Model Reference Adaptive Control Design for Self Balancing Robot

Supervisor

Dr. Bharat Verma

Students

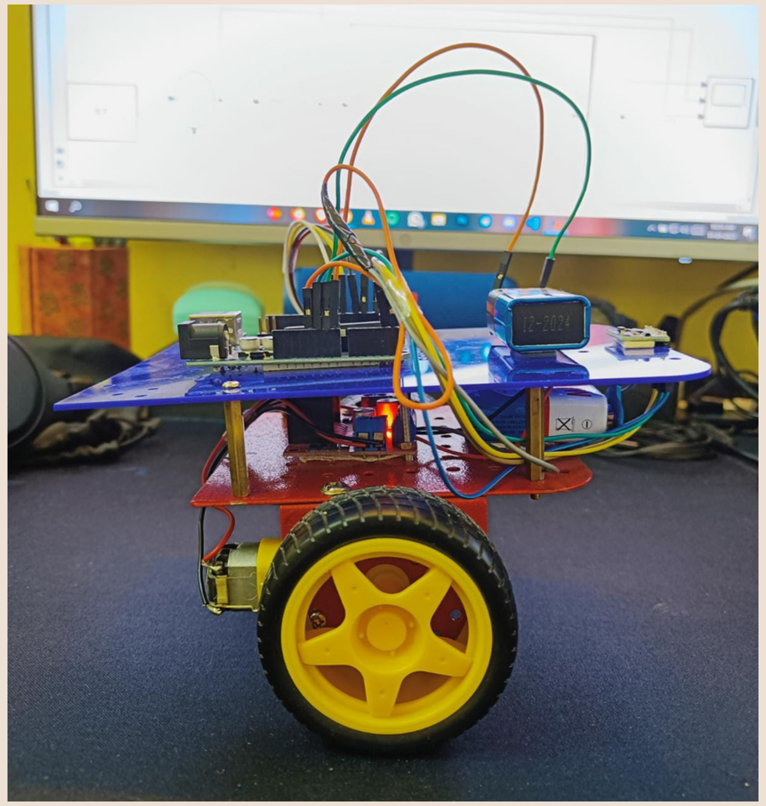
Divyansh Jain - 20uec049
Satvik Sharma - 20uec116
Sayan Chatterjee - 20uec120

