# Model Reference Adaptive Control Design for Self Balancing Robot

Project report submitted in partial fulfillment
of the requirements for the degree of

*Bachelor of Technology*
*in*
*Electronics and Communication Engineering*

by


Divyansh Jain - 20UEC049
Satvik Sharma - 20UEC116
Sayan Chatterjee - 20UEC120

Under Guidance of
Dr. Bharat Verma

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur


November 2023

The LNM Institute of Information Technology
Jaipur, India

## CERTIFICATE

This is to certify that the project entitled “Model Reference Adaptive Control Design for Self Balancing Robot” , submitted by Divyansh Jain (20uec049), Satvik Sharma (20uec116) and Sayan Chatterjee (20uec120) in partial fulfillment of the requirement of degree in Bachelor of Technology (B. Tech), is a bonafide record of work carried out by them at the Department of Electronics and Communication Engineering, The LNM Institute of Information Technology, Jaipur, (Rajasthan) India, during the academic session 2023-2024 under my supervision and guidance and the same has not been submitted elsewhere for award of any other degree. In my/our opinion, this report is of standard required for the award of the degree of Bachelor of Technology (B. Tech).

Date

Adviser: Dr. Bharat Verma

# Acknowledgments

We would like to express our sincere gratitude to all those who have supported and encouraged us throughout our B.Tech project.
We would like to thank our project supervisor 'Dr. Bharat Verma', for his invaluable guidance, advice, and support throughout the project. His insights and expertise were invaluable in shaping our ideas and helping us overcome technical challenges. His feedback and suggestions were instrumental in improving the quality of our work.

# Abstract

This presentation provides an overview of the design and implementation of a self-balancing robot with a PID as well as MRAC controller design. The project aims to develop a two-wheeled robot that is capable of maintaining its balance in an upright position. The robot is built using off-the-shelf components, including an Arduino UNO microcontroller, Simulink for modelling and simulation, and an L298N motor driver to control the motors.

The design process involves several stages. First, the mechanical structure of the robot is constructed using a stacked multilevel chassis, DC motors and two wheels. The Arduino microcontroller is then programmed with the various controller algorithms, which uses the sensor data from an accelerometer and gyroscope to calculate the control signals for the motors. The Simulink model is then deployed to the Arduino for real-time control of the robot.

Experimental results demonstrate the successful implementation of the self-balancing robot. The robot is able to maintain its balance in an upright position even when subjected to external disturbances, such as pushes or tilting of the chassis. The control algorithm effectively adjusts the motor speeds and direction of rotation to maintain balance by continuously monitoring the sensor data and making appropriate corrections.

The proposed self-balancing robot has potential applications in various fields, including robotics, automation, and transportation. It can be used as a platform for further research and development of advanced control algorithms for self-balancing systems. The project also provides valuable insights into the practical implementation of adaptive control designs on a real-time embedded system. Overall, the project serves as a valuable learning experience in robotics, control systems, and embedded system design, with potential for further exploration and application in various domains.

# Contents


**List of Figures** vii

**List of Tables** viii

**1 Work Done Previously** 1

1.1 PID Designing . . . . 1

**2 Introduction to Model Reference Adaptive Control** 4

2.1 The Area of Work . . . . 4

2.2 Advantages of Two-Wheeled Self Balancing Systems . . . . 4

2.3 Components Used . . . . 5

**3 Working** 7

3.1 Theory of Controller . . . . 7

3.2 System Designing . . . . 8

**4 Simulation and Results** 9

**5 Conclusions and Future Work** 14

5.1 Future Work . . . . 14

5.2 Conclusion . . . . 15

**Bibliography** 15

# List of Figures

1.1 Simulink Model . . . . . . . . . . 2
1.2 Simulink Subsystem . . . . . . . . . . 3

2.1 Old Hardware Model . . . . . . . . . . 5
2.2 Redesigned Hardware Model . . . . . . . . . . 6

4.1 Balanced Model . . . . . . . . . . 9
4.2 Balanced Model(new) . . . . . . . . . . 10
4.3 Simulink Model . . . . . . . . . . 11
4.4 Simulink Subsystem . . . . . . . . . . 11
4.5 MATLAB LiveScript . . . . . . . . . . 12
4.6 Control Action Generated . . . . . . . . . . 12
4.7 Learning Rates . . . . . . . . . . 13

5.1 Simulink Model . . . . . . . . . . 14
5.2 Simulink Subsystem . . . . . . . . . . 15

# List of Tables

1.1 PID values for various lambda . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

# Chapter 1

# Work Done Previously

## 1.1 PID Designing

Earlier we designed the system chassis using different fibre plates and were able to balance it on its own using PID control design. The model is assumed to be a first order unstable system with its transfer function given as:

$$G(s) = \frac{Ke^{-\theta s}}{\tau s - 1} \tag{1.1}$$

where the values are found out to be K = 800 is the Gain, $\theta$ = 0.02 is the Delay and $\tau$ = 1000 is the time constant.
We are using PID controller for our model and we have used the IMC-PID controller technique to design the PID.

