

**In the Name of God**



Project Title:

**Control of Ball Position on a Horizontal Rod**

Group Members:

**Alireza Padash**

**Mohammad Maleki Abyaneh**

**Mehdi Khayami**

Course Instructor:

**Dr. Ali Nahvi**

Faculty:

**Mechanical Engineering**

Academic Term:

**Second Half of 2023-2022**

## Table of Contents:

1. Introduction.....	4
2. Modeling, Theoretical Review, and Dynamic Analysis.....	5
3. Introduction of Components.....	8
3.1 AC Servo Motor (Delta Asda A2).....	8
3.2 Servo Motor Driver (ASD-A2-0721-M).....	9
3.3 STM32 Microcontroller Board (F407ZET6).....	10
3.4 Planetary Gearbox (C002F0420MT10).....	10
3.5 Ultrasonic Distance Sensor (SRF05).....	11
3.6 Encoder.....	12
3.7Interface Board.....	12
3.8 Power Supply and Fuse.....	13
3.9 Voltage Regulator (Lm2596) .....	14
3.10 ST-LINK/V2 Programmer.....	14
3.11 Magnetic Limit Switches.....	15
3.12 Final Assembly of the System.....	15
4. System Specifications.....	16
5. STM32 Pin Descriptions.....	16
5.1 Important Pin Functions (TRIG, ECHO, Pulse, Direction, USART) .....	16
5.2 General Configuration of Pins.....	18
6. STM32 Code Logic.....	19
6.1 Library Initialization.....	19
6.2 Sensor and Motor Control Setup.....	19
6.3 PID Controller Setup.....	20
6.4 Loop Structure and Sensor Feedback.....	20
6.5 While Loop and Motor Control.....	20
6.6Ultrasonic Sensor Data Reading.....	21
7. MATLAB Code.....	22

8. Graphs and Analysis.....	24
9. PID Explanation.....	31
10. Bonus Section.....	33
11. Task Assignments .....	33
12. References.....	34

## 1. Introduction:

One of the most fascinating and relatively simple examples of mechanical engineering and control systems is the ball and rod system. The overall function of this system is to control and maintain a ball on a horizontal rod, which is connected to a motor and specific supports. The system should be able to keep the ball in position, even if an external disturbance affects the ball.

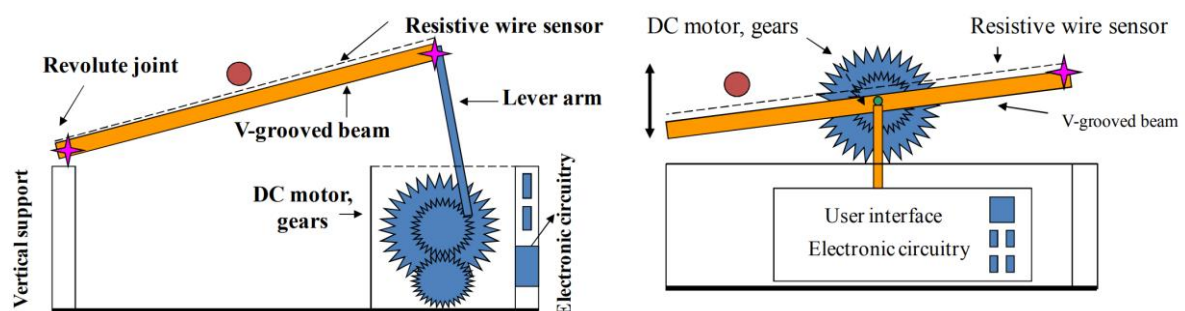
The position of the ball is determined by sensors, which will be explained in more detail later. Using these sensors, the motor generates the necessary torque to rotate the rod and maintain the ball's balance. Any potential errors detected by the sensors are sent to the controller, which then sends appropriate commands to the motor.

## 2. Modeling, Theoretical Review, and Dynamic Analysis:

The ball and rod system can be examined in two main configurations:

1. The motor is positioned at the center of the rod.
2. The motor is located at the end of the rod and connected by a lever.

A schematic of these two types is shown in Figure 1.



A schematic of these two types is shown in Figure 1.

Our project is based on the right-side configuration, where the motor is located at the center of the rod. This system can be solved using both Newtonian and Lagrangian methods (related to energy). For the Newtonian method, we first draw a free-body diagram for the system as shown in Figure 2.

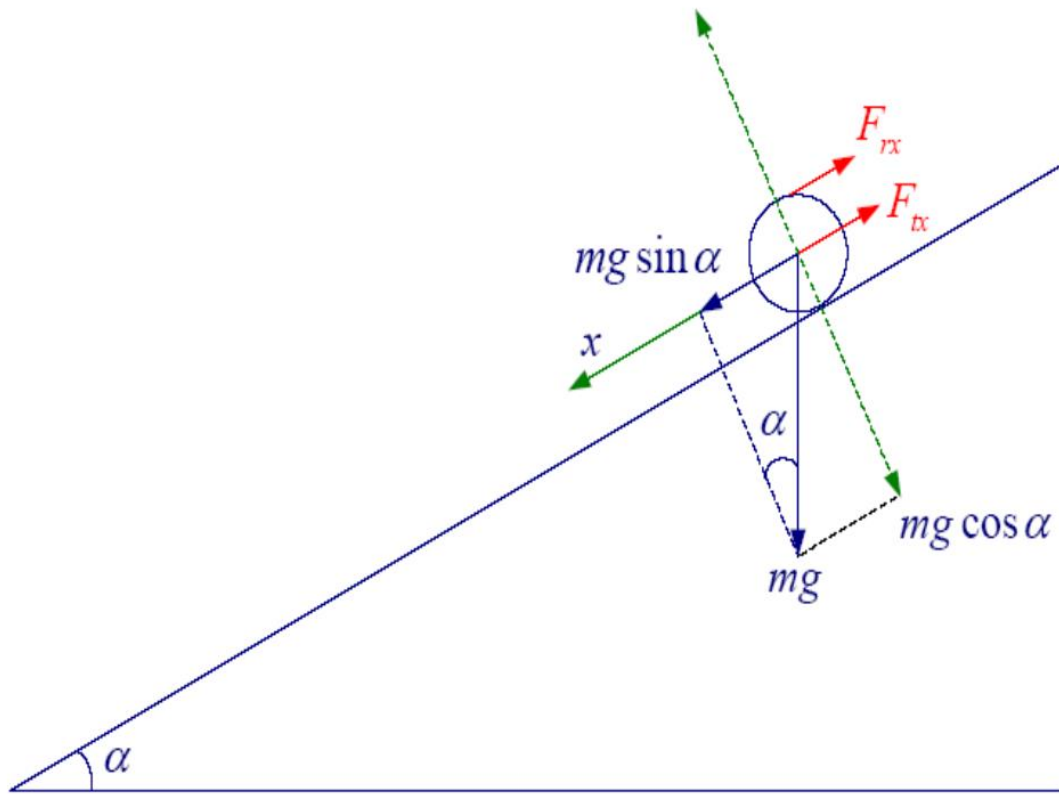


Figure 2 - Free-body diagram of the

The forces created by translational motion along the rod and the acceleration generated in this direction are represented by  $F_{tx}$ , and the forces created by rotational motion are represented by  $F_{rx}$ . This rotational force itself creates a twisting torque around the ball, which is calculated by multiplying  $F_{rx}$  by the radius of the ball. Additionally, we know that the sum of the torques is equal to the moment of inertia of the object times the angular acceleration.

The result of all these forces and torques can be expressed by the following equations:

$$F_{tx} = m\ddot{x}$$

$$T_r = F_{rx}R = J\dot{\omega}$$

$$\omega = \frac{v}{R} \rightarrow T_r = \frac{J}{R}\ddot{x} \rightarrow F_{rx} = \frac{J}{R^2}\ddot{x}$$

In these equations:

- $m$  is the mass of the ball,
- $J$  is the moment of inertia of the ball,
- $x$  is the position of the ball on the rod,
- $\text{TrTr}$  is the rotational torque of the ball,
- $R$  is the radius of the ball.

Furthermore, the sum of the forces can be expressed as:

$$F_{rx} + F_{tx} = mg \sin \alpha$$

Next, using the energy method, we know that the total energy of the system is the sum of the energy of the rod and the ball:

$$T = T_{beam} + T_{cylinder}$$

$$T_{beam} = \frac{1}{2} I_b \dot{\theta}_b^2$$

For the ball, which is modeled as a cylinder, the kinetic energy is given by:

$$T_{cylinder} = \frac{1}{2} m_c \dot{r}_c^2 + \frac{1}{2} I_c \dot{\theta}_c^2$$

$$I_c = \frac{1}{2} m_c R^2$$

where  $r_c$  is the position of the ball on the rod and  $R$  is the radius of the circular cross-section of the cylindrical ball.

$$\dot{r}_c = R \dot{\theta}_c$$

Therefore, the total kinetic energy of the system can be expressed as follows:

$$\begin{aligned} T &= \frac{1}{2} I_b \dot{\theta}_b^2 + \frac{1}{2} m_c R^2 \dot{\theta}_c^2 + \frac{1}{4} m_c R^2 \dot{\theta}_c^2 = \frac{1}{2} I_b \dot{\theta}_b^2 + \frac{3}{4} m_c R^2 \dot{\theta}_c^2 \\ &= \frac{1}{2} I_b \dot{\theta}_b^2 + \frac{3}{4} m_c \dot{r}_c^2 \end{aligned}$$

Now, provided that the angle the rod makes with the horizontal is  $\theta_b$ , we will also have for the potential energy of the system:

$$U = m_c g \cdot (r_c \sin \theta_b)$$

It should be noted that the reference point for potential energy is the horizontal line passing through the center of mass of the rod, or the point of attachment of the motor, and for this reason, the potential energy of the rod remains unchanged, and the equation only includes the term related to the ball.