15th April , 2023

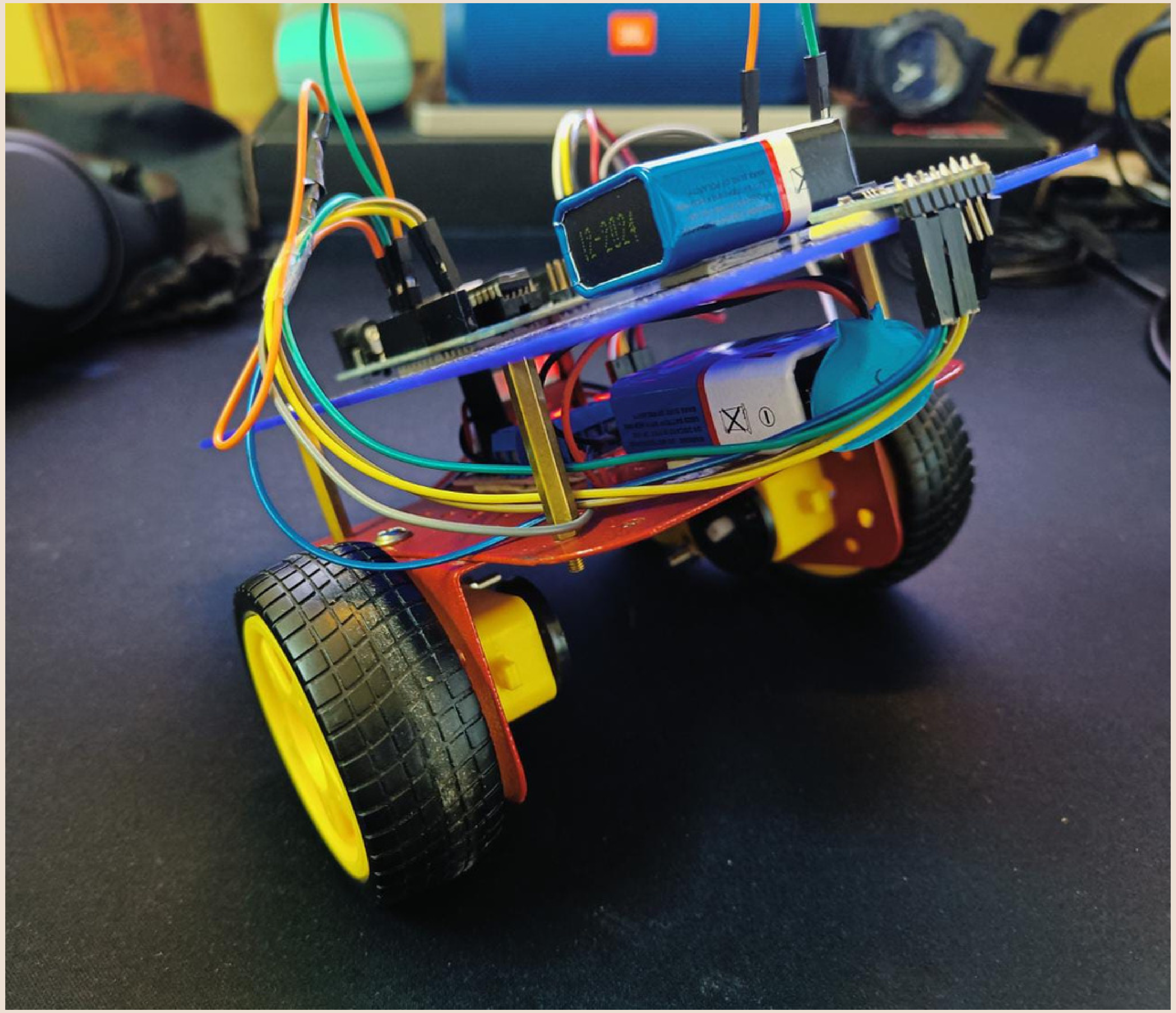
BTP Presentation

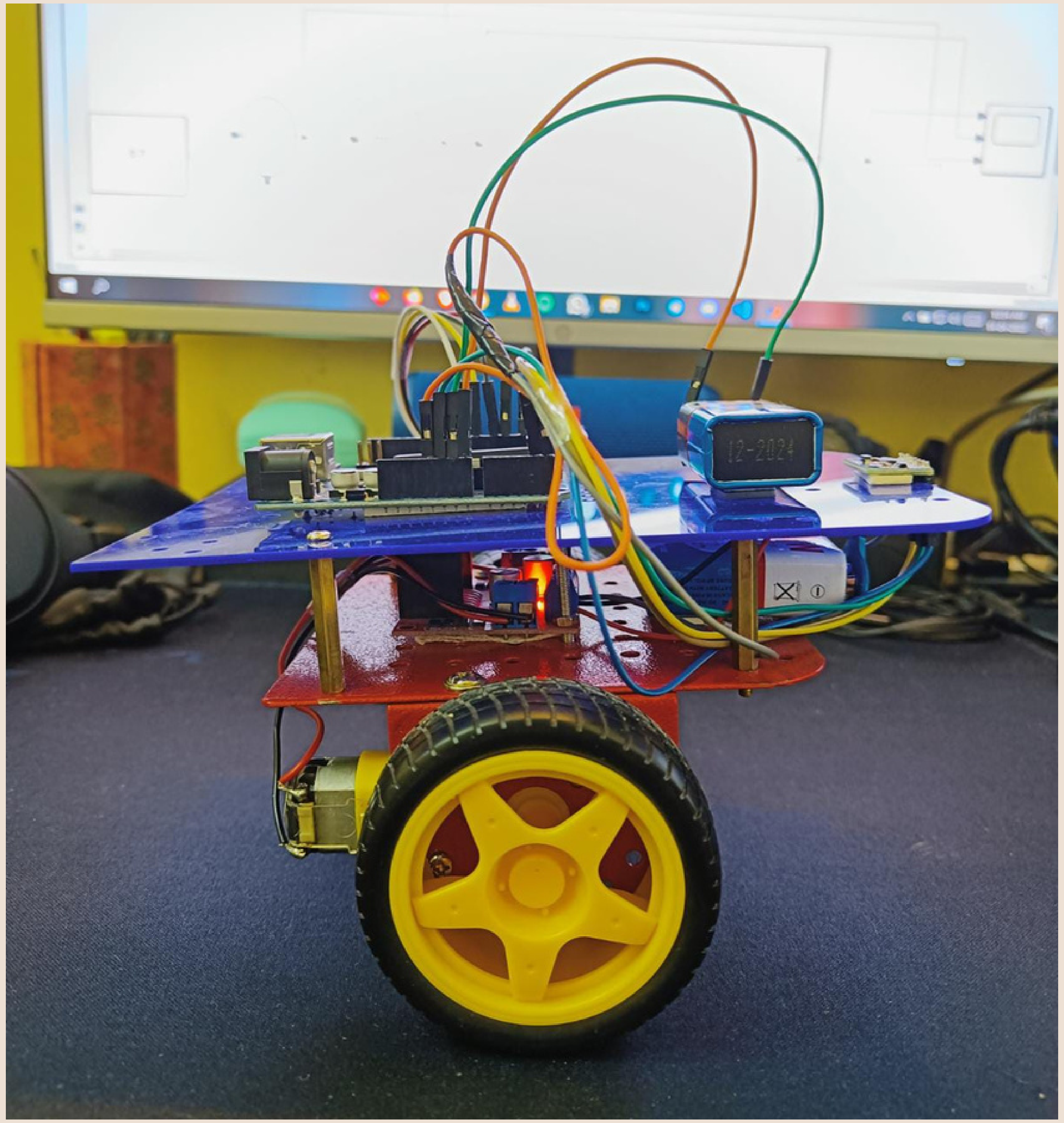
Table of Contents

- **A self-balancing robot is a type of robot that can maintain its balance on two wheels without falling over. The self-balancing feature is achieved through the use of a PID (Proportional-Integral-Derivative) controller, which is a control loop feedback mechanism widely used in control systems**

