



# MCT441 | Rehabilitation Robotics

## Major Task – Team 1

**Name:**

Omar Shawky Mohamed

Mohamed Eyad Sayed

Muhammed Abdullatif Saber

**ID:**

18P5316

18P8431

18P6971

**Submitted to:**

Prof. Dr. Mohammed Awad

Eng. Hamdy Osama

## Table of Contents

1	Introduction.....	4
2	Components Used:.....	5
3	Methods of Control Implementation:.....	7
3.1	Current control .....	7
3.2	Force control .....	9
3.3	Position control .....	11
3.4	Admittance control.....	13
3.5	Impedance control .....	15
4	Cad Overview .....	17
4.1	Home position .....	17
4.2	Top view.....	17
4.3	Side view .....	18
4.4	Base view .....	19
4.5	Close up.....	19
5	MATLAB Simulation:.....	20
5.1	Current Control .....	20
5.2	Force Control.....	21
5.3	Position Control.....	22
5.4	Admittance Control .....	23
5.5	Impedance Control .....	25
6	Full project link / videos link:.....	27

## Table of Figures

Figure 1: Robots used in Rehabilitation.....	4
Figure 2: Current Control Flowchart .....	7
Figure 3: Current values before Filtering.....	8
Figure 4: Current values after Filtering .....	8
Figure 5: Force Control Flowchart .....	9
Figure 6: Force Control Serial Plotter, with Set Point = Zero .....	10
Figure 7: Force Control Serial Plotter, Force Readings vs Motor Position.....	10
Figure 8: Position Control Flowchart .....	11
Figure 9: Position Control Serial Plotter, Set Point = 90 degrees.....	12
Figure 10: Admittance Control Flowchart.....	13
Figure 11: Impedance Control Flowchart.....	15
Figure 12: Home Position of the motor .....	17
Figure 13: Top View of the CAD .....	17

Figure 14: Side View of the CAD.....	18
Figure 15:Side View of the CAD.....	18
Figure 16: Base View of the CAD.....	19
Figure 17: Full CAD showing all Sensor and the Motor.....	19
Figure 18: Current Control Block Diagram on MATLAB.....	20
Figure 19: Current Control Outputs on MATLAB.....	21
Figure 20: Force Control Block Diagram on MATLAB.....	21
Figure 21: Force Control Output on MATLAB.....	22
Figure 22: Position Control Block Diagram on MATLAB.....	22
Figure 23: Position Control Output on MATLAB.....	23
Figure 24: Admittance Control Block Diagram on MATLAB.....	24
Figure 25: Force Output in Admittance Control.....	24
Figure 26: Position Output in Admittance Control.....	25
Figure 27: Impedance Control Block Diagram on MATLAB.....	26
Figure 28: Tuning of PID Parameters on MATLAB.....	26

# 1 Introduction

Rehabilitation robotics is an increasingly important field that aims to assist individuals with physical impairments in their recovery and rehabilitation using robots. One of the key considerations in the design and use of these robots is the type of control that is implemented. There are several different types of control that can be used in rehabilitation robotics, including current control, position control, force control, admittance control, and impedance control. Each of these types of control has its own unique advantages and limitations, and the most appropriate control method will depend on the specific needs and requirements of the user.




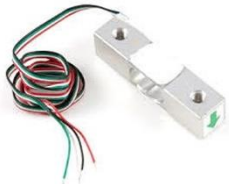




*Figure 1: Robots used in Rehabilitation*


## Control Methods theory:

- **Current control** is a type of control that involves the manipulation of the electrical current flowing through the motors of a rehabilitation robot. This allows in direct control over the output torque of the actuators and thus the joints. This can be also be used as a safety feature to limit the current passing through the coils and avoid damaging the actuators.
- **Position control** is another type of control that is commonly used in rehabilitation robotics. It involves the use of sensors and algorithms to accurately control the position of the robot in relation to a set of predetermined coordinates.
- **Force control** is a type of control that is used to manipulate the forces applied by a rehabilitation robot. This can be useful for tasks that require the robot to interact with the environment or with a human user, as it allows the robot to respond to external forces in a more natural and intuitive manner.
- **Admittance control** is a type of control that enables the user to move the robot to any position without resistance and after the user applied force is released the robot will return to its original position
- **Impedance control** is a type of control that is used to manipulate the resistance of a rehabilitation robot to external forces. This allows implementing a virtual spring that applies higher forces as the joint gets further from the initial position.

By understanding the different types of control available, researchers and clinicians can choose the most appropriate control method for the specific needs of their users.

## 2 Components Used:

<p><b>DC motor with encoder</b></p>	
<p><b>Load cell 10KG</b></p>	
<p><b>Load cell Amplifier HX711</b></p>	
<p><b>Current Sensor ACS712 (AC or DC) 5A</b></p>	
<p><b>Dual Monster Motor Shield VNH2SP30</b></p>	
<p><b>Arduino Uno</b></p>	

<b>12 Volt battery</b>	
<b>Laser Cut wood</b>	
<b>3D printed parts</b>	
<b>Breadboard</b>	

### 3 Methods of Control Implementation:

#### 3.1 Current control

this code is setting up a system to control a motor using current feedback. The variables **prevT**, **eprev**, and **eintegral** is related to the control algorithm being used, as part of a PID (proportional-integral-derivative) controller. The variable **currentReading** is used to store the raw, unfiltered current reading from the motor, while **initial** and **current\_initial** is used to keep track of the initial current reading when the system is first started. The variable **CurrentReal** is used to store the filtered, real-time current reading of the motor.

The code is also setting up a low-pass filter object using the LowPass class, with a cutoff frequency of 3 Hz and a sample frequency of 1000 Hz. The filter object is being created with the adaptive flag set to true, which means that the sample frequency will be automatically set based on the time history. The LowPass class has a number of functions for setting the filter coefficients, filtering raw values, and updating the filter coefficients as needed.

In the setup function, the pins are being set up as either inputs or outputs, and the serial port is being initialized with a baud rate of 9600. The limit switch objects are being set up with the ezButton library and are being attached to pins 6 and 9.

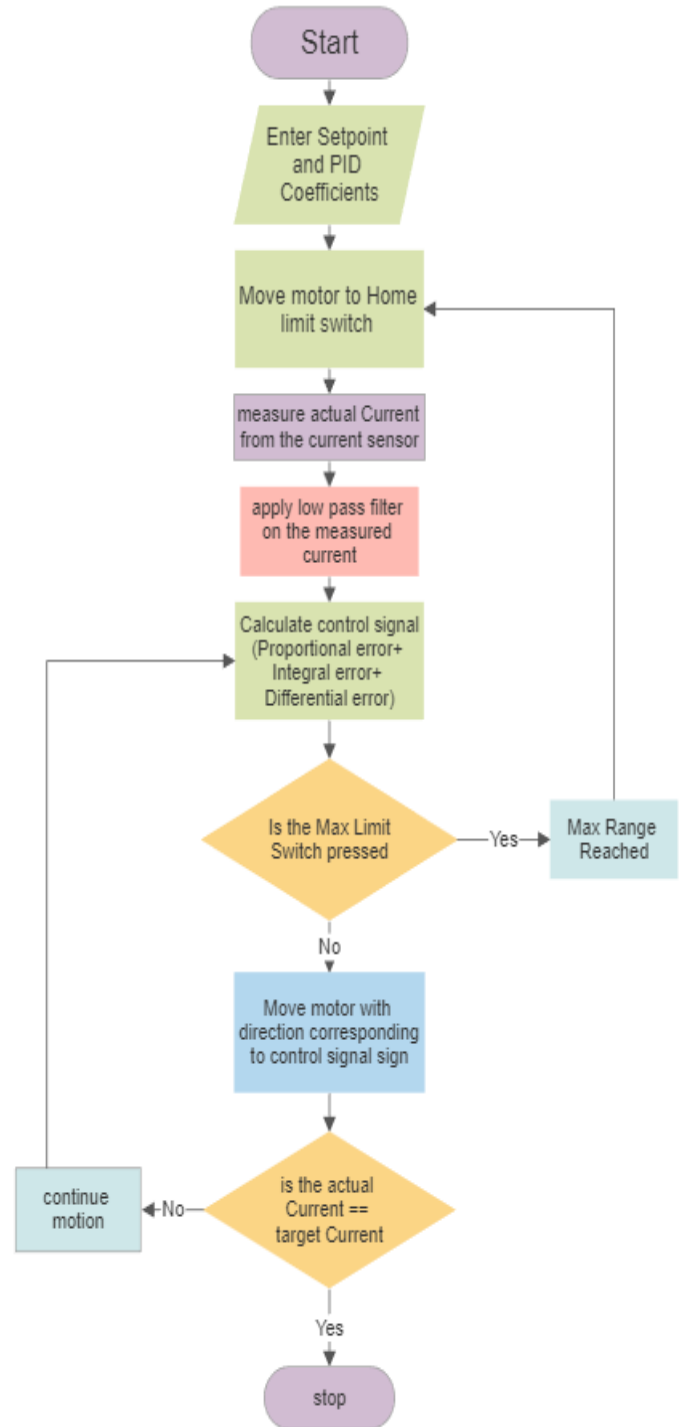


Figure 2: Current Control Flowchart

The next screenshot from the serial monitor of the Arduino IDE shows the current readings of the current sensors before filtering.