Now, we will derive the expression for the Lagrangian, which is the result of the difference between kinetic and potential energy:

$$L = \frac{1}{2} I_b \dot{\theta}_b^2 + \frac{3}{4} m_c \dot{r}_c^2 - m_c g \cdot (r_c \sin \theta_b)$$

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q)$$

$$\frac{d}{dt} \left[ \frac{\partial L}{\partial \dot{q}_i} \right] - \frac{\partial L}{\partial q_i} = F_{qi}$$

*Now, in this system, we have two types of motion and consequently two generalized coordinates, which are  $\theta_b$  and  $r_c$ . Instead of  $F_{qi}$ , we will have the torque generated by the motor, which we denote as  $\tau$ :*

$$\frac{\partial L}{\partial \theta_b} = -m_c g r_c \cos \theta_b$$

$$\frac{\partial L}{\partial r_c} = \frac{3}{2} m_c \dot{r}_c$$

$$\frac{\partial L}{\partial \dot{\theta}_b} = I_b \dot{\theta}_b \rightarrow \frac{d}{dt} \left[ \frac{\partial L}{\partial \dot{\theta}_b} \right] = I_b \ddot{\theta}_b$$

$$\frac{\partial L}{\partial \dot{r}_c} = \frac{3}{2} m_c \dot{r}_c \rightarrow \frac{d}{dt} \left[ \frac{\partial L}{\partial \dot{r}_c} \right] = \frac{3}{2} m_c \ddot{r}_c$$

On the other hand, the equations with respect to  $\theta_b$  correspond to the torque of the system, and with respect to  $r_c$  relate to the resultant force applied, which will result in:

$$I_b \ddot{\theta}_b + m_c g r_c \cos \theta_b = \tau$$

$$\frac{3}{2} m_c \ddot{r}_c - \frac{3}{2} m_c \dot{r}_c = 0$$

Now, we can find the transfer function using one of the two methods discussed above. For example, based on the derived equation, we can take the Laplace transform, and if we set the initial conditions to zero, we will have:

$$\frac{J}{R^2} s^2 X(s) + m s^2 X(s) = m g \cdot \alpha(s)$$

$$G(s) = \frac{X(s)}{\alpha(s)} = \frac{m g}{m s^2 + \frac{J}{R^2} s^2}$$

For this system, we will have  $g=9.81 \frac{m}{s^2}$ ,  $m=50.7$  g,  $R=2$  cm, and  $J=0.5$  mR<sup>2</sup>, which results in the value of the transfer function being equal to:

$$J = 1.014 * 10^{-5} [kg.m^2]$$

$$G(s) = \frac{6.54}{s^2} \left[ \frac{m}{rad} \right]$$

It should be noted that the angle of the rod is considered as the input, and the position of the ball is considered as the output.

### 3. Introduction of Components:

#### 3.1 AC Servo Motor, Model Delta Asda A2:

- This model of servo motor is used as the system's driving force.
- Inputs: VAC110 and 5.1A, Output: 0.75KW
- Maximum Output Speed: 3000rpm, Maximum Torque: 2.39N.m





Figure 3 - AC Servo Motor, Model Delta Asda A2

### 3.2 Servo Motor Driver, ASD-A2-0721-M:

- The STM32 board sends commands to the driver, which adjusts the motor's position and applies the correct voltage.
- Input: 200-230V, Output: 110V, 5.1A



Figure 4 - Servo Motor Driver, ASD-A2-0721-M

### 3.3 STM32 Board, Model F407ZET6:

- This board controls the system and sends signals to the motor driver to adjust the motor's position.

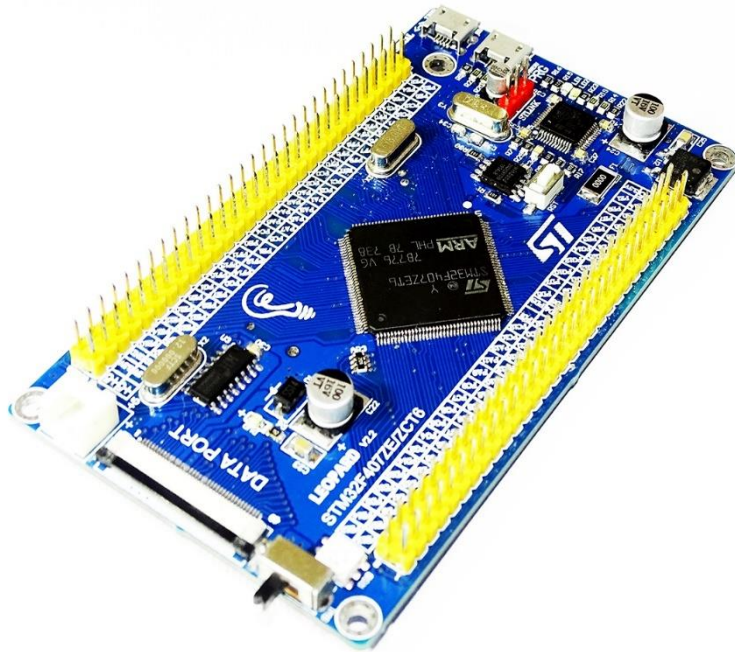


Figure 5 - STM32 Board, Model F407ZET6

### 3.4 Planetary Gearbox, Model C002F0420MT10:

- This planetary gearbox is connected to the motor and has a gear ratio of 41.8:1, meaning the motor must rotate 41.8 times for the rod to rotate once.



Figure 6 - Planetary Gearbox, Model C002F0420MT10

### 3.5 Ultrasonic Distance Sensor Model SRF05:

The SRF05 ultrasonic distance sensor module is used for measuring distances and is capable of measuring distances from 2 centimeters to 450 centimeters with an accuracy of 3 millimeters. Other specifications are as follows: Operating voltage: 4.5 to 5.5 volts DC, Current: 10 to 40 milliamps, Operating temperature: 0 to 70 degrees Celsius.

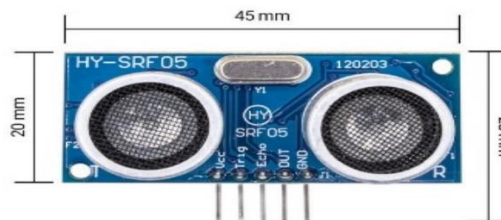


Figure 7 – Ultrasonic Distance Sensor Model SRF05

### 3.6 Encoder:

In general, an encoder is a device or process that converts data from one format to another. In a position sensor, this device can identify and convert mechanical movement into an analog or digital coded output signal and is located at the back of the motor attached to it. (Specifications: Encoder type: Incremental, 17 bits, and pulses per revolution: ppr160000)



Figure 8 – Encoder

### 3.7 Interface Board:

The CN1 socket conversion board for the motor driver is used to connect the control wires to the driver.



Figure 9 – Interface Board

### 3.8 Power Supply with Delta Brand and Fuse:

The power supply used is 24 volts and 2.5 amps with the Delta brand. Additionally, a 16-amp fuse is placed alongside it to protect the device against unauthorized electrical currents.



Figure 10 – Power Supply with Delta Brand and 16-Amp Fuse

### 3.9 Module with Model LM2596:

In this device, we have used a module that is connected to the 24-volt power supply. Its function is to take 24 volts from the power supply and convert it to 5 volts (it is a voltage step-down module). This 5 volts is then transferred to the STM32 board and the ultrasonic sensors.



Figure 11 – Module with Model LM2596

### 3.10 ST-LINK/V2/01-0 Programmer:

The programmer connects to one side to the laptop and the other side to the STM32, transferring information from the laptop to the STM32.



Figure 12 – ST-LINK/V2/01-0 Programmer



### 3.11 Magnetic Limit Switch:

The magnetic limit switch is used to prevent excessive rotation of the arm and damage to the system. This limit switch is installed on both sides of the arm and does not allow the arm to rotate beyond positive and negative 45 degrees.



Figure 13 – Magnetic Limit Switch

### 3.12 Final Assembled Device:

After connecting the components that were previously described, the final device, as shown below, is assembled.

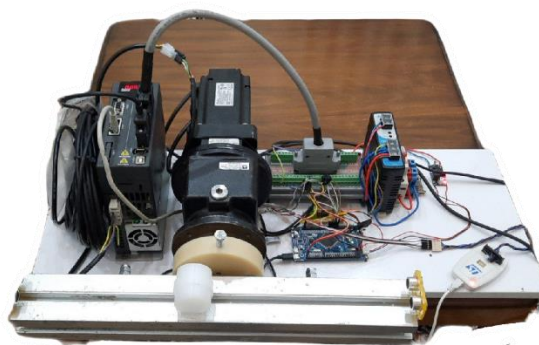


Figure 14 – Final Assembled Device

#### 4. Other Specifications of the Assembled System Components:

Other specifications include the mass of the cylindrical ball used, which is 50.7 grams, and the diameter of the ball is 4 centimeters; additionally, the length of the rod is 45 centimeters.

#### 5. Explanation of Several STM32 Pins:

In the figure below, you can see an overview of the STM32 IC, where we have identified the input and output pins (it has a total of 144 pins). We will later elaborate on its important pins.



Figure 15 – Overview of the STM32 IC

##### 5.1 Explanation of Important Pins:

As shown in the overview of the STM32, there are several yellow pins referred to as power pins, which are used to keep the STM32 powered on. Additionally, these pins cannot be defined as input or output.



In the figure, there are also green pins, which are defined as input and output pins.

- As shown in the figure below, the trig and echo pins are indicated, which are for the ultrasonic sensor. Symbol 1 represents Sensor 1, and symbol 2 represents Sensor 2. (The ultrasonic sensor receives a command from trig and sends the result through echo, allowing it to measure distance.)

