2x6) is the matrix of the LQR3 controller in the high position.
+ $h_{trungbinh}$ is the average height of the robot, h_{low} is the lowest height and h_{high} is the maximum height, the rest h is the current height that changes over time.

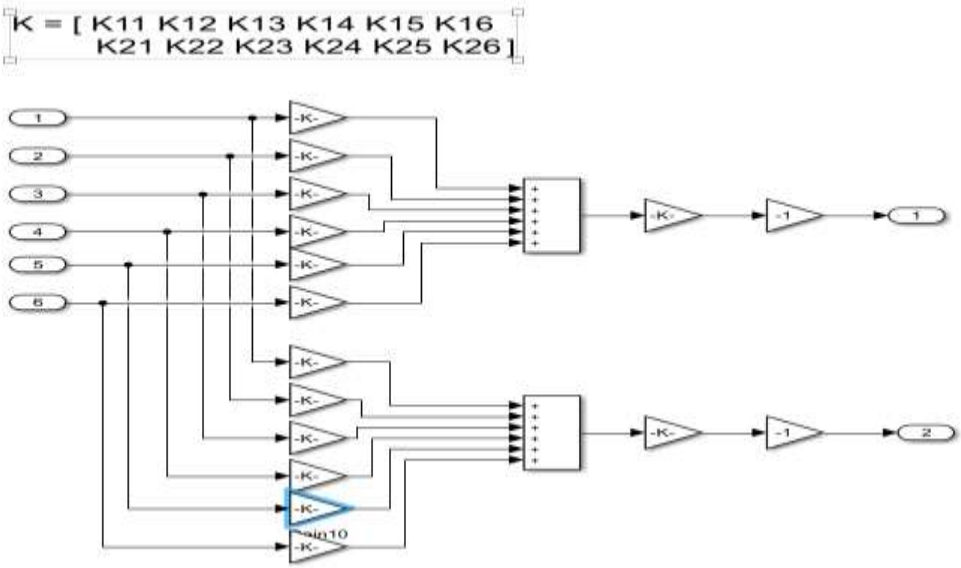


Figure 3.23 Modified K-factor matrix of LQR controller

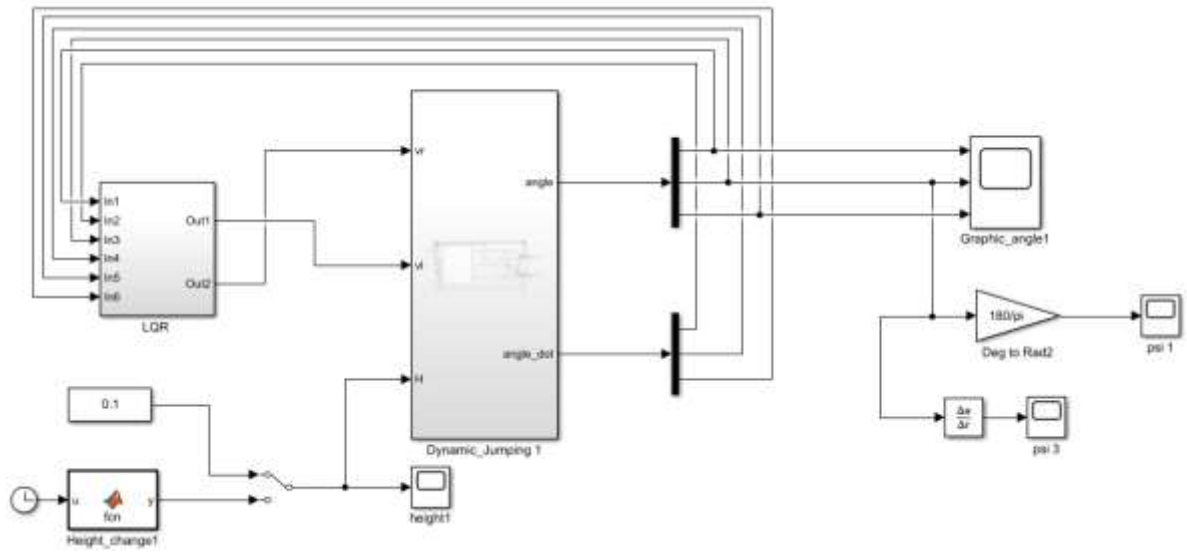


Figure 3.24 Vehicle control model after transformation

3.4.2. Design Fuzzy – LQR blurred

- Blur design using Max-Min roofing and focused blur disassembly.

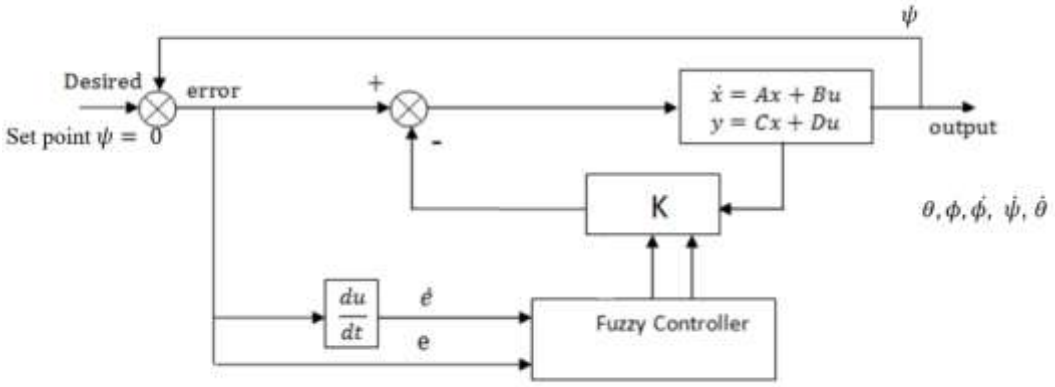


Figure 3.25 Presentation of Fuzzy Structural Chart – LQR

- The goal of building an advanced controller to keep the robot in balance when height changes. The K3 and K4 parameters are calibrated to match the robot's height change, which will stabilize the Psi angle of the robot.
- Defines language variable:
  + Input two variables: error e (psi) and de/dt (edpsi)
  + Two variable outputs: K3 and K4 coefficients
- The number of language variables blurred as follows:
  + Input variables: NB is Negative Big, NS is Negatively Small, ZE is Zero, PS is Positive, PB is Positively Big.

+ Output variable: NB are Negative Large, NS are Negatives Small, Z are Zero, PS are Positive Small, PB are Pozitive Large.

- The input values of the two inputs may vary depending on the desired controller's response values.