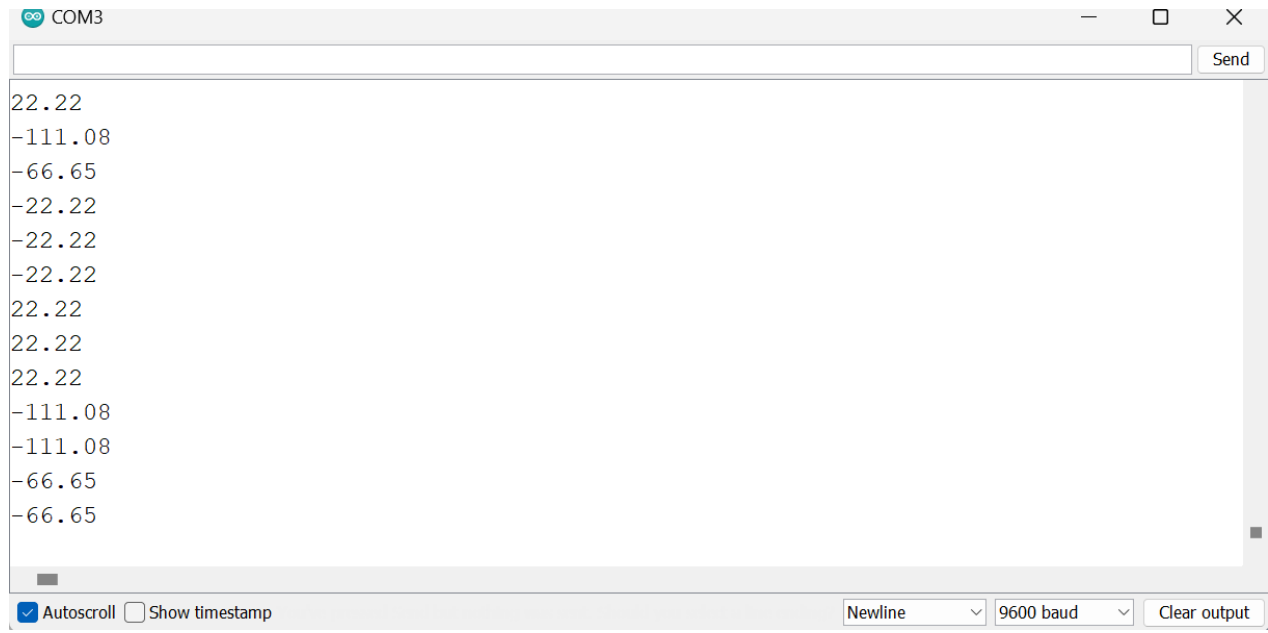


Figure 3: Current values before Filtering

We can notice that the readings are very random and cannot be controlled in this way. That is why the idea of filtering came, we can notice in the following screenshot that the readings became more logic and can be controlled in this way.

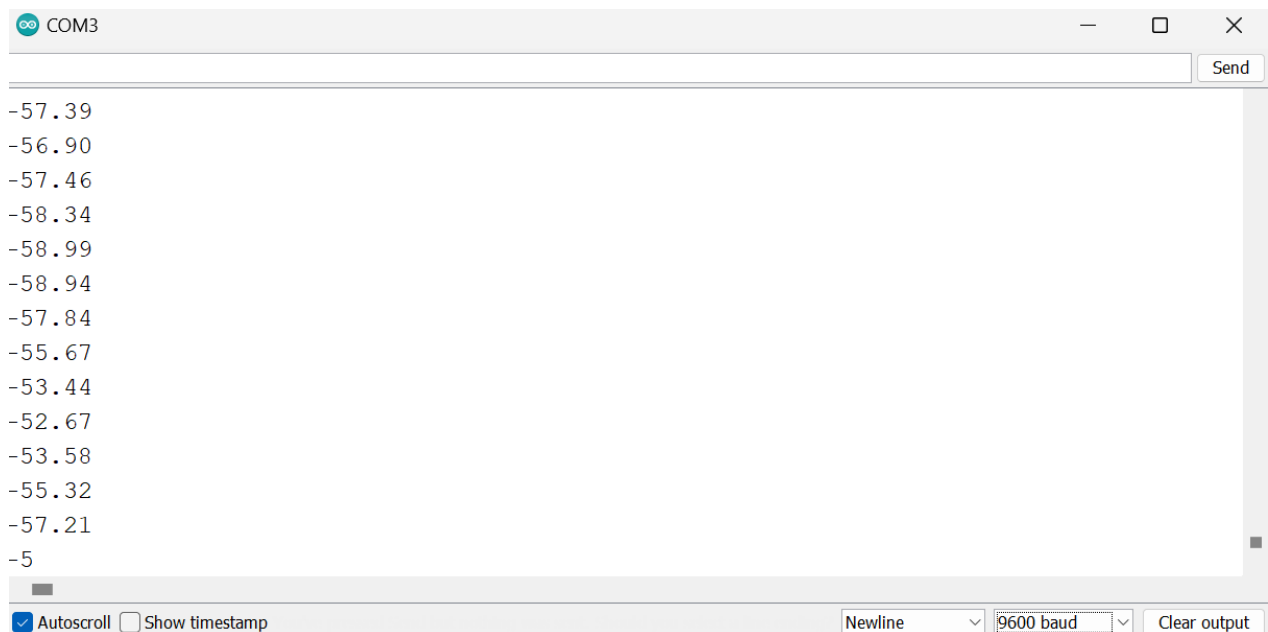


Figure 4: Current values after Filtering

**Code link:**

[https://drive.google.com/drive/folders/1MJxtsjhnb4KKBf1W4pJ2arA6Cifh1DAL?usp=share\\_link](https://drive.google.com/drive/folders/1MJxtsjhnb4KKBf1W4pJ2arA6Cifh1DAL?usp=share_link)



### 3.2 Force control

This code is for a motor control system that uses **force control**. Force control involves the use of sensors to measure the force being applied to or by the system, and algorithms to control the movement of the system based on this information.

In this code, the force being applied to the system is measured using a load cell, which is a device that converts applied force into an electrical signal. The load cell readings are smoothed using a moving average filter and then used to calculate the error between the target force and the actual force.

The error is then used in a PID (proportional-integral-derivative) control algorithm to calculate the control signal ( $u$ ) that will be applied to the motor. The PID algorithm uses the current error (the difference between the target force and the current force), the derivative of the error (the rate of change of the error), and the integral of the error (the accumulated error over time) to calculate the control signal.

The control signal is then used to set the direction and power of the motor, using the `setMotor()` function. The direction of the motor is set based on the sign of the control signal (positive for one direction, negative for the other), and the power of the motor is set to the absolute value of the control signal, with a maximum value of 100.

Overall, this code is implementing a force control system for a motor, using a load cell and a PID control algorithm.

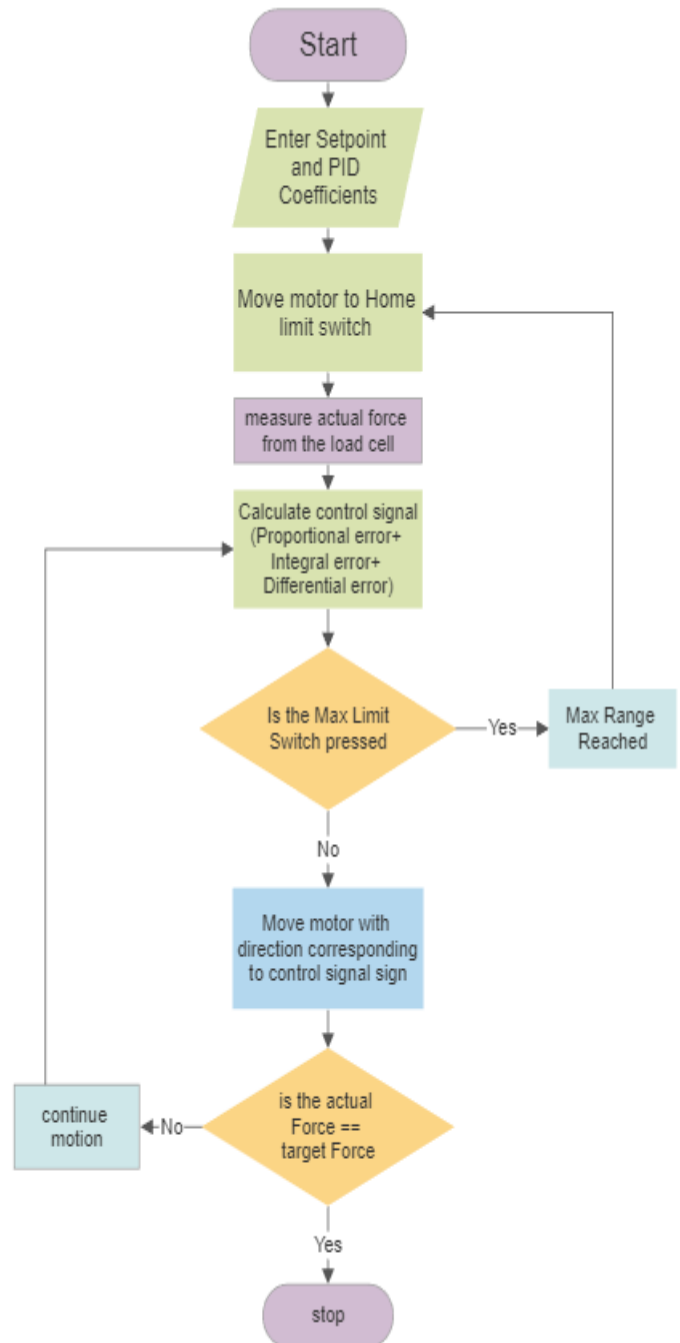


Figure 5: Force Control Flowchart

The following screenshot shows a comparison between the set point force and the actual force that comes from the load cell, these peaks show the transitions when someone tries to move the motor (exciting force into it), but the motor keeps as it is with zero force, no reaction force. Here the setpoint was Zero



Figure 6: Force Control Serial Plotter, with Set Point = Zero

The following screenshot shows the force readings (the blue is the actual force while the red is the setpoint which is zero in this example) the green line shows the position of the motion, it keep changing its position as I move the motor.



