«In the name of God»

Design and Implementation of Ball and Plate System

A Bachelor's degree project

Presented to the

Faculty of

University of Tabriz

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of

Electrical Engineering

Supervisor and advisor

Dr. Mahdi Baradaran Nia

By

Alireza Jafarnezhad Moghaddam Ozbak

December, 2023

ABSTRACT

The "**Control of Ball and Plate System**" project explores the unique challenges of controlling open-loop unstable systems, using the inherently nonlinear and under-actuated ball and plate system as a case study. A state-feedback controller was designed and implemented to manage the complex dynamics of a freely rolling ball on a plate. Despite not fully meeting initial expectations, the system effectively maintained control of the ball around the equilibrium point. This project not only demonstrates the practical application of control theory principles but also provides valuable insights for future enhancements in the field of control systems.

Table of Contents

LIST OF FIGURES

ACKNOWLEDGMENTS

I am grateful to my major professor, Dr. Mahdi Baradarannia, for giving me the opportunity to conduct this research. He provided me with valuable advice, support and guidance throughout the research process. I would not have been able to complete this research without the constant support and encouragement of my friends and family, especially my parents. I thank them from the bottom of my heart.

1. Introduction

1.1 Background

The ball and plate system, an extension of the ball and beam system, is a four-degree of freedom system where a ball can roll freely on a plate. This under-actuated system, influenced by nonlinearities such as motion resistance and backlash in transmission, serves as a benchmark to test various nonlinear control schemes. The experimental platform includes a flat plate, a ball, motors, their driving systems, and a camera. The plate rotates around its x and y axis in two perpendicular directions, with inclinations changed by a servo control system. Despite the challenges in creating a ball and plate mechanism, its potential for performing manifold control strategies makes it desirable.

1.2 Project Objective

The ball and plate (B&P) system has direct applications in robotic manipulation of round objects and serves as an educational tool for students studying control systems engineering. Unlike other educational models, the B&P system offers the opportunity to study trajectory tracking problems in addition to stability problems. The system has been studied and controlled using several methods ranging from linear control techniques to optimal, intelligent, and nonlinear control.

2. Literature Review

Various control methods for the ball and plate system have been explored [4]-[7]. These methods encompass a controller design rooted in classical and modern control theory, a supervisory fuzzy controller for motion control of the system [4], a nonlinear velocity observer for output regulation [5], position regulation of the ball using double feedback loops with Recursive backstepping design for the external loop and a switching control scheme for the inner feedback loop [6], and a PID neural network controller developed through a genetic algorithm [7].

Several studies have combined observer and fuzzy control to construct a trajectory tracking control law for the ball and plate system, incorporating a touch screen and a rotary cylinder [8]. Other research has utilized visual servo control skill and Linear Quadratic Regulator control law for ball positioning [9]. Improved vertex control technology has also been reported [10]. Some studies have provided a hierarchical fuzzy control scheme to address the trajectory planning and tracking problem of the ball and plate system [11]. An approximate input-output feedback linearization approach has been used to design a trajectory tracking controller, simplifying the ball and plate system into two decoupled ball and beam systems [12]. However, a complete stability analysis for the closed-loop system is notably absent.

3. Methodology

3.1 Mathematical Modelling

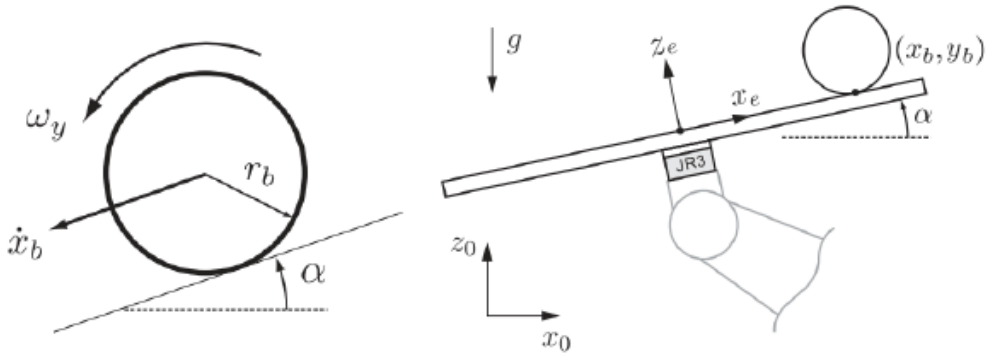

Figure 1: Side view of ball on plate system.

In this section we are going to derive the motion equations of system. In this part we assume following simplifications:

- There is no slipping for ball.
- The ball is completely symmetric and homogeneous.
- All frictions are neglected.
- The ball and plate are in contact all the time.

Here we derive dynamical equations of ball-on-plate system by the help of Lagrangian. The following precise mathematical equations are based on [1][2]. The Euler-Lagrange equation of ball-plate system is as followings:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = \mathrm{Q}_i \qquad \textbf{(1)}$$

Where qi stands for i-direction coordinate, T is kinetic energy of the system, V is potential energy of system and Q is composite force.

The system has 4 degrees of freedom; two in ball motion and two in inclination of plate. Here we assume the generalized coordinates of system are x_b, y_b and $\dot{x}_b, \dot{y}_b$ ball's position on plate

and α and β the inclination of the plate. It is important to note that we assume the center of x-y coordinates be at center of plate. The kinetic energy of ball consists of its both rotational with respect to its center of mass and translational energy:

$$T_b = \frac{1}{2} m_b (\dot{x}_b^2 + \dot{y}_b^2) + \frac{1}{2} I_b (\omega_x^2 + \omega_y^2) \tag{2}$$

Where m_b is mass of the ball and I_b is moment of inertia of the ball; x_b and y_b are ball's translational velocities along x-axis and y-axis; ω_α and ω_β are ball's rotational velocities along x-axis and y-axis. The following relations between translational velocities and rotational velocities:

$$\dot{x}_b = r_b \omega_x \quad , \quad \dot{y}_b = r_b \omega_y \tag{3}$$

In which r_b denotes ball's radius. By substituting equations 3 into equations 4 we will have:

$$T_b = \frac{1}{2}[m_b (\dot{x}_b^2 + \dot{y}_b^2) + \frac{I_b}{r_b^2}(\dot{x}_b^2 + \dot{y}_b^2)] = \frac{1}{2}(m_b + \frac{I_b}{r_b^2})(\dot{x}_b^2 + \dot{y}_b^2) \tag{4}$$

The kinetic energy of the plate, by considering ball as a point mass which is placed in (x_b, y_b) consists of its rotational energy with respect to its center of mass:

$$T_p = \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2} m_b (x_b \dot{\alpha} + y_b \dot{\beta})^2 \tag{5}$$

Where α and β are plate's angle of inclination along x-axis and y-axis, respectively. Therefore $\dot{\alpha}$ and $\dot{\beta}$ are plate's rotational velocity. Here we can calculate the kinetic energy of system as followings:

$$\begin{aligned} T = T_b + T_p = & \frac{1}{2}(m_b + \frac{I_b}{r_b^2})(\dot{x}_b^2 + \dot{y}_b^2) \\ & + \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2} m_b (x_b \dot{\alpha} + y_b \dot{\beta})^2 \end{aligned} \tag{6}$$

