LNMIIT
The LNM Institute of Information Technology

20th November , 2023

Model Reference Adaptive Control Design for Self Balancing Robot

Supervisor

Dr. Bharat Verma

Students

Divyansh Jain - 20uec049
Satvik Sharma - 20uec116
Sayan Chatterjee - 20uec120

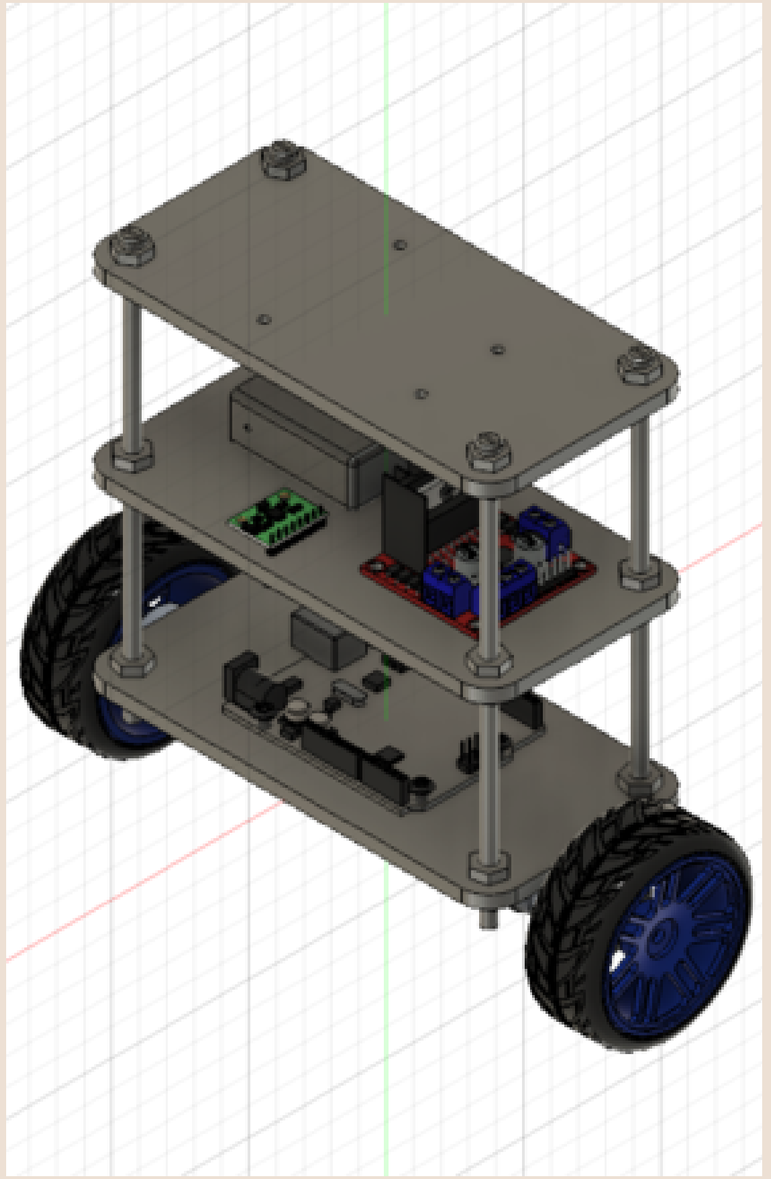
BTP Presentation

Table of Contents

I	Overview	3
II	Mechanical Structure of the Robot	6
III	Designing the System (PID)	8
IV	Model Reference Adaptive Control	13
V	Conclusions & Future Work	21

A **self-balancing robot** is a type of robot that :

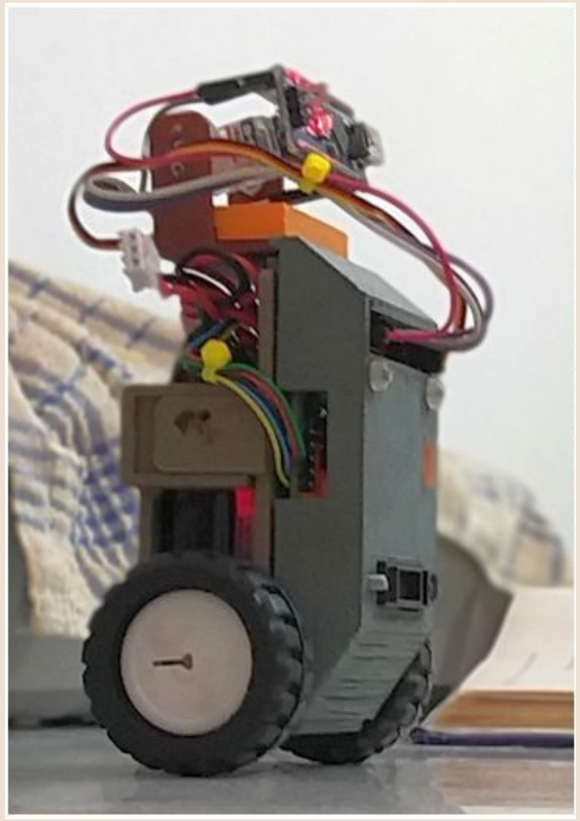
- **Maintains its balance** on its own without any external support.
- Uses the **movement of its wheels** to stay upright.
- Has a unique **stability control system** that keeps it in upright position even after external disturbances.
- Uses the **principle of a inverted pendulum** to develop and implement a suitable stability control system.



TWO WHEELED SYSTEM

Advantages

- Maneuverability
- Energy Efficient
- Compact size
- Agility and Speed
- Climbing difficult terrains