Figure 7: Force Control Serial Plotter, Force Readings vs Motor Position

**Code link:**

[https://drive.google.com/drive/folders/1QgNLvgTs7j9K mzNwtJqf21UMwl62XkhN?usp=share\\_link](https://drive.google.com/drive/folders/1QgNLvgTs7j9K mzNwtJqf21UMwl62XkhN?usp=share_link)

### 3.3 Position control

This code is for a motor control system that uses **position control**. Position control involves the use of sensors to monitor the position of the motor or other system, and algorithms to control the movement of the system based on this information.

The code uses an interrupt service routine to track the position of the motor using an encoder, and it uses a PID (proportional-integral-derivative) control algorithm to calculate the control signal (u) that will be applied to the motor. The PID algorithm uses the current error (the difference between the target position and the current position), the derivative of the error (the rate of change of the error), and the integral of the error (the accumulated error over time) to calculate the control signal.

The control signal is then used to set the direction and power of the motor, using the setMotor() function. The direction of the motor is set based on the sign of the control signal (positive for one direction, negative for the other), and the power of the motor is set to the absolute value of the control signal, with a maximum value of 150.

Overall, this code is implementing a position control system for a motor, using an encoder and a PID control algorithm.

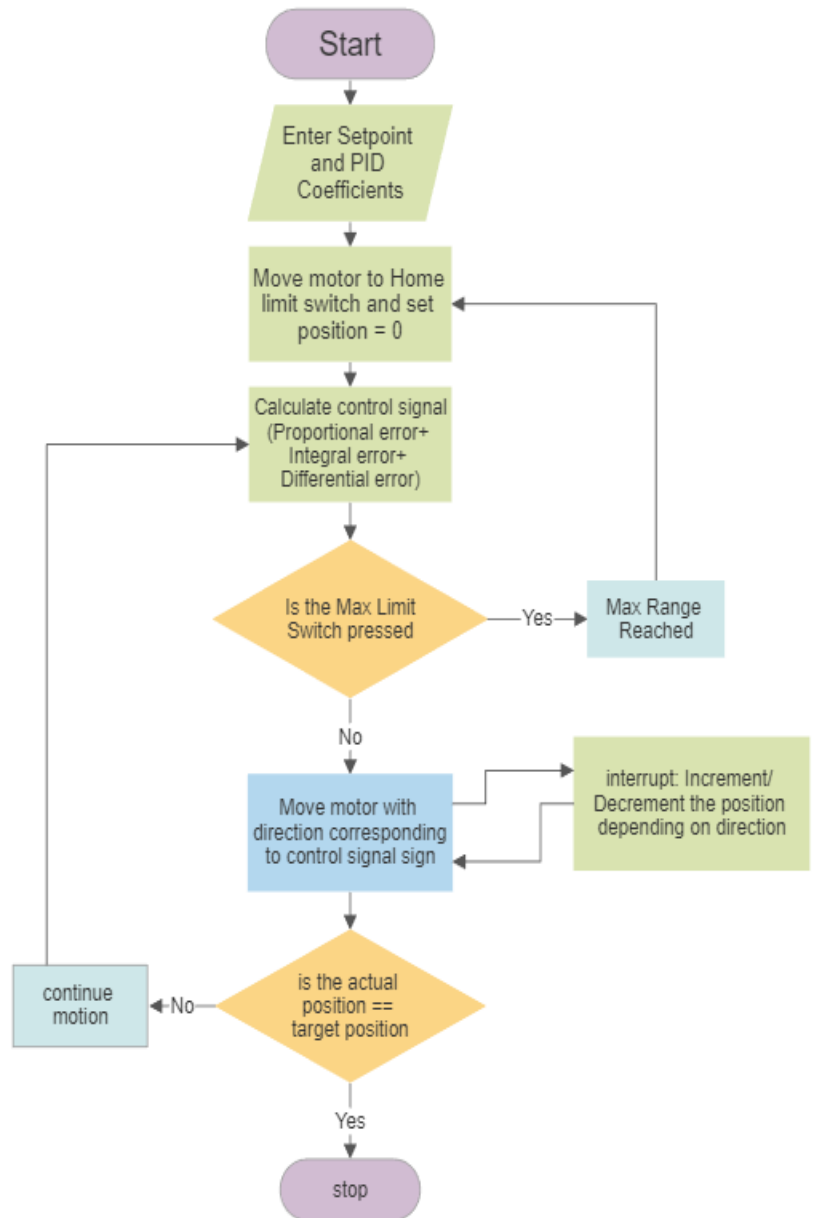


Figure 8: Position Control Flowchart

The following screenshot shows the actual position of the motor that is calculated from the pulses of the encoder versus the setpoint which is  $90^\circ$  in this example. We can see that the motor reaches the setpoint with very small overshooting and then it corrects its position again back to  $90^\circ$ . These graphs can be compared with the simulation output on the MATLAB.

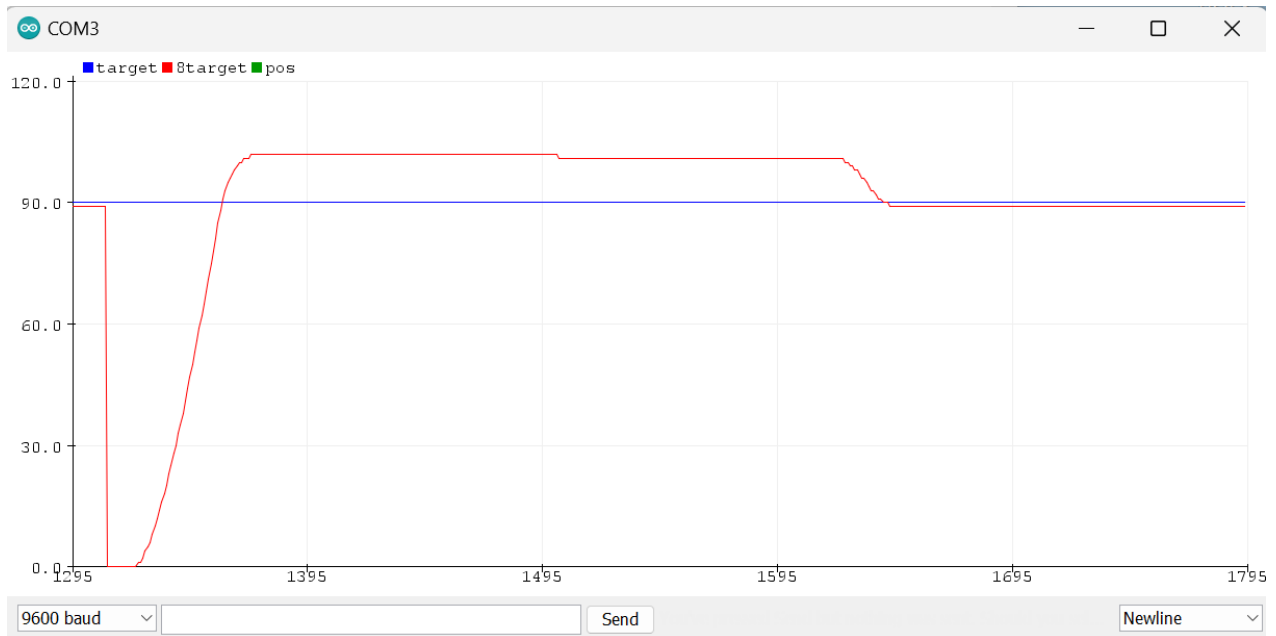


Figure 9: Position Control Serial Plotter, Set Point = 90 degrees

**Code link:**

[https://drive.google.com/drive/folders/1s1gBk3mnbozDsvb\\_hxbYiPxFMIZZDcnY?usp=share\\_link](https://drive.google.com/drive/folders/1s1gBk3mnbozDsvb_hxbYiPxFMIZZDcnY?usp=share_link)

### 3.4 Admittance control

Admittance control is a type of control strategy used in rehabilitation robotics to enable a robot to assist a human user in a natural and intuitive manner. In admittance control, the robot follows the motion of the human user by providing a force that is proportional to the user's motion. This allows the user to easily control the robot and perform tasks such as reaching and grasping objects.

One key feature of admittance control is that it is an impedance-based control method, which means that the robot's impedance (resistance to motion) can be adjusted in real-time based on the user's motion and the desired task. This allows the robot to adapt its motion and force output to the user's needs, and provides a more intuitive and natural interaction between the user and the robot

Admittance control has been used in a variety of applications, including robotic rehabilitation for stroke survivors, assistive robots for individuals with mobility impairments, and robot-assisted surgery. It is a promising control strategy for enhancing the usability and effectiveness of rehabilitation robotics systems.

The purpose of this code is to implement a force control loop for a motor, as well as an admittance control loop for a load cell.

The code begins by including a number of libraries and defining some constants for pin assignments. It then creates some objects and variables, including an object for a load cell and two objects for buttons.

In the setup() function, the code initializes the serial port and sets the mode for a number of pins. It also initializes the load cell and sets up an interrupt to read the encoder values.

In the loop() function, the code first checks the state of the second limit switch and homes the motor if the switch is pressed. It then updates the force value from the load cell and calculates the error in the outer loop "Force" control, as well as the derivative and integral of this error. The code then calculates the output of the first PID controller, which will be used to set the position of the motor.

