MARMARA UNIVERSITY
FACULTY OF ENGINEERING

POSİTİON CONTROL OF THE BALL ROLLİNG ON A FLAT PLATFORM

MUHAMMED SELMAN TURUNÇ

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Prof. Dr. ERTUĞRUL TAÇGIN

ISTANBUL, 2021

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

POSİTİON CONTROL OF THE BALL ROLLİNG ON A FLAT PLATFORM

MUHAMMED SELMAN TURUNÇ

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Prof. Dr. ERTUĞRUL TAÇGIN

ISTANBUL, 2021

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

Position Control of The Ball Rolling on A Flat Platform

by

Muhammed Selman TURUNÇ

August 2, 2021, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE AT MARMARA UNIVERSITY

The author(s) hereby grant(s) to Marmara University permission to reproduce and to distribute publicly paper and electronic copies of this document in whole or in part and declare that the prepared document does not in anyway include copying of previous work on the subject or the use of ideas, concepts, words, or structures regarding the subject without appropriate acknowledgement of the source material.

Signature of Author(s) Muhammed Selman TURUNÇ .....

Department of Mechanical Engineering

Certified By .....

Project Supervisor, Department of Mechanical Engineering

Accepted By .....

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, we would like to thank our supervisor Prof. Dr. ERTUĞRUL TAÇGIN, for the valuable guidance and advice on preparing this preliminary report and giving us moral and material support.

August 2, 2021

CONTENTS

ABSTRACT

In this study, the motion of the ball on the plate is controlled by both PID and fuzzy logic controllers. The position of the ball on the plate is obtained with the help of the touch panel. Position data obtained is input to the controller, the controller calculates the required servo angles, and the microcontroller sends the required angle to the servo motors in the form of PWM signal. The servo motors achieve the required angle and balance the system.

A mathematical model was created for the PID controller, but it was not used because the delays in the servo motor and Arduino could not be modeled. An iterative method was used to determine the PID parameters. We used the Arduino libraries to implement the fuzzy logic controller. Various studies were examined and a rule base table for fuzzy logic controller was prepared based on these studies. The fuzzy logic controller we designed did not work well.

Key Words: Ball & Plate, Control Theory, Balancing, Fuzzy Logic Controller, PID, Arduino, Servo, Touchscreen

SYMBOLS

a_a : Absolute acceleration of ball

r : Radius of the ball

ω : Angular velocity of the ball

α : The tilt angle of the platform

x_p : Position of ball

I_b : Mass moment of inertia for the ball

β : Angle of the ball relative to its initial position in the center of the platform

F_r : Force acting on the ball from the platform by friction

N : Normal force acting on the ball from the platform

m_b : Mass of the ball

g : Gravitational acceleration

$\mathrm{u}(t)$: Output signal

$\mathrm{e}(t)$: Error value

K_P : Proportional gain

K_I : Integral gain

K_D : Derivative gain

H : Transfer function of the system

ABBREVIATIONS

PID : Proportional Integral Derivative

DOF : Degree of Freedom

PCB : Printed Circuit Board

P : Proportional

PI : Proportional Integral

PD : Proportional Derivative

FLC : Fuzzy Logic Control

MF : Membership Function

PLA : Polylactic Acid

CAD : Computer-Aided Design

3D : Three Dimensional

USB : Universal Serial Bus

ITO : Indium Tin Oxide

PWM : Pulse Width Modulation

LED : Light Emitting Diode

IDE : Integrated Development Environment

IoT : Internet of Things

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Control theory deals with the control of dynamical systems in engineered processes and machines. The goal is developing a model or algorithm to stabilize an unstable system or modulating the desired response and outcome of a given system, while minimizing any delay, overshoot, or steady-state error and ensuring a level of control stability. With the developing technology, control systems are of great importance in the engineering subject. Therefore, the purpose of this thesis is to show how theory is applied in a physical model and how changes and defects in the application cause differences in physical response.

Balancing systems are one of the most challenging systems in control field. Inverted pendulum, ball on beam is some of the classic examples of balancing systems. Ball and plate system is an enhanced version of ball and beam system. In this system, the plate is manipulated in two perpendicular directions and the ball is balanced at a specific position on the plate. This system is widely used because of its simplicity to understand as a system and provides the opportunity to analyze the control techniques.

1.1. Problem Statement

The problem is balancing a ball in a specific position on the plate by manipulating the plate angles. A dynamic system should be built to stabilize the ball at the desired point on the plate by considering cost, performance, safety and functionality constraints.

1.2. Objectives

The main objectives of this study are to

- Design proportional integral derivative (PID) controller to control ball position on plate.
- Implement the real time response of ball and plate system.
- Study the effects of PID controller parameters.
- Design fuzzy logic controller to control ball position on plate.

2. CONCEPTUAL DESIGN

Conceptual design is an early phase of the design process, in which the broad outlines of function and form of something are articulated. In this section, we will determine the general outlines of our system.

2.1. Plant Selection

There are different design approaches for the ball and plate system, these are 2 DOF, 3 DOF and 6 DOF systems. Apart from these, although there is a 1 degree of freedom design approach, this system is no longer a ball and plate system, but a ball and beam system. Among these design forms, the form with 2 degrees of freedom was chosen for this study. This is the simplest compared to other forms and consists of only 2 actuators. 2 DOF system has the ability to decouple these two activators that means each of these two activators responsible to tilt this plate in one axis of rotation. If I need to explain why the systems with 3 DOF and 6 DOF are not chosen, these are over activated systems. More than 2 degrees of freedom are not needed to fully achieve the goal. These systems do not only give the position of the ball, they can give the specific height of the ball, 6 DOF systems can also give the rotation. However, this information are not very important and is often ignored.

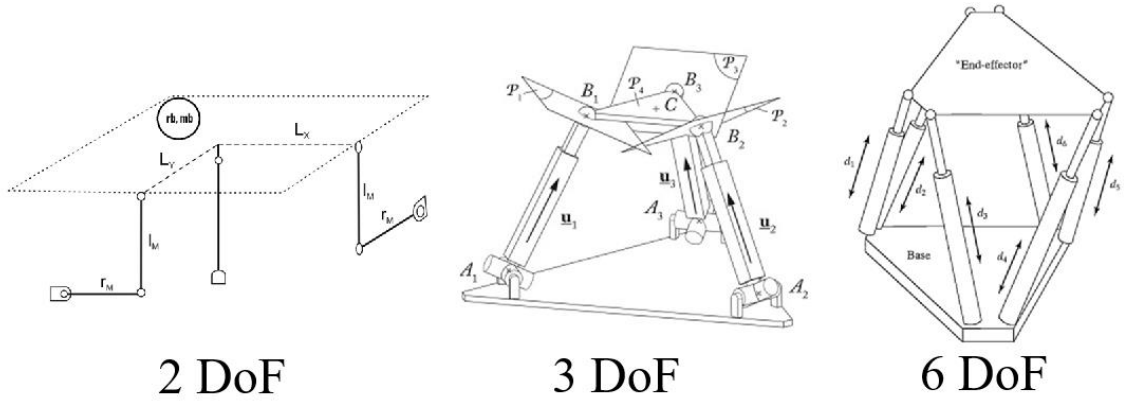

Figure 2.1 Different design forms for ball and plate system

2.2. Sensor Selection

We need a sensor to read the ball position. There are 2 sensors available that we can use to do this, the camera and the resistive touch panel. Resistive touchscreen is very hard to handle sometimes because they are very noisy and some of them do not work under a certain threshold. So, for a resistive touchscreen to give you information about the ball's position, the ball must apply a certain amount of pressure to these touchscreens, and sometimes this threshold is large. Therefore, you may need to use a heavy ball to read the position on this touch screen. But when the plate rotates, the contact between the ball and the plate decreases and this pressure decreases and the screen may not detect the ball, which is undesirable. On the other hand, the camera requires image processing. Although it has some disadvantages, resistive touch screen is preferred because it is easier in terms of application.

2.3. Actuator Selection

In order to control the system, we need to be able to give a certain angle to the plate momentarily, and we need an activator to do this. We can use step motor or servo motor. We can use step motor or servo motor; both are suitable for this job. But for using step motor we need motor driver, and it requires encoder to get feedback. On the other hand, in the servo motor, these are inside the motor, and it is easier to use. For this reason, the servo motor is preferred.

2.4. Processing Unit Selection

There are many options for the processing unit, such as Arduino, Raspberry Pi and custom-made PCB. Arduino was preferred for this study.

3. THEORY

3.1. Mathematical Model

The model of the dynamical system is derived from the laws of physics and expressed as one or multiple differential equations. These equations can be acquired by either exercising Newtonian mechanics or Euler-Lagrangian mechanics to the system setup. As of this thesis Newtonian mechanics will be used. However, the Euler-Lagrangian method arrives to the same conclusion. A accurate model of the system allows for better system analysis and prerequisites for control design. It is therefore a critical aspect in the process of stabilizing an unstable system in the way that this thesis will demonstrate.

3.1.1. Simplifications and assumption

In order to achieve the equations of motion for a dynamical system consisting of a

ball on a platform the following simplifications has to be assumed:

- The ball is rolling and not slipping on the platform.
- No friction is considered.
- The geometry of the ball is perfectly spherical and homogeneous.
- The ball has no translation upwards relative to the platform.

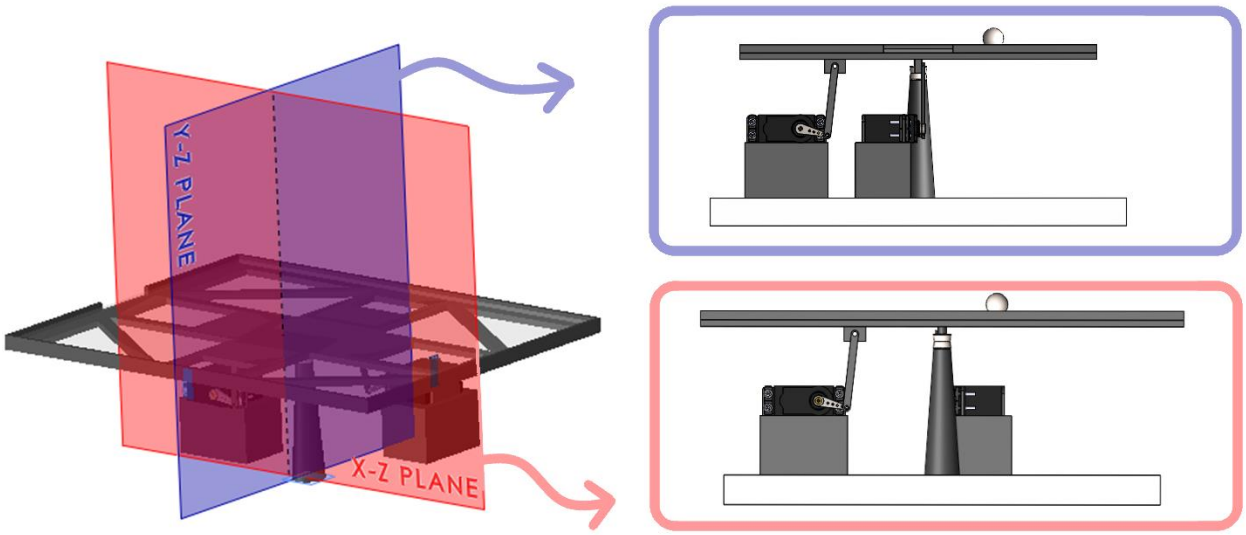

Figure 3.1 Representative system view and planes.

The three-dimensional ball on platform system will from here on be separated into a pair of two-dimensional ball and plate systems as shown in Figure 3.1, consisting of the same equations but different variables. The red plane represents the x-z plane for the system and the blue plane represents the y-z plane for the system. The equations will be derived within the x-z plane but later translated into its equivalent equation within the y-z plane.