- Input and output variables of the Fuzzy controller

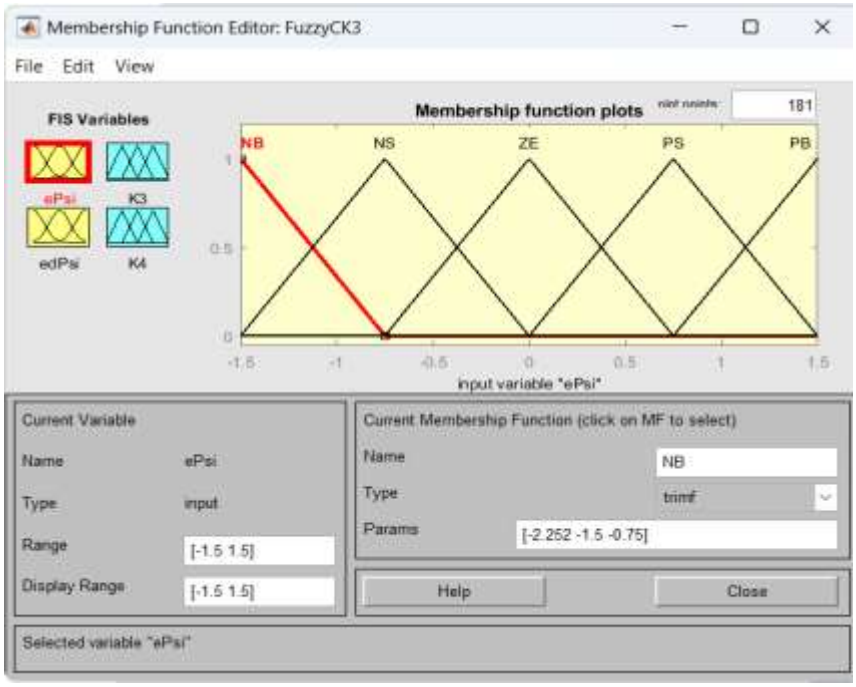


Figure 3.26 Number of signals entering and exiting Fuzzy

❖ Build related functions

- Error e Psi

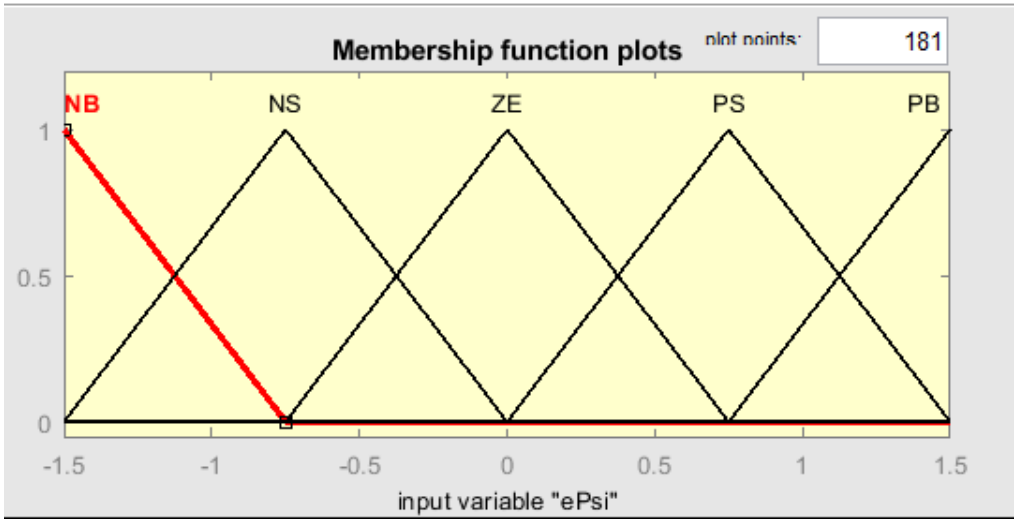


Figure 3.27 Error entering of fuzzy function e

- Error function edot Psi

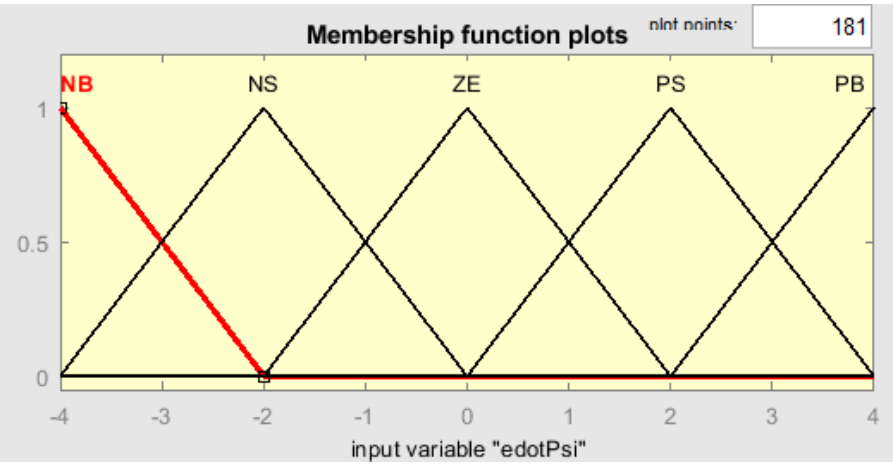


Figure 3.28 Input function error over time

- K3 and K4 exit members

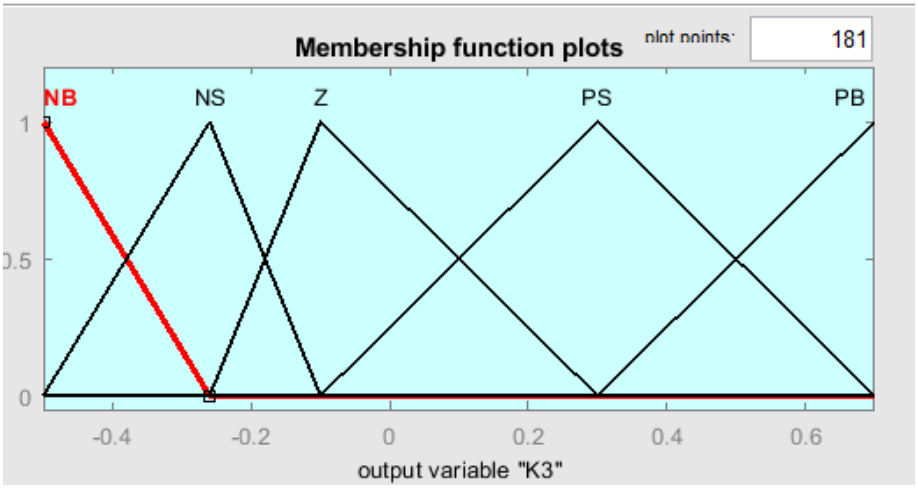


Figure 3.29 K3 forecast output

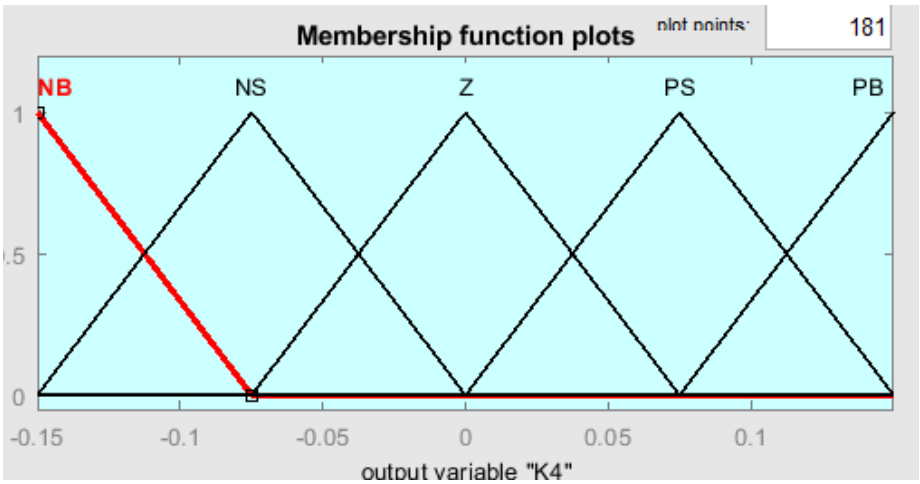


Figure 3.30 K4 forecast output

❖ Build the law of unification

Table 3.4: K3 parameter adjustment rules

K3		e				
		NB	NS	ZE	PS	PB
de/dt	NB	NB	NB	NS	NS	Z
	NS	NB	NS	NS	Z	PS
	ZE	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	Z	PS	PS	PB	PB

Table 3.5: Rule of K4 parameter adjustment

K4		e				
		NB	NS	ZE	PS	PB
de/dt	NB	NB	NB	NS	NS	Z
	NS	NB	NS	NS	Z	PS
	ZE	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	Z	PS	PS	PB	PB

3.4.3. Simulation Chart

- Selection parameters for the controller Fuzzy-LQR:

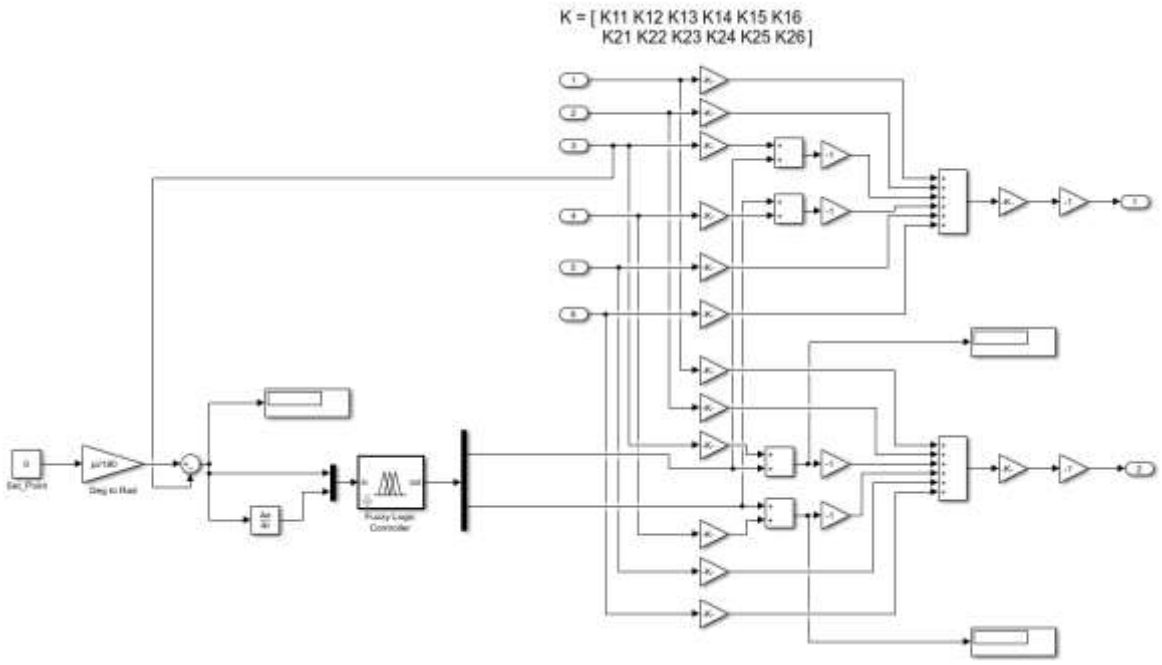


Figure 3.31 K3 and K4 calibration Fuzzy models using height fuzzy

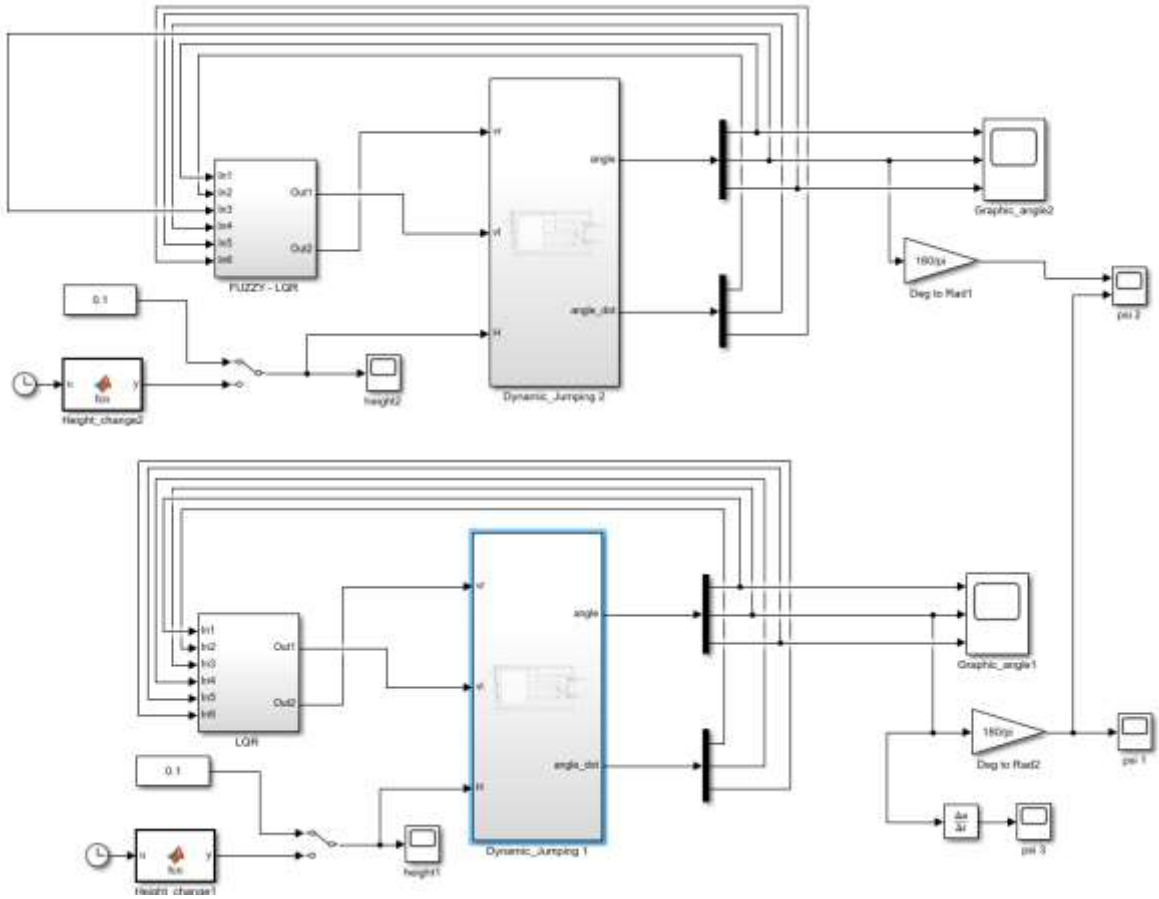


Figure 3.32 Fuzzy model combination compared to LQR

3.4.4. Controls in the Fuzzy set

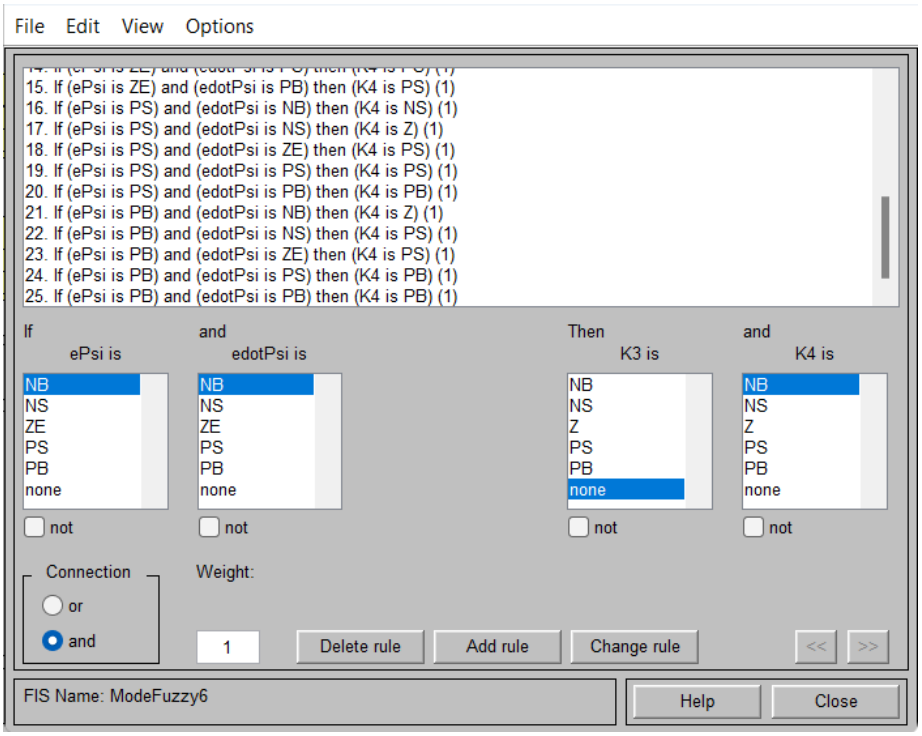


Figure 3.33 Rules of Control

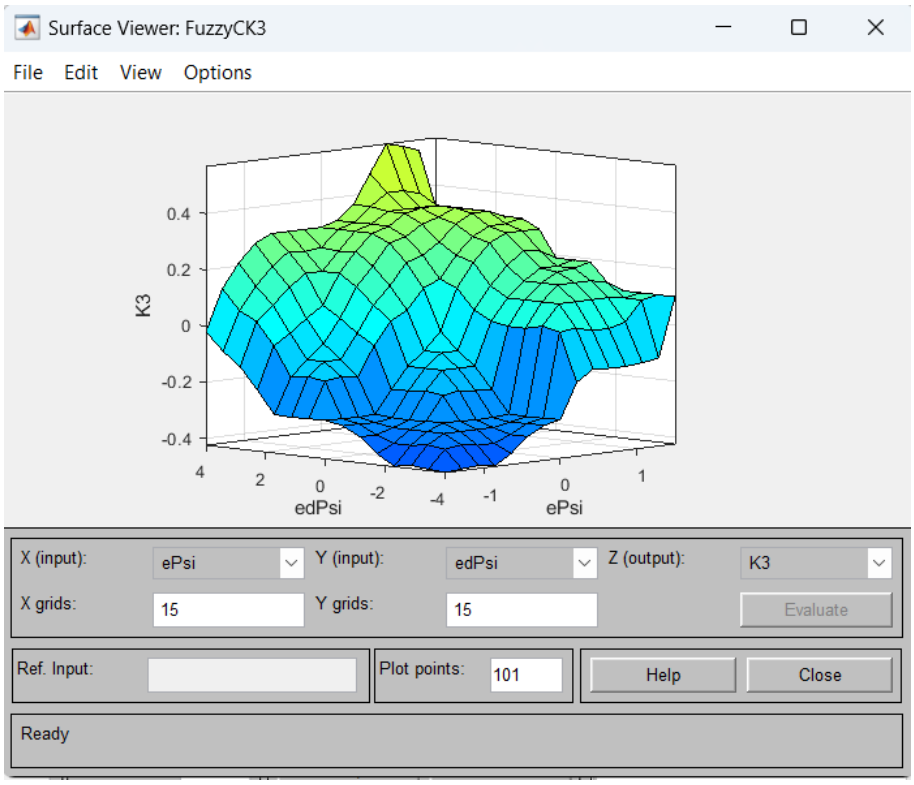


Figure 3.34 Relationship between input and output

3.5. Dynamic simulation

❖ From the mathematical dynamics analysis of chapter 4 above, we simulate the motion dynamics of a robot's leg frame that changes in height:

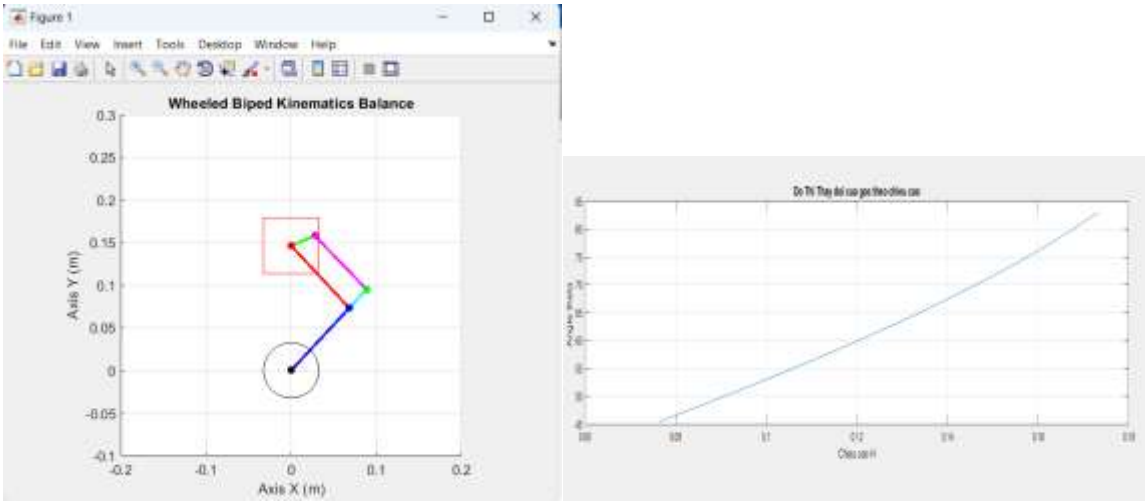


Figure 3.35 Simulates a robot's left leg frame

❖ From these mathematical analyses, we're going to simulate a robot's full and square orbit based on its position and direction in equation (2.10) in section 3. We performed a simulation of the movement of the robot and we obtained the following results:

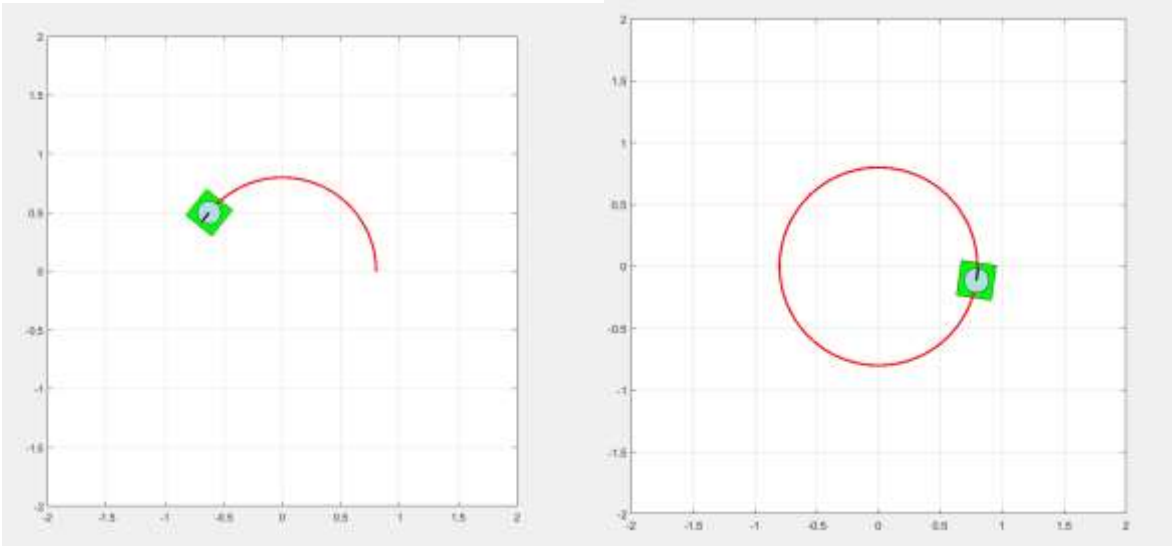


Figure 3.36 Simulates circular orbit movement

- Moving in a square orbit.

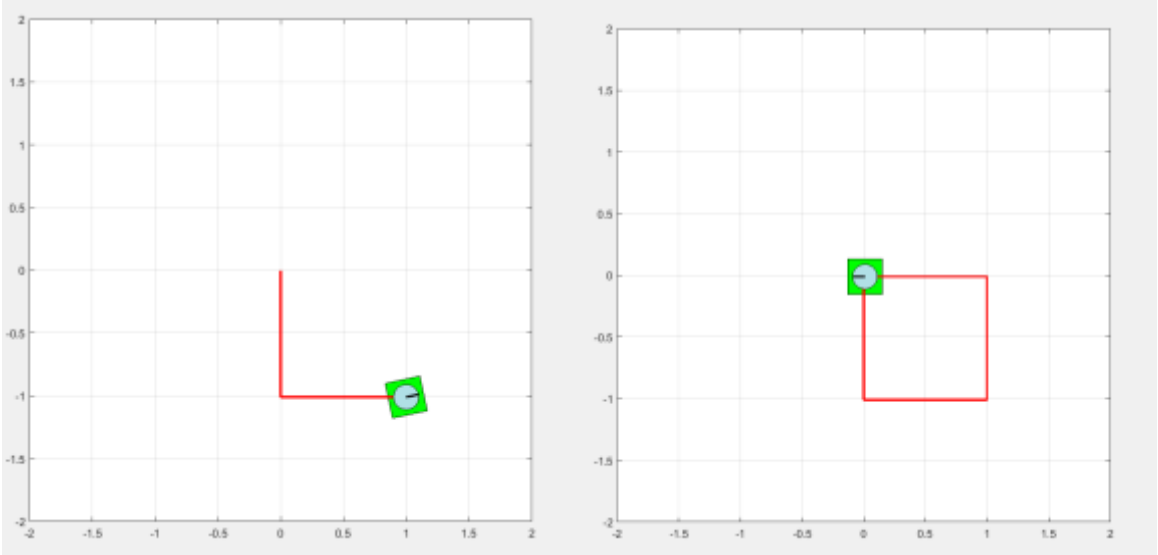


Figure 3.37 Simulates the process of moving in a square orbit

3.6. FUZZY – LQR altitude change application

❖ From the analysis and the argument to the construction of the law of dimming and the controller will automatically calibrate the set of K matrix parameters to suit the robot's height to bring the psi deviation angle to zero, we proceed to the simulation of t = 8s and get the following result:

. Case 1: The robot is at the highest state of 0.22 m and Psi angle = 0

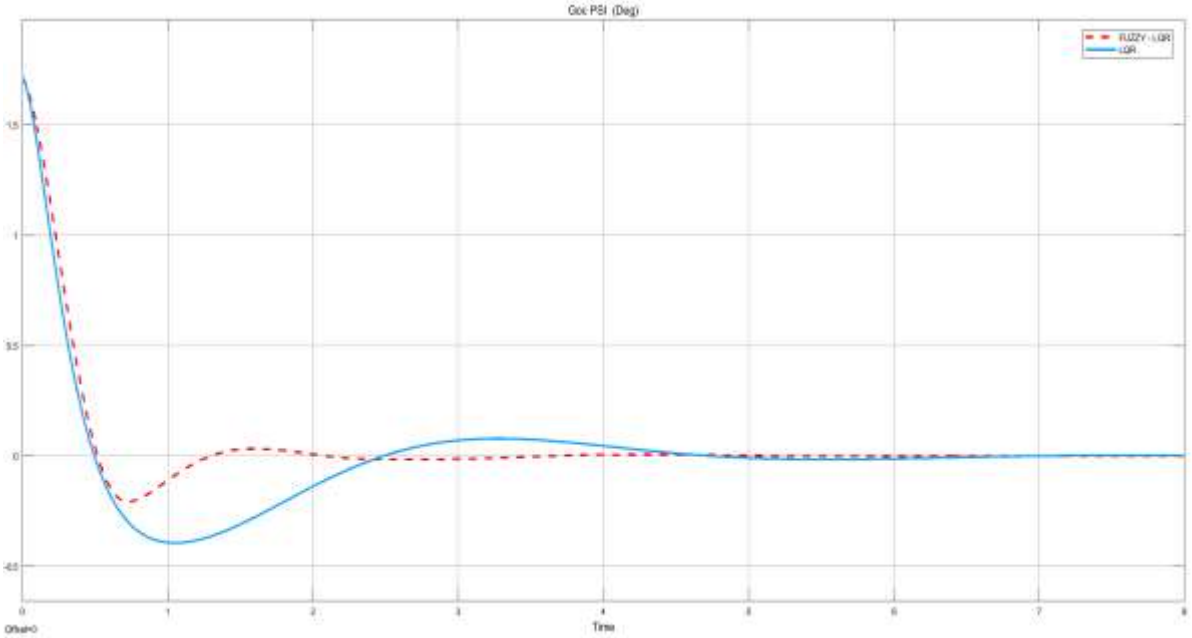


Figure 3.38 Fuzzy-LQR and LQR diagrams with h = 0.22 m

⇨ Comments: Based on the above graph, we can see that the intersection of the Fuzzy-LQR controller responds faster in equilibrium position with only the application of

each LQR controller and the time set within 3s < 4.5 more than that of the LQr controller

. Case 2: The robot is at an average of 0.16 m and Psi angle = 0

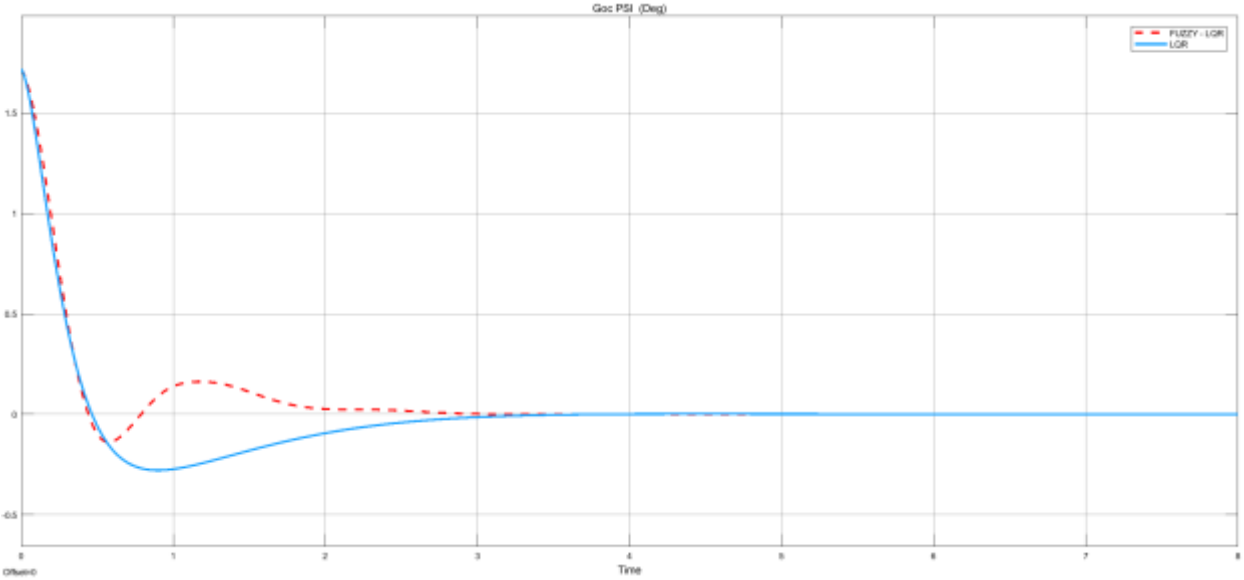


Figure 3.39 Fuzzy-LQR and LQR diagrams with h = 0.16 m

⇨ Comments: Based on the above graph, we can see that the intersection of the Fuzzy-LQR controller responds faster in equilibrium position with only the application of each LQR controllers and the time set within about 2.5 s < 3s more than the controller.