The relative potential energy of the ball to horizontal plane in the center of the inclined plate can be calculated as:

$$V_b = m_b g h = m_b g (x_b \sin\alpha + y_b \sin\beta) \tag{7}$$

Here we can derive the system's equation by Lagrangian and equations 4-7:

$$L = T_b + T_p - V_b \tag{8}$$

We use L to derive system's equations:

$$\frac{\partial T}{\partial \dot{\alpha}} = (I_p + I_b)\dot{\alpha} + m_b x_b (x_b\dot{\alpha} + y_b\dot{\beta}) \quad , \quad \frac{\partial L}{\partial \alpha} = mgx_b \cos(\alpha) \tag{9}$$

$$\frac{\partial T}{\partial \dot{\beta}} = (I_p + I_b)\dot{\beta} + m_b y_b (y_b\dot{\beta} + x_b\dot{\alpha}) \quad , \quad \frac{\partial L}{\partial \beta} = mgy_b \cos(\beta) \tag{10}$$

$$\frac{\partial T}{\partial \dot{x}_b} = (m_b + \frac{I_b}{r_b^{\,2}})\dot{x}_b \quad , \quad \frac{\partial L}{\partial x_b} = m_b(x_b\dot{\alpha} + y_b\dot{\beta})\dot{\alpha} - m_b g \sin(\alpha) \tag{11}$$

$$\frac{\partial T}{\partial \dot{y}_b} = (m_b + \frac{I_b}{r_b^{\,2}})\dot{y}_b \quad , \quad \frac{\partial L}{\partial y_b} = m_b(x_b\dot{\alpha} + y_b\dot{\beta})\dot{\beta} - m_b g \sin(\beta) \tag{12}$$

We assume generalized toques as τ_x *and* τ_y which are exerted torques on the plate. From Lagrange-Euler equation we can write:

$$\begin{aligned} \frac{d}{dt}\frac{\partial T}{\partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} &= (I_p + I_b)\ddot{\alpha} + m_b x_b^{\,2}\ddot{\alpha} + 2m_b x_b \dot{x}_b \dot{\alpha} + m_b x_b y_b \ddot{\alpha} \\ &+ m_b \dot{x}_b y_b \dot{\beta} + m_b x_b \dot{y}_b \dot{\beta} - m_b g \cos(\alpha) = \tau_x \end{aligned} \tag{13}$$

$$\begin{aligned} \frac{d}{dt}\frac{\partial T}{\partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} &= (I_p + I_b)\ddot{\beta} + m_b x_b^{\,2}\ddot{\beta} + 2m_b y_b \dot{y}_b \dot{\beta} + m_b x_b y_b \ddot{\beta} \\ &+ m_b \dot{y}_b x_b \dot{\beta} + m_b y_b \dot{x}_b \dot{\alpha} - m_b g \cos(\beta) = \tau_y \end{aligned} \tag{14}$$

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{x}_b} - \frac{\partial L}{\partial x_b} = (m_b + \frac{I_b}{r_b^{\,2}})\ddot{x}_b - m_b(x_b\dot{\alpha} + y_b\dot{\beta})\dot{\alpha} + m_b g \sin(\alpha) = 0 \tag{15}$$

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{y}_b} - \frac{\partial L}{\partial y_b} = (m_b + \frac{I_b}{r_b^{\,2}})\ddot{y}_b - m_b(x_b\dot{\alpha} + y_b\dot{\beta})\dot{\beta} + m_b g \sin(\beta) = 0 \tag{16}$$

So the non-linear differential equations for the ball-plate-system as followings:

$$(m_b + \frac{I_b}{r_b^{\,2}})\ddot{x}_b - m_b(x_b\dot{\alpha}^2 + y_b\dot{\alpha}\dot{\beta}) + m_b g \sin(\alpha) = 0 \tag{17}$$

$$(m_b + \frac{I_b}{r_b^{\,2}})\ddot{y}_b - m_b(y_b\dot{\beta}^2 + x_b\dot{\alpha}\dot{\beta}) + m_b g \sin(\beta) = 0 \tag{18}$$

$$\begin{aligned} \tau_x &= (I_p + I_b + m_b x_b^{\,2})\ddot{\alpha} + 2m_b x_b \dot{x}_b \dot{\alpha} + m_b x_b y_b \ddot{\alpha} \\ &+ m_b \dot{x}_b y_b \dot{\beta} + m_b x_b \dot{y}_b \dot{\beta} - m_b g \cos(\alpha) \end{aligned} \tag{19}$$

$$\tau_y = (I_p + I_b + m_b y_b^{\,2})\ddot{\beta} + 2m_b y_b \dot{y}_b \dot{\beta} + m_b x_b y_b \ddot{\beta} + m_b \dot{y}_b x_b \dot{\beta} + m_b y_b \dot{x}_b \dot{\alpha} - m_b g \cos(\beta) \quad (20)$$

In these equations, m (kg) is the mass of the ball; r is the radius of the ball; x (m) is the ball position at X-axis; y (m) is the ball position at Y-axis; α (rad)is the plate deflection angle at X-axis;(rad) is the plate deflection angular at Y-axis. τ_x (N•m) is the torque exerted on the plate in x-axis and τ_y (N•m) is the torque exerted on the plate in the y-axis. I_p is the rotational inertia for plate, I_b is the rotational inertia for ball, g is the acceleration of gravity.

Motor Model

Given the specific requirements and constraints of our project, we decided to model the motor using a first-order system. This decision was influenced by two main factors:

- Motor Specifications: The MG996R has a speed of 150ms/60°.
- Project Constraints: The inclination boundaries for our project are -5° to 5°.

Considering these factors, we calculated the motor speed for our specific range of operation (10°) to be 25ms/10°.

Time Constant of the System

The time constant of a system is a measure of the system's response speed. In our case, given the motor speed calculated above, we determined the time constant of our system to be 5ms. This means that our system will reach approximately 63.2% of its total change in 5ms after a step input is applied.

This first-order model of our motor, along with the calculated time constant, will be used in the subsequent design and analysis of our control system.

$$\frac{Command}{\theta} = \frac{1}{0.005s + 1}$$

Model Linearization

Equations (17) and (18) describe the movement of the ball on the plate, show how the effects of ball acceleration rely on plate deflection angular and plate deflection angular velocity. Equations (19) and (20) show how the dynamics of plate deflection rely on exterior drive and ball position. In fact, the system inputs are α and β, not τ_x *and* τ_y, because load moment cannot affect the position of motor. For this assumption, study on ball on the plate only using (17) and (18).