We are using a first order IMC filter for which the transfer function is given as:

$$F(s) = \frac{\alpha s + 1}{\lambda s + 1} \tag{1.2}$$

The final equation for the IMC-PID design technique will look like:

$$C(s) = \frac{(\alpha s + 1)(\theta s + 2)}{Ks(2\lambda - 2\alpha + 2 * \theta)} \tag{1.3}$$

The controller can be expressed in PID form as:

$$C(s) = k_c(1 + \frac{1}{\tau_i s} + \tau_d s) \tag{1.4}$$

The values of the variable are given as:

$$\alpha = \frac{2 * \lambda * \tau + \theta * \lambda + 2 * \tau * \theta}{2 * \tau - \theta} \tag{1.5}$$

$$k_c = \frac{2\alpha + \theta}{2 * K(\alpha - \lambda - \theta)} \tag{1.6}$$

$$\tau_i = \alpha + \frac{\theta}{2} \tag{1.7}$$

$$\tau_d = \frac{\theta\alpha}{2\alpha + \theta} \tag{1.8}$$

and the values of Kp, Ki and Kd are given by:

1. $K_p = k_c$
2. $K_i = \frac{k_c}{\tau_i}$
3. $K_d = k_c * \tau_d$

Now we will be finding the values of 'Proportional Gain ($K_p$)', 'Integral gain ($K_i$)', and 'Derivative gain ($K_d$)' for different values of $\lambda$.
Finally we will be using that values of control parameters on which the system (that is the robot) is giving the best and stable output.

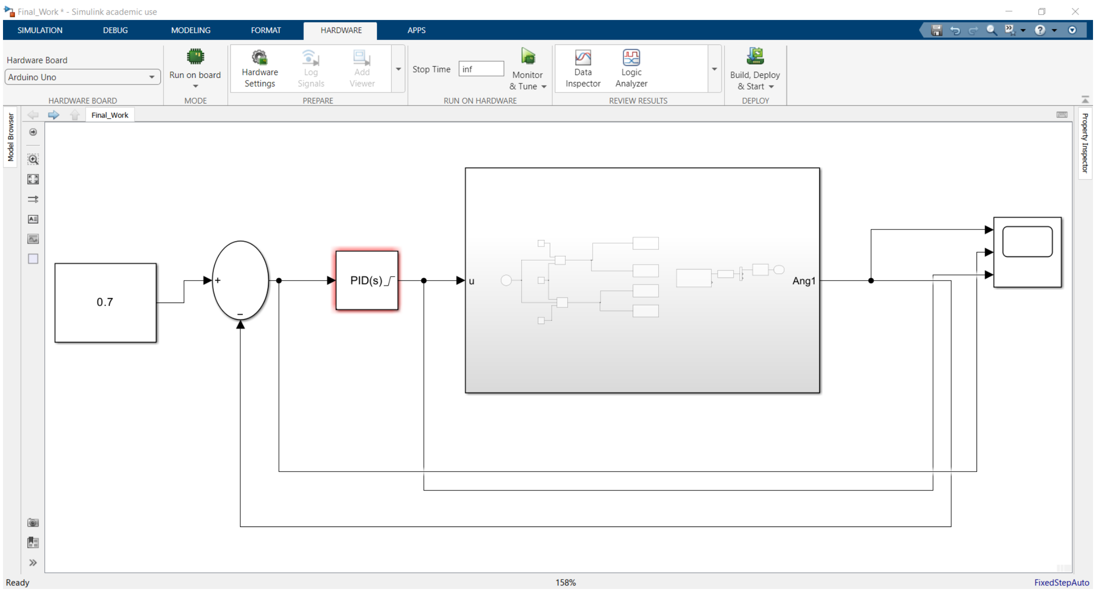


FIGURE 1.1: Simulink Model

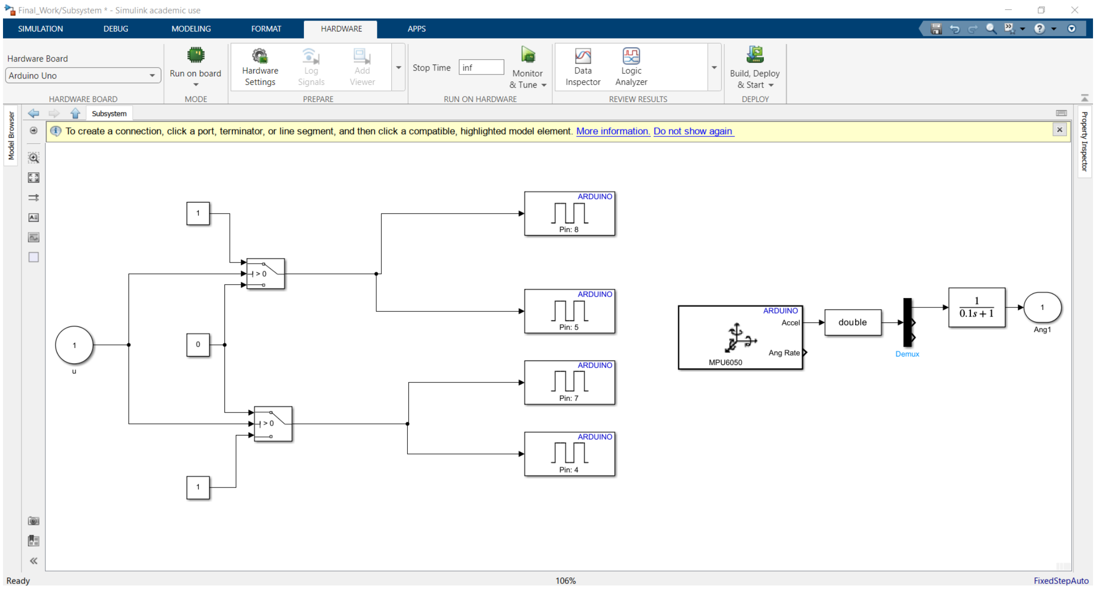

FIGURE 1.2: Simulink Subsystem

| Lambda Values | Kp | Ki | Kd |
|---|---|---|---|
| 0.2 | 104.8828 | 22.5520 | 0.5122 |
| 0.7 | 101.4234 | 72.5249 | 0.5036 |
| 1 | 101.0000 | 102.5251 | 0.5025 |
| 2 | 100.5037 | 202.5352 | 0.5013 |
| 4 | 100.2547 | 402.5627 | 0.5006 |

TABLE 1.1: PID values for various lambda

# Chapter 2

# Introduction to Model Reference Adaptive Control

## 2.1 The Area of Work

Control design in the realm of self-balancing robots involves the application of various algorithms and techniques to enable a robot to maintain its balance and stability while in motion. This area of work primarily explores control systems and their response generated on the basis of the deviation of the behaviour real life systems from an ideal reference system.