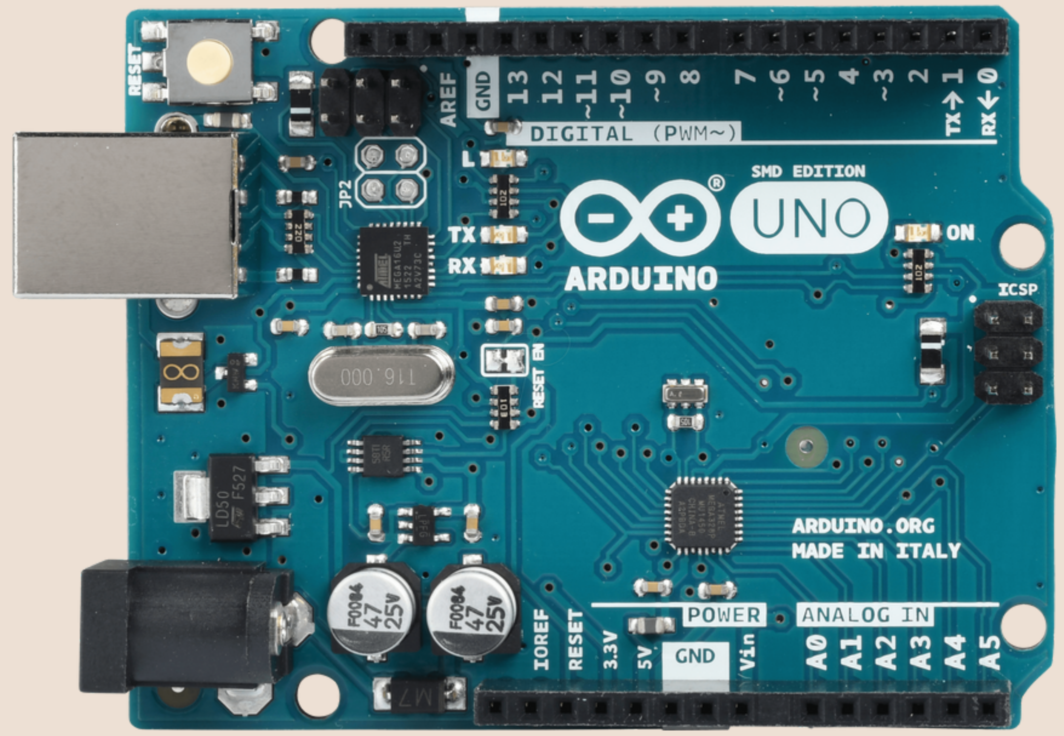


The PID controller constantly measures the robot's angle and compares it to the desired angle. If the robot's angle is not equal to the desired angle, the controller calculates an error value and adjusts the motor to correct the error. The proportional component of the PID controller adjusts the motor proportionally to the error value, the integral component accumulates the error over time, and the derivative component predicts the future error value based on the current rate of change.