The approximate value for a solid ball's moment of inertia is $I_b = \frac{2}{5} m_b r_b^2$. Therefore equations (17) and (18) can be written as:

$$m_b[\frac{5}{7}\ddot{x}_b - (x_b\dot{\alpha}^2 + y_b\dot{\alpha}\dot{\beta}) + g\sin(\alpha)] = 0 \tag{21}$$

$$m_b[\frac{5}{7}\ddot{y}_b - (y_b\dot{\beta}^2 + x_b\dot{\alpha}\dot{\beta}) + g\sin(\beta)] = 0 \tag{22}$$

We can linearize above equations by assuming:

• Small angle of inclination for the plate (up to ± 5°):

$$|\alpha| < 5^\circ, |\beta| < 5^\circ \Rightarrow \sin(\alpha) \simeq \alpha, \sin(\beta) \simeq \beta$$

• Slow rate of change for the plate:

$$\dot{\alpha} \simeq 0, \dot{\beta} \simeq 0 \Rightarrow \dot{\alpha}\dot{\beta} \simeq 0, \dot{\alpha}^2 \simeq 0, \dot{\beta}^2 \simeq 0$$

$$\frac{5}{7}\ddot{x}_b + g\alpha = 0 \tag{23}$$

$$\frac{5}{7}\ddot{y}_b + g\beta = 0 \tag{24}$$

By linearizing the above equations, we find two separate deferential equations for each of x and y axis. Note that we can use above linear deferential equations to estimate the ball-on-plate system's states $x_b,\ y_b, \dot{x}_b,\ \dot{y}_b$. By assuming α(s) and β(s) as inputs to ball-on-plate system, we find the transfer functions:

$$P_x(s) = \frac{X_b(s)}{\alpha(s)} = \frac{-g}{\frac{5}{7}s^2} \tag{25}$$

$$P_y(s) = \frac{Y_b(s)}{\beta(s)} = \frac{-g}{\frac{5}{7}s^2} \tag{26}$$

3.2 Design of the State-Feedback Controller

In the design of the state-feedback controller, the first step was the linearization of the system. While we had previously calculated the linearized model of the system, this time we used MATLAB Simulink for the linearization process. Specifically, we used the linmodv5 function to perform the linearization. Simulink provides tools for system linearization around an equilibrium point.

Since we had decoupled the system along the x and y axes, it was sufficient to linearize the system along one axis and then generalize it to the other. The linearization process resulted in a state-space model with two states. This model will be used in the subsequent design of our state-feedback controller.

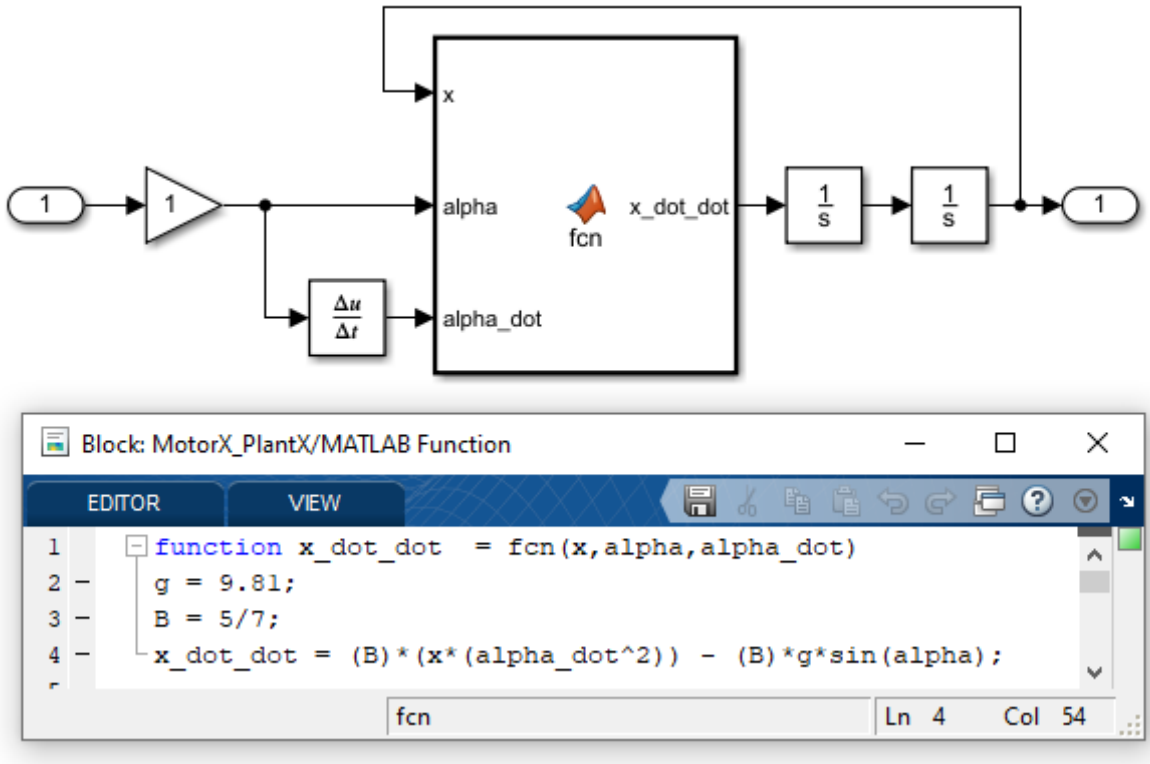

Figure 2: Model of System Along X-Axis

```
[A,B,C,D] = linmodv5('MotorX_PlantX');
rank(ctrb(A,B)); %Controllability check
rank(obsv(A,C)); %Observability check
sys0 = ss(A,B,C,D) %Conversion to state-space model
```

```
sys0 =

 A =
      x1  x2
 x1    0   1
 x2    0   0

 B =
         u1
 x1       0
 x2  -7.007

 C =
      x1  x2
 y1    1   0

 D =
      u1
 y1    0

Continuous-time state-space model.
```

```
sys0 = tf(sys0) %Conversion to transfer function
```

```
sys0 =

  -7.007
  ------
   s^2

Continuous-time transfer function.
```

Considerations Regarding the Motor Model

While we have calculated a detailed model for the MG996R motor in a previous section, it is not incorporated into our state-space model for the following reasons:

- Motor Speed: The MG996R motor has a high speed, which allows it to reach the desired angle almost immediately after a command is issued. This rapid response time negates the need for us to include the state of the angle in our model.

- Hardware Limitations: The type of motor we used does not allow for hardware manipulation using state feedback.

Therefore, we decided to proceed with the state-space model obtained from the linearization process, without incorporating the motor model.