. Case 3: Similar to a robot in the lowest state of 0.1 m and Psi angle = 0

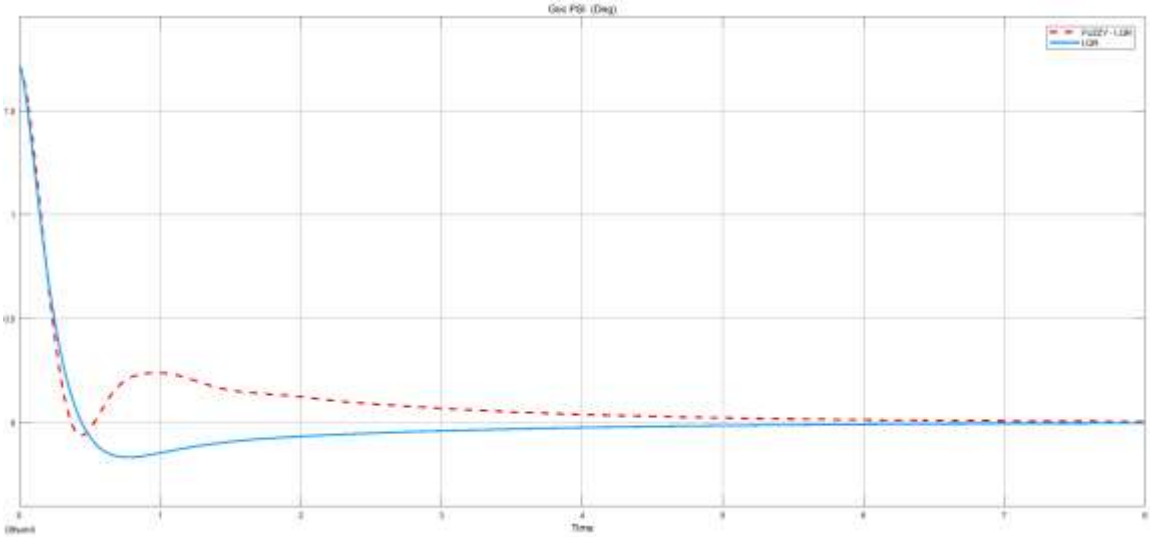


Figure 3.40 Fuzzy-LQR and LQR diagrams with h = 0.1 m

⇨ Comment: Based on the graph above, we can see that the intersection line applies to the Fuzzy-LQR controller and the LQR is moving towards the balance position, but

the Fuzu-LqR line is defined slightly earlier. So we can conclude that with the application of this advanced controller, the response of the system will be significantly improved. We can optimize the system further by testing multiple cases for an optimal set of numbers.

. Interference impact on the Psi angle system with t = 10 – 11s and t = 20 - 21 s simulation time 24s

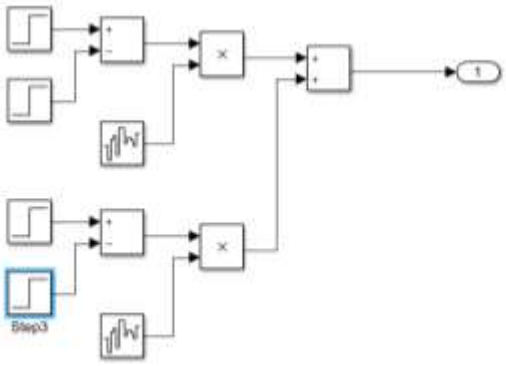


Figure 3.41 Generate interference signal impacting the Psi angle

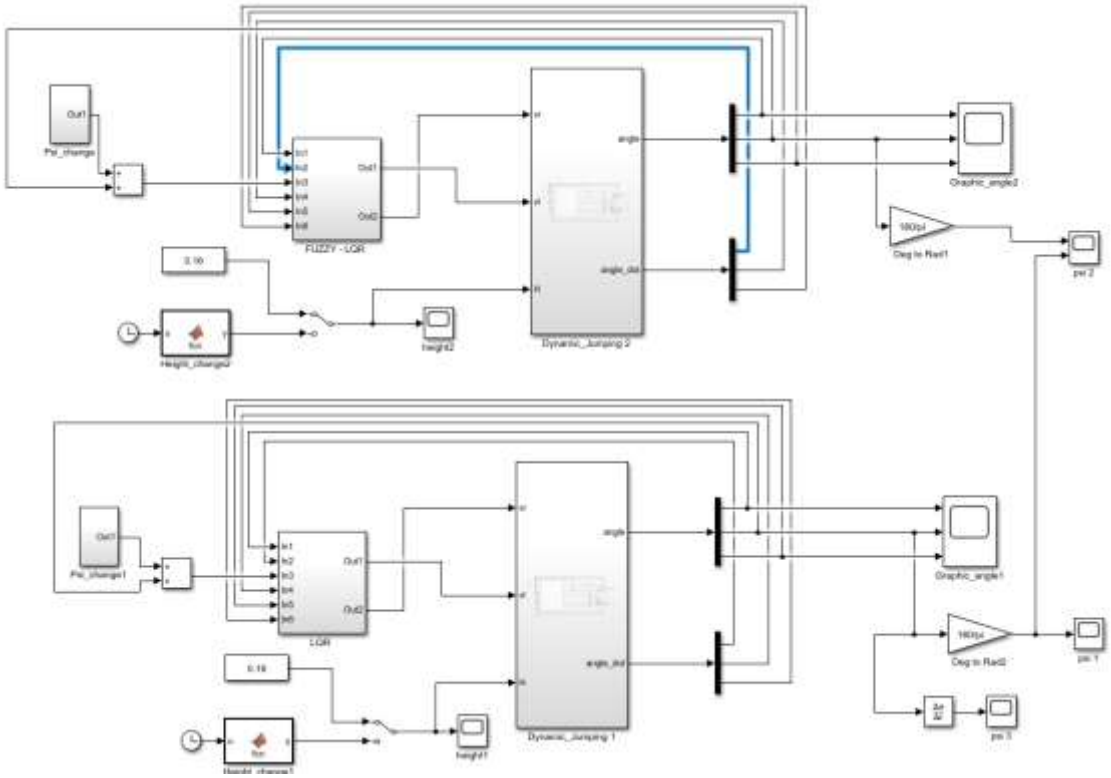


Figure 3.42 Attach the interference block to the model

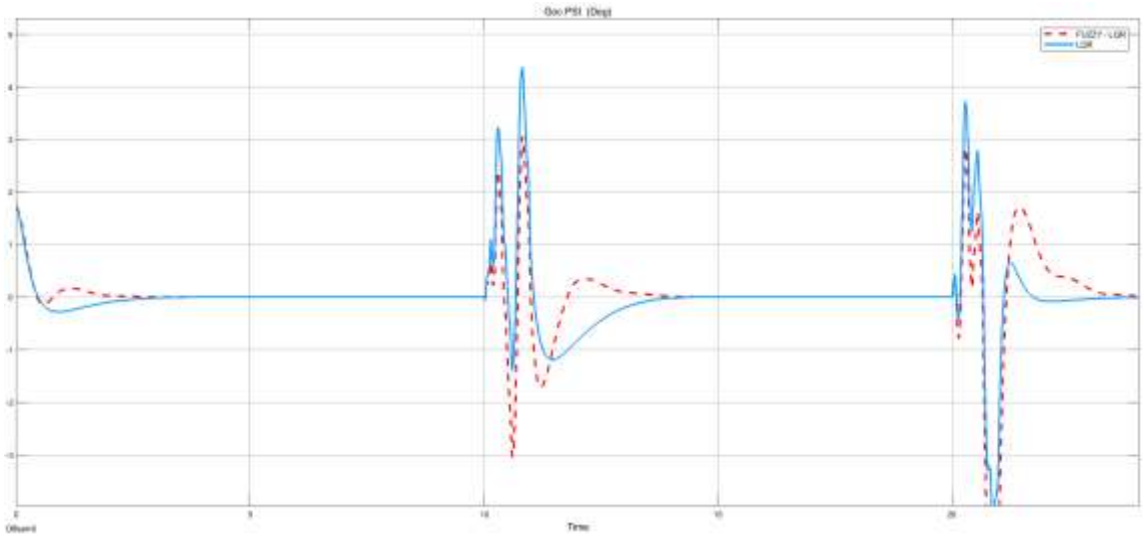


Figure 3.43 Impact of interference on the Psi angle of the system

⇨ Comment: Based on graph 2.24 below, we can see that the response of the Fuzzy-LQR set to the Psi angle of speed compared to the equilibrium position shows that the above controller has met the design requirement of keeping the psi angle of inclination balanced so that the vehicle does not fall when changing the height and impact on its incline angle.

3.7. Microcontroller programming

- The control system consists of a microcontroller (ESP32) to process and output control signals for 4 motors corresponding to 4 joints of the robot. The microcontroller's tasks include reading signals from the accelerometer sensor, reading and converting encoder pulses, calculating the necessary voltage to control the motors according to the set signals, and then communicating via Bluetooth to transmit data to a PC for display on the control interface.

❖ Operating Principle:

- The robot operates based on the inverted pendulum principle, using the tilt angle to adjust the balance of the two-wheeled robot. The circuit is designed with several features: a startup signal emitted by a speaker and an alert indicated by flashing LED. A 12V input voltage powers the entire system, which is then converted to 5V by a power supply and provided to the ESP32, further stepping down to 3.3V to power the encoder and MPU6050 sensor. The circuit uses an L298N driver to control the two DC encoder motors.