Equations of motion

The equation for absolute acceleration of the ball is given by:

$$a_a = \dot{\omega} \times \boldsymbol{r} + \omega \times (\omega \times r) + 2\omega \times \boldsymbol{v_{rel}} + \boldsymbol{a_{rel}} \qquad \text{(Eq 3.1)}$$

The above can be rewritten as follows for one of the two-dimensional ball and plate systems.

$$a_1 = \ddot{\alpha}_1 e_{k1} \times x_p e_{i1} + \dot{\alpha}_1 e_{k1}\, x \left(\dot{\alpha}_1 e_{k1} \times x_p e_{i1}\right) + 2\dot{\alpha}_1 e_{k1} \times \dot{x}_p e_{i1} + \ddot{x}_p e_{i1} \qquad \text{(Eq 3.2)}$$

As seen in Figure 3.2 is the inclination of the platform in the first two-dimensional system and x_p is the position of the ball relative to the coordinate system e_{i1}, e_{j1}, e_{k1} fixed to the platform in the current view. After simplifications and vector multiplication the following equation is derived for the x-z system as seen below.

$$a_1 = \left(\ddot{x}_p - x_p \dot{\alpha}_1{}^2\right)e_{i1} + \left(x_p \ddot{\alpha}_1 + 2\dot{\alpha}_1 \dot{x}_p\right)e_{j1} \qquad \text{(Eq 3.3)}$$

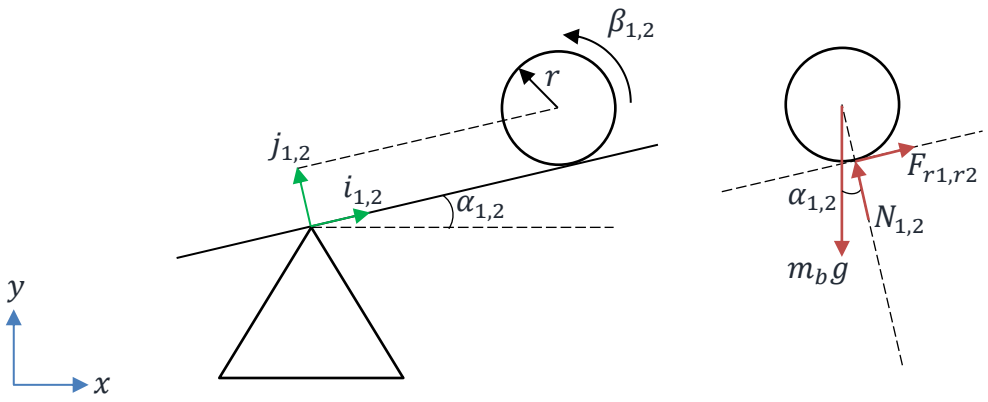

Figure 3.2 Two-dimensional representation of the system and free body diagram.

From equilibrium of torques in the free body diagram seen in Figure 3.2 it is possible to derive the remaining forces on the ball.

$$I_b \ddot{\beta}_1 = F_{r1} r \qquad \text{(Eq 3.4)}$$

Where I_b is the mass moment of inertia for the ball, β_1 is the angle of the ball relative to its initial position in the center of the platform and r is the radius of the ball. F_r is the force from the platform acting on the ball with fully developed friction. With regards to the assumption of no slip on the platform the relative angle β can be defined by the position as follows.

$$\beta_1 = -\frac{x_p}{r} \qquad \text{(Eq 3.5)}$$

In order to solve equation (Eq 3.4) for F_r, the second time derivative of equation (Eq 3.5) is combined with first equation. Resulting in equation (Eq 3.6) as stated below.

$$F_r = -\frac{I_b x_p}{r^2} \qquad \text{(Eq 3.6)}$$

The equilibrium of forces exerted on and by the ball parallel to the platform given by acceleration in equation (Eq 3.3) and the force in equation (Eq 3.6) results in the two principal equations of motion of the dynamic system.

$$\left(\frac{I_b}{r^2} + m_b\right)\ddot{x}_p + m_b g \sin\alpha_1 - m_b x_p {\dot{\alpha}_1}^2 = 0 \qquad \text{(Eq 3.7)}$$

$$\left(\frac{I_b}{r^2} + m_b\right)\ddot{y}_p + m_b g \sin\alpha_2 - m_b y_p {\dot{\alpha}_2}^2 = 0 \qquad \text{(Eq 3.8)}$$

By rearranging the equations of motion, the following equations suitable for Laplace transformation is achieved.

$$\ddot{x}_p = \frac{m_b {r_b}^2\left(x_p {\dot{\alpha}_1}^2 - g\sin\alpha_1\right)}{m_b {r_b}^2 + I_b} \qquad \text{(Eq 3.9)}$$

$$\ddot{y}_p = \frac{m_b {r_b}^2\left(x_p {\dot{\alpha}_2}^2 - g\sin\alpha_2\right)}{m_b {r_b}^2 + I_b} \qquad \text{(Eq 3.10)}$$

3.1.2. Linearization

The dynamical ball on platform system can thereafter be described by two nonlinear differential equations, one for each axis. Dealing with nonlinear systems is beyond the scope of this thesis and thus, the system needs to be linearized around a working point henceforth. The preferred and desired working point of the apparatus will be based in the center of the platform. Hence, equation (Eq 3.9) and equation (Eq 3.10) are linearized around $x_p = 0$, $\alpha_1 = 0$, $y_p = 0$, $\alpha_2 = 0$. The following linearized equations are valid for small deviations in α_1 and α_2.

$$\ddot{x} = \frac{m_b g \alpha_1 r^2}{m_b {r_b}^2 + I_b} \qquad \text{(Eq 3.11)}$$

$$\ddot{y} = \frac{m_b g \alpha_1 r^2}{m_b {r_b}^2 + I_b} \qquad \text{(Eq 3.12)}$$

Further, if the complete expression of the moment of inertia for the ball, I_b is inserted, the linearized equation can be written as follows below. Noteworthy is that the theoretical system is independent of the mass as well as of the radius of the ball.

$$I_{b,Solid} = \frac{2}{5} mr^2 \quad \text{(Eq 3.13)}$$

$$I_{b,Hollow} = \frac{2}{3} mr^2 \quad \text{(Eq 3.14)}$$

$$\ddot{x}_{\,Solid} = \frac{5}{7} g\alpha_1 \quad \text{(Eq 3.15)}$$

$$\ddot{y}_{\,Solid} = \frac{5}{7} g\alpha_2 \quad \text{(Eq 3.16)}$$

$$\ddot{x}_{\,Hollow} = \frac{3}{5} g\alpha_1 \quad \text{(Eq 3.17)}$$

$$\ddot{y}_{\,Hollow} = \frac{3}{5} g\alpha_2 \quad \text{(Eq 3.18)}$$

3.1.3. Laplace Transformation

Equations (Eq 3.15),(Eq 3.16),(Eq 3.17),(Eq 3.18) are defining the dynamics of the system within the time domain. To allow controller design of the system, these equations need to be translated into the frequency domain using Laplace transformation. The resulting Equations, now within the frequency domain as seen below.

for solid sphere

$$s^2 X = \frac{5}{7} g A_1 \quad \text{(Eq 3.19)}$$

$$s^2 Y = \frac{5}{7} g A_2 \quad \text{(Eq 3.20)}$$

for hollow sphere

$$s^2 X = \frac{3}{5} g A_1 \quad \text{(Eq 3.21)}$$

$$s^2 Y = \frac{3}{5} g A_2 \quad \text{(Eq 3.22)}$$

3.2.Control Theory

3.2.1. PID Controller

The PID, proportional-integral-derivative, controller is a control function which applies corrections automatically given a feedback loop mechanism. Through the PID, modulated systems requiring continuity in the control can be monitored and stabilized to desired response. The value evaluated in the controller is the error e(t). Where the error value is given by the offset from a setpoint. However, any combinations of respective parts of the controller may be implemented e.g., P, PI, PD. The regulated output signal, discussed in detail below, is given by equation:

$$\mathrm{u}(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \qquad \text{(Eq 3.23)}$$

3.2.2. Proportional Controller

The purpose of the proportional controller is to reduce the steady state error through increasing the proportional gain of the system. Through introduction of a constant K_P , the proportional gain, the proportional response is adjusted accordingly i.e., the steady state error behaves inversely of the proportional gain. The proportional term or output signal is further given by equation:

$$u_P(t) = K_P e(t) \qquad \text{(Eq 3.24)}$$

Nevertheless, if the gain is increased too much it may alter the output causing instability in the system. Opposite, if a gain becomes too small it could possibly cause issues handling external or internal disturbances.

3.2.3. Integral Controller

The integral controller is proportionate to the, desirably, finite duration time of the error and the magnitude. The accumulated offset caused previously in the system is corrected by the sum of errors over the finite time. Thus, the steady state error is eliminated. However, K_I , the integral control, could cause the response of the system to become worse e.g., crippling the system with regards to response causing transient or oscillatory behavior. The output signal from the integral term is defined as equation:

$$u_I(t) = K_I \int_0^t e(\tau) d\tau \qquad \text{(Eq 3.25)}$$

3.2.4. Derivative Controller

The derivative controller is determined through the calculation of the error's response slope over time. The slope of the error is then multiplied with K_D, the derivative gain. The derivative term is defined in equation:

$$u_D(t) = K_D \frac{de(t)}{dt} \qquad \text{(Eq 3.26)}$$

This term does alter the rate of change of the output, slowing it down. K_D will however further reduce overshoot, increase stability and enhance the transient system response.

3.3. Fuzzy Logic Control

The main idea of FLC is very well explained by Kickert and Mamdani as:

"The basic idea behind this approach was to incorporate the "experience" of a human process operator in the design of controller. From the set of linguistic rules which describe the operator's control strategy a control algorithm is constructed where the words are defined as fuzzy sets. The main advantage of this approach seem to be the possibility of implementing rule of thumb experience, intuition, heuristics and the fact that it does not need a model of the process."

The input of a FLC is always crisp, which is fuzzified in a fuzzification process based on the rules in the rule base. After the fuzzy decisions are made by inference, the output of the FLC is converted into a crisp value. This is called the defuzzification process.

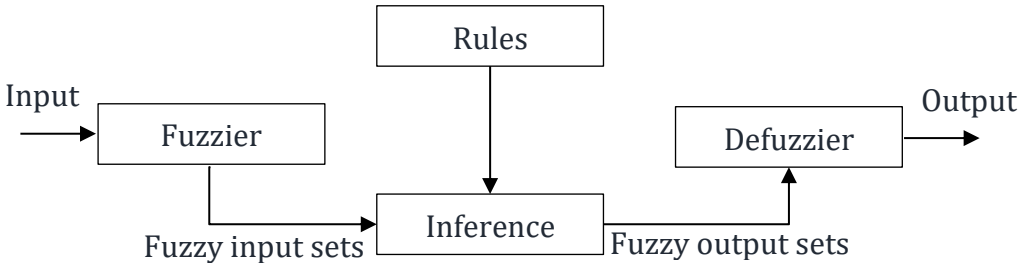

Figure 3.3 Fuzzy controller block diagram.

A FLC can be divided into four main sub-groups: fuzzification, inference, rule base, and defuzzification. The fuzzy logic controller block diagram is given in the Figure 3.3.

3.3.1. Fuzzification

Fuzzification is the process of converting a crisp input value into a fuzzy value, performed by using the information in the knowledge base. Gaussian, triangular and trapezoidal MFs are most commonly used in the fuzzification process. These types of MFs can be easily implemented. The MFs are defined mathematically with several parameters. To fine tune the performance of a FLC, these parameters or the shape of the MFs can be adjusted.

3.3.2. Rule Base

In this step, expert knowledge is formulated as rules. The rule base contains the rules to be used in decision making. These rules are often based on personal experience and intuition.