Conversion to Discrete-Time Model

Given the continuous-time state-space model, we then converted it to a discrete-time model with a desired sample time of 0.01 seconds.

```
Ts = 0.01; %Desired Sample time
[A_D,B_D] = c2d(A,B,Ts) %Conversion from Continuous-time to Discrete-time
```

```
A_D = 2×2
   1.0000    0.0100
        0    1.0000
B_D = 2×1
   -0.0004
   -0.0701
```

```
[numz,denz] = ss2tf(A_D,B_D,C,D); %Conversion to Discrete transfer function
Gz = tf(numz,denz,Ts) %Discrete transfer function of system
```

```
Gz =

  -0.0003504 z - 0.0003504
  ------------------------
      z^2 - 2 z + 1

Sample time: 0.01 seconds
Discrete-time transfer function.
```

The next step was the design of the state-feedback controller. We used the Ackermann's formula to place the poles of the system at the desired locations. The initial location and speed of the ball on the plate was set to [0.1, 0].

```
x_initial = [0.1, 0]; %Initial Location and Speed of the ball on the plate
p = [exp(-60*Ts), exp(-60*Ts)]; %Desired pole values for system
K = acker(A_D,B_D,p) %Pole placement using acker function and obtaining the gain values for state-feedback
```

```
K = 1×2
 -290.5192  -11.4254
```

```
A_bar = A_D-B_D*K; % the feedback to the system
sys1 = ss(A_bar,B_D,C,D,Ts);
Gc = tf(sys1);
[y,tOut] = initial(sys1,x_initial,0:Ts:0.2);
plot(tOut,y) %Plotting the response of system to initial value of ball
```

Application of the Controller to the Nonlinear System

Finally, we applied the gains from the Ackermann's formula to the nonlinear system in our Simulink model. In the Simulink model, we used a quantizer for sampling data and a zero-order holder, which is inside the motor. Our model has no inputs and the response of itis related to the initial values of x and y, specifically x0 = 0.1m and y0 = -0.1m.