## 2.2 Advantages of Two-Wheeled Self Balancing Systems

- Maneuverability: Two-wheeled systems typically have a smaller footprint, allowing them to navigate tight spaces more efficiently than four-wheeled systems.
- Energy Efficient: They require lesser power to operate than their 4-wheeled counterparts.
- Compact size: They require fewer components and are therefore lesser in size and weight.
- Agility and Speed: The lesser weight promotes better speeds.
- Climbing difficult terrains: They can adapt more easily to uneven or incline terrains due to their efficient balance control mechanisms.

## 2.3 Components Used

- Arduino Uno
- Motor Driver IC (L293D)
- Gyroscope (MPU 6050)
- DC Motors
- Lithium Polymer Battery
- Stacked Parallel Plate Chassis
- Wheels

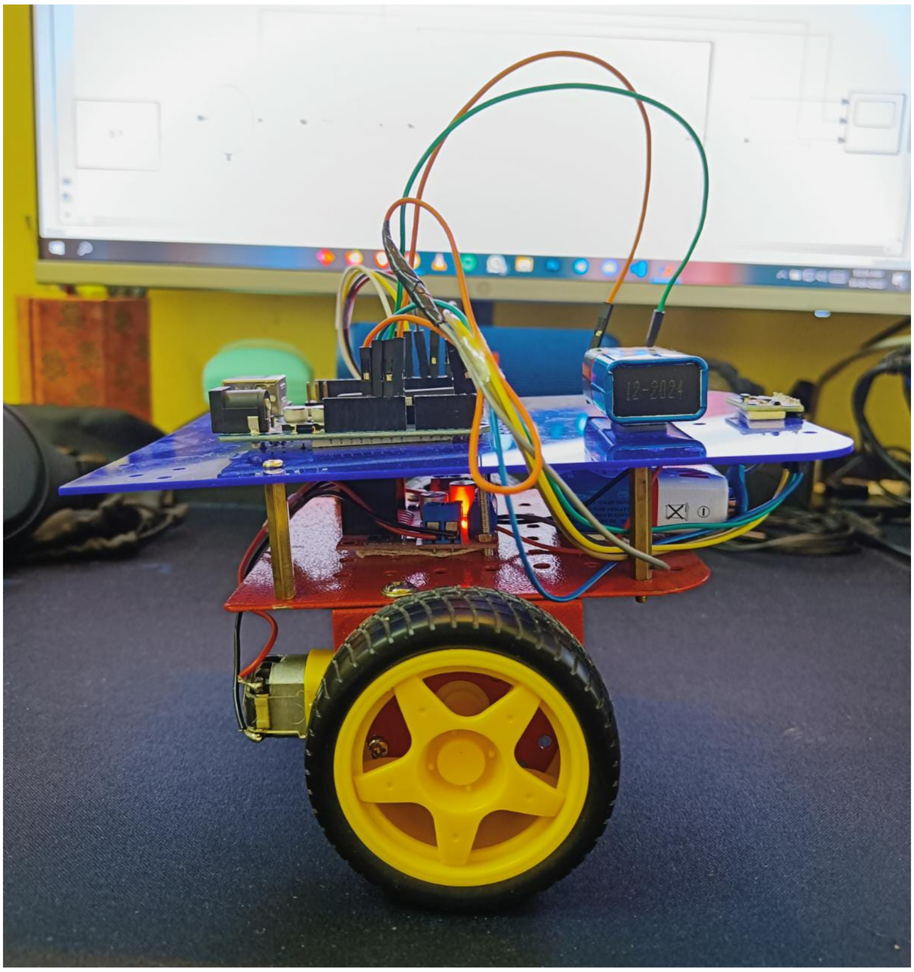


FIGURE 2.1: Old Hardware Model

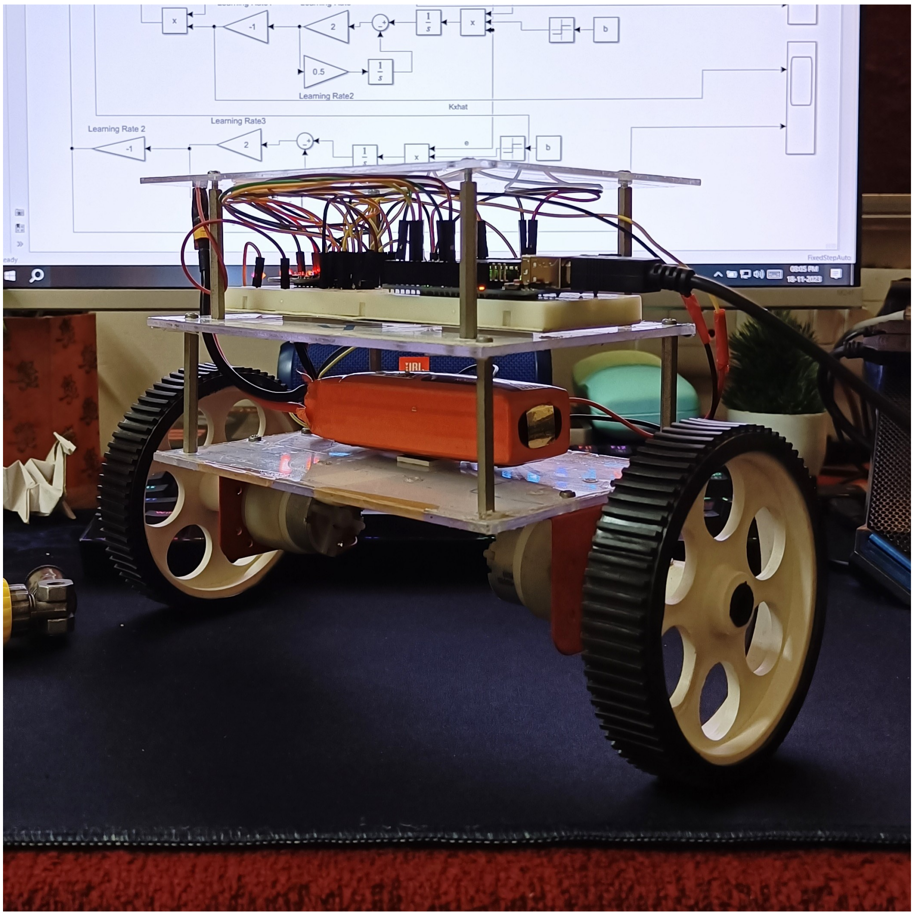


FIGURE 2.2: Redesigned Hardware Model

# Chapter 3

# Working

## 3.1 Theory of Controller

Model Reference Adaptive Control (MRAC) is a control technique that aims to make the output of a plant (the process or system being controlled) track a reference model, even when the plant dynamics are unknown or changing. MRAC is often used in situations where a traditional Proportional-Integral-Derivative (PID)controller may not be sufficient to achieve the desired performance.
The implementation of MRAC after a PID controller typically involves the following steps:

- Define the reference model: The reference model represents the desired behavior of the plant. It is typically a linear or nonlinear mathematical model that describes the ideal output of the plant. The reference model should be chosen based on the desired performance criteria.