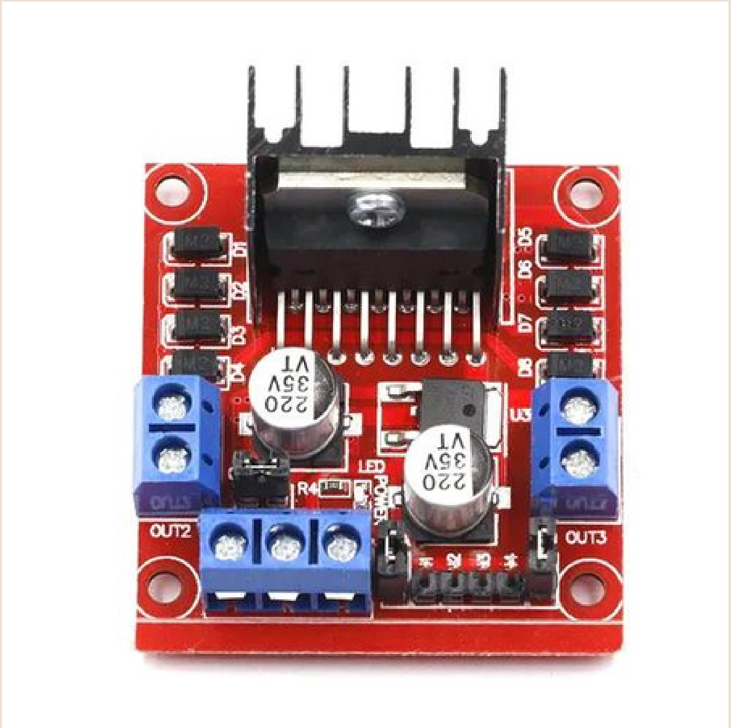




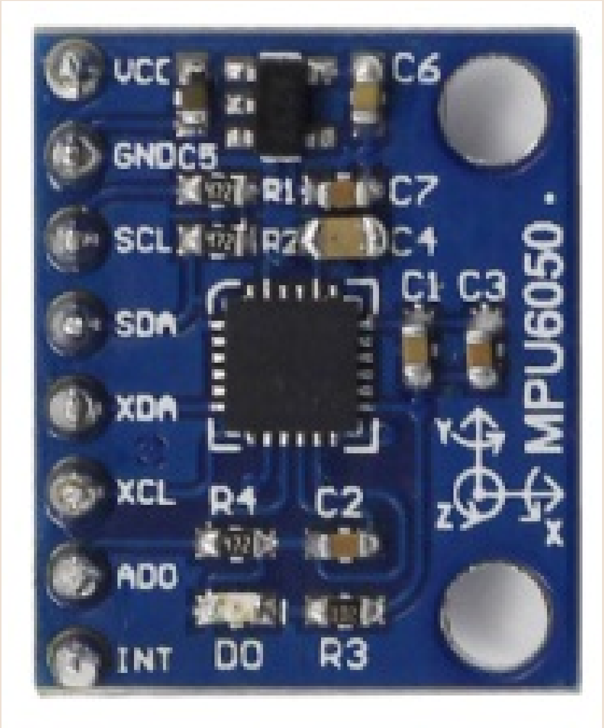
The Components used ….



Arduino Uno Microcontroller



L298N Motor Driver Module



MPU6050 Gyroscope

BO Motors

Wheels

2 Wheel Chassis

9V Batteries

The Transfer Function

The model is assumed to be a first order unstable system with its transfer function assumed to be :

$$G(s) = \frac{100e^{-0.01s}}{100s - 1}$$

where K= 100 is the Gain, τ = 100 is the Time Constant and θ = 0.01.

The Internal Model Control (IMC) Filter

We are using PID controller for our model and we have used the IMC-PID algorithm to design the PID. 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.

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

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

where α is used to cancel the right half of s-plane zero and λ is the Filter Time Constant.

The Final Equation for IMC-PID Design Technique :

$$C(s) = \frac{(\alpha s + 1)(0.01s + 2)}{100s(2\lambda - 2\alpha + 2 * 0.01)}$$

The Controller can be expressed in PID form as :

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

The values of the variables can be given as :

$$\alpha = \frac{2 * \lambda * 100 + 0.01 * \lambda + 2 * 100 * 0.01}{2 * 100 - 0.01}$$

$$k_c = \frac{2\alpha + 0.01}{2 * 100(\alpha - \lambda - \ 0.01)}$$

$$\tau_i = \alpha + \frac{0.01}{2}$$

$$\tau_d = \frac{0.01\alpha}{2\alpha + 0.01}$$

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

1. $K_p = k_c$

$$K_p = \frac{2\alpha + 0.01}{2 * 100(\alpha - \lambda - \ 0.01)}$$

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

$$K_i = \frac{2\alpha + 0.01}{100(\alpha - \lambda - \ 0.01)(\alpha + 0.01)}$$

3. $K_d = k_c * \tau_d$

$$K_d = \frac{0.01\alpha}{2 * 100(\alpha - \lambda - \ 0.01)}$$

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

Live Editor - C:\Users\Sayan\Desktop\LNMIIT\Sem6\NLACD\Final Working\lvsc.mlx

lvsc.mlx

```
clear;
k = 100
tau = 100
theta = 0.01
lambda =1

alfa = (2*lambda*tau + lambda*theta + 2*tau*theta)/(2*tau-theta)

kp = (2*alfa + theta)/(2*k*(alfa-lambda-theta))
ki = kp*(alfa + (theta/2))
kd = (kp*alfa*theta)/(2*alfa + theta)
```

```
k = 100
tau = 100
theta = 0.0100
lambda = 1

alfa = 1.0101

kp = 101.0000
ki = 102.5251
kd = 0.5025
```