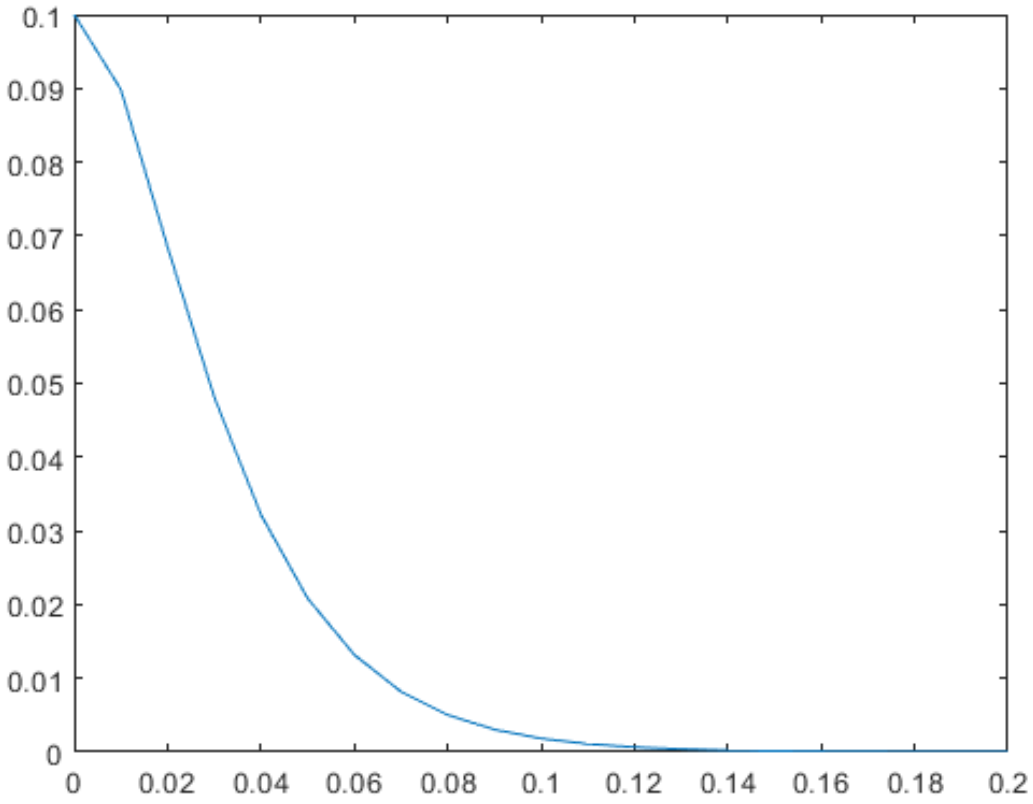

Figure 3: Response of Discrete-time System for Initial Value Along X-Axis

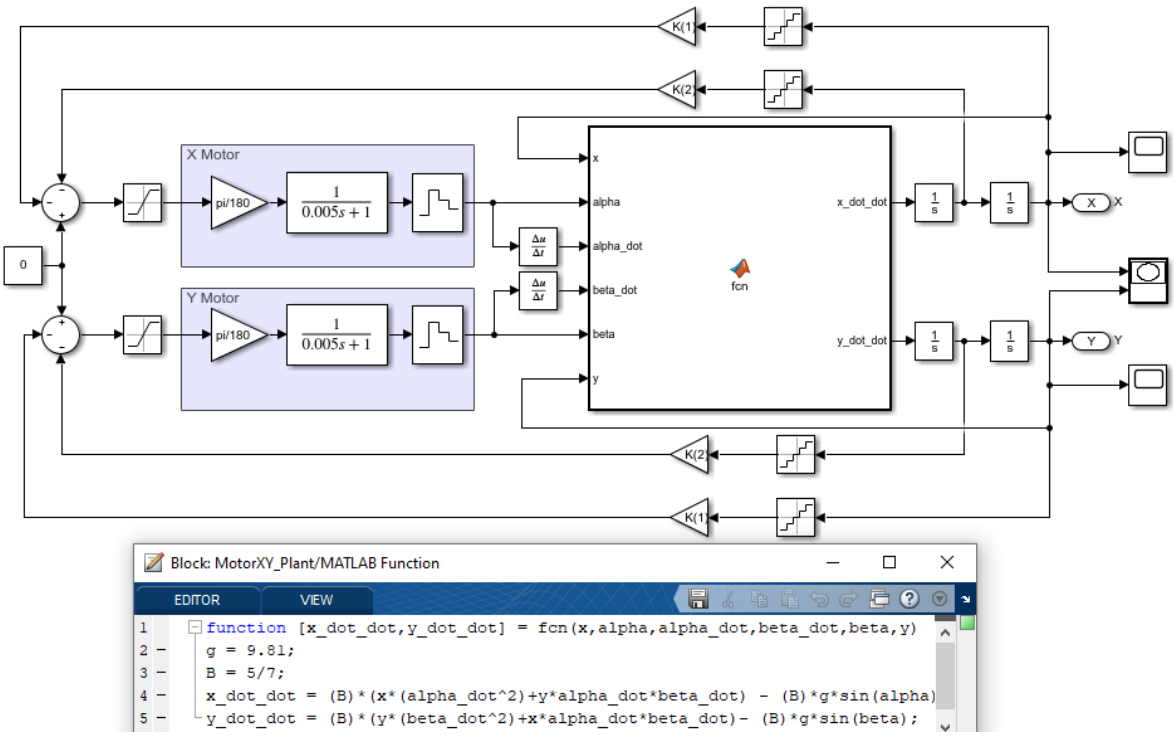

Figure 4: Model of System After Applying State-Feedback Controller

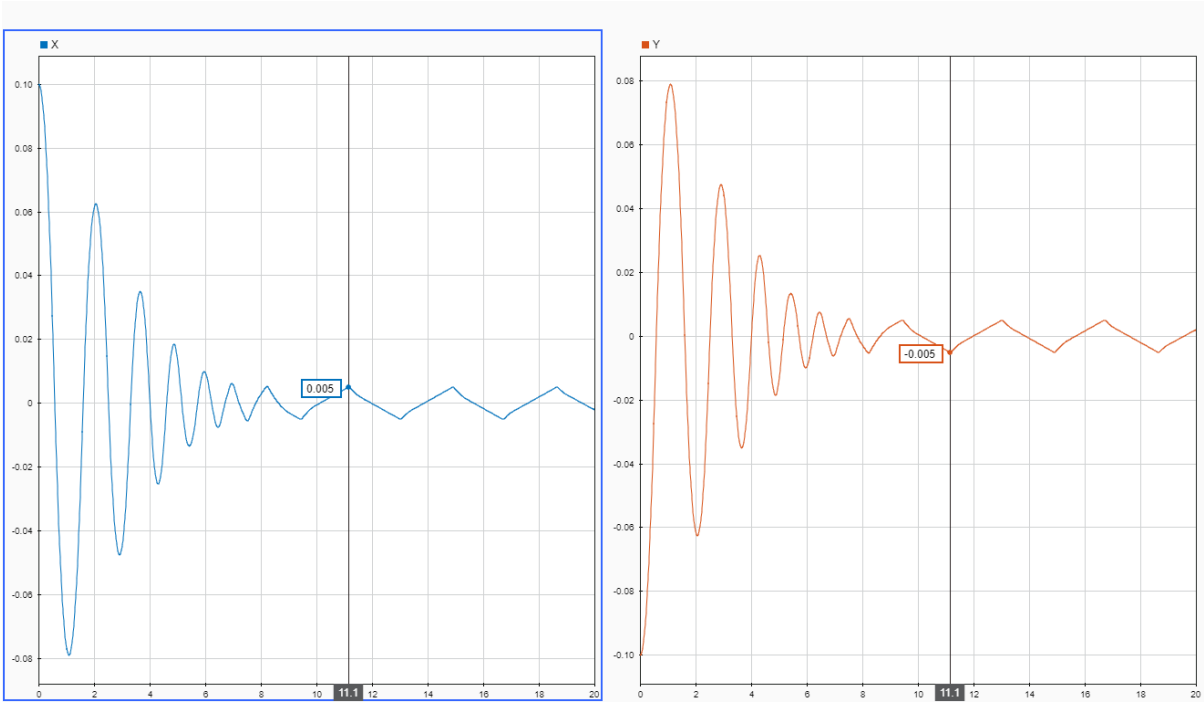

Figure 5: Response of Non-Linear System to Initial Values

Simulation Results and Next Steps

After running the simulation and plotting the location of the ball along both axes, we noticed fluctuations between 0.005m and -0.005m. This observation suggests that our state-feedback controller is effectively maintaining the ball's position within a small range around the desired equilibrium point.

Encouraged by these promising simulation results, we are now preparing to construct the real model of the system and implement the controller. This will enable us to test our controller design with the actual hardware and assess its performance under real-world conditions.

3.3 Implementation Details

list of materials used in project:

1. Arduino mega2560
2. 2x MG996r RC servo
3. IP webcam
4. Universal joint
5. Binding wire
6. Marble
7. MDF wood
8. 3mm MDF wood
9. 5mm white foam
10. Screws (as needed)
11. Double-sided glue
12. Hot glue
13. Breadboard
14. Power supply
15. Connecting Wires & Jumper wires

After reviewing various models of the system available on the internet, we chose a specific design for our implementation. We aimed to create a simple yet effective system using the materials we had at hand.

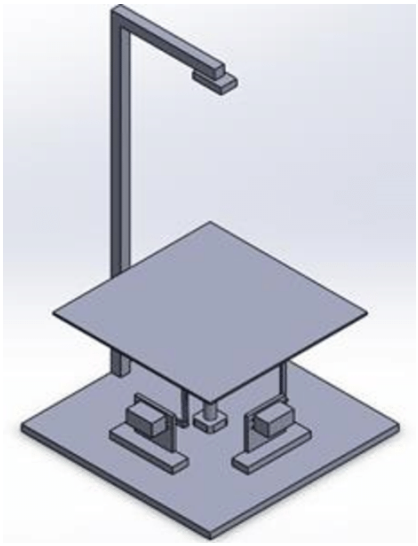

Figure 6: CAD Model of The Ball and Plate System

The main structure of our system was crafted from MDF wood, cut into the desired shape. At the center of this structure, we installed a universal joint. Two links, constructed from binding wire, were connected to two MG996R motors. This setup allowed for controlled movement along both axes.

We also included a mount for an IP webcam. This allowed us to monitor the system and gather data for further analysis.

By utilizing readily available materials and keeping the design simple, we were able to create a functional model for our project.