TWO WHEELED SYSTEM

Real Life Usages

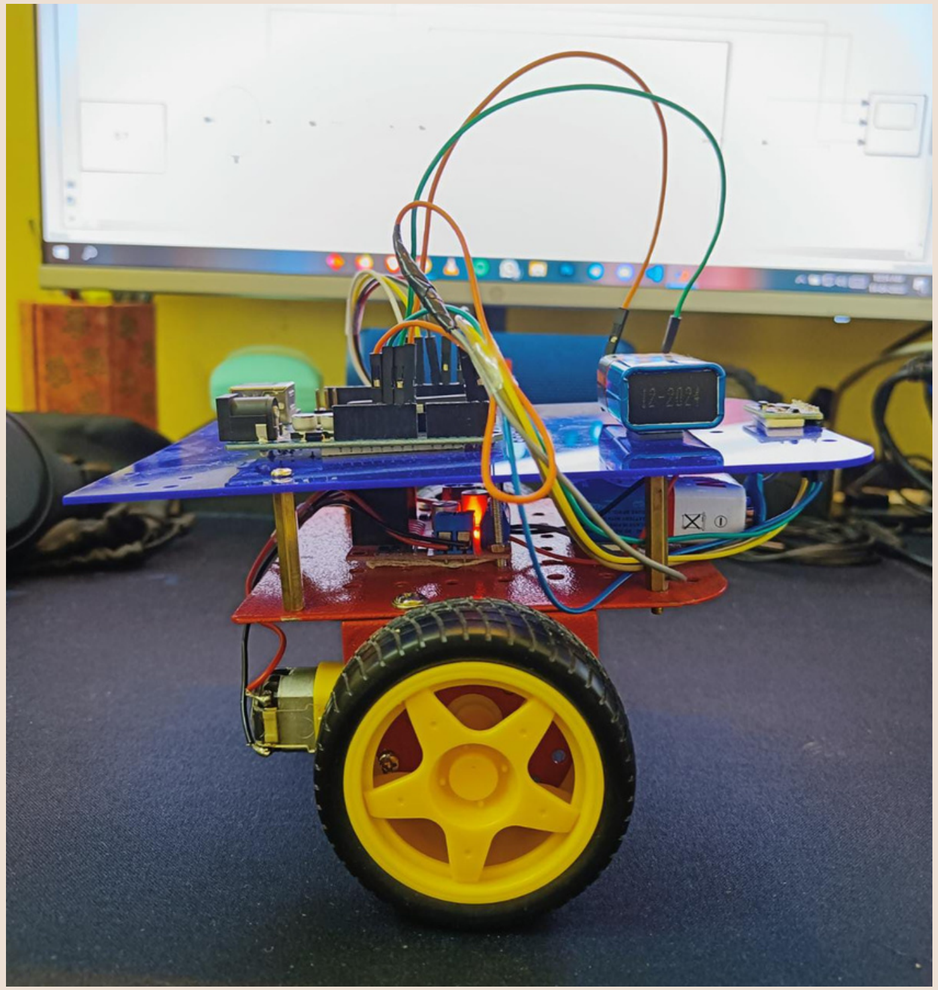
Warehouses for handling

Personal Transporters

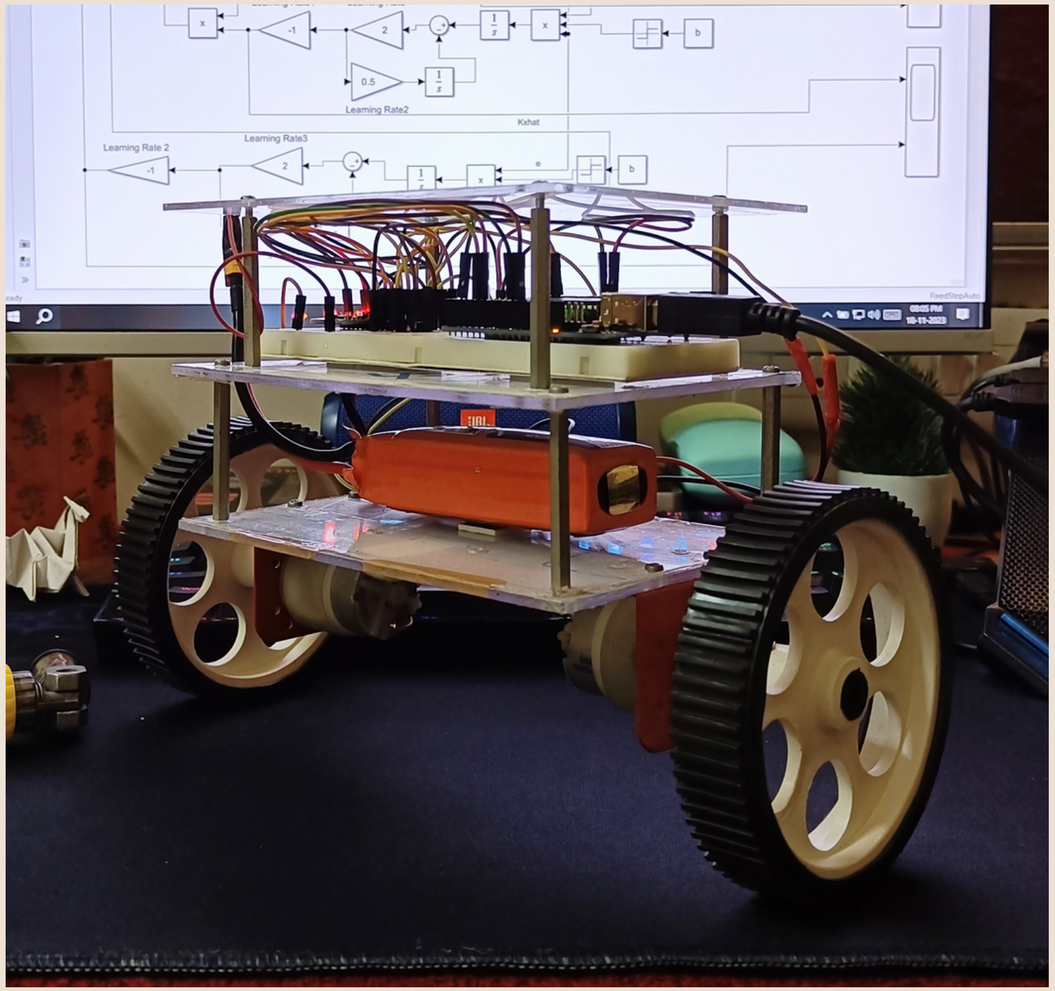
Autonomous Surveillance

Concepts of balancing

Rovers

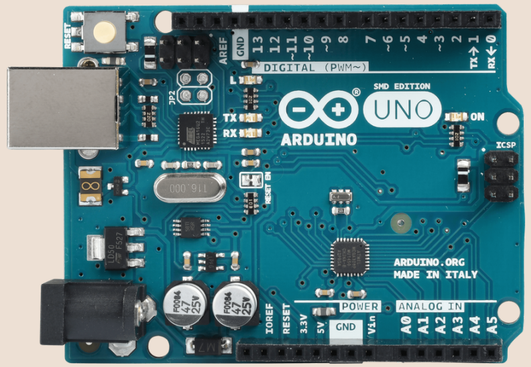


ORIGINAL MODEL

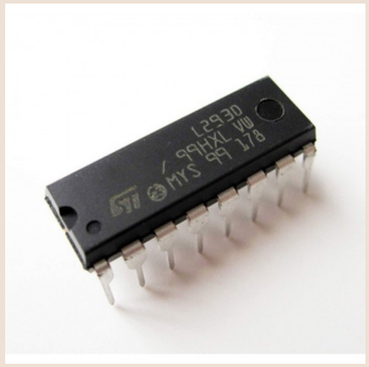


REDESIGNED MODEL

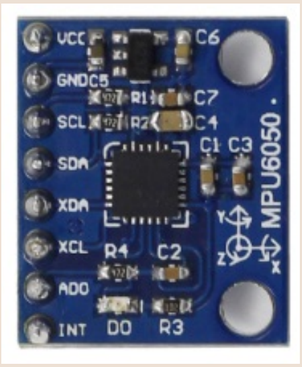
The Components used ....