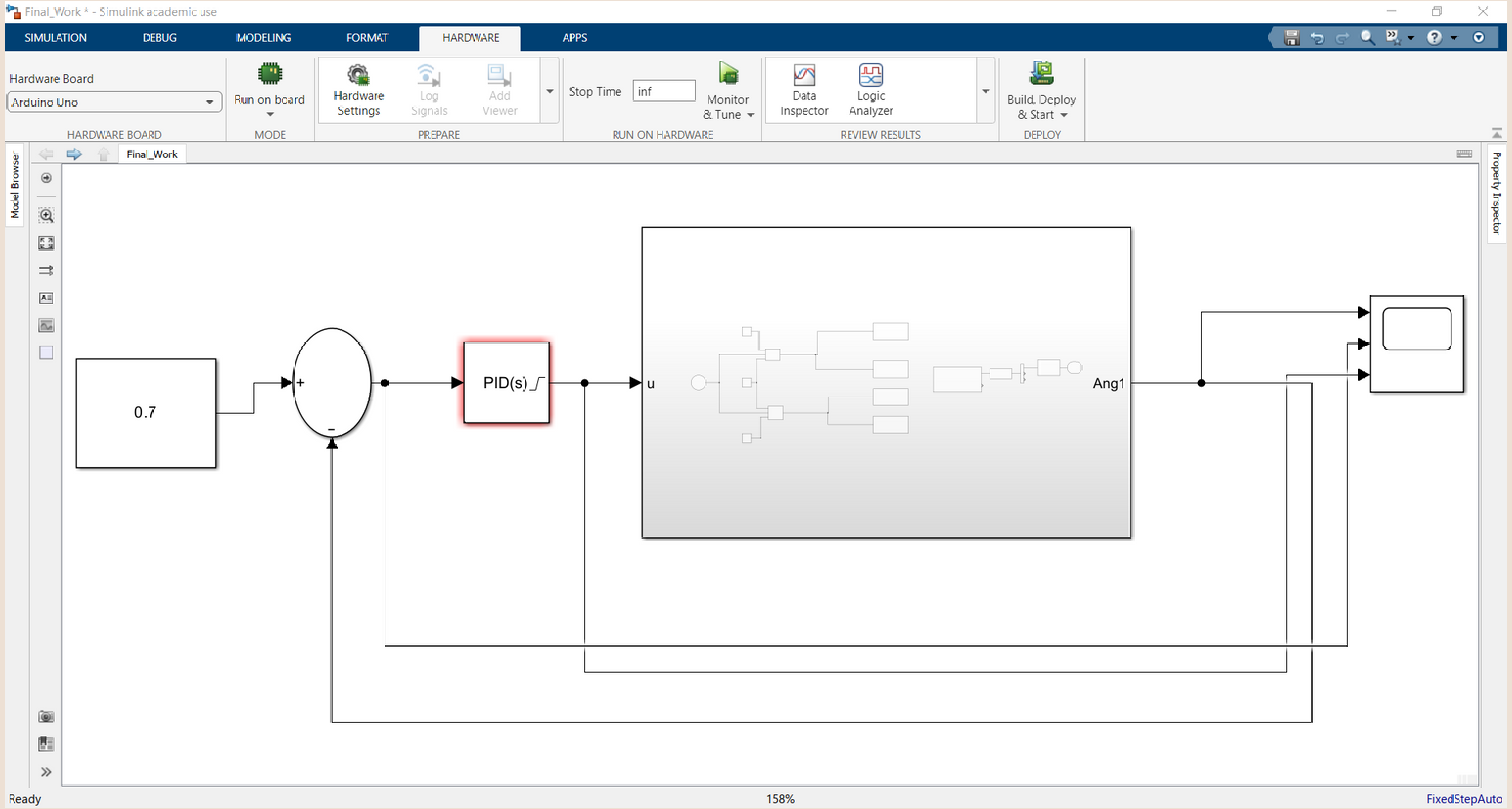
Command Window

For various values of λ, the PID gains are found. These values are used to design the PID controller.