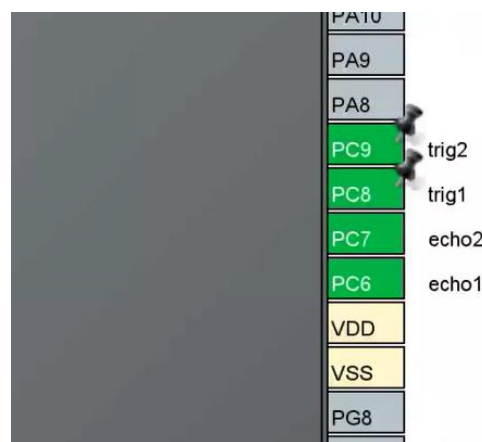


Figure 16 – Trig and Echo Pins

- As shown in the figure below, the pulse and Dir pins are indicated, to which the motor is connected. (Direction is used to change the motor's direction, and Pulse is the signal that controls distance and speed.)

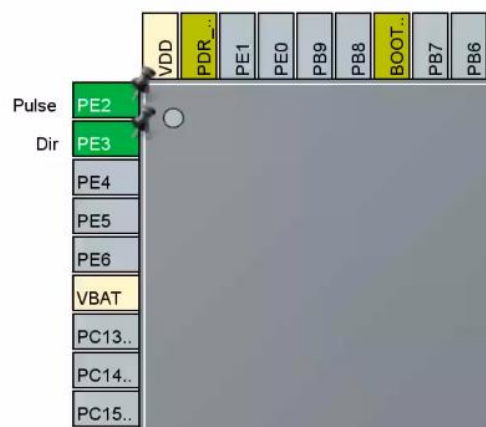


Figure 17 – Pulse and Direction Pins

- As shown in the figure below, the USART2\_TX and USART2\_RX pins are indicated, which are used to send data to the computer.

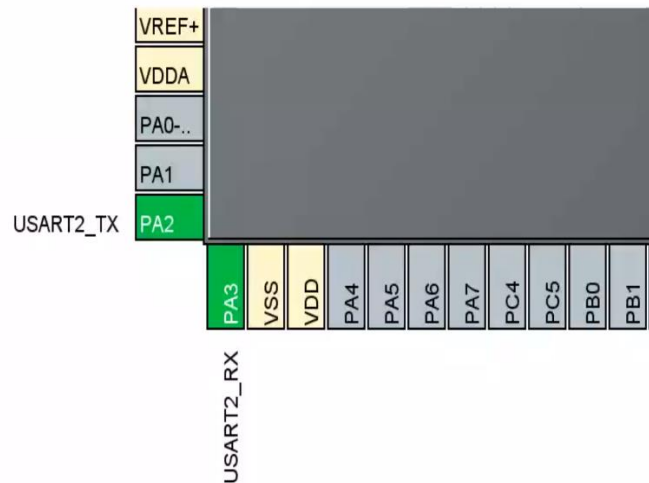


Figure 18 – USART2\_TX and USART2\_RX Pins

## 5.2 Brief Explanation of General Settings:

As shown in the figure below, settings such as data transmission speed, type, and more can be configured from this menu.

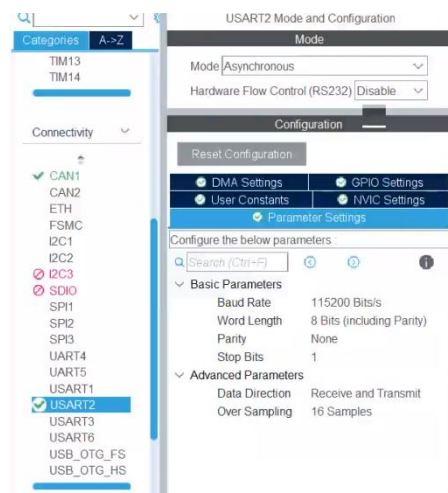


Figure 19 – General Settings

## 6. STM32 Code and Its Logic:

To analyze the code, we will divide it into different sections and then provide explanations regarding each section.

### 6.1

```
22=/* Private includes -----*/
23 /* USER CODE BEGIN Includes */
24 #include <string.h>
25 #include <stdlib.h>
26 #include <stdio.h>
27 #include "pid.h"
28 /* USER CODE END Includes */
```

Figure 20

In this section, we have defined libraries; for example, lines 24, 25, and 26 are for converting text to numbers and vice versa. Line 27 includes the PID library, which takes parameters such as the k values and position, and then outputs the required value.

### 6.2

```
53 /* USER CODE BEGIN PV */
54 PID_TypeDef TPID;
55 double p = 150; //kp
56 double i = 40; //ki
57 double d = 100; //kd
58 double PIDOut = 0;
59 double setposition = 0;
60 double actualDistancedouble = 0;
```

Figure 21

In lines 55, 56, and 57, we defined the initial values of the K parameters as type double. Line 58 is the PID output, line 59 is the target position we want to reach, and line 60 is the current position of the ball.

## 6.3

```
104=void delay (uint16_t time)
105 {
106     HAL_TIM_SET_COUNTER(&htim8, 0);
107     while (__HAL_TIM_GET_COUNTER (&htim8) < time);
108 }
109
110=void SR05_Read1 (void){
111     HAL_GPIO_WritePin(trig1_GPIO_Port, trig1_Pin, GPIO_PIN_SET); // pull the TRIG pin HIGH
112     delay(10);
113     HAL_GPIO_WritePin(trig1_GPIO_Port, trig1_Pin, GPIO_PIN_RESET); // pull the TRIG pin low
114     __HAL_TIM_ENABLE_IT(&htim8, TIM_IT_CC1);
115 }
```

Figure 22

This code is for configuring the ultrasonic sensor, and GPIO refers to the pins we mentioned earlier. In this code, it is first turned on with the 'set' command, followed by a delay of 10 milliseconds, and then it is turned off. This instructs the ultrasonic sensor to send data.

## 6.4

```
171 PID(&TPID, &actualDistancedouble, &PIDOut, &setposition, p, i, d, _PID_P_ON_E, _PID_CD_DIRECT);
172 PID_SetMode(&TPID, _PID_MODE_AUTOMATIC);
173 PID_SetSampleTime(&TPID, 10);
174 PID_SetOutputLimits(&TPID, -3000, 3000);
```

Figure 23

Line 171 defines the PID library (in which the variables P, I, and D, actualDistance, PIDOut, and setposition are defined, and the PID variables are stored in PIDOut). Line 174 indicates the minimum and maximum angle. It is noteworthy that since we could not input decimal numbers, we multiplied by 100, meaning that the angle range can be  $\pm 30$  degrees (the last two digits are considered as decimals). When the information is sent to the motor, it is divided by 100.

## 6.5

```
176 /* USER CODE END 2 */
177
178 /* Infinite loop */
179 /* USER CODE BEGIN WHILE */
180 while (1)
181 {
182     SR05_Read1();
183     //SR05_Read2();
184     Distance1 = Distance1;
185     Distance2 = Distance2;
186     actualDistance = 24-Distance1;
187     actualDistancedouble = actualDistance;
188     HAL_Delay(10);
189     PID_Compute(&TPID);
190     moveto(PIDOut);
191     sprintf(uartsendbuffer, "%d\r\n", actualDistance);
192     HAL_UART_Transmit(&uart2, uartsendbuffer, sizeof(actualDistance), HAL_MAX_DELAY);
193     /* USER CODE END WHILE */
```

Figure 24

In this part of the code, we have a while loop that causes this section to repeat. Line 182 instructs to read the SR05, line 189 calculates the PID, and line 190 commands the motor to move to a specific position. Lines 191 and 192 send the position to the computer every 10 milliseconds (as indicated in line 188).

## 6.6

```

517 //SR05_Read
518=void input_capture1(void)
519 {
520     if (Is_First_Captured1==0)
521     {
522         IC_Val11 = HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_1);
523         Is_First_Captured1 = 1;
524
525         __HAL_TIM_SET_CAPTUREPOLARITY(&htim8, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING);
526     }
527     else if (Is_First_Captured1==1)
528     {
529         IC_Val21 = HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_1);
530         __HAL_TIM_SET_COUNTER(&htim8, 0);
531
532         if (IC_Val21 > IC_Val11)
533         {
534             Difference1 = IC_Val21-IC_Val11;
535             if(((Difference1*2)/100)<60) Distance1 = (Difference1*2)/100;
536         }
537
538         Is_First_Captured1 = 0;
539
540         __HAL_TIM_SET_CAPTUREPOLARITY(&htim8, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_RISING);
541         __HAL_TIM_DISABLE_IT(&htim8, TIM_IT_CC1);
542     }
543 }
544