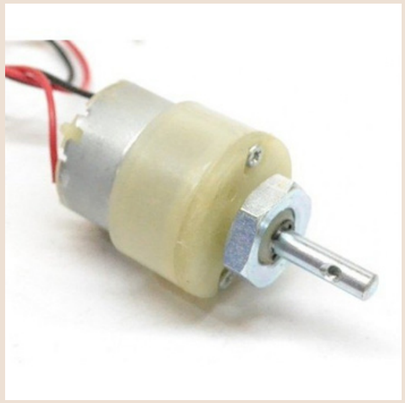
Arduino Uno Microcontroller



L293D Motor Driver IC



MPU6050 Gyroscope



BO Motors

Wheels

LiPo Batteries

Fibre Chassis

Earlier Work : PID System Designing

Previously, we designed the system using a PID Controller. The model is assumed to be a first order unstable system with its transfer function assumed to be :

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

The IMC-PID algorithm uses the dynamics of the process being controlled to improve the performance of the PID controller. This can result in faster response times and better disturbance rejection compared to a standard PID controller.

The values of the variables can be given as :

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

$$\tau_I = \alpha + \frac{\theta}{2}$$

$$\tau_D = \frac{\alpha\,\theta}{2\,\alpha + \theta}$$

$$\alpha = \frac{2\,\lambda\,\tau + \lambda\,\theta + 2\,\tau\,\theta}{2\,\tau - \theta}$$

Hence, the final PID values, viz, Kp, Ki and Kd are given by :