A rule consists of two main parts: an antecedent block (between the If and Then) and a consequent block (following Then).

$$If\ (antecedent)\ Then\ (consequent) \qquad \text{(Eq 3.27)}$$

3.3.3. Inference

In this process, fuzzy decisions are generated based on the rules in the rule base. In this process, each rule is evaluated separately and then a decision is made for each rule. The result is a set of fuzzy decisions. Logical operators, such as “AND”, “OR” and “NOT” define how the fuzzy variables are combined.

3.3.4. Defuzzification

The final step is a defuzzification process, where the fuzzy output is translated into a single crisp value by the degree of membership values, as in the fuzzification process. Defuzzification is a reverse transformation compared to the fuzzification process, as this process converts the fuzzy output into crisp values that can be applied to the system.

There are several heuristic defuzzification methods. For example, some methods produce an integral output considering all the elements of the resulting fuzzy set with their corresponding weights. One of the widely used methods is the center-of-area method, which takes the centroid of the fuzzy set.

4. SYSTEM DESIGN

In this section, the system will be designed to acquire the desired function of stabilizing a ball on a flat surface with using the selections we made in section 2.

4.1. Requirements

The development of the system has been constrained by a set of requirements, both in terms of mechanical construction and performance. These requirements are necessary for system design.

- Allow the platform to tilt 15 degrees in each axis.
- Minimize rotational play around the z-axis of the platform.
- Steady state error $<$ 6 mm

4.2. Geometrical Analysis

A servo motor is connected to each axis as shown in Figure 4.1 to tilt the platform. Here point A is the center of rotation of the platform and point C is the center of rotation of the motor, points B and D represent the connections. L_1 is the distance between the platform connections. L_2 and L_3 are the lengths of the connecting rods. h and b are the vertical and horizontal distances between the motor connection and the platform connection. All dimensions are given in the Table 4.1.

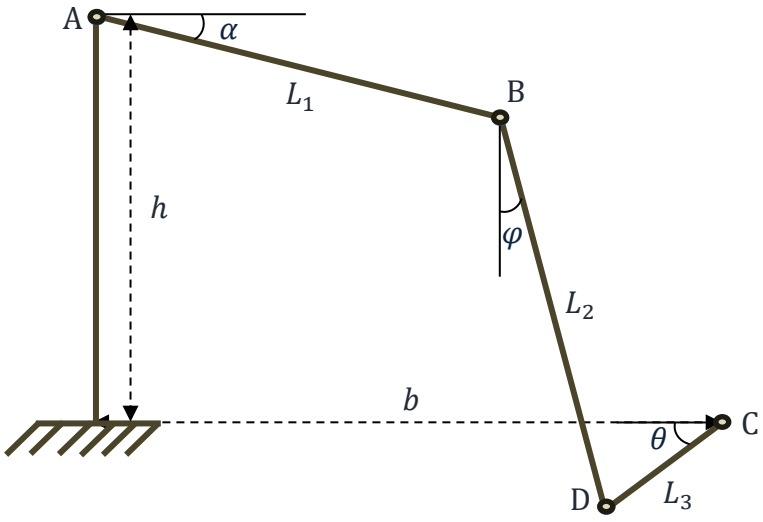

Figure 4.1 Geometry of platform

Table 4.1 System dimensions

L_1	L_2	L_3	h	b
60 mm	60 mm	31 mm	60 mm	90 mm

Using geometry and vector relations, the equations (Eq 4.1) and (Eq 4.2) can be obtained.

$$\overrightarrow{CD} + \overrightarrow{DB} = \overrightarrow{CA} + \overrightarrow{AB} \quad \text{(Eq 4.1)}$$

$$\begin{bmatrix} -L_3 * \sin\theta \\ -L_3 * \cos\theta \end{bmatrix} + \begin{bmatrix} L_2 * \cos\varphi \\ -L_2 * \sin\varphi \end{bmatrix} = \begin{bmatrix} h \\ b \end{bmatrix} + \begin{bmatrix} -L_1 * \sin\alpha \\ -L_1 * \cos\alpha \end{bmatrix} \quad \text{(Eq 4.2)}$$

Now if the assumption of a small angle changes are considered,

$$\sin\theta \approx \theta \quad \text{(Eq 4.3)}$$

$$\cos\theta \approx 1 \quad \text{(Eq 4.4)}$$

The following relationship is obtained.

$$\alpha = \frac{L_1}{L_3}\theta \quad \text{(Eq 4.5)}$$

The equations (Eq 4.2) and (Eq 4.5) was solved using MATLAB R2020b for the servo arm angle between $\theta = -30°$ and $\theta = 30°$. Figure 4.2 shows the differences between an analytical relationship and a linearized relationship based on the geometry of the system.

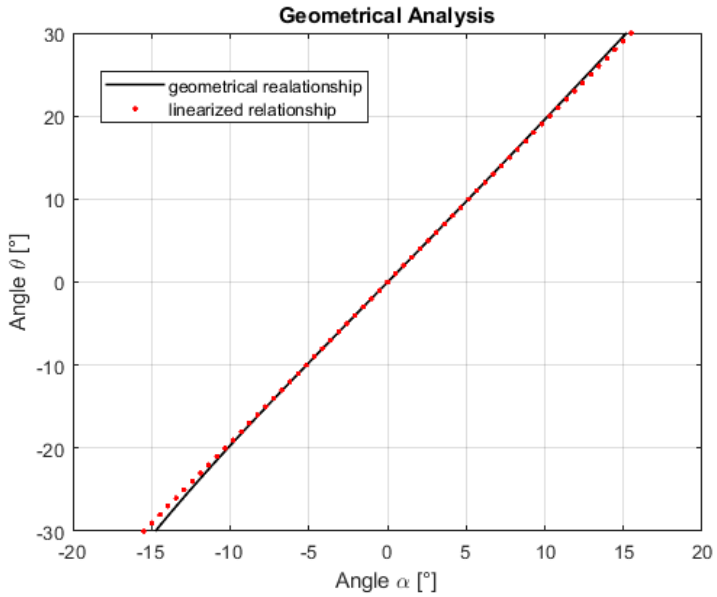

Figure 4.2 Relation between platform inclination and servo arm angle

4.3. Mechanical Design

A plate was designed to carry the screen. PLA material was preferred for lightness. The CAD view of the part is shown in Figure 4.3.

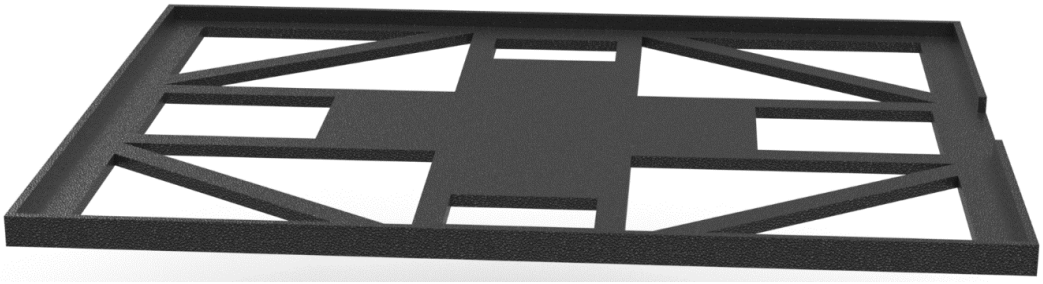

Figure 4.3 Base Plate CAD view.

Appropriate options were selected and purchased according to the lengths calculated in the geometric analysis for the servo arm and connecting rod. The selected connecting rod and servo arm are shown in Figure 4.4.

Figure 4.4 Connecting rod and servo arm.

In order to fix the servo motors, a servo holder was designed and produced with a 3D printer, the CAD view is shown in the Figure 4.5.

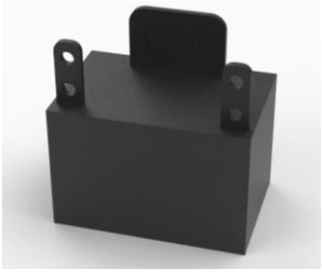

Figure 4.5 Servo holder CAD view

A universal joint connection is used between the plate and the pivot. The universal joint used is shown in Figure 4.6.

Figure 4.6 Universal Joint

Rod end bearing connection is used between motor arms and connecting rods. A steel ball with a diameter of 2.2 cm and a weight of 42 grams was used for this study.

For the servo motor selection, it was assumed that the system was carried by a single servo motor in the critical situation, the safety factor was used as 2 and the factor of 10 used to ensure operation in continuous region. The required torque is calculated as approximately $10.94\ kg.cm$. MG996R servo motor providing 6V sufficient torque is preferred, the specifications of the motor are in Table 4.2 and motor is shown in the Figure 4.7.

Table 4.2 MG996R Specifications

MG996R	4.8 V	6.0 V
Speed ($in\ second/60°$)	0.17	0.14
Torque ($kg.cm$)	9.4	11

Figure 4.7 Tower Pro MG996R Servo Motor

4.4. Electrical Design

Figure 4.8 shows the electrical circuit of the system and the wiring of the related components. The Arduino and 5 wire resistive touchscreen are powered through USB and the servo motors from an external power source stepped up to 6V. The list of components is given in Table 4.3.

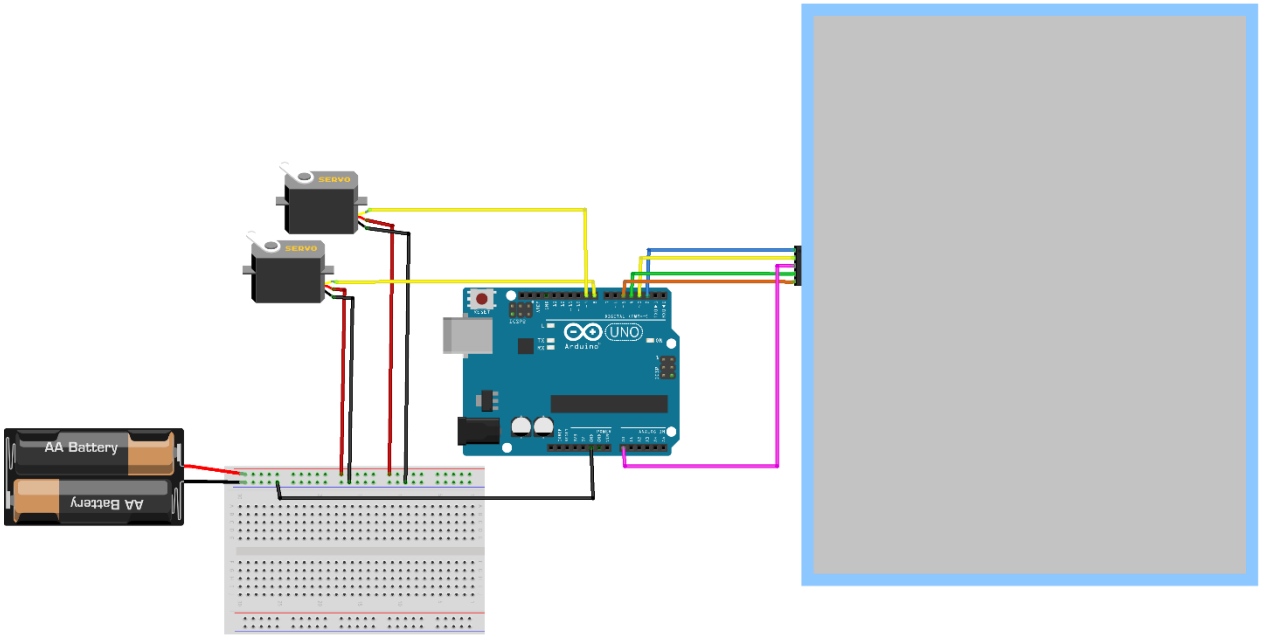

Figure 4.8 Electrical circuit and related components.

Table 4.3 List of components.