Next, the code calculates the error in the inner loop "Position" control and the derivative and integral of this error. It then calculates the output of the second PID controller, which will be used

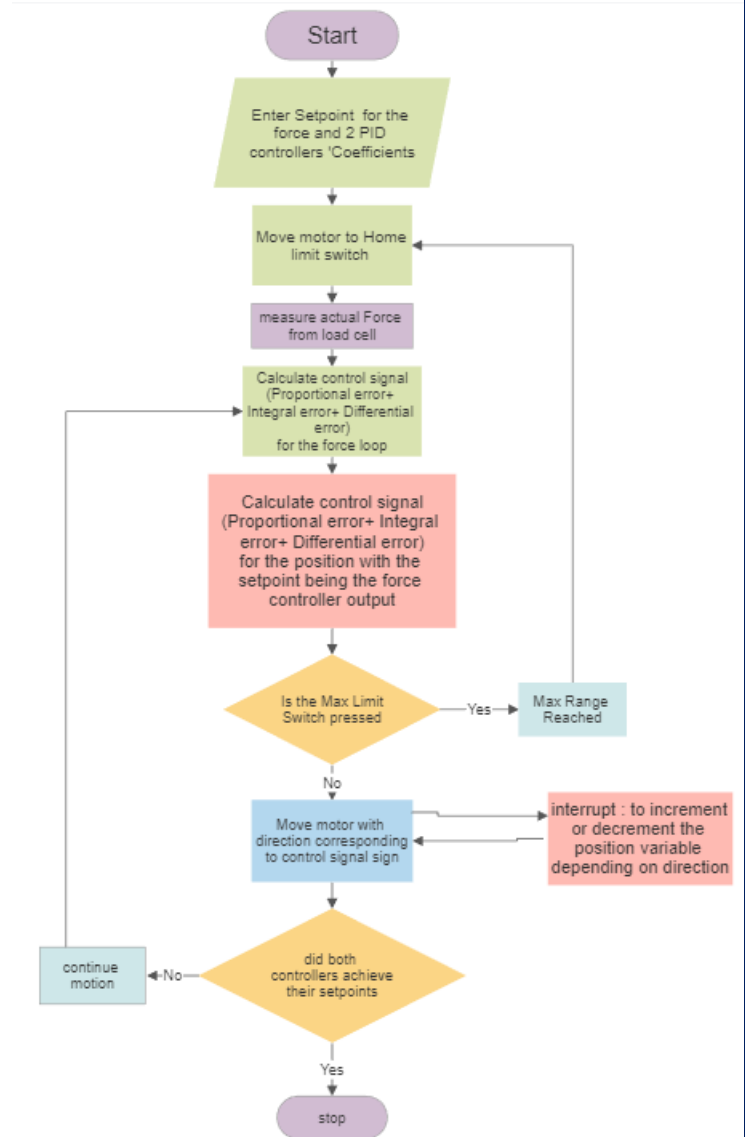


Figure 10: Admittance Control Flowchart

to set the PWM signal for the motor. The code also applies a low pass filter to the PWM signal before setting it as the output for the motor.

Finally, the code updates the previous error and integral values for both PID controllers and checks the state of the first limit switch. If the switch is pressed, the code sets the motor position to zero

**Code link:**

[https://drive.google.com/drive/folders/12UJG16gl6RYStosRtQDcJQfS7ayI6r-b?usp=share\\_link](https://drive.google.com/drive/folders/12UJG16gl6RYStosRtQDcJQfS7ayI6r-b?usp=share_link)

### 3.5 Impedance control

Impedance control is a type of control strategy used in robotics to achieve a desired interaction between a robot and its environment.

This code is for a robotic arm that is to be controlled using impedance control. The robotic arm has a load cell attached to it for measuring the force being applied to the arm. The arm is controlled by a motor and can rotate around a hinge joint. There is also a limit switch attached to the arm that is used to detect when the arm hits its maximum range of motion.

The code first includes some libraries and defines some constants and variables that are used later in the program. It also initializes some objects, such as the **limitSwitch** and **limitSwitch2** objects, which are used to detect when the limit switch is pressed.

In the **setup()** function, the program initializes the serial port and sets the pin modes for the motor control pins and the limit switch pin. It also initializes the load cell, and the limit switch objects.

The main loop of the program first checks if the maximum limit switch has been pressed. If it has, the arm is "homed" (moved to its starting position). Next, the program reads the force measurement from the load cell and calculates the error between the current position of the arm and the target position. It then uses this error to compute the output of a PID controller (a type of control algorithm) which is used to control the motor.

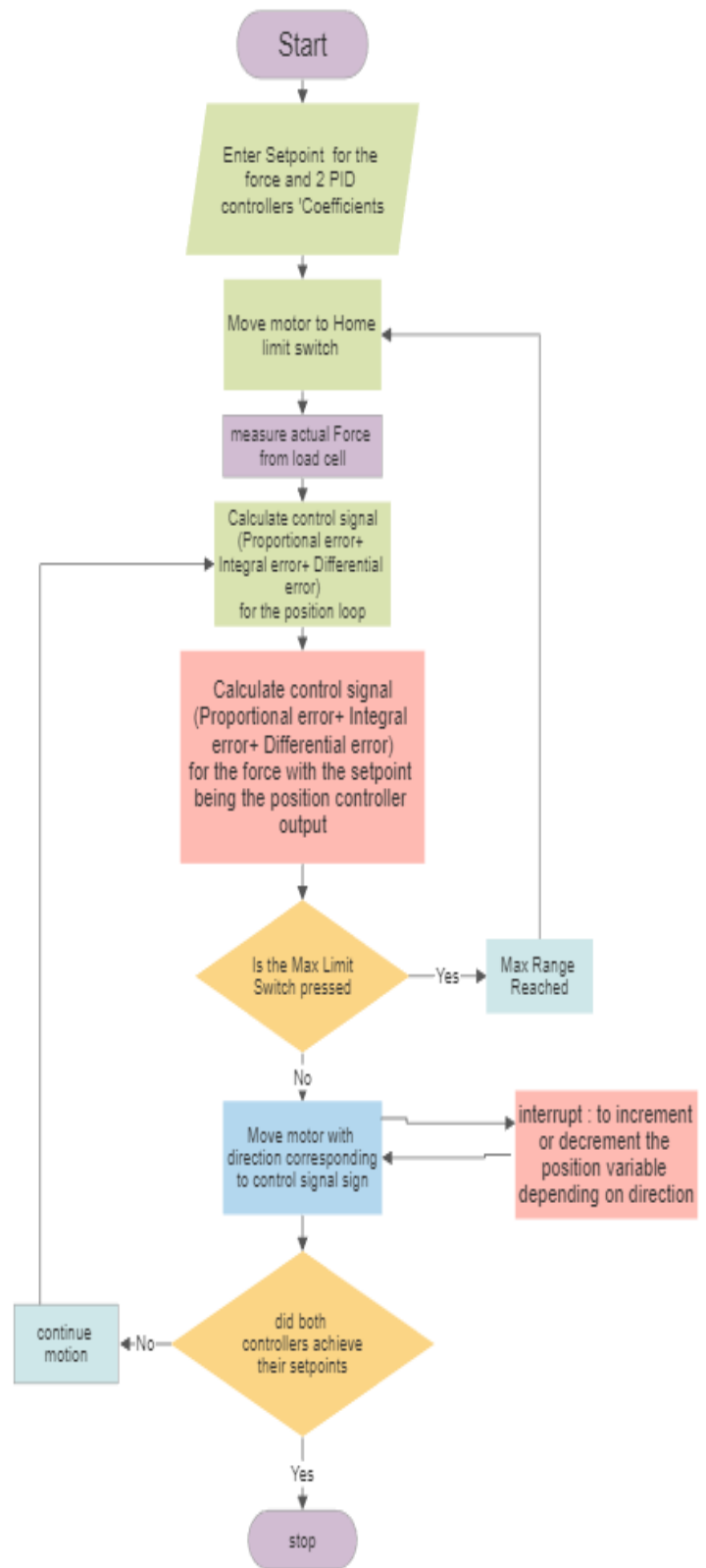


Figure 11: Impedance Control Flowchart

The code then checks if the arm is within a certain range of the target position. If it is, the error and integral values used in the PID controller are reset to zero. This is done to prevent the control output from getting too large and potentially causing the arm to move too quickly.

Finally, the code uses the output of the PID controller to set the setpoint on which the second controller of the force will operate and calculate another control signal. This signal will then be fed as pwm signal to control the speed of the motor. The arm will rotate in one direction if the error is positive and in the other direction if the error is negative. The code also checks if the arm has reached the limit of its range of motion using the **limitSwitch** object, and if it has, the motor is stopped to prevent the arm from colliding with the limit.