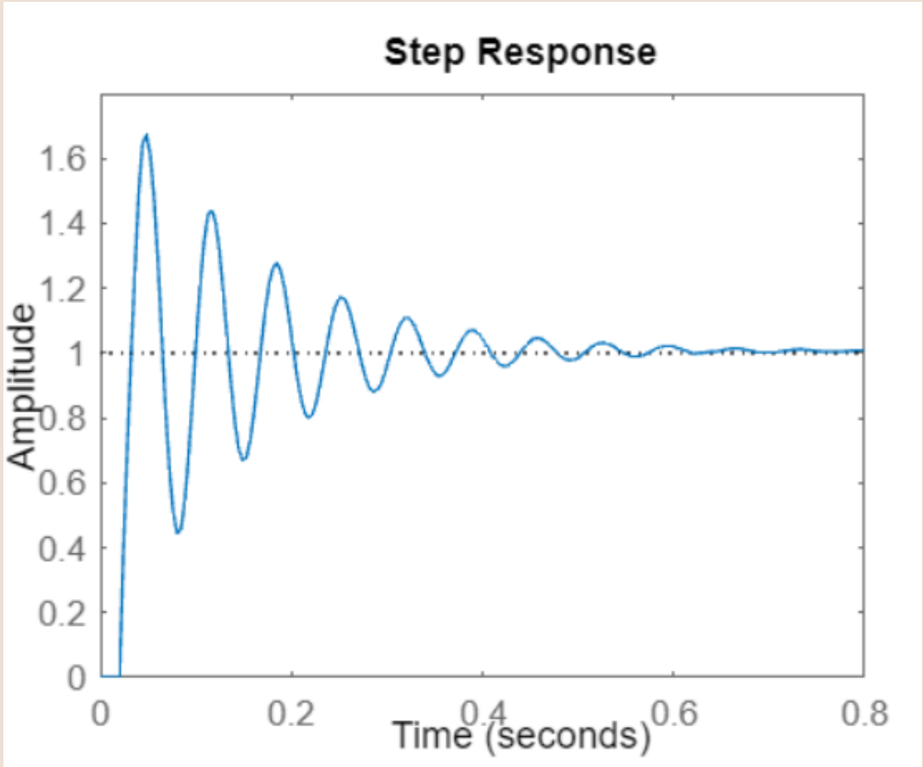
1. $K_p = k_c$

2. $K_i = \frac{k_c}{\tau_i}$

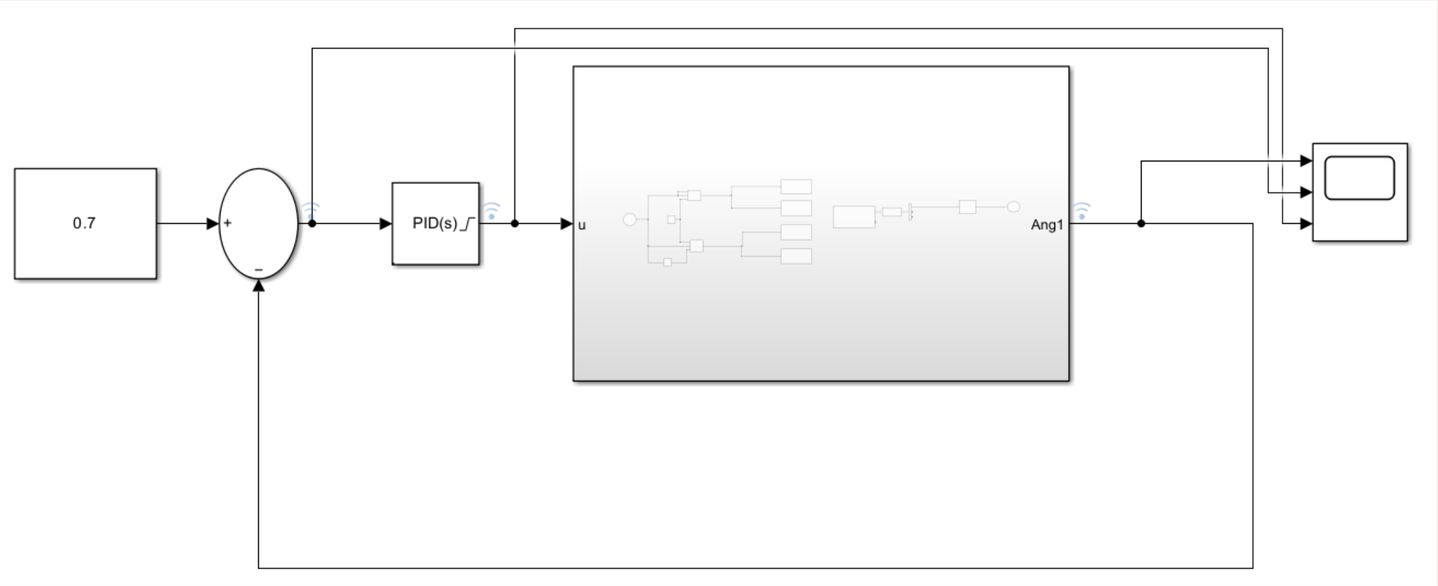
3. $K_d = k_c * \tau_d$

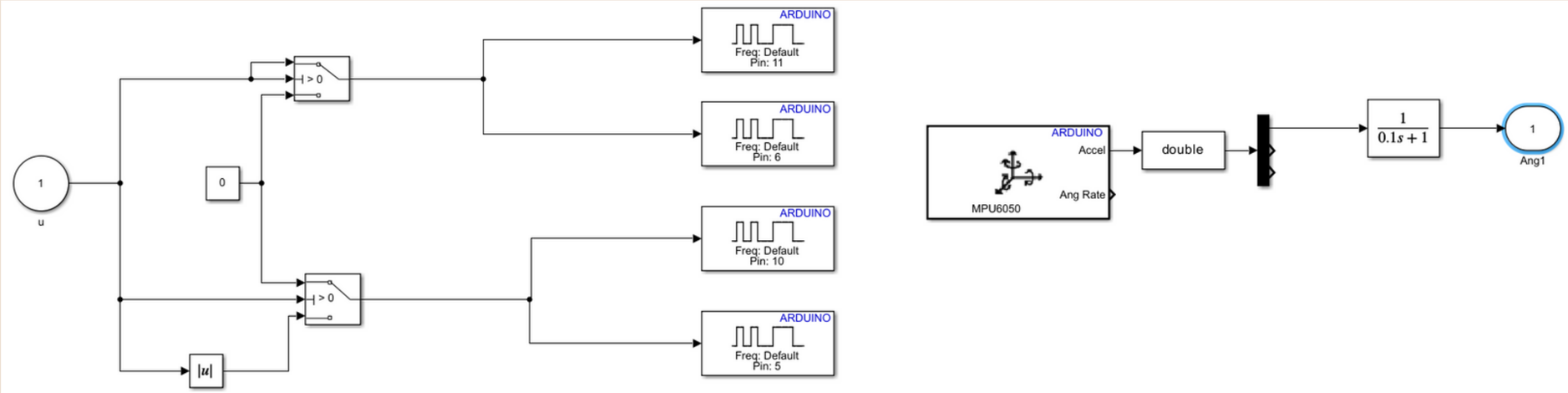
A MatLab LiveScript is used to compute these values and is then used in the Simulink Model.

```
k = 800
tau = 1000
theta = 0.0200
lambda = 0.5000

alfa = 0.5200

kp = 64.9516
ki = 34.4250
kd = 0.6373
G =

                  0.8
  exp(-0.02*s) * ---
                  s

Continuous-time transfer function.
```