- Estimate the plant parameters: In order to design an MRAC controller, the parameters of the plant (e.g. the transfer function, time constants, etc.) need to be estimated. This can be done using various techniques, such as system identification or parameter estimation.

- Design the adaptive controller: The adaptive controller is designed to modify the control inputs to the plant in order to make the plant output track the reference model. The adaptive controller typically consists of two parts: a feedback controller (such as a PID controller) and an adaptation law.

- Tune the controller gains: The gains of the feedback controller and the adaptation law need to be tuned to ensure that the control system is stable and provides the desired performance. This can be done using various techniques, such as simulation or experimental testing.

- Implement the controller: The controller is implemented in hardware or software, and the system is tested to ensure that it meets the desired performance criteria.

The key advantage of MRAC is that it can provide robust performance even in the presence of uncertainties and variations in the plant dynamics. This makes it a useful tool for controlling complex and dynamic systems. However, MRAC requires accurate modeling of the plant and careful tuning of the controller gains,which can be challenging in practice.

## 3.2 System Designing

- The Process Model : The subsystem model for motor control and gyroscope reading were created using the appropriate blocks.
- The Reference Model : The algorithm uses Direct MRAC Method. So the ideal reference model was defined by the transfer function -

$$G(s) = \frac{1}{0.05s + 1} \tag{3.1}$$

- Designing the Controller : The adaptive controller was designed to run the motors in either direction depending on the gyro reading in order to achieve the output similar to the reference model.
- Tuning : The reference values and learning rated were tuned such as to give the best possible response.