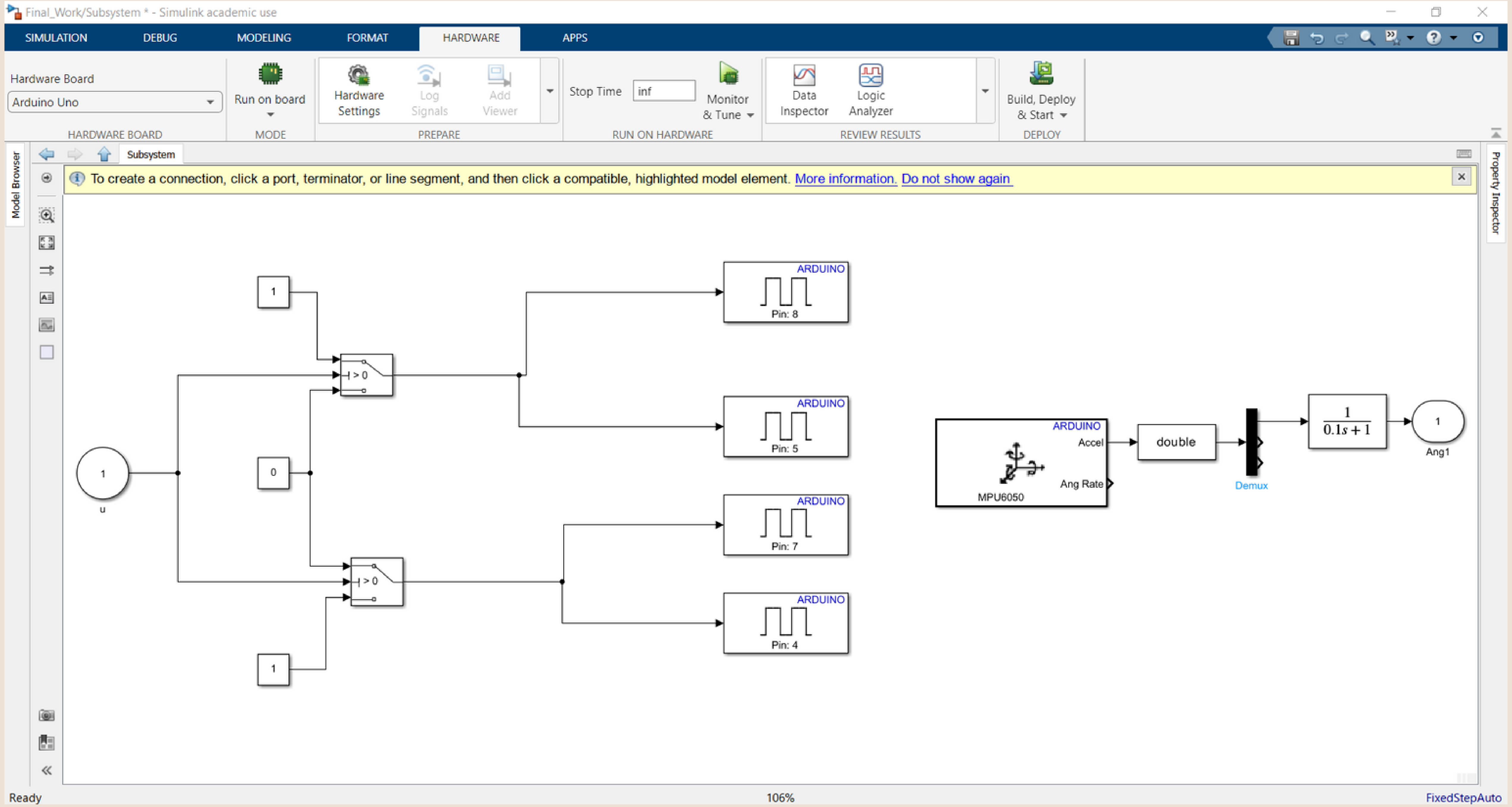
Lambda Values	Kp	Ki	Kd
0.2	104.8828	22.5520	0.5122
0.7	101.4234	72.5249	0.5036
1	101.0000	102.5251	0.5025
2	100.5037	202.5352	0.5013
4	100.2547	402.5627	0.5006