#	Component
1	Arduino UNO R3 Clone
2	Tower Pro MG996R Servo Motor
1	Green Touch GT-5W-15.0A-1 15" Touch Screen
2	6V Battery

4.5. System Components

4.5.1. Touch Screen

Touch screen is a device that converts a physical touch into an electrical signal. There are many types of touch screens, but we will examine resistive touch screens, these screens will be beneficial for us both in terms of cost and in terms of working principle because it sense the pressure on the screen.

4.5.1.1.Resistive Touch Screens

A resistive touchscreen panel comprises several thin layers, the most important of which are two transparent electrically resistive layers facing each other with a thin gap between. The top layer (that which is touched) has a coating on the underside surface; just beneath it is a similar resistive layer on top of its substrate. One layer has conductive connections along its sides, the other along top and bottom. A voltage is applied to one layer and sensed by the other. When an object, such as a fingertip or stylus tip, presses down onto the outer surface, the two layers touch to become connected at that point. The panel then behaves as a pair of voltage dividers, one axis at a time. By rapidly switching between each layer, the position of pressure on the screen can be detected.

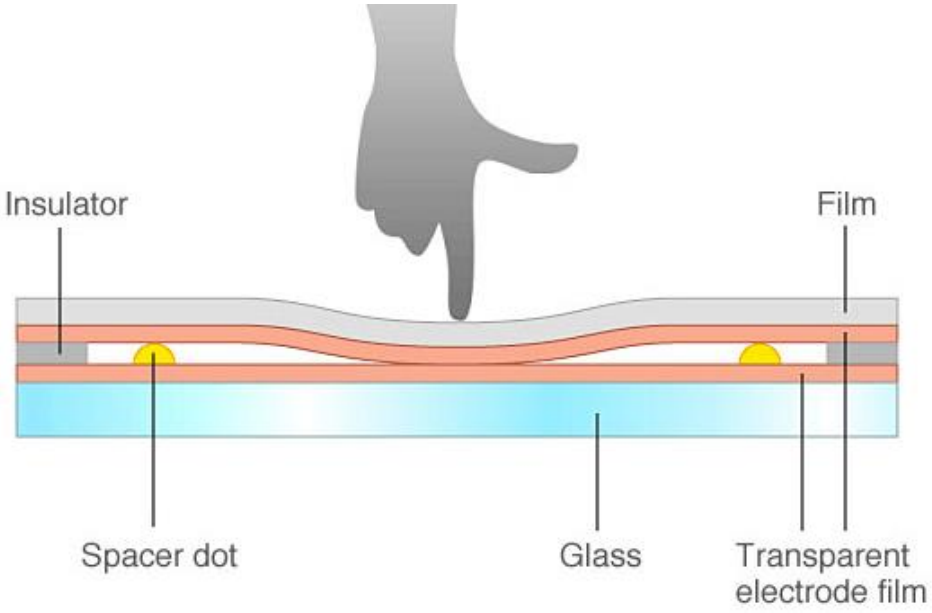

Figure 4.9 Structure of resistive touchscreen.

4.5.1.2. Four-Wire Resistive

Four-wire resistive technology is the simplest to understand and manufacture. It uses both the upper and lower layers in the touchscreen "sandwich" to determine the X and Y coordinates. Typically constructed with uniform resistive coatings of indium tin oxide (ITO on the inner sides of the layers and silver buss bars along the edges, the combination sets up lines of equal potential in both X and Y.

In the illustration in Figure 4.10, the controller first applies 5V to the back layer. Upon touch, it probes the analog voltage with the coversheet, reading 2.5V, which represents a left-right position or X axis. It then flips the process, applying 5V to the coversheet, and probes from the back layer to calculate an up-down position or Y axis. At any time, only three of the four wires are in use (5V, ground, probe).

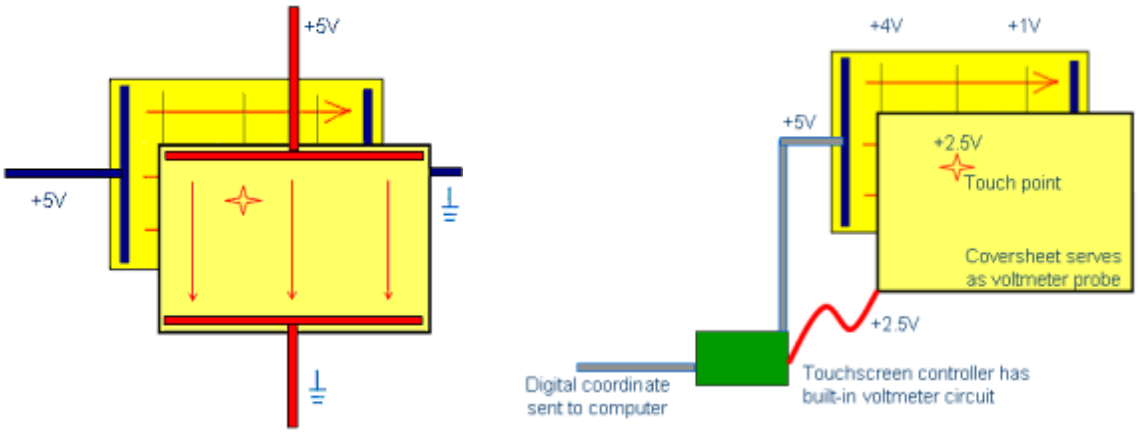

Figure 4.10 Illustration of the working principle of the 4-wire resistive touch screen.

The primary drawback of four-wire technology is that one coordinate axis (usually the Y axis), uses the outer layer, the flexible coversheet, as a uniform voltage gradient. The constant flexing that occurs on the outer coversheet with use will eventually cause microscopic cracks in the ITO coating, changing its electrical characteristics (resistance), degrading the linearity and accuracy of this axis.

4.5.1.3. Five-Wire Resistive

In the five-wire design, one wire goes to the coversheet (E) which serves as the voltage probe for X and Y. Four wires go to corners of the back glass layer (A, B, C, and D). The controller first applies 5V to corners A and B and grounds C and D, causing voltage to flow uniformly across the screen from the top to the bottom. Upon touch, it reads the Y voltage from the coversheet at E. Then the controller applies 5V to corners A and C and grounds B and D and reads the X voltage from E again.

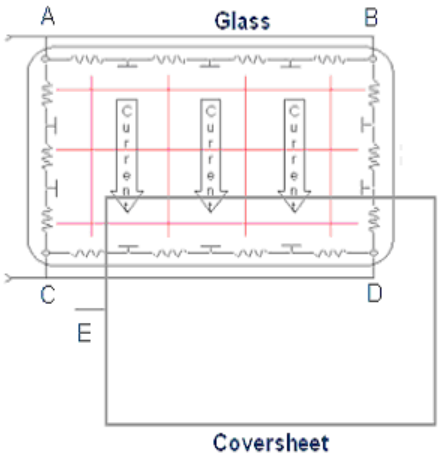

Figure 4.11 Illustration of the working principle of the 5-wire resistive touch screen.

So, a five-wire touchscreen uses the stable bottom layer for both X- and Y-axis measurements. The flexible coversheet acts only as a voltage-measuring probe. This means the touchscreen continues working properly even with non-uniformity in the coversheet's conductive coating. The result is an accurate, durable and more reliable touchscreen over four-wire design.

4.5.2. Servo motors

Also called servos, they are actuation devices for the precise control of speed, torque and position. They have a better performance and precision when compared to actuations based on frequency converters, since these do not offer position control and have low effectiveness at low speeds.

A servo motor is a device that contains an encoder which converts the mechanical motion (turns of the shaft) into digital pulses interpreted by a motion controller. It also contains a driver; and in conjunction, they make up a circuit that governs the position, torque and speed.

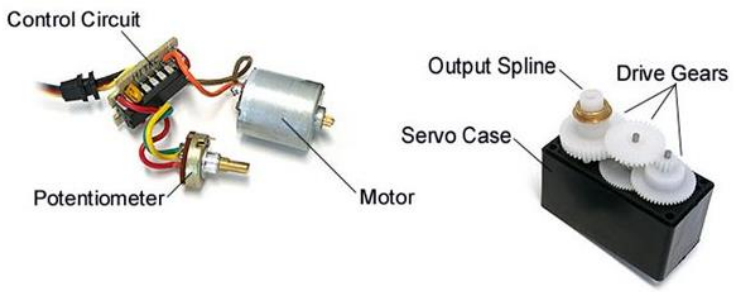

Figure 4.12 Parts of a servo motor.

4.5.2.1. Parts of a servo motor

- An electric motor: That is in charge of generating the motion through its shaft.
- A control system: This system allows for control over the motor's motion by sending electric pulses.
- A drive system: It is formed by gears which may increase or decrease the speed and torque.
- A potentiometer: It is connected to the central shaft and informs at all times the angle in which the motor's shaft is positioned.

4.5.2.2. How is the servo controlled?

Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise and counterclockwise direction. The PWM sent to the motor determines position of the shaft and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counterclockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position.

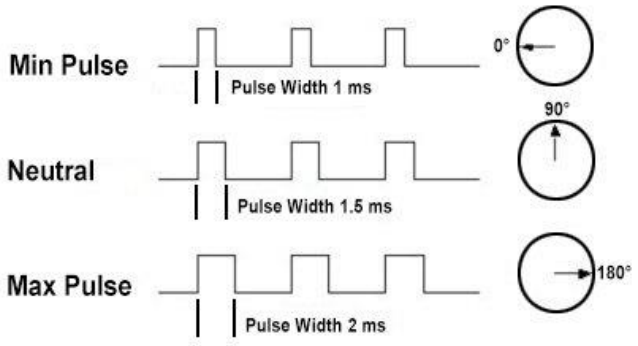

Figure 4.13 Variable Pulse width control servo position.

When these servos are commanded to move, they will move to the position and hold that position. If an external force pushes against the servo while the servo is holding a position, the servo will resist from moving out of that position. The maximum amount of force the servo can exert is called the torque rating of the servo. Servos will not hold their position forever though; the position pulse must be repeated to instruct the servo to stay in position.

4.5.3. Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

4.6. PID Control Loop

The user firstly defines a setpoint. A difference between the setpoint and the actual ball position(x,y coordinates read from the touch panel) results in the error. Then, the microcontroller calculates the desired servo angle of the system according to the error to reach the setpoint and sends the command to the servos in the form of PWM signal. The slope of the plates is geometrically related to the angle of the servos, and the change in angle affects as an input to the dynamics of the system. The ball position obtained is read by the touch panel and it is fed backwards, forming a closed loop. The block diagram in Figure 4.14 shows the hardware structure and control loop.

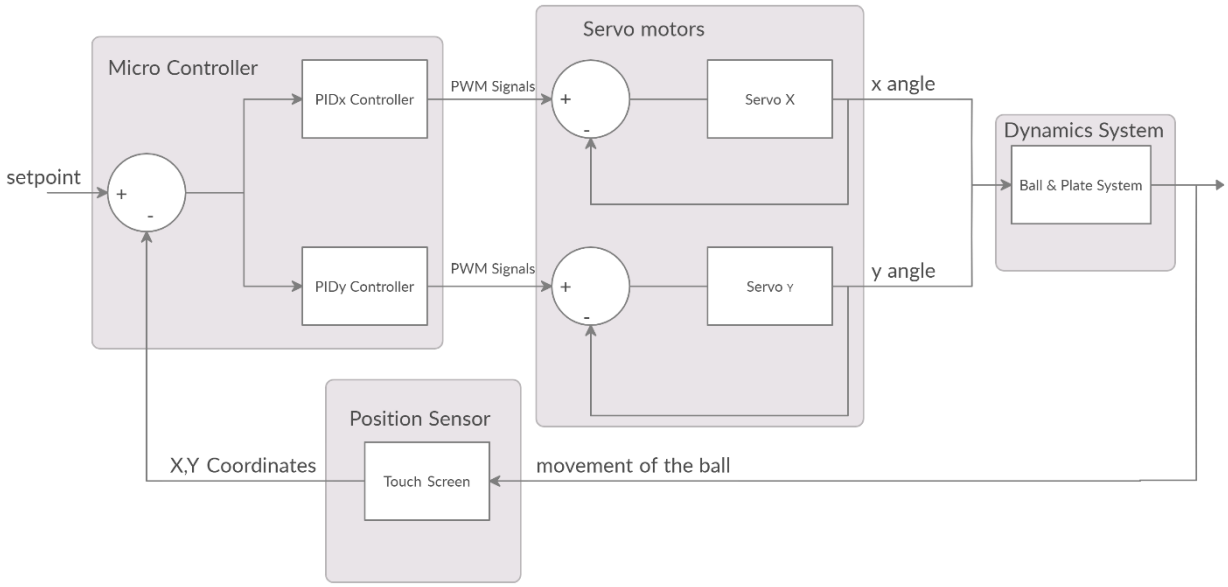

Figure 4.14 Hardware structure and control loop for PID controller.