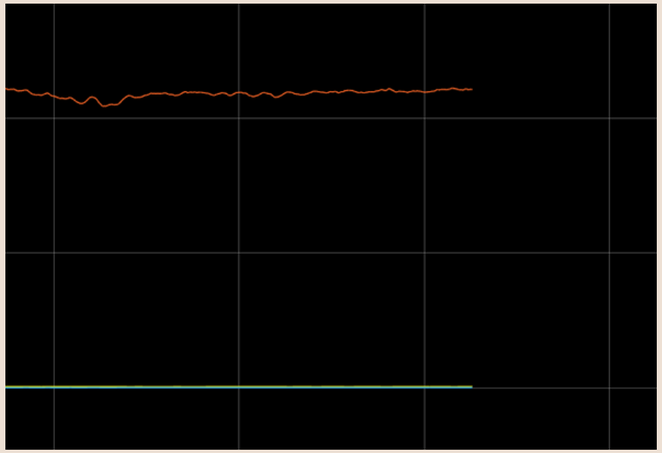


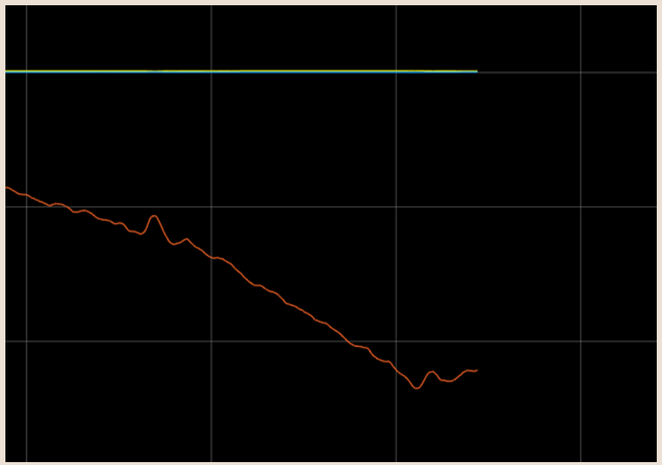
The Simulink Model

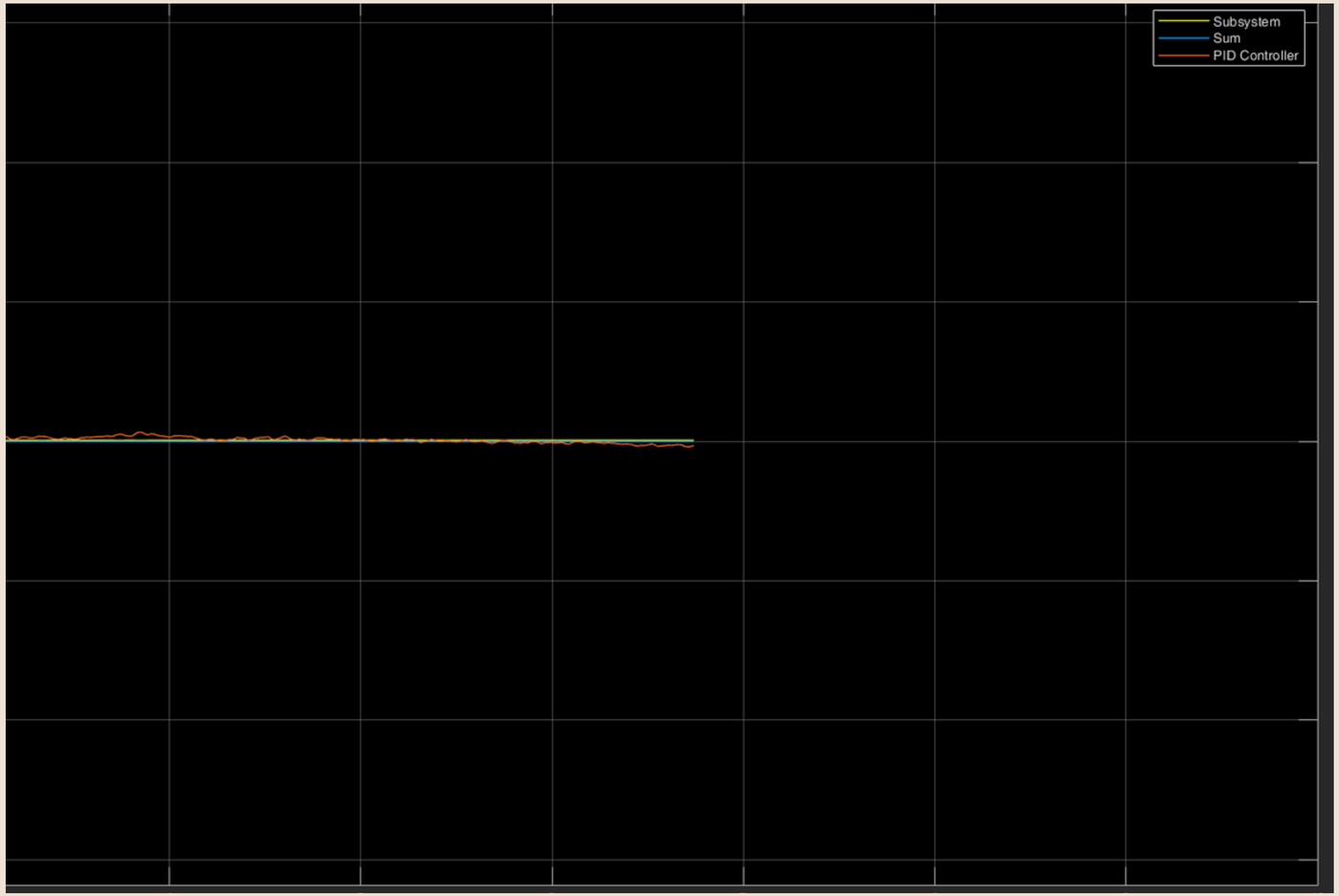




Results







Model Reference Adaptive Control

What is MRAC?

A control algorithm 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.

Advantages of MRAC over PID

- provide robust performance even in the presence of uncertainties and variations in the plant dynamic.
- useful tool for controlling complex and dynamic systems.

MRAC Implementation

It involves the following steps :

- Defining the reference model
- Estimating the plant parameters
- Designing the adaptive controller
- Tuning the controller gains

System Designing

- **The Process Model :** The subsystem model for motor control and gyroscope reading were created using the appropriate blocks.

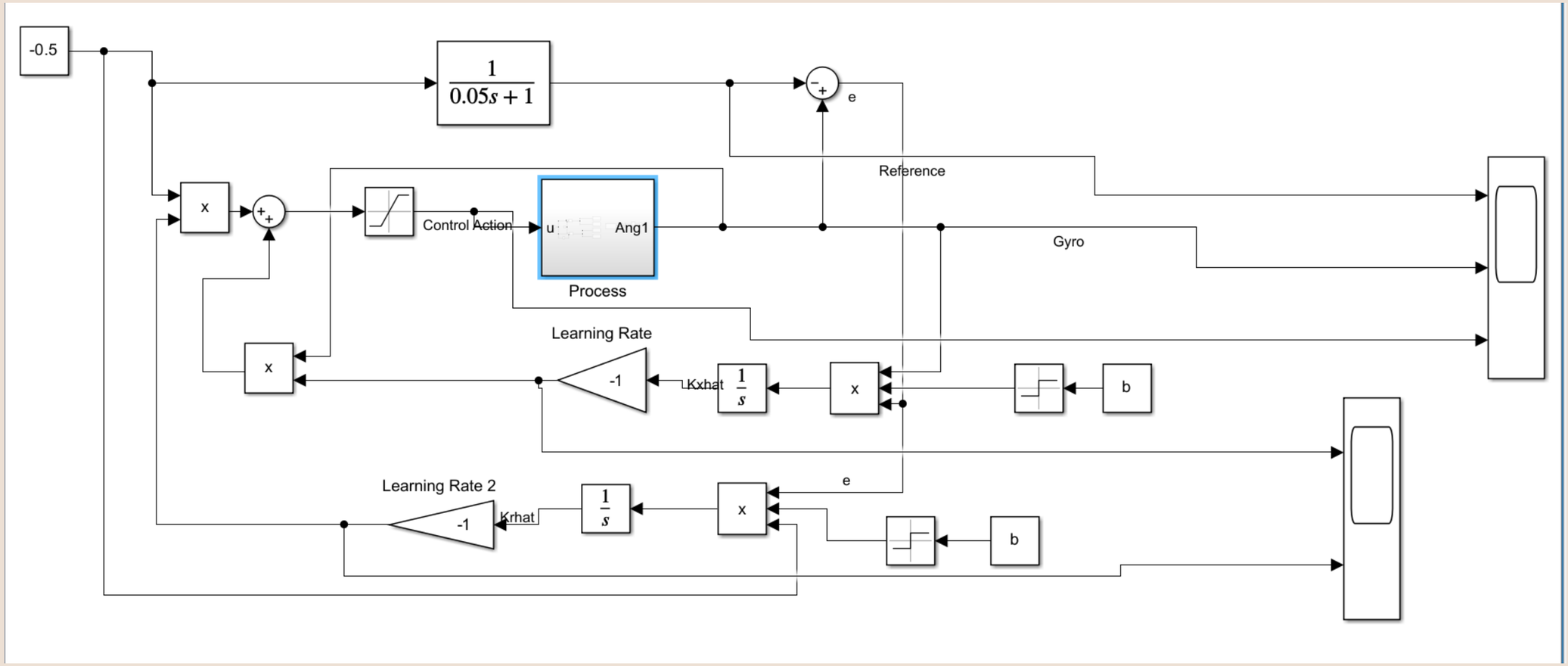
- **The Reference Model :**

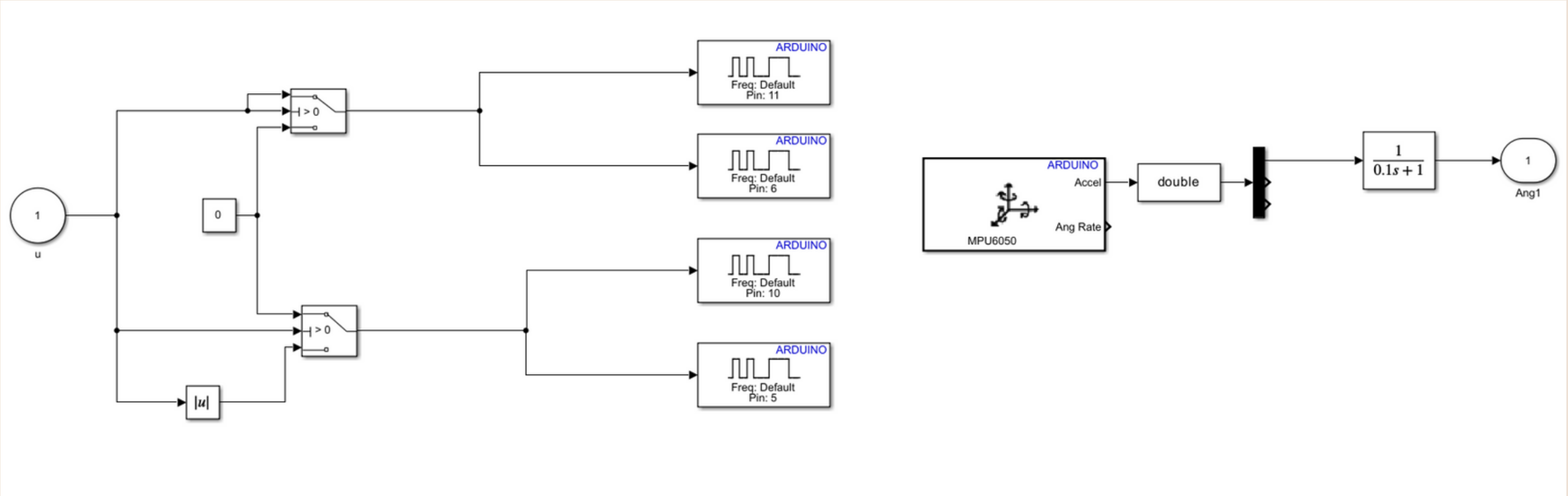
$$G(s) = \frac{1}{0.05s + 1}$$

- **Designing the Controller :** Using the Direct MRAC algorithm

- **Tuning:** Adjusting the values of learning rates accordingly.

