

Model Reference Adaptive Control Design for Self Balancing Robot

Supervisor

Dr. Bharat Verma

Students

Divyansh Jain - 20uec049

Satvik Sharma - 20uec116

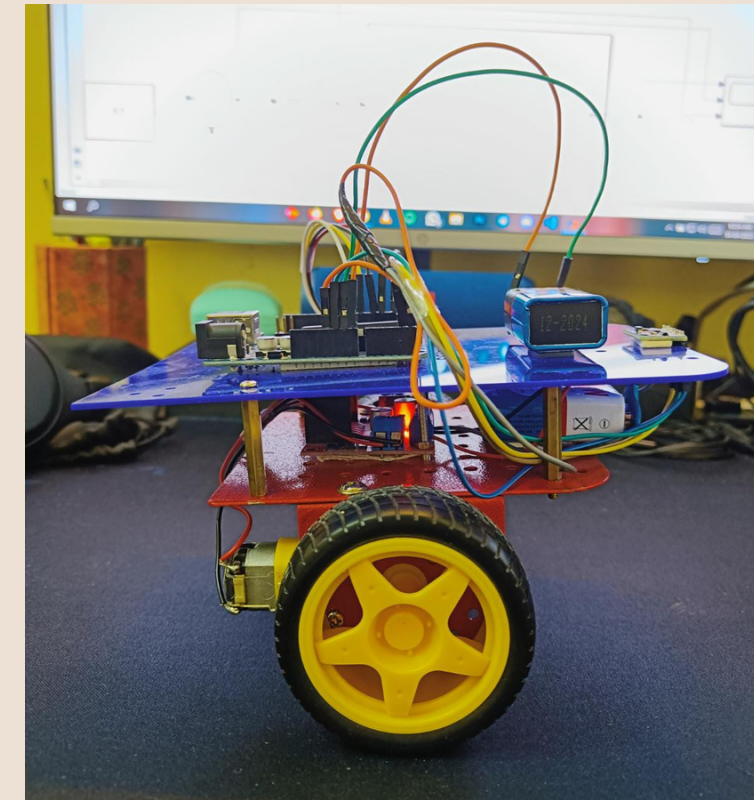
Sayan Chatterjee - 20uec120

Table of Contents

I	Overview	3
II	Mechanical Structure of the Robot	4
III	Designing the System	8
IV	Results	17
V	Conclusions & Future Work	20