4.7. PID Controller Design

If equations (Eq 3.19) and (Eq 3.20) are combined with equation (Eq 4.5) for the relationship between plate angle and servo angle, the transfer functions of the plant are obtained as follows.

$$H_x = \frac{X}{\theta} = \frac{1.38g}{s^2} \quad \text{(Eq 4.6)}$$

$$H_y = \frac{Y}{\theta} = \frac{1.38g}{s^2} \quad \text{(Eq 4.7)}$$

Looking at the denominator of these transfer functions, we can say that the system is marginally stable since there are 2 roots at the origin. That is, when the system is included in a feedback loop it results in only having undamped oscillation components in the step response. To stabilize the system a PID controller is designed.

An iterative method was used to set the PID parameters. The process is listed below.

- Estimate initial values.
- Decrease K_D until the ball remains on the plate.
- Decrease K_P until the ball approaches the setpoint.
- Fine-tune to get the desired response.

Iteration	K_P	K_ι	K_D
1	3	0	2
2	3	0	1.5
3	3	0	1
4	3	0	0.8
5	3	0	0.7
6	3	0	0.6
7	2.5	0	0.6
8	2.4	0	0.6
9	2.2	0	0.6
10	2.1	0	0.6
11	2.0	0	0.6
12	1.9	0	0.6
13	1.95	0	0.6
14	1.95	0	0.5

4.8.Fuzzy Logic Controller Design

The system needs to be controlled in two axes. Therefore, two fuzzy logic controllers are required, but since our system is symmetrical and the responses are the same in both cases, the controller designed for one axis can be used for two axes. In fuzzy logic, it is necessary to create a rule base table first. This may require an expert opinion or the results from the experiment may be used. For establishing the rule base table, various studies have been investigated and based on these studies, Table 4.4 has been prepared.

Table 4.4 Fuzzy rule base.

$\dot{x}$ \ x	NB	N	Z	P	PB
N	NB	N	NL	Z	PL
Z	N	NL	Z	PL	P
P	NL	Z	PL	P	PB

The fuzzy logic controller has 2 inputs. One is the position (x) error; the other is the velocity ($\dot{x}$) error. There are 5 membership functions for input 1(position) namely positive big (PB), positive (P), zero (Z), negative (N), and negative big (NB), and there are 3 membership functions for input 2 (velocity) namely positive (P), zero (Z), negative (N). Besides, there are 7 membership functions for output. These are positive big (PB), positive (P), positive low (PL), zero (Z), negative low (NL), negative (N) and negative big (NB). Membership functions of position error, velocity error, and the output angle of fuzzy controller are shown in Figure 4.15, Figure 4.16 and Figure 4.17, respectively.

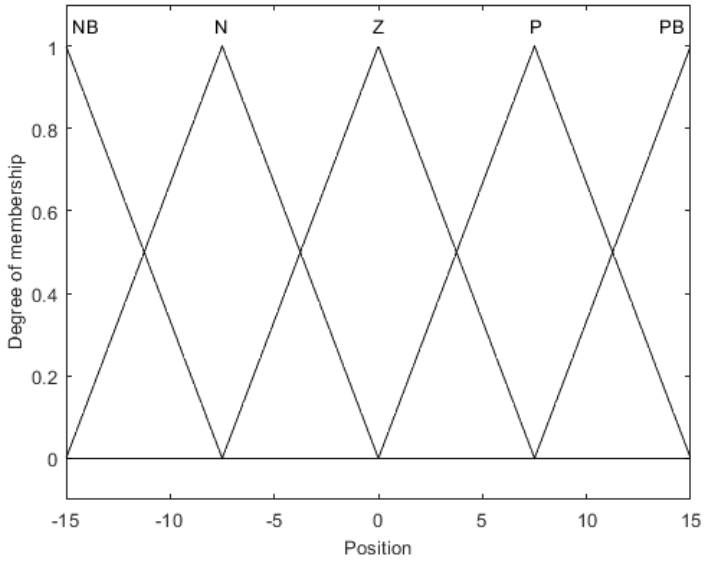

Figure 4.15 Membership function of position error

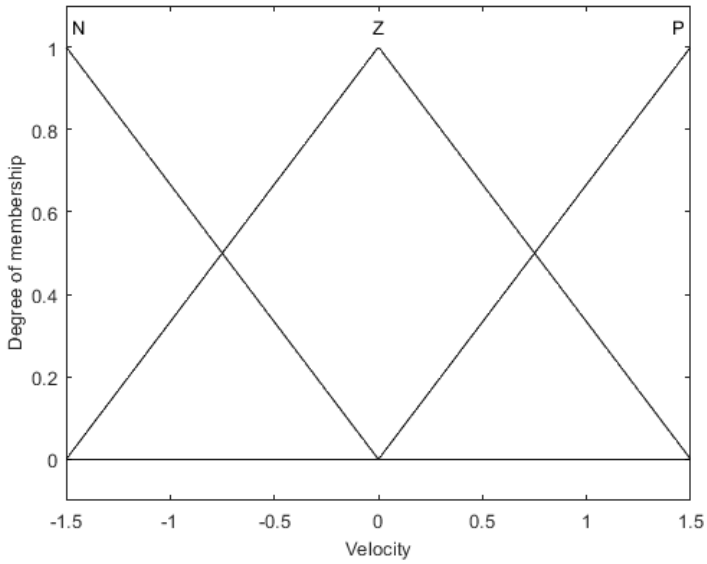

Figure 4.16 Membership function of velocity error

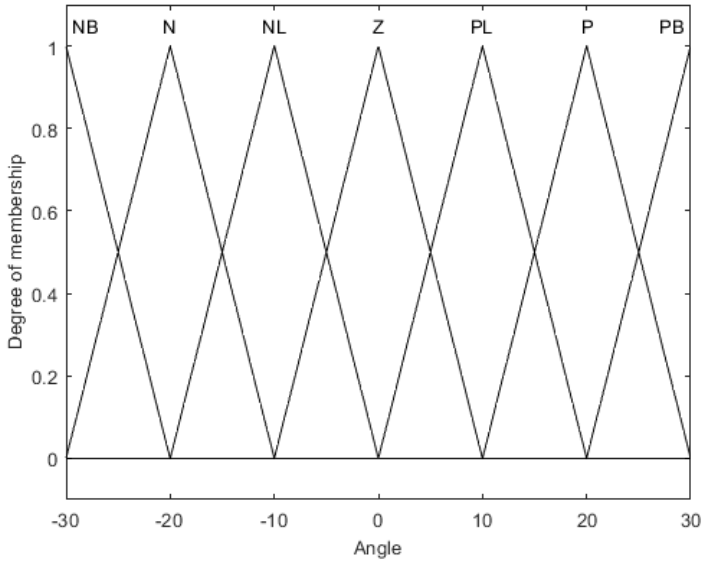

Figure 4.17 Membership function of output angle.

For simplicity, the triangular membership function was chosen. The Gaussian membership function could also be used, but the Arduino library we used for our system does not support this function. The fuzzy logic control surface resulting from the input and output functions is shown in Figure 4.18. As can be seen, the transitions on the surface are quite smooth, which means that a good controller has been designed.

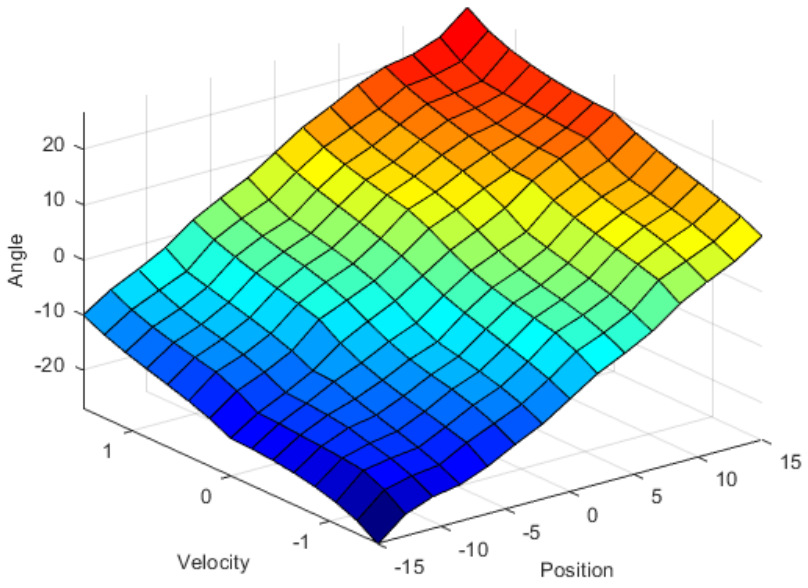

Figure 4.18 Fuzzy logic control surface.

The user firstly defines a setpoint. A difference between the setpoint and the actual ball position (x, y coordinates read from the touch panel) results in the error. Then the fuzzy logic controller calculates the desired servo angle of the system according to the position error and the velocity error to reach the set point and sends the command to the servos in the form of PWM signal. The slope of the plates is geometrically related to the angle of the servos, and the change in angle affects as an input to the dynamics of the system. The ball position obtained is read by the touch panel and it is fed backwards, forming a closed loop. The block diagram in Figure 4.19 shows the hardware structure and control loop.

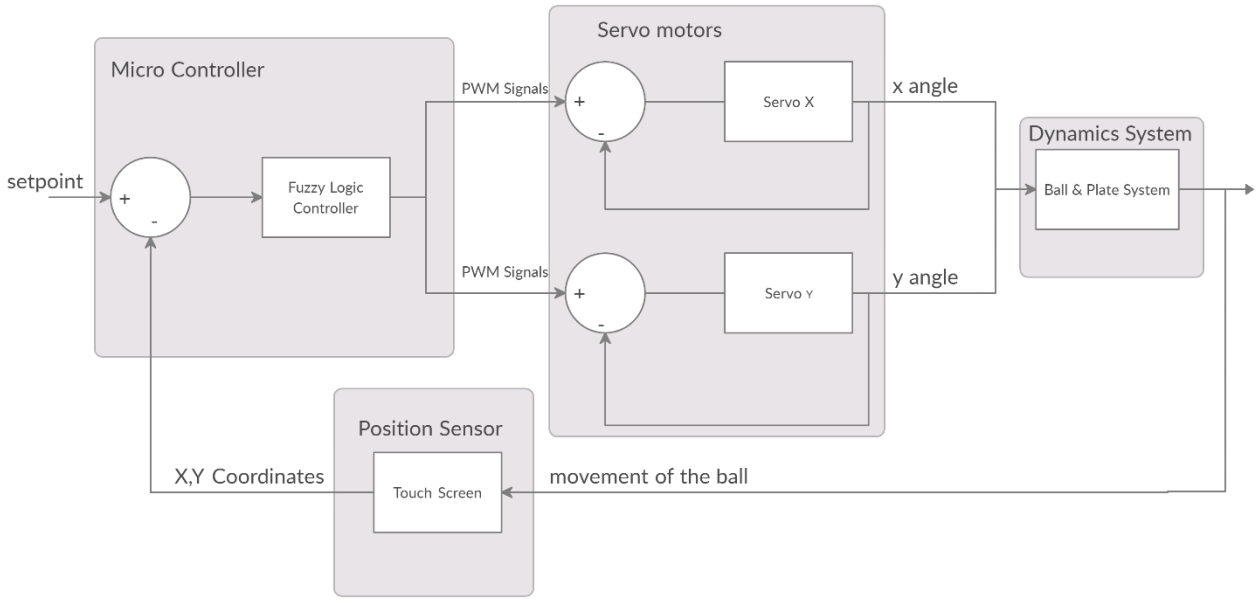

Figure 4.19 Hardware structure and control loop for fuzzy logic controller.