```

```

548=void input_capture2(void)
549 {
550     if (Is_First_Captured2==0)
551     {
552         IC_Val12 = HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_2);
553         Is_First_Captured2 = 1;
554
555         __HAL_TIM_SET_CAPTUREPOLARITY(&htim8, TIM_CHANNEL_2, TIM_INPUTCHANNELPOLARITY_FALLING);
556     }
557     else if (Is_First_Captured2==1)
558     {
559         IC_Val22 = HAL_TIM_ReadCapturedValue(&htim8, TIM_CHANNEL_2);
560         __HAL_TIM_SET_COUNTER(&htim8, 0);
561
562         if (IC_Val22 > IC_Val12)
563         {
564             Difference2 = IC_Val22-IC_Val12;
565             if(((Difference2*2)/100)<60) Distance2 = (Difference2*2)/100;
566         }
567
568         Is_First_Captured2 = 0;
569
570         __HAL_TIM_SET_CAPTUREPOLARITY(&htim8, TIM_CHANNEL_2, TIM_INPUTCHANNELPOLARITY_RISING);
571         __HAL_TIM_DISABLE_IT(&htim8, TIM_IT_CC2);
572     }
573 }

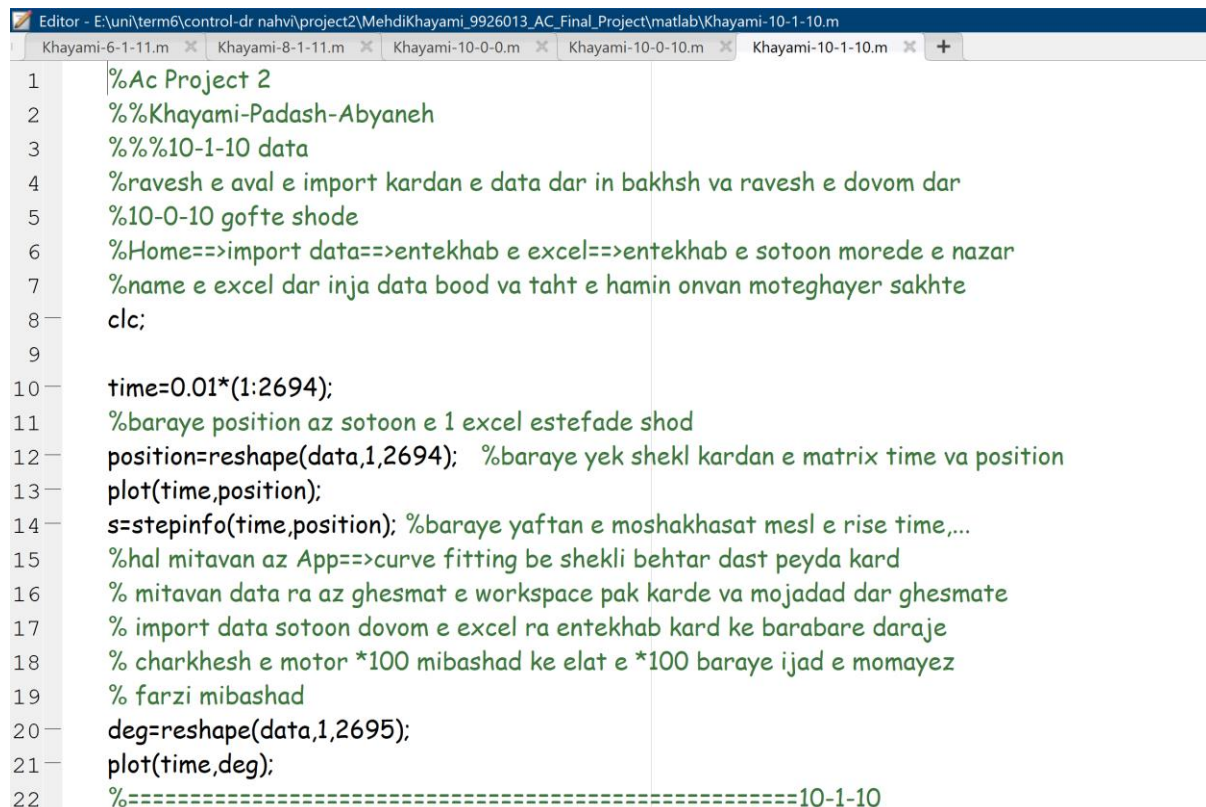
```

Figures 25 and 26

These two codes are also for reading data from ultrasonic sensors 1 and 2, which are specified in the code.

## 7. MATLAB Codes for the Graph:

To draw the graph, we first receive the data from the computer using the C code written in the STM32 environment in a .txt format. The numbers in the first column correspond to the position of the ball, and the numbers in the second column represent the motor's rotation angle multiplied by 100. This multiplication simulates decimal representation; it should be noted that the text files are also attached. We then transfer this data to Excel, and subsequently, MATLAB reads the Excel file, allowing us to create the desired variables. More detailed explanations are provided in the images below, which relate to the MATLAB software.



```
Editor - E:\uni\term6\control-dr nahvi\project2\MehdiKhayami_9926013_AC_Final_Project\matlab\Khayami-10-1-10.m
Khayami-6-1-11.m  Khayami-8-1-11.m  Khayami-10-0-0.m  Khayami-10-0-10.m  Khayami-10-1-10.m  +
1  %Ac Project 2
2  %%Khayami-Padash-Abyaneh
3  %%%10-1-10 data
4  %raveshe aval e import kardan e data dar in bakhsh va raveshe dovom dar
5  %10-0-10 gofte shode
6  %Home==>import data==>entekhab e excel==>entekhab e sotoon morede e nazar
7  %name e excel dar inja data bood va taht e hamin onvan moteghayer sakhte
8  clc;
9
10 time=0.01*(1:2694);
11 %baraye position az sotoon e 1 excel estefade shod
12 position=reshape(data,1,2694); %baraye yek shekl kardan e matrix time va position
13 plot(time,position);
14 s=stepinfo(time,position); %baraye yaftan e moshakhasat mesl e rise time,...
15 %hal mitavan az App==>curve fitting be shekli behtar dast peyda kard
16 % mitavan data ra az ghesmat e workspace pak karde va mojadad dar ghesmate
17 % import data sotoon dovom e excel ra entekhab kard ke barabare daraje
18 % charkhesh e motor *100 mibashad ke elat e *100 baraye ijad e momayez
19 % farzi mibashad
20 deg=reshape(data,1,2695);
21 plot(time,deg);
22 %=====10-1-10
```

Figure 27 -MATLAB code:  $k_p = 10$ ,  $k_i = 1$ ,  $k_d = 10$



```

Editor - E:\uni\term6\control-dr nahvi\project2\MehdiKhayami_9926013_AC_Final_Project\matlab\Khayami-10-0-10.m
Khayami-6-1-11.m  Khayami-8-1-11.m  Khayami-10-0-0.m  Khayami-10-0-10.m  Khayami-10-1-10.m  +
1  %Khayami-Padash-Abyaneh
2  %raveshe dovom e import data
3  %Home==>import data==>entekhab e har do sotoon va ijad e yek Numeric Matrix
4  %ke dar natije yek moteghayer e 'X*2' khahad sakht taht e onvan e data sepas baraye har kari
5  %mibayest az sotoon ya radife mad e nazar estefade kard
6  data=reshape(data,2,2632);
7  time=0.01*(1:2632);
8  position=data(1,:);%entekhab e hameye sotoon haye radif e aval
9  plot(time,position);
10 s=stepinfo(time,position);%baraye yaftan e moshakhasat mesl e rise time,...
11 deg=data(2,:);
12 plot(time,deg);
13 %=====10-0-10

```

Figure 28-MATLAB code:  $k_p = 10$ ,  $k_i = 0$ ,  $k_d = 10$

```

Editor - E:\uni\term6\control-dr nahvi\project2\MehdiKhayami_9926013_AC_Final_Project\matlab\Khayami-10-0-0.m
Khayami-6-1-11.m  Khayami-8-1-11.m  Khayami-10-0-0.m  Khayami-10-0-10.m  Khayami-10-1-10.m  +
1  %Khayami-Padash-Abyaneh
2  %raveshe dovom e import data
3  %Home==>import data==>entekhab e har do sotoon va ijad e yek Numeric Matrix
4  %ke dar natije yek moteghayer e 'X*2' khahad sakht taht e onvan e data sepas baraye har kari
5  %mibayest az sotoon ya radife mad e nazar estefade kard
6  data=reshape(data,2,2788);
7  time=0.01*(1:2788);
8  position=data(1,:);%entekhab e hameye sotoon haye radif e aval
9  plot(time,position);
10 s=stepinfo(time,position);%baraye yaftan e moshakhasat mesl e rise time,...
11 deg=data(2,:);
12 plot(time,deg);
13 %=====10-0-0

```

Figure 29-MATLAB code:  $k_p = 10$ ,  $k_i = 0$ ,  $k_d = 0$

```

Editor - E:\uni\term6\control-dr nahvi\project2\MehdiKhayami_9926013_AC_Final_Project\matlab\Khayami-8-1-11.m
Khayami-6-1-11.m  Khayami-8-1-11.m  Khayami-10-0-0.m  Khayami-10-0-10.m  Khayami-10-1-10.m  +
1  %Khayami-Padash-Abyaneh
2  %raveshe dovom e import data
3  %Home==>import data==>entekhab e har do sotoon va ijad e yek Numeric Matrix
4  %ke dar natije yek moteghayer e 'X*2' khahad sakht taht e onvan e data sepas baraye har kari
5  %mibayest az sotoon ya radife mad e nazar estefade kard
6  data=reshape(data,2,1953);
7  time=0.01*(1:1953);
8  position=data(1,:);%entekhab e hameye sotoon haye radif e aval
9  plot(time,position);
10 s=stepinfo(time,position);%baraye yaftan e moshakhasat mesl e rise time,...
11 deg=data(2,:);
12 plot(time,deg);
13 %=====8-1-11