**Code link:**

<https://drive.google.com/drive/folders/1135BoMIXJObdmfTmvS-LxEqCAIdwZHtb>



## 4 Cad Overview

### 4.1 Home position

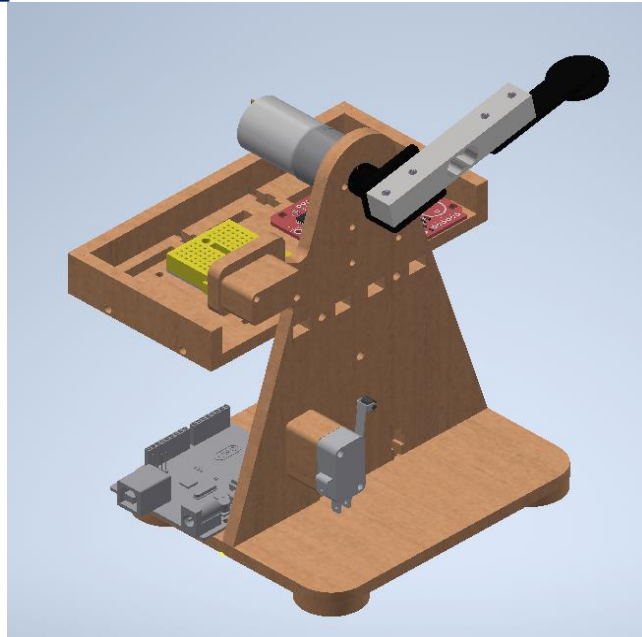


Figure 12: Home Position of the motor

### 4.2 Top view

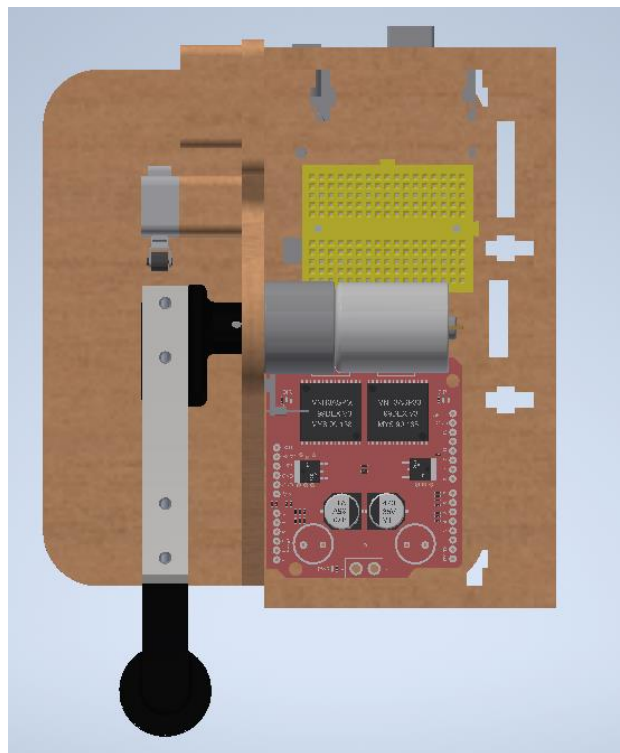


Figure 13: Top View of the CAD

### 4.3 Side view

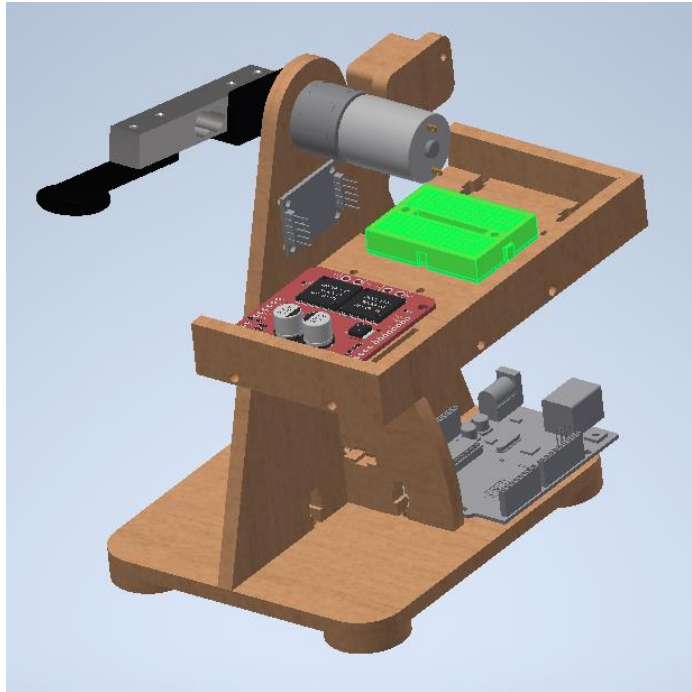


Figure 14: Side View of the CAD

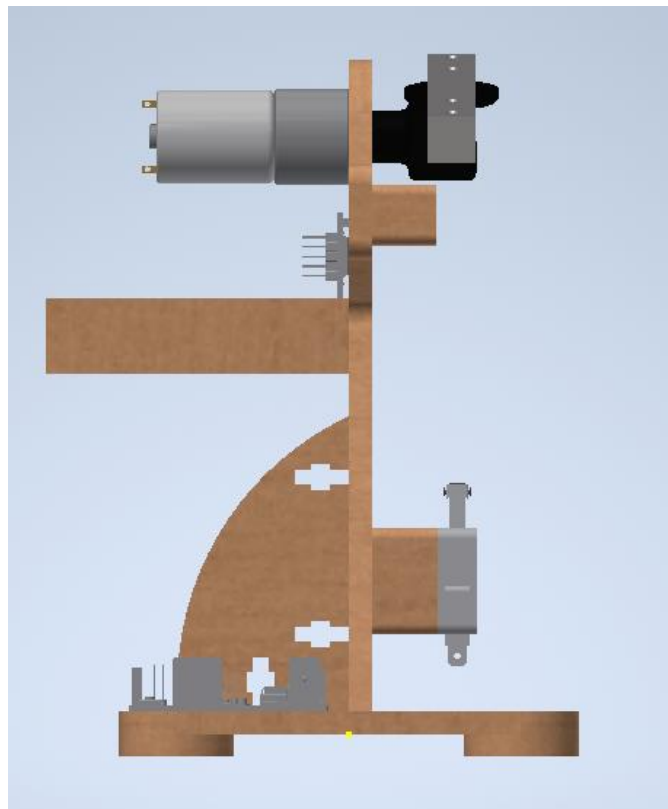


Figure 15: Side View of the CAD

## 4.4 Base view

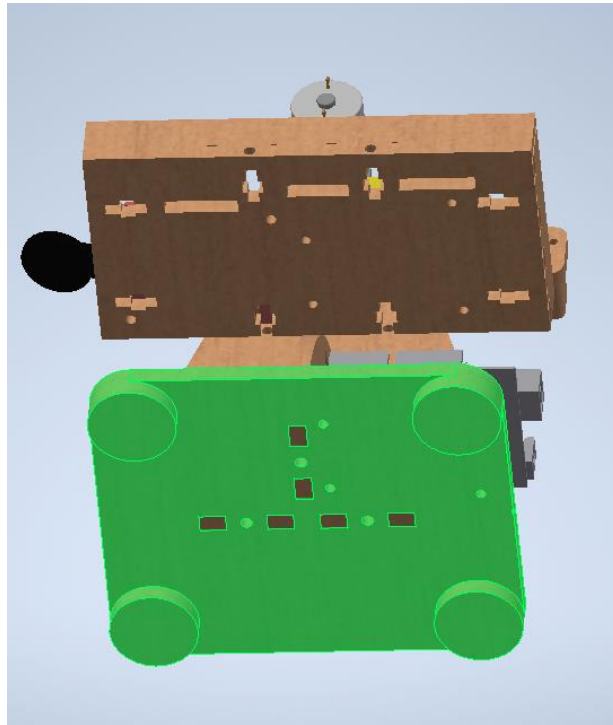


Figure 16: Base View of the CAD

## 4.5 Close up

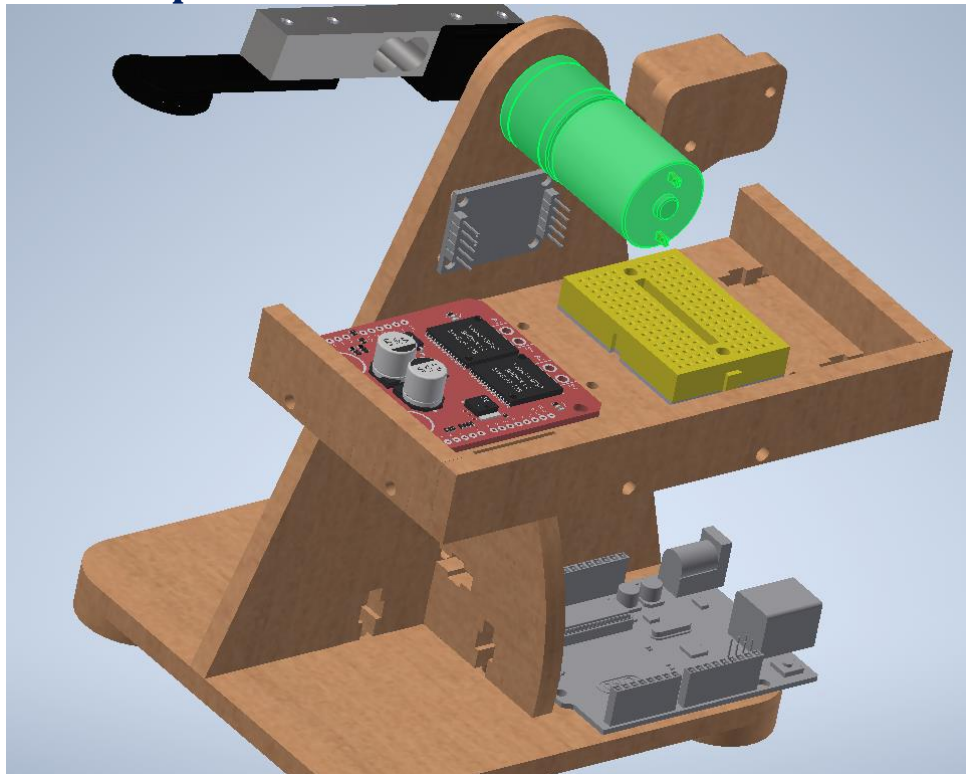


Figure 17: Full CAD showing all Sensor and the Motor

## 5 MATLAB Simulation:

Using MATLAB Simulink, we will model the DC motor, but we will assume the parameters of the motors to be close as possible to our DC motor that we have used in real environment. In each control algorithm we will tune the PID controller to have accurate and best results of our model and we will use the proper sensors using Simscape to provide us with actual force, torque or current, or position of our modeled DC Motor. Also, a step input with magnitude 1 will be given to our model and a comparison between the output and the input will be provided as well.