5. RESULTS

In this section, experimental results are used to show how our system responds to the PID and fuzzy logic controller we designed.

5.1. PID Controller

K_P and K_D values for the x-axis and y-axis are the same, and K_P and K_D values for the PID controller were chosen to be 1.95 and 0.5, respectively. How to make this selection is explained in section 4.7. In Figure 5.1 and Figure 5.2, the ball is placed on a point on the plate and its response is acquired in real time. In Figure 5.3, when the ball was at the balance point, the ball was first accelerated, and the response of the system was obtained in real time. As can be clearly seen in the figures, there is high amount of noise in our system due to the touchscreen.

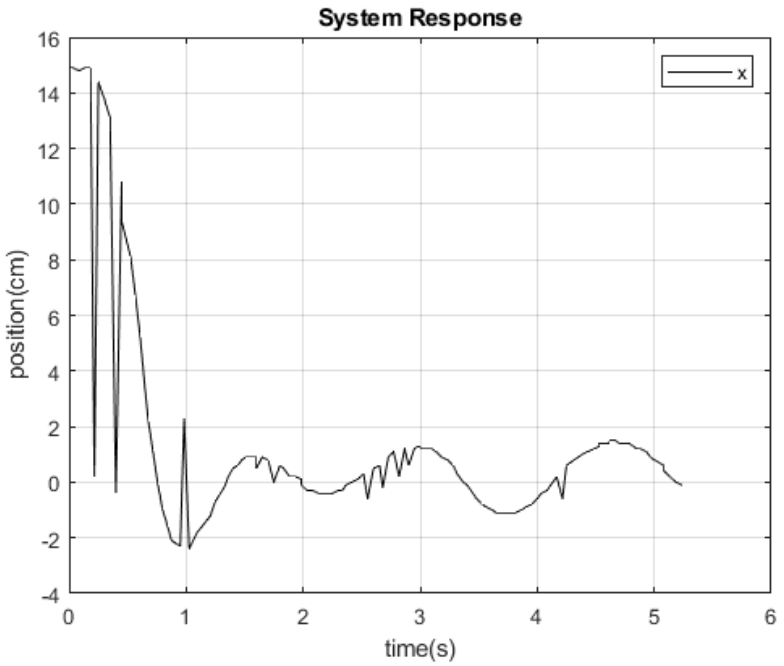

Figure 5.1 Real time system response with initial position for PID controller.

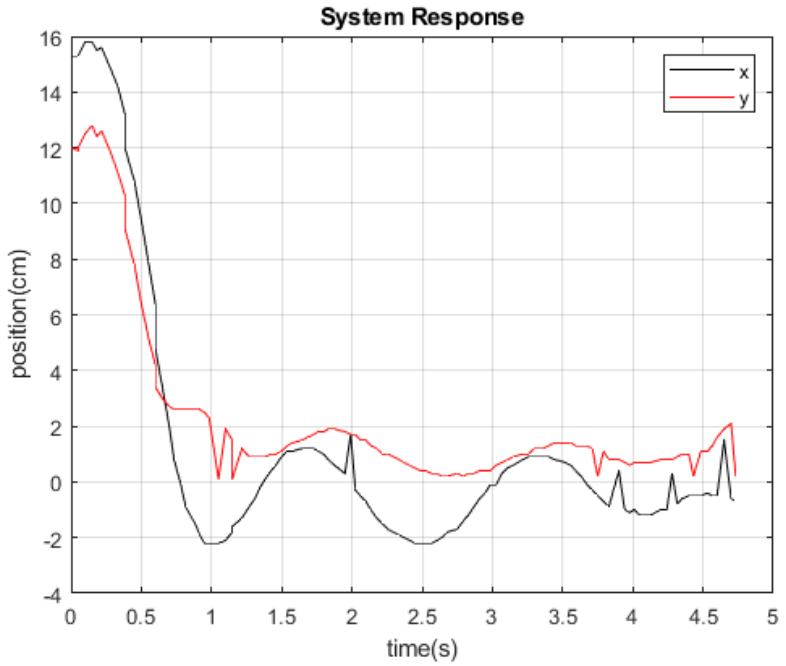

Figure 5.2 Real time system response with different initial position for PID controller.

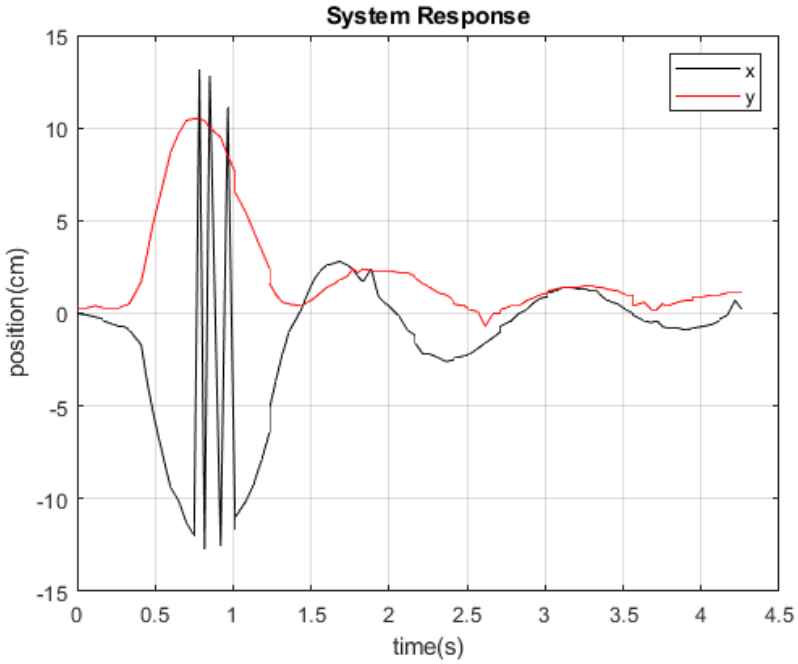

Figure 5.3 Real time system response with initial velocity for PID controller.

5.2. Fuzzy Logic Controller

In Figure 5.4, the ball is placed on a point of the plate and the response of the system is observed. It was observed that although the ball was supposed to remain balanced at one point on the plate, the ball made a circular motion on the plate and fell off the plate. We can say that the fuzzy logic controller does not work well.

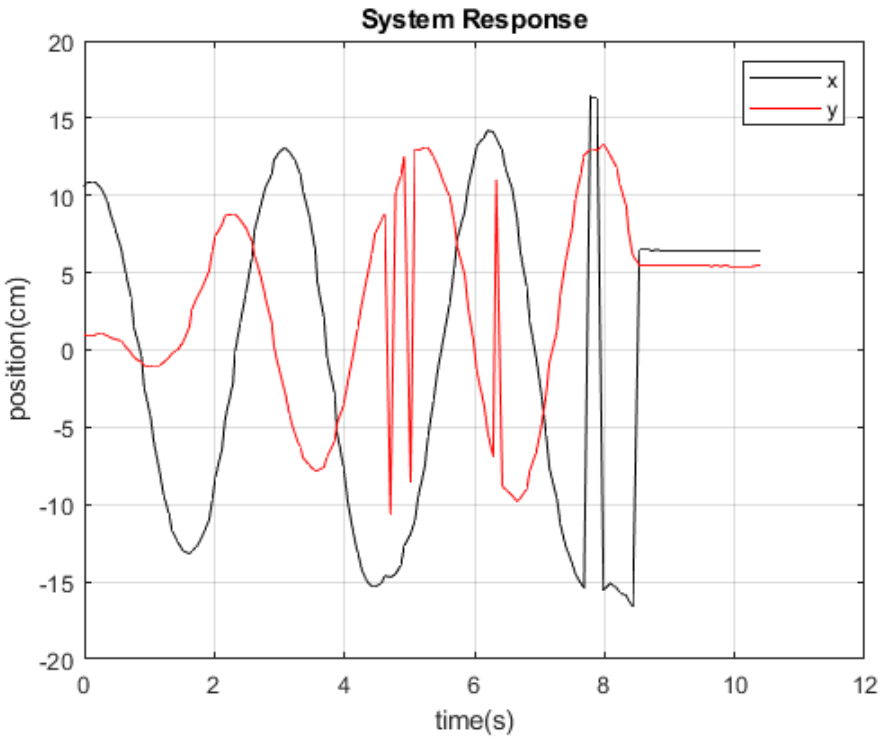

Figure 5.4 Real time system response with initial position for fuzzy logic controller.

6. DISCUSSION

As we can see from the results, there is a large amount of noise in our system and our system oscillates around a certain point instead of stopping at a point. In this section, we discuss the causes of these errors and the difficulties we encountered in creating our system.

The main cause of noise in our system is the touch screen. These noises from the touch screen can be eliminated by using different filters, but in this case, it may result in a slow response of the system. The way to get rid of this noise completely is image processing, but it has its own difficulties. Vibrations are seen in the entire structure of our system due to shattering in the servo motors. A lead controller can be used to reduce these vibrations.

The main reasons why the ball does not stop at a point in our system are as follows:

- Play in the universal joint we use.
- Rotational allowance of the platform around the z-axis.
- Deformations in the solid parts.
- Inability to hold the platform in a fixed horizontal position of exactly 0°.

To eliminate the play in the universal joint, a higher-grade universal joint or ball joint can be used. The base plate we made with a 3D printer was bent due to manufacturing errors. To reduce this bending, the base plate was tightened by fixing it to the screen with tape.

We did not use the mathematical model of the system that we created in Chapter 3 during the coding phase since we did not know the delays originating from servo motors and Arduino. In order to model the delays of the servo motor, the parameter estimation experiment can be performed, and the servo motor can be modeled. In order to model the delays of the servo motor, the parameter estimation experiment can be performed, and the servo motor can be modeled. Since we do not have the necessary data acquisition equipment for this experiment, the experiment could not be performed. An iterative method was used to tune the PID parameters. If we could model the entire system, the state space could be used, and I think that would yield better results.

In fuzzy logic, it is not necessary to know the mathematical model of the system to be controlled. Fuzzy logic controller is designed according to the instructions of the expert who knows how the system reacts, or according to the results obtained by performing experiments on the system. In our system, various studies were examined to construct the rule base table, and a rule base table was constructed based on these studies. Based on the results obtained, we can say that the controller is not working properly. The fact that our controller is not working properly may be due to both the inability to get the speed in the encoding right, and noise in our system and the rule table we have created.

7. CONCLUSİON

A ball and plate system was developed. Various design concepts for the ball and plate system were investigated. The general lines of our system were established. The system was mathematically modeled.

Mechanical system design was done in CAD environment. The system was analyzed geometrically. Suitable parts were printed with a 3D printer. Links and joints suitable for our system were determined and purchased. Servo motors suitable for our system were selected and purchased. A resistive touch screen was purchased to detect the position of the ball.

PID control has been successfully applied to the system. An iterative method was used to determine the PID parameters, since the mathematical model of the system could not be fully constructed. In order to apply PID to our system, Arduino libraries were used during the coding phase.