# Chapter 4

# Simulation and Results

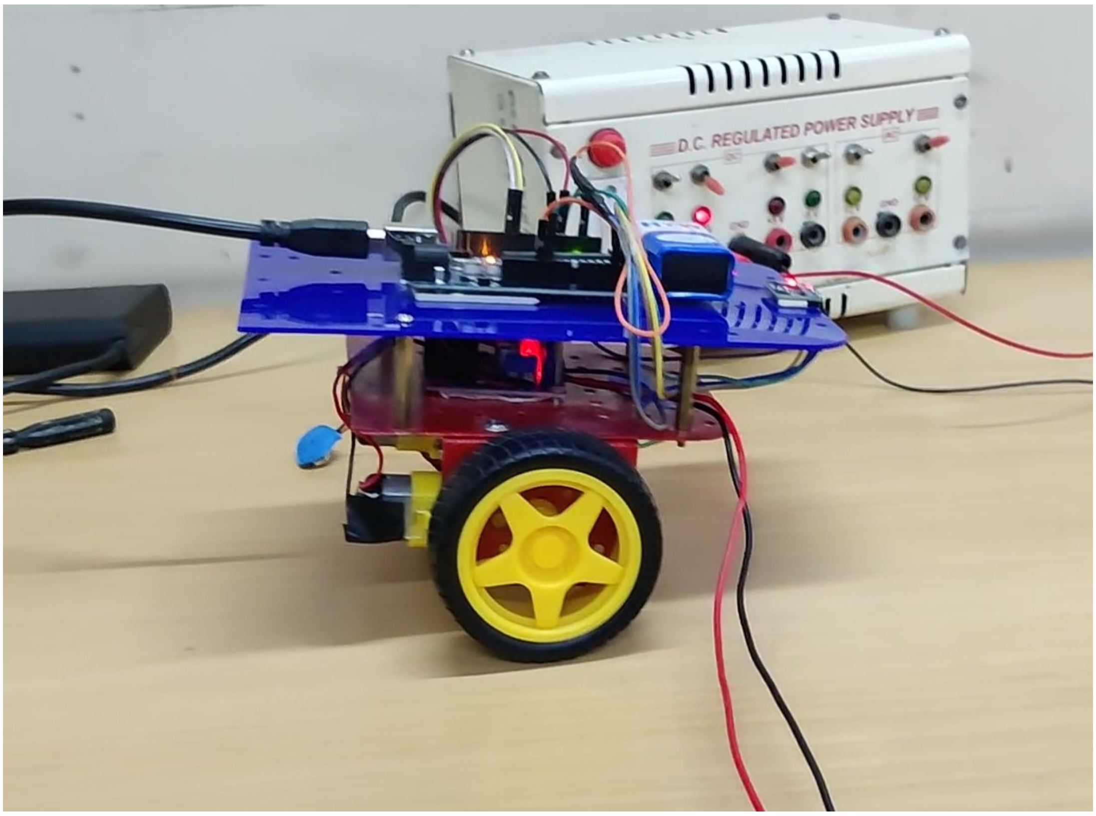


FIGURE 4.1: Balanced Model

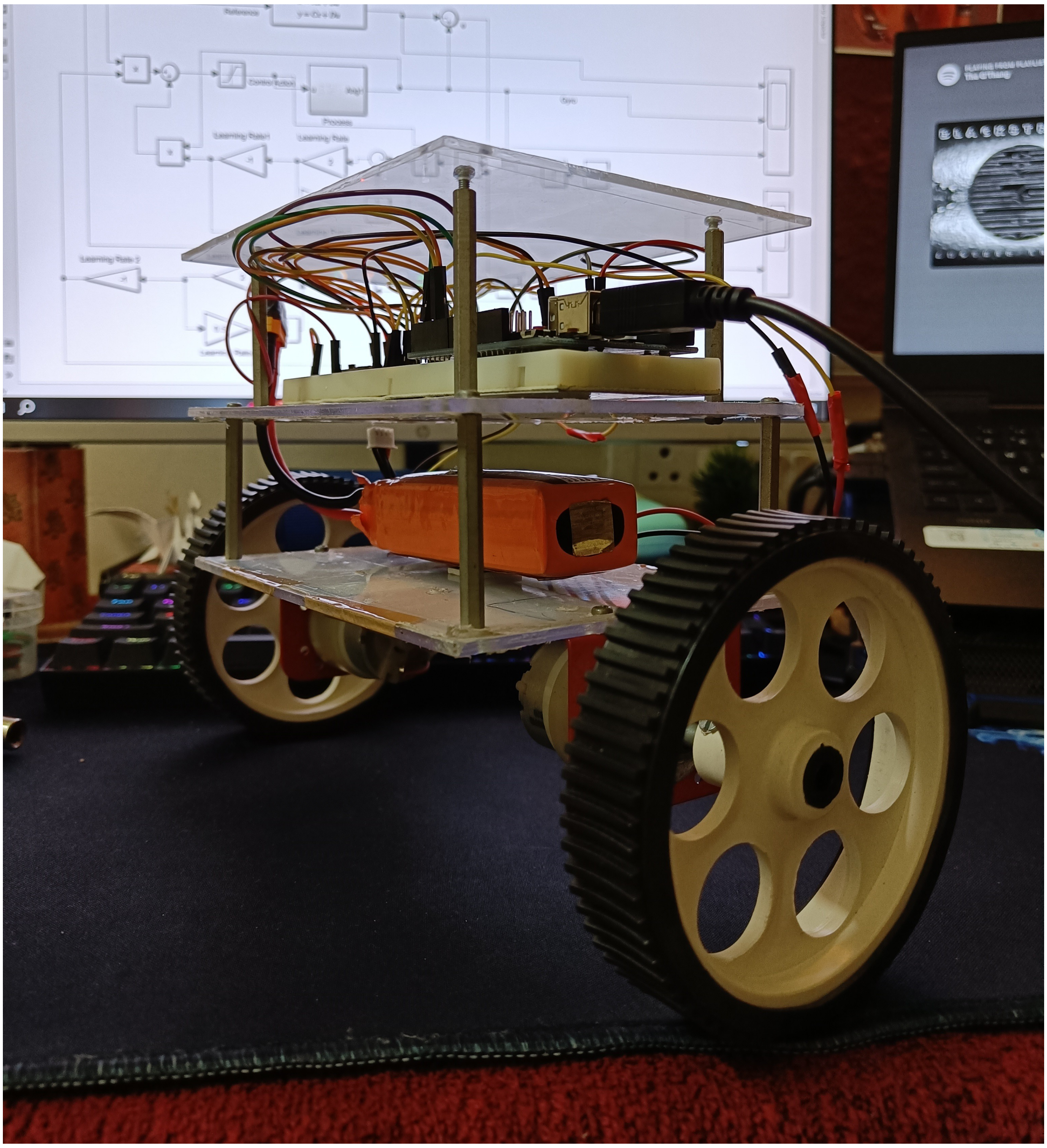

FIGURE 4.2: Balanced Model(new)