### 5.1 Current Control

The following snippet shows the model of the DC motor with closed-loop feedback from the current sensor with PID controller to control our plant (DC Motor). The current sensor here is added in the electrical model part of the DC motor that measures the actual current and compare it with the input and compute the error and then the PID computes the control signal by knowing the error.

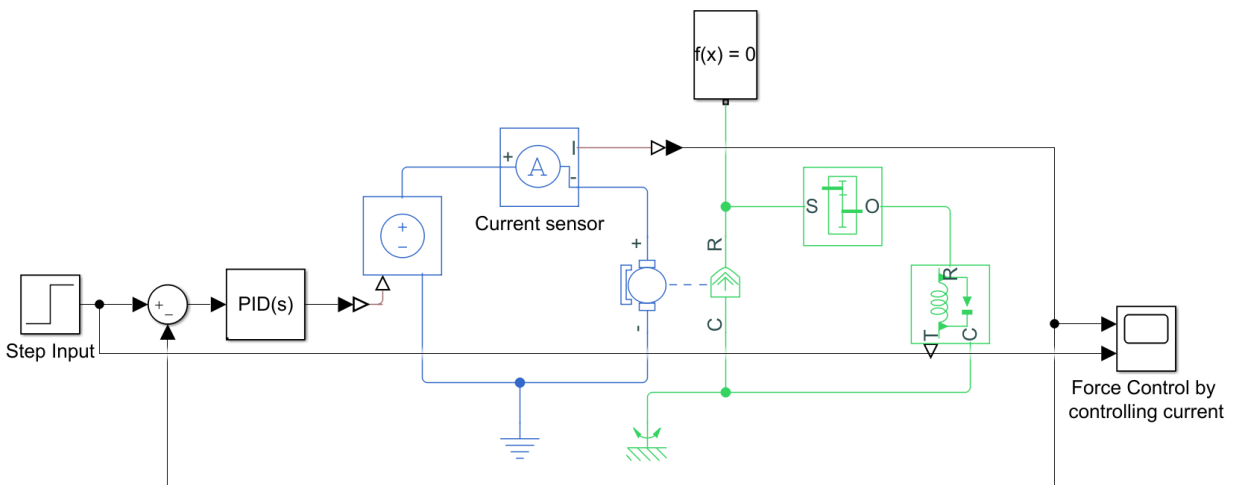


Figure 18: Current Control Block Diagram on MATLAB

The following graph shows the output of the indirect force control using readings from the current sensor and comparing it with the step input.

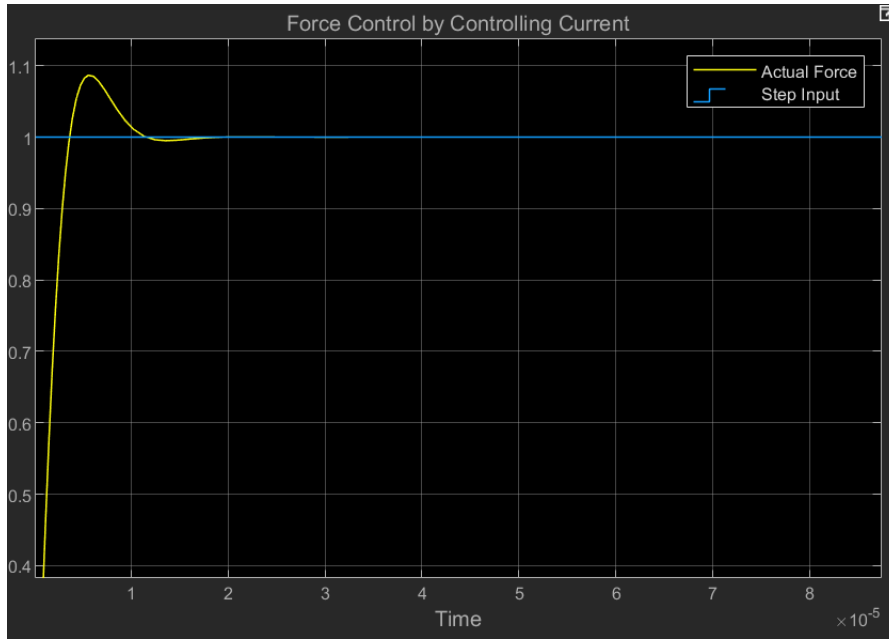


Figure 19: Current Control Outputs on MATLAB

**Model link:**

<https://drive.google.com/drive/folders/1fT0-jGSRswBMR1WUnSxVs6oHnNzGpiBI>

## 5.2 Force Control

The following snippet shows the model of the DC motor with closed-loop feedback from the torque sensor with PID controller to control our plant (DC Motor). The torque sensor here is added in the mechanical model part of the DC motor that measures the actual torque, compare it with the input, and compute the error and then the PID computes the control signal by knowing the error.

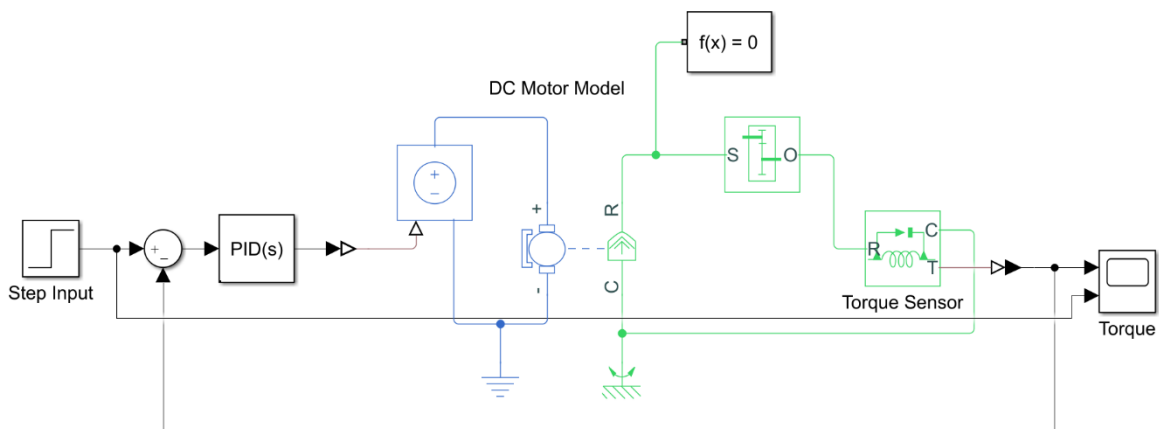


Figure 20: Force Control Block Diagram on MATLAB

The following graph shows the output of the torque control using readings from the torque sensor (Actual torque) and comparing it with the step input.

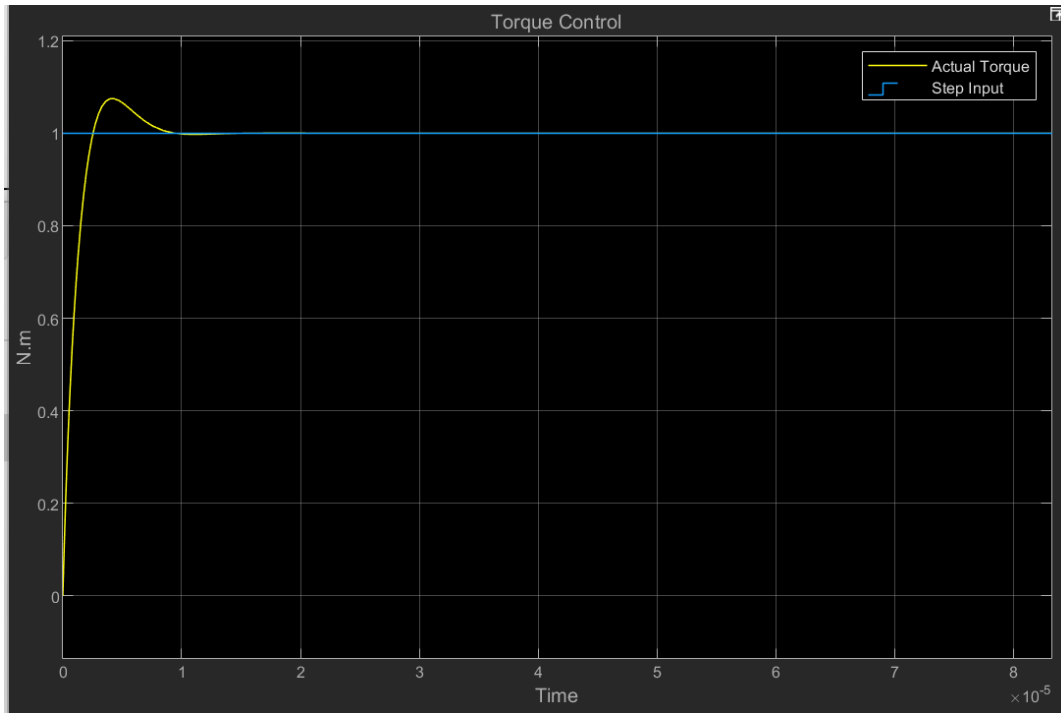


Figure 21: Force Control Output on MATLAB

**Model link:**

<https://drive.google.com/drive/folders/1D14yR99MVGQVIhbLkm532Z9xXvy-ps8W>

### 5.3 Position Control

The following snippet shows the model of the DC motor with closed-loop feedback from the rotational motion sensor with PID controller to control our plant (DC Motor). The rotational motion sensor here is added in the mechanical model part of the DC motor that measures the actual position  $\theta$  and compare it with the input and compute the error and then the PID computes the control signal by knowing the error.