```

Figure 30-MATLAB code:  $k_p = 8$ ,  $k_i = 1$ ,  $k_d = 11$

```

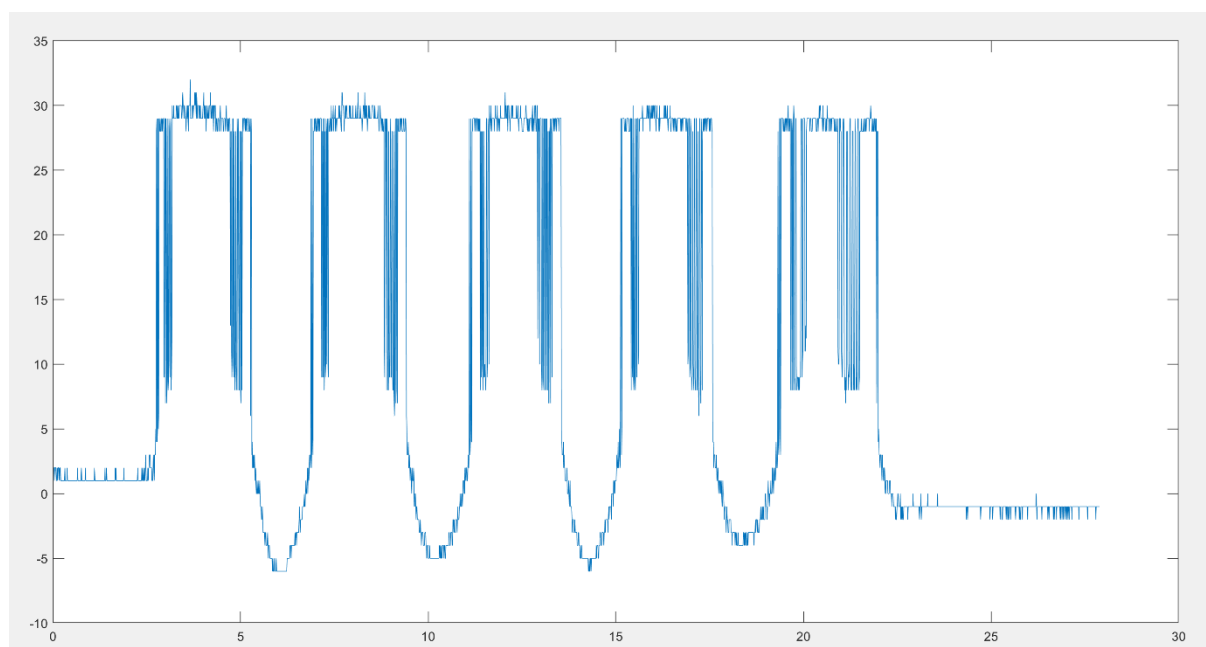
Editor - E:\uni\term6\control-dr nahvi\project2\MehdiKhayami_9926013_AC_Final_Project\matlab\Khayami-6-1-11.m
Khayami-6-1-11.m  Khayami-8-1-11.m  Khayami-10-0-0.m  Khayami-10-0-10.m  Khayami-10-1-10.m  +
1  %Khayami-Padash-Abyaneh
2  %raveshe dovom e import data
3  %Home==>import data==>entekhab e har do sotoon va ijad e yek Numeric Matrix
4  %ke dar natije yek moteghayer e 'X*2' khahad sakht taht e onvan e data sepa baraye har kari
5  %mibayest az sotoon ya radife mad e nazar estefade kard
6  data=reshape(data,2,2695);
7  time=0.01*(1:2695);
8  position=data(1,:);%entekhab e hameye sotoon haye radif e aval
9  plot(time,position);
10 s=stepinfo(time,position);%baraye yaftan e moshakhasat mesl e rise time,...
11 deg=data(2,:);
12 plot(time,deg);
13 %=====6-1-11

```

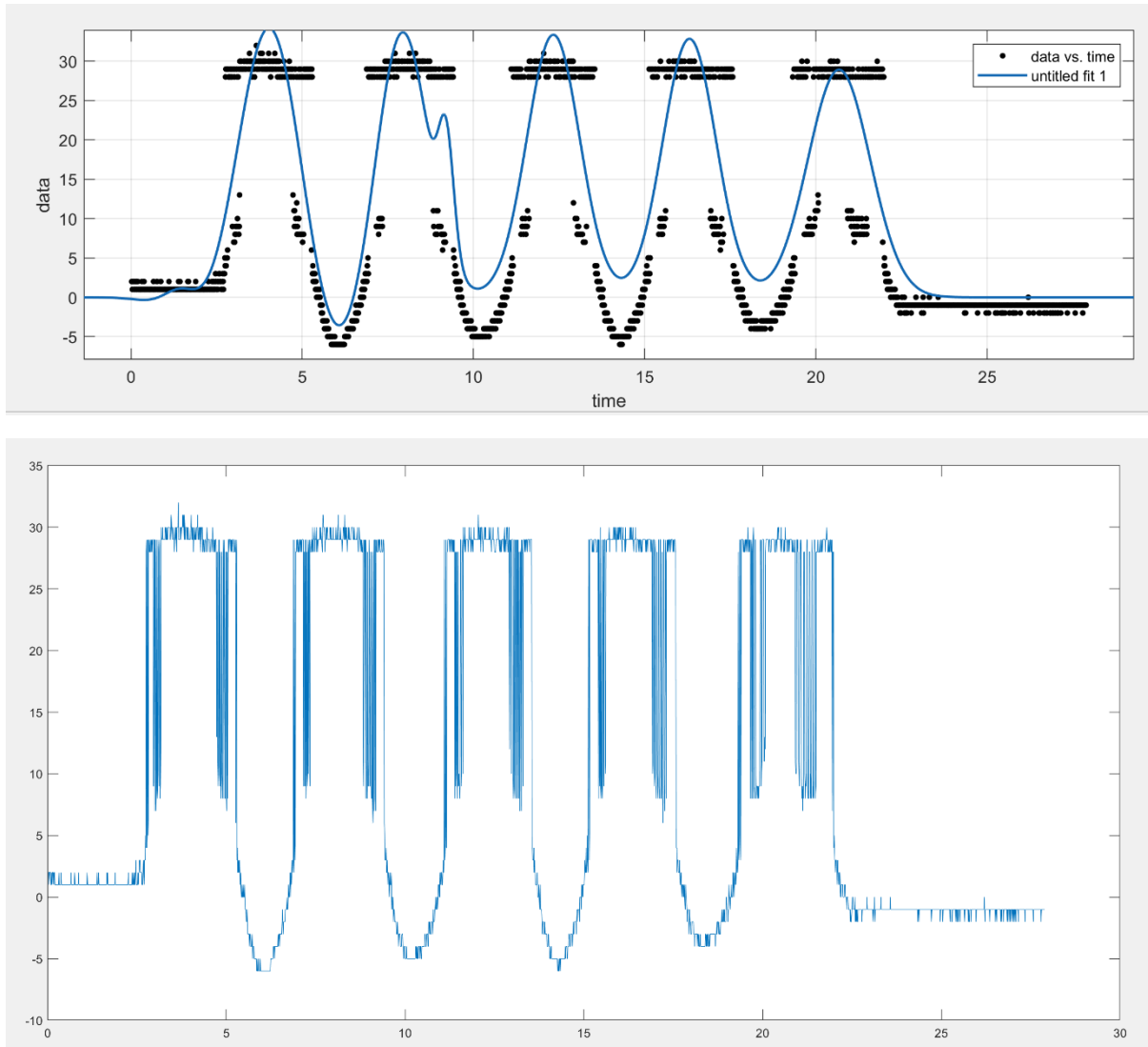
Figure 31-MATLAB code:  $k_p = 6$ ,  $k_i = 1$ ,  $k_d = 11$ .

## 8. Graphs and Analysis:

Initially, the graph related to  $k_p=10$ ,  $k_i=k_d=0$  is presented. The first graph shows the exact data of the ball's position over time, the second graph is the fitted graph, and the third graph represents the angle of the rod multiplied by 100. The reason for this multiplication is to create a functionality similar to decimals for the numbers, as the software did not provide a way to create decimal points and obtain decimal numbers.







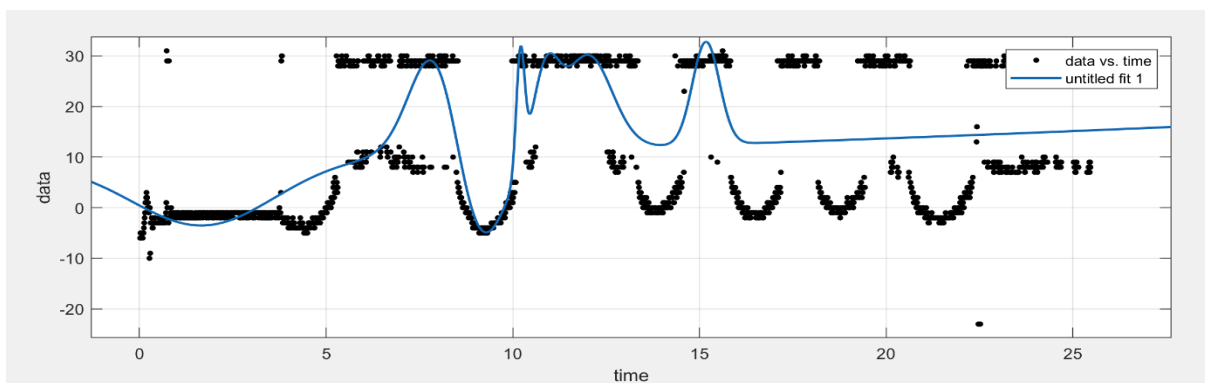
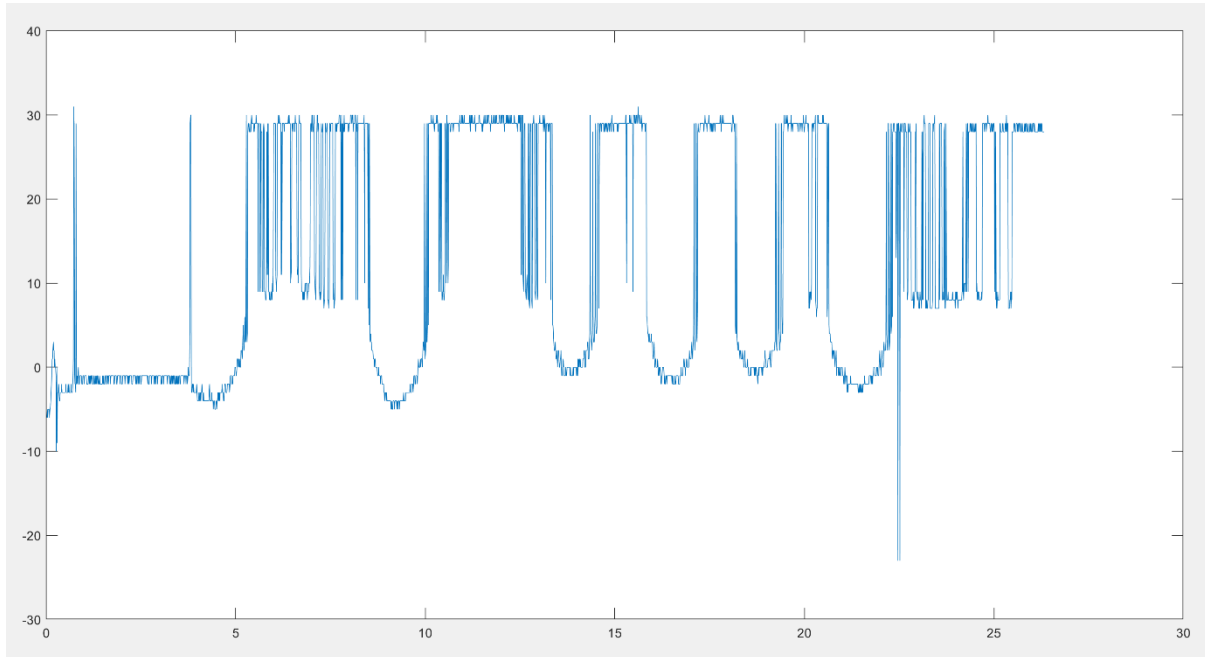
Figures 32,33 and 34- graph related to  $k_p=10$ ,  $k_i=k_d=0$