TABLE 3.1: PID values for various lambda

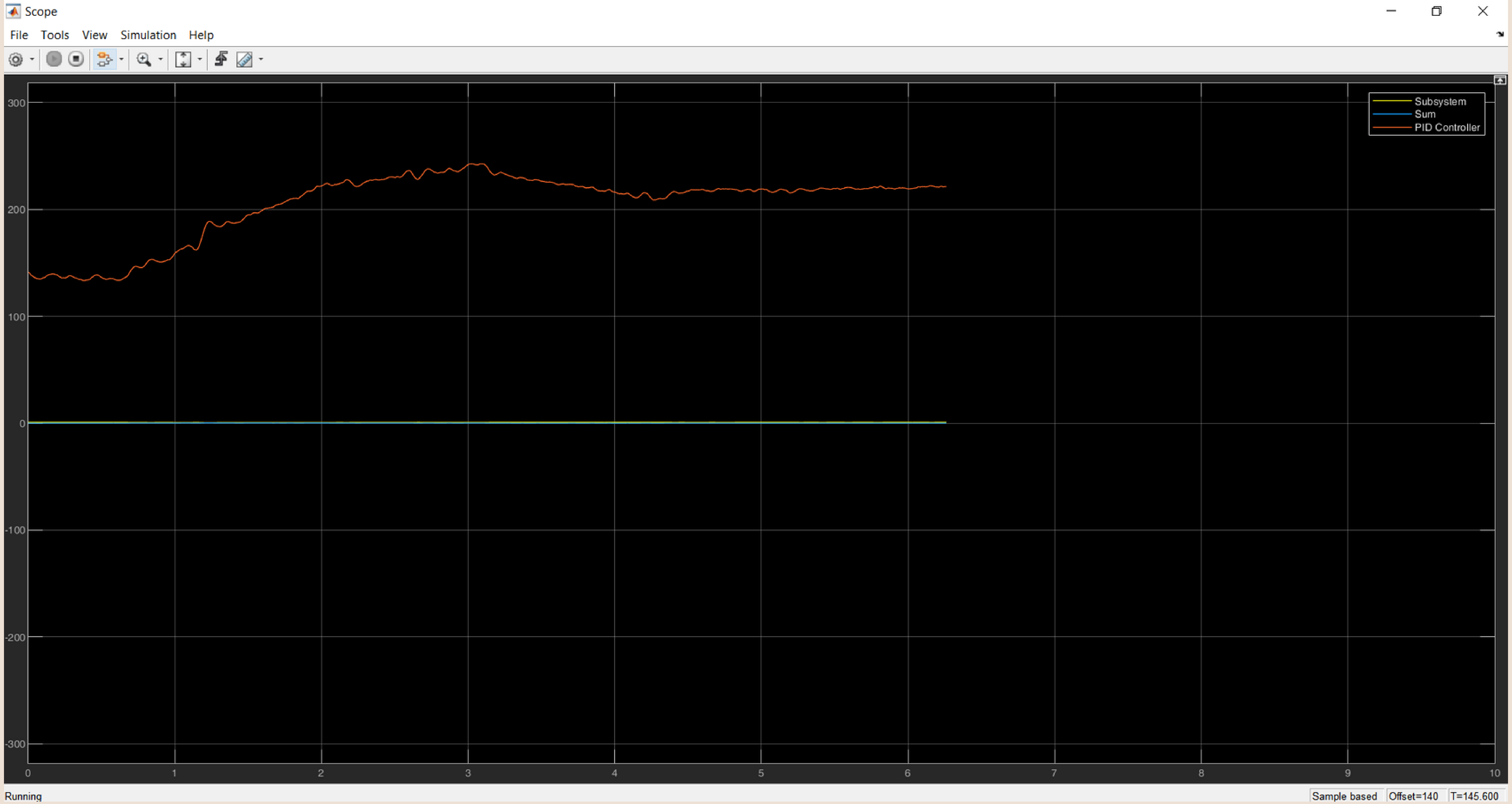
The Simulink Model



The Subsystem

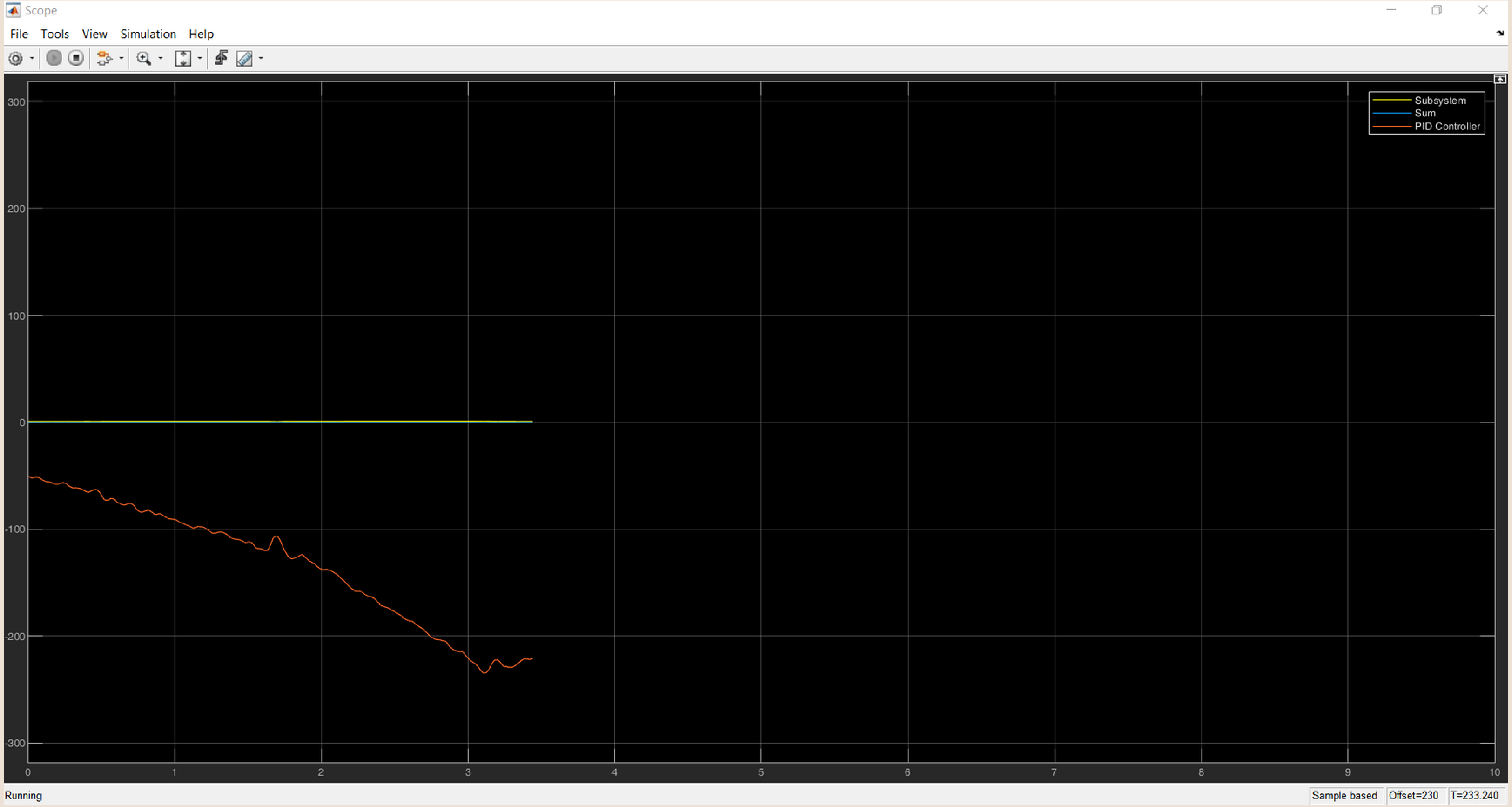


The following graph shows the control action depending on the values of the gyroscope :