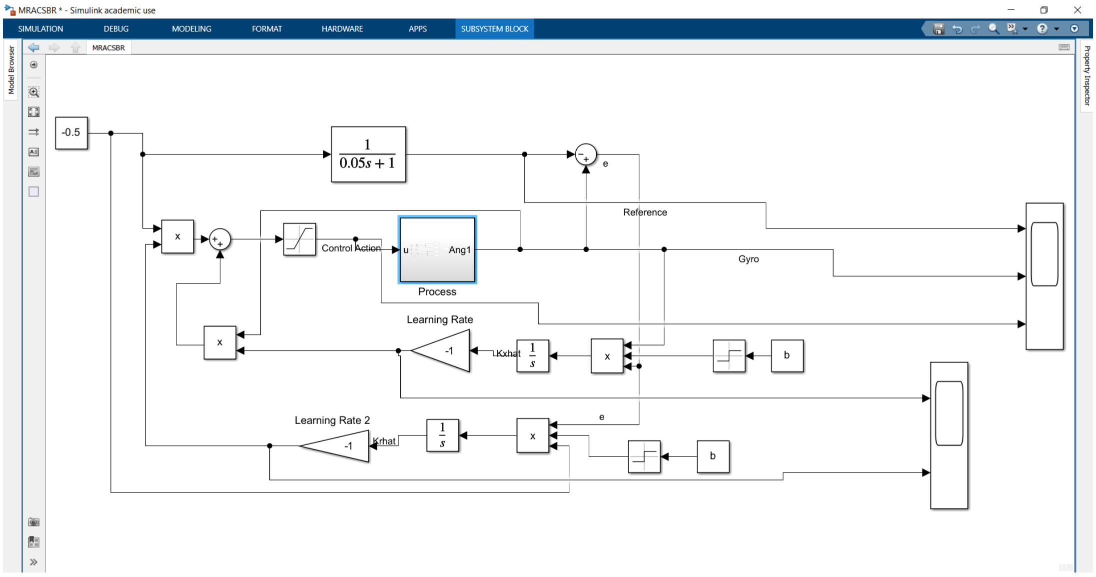


FIGURE 4.3: Simulink Model

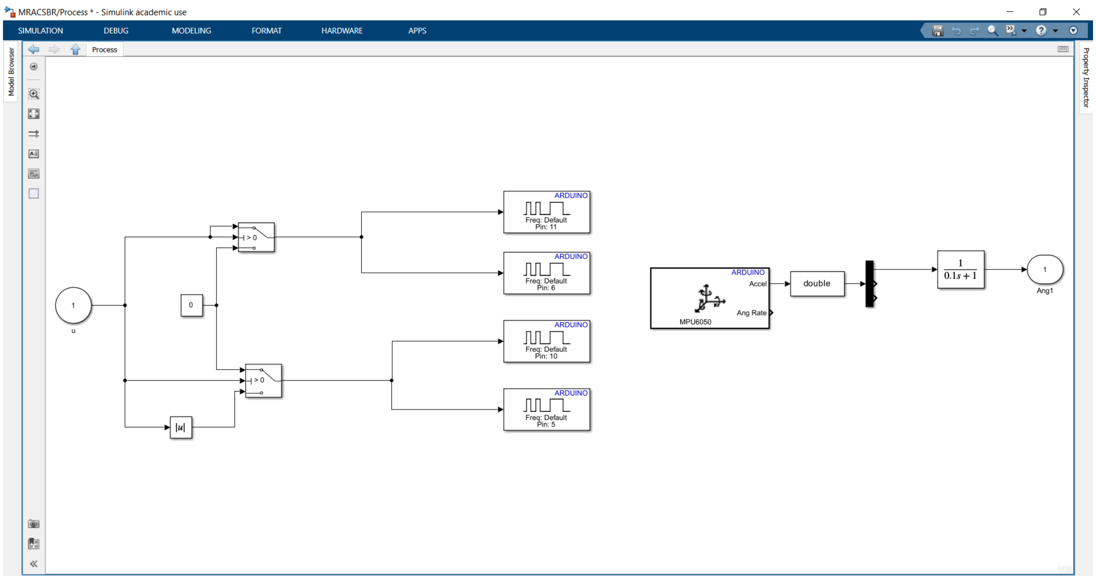


FIGURE 4.4: Simulink Subsystem

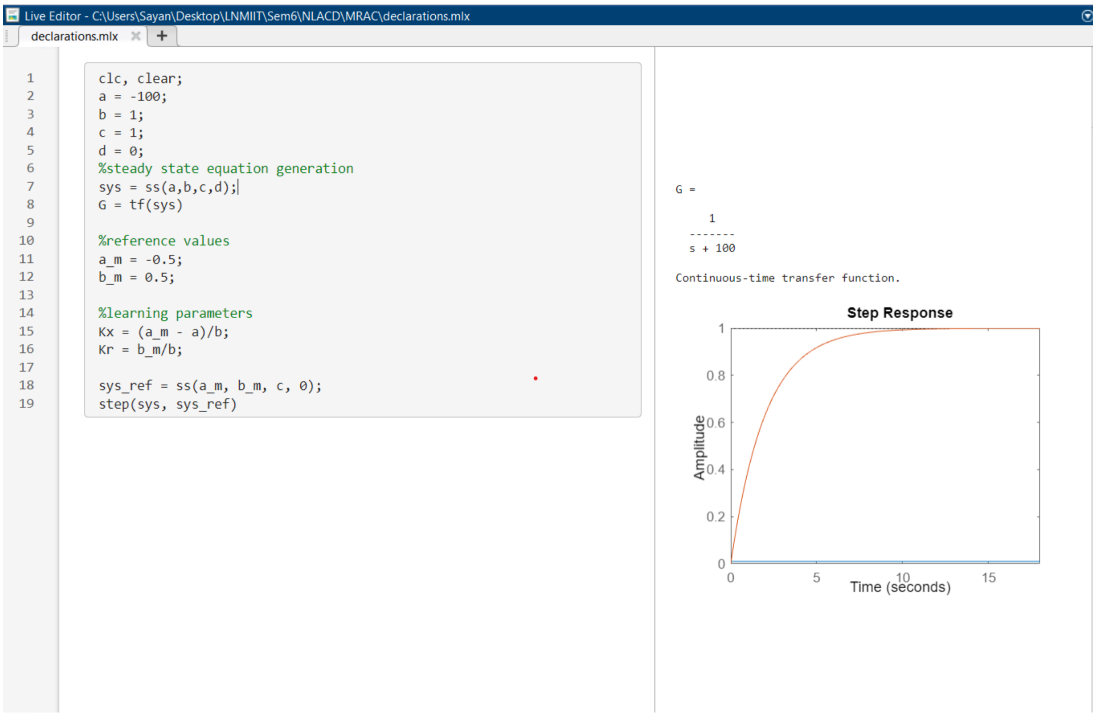


FIGURE 4.5: MATLAB LiveScript

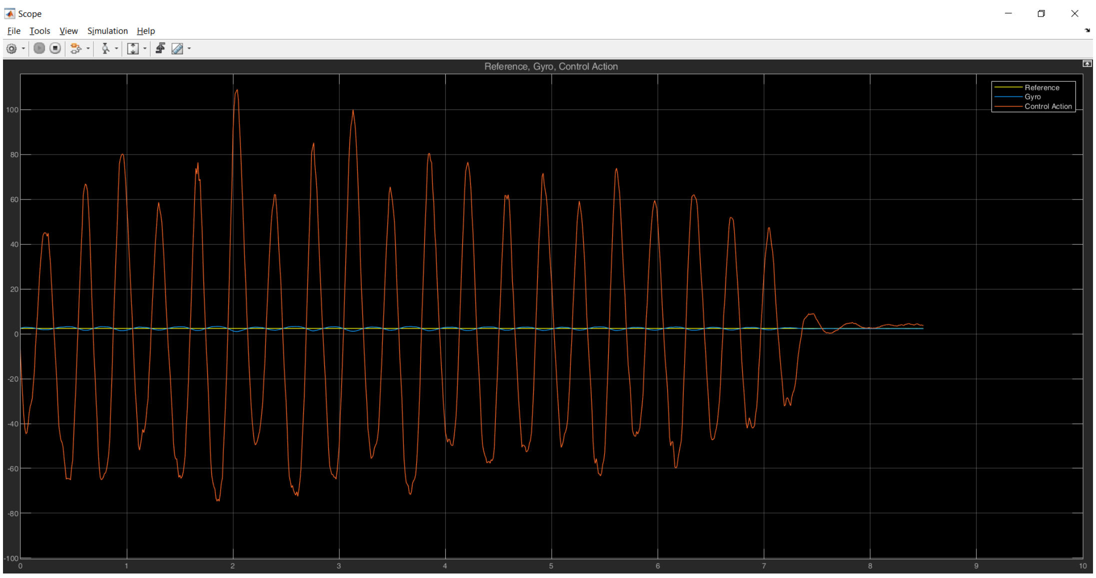


FIGURE 4.6: Control Action Generated

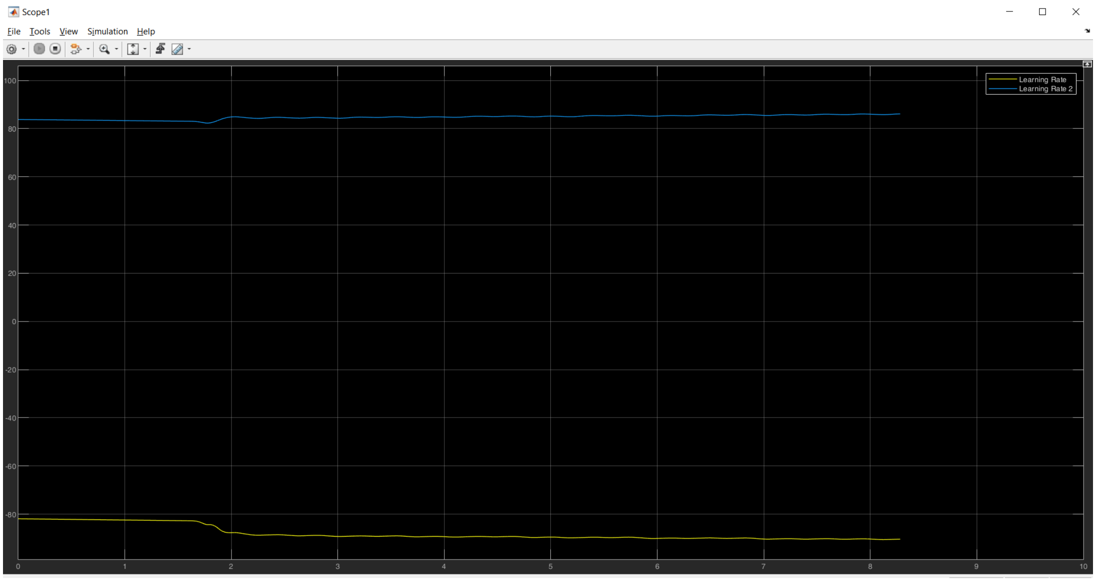


FIGURE 4.7: Learning Rates

# Chapter 5

# Conclusions and Future Work

## 5.1 Future Work

- Sigma Modification - To stabalize our robot we can use Sigma Modification in our MRAC controller which will provide more robustness to our robot as it keeps the error bounded and does not let it go to a value from where it will become difficult for the robot to recover. The sigma parameter also influences the rate at which the controller parameters are updated. A higher value of sigma value leads to faster updation of controller parameters, which is beneficial for quickly adapting to the changes.