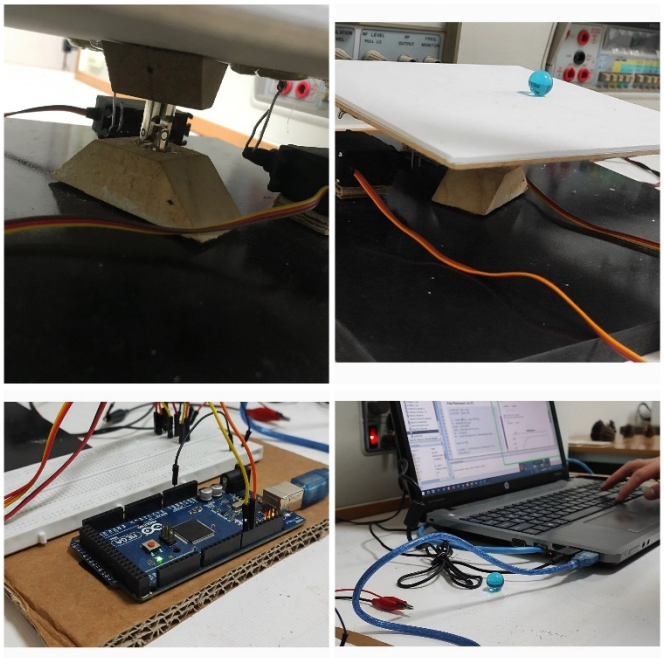

Figure 7: Many Photos of Implemented System

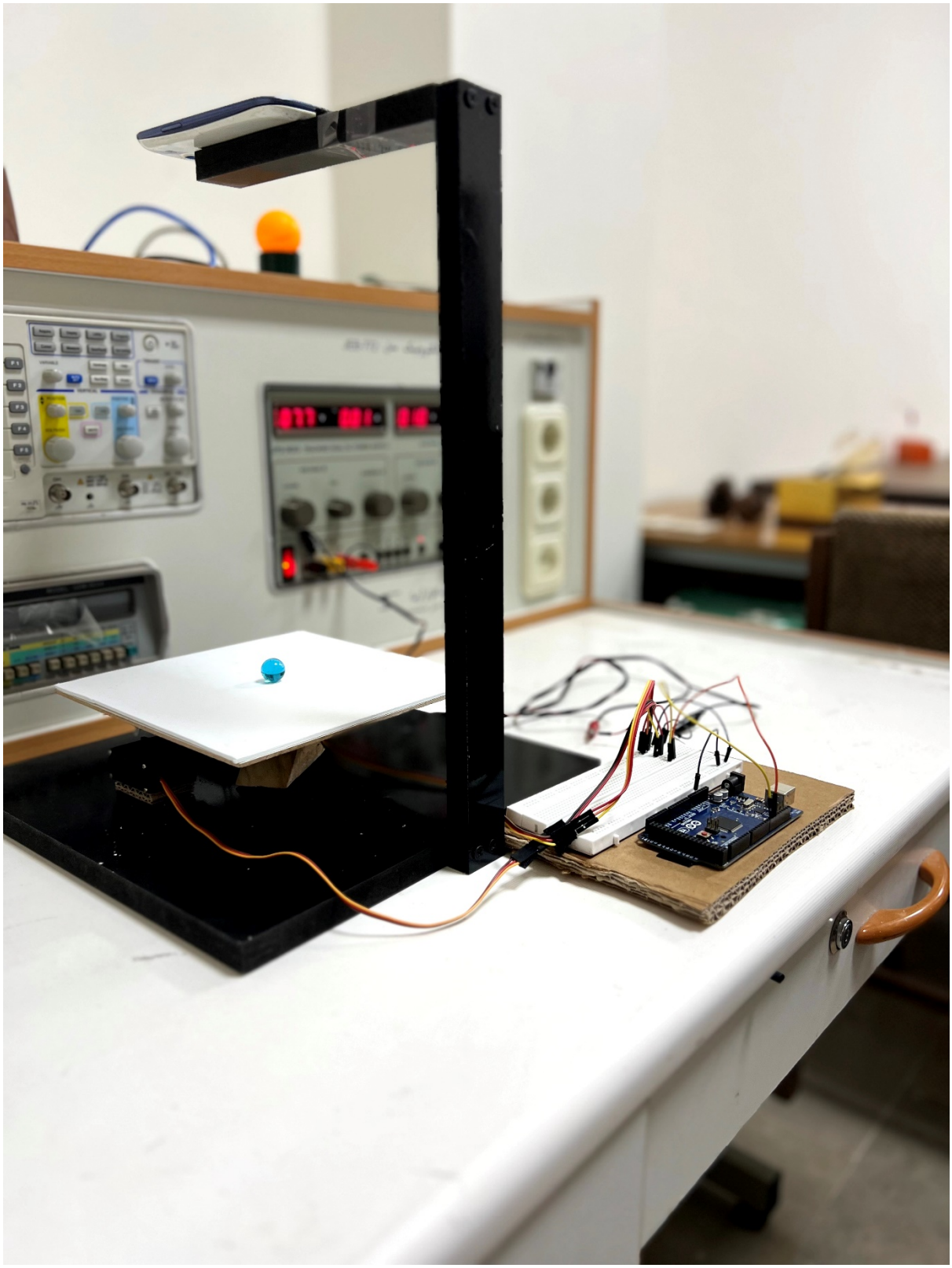

Figure 8: Side View of System

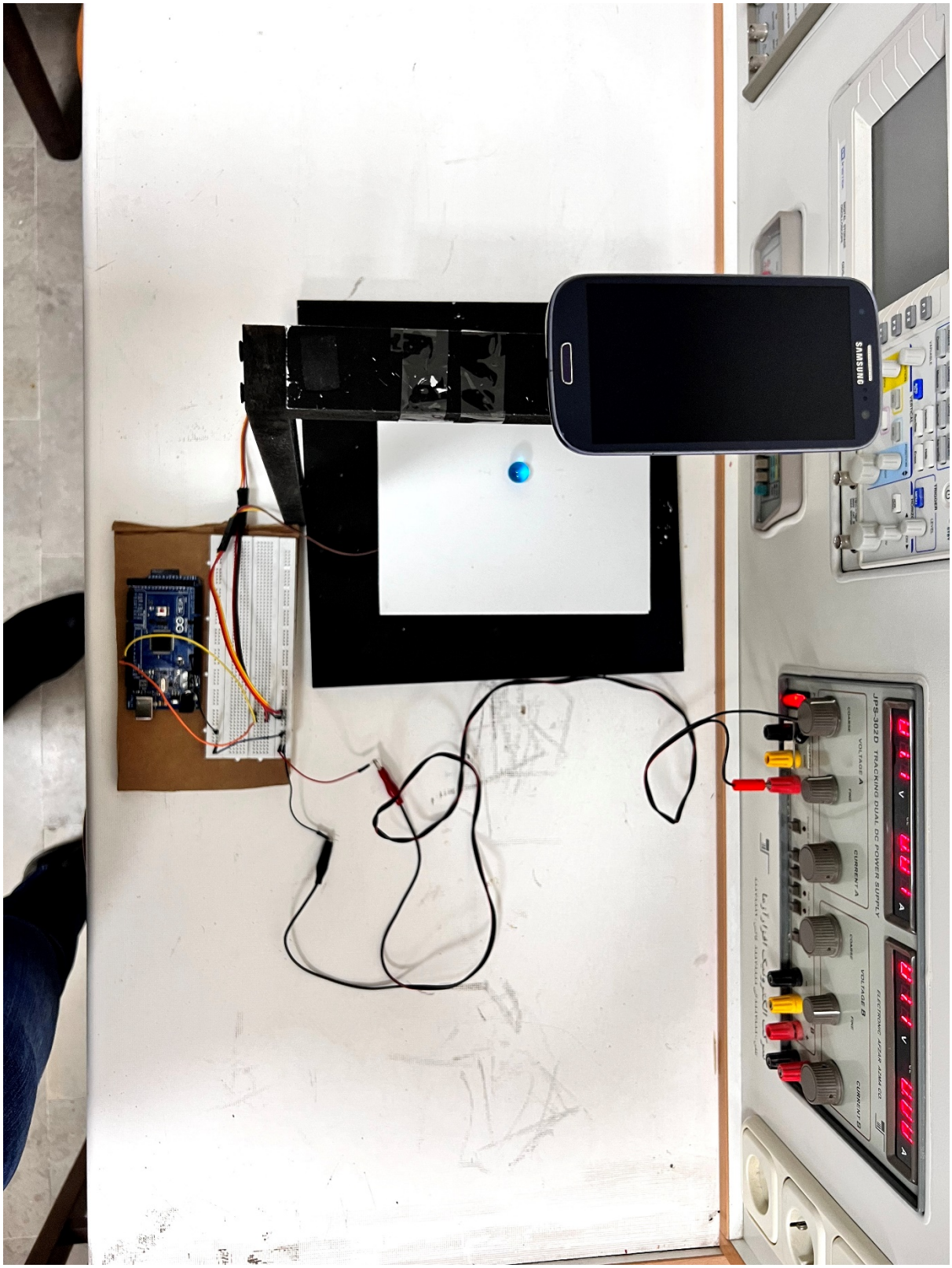

Figure 9: Top View of System

3.4 Experimental Setup

The experimental setup involved connecting motors to an Arduino, which was then connected to a laptop. The laptop was primarily used for ball detection, utilizing OpenCV Python for feedback. A blue ball (Marble) was detected using a code written in Visual Studio Code (VSCode). A serial connection was established between the Arduino and Python, which was used to send the desired angles to the motors. All necessary parts of the code were commented for clarity.