❖ Algorithm flowchart:

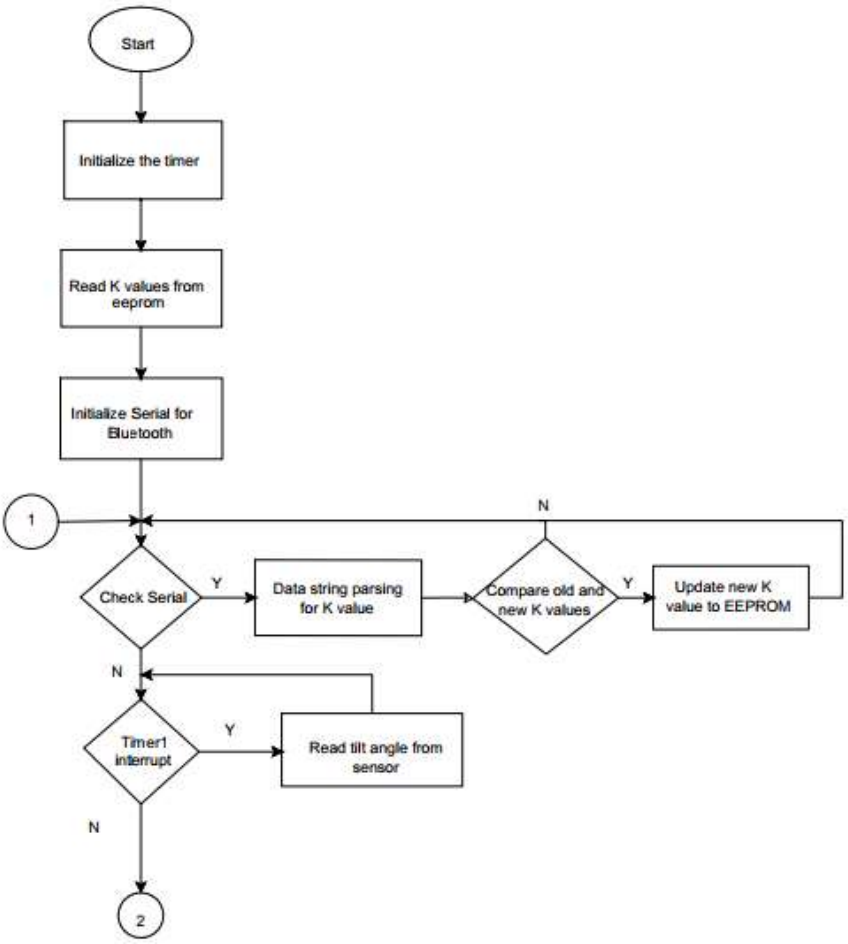


Figure 3.44 Program for processing signals from encoder and sensor

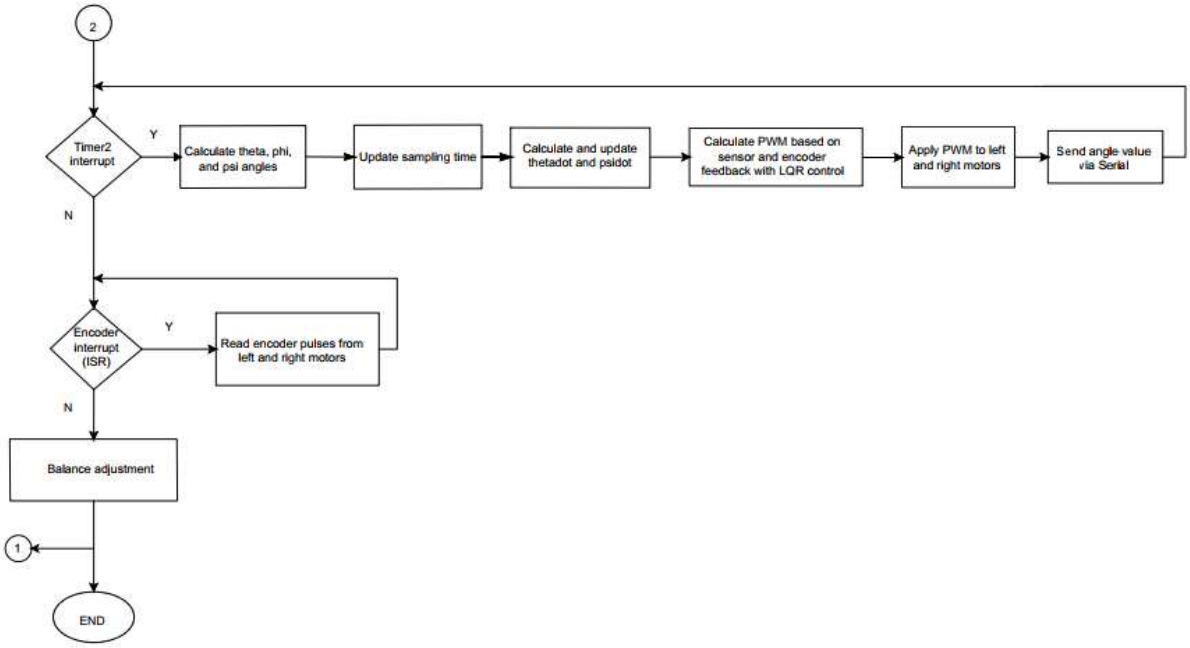


Figure 3.45 Flowchart of the main program of the system

CHAPTER 4 : RESULTS AND DIRECTIONS

4.1. Experimental model and results

4.1.1. Actual Model

- The team has successfully designed and manufactured a two-wheeled balancing vehicle model with adjustable height. The model features a three-link mechanism with five joints, as shown in Figure 4.1 below:

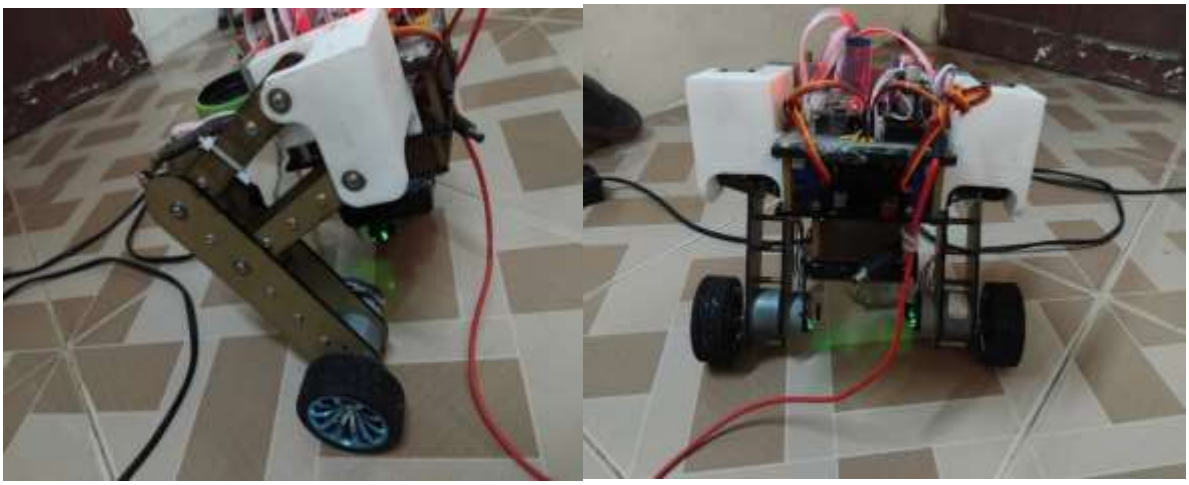


Figure 4.1 Completed Robot Model

+ Overall, the robot consists of three basic components: wheels, chassis, and a central controller.

b) Specifications of the Mobile Robot

Table 4.1 Specifications of the Mobile Robot

No.	Parameter	Description
1	Dimensions	250 x 280 mm
2	Speed	0,2m/s
3	Material	PVC Plastic - Mica - Copper
4	Voltage - Current	12V – 1A
5	Operating Time	3 hours

4.1.2. Graph Survey

- We conduct experiments with the proposed controller: in this model, the team uses wireless communication via Bluetooth to transmit and receive signals with the robot. This method allows us to monitor and control the device in a simple and easy manner.

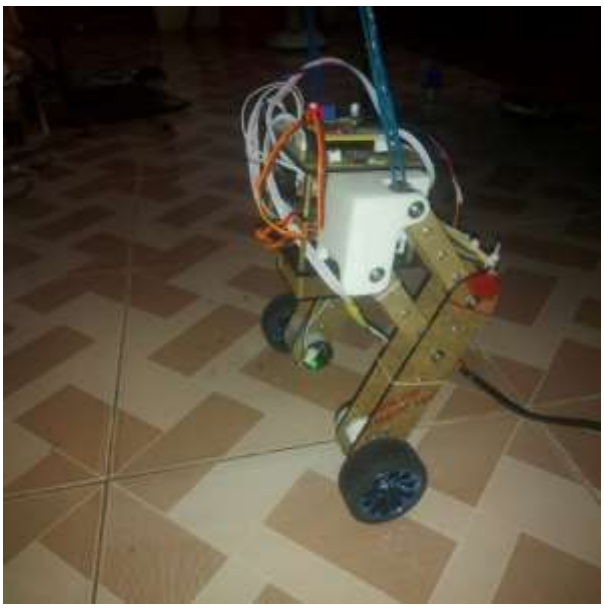
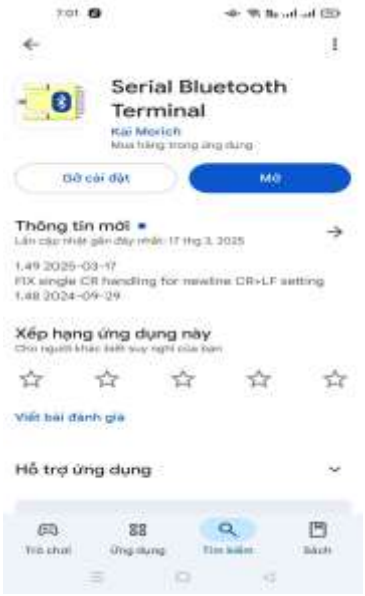


Figure 4.2 Measuring the Tilt Angle of the Vehicle

❖ Using a Phone for Monitoring



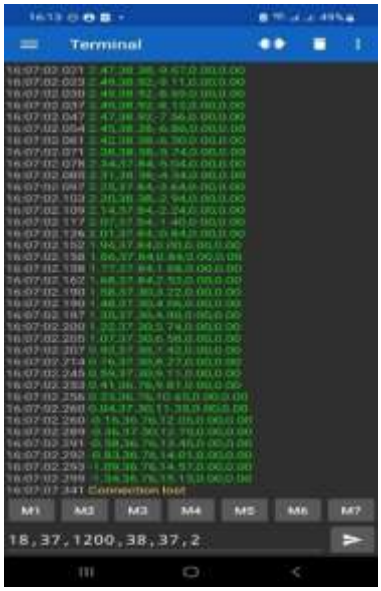


Figure 4.3 Monitoring and Adjusting the Weight K Application

The team extensively experimented with various weight sets to find the suitable parameters for the device, enabling the robot to balance on two wheels. The weight sets

tested during the robot's trials were recorded into the device's memory, ensuring that the latest weight set is retained even in case of power loss or shutdown.

4.2. Result

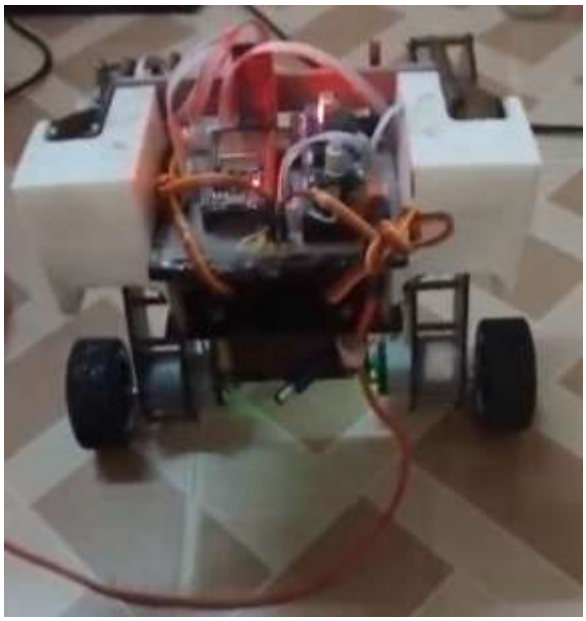


Figure 4.4 Robot balancing

❖ Robot testing:

Table 4.2: Experimental results

stt	K	Note
1	[12.5, 1.8, 18, 3.2, 1.1, 0.7]	The robot seems to wobble a lot and is not very stable.
2	[12.5, 1.8, 9, 3.2, 1.1, 0.7]	The robot is more stable and wobbles less, but it still doesn't seem to be balanced.
3	[15.0, 1.8, 18, 3.2, 1.1, 0.7]	The robot responds quickly, but still shows some shaking.
4	[10.0, 1.5, 15, 2.5, 0.9, 0.5]	The robot wobbles a lot and seems to lack enough force to maintain balance.
5	[12.5, 2.5, 18, 3.2, 1.1, 0.7]	The robot is fairly stable and responds quickly, but it can only stay balanced for a few seconds.