I OVERVIEW

- **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**

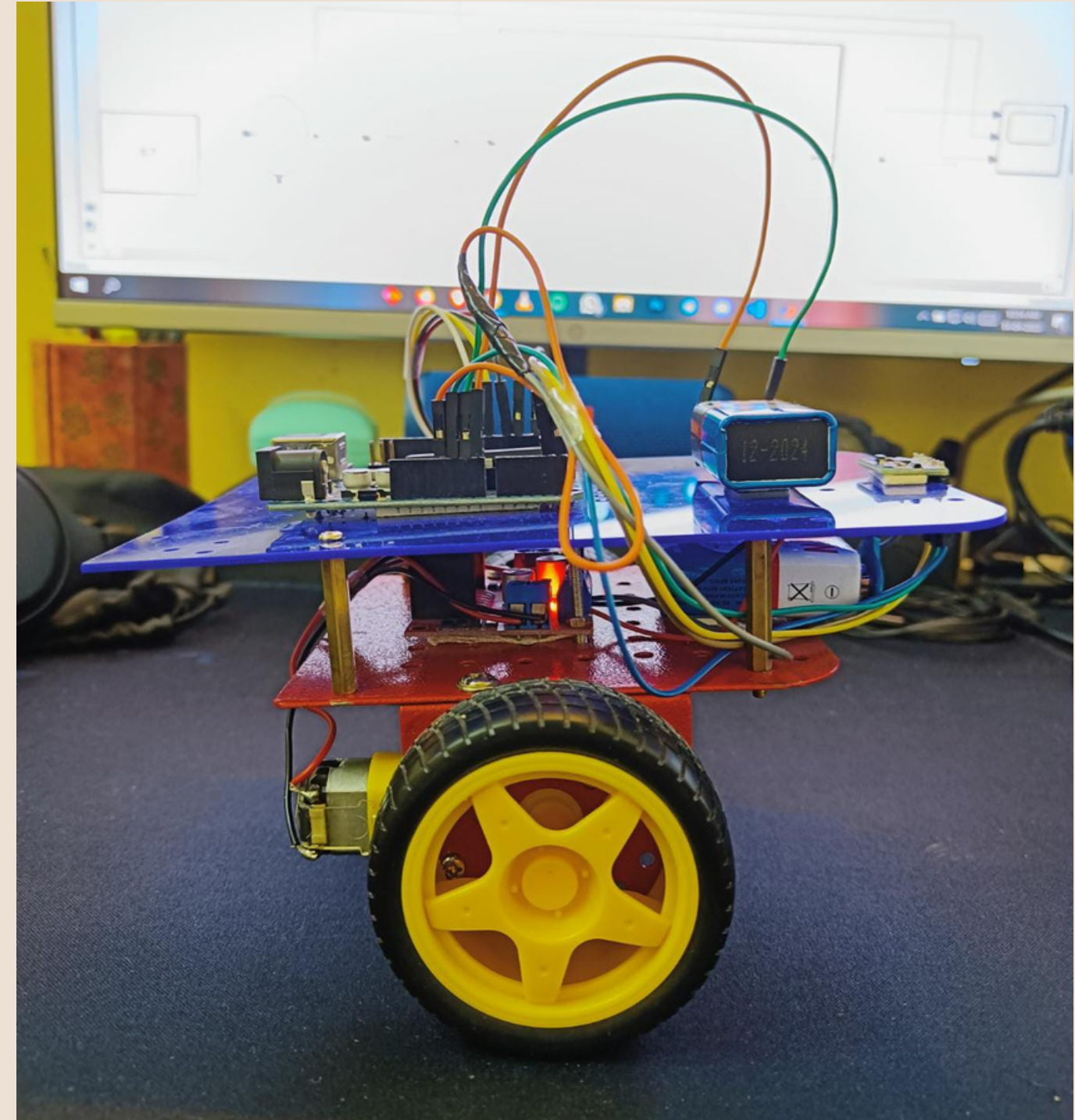
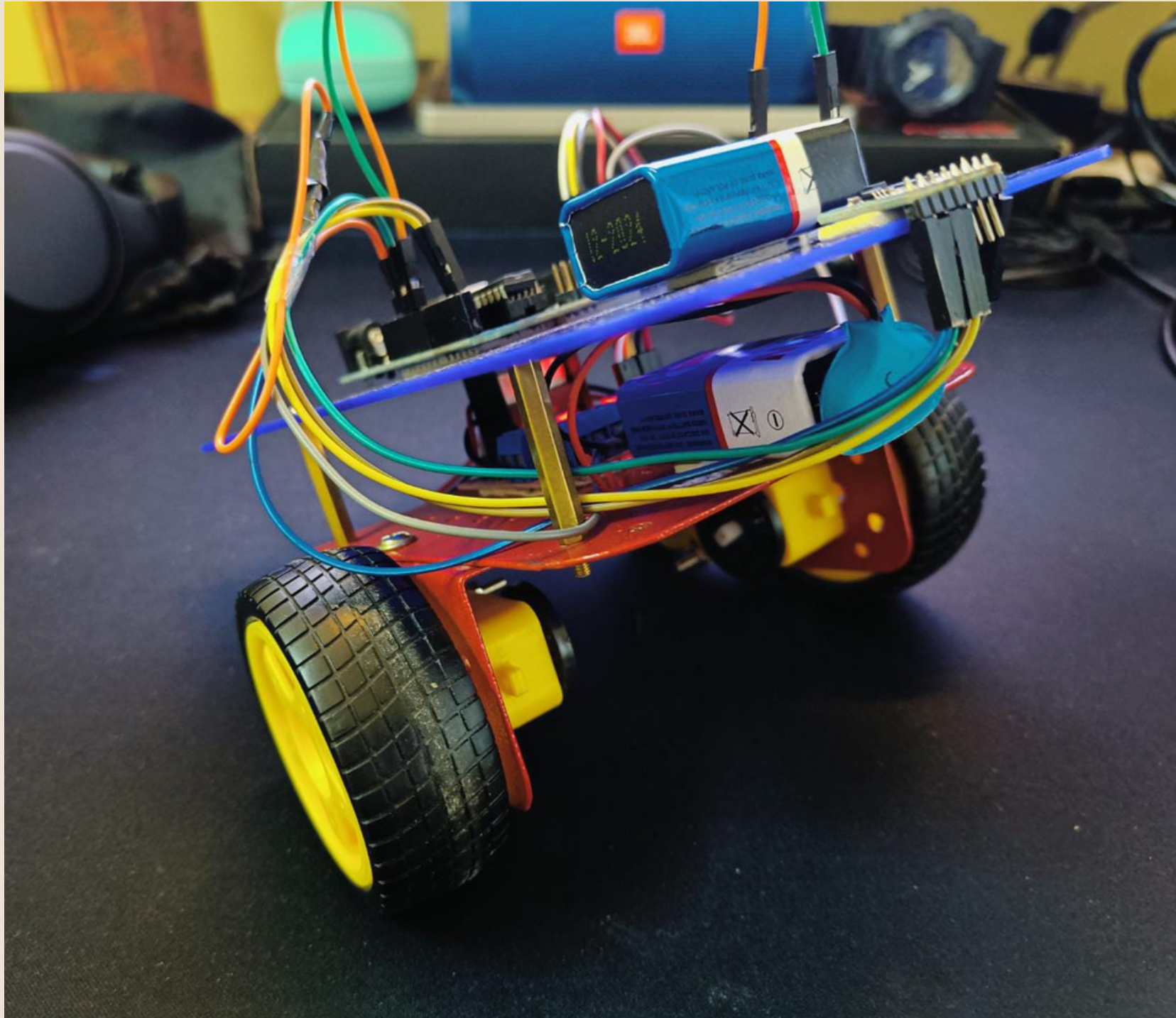


The PID controller constantly measures the robot's angle and compares it to the desired angle. If the robot's angle is not equal to the desired angle, the controller calculates an error value and adjusts the motor to correct the error. The proportional component of the PID controller adjusts the motor 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 purpose of Derivative control is to provide a “dampening” effect that can help limit overshoot and converge more quickly upon the Setpoint