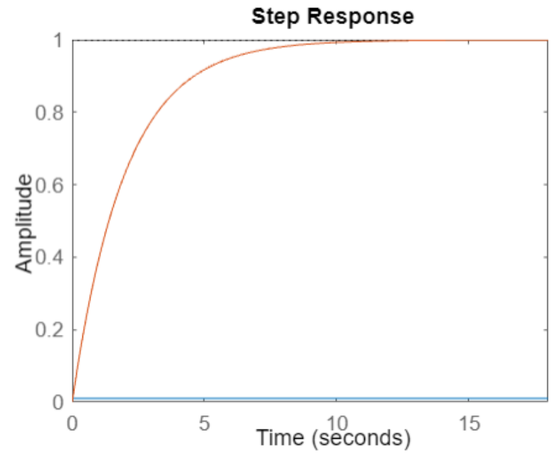
Simulink Implementation



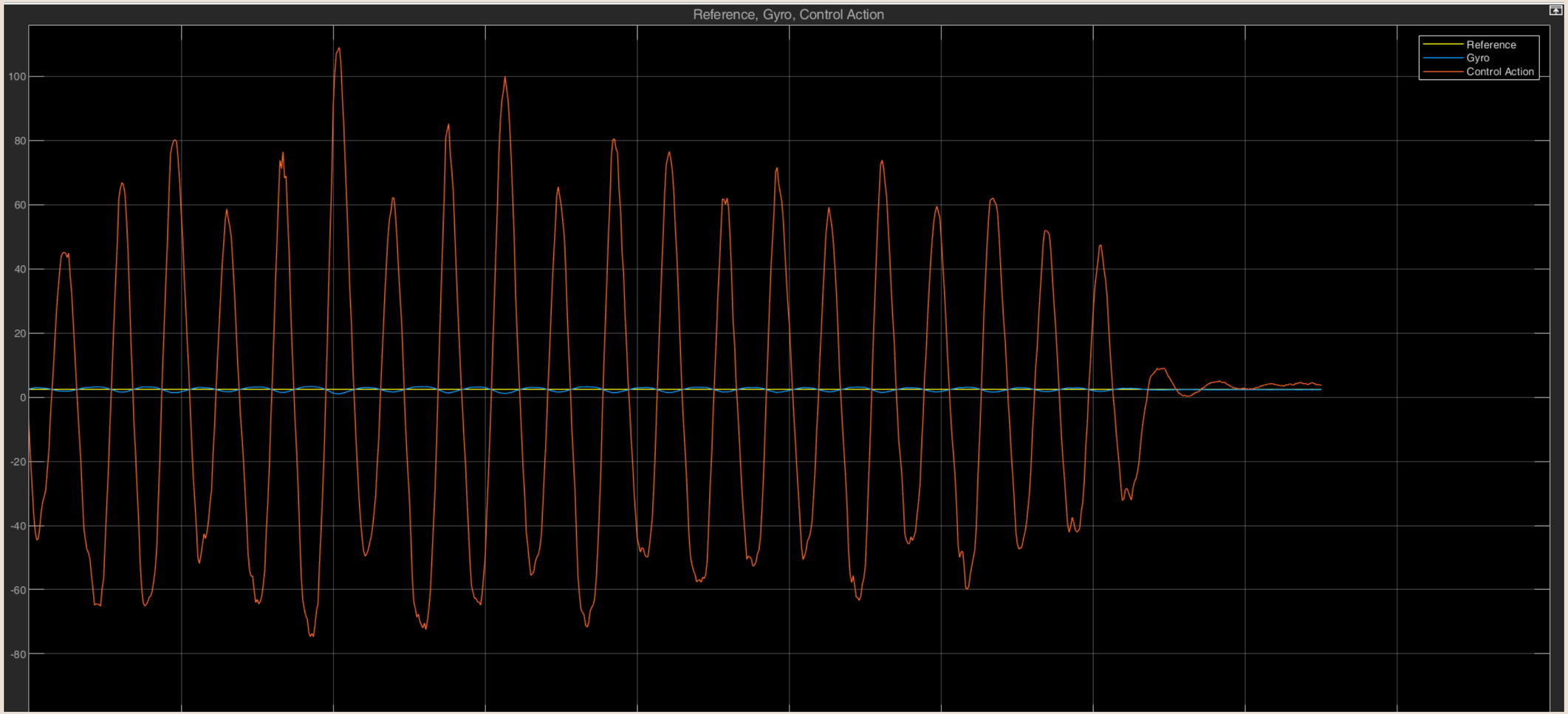



```
clc, clear;
a = -100;
b = 1;
c = 1;
d = 0;
%steady state equation generation
sys = ss(a,b,c,d);
G = tf(sys)

%reference values
a_m = -0.5;
b_m = 0.5;

%learning parameters
Kx = (a_m - a)/b;
Kr = b_m/b;

sys_ref = ss(a_m, b_m, c, 0);
step(sys, sys_ref)
```

```
G =

     1
  -------
  s + 100

Continuous-time transfer function.
```