⇨ Comments: Based on the results, we can observe that the robot is attempting to return to the balanced position (0 degrees) due to the controller programmed and calculated by the team. However, there are still some fluctuations due to suboptimal

programming and the influence of noise, causing instability. Therefore, it can be concluded that the controller still needs optimization and greater accuracy to improve the system's response speed. Compared to simulations, it's evident that real-world challenges lead to discrepancies and issues that need to be addressed.

4.3. Product Evaluation

After researching and implementing the project, the team has drawn preliminary conclusions as follows:

- ❖ Advantages:
  - + Successfully constructed a variable-height robot model.
  - + Designed and implemented an LQR control algorithm for both practical and theoretical models.
  - + Verification on the practical model was successful.
  - + Data transmission time was reasonably stable.
- ❖ Disadvantages:
  - + The robot still fails to meet certain requirements, such as height adjustment and traversing thresholds, due to limitations in its balancing capability.
  - + Calculating the center of mass in reality is challenging. The team used SolidWorks software to determine it, but uncertainty remains.
  - + Real-world balancing of the model is still not as effective as in theoretical simulations.

Programming limitations and errors persist

4.4. Development Directions

Figure 4.5 Robot Competition Model of the Chinese National Team 2023

- Based on the knowledge gained from reading various articles and documents during the project, as well as practical experience, the team has identified several areas for improvement in the future:
    - Optimize Control Program: Instead of controlling the robot's tilt angles directly, focus on controlling the position of the vehicle and the two tilt angles to create specific motions for the robot.
    - Incorporate Frictional Force Calculations: Calculate the frictional force between the wheels and the surface to provide enough force for the robot to jump.
    - Enhance Programming Skills: Improve programming skills and knowledge, as there are still limitations in this area that need to be addressed.
    - Upgrade Hardware: Consider upgrading hardware components, such as replacing servos with brushless motors equipped with encoders, for improved performance.
    - Integrate AI to enable the robot to detect and avoid obstacles, enhancing its autonomous navigation capability.

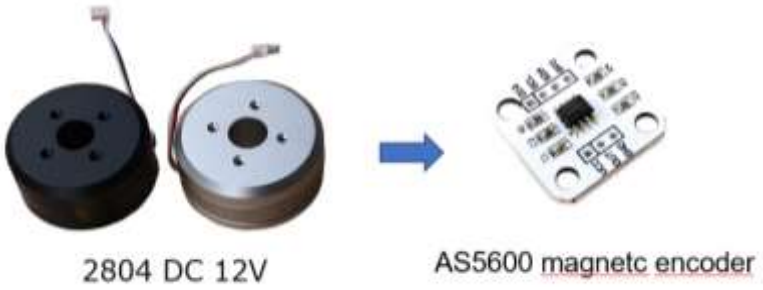


Figure 4.6 Encoder Fabrication for the Brushless DC Motor (BLDC)

REFERENCE

[1] Hsu, C.-F.; Chen, B.-R.; Lin, Z.-L. “Implementation and Control of a Wheeled Bipedal Robot Using a Fuzzy Logic Approach”. Actuators 2022, 11, 357. https://doi.org/10.3390/ act11120357

[2] Xujiong Feng, Shuaishuai Liu, Qiang Yuan, Junbo Xiao, Daxu Zhao. “Research on wheel-legged robot based on LQR and ADRC” IEEE Trans. Ind. Electron 2022. https://doi.org/10.1038/s41598-023-41462-1

[3] Ali Unluturk and Omer Aydogdu “Adaptive control of two-wheeled mobile balance robot capable to adapt different surfaces using a novel artificial neural network–based real-time switching dynamic controller” Doi: 10.1177/1729881417700893

[4] K. M. Lynch and F. C. Park, ‘‘Modern robotics” Cambridge University Press, 2017

[5] “Robot công nghiệp” – Phạm Đăng Phước. Xuất bản năm 2007

[6] Y. Tazaki and M. Murooka, "A survey of motion planning techniques for humanoid robots," Advanced Robotics, vol. 34, no. 21-22, pp. 1370-1379, 2020.

[7] V. Klemm et al., "LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3745-3752, 2020.

[8] J. Dong, R. Liu, L. Biao, X. Guo, and H. Liu, "LQR-based balance control of two-wheeled legged robot," in 2022 41st Chinese Control Conference (CCC), 2022: IEEE, pp. 450-455

[9] W. Zhu, F. Raza, and M. Hayashibe, "Reinforcement learning based hierarchical control for path tracking of a wheeled bipedal robot with sim-to-real framework," in 2022 IEEE/SICE International Symposium on System Integration (SII), 2022: IEEE, pp. 40-46.

[10] L. Cui et al., "Learning-based balance control of wheel-legged robots," IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 7667-7674, 2021.

[11] Q. D. Le and H.-J. Kang, "Finite-time fault-tolerant control for a robot manipulator based on synchronous terminal sliding mode control," Applied Sciences, vol. 10, no. 9, p. 2998, 2020.

[12] V. Klemm et al., "Ascento: A two-wheeled jumping robot," in 2019 International Conference on Robotics and Automation (ICRA), 2019: IEEE, pp. 7515-7521

[13] M. T. Dat, "Xe hai bánh tự cân bằng di chuyển trên địa hình phẳng," 2005

[14] N. G. M. Thao, D. H. Nghia, and N. H. Phuc, "A PID backstepping controller for two-wheeled self-balancing robot," in International Forum on Strategic Technology 2010, 2010: IEEE, pp. 76-81

[15] A. K. Vo, H. T. Nguyen, V. D. H. Nguyen, M. T. Nguyen, and T. T. H. Le, "Trajectory Tracking Pid-Sliding Mode Control for Two-Wheeled Self-Balancing Robot," in Intelligent Computing in Engineering: Select Proceedings of RICE 2019, 2020: Springer, pp. 885-898