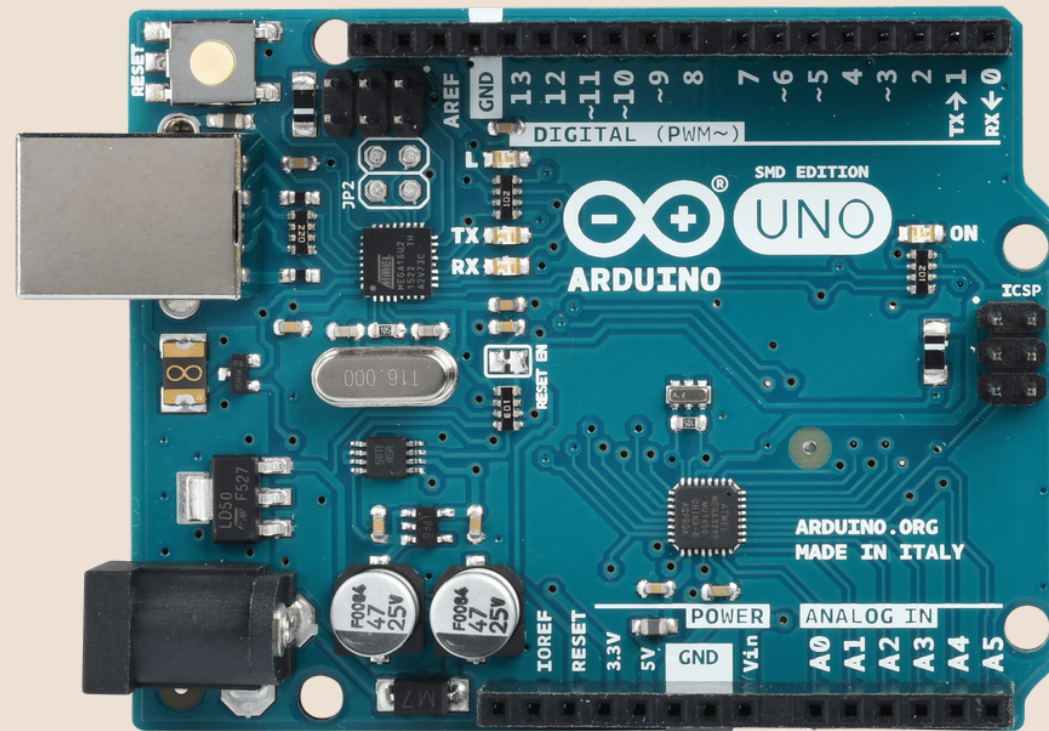
The integral component of the control algorithm can remove any steady-state error in the system