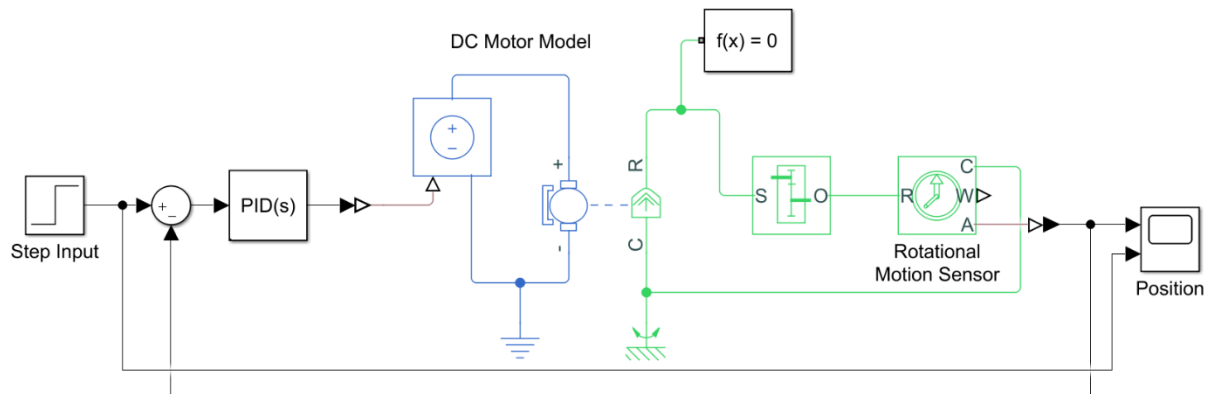


Figure 22: Position Control Block Diagram on MATLAB

The following graph shows the output of the position control using readings from the rotational motion sensor (Actual position) and comparing it with the step input.

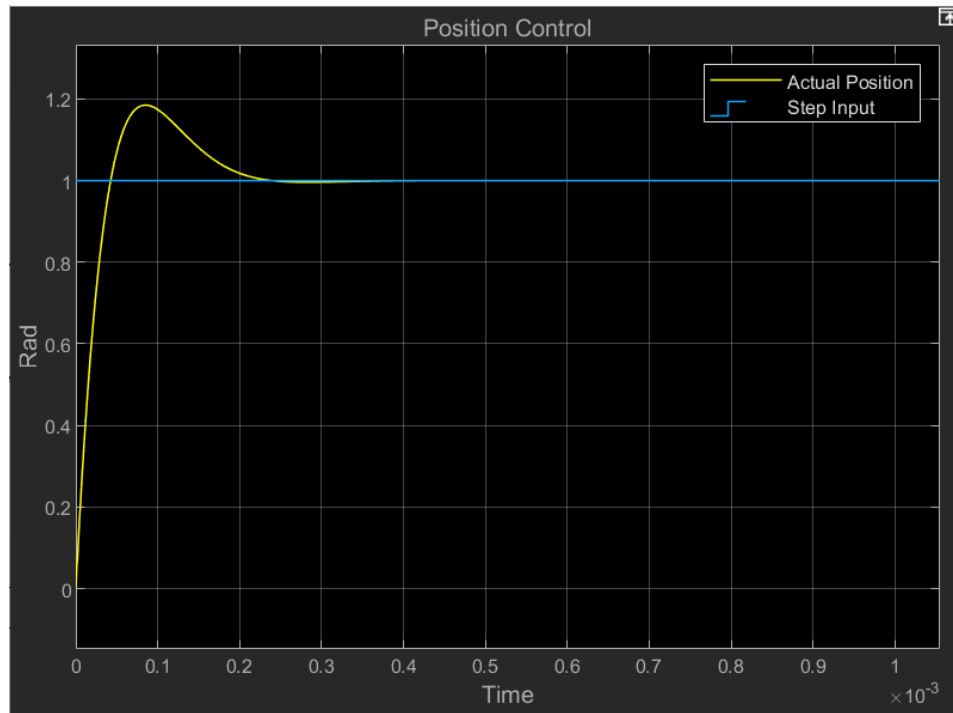


Figure 23: Position Control Output on MATLAB

**Model link:**

<https://drive.google.com/drive/folders/1R7dSVdLr7Z2QE0j9ILVfL6yKH5HUU4f>

## 5.4 Admittance Control

The following snippet shows the model of the DC motor with cascaded control loops. The outer loop is for controlling the torque (using torque sensor) while the inner loop is for controlling the position (using rotational motion sensor). Each sensor measures the actual data from the motor and compare it with the set point. The setpoint for the force here is out step input, while the setpoint for the position here is the output control signal of the PID controller for the force control. By tuning both controllers to get the best output we will have following outputs.

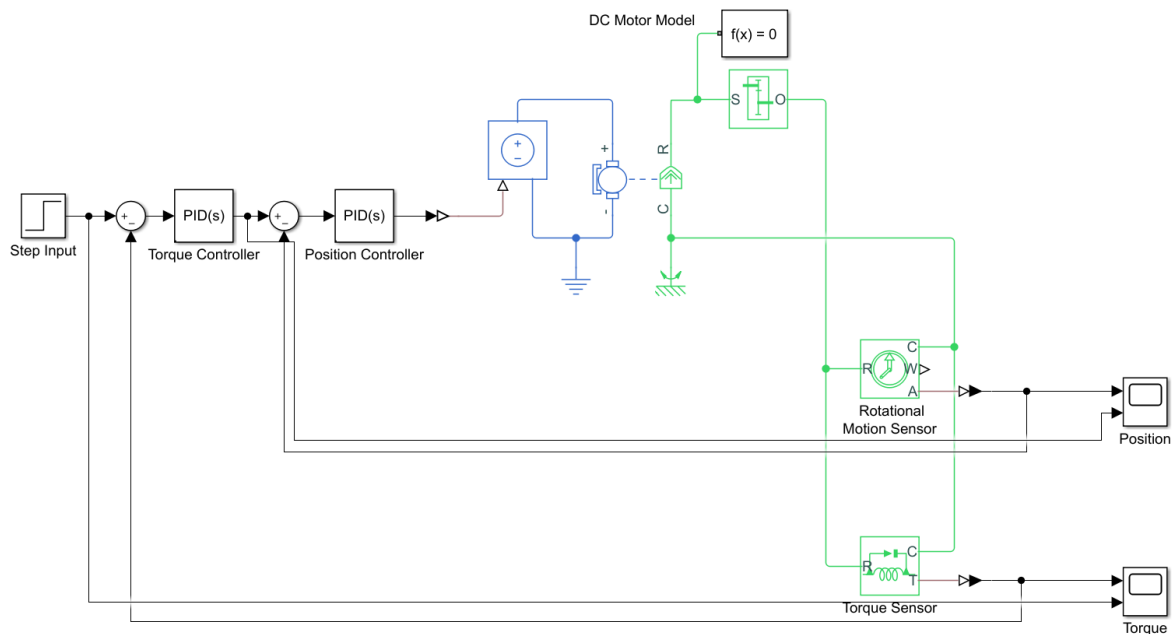


Figure 24: Admittance Control Block Diagram on MATLAB

The following graph shows the output of the force control using readings from the torque sensor and comparing it with the step input.

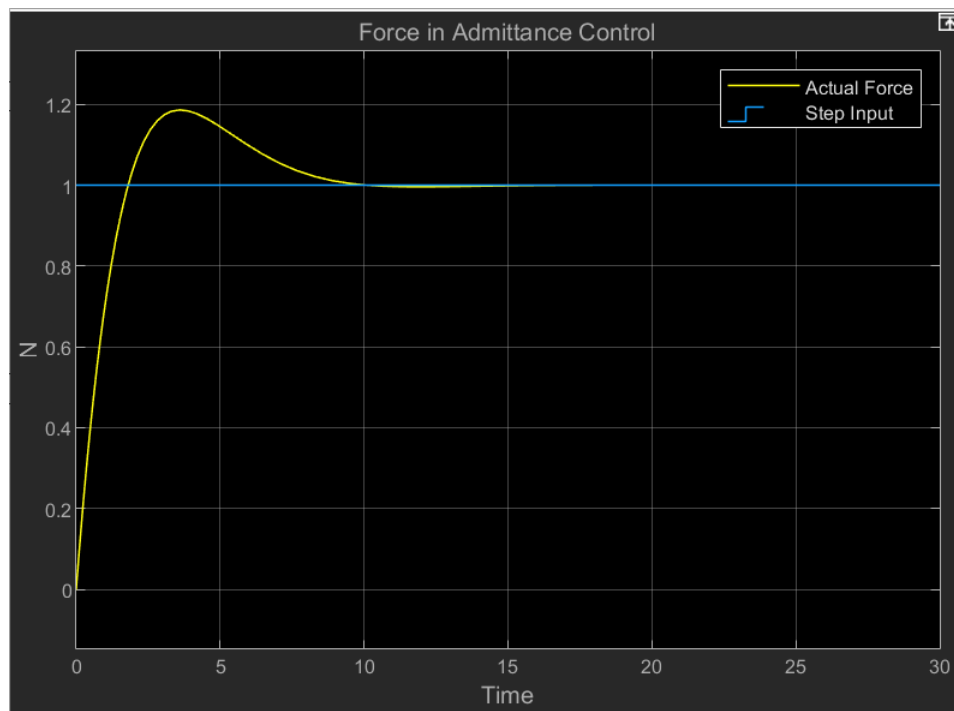


Figure 25: Force Output in Admittance Control

The following graph shows the output of the position control using readings from the rotational motion sensor and comparing it with the control signal that come from the force controller. Here



we should have converted the output from the force controller into position (converting F into degrees) by assuming a suitable stiffness. But we have tuned the Position controller to compensate this gain. (Here the initial position of the motor was 0 rad)