In order to apply different control methods to our system more easily, we tried to communicate with Simulink. However, when our system was communicating with Simulink, the correct x and y values could not be read due to noise when the sampling time was between 20 ms and 400 ms. When the sampling time was over 400 ms (0.4 s), the response of the system was too slow to keep the ball on the plate.

We designed a fuzzy logic controller for the system, but it cannot be said to work properly. Normally we intended to compare a PID controller with a fuzzy logic controller, but there is no point in comparing it with a controller that is not working properly.

All in all, this study is to show how theory is applied in a physical model and how changes and defects in the application cause differences in physical response. major engineering design steps were practically applied, the basic structure of report writing, theoretical design, prototyping and optimization were learned. In addition, this study provided a real engineering design practice.

REFERENCES

[1] "Arduino - Introduction ". 2021. *Arduino.Cc*. https://www.arduino.cc/en/guide/introduct

[2] "Servo Motors Work | How Servo Motors Work". 2021. Jameco.Com. https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html

[3] "Winmate Resistive Touch -Panel PC/ Industrial LCD Display / Digital Signage/ Marine LCD/HMI Solution Provider And Manufacturer". 2021. Cn.Winmate.Com.Tw. https://cn.winmate.com.tw/resistive_touch.asp.

[4] Spaček, B.L.,(2016). Digital Control of CE 151 Ball & Plate Model.

[5] Dušek, F., Honc, D., Sharma, R. (2017). Modelling of ball and plate system based on first principle model and optimal control, Pardubice, Czech Republic.

[6] Nise, N.S., (2010). Control Systems Engineering, 6th edition. John Wiley & Sons, Inc.

[7] Bruce, J., Keeling, C., Rodriguez, R. (2011) Four Degree of Freedom Control System Using a Ball on a Plate.

[8] Kumar, J., Showme, N., Aravind, M., Akshay, R. (2016) Design and control of ball on plate system

[9] Itani, A. (2017) Ball Plate Balancing System Using Image Processing,

[10] Frank, A.H., Tjernström, M. (2019). Construction and theoretical study. KTH, Stockholm, Sweden.

[11] Kayacan, E., Khanesar, M.A. (2016). Fuzzy Neural Networks for Real Time Control Applications, Butterworth-Heinemann, Massachusetts, USA.

[12] W. Kickert, E. Mamdani, Analysis of a fuzzy logic controller, Fuzzy Sets Syst. 1 (1978) 29-44.

[13] Pattanapong, Y., Deelertpaiboon, B. (2013). Ball and Plate Position Control Based on Fuzzy Logic with Adaptive Integral Control Action, Bangkok, Thailand.

[14] Kuncan, M., Kaplan, K., Acar, F., Kundakçi, I.M., Ertunç, M. (2016). Fuzzy Logic Based Ball on Plate Balancing System Real Time Control by Image Processing, Kocaeli, Turkey

APPENDICES

A. Geometrical Analysis Code

```
A=0.06;
B=0.06;
C=0.031;
h=0.06;
b=0.09;
theta=-30;
syms  fi alfa
s=zeros(1,61);
mn=zeros(1,61);
ss=zeros(1,61);
for i=1:61
eqn1=-C*sind(theta)+B*cosd(fi)==h-A*sind(alfa);
eqn2=-C*cosd(theta)-B*sind(fi)==-b+A*cosd(alfa);
[L]=solve([eqn1,eqn2],[fi,alfa]);

q=double(L.alfa);
s(i)=q(1);
ss(i)=q(2);
mn(i)=theta;
theta=theta+1;
end
plot(s,mn,'k','LineWidth',1.2);
hold on
plot(C/A*mn,mn,'r.')
grid on
title('Geometrical Analysis');
xlabel('Angle \alpha [°]');
ylabel('Angle \theta [°]');
legend('geometrical realationship','linearized relationship');
```

B. CAD view and Technical Drawings

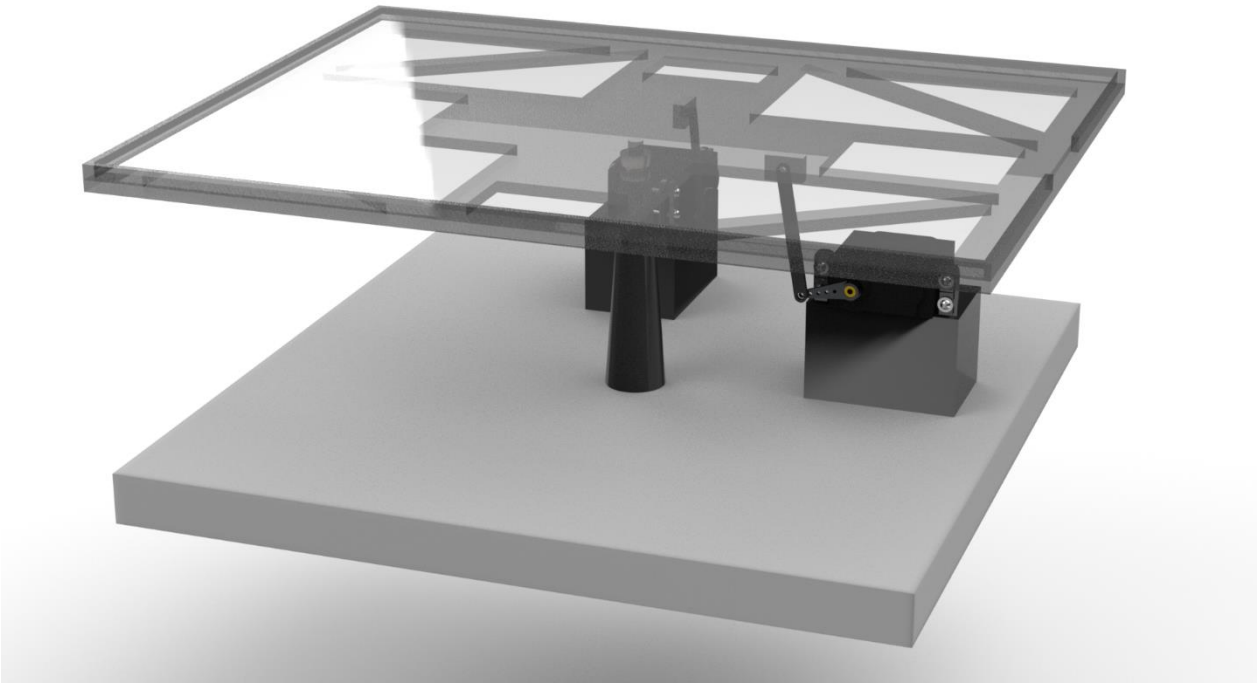

Figure B- 1 Full view of mechanical system.

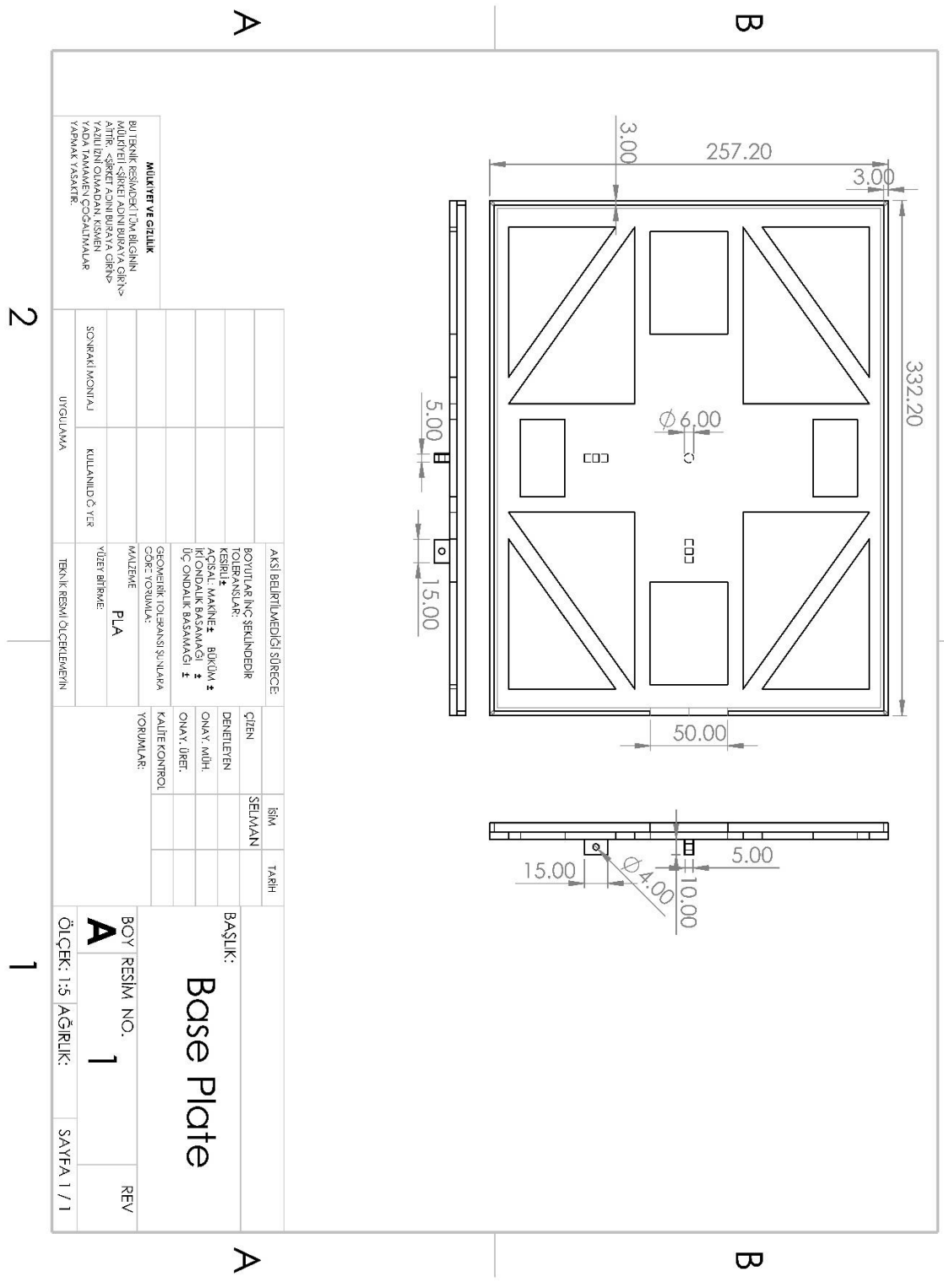

Figure B- 2 Detailed view of base plate.