II MECHANICAL STRUCTURE OF THE ROBOT

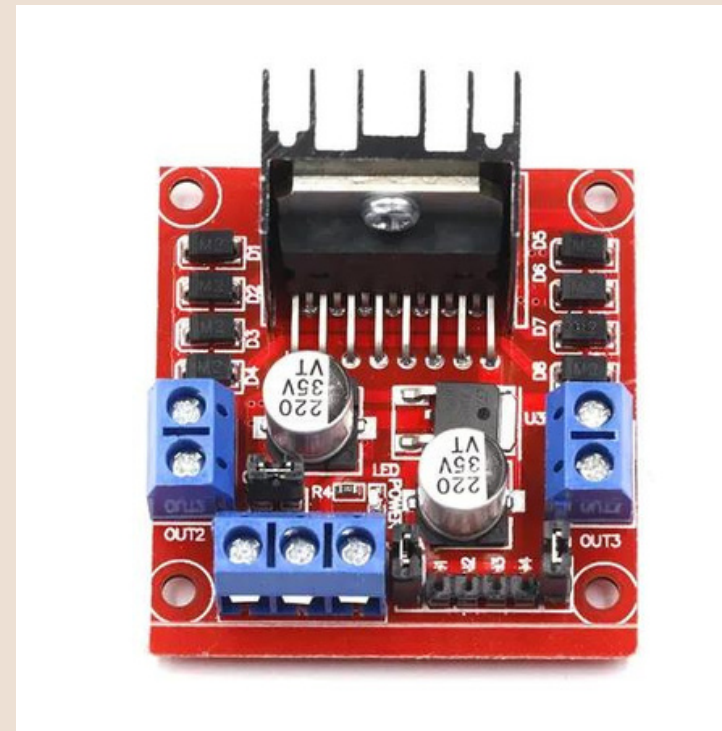


The Components used

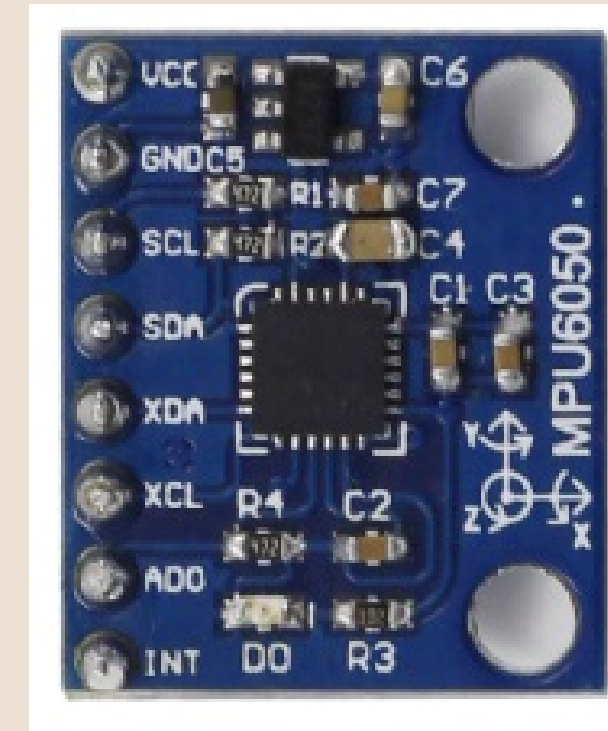
II MECHANICAL STRUCTURE OF THE ROBOT



Arduino Uno Microcontroller



L298N Motor Driver Module



MPU6050 Gyroscope

II MECHANICAL STRUCTURE OF THE ROBOT



BO Motors



Wheels



2 Wheel Chassis



9V Batteries

III DESIGNING THE SYSTEM

The Transfer Function

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

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

where, $K = 800$ is the Gain,
 $\tau = 1000$ is the Time Constant
 $\theta = 0.02$ is the Delay

III DESIGNING THE SYSTEM

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 :

$$C(s) = \frac{(\alpha s + 1) (s \theta + 2)}{K s (2 \lambda - 2 \alpha + 2 \theta)}$$

The Controller can be expressed in PID form as :

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

The values of the variables can be given as :

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

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

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

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

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

$$1. K_p = k_c$$

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

$$3. K_d = k_c * \tau_d$$

III DESIGNING THE SYSTEM

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

```
clear;  
k = 800  
tau = 1000  
theta = 0.02  
lambda = 0.5  
  
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)  
  
s=tf('s');  
G=.8*exp(-theta*s)/s  
C=pid(kp,ki,kd,0.01);  
step(feedback(G*C,1))
```

III DESIGNING THE SYSTEM

The results of the LiveScript

```

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

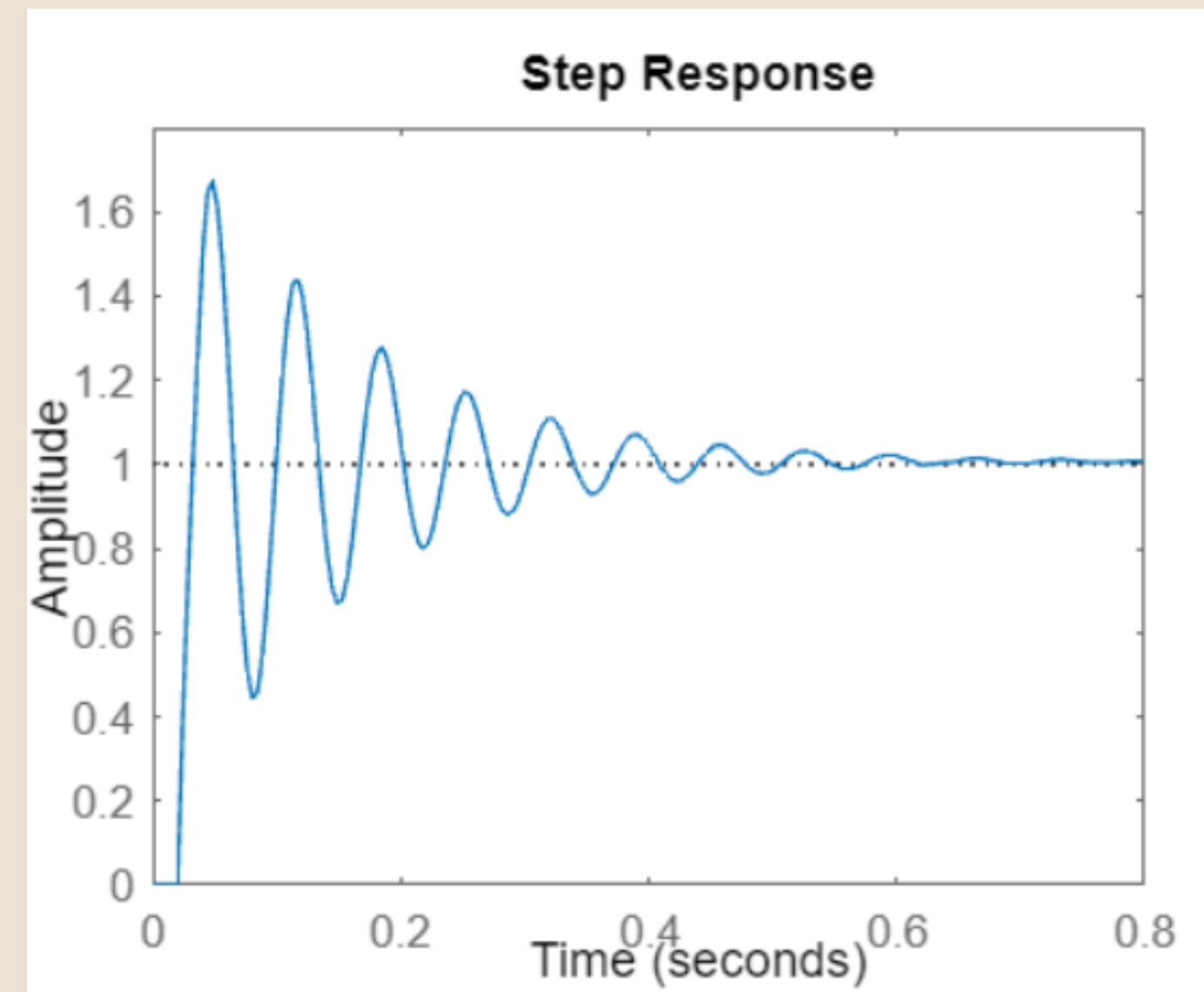
alfa = 0.5200

kp = 64.9516
ki = 34.4250
kd = 0.6373
G =

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

Continuous-time transfer function.