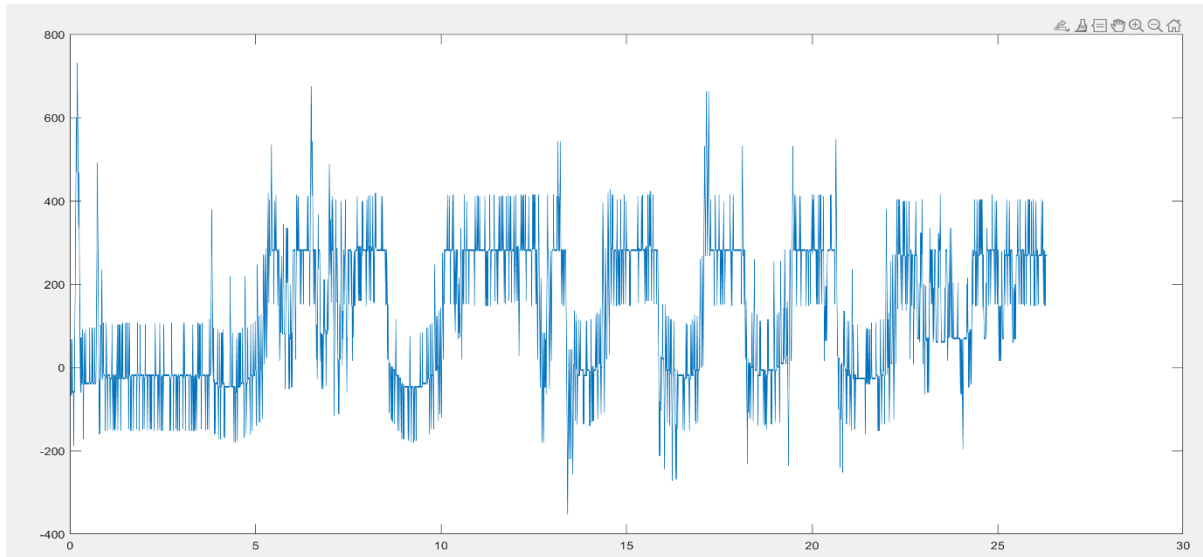
As can be seen from the above figures, the graph starts from a positive value, which represents the ball's position at the initial moments for observing and analyzing the system, and continues with significant fluctuations. This means that the system oscillates periodically, indicating that the controller is a proportional controller. It should be noted that even at the end of the time interval, the ball does not return to the desired position, which is zero.

In this system, the settling time is 23 seconds, the rise time is 5.4 seconds, the peak time is 3.6 seconds, and the peak value is 32 centimeters. The reason for the excessively high peak value is due to the low accuracy of the ultrasonic sensors and the issues arising from the ball getting too close to these sensors, causing temporary

disruptions. This problem can also be observed in the subsequent graphs at times.

The next three figures relate to the system with the characteristics  $k_p=10$ ,  $k_i=0$ ,  $k_d=10$ , which include the time-position graph, the fitted time-position graph, and the angle graph, as shown below.



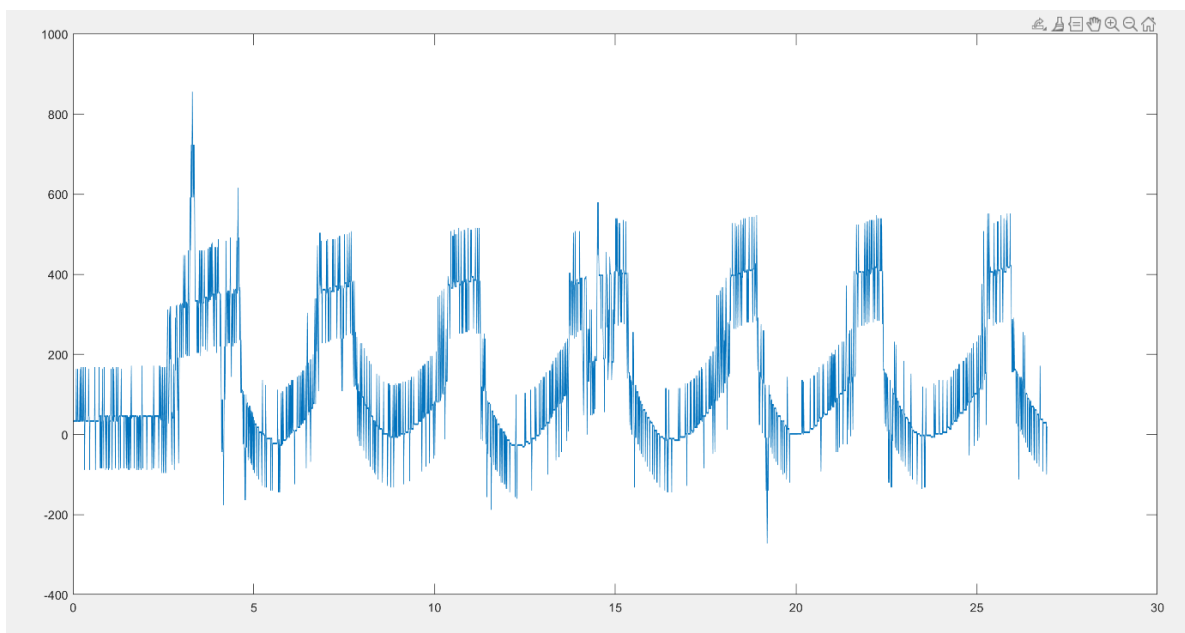
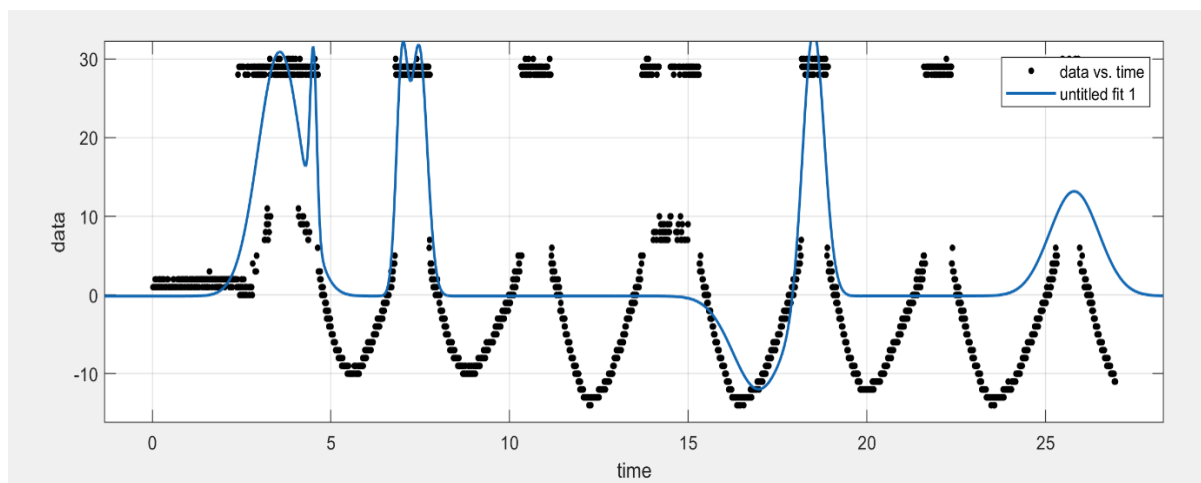
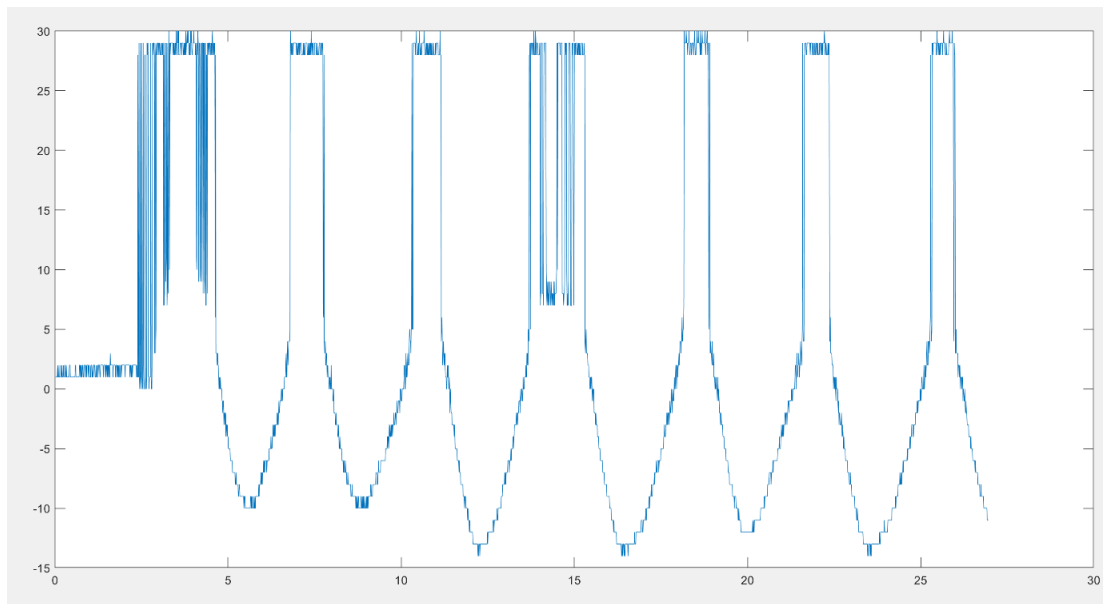


Figures 35,36 and 37- graph related to  $k_p=10$ ,  $k_i=0$ ,  $k_d=10$

With the addition of  $k_d$ , we observe that in the graphs, the oscillation amplitude has decreased, and in fact, the response speed of the system has increased. This is due to the nature of this type of coefficient, which will be explained in more detail in the next section.