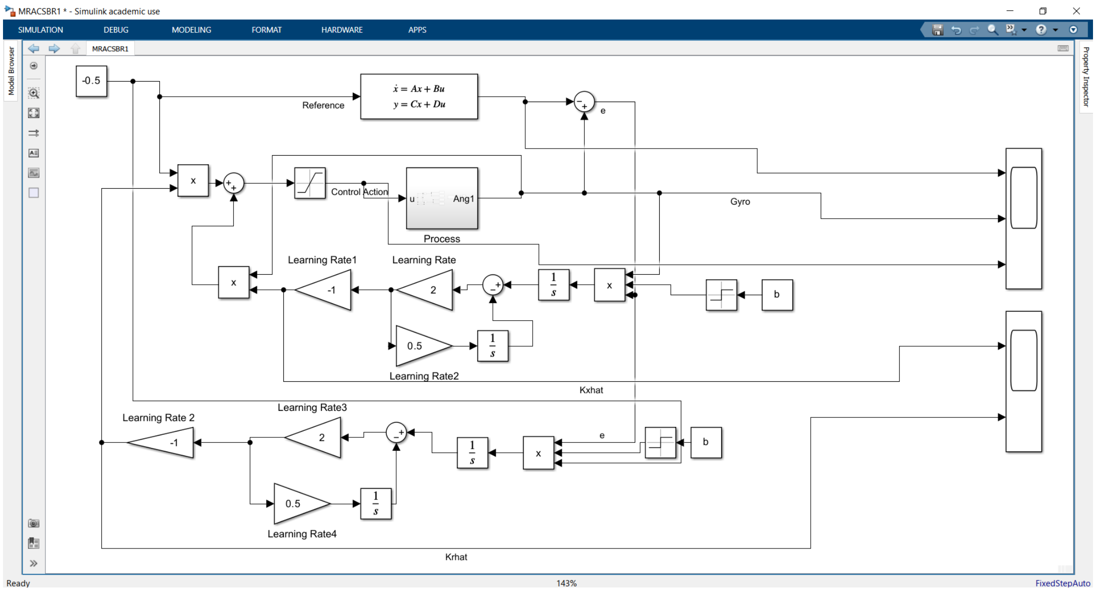

FIGURE 5.1: Simulink Model

- Introducing velocity state in the controller design can help in improving the quality of the controller as now we can also control the velocity with which the wheels will move because up till now the wheels move at a much faster speed, and by introducing velocity state we can control our wheels' speed because of which it will make our controller more robust.

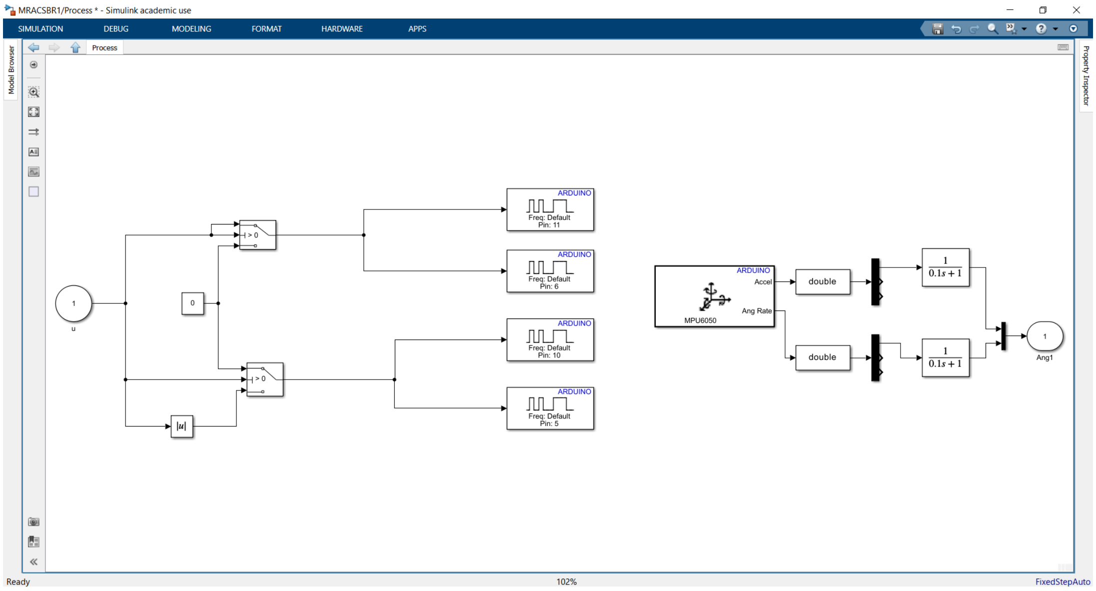


FIGURE 5.2: Simulink Subsystem

- Adding encoder motors in the chassis as it will allow our robot to move forward and backwards till a desired distance and this distance can be passed as a parameter to the encoder motor.

## 5.2 Conclusion

The goal of the project is to design and build a robot that can balance itself on two wheels using a suitable control action technique. While using the PID controller, the robot balanced by continuously monitoring the angle of the robot using gyroscope and making adjustments to the motor speed and direction of rotation. While using MRAC Design, the robot was able to learn and adapt according to an ideal reference model and was able to balance itself on its wheels even after deliberately throwing it off balance.

Through our project, we were able to identify when the robot went off balance and designed and tuned an algorithm to correct it in a stable and robust manner. This project has helped us gain practical knowledge in the field of controller design and has given us the opportunity to apply theoretical concepts to a real-world application.

# Bibliography


- Verma, B. (2023). Title of Paper. Retrieved November 19, 2023, from https://lnmiit-my.sharepoint.com/:w:/g/personal/bharat_verma_lnmiit_ac_in/EbewGkF5kjFBnm7JAluDLHkBydVxXL7g1Pag59GSEWJ-9Q?rtime=XVC7Fufo20g
- Åström, K. J., and Hägglund, T. (1995). PID controllers: theory, design and tuning [Book]. In International Society of Automation (Vol. 2). Instrument Society of America.
- Nguyen, N. T. (2018). Model-Reference Adaptive Control. In Advanced Textbooks in Control and Signal Processing (Issue 9783319563923, pp. 83–123). Springer International Publishing.
- Introduction to PID Controller Design - Control Tutorials for MATLAB and Simulink
- Model Reference Adaptive Control - Mathworks
- Robust Model Reference Adaptive Control - IIT Delhi 2018