```

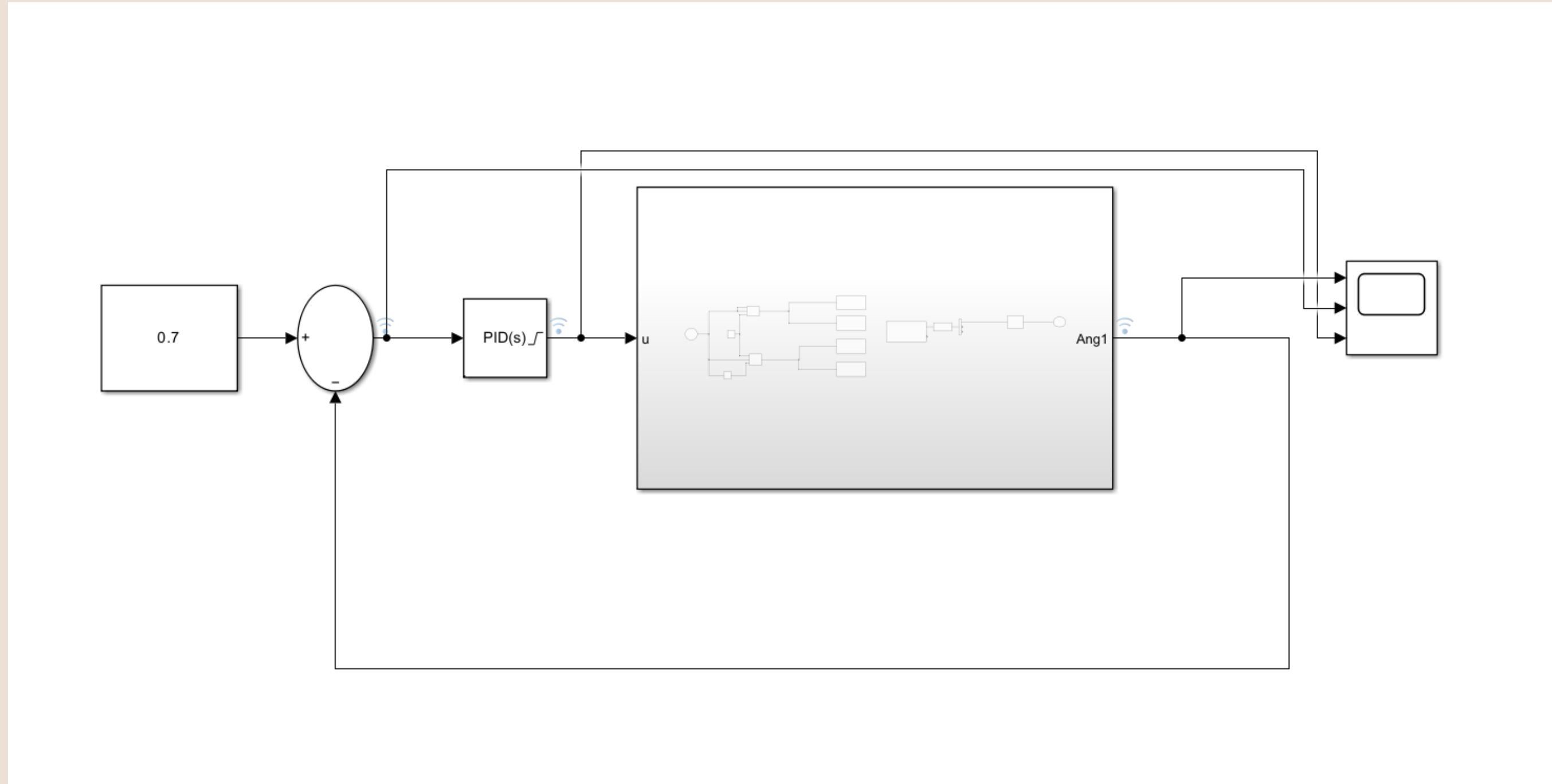


For various values of λ , the PID gains are found. These values are used to design the PID controller.