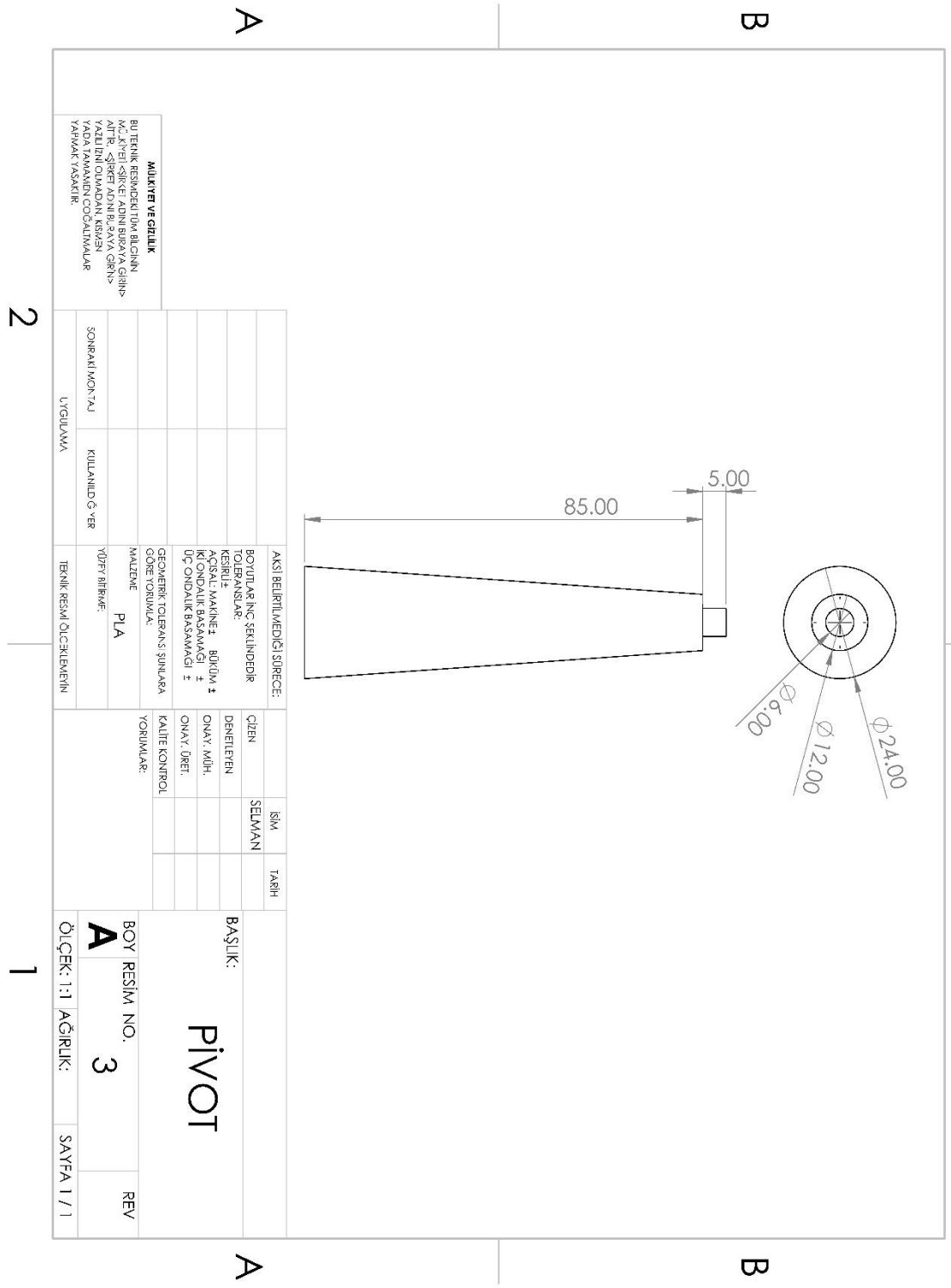

Figure B- 3 Detailed view of pivot.

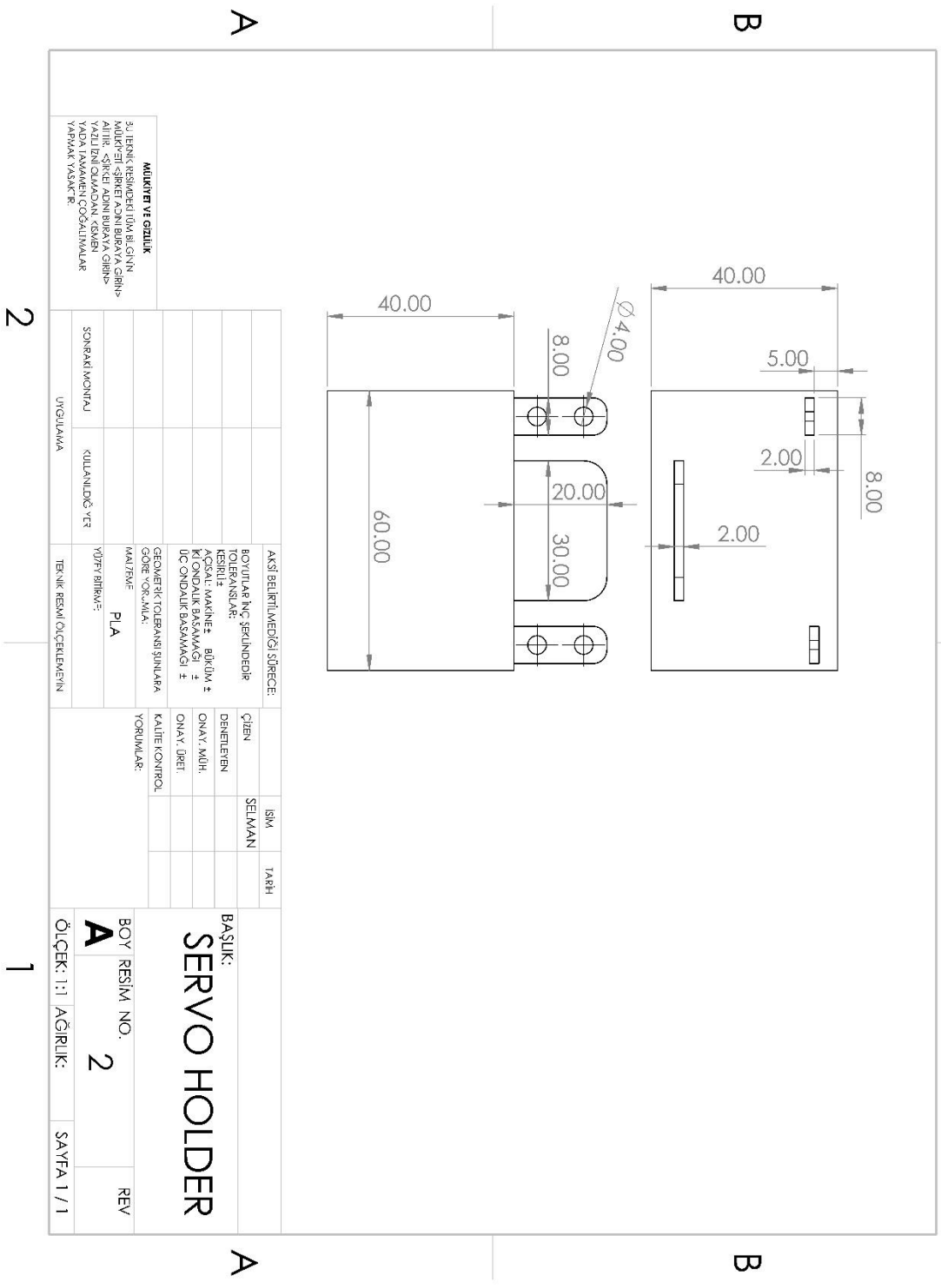

Figure B- 4 Detailed view of servo holder.

C. Final Construction of the System view

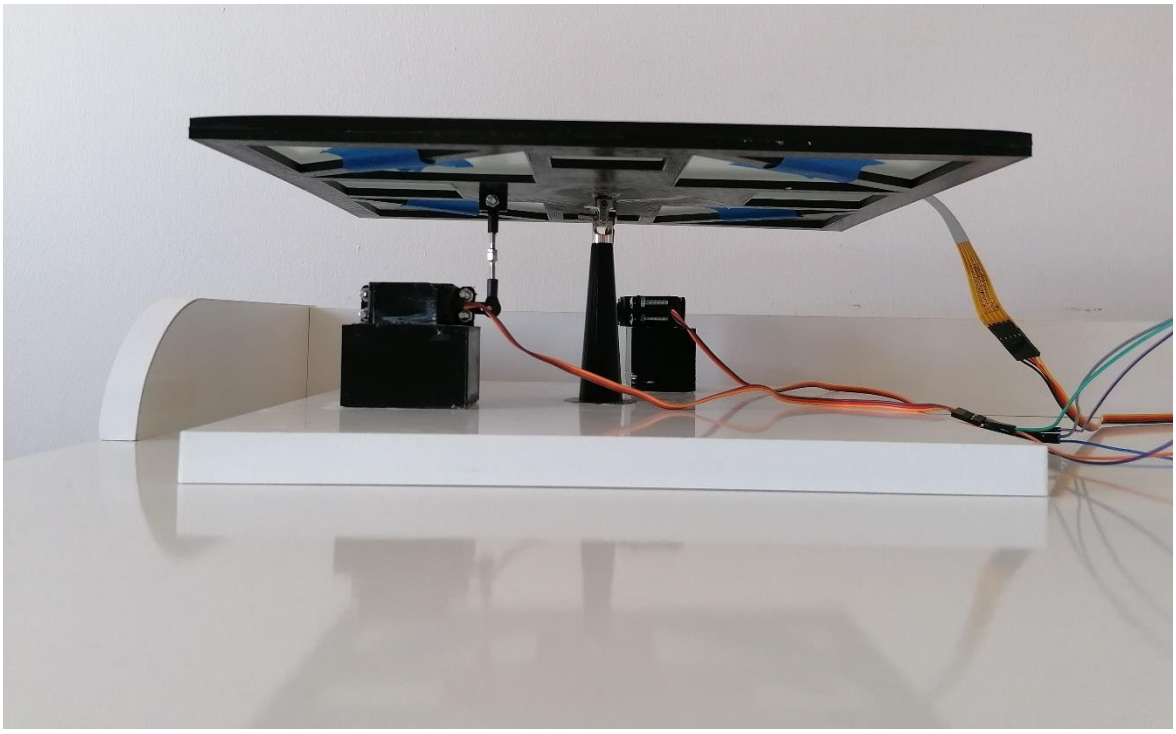

Figure C- 1 Ball and plate system view.

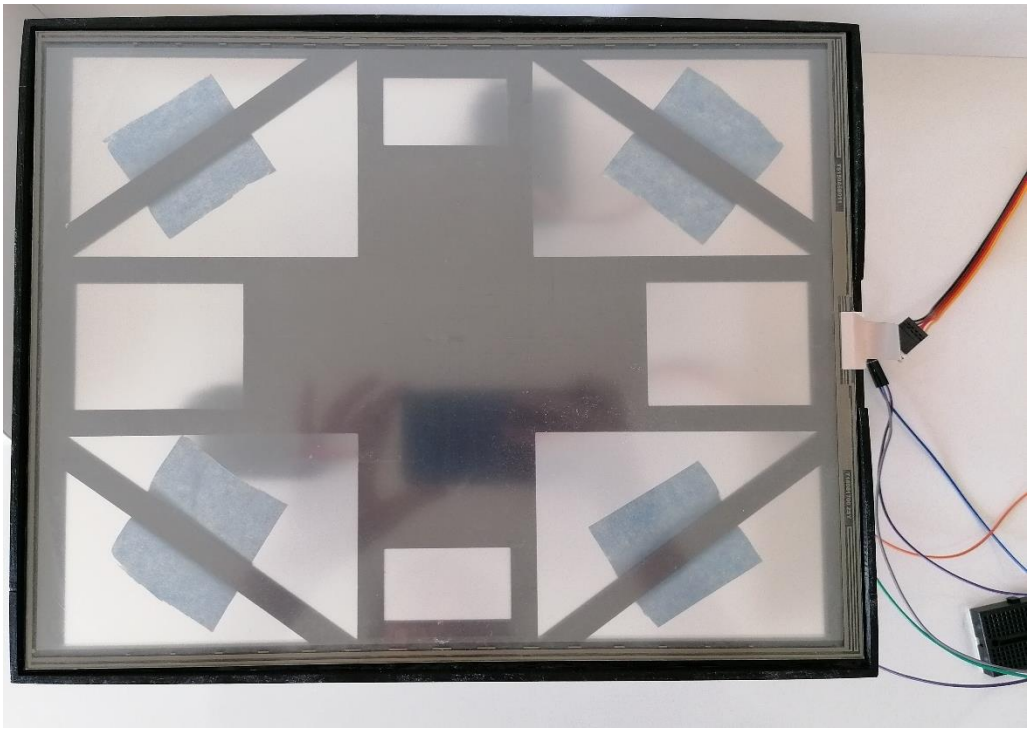

Figure C- 2 System top view.