Arduino Code

The Arduino code is designed to receive the desired angle for each motor from the Python code. It uses the Arduino library to map the angles and then sends them to the motors. The angles are mapped from a range of -90 to 90 degrees to a range of 0 to 180 degrees. The servo1.write() and servo2.write() functions are used to send the mapped angles to the respective motors.

```
#include <Servo.h>

Servo servo1;
Servo servo2;

void setup() {
  servo1.attach(9);  // attaches the servo on pin 9 y axis
  servo2.attach(10); // attaches the servo on pin 10 x axis
  Serial.begin(9600); // starts the serial communication
}

void loop() {
  if (Serial.available()) {
    String data = Serial.readStringUntil('\n'); // read the incoming data until newline
    int separatorIndex = data.indexOf(','); // find the index of the comma separator

    int angle1 = data.substring(0, separatorIndex).toInt(); // extract the first angle
    int angle2 = data.substring(separatorIndex + 1).toInt(); // extract the second angle

      // Map the received angles from -90 to 90 to 0 to 180 degrees
      int mappedAngle1 = map(angle1, -90, 90, 0, 180);
      int mappedAngle2 = map(angle2, -90, 90, 0, 180);
```

```
        servo1.write(mappedAngle1);
    servo2.write(mappedAngle2);
  }
}
```

Python Code

The Python code is responsible for detecting the blue ball in the video feed and calculating its position and speed relative to the center of the frame. It also calculates the angles to be sent to the Arduino based on the ball's position and speed.

```
from math import floor
import cv2
import numpy as np
from time import time
import serial
import imutils
from math import trunc
# Define the serial port and baud rate
ser = serial.Serial('COM7', 9600)  # Change 'COM3' to your Arduino's serial port

# Define the lower and upper bounds for the white plate in HSV format
lower_white = np.array([0, 0, 135])
upper_white = np.array([106, 255, 205])

# Define the lower and upper bounds for the orange ball in HSV forma
lower_blue = np.array([89, 147, 0])
upper_blue = np.array([179, 255, 255])

ip_webcam_url = 'http://192.168.137.235:4747/video'
# Create a VideoCapture object to connect to the IP webcam.
cap = cv2.VideoCapture(ip_webcam_url)

roi_x = 261
roi_y = 88
roi_w = 353  # Define the initial width of the ROI
roi_h = 345  # Define the initial height of the ROI

plate_dimensions = None

setPoint_x = 0.0
setPoint_y = 0.0
speed_x = 0.0
speed_y = 0.0

# Define the desired width of the output frame
output_width = 640  # Adjust to your desired width
```

```
# Control Loop Properties
frequency = 100.0      # Frequency in Hz
period = 1.0 / frequency       # Time period between loop iterations
start_time = time()

fps = int(cap.get(5))
print("fps:", fps)

previous_ball_x = None
previous_ball_y = None
font_size = 0.5  # Adjust the font size as needed

angle1 = 0
angle2 = -3
ser.write(f"{angle1},{angle2}\n".encode())
while True:



    ret, frame = cap.read()
    if not ret:
        break
    cropped_frame1 = frame[roi_y:roi_y + roi_h, roi_x:roi_x + roi_w]
    cv2.imshow("Vi", imutils.rotate(cropped_frame1, -90))

    # Convert the frame to the HSV color space
    hsv = cv2.cvtColor(cropped_frame1, cv2.COLOR_BGR2HSV)

    # Create a mask to extract the white color
    white_mask = cv2.inRange(hsv, lower_white, upper_white)

    # Apply a series of morphological operations to remove noise for the mask
    white_mask = cv2.erode(white_mask, None, iterations=2)
    white_mask = cv2.dilate(white_mask, None, iterations=2)

    # Find contours for the white color
    white_contours, _ = cv2.findContours(white_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if plate_dimensions is None:
        for contour in white_contours:
            if cv2.contourArea(contour) > 500:  # Adjust the area threshold as needed
                epsilon = 0.03 * cv2.arcLength(contour, True)
                approx = cv2.approxPolyDP(contour, epsilon, True)

                if len(approx) == 4:
                    x, y, w, h = cv2.boundingRect(approx)
```

```
                plate_dimensions = (x, y, w, h)
                break

    if plate_dimensions is not None:
        x, y, w, h = plate_dimensions

        # Crop the frame to the dimensions of the white plate
        cropped_frame = cropped_frame1[y:y + h, x:x + w]
        cropped_frame = imutils.rotate(cropped_frame, -90)
        # Resize the cropped frame while maintaining the aspect ratio
        aspect_ratio = float(output_width) / w
        new_height = int(h * aspect_ratio)
        cropped_frame = cv2.resize(cropped_frame, (output_width, new_height))

        # Convert the cropped frame to the HSV color space
        hsv_cropped = cv2.cvtColor(cropped_frame, cv2.COLOR_BGR2HSV)

        # Create a mask to extract the orange ball within the cropped frame
        blue_mask = cv2.inRange(hsv_cropped, lower_blue ,upper_blue)

        # Find contours for the orange ball within the cropped frame
        blue_contours, _ = cv2.findContours(blue_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

        for blue_contour in blue_contours:
            if cv2.contourArea(blue_contour) > 100:  # Adjust the area threshold as needed
                (x_o, y_o), radius = cv2.minEnclosingCircle(blue_contour)
                center = (int(x_o), int(y_o))
                radius = int(radius)
                cv2.circle(cropped_frame, center, radius, (0, 0, 255), 2)

                # Calculate the coordinates of the ball's center within the cropped frame
                ball_x = int(x_o)
                ball_y = int(y_o)

                # Calculate the frame's center
                frame_center_x = cropped_frame.shape[1] // 2
                frame_center_y = cropped_frame.shape[0] // 2

                # Calculate the ball's position relative to the frame's center
                relative_xx = "{:.2f}".format(((ball_x - frame_center_x)*float(0.1 / w))) #
                relative_x = float(relative_xx)
                relative_yy = "{:.2f}".format(-((ball_y - frame_center_y)*float(0.1 / h))) # Note: Inverted Y-
axis
                relative_y = float(relative_yy)
                # Print the relative coordinates on the output frame
                cv2.putText(cropped_frame, "X: {}".format(relative_x), (10, 20),
cv2.FONT_HERSHEY_SIMPLEX, font_size, (0, 255, 0), 1)
```

```
            cv2.putText(cropped_frame, "Y: {}".format(relative_y), (10, 40),
cv2.FONT_HERSHEY_SIMPLEX, font_size, (0, 255, 0), 1)
            if previous_ball_x is not None and previous_ball_y is not None:
                delta_x = relative_x - previous_ball_x
                delta_y = relative_y - previous_ball_y

                speed_x = (delta_x / period)
                speed_y = (delta_y / period)

                # Print the speed with the smaller font
                cv2.putText(cropped_frame, "Speed X: {:.2f} m/s".format(speed_x), (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, font_size, (0, 255, 0), 1)
                cv2.putText(cropped_frame, "Speed Y: {:.2f} m/s".format(speed_y), (10, 80),
cv2.FONT_HERSHEY_SIMPLEX, font_size, (0, 255, 0), 1)

            previous_ball_x = relative_x
            previous_ball_y = relative_y

            k1 = -290.0
            k2 = -11.0

            angle1 = -2 + round(max(-5, min(-(k1*relative_y + k2*speed_y), 5)), 1)
            angle2 = 2 + round(max(-5, min(-(k1*relative_x + k2*speed_x), 5)), 1)
            print((k1*relative_y + k2*speed_y), (k1*relative_x + k2*speed_x), angle1, angle2)
            # Send angles to the Arduino
            ser.write(f"{angle1},{angle2}\n".encode())

        # Draw a green rectangle around the white plate
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        cv2.imshow("Cropped Video with Ball Detection", cropped_frame)
    while time() - start_time < period:

        pass

    start  time = time()

    if cv2.waitKey(1) & 0xFF == 27:  # Press 'Esc' to exit
        break

cap.release()
cv2.destroyAllWindows()
```