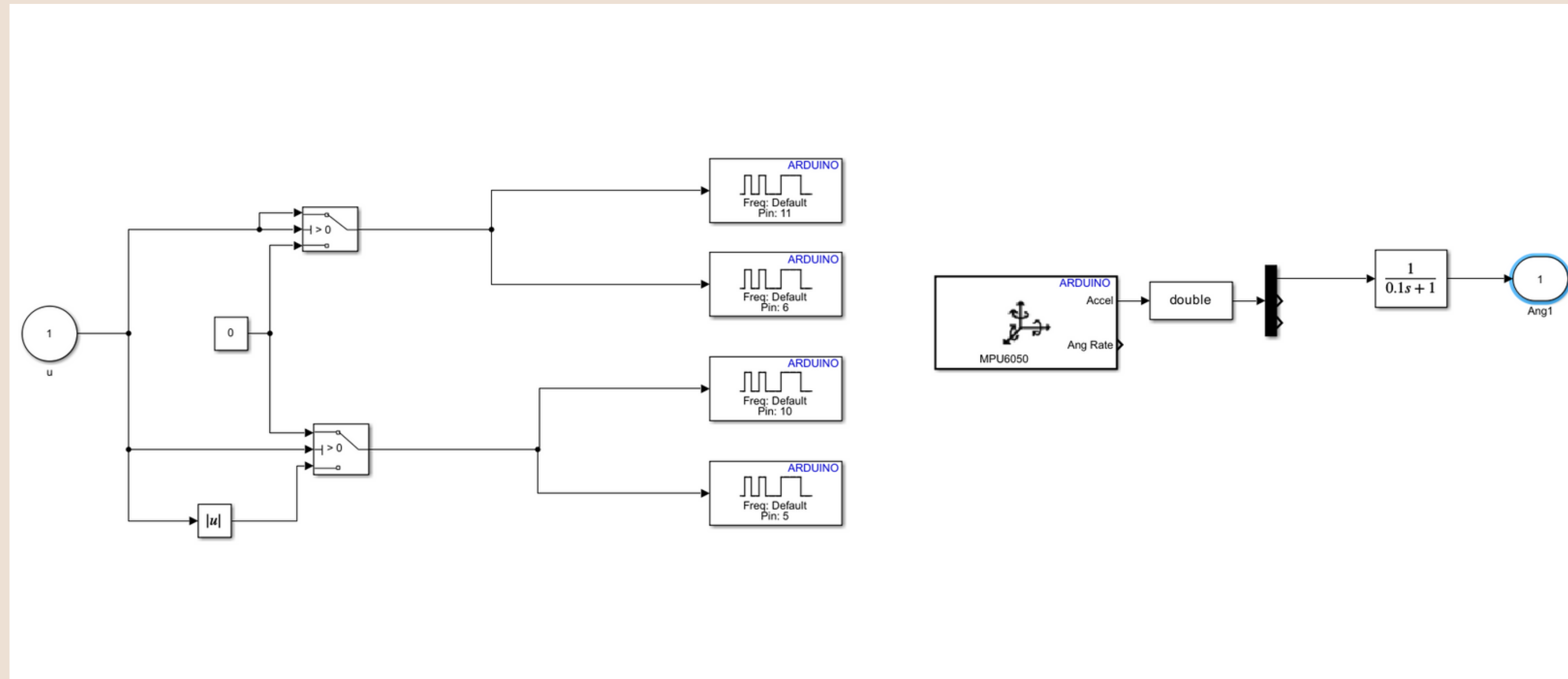
Lambda Values	Kp	Ki	Kd
0.2	104.8828	22.5520	0.5122
0.7	101.4234	72.5249	0.5036
1	101.0000	102.5251	0.5025
2	100.5037	202.5352	0.5013
4	100.2547	402.5627	0.5006