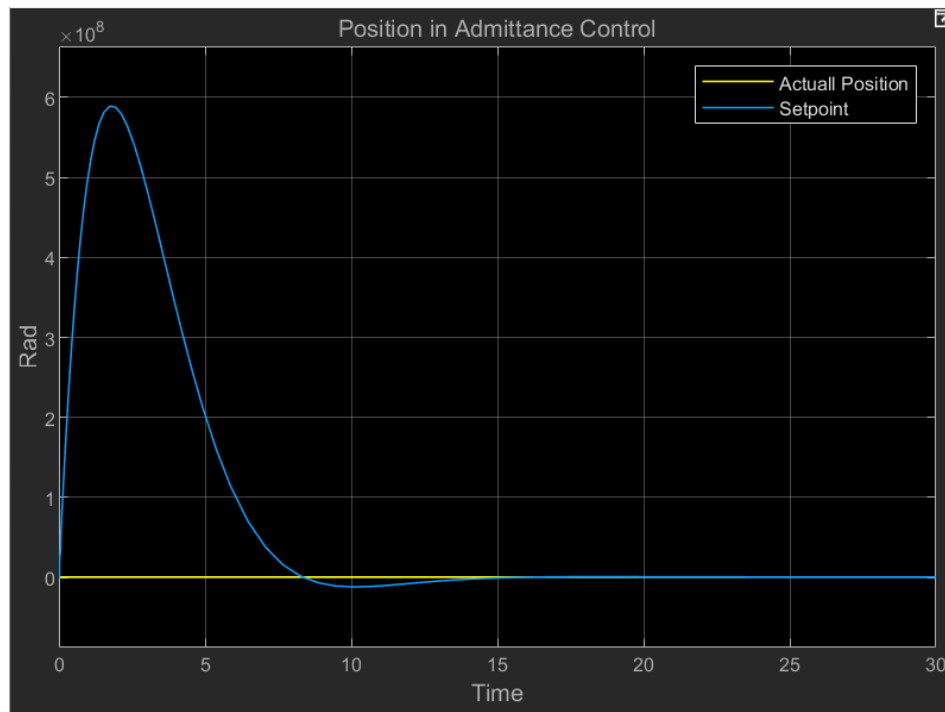


Figure 26: Position Output in Admittance Control

**Model link:**

[https://drive.google.com/drive/folders/1B5\\_OEIBmo2i2WgeB257hsnpfMIFZ7vUk](https://drive.google.com/drive/folders/1B5_OEIBmo2i2WgeB257hsnpfMIFZ7vUk)

## 5.5 Impedance Control

The following snippet shows the model of the DC motor with cascaded control loops. The outer loop is for controlling the position (using rotational motion sensor) while the inner loop is for controlling the torque (using torque sensor). Each sensor measures the actual data from the motor and compare it with the set point. The setpoint for the position here is out step input, while the setpoint for the force here is the output control signal of the PID controller for the position control. Here we should have converted the output from the position controller into force (converting degrees into F) by assuming a suitable stiffness. But we have tuned the force controller to compensate this gain.

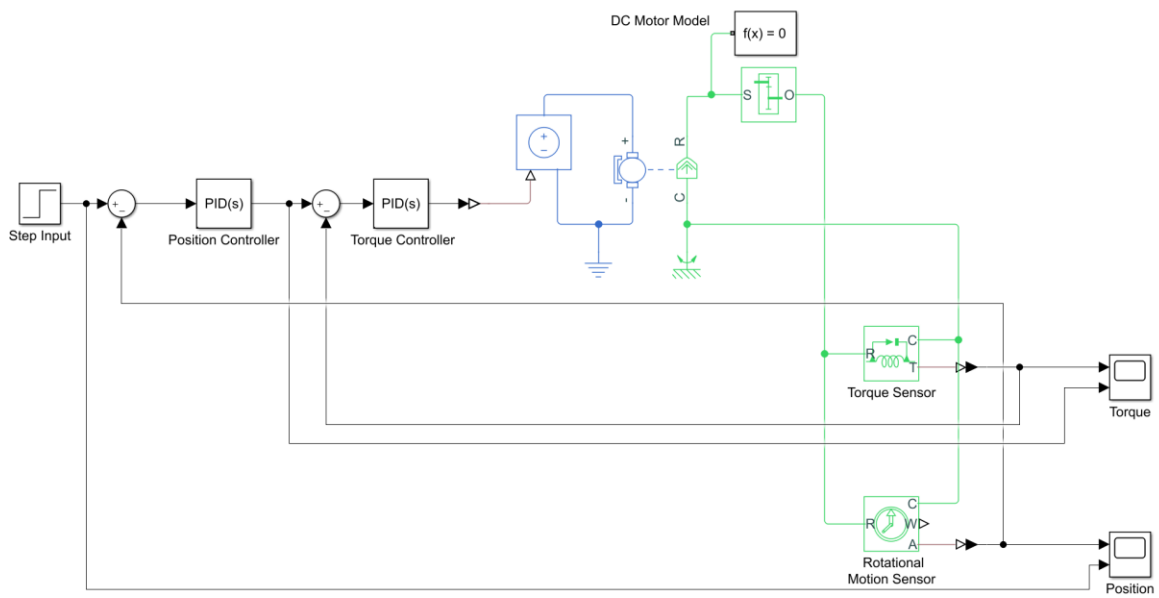


Figure 27: Impedance Control Block Diagram on MATLAB

The figure shows an example of the tuning process that we have used on MATLAB for all the controllers to determine suitable values of the  $K_p$ ,  $K_i$  and  $K_d$ .

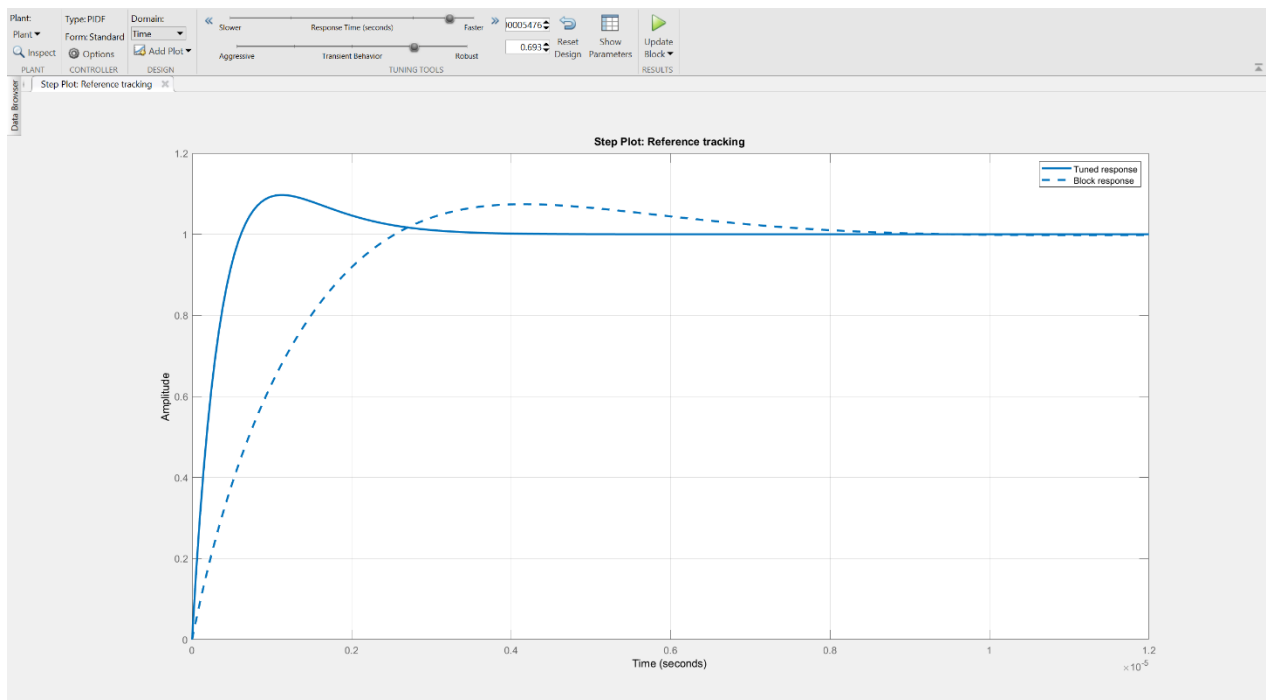


Figure 28: Tuning of PID Parameters on MATLAB

**Model link:**

<https://drive.google.com/drive/folders/1krUcaX6Jaqghle2IapWUaBcnPf9XNB2L>

## **6 Full project link / videos link:**

### **Full project link:**

<https://drive.google.com/drive/folders/1DsxaYB82BId5HImCO52jqJgS0bhjqRT>

### **Real implementation videos link:**

[https://drive.google.com/drive/folders/1JGJGT6nH1xi4kTSIY4ND3dweNMyAFLp1?usp=share\\_link](https://drive.google.com/drive/folders/1JGJGT6nH1xi4kTSIY4ND3dweNMyAFLp1?usp=share_link)