APPENDIX

❖ Drawing

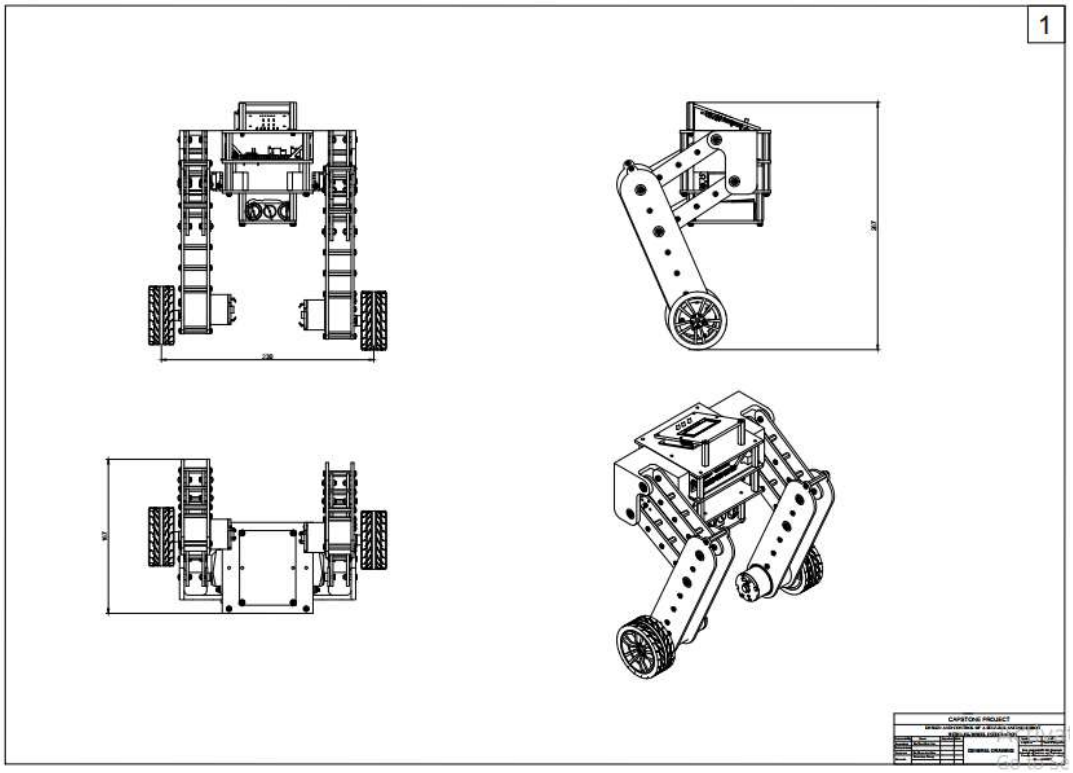


Figure 1 General Drawing

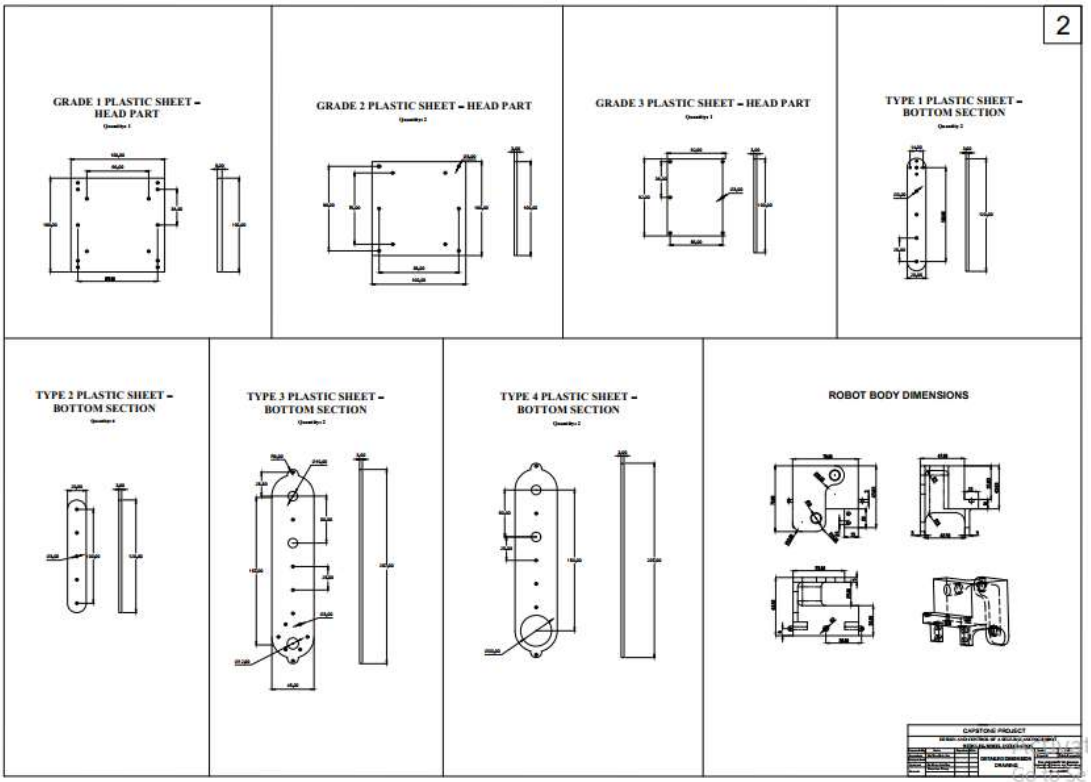


Figure 2 Detailed Dimension Drawing

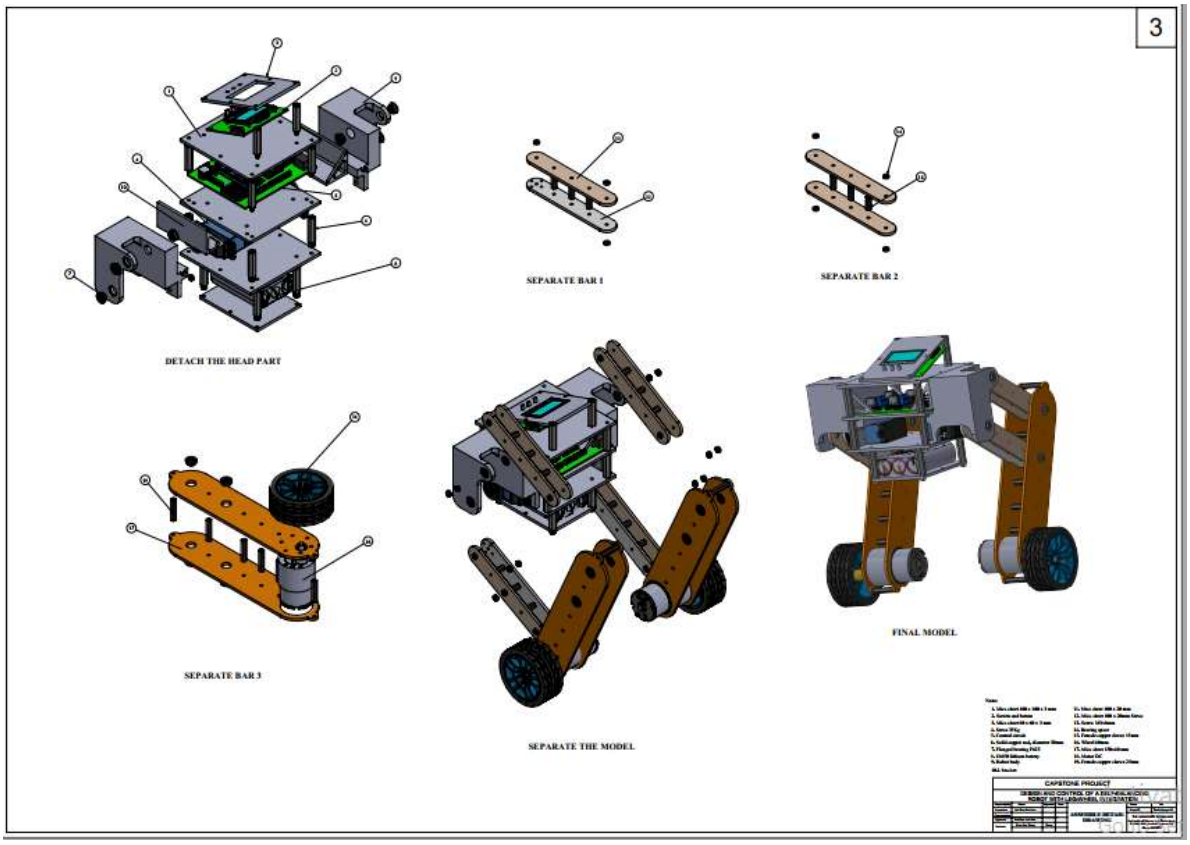


Figure 3 Assembly Detail Drawing

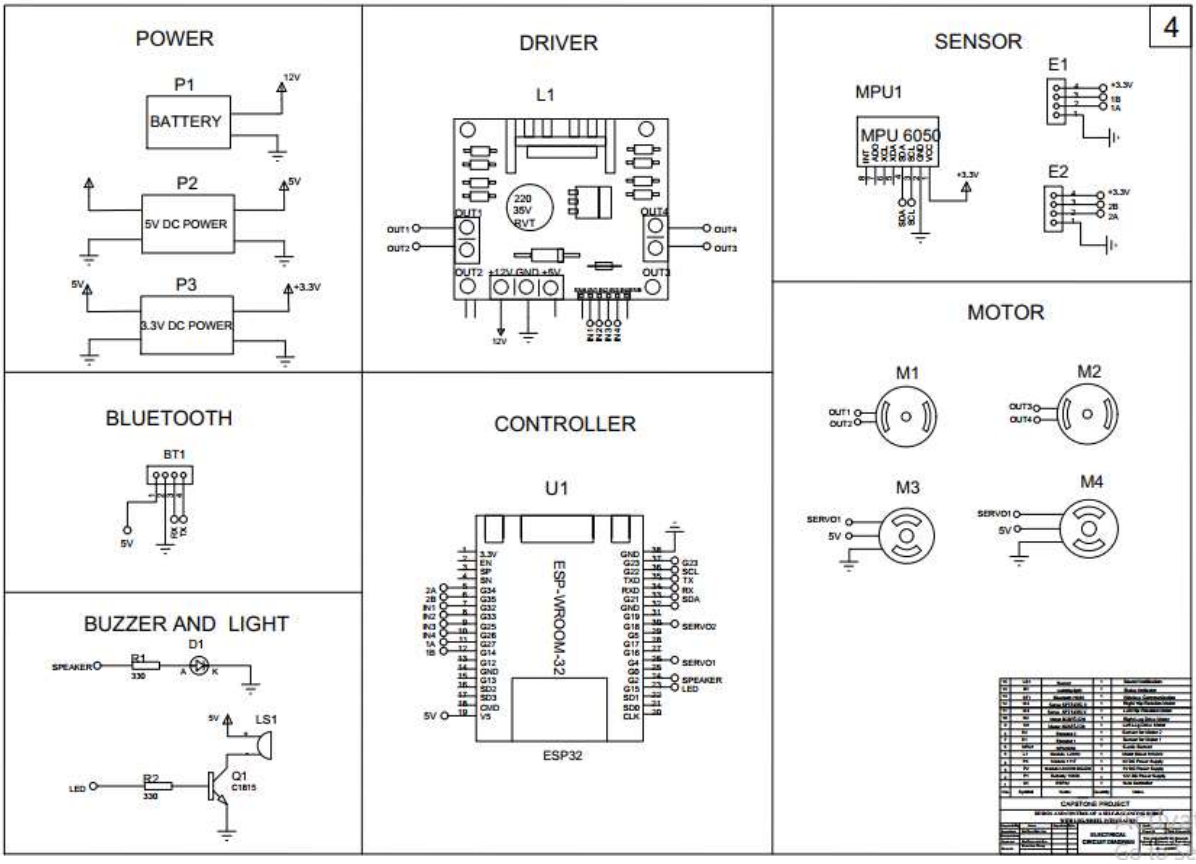


Figure 4 Electrical Circuit Diagram

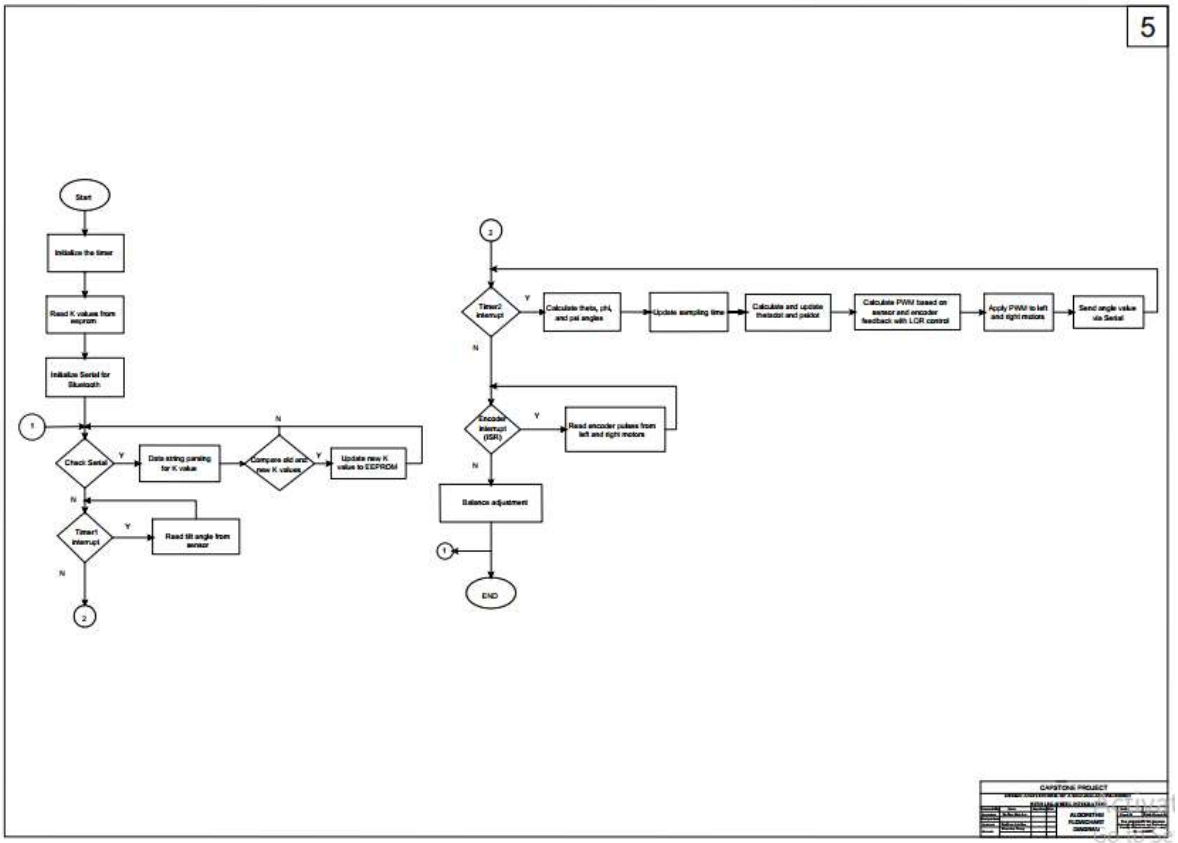


Figure 5 Algorithm Flowchart Diagram

❖ Code Robot for ESP32

```
#include <Arduino.h>
#include <Wire.h>
#include "BluetoothSerial.h"
#include "encoder.h"
#include "motorControl.h"
#include "MPU6050.h"
#include <EEPROM.h>
#define EEPROM_SIZE 24 // Each float is 4 bytes, 6 floats * 4 bytes = 24 bytes
#define ToRad PI / 180
#define ToDeg 180 / PI
#define factortheta PI / 20
#define factorphi PI / 180
#define LED 15
#define LOA 2
//-------------------------------------Define Pin_Mode-------------------------------------------
#define IN1 32 // 32
#define IN2 33 // 33
#define IN3 25 // 25
#define IN4 26 // 26
#define CH_SERVO1 4
#define CH_SERVO2 18
```

```
   #define ENCODER_1A 27 //
   #define ENCODER_1B 14 //
   #define ENCODER_2A 34 //
   #define ENCODER_2B 35 //

   #define PRECISION 10
   #define FREQUENCE 12000 // 12 kHz PWM, 12-bit resolution
   #define PWM_MAX 1024
   #define PWM_MIN -1024
   #define CHANNEL_1 1
   #define CHANNEL_2 2
   #define CHANNEL_3 3
   #define CHANNEL_4 4
   //-------------------------------------Define  Variable-------------------
-----------------------------------------
   float leftvolt; // output volt left motor in LQR
   float righvolt; // output volt right motor in lQR

   int channel_PWM1 = 3; // su dung 2 kenh de kiem soat huong va toc do cua dong
co , // PWM voi 8 kenh dau voi tan so 80MHz (high-speed channel), và 8 kênh cuối
clock 1MHz (low-speed channel).
   int channel_PWM2 = 4; // co 16 kenh ledc co san, ta co the lua chon bat ki
tu 0 - 15
   int channel_PWM3 = 5;
   int channel_PWM4 = 6;
   // bien toan cuc
   long PWML = 0, PWMR = 0; // Tinh so xung can cap cho banh trai va phai
   float k1, k2, k3, k4, k5, k6;
   bool falldown;
   float mpudata = 0;
   float theta = 0, psi = 0, phi = 0;
   float thetadot, psidot, phidot;
   float thetaold, psiold, phiold;
   float addtheta;
   float addphi;
   int ForwardBack;
   int LeftRight;
   // Khoi tao Timer
   uint32_t timer;
   int inChar;
   String myString;
   unsigned long prevTime_T1;
   // bien tao xung cho DC
   int freq_PWM = 5000;    // 10khz pwm xac dinh khoang thoi gian
   int resolution_PWM = 8; // 2^10 + 1 = 1025    // Dat do phan giai cua PWM 8,
10, 12 BIT
   BluetoothSerial SerialBT;
```

```
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable
it
#endif
String pass = "12345678";
String Data;
void Loa_setup()
{
  pinMode(LED, OUTPUT);
  pinMode(LOA, OUTPUT);
  digitalWrite(LED, LOW);
  digitalWrite(LOA, LOW);

  // Phát âm thanh khi khởi động
  digitalWrite(LOA, HIGH);
  delay(500);
  digitalWrite(LOA, LOW);
  delay(500);
  digitalWrite(LOA, HIGH);
  delay(500);
  digitalWrite(LOA, LOW);
  delay(500);
  digitalWrite(LOA, HIGH);
  delay(500);
  digitalWrite(LOA, LOW);
}
void docgiatri_eeprom()
{
  EEPROM.begin(EEPROM_SIZE);
  // Đọc giá trị K từ EEPROM
  k1 = EEPROM.readFloat(0);
  k2 = EEPROM.readFloat(4);
  k3 = EEPROM.readFloat(8);
  k4 = EEPROM.readFloat(12);
  k5 = EEPROM.readFloat(16);
  k6 = EEPROM.readFloat(20);
}
void setup()
{
  Serial.begin(115200);
  SerialBT.begin("BKTECH GROUP"); // Khoi tao kết nối Bluetooth
  Loa_setup();
  docgiatri_eeprom();
  pinMode(ENCODER_1A, INPUT_PULLUP);
  pinMode(ENCODER_1B, INPUT_PULLUP);
  pinMode(ENCODER_2A, INPUT_PULLUP);
  pinMode(ENCODER_2B, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENCODER_1A), encoder1_isr, RISING);
```

```
    attachInterrupt(digitalPinToInterrupt(ENCODER_2A), encoder2_isr, RISING);
    ledcSetup(channel_PWM1, freq_PWM, resolution_PWM);
    ledcAttachPin(IN1, channel_PWM1);
    ledcSetup(channel_PWM2, freq_PWM, resolution_PWM);
    ledcAttachPin(IN2, channel_PWM2);
    ledcSetup(channel_PWM3, freq_PWM, resolution_PWM);
    ledcAttachPin(IN3, channel_PWM3);
    ledcSetup(channel_PWM4, freq_PWM, resolution_PWM);
    ledcAttachPin(IN4, channel_PWM4);
    setup_MPU6050();
    // Set factor of K maxtrix
    // K = | k1   k2   k3   k4   k5   k6 |
    //     | k1   k2   k3   k4  -k5  -k6 |
    // k1 = 12.5;                          //k1*theta
    // k2 = 1.8;                          //k2*thetadot
    // k3 = 18;                          //k3*psi
    // k4 = 3.2;                          //k4*psidot
    // k5 = 1.1;                          //k5*phi
    // k6 = 0.7;                           //k6*phidot
    ForwardBack = 0, LeftRight = 0, addphi = 0, addtheta = 0, mpudata = 0,
theta = 0, psi = 0, phi = 0;
  }
  void readMPU()
  {
    gyro_signals();
    RateRoll -= RateCalibrationRoll;
    RatePitch -= RateCalibrationPitch;
    RateYaw -= RateCalibrationYaw;
       kalman_1d(KalmanAngleRoll,    KalmanUncertaintyAngleRoll,    RateRoll,
AngleRoll);
    KalmanAngleRoll = Kalman1DOutput[0];
    mpudata = KalmanAngleRoll;
    KalmanUncertaintyAngleRoll = Kalman1DOutput[1];
      kalman_1d(KalmanAnglePitch,   KalmanUncertaintyAnglePitch,   RatePitch,
AnglePitch);
    KalmanAnglePitch = Kalman1DOutput[0];
    KalmanUncertaintyAnglePitch = Kalman1DOutput[1];
    while (micros() - LoopTimer < 4000)
      ;
    LoopTimer = micros();
    if (abs(mpudata) > 30)
    {
      falldown = true;
      leftencoder = 0;
      rightencoder = 0;
      // Reset zero setpoint
      addtheta = 0;
      addphi = 0;
```

```
    }
    else
    {
      falldown = false;
    }
  }
  // Read theta angle function//
  float gettheta(long lencoder, long rencoder)
  {                                                          // deg value
    float angle = (0.5 * 360 * (lencoder + rencoder)) / 333; //// 0.5*360/333
    return angle;
  }
  // Read phi angle function//
  float getphi(long lencoder, long rencoder)
  {                                                    // deg value
    float angle = (32.5 / 232) * (lencoder - rencoder); //(R/W) (32.5/232)
    return angle;
  }
  // LQR function//
  void getlqr(float theta_, float thetadot_, float psi_, float psidot_, float
phi_, float phidot_)
  {
    leftvolt = k1 * theta_ + k2 * thetadot_ + k3 * psi_ + k4 * psidot_ - k5 *
phi_ - k6 * phidot_;
    righvolt = k1 * theta_ + k2 * thetadot_ + k3 * psi_ + k4 * psidot_ + k5 *
phi_ + k6 * phidot_;

    PWML = map(leftvolt, -(k3 * PI) / 15, (k3 * PI) / 15, -250, 250); // Limit
15 deg.
    PWMR = map(righvolt, -(k3 * PI) / 15, (k3 * PI) / 15, -250, 250);

    PWML = constrain(PWML, -240, 240); // limit pwm value in (-240, 240) because
we using high frequency pwm (31 khz)
    PWMR = constrain(PWMR, -240, 240);
  }
  void serialEvent()
  {
    while (Serial.available() > 0)
    {
      inChar = Serial.read();
    }
    // Control motor forward
    if (inChar == 70) // F -->run
    {
      ForwardBack = 1;
    }