TABLE 3.1: PID values for various lambda

The Simulink Model

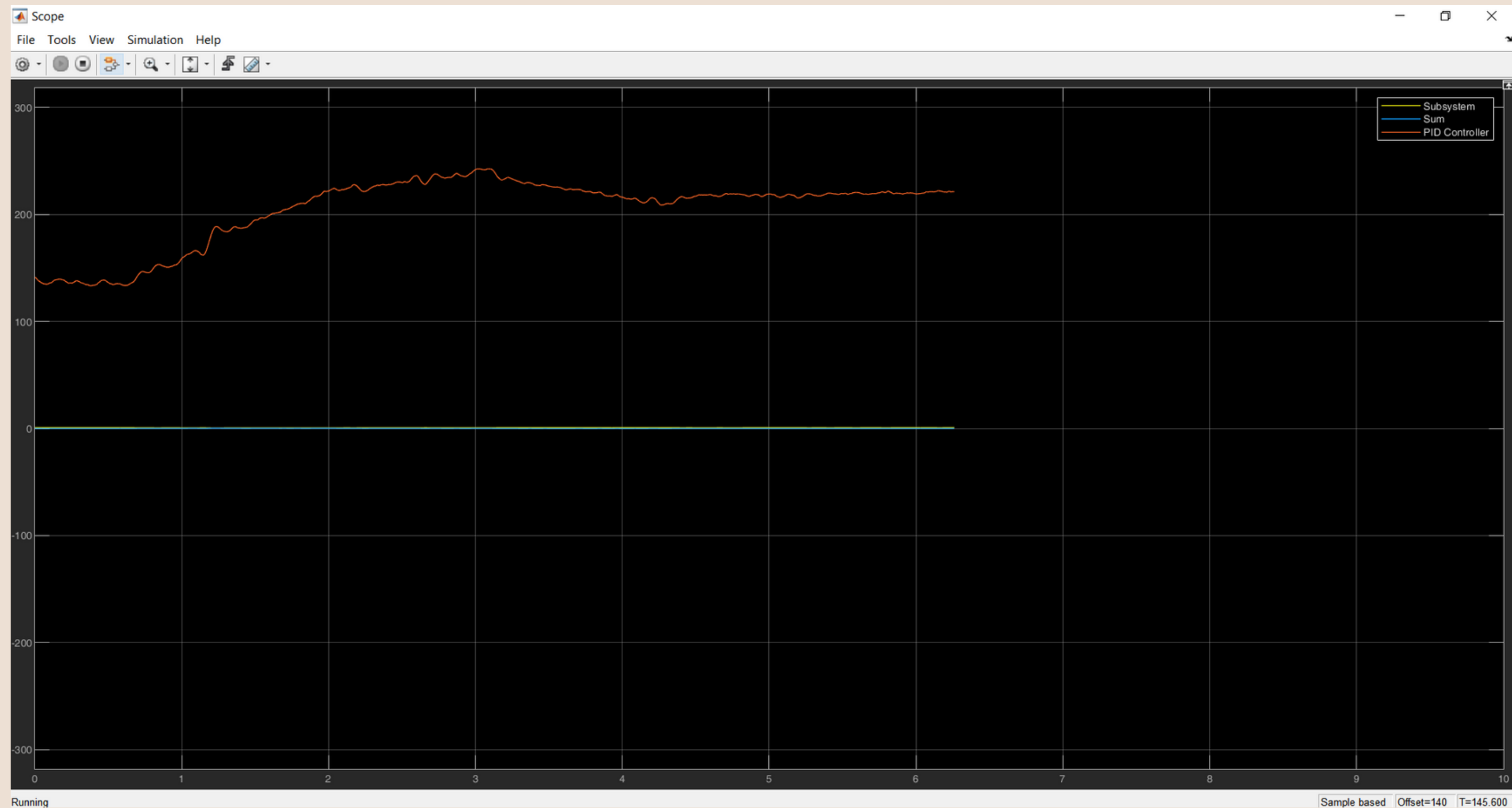


The Subsystem



IV RESULTS

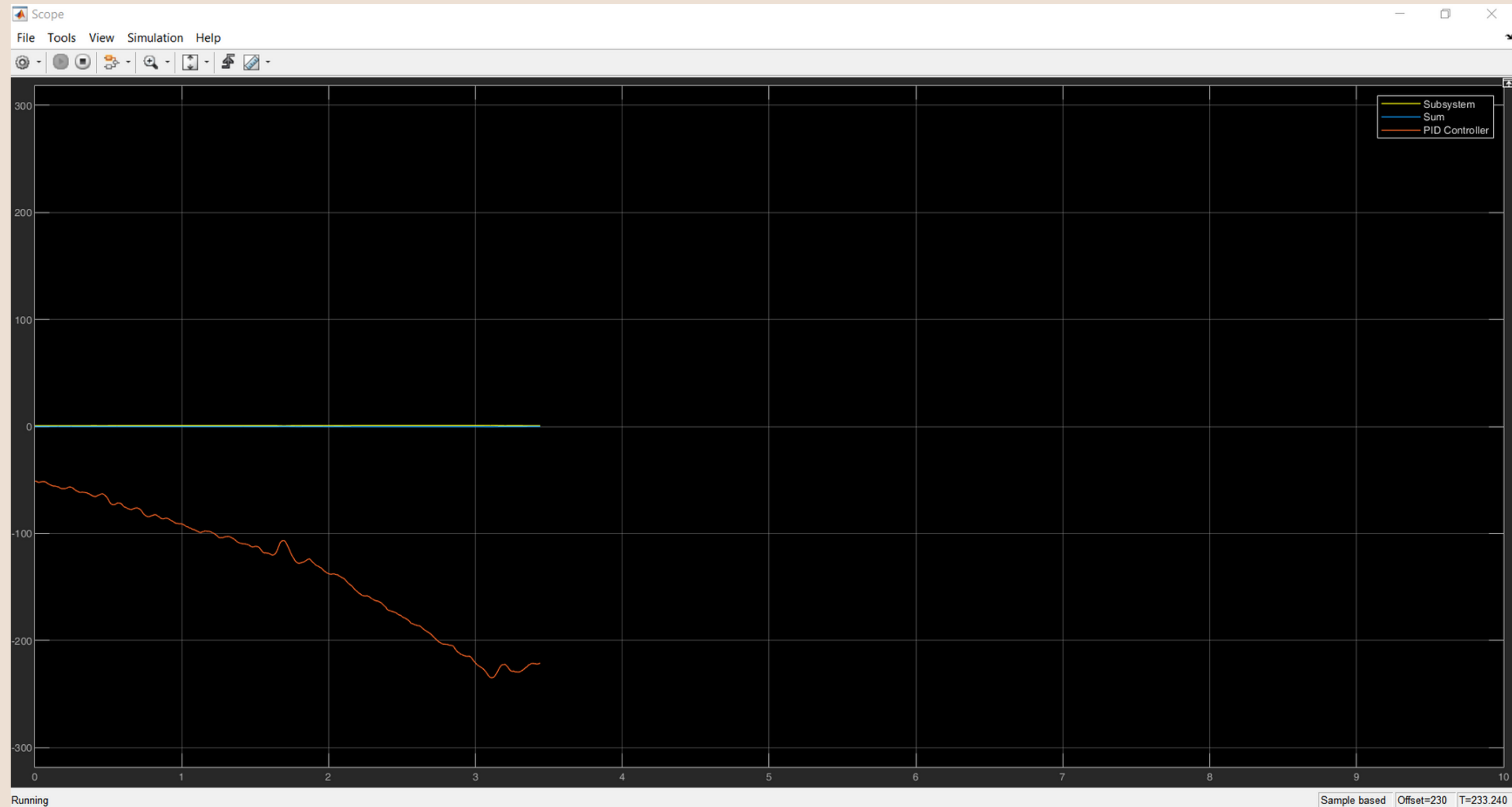
The following graph shows the control action depending on the values of the gyroscope :