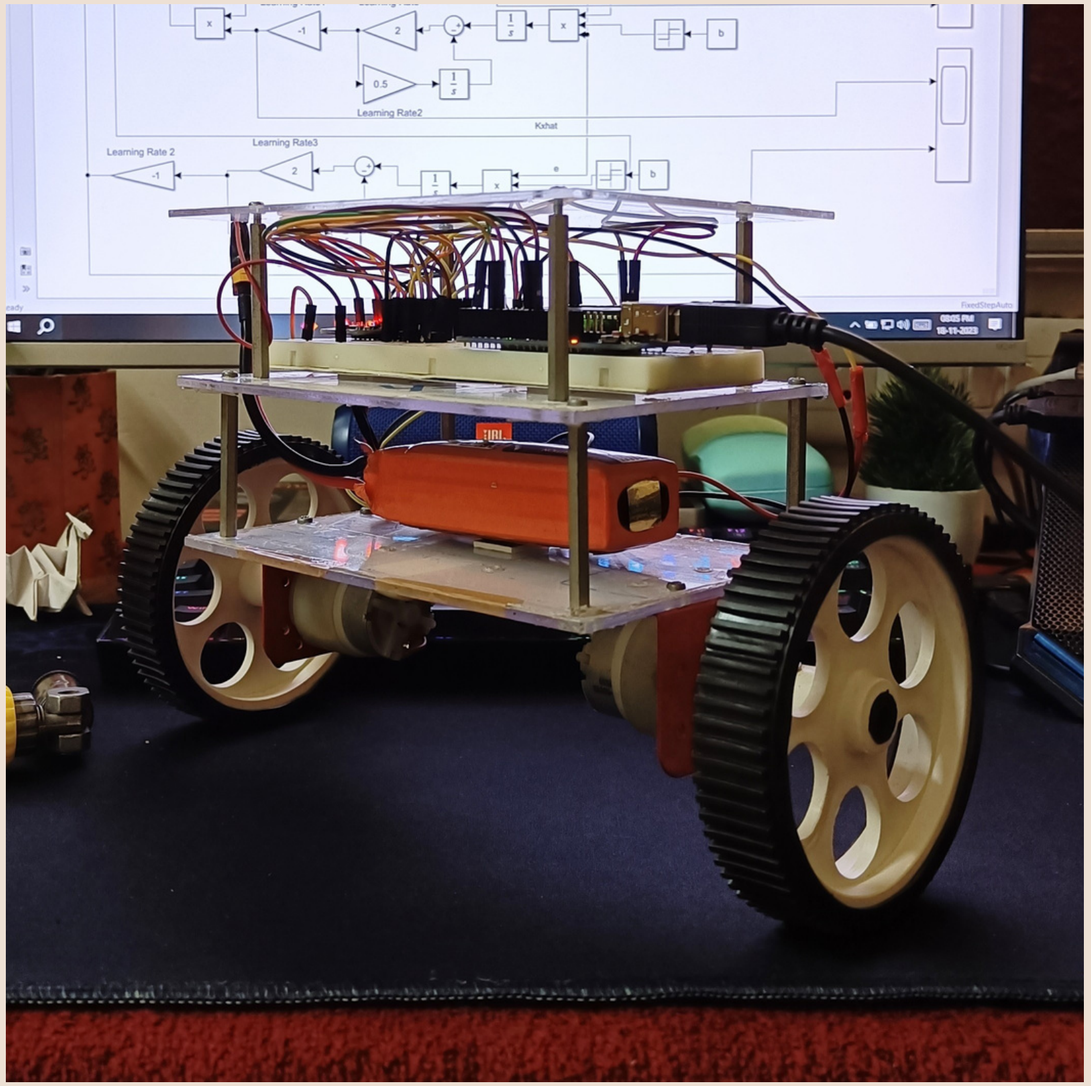


Control Action Generated



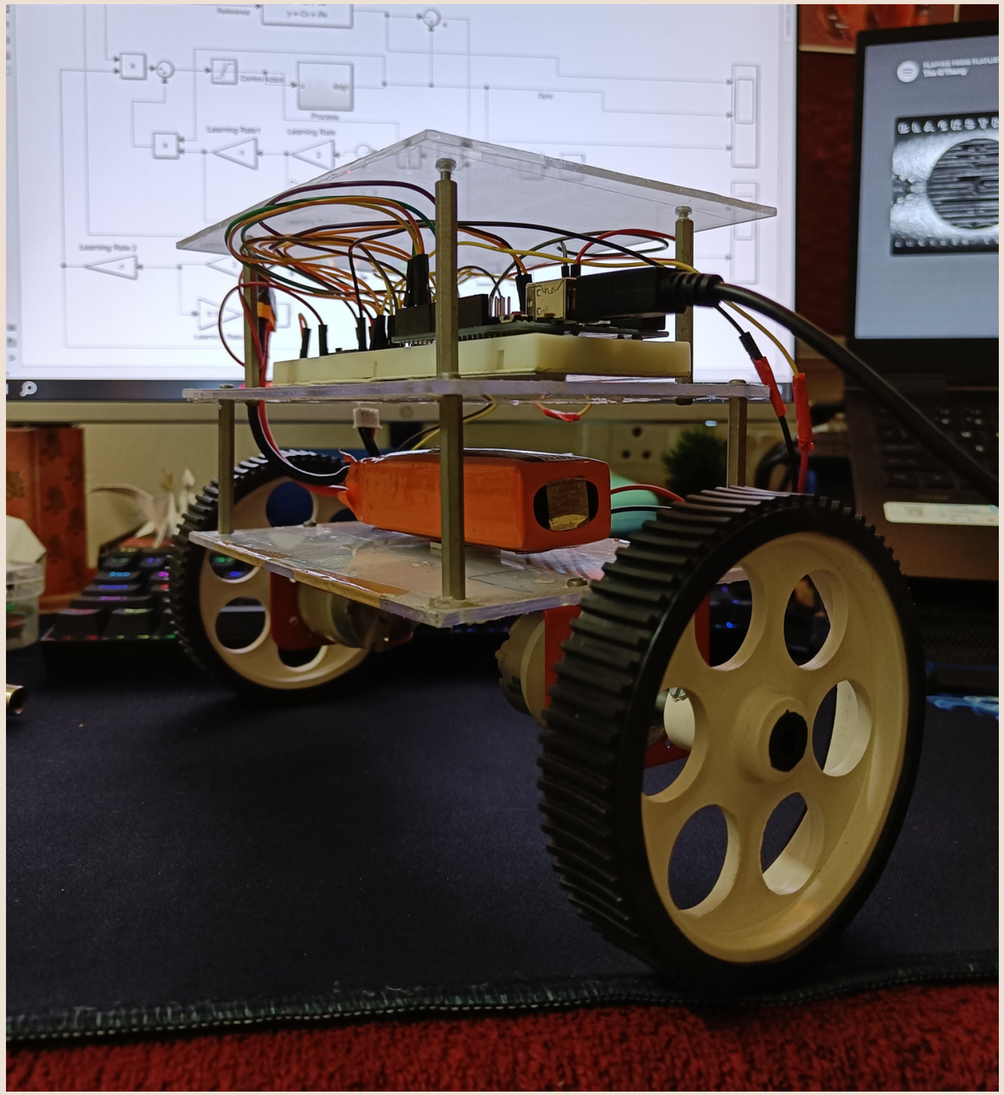
Design Updation:

- Physical Restrictions
- Weight Balancing



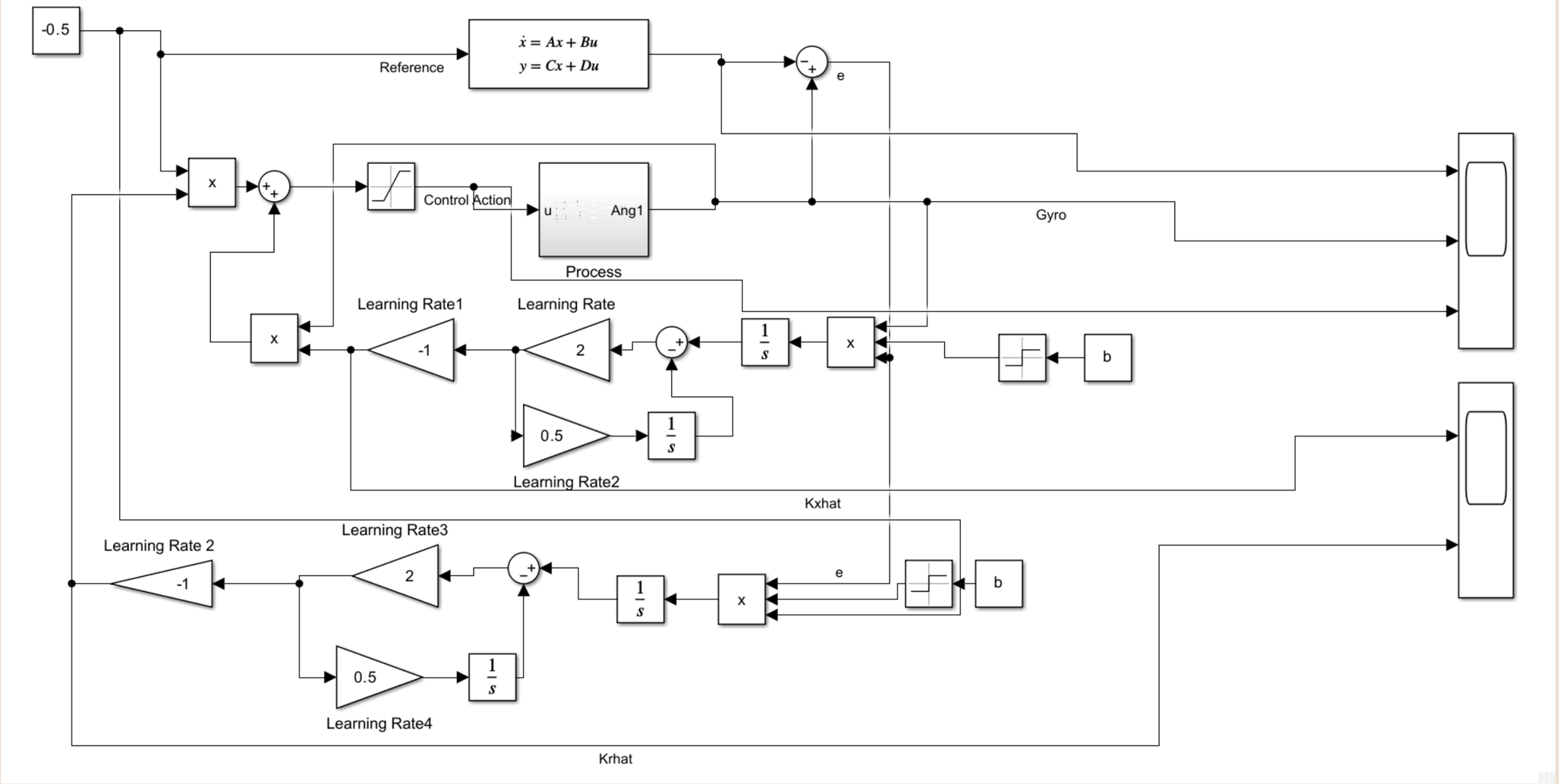
Issues faced:

- The control algorithms were not able to maintain the balance when the error went above a certain value.
- There was a continuous jitter encountered in the control action.



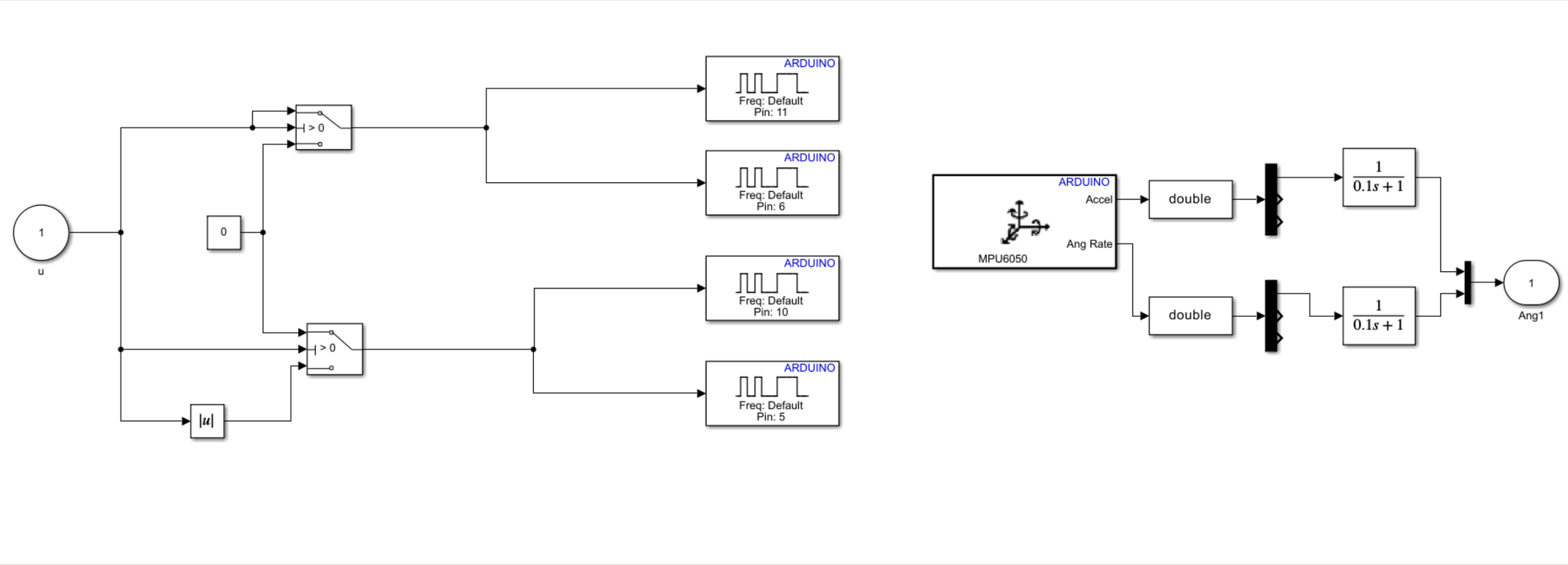
Proposed Work

- Sigma Modification -



Proposed Work

- Introducing velocity state in the controller design can help in improving the quality of the controller



Proposed Work

- 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.

Conclusion

- We designed a model and used PID controller to balance it.
- MRAC controller was then implemented to improve the control action generated.
- The Model was redesigned to eliminate mechanical factors causing errors.
- The MRAC control wasn't much accurate in the newer model so usage of Sigma modification was suggested to improve upon it.

LNMIIT
The LNM Institute of Information Technology

Thank you!

Supervisor

Dr. Bharat Verma

Students

Divyansh Jain - 20uec049
Satvik Sharma - 20uec116
Sayan Chatterjee - 20uec120