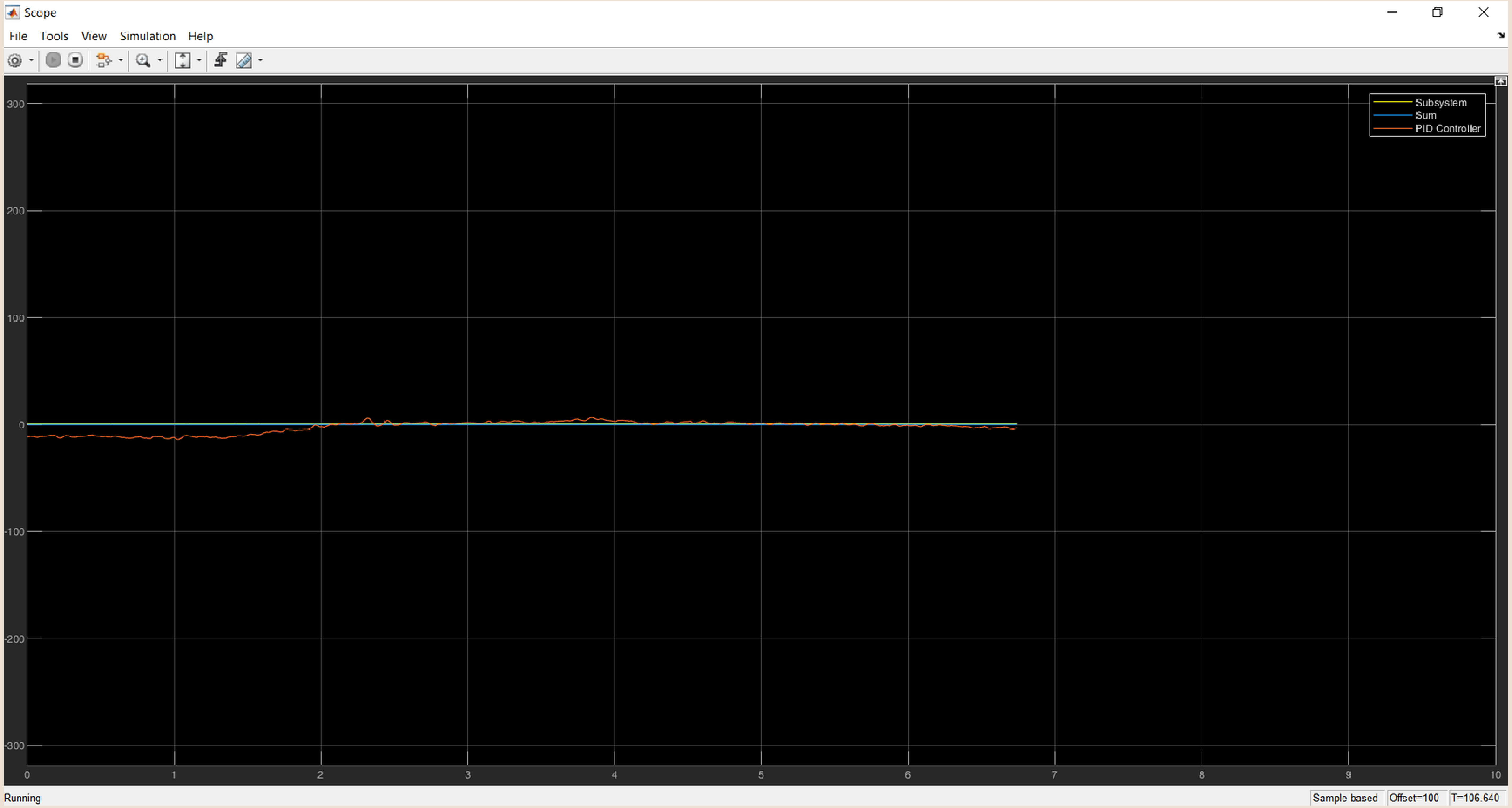
When the model falls forward

The following graph shows the control action depending on the values of the gyroscope :



When the model falls backward

The following graph shows the control action depending on the values of the gyroscope :



When the model is at equilibrium position

Conclusions :

- We were able to design the PID controller that provides the required control action to drive the motors which should balance the motors.

Future Work Required :

- Further Tuning of the PID controller for smoother working of the model and removal of the jerking action.
- Using Model Reference Adaptive Control to make the system adjust the controller in real-time.
- Adding a LiPo Battery Source to provide a stable power supply to the model.

LNMIIT
The LNM Institute of Information Technology

Thank you!

Supervisor

Dr. Bharat Verma

Students

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