When the model falls forward

IV RESULTS

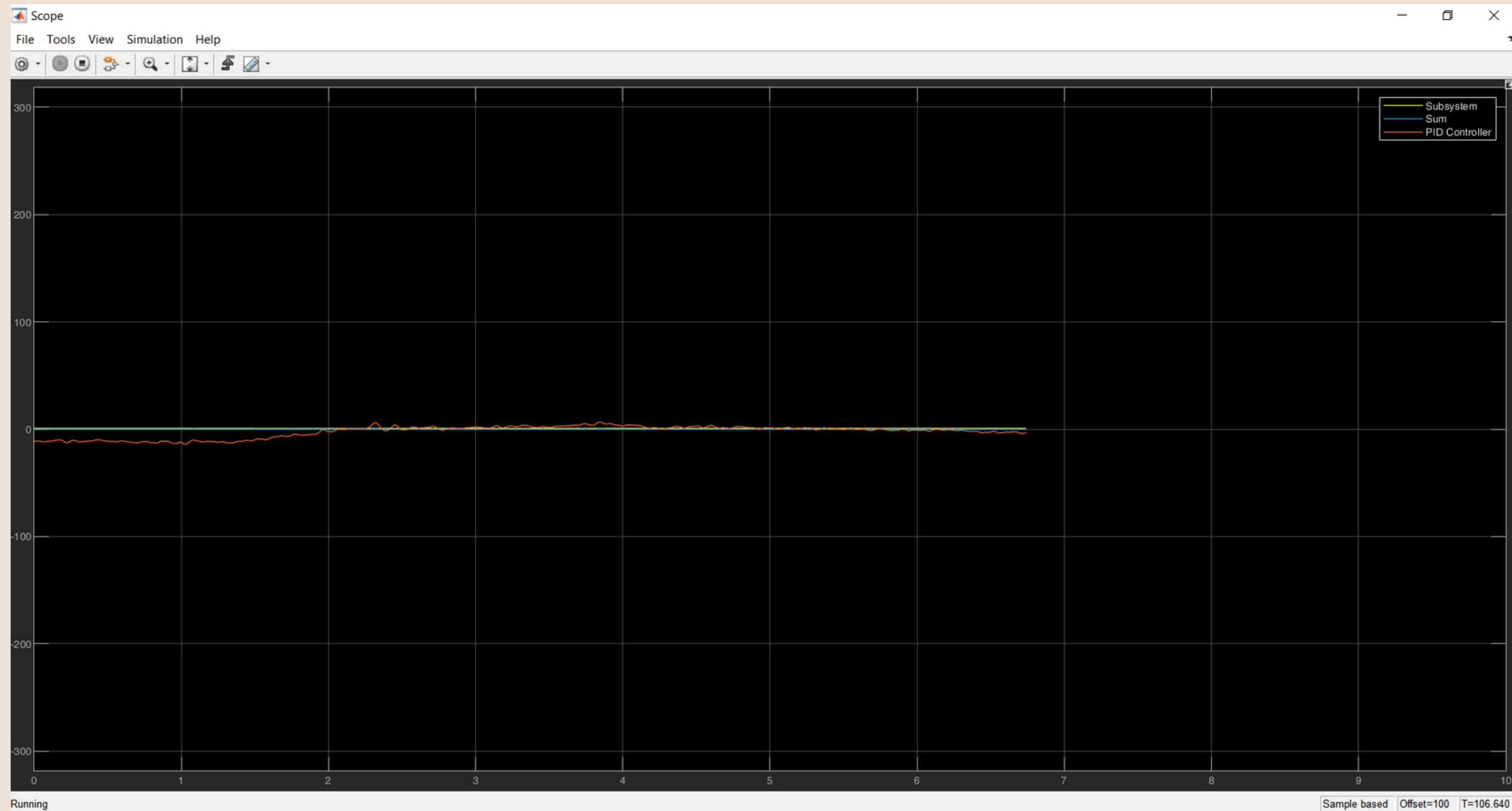
The following graph shows the control action depending on the values of the gyroscope :



When the model falls backward

IV RESULTS

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



When the model is at equilibrium position

V CONCLUSION & FUTURE WORK

Conclusion :

- 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.
- Adjust the chassis so that the centre of mass falls at the centre of the body.
- Adding an algorithm that stops the motor when the model gets in an irrecoverable state.
- Adding a LiPo Battery Source to provide a stable power supply to the model.

Thank you!

Supervisor

Dr. Bharat Verma

Students

Divyansh Jain - 20uec049

Satvik Sharma - 20uec116

Sayan Chatterjee - 20uec120