It is important to note that in this system, the settling time approaches infinity and is divergent. Additionally, the rise time is 0.1 seconds, the peak time is 15.7 seconds, and the peak value is 32 centimeters.

The next three graphs pertain to a system with the characteristics  $k_p=10$ ,  $k_i=1$ ,  $k_d=10$ . As before, the first graph shows the exact data of the position over time, the second graph is the fitted version, and the third graph represents the angle.

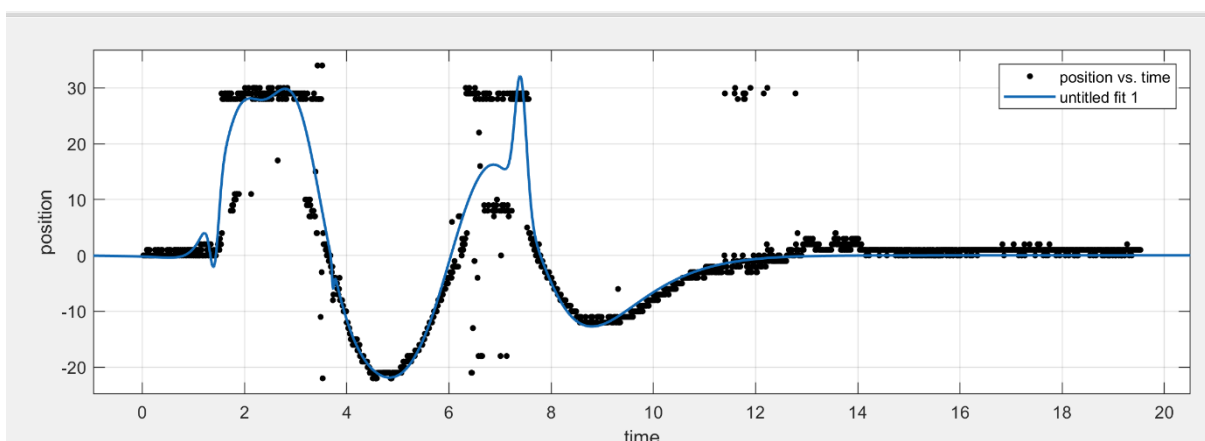
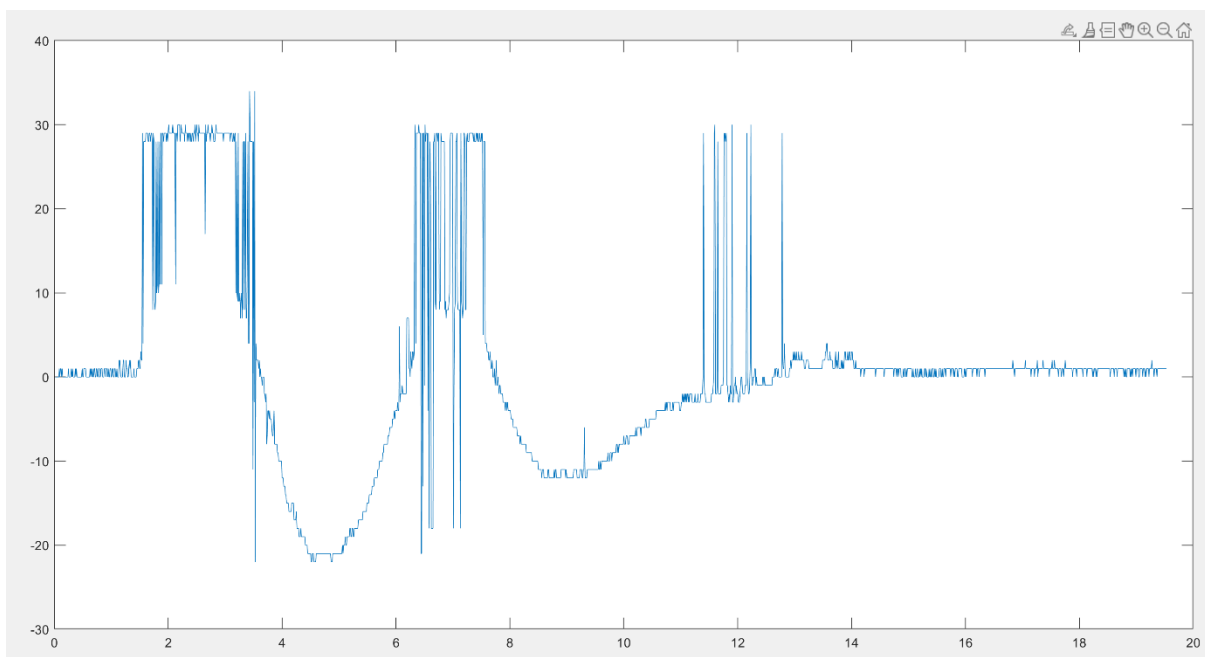


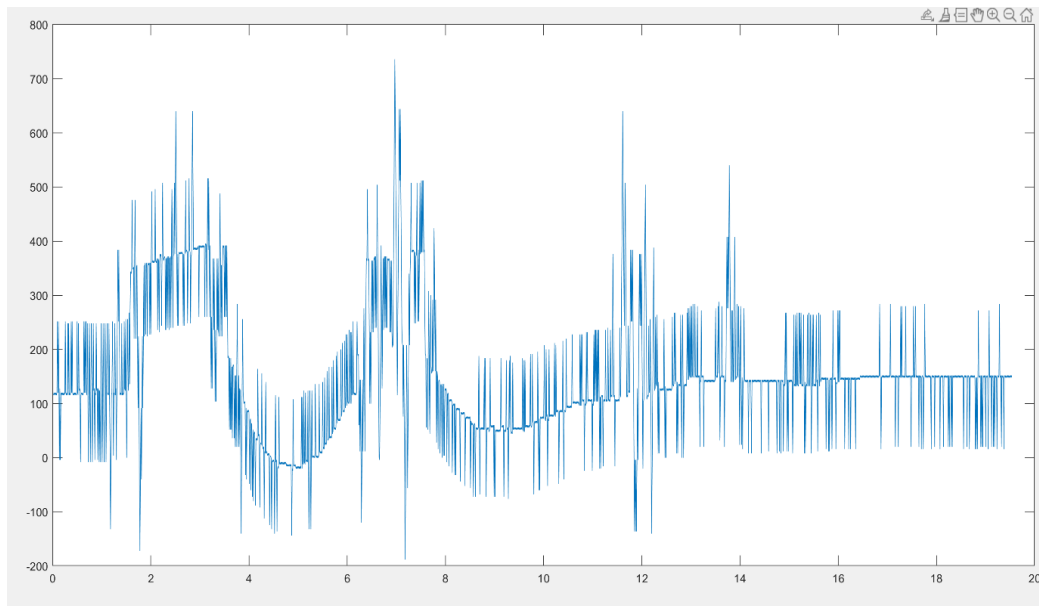
Figures 38,39 and 40- graph related to  $k_p=10$ ,  $k_i=1$ ,  $k_d=10$

Based on these graphs, it can be said that with the addition of the integral coefficient, the system has become more oscillatory and presents greater oscillation amplitudes.

The characteristics of this system are as follows: the settling time is 29 seconds, the rise time is 2.5 seconds, and the peak time is 18 seconds; additionally, the peak value is 30 centimeters.

Next, the graphs of a system with the characteristics  $k_p=8$ ,  $k_i=1$ ,  $k_d=11$  are presented.