Debrief of Python Code

The Python code is primarily used for ball detection on a plate. A white foam is used to easily detect the plate and crop it from the frame. The code establishes connections with both the camera and Arduino. It then begins to find a white contour in the frame and crops it as the main frame for further processing.

In the main frame, the code looks for the ball's position along both axes. The speed of the ball is calculated based on its position and the frequency of the while loop, which runs at 100Hz. After calculating the position and speed of the ball (which are considered as the states of the system), the feedback gains calculated with MATLAB are applied to them.

A saturation limit is assumed, where the angles must be between -5 and 5 degrees. These angles are then sent to the motors. This ensures that the system operates within safe limits while tracking the ball.

4. Results and Discussion

The use of foam in the experimental setup served a dual purpose. Apart from aiding in the detection of the plate, it also increased the friction in the system. This increase in friction was instrumental in stopping the ball at the center of the plate.

As per the simulation results, the ball exhibited a fluctuating behavior. This behavior was also observed in the real system, indicating a good correlation between the simulation and the actual experiment. Despite the fluctuations, the ball was able to stop within a vicinity of a circle with a radius of 0.01 m.

While the system did not consistently stop the ball exactly at the center, the results obtained still represent a significant achievement. The correlation between the simulation results and the actual experiment indicates the validity of the mathematical modeling and control theory applied in this project.

Figure 10: Implemented Real-Time System

5. Conclusion

The system built for this project was unable to fully achieve the tracking goal. This has highlighted several areas for improvement in future iterations of the project. One of the main enhancements to be made is to change the motor types, so as to have control over the angle, speed, and acceleration of the motor and ensure smooth movement of the plate. This would allow for control over all of the system's states.

In addition, changing the control algorithm, such as using a PID controller for each of the axes, could help in achieving the desired result. However, one of the main purposes of this project was to implement a different controller to the system.

Overall, despite the challenges, this project was a great learning experience. It provided valuable insights into the complexities of designing and implementing a real-world control system. The journey that started with this project is worth continuing, with the aim of further improving the system and achieving the tracking goal in future iterations

6. References

[1] M. Nokhbeh, D. Khashabi, "Modelling and Control of Ball-Plate System, Final Project Report". Amirkabir University of Technology, 2011.

[2] L. Spacek, V. Bobal, J. Vojtesek, "Digital control of Ball & Plate model using LQ controller" 21st International Conference on Process Control (PC), June 6–9, 2017.

[3] H. Jafari, A. Rahimpour, M. Pourrahim, F. Hashemzadeh, "Linear Quadratic Gaussian Control for ball and plate system". Unknow Conference, July 2012.

[4] Ming Bai, Huiqiu Lu, Jintao Su and Yantao Tian "Motion Control of Ball and Plate System Using Supervisory Fuzzy Controller" 6th World Congress on Intelegent Control and Automation, June, 2006

[5] Hongrui Wang, Yantao Tian, Le Ding, Qing Gu and Feng Guo "Output Regulation of the Ball and Plate System with a Nonlinear Velocity Observer" 7th World Congress on Intelegent Control and Automation, june, 2008

[6] Wang Hongrui, Tian Yantao, Fu Siyan, Sui Zhen" Nonlinear Control for Output Regulation of Ball and Plate System" Proceedings of the 27th Chinese Control Conference, July 16-18, 2008, Kunming, Yunnan, China

[7] Xiucheng Dong, Zhang Zhang, Chao Chen "Applying Genetic Algorithm to On-Line Updated PID Neural Network Controllers for Ball and Plate System" 4th International Conference on Innovative Computing Information and Control, 2009

[8] D. Yuan, Z. Zhang, "Modelling and control scheme of the ball–plate trajectory-tracking pneumatic system with a touch screen and a rotary cylinder" International Journal of Mechanical Engineering and Robotics Research, Volume 4, Issue 4, April 2010

[9] Chi-Cheng Cheng, Chen-Hsun Tsai, "Visual Servo Control for Balancing a Ball-Plate System" IET Control Theory & Applications, 2016

[10] H.-N. Nguyen, S. Olaru, P.-O. Gutman, M. Hovd, "Improved vertex control for a ball and plate system" IFAC Proceedings Volumes, Volume 45, Issue 13, 2012, Pages 400-405

[11] Xingzhe Fan, Naiyao Zhang, Shujie Teng, "Trajectory planning and tracking of ball and plate system using hierarchical fuzzy control scheme" Fuzzy Sets and Systems, Volume 144, Issue 2, June, 2004, Pages 297-312

[12] Ming-Tzu Ho, Yusie Rizal, Li-Ming Chu, "Visual Servoing Tracking Control of a Ball and Plate System: Design, Implementation and Experimental Validation" International Journal of Advanced Robotic Systems, April, 2013

[13] https://www.youtube.com/wtch?v=9Cm7HZLUdoY

[14] https://www.youtube.com/watch?v=sFtfWoAzHwg

[15] https://www.youtube.com/watch?v=kAaYaZcpbLo

[16] https://www.youtube.com/watch?v=v4F-cGDGiEw

[17] https://www.youtube.com/watch?v=KCS_gPH867E

[18] https://www.youtube.com/watch?v=0BDvbljP4Yk

[19] https://www.youtube.com/watch?v=gO4dPVd7bw8

[20] https://www.youtube.com/watch?v=uERF6D37E_o

[21] https://www.youtube.com/watch?v=0DqcnHE6r9M