    if (inChar == 84) // T -->stop
    {
```

```
      ForwardBack = 0;
    }
    // Control motor Back
    if (inChar == 66) // B
    {
      ForwardBack = -1;
    }
    if (inChar == 86) // V
    {
      ForwardBack = 0;
    }
    // Control motor Left
    if (inChar == 76) // L
    {
      LeftRight = 1;
    }
    if (inChar == 85) // U
    {
      LeftRight = 0;
    }
    // Control motor right
    if (inChar == 82) // R
    {
      LeftRight = -1;
    }
    if (inChar == 79) // O
    {
      LeftRight = 0;
    }
  }
  void parseData(String data)
  {
    int index1 = data.indexOf(',');              // Find the first comma
    int index2 = data.indexOf(',', index1 + 1); // Find the second comma
    int index3 = data.indexOf(',', index2 + 1); // Find the third comma
    int index4 = data.indexOf(',', index3 + 1); // Find the fourth comma
    int index5 = data.indexOf(',', index4 + 1); // Find the fifth comma

    if (index1 > 0 && index2 > index1 && index3 > index2 && index4 > index3 &&
index5 > index4)
    { // Ensure that all commas are found
      float newK1 = data.substring(0, index1).toFloat();
      float newK2 = data.substring(index1 + 1, index2).toFloat();
      float newK3 = data.substring(index2 + 1, index3).toFloat();
      float newK4 = data.substring(index3 + 1, index4).toFloat();
      float newK5 = data.substring(index4 + 1, index5).toFloat();
      float newK6 = data.substring(index5 + 1).toFloat();
```

```
        // Check if any value has changed
        if (newK1 != k1 || newK2 != k2 || newK3 != k3 || newK4 != k4 || newK5 != 
k5 || newK6 != k6)
        {
          k1 = newK1;
          k2 = newK2;
          k3 = newK3;
          k4 = newK4;
          k5 = newK5;
          k6 = newK6;

          // Save new values to EEPROM
          EEPROM.writeFloat(0, k1);
          EEPROM.writeFloat(4, k2);
          EEPROM.writeFloat(8, k3);
          EEPROM.writeFloat(12, k4);
          EEPROM.writeFloat(16, k5);
          EEPROM.writeFloat(20, k6);
          EEPROM.commit();

          // Print values to verify
          Serial.print("k1: ");
          Serial.println(k1);
          Serial.print("k2: ");
          Serial.println(k2);
          Serial.print("k3: ");
          Serial.println(k3);
          Serial.print("k4: ");
          Serial.println(k4);
          Serial.print("k5: ");
          Serial.println(k5);
          Serial.print("k6: ");
          Serial.println(k6);
        }
      }
      else
      {
        Serial.println("Invalid data format");
      }
    }

    void loop()
    {
      while (SerialBT.available())
      {
        char receivedChar = SerialBT.read(); // Read a single character

        if (receivedChar == '\n')
```

```
      {                     // Check if it's the end of the message
        parseData(myString); // Parse the received message
        myString = "";       // Reset the string for the next message
      }
      else
      {
        myString += receivedChar; // Accumulate characters until newline
      }
    }
    readMPU();
    if ((micros() - prevTime_T1) > 6000)
    {                                                    // Set time loop
update and control motor
      theta = gettheta(leftencoder, rightencoder) * ToRad; // Read theta value
and convert to Rad
      psi = (mpudata)*ToRad;                            // Read psi value
and convert to Rad
      phi = getphi(leftencoder, rightencoder) * ToRad;     // Read phi value
and convert to Rad

      // Update time compare with timeloop
      float dt = (float)(micros() - prevTime_T1) / 1000000.0;
      prevTime_T1 = micros();
      // Update input angle value
      thetadot = (theta - thetaold) / dt;
      psidot = (psi - psiold) / dt;
      phidot = (phi - phiold) / dt;
      // Upadte old angle value
      thetaold = theta;
      psiold = psi;
      phiold = phi;
      //
      addtheta = addtheta + ForwardBack * factortheta;
      addphi = addphi + LeftRight * factorphi;

      getlqr(theta + addtheta, thetadot, psi, psidot, phi + addphi, phidot);
      motorControl(PWML, PWMR, (mpudata), falldown);
      String S = "";
       // S = (String)(psi*ToDeg) + ',' + (String)(theta*ToDeg) + ',' +
(String)(phi*ToDeg) + ',' + (String)(-addtheta*ToDeg) + ',' + (String)(-
addphi*ToDeg)+   ','+   (String)(PWML)   +   ','   +   (String)(PWMR)+','+
(String)(KalmanAnglePitch) + ',' + (String)(k3);
       // S = (String)(psi*ToDeg) + ',' + (String)(theta*ToDeg) + ',' +
(String)(phi*ToDeg) + ',' + (String)(-addtheta*ToDeg) + ',' + (String)(-
addphi*ToDeg);
      S = (String)(k1) + ',' + (String)(k2) + ',' + (String)(k3) + ',' +
(String)(k4) + ',' + (String)(psi * ToDeg) + ',' + (String)(PWML) + ',' +
(String)(PWMR);
```

```
        SerialBT.println(S);
        // Serial.println(S);
      }
      // motorControl(100, 100,(0 + 0), false);
    }

    # include "motorControl.h"
    void Leftmotor(int dir, uint8_t pwmValue){
        if(dir == 1){
            ledcWrite(channel_PWM1, pwmValue);
            ledcWrite(channel_PWM2, 0);
        }
        else if(dir == -1){
            ledcWrite(channel_PWM1,0);
            ledcWrite(channel_PWM2,pwmValue);
        }
    }
    void Rightmotor(int dir, uint8_t pwmValue){
        if(dir == 1){
            ledcWrite(channel_PWM3, pwmValue);
            ledcWrite(channel_PWM4, 0);
        }else if(dir == -1){
            ledcWrite(channel_PWM3,0);
            ledcWrite(channel_PWM4,pwmValue);
        }
    }
    void StopMotor(){
        ledcWrite(channel_PWM1,0);
        ledcWrite(channel_PWM2,0);
        ledcWrite(channel_PWM3,0);
        ledcWrite(channel_PWM4,0);
    }
    void motorControl(long leftPWM, long rightPWM, float angle_psi, bool
stopstate){
        if(stopstate){
            StopMotor();
        }else{
            if(abs(angle_psi) > 30){
                StopMotor();
            }
            else{
                if(leftvolt > 0){
                    Leftmotor(1, abs(leftPWM));
                }else if(leftvolt < 0){
                    Leftmotor(-1, abs(leftPWM));
                }else{
                    ledcWrite(channel_PWM1,0);
                    ledcWrite(channel_PWM2,0);
```

```
            }

            if(righvolt > 0){
                Rightmotor(1, abs(rightPWM));
            }else if(righvolt < 0){
                Rightmotor(-1, abs(rightPWM));
            }else{
                ledcWrite(channel_PWM3,0);
                ledcWrite(channel_PWM4,0);
            }
        }
    }
}

#include "encoder.h"
volatile long leftencoder = 0;      // position update by encoder
volatile long rightencoder = 0;      // position update by encoder
void encoder1_isr(){;
    int xung_1B = digitalRead(ENCODER_1B);
    if(xung_1B > 0){
        leftencoder++;
    }
    else{
        leftencoder--;
    }
}

void encoder2_isr(){;
    int xung_2B = digitalRead(ENCODER_2B);
    if(xung_2B >0){
        rightencoder++;
    }
    else{
        rightencoder--;
    }
}
```