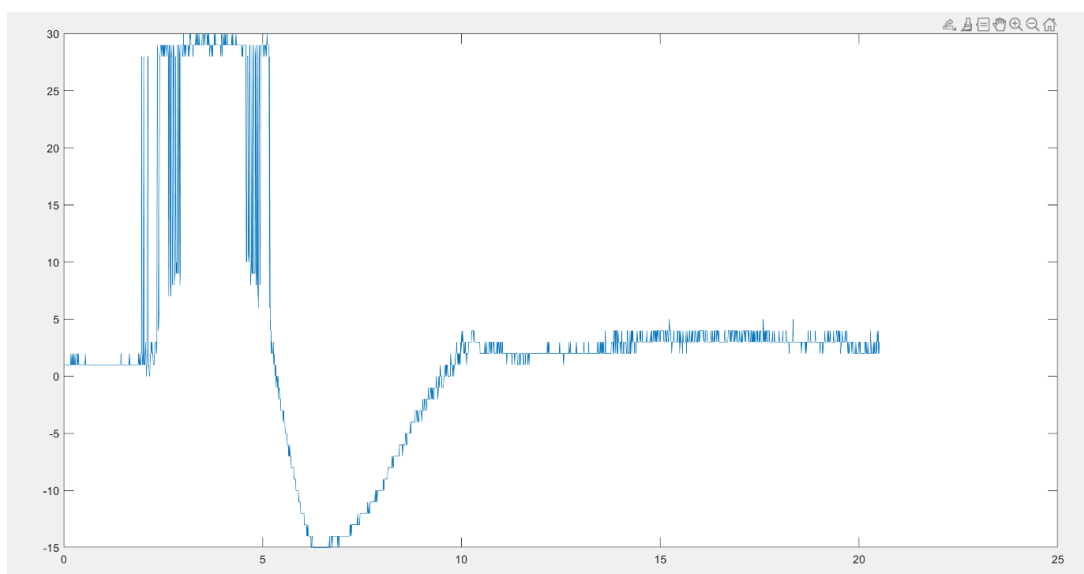


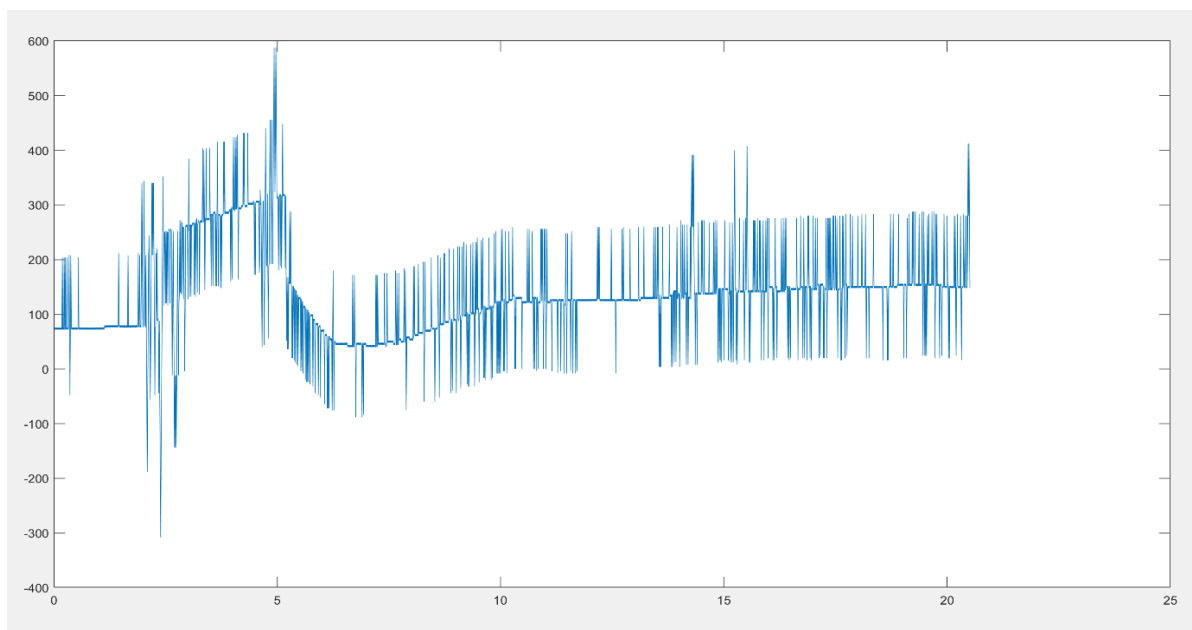
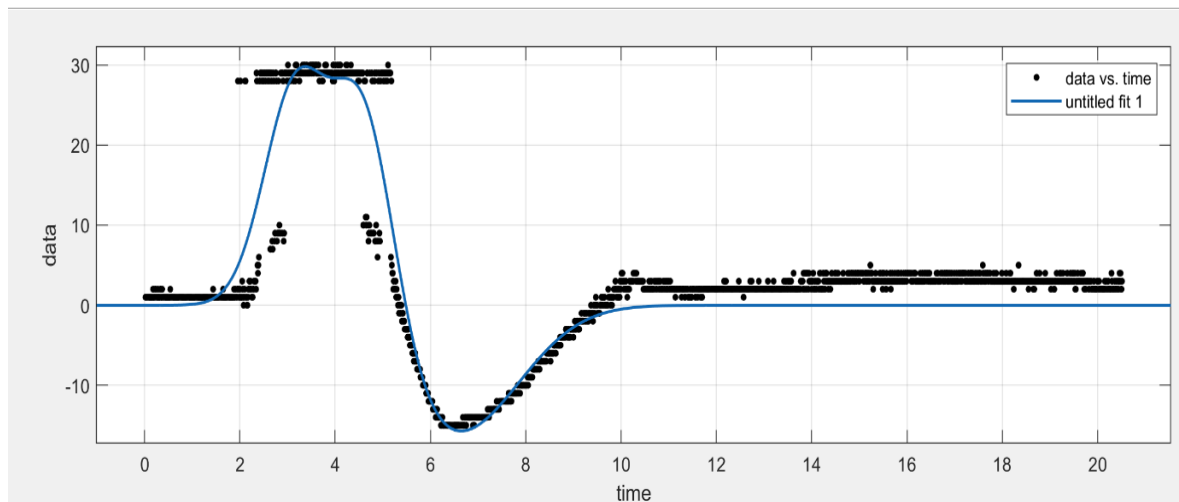
Figures 41,42 and 43- graph related to  $k_p=8$ ,  $k_i=1$ ,  $k_d=11$

The graphs clearly indicate that the system experiences fewer fluctuations and provides a better response to the input, ultimately approaching zero, which is our ideal position. We will see that this system performs better than the next system because it reaches the desired response more quickly and has a shorter settling time.

The characteristics of the mentioned system are as follows: the settling time is 13.95 seconds, the rise time is 3.6 seconds, the peak time is 3.7 seconds, and the peak value is 34 centimeters.

Finally, the graphs related to a system with the properties  $k_p=6$ ,  $k_i=1$ ,  $k_d=11$  are presented, which can be observed below.





Figures 44,45 and 46- graph related to  $k_p=6$ ,  $k_i=1$ ,  $k_d=11$

Based on the graphs, it can be observed that the magnitude and amplitude of the oscillations have decreased, and the final value approaches the desired value, which is zero. However, it provides a slower response compared to the previously mentioned system.

The results obtained from this system are as follows: the settling time is 11.7 seconds, the rise time is 5.6 seconds, the peak time is 3.8 seconds, and the peak value is 32 centimeters.

## 9. Explanation of PID Coefficients:

A PID controller is a feedback controller that uses three parameters—P, I, and D—to convert a linear control input into a control output.

The parameters P, I, and D each have specific roles in controlling the system:

- **(Proportional) P:** This parameter determines how the control error (i.e., the difference between the desired and actual values) should influence the output at any moment. Increasing the value of P increases the control output and makes the response to the control error faster.
- **(Integral) I:** This parameter calculates the cumulative effect of the control error over time. Increasing the value of I improves the stability of the system.
- **(Derivative) D:** This parameter considers the rate of change of the control error. Increasing the value of D allows the controller to respond to changes more quickly and results in a response with less oscillation.

Based on the above explanations, the impact of each of the parameters P, I, and D on the control of the system in the PID controller regarding the angle of the pendulum is as follows:

- If the value of P is large, the control output responds more quickly to the controller signal, resulting in a faster response. However, if the value of P is too large, the control output will exhibit severe oscillatory responses, leading to increased instability.
- If the value of I is large, the control error is generally reduced, and the stability of the system improves. However, if the value of I is too large, the control output will also exhibit severe oscillatory responses, increasing instability.
- If the value of D is large, the controller responds more quickly to changes, and the response to the control error is accompanied by less oscillation. However, if the value of D is too large, the control output will again show severe oscillatory responses, leading to increased instability.



## 10. Scoring Section of This Project:

As we know, the goal of this project is to return the ball to its initial position, but we have expanded this part so that we can also define the position for the system. For this purpose, additional code has been added, such that for  $k_p$ , we enter 8k, for changing  $k_i$  we enter 1i, for  $k_d$  we enter 11d, and for the desired position we type, for example, 2m (where m stands for position; note that the unit is in centimeters). Another way to change all these values in one line is to write 8k1i11d2m.

## 11. Task Assignments:

The responsibilities of the group members are as follows:

- **Mahdi Khiyami:** 33% for preparing materials, 33% for STM32 coding, 33% for modeling and theoretical analysis, 50% for report preparation, and 45% for MATLAB coding and graph analysis.
- **Mohammad Maleki Abyaneh:** 33% for preparing materials, 33% for STM32 coding, 33% for modeling and theoretical analysis, 50% for report preparation, and 45% for MATLAB coding and graph analysis.
- **Alireza Padas:** 100% for device construction, 33% for STM32 coding, 33% for modeling and theoretical analysis, 10% for MATLAB coding and graph analysis, and 33% for preparing materials.

## 12. References:

- [https://en.wikipedia.org/wiki/Rise\\_time](https://en.wikipedia.org/wiki/Rise_time)
- <https://www.youtube.com/watch?v=ARewp6SCwGs>
- [https://www.youtube.com/watch?v=a\\_DW7xznPco](https://www.youtube.com/watch?v=a_DW7xznPco)
- <https://ch.mathworks.com/help/simulink/slref/step.html>
- <https://ch.mathworks.com/help/control/ug/design-a-pid-controller-using-simulated-i-o-data.html#:~:text=To%20open%20a%20tool%20that,Controller%20block%20from%20the%20model.>
- <https://ch.mathworks.com/help/ident/ref/dynamicsystem.stepinfo.html;jsessionid=6b28e793e5065351890a41478d7c>
- [https://www.youtube.com/results?search\\_query=overshoot+in+matlab](https://www.youtube.com/results?search_query=overshoot+in+matlab)
- <https://ch.mathworks.com/help/curvefit/curve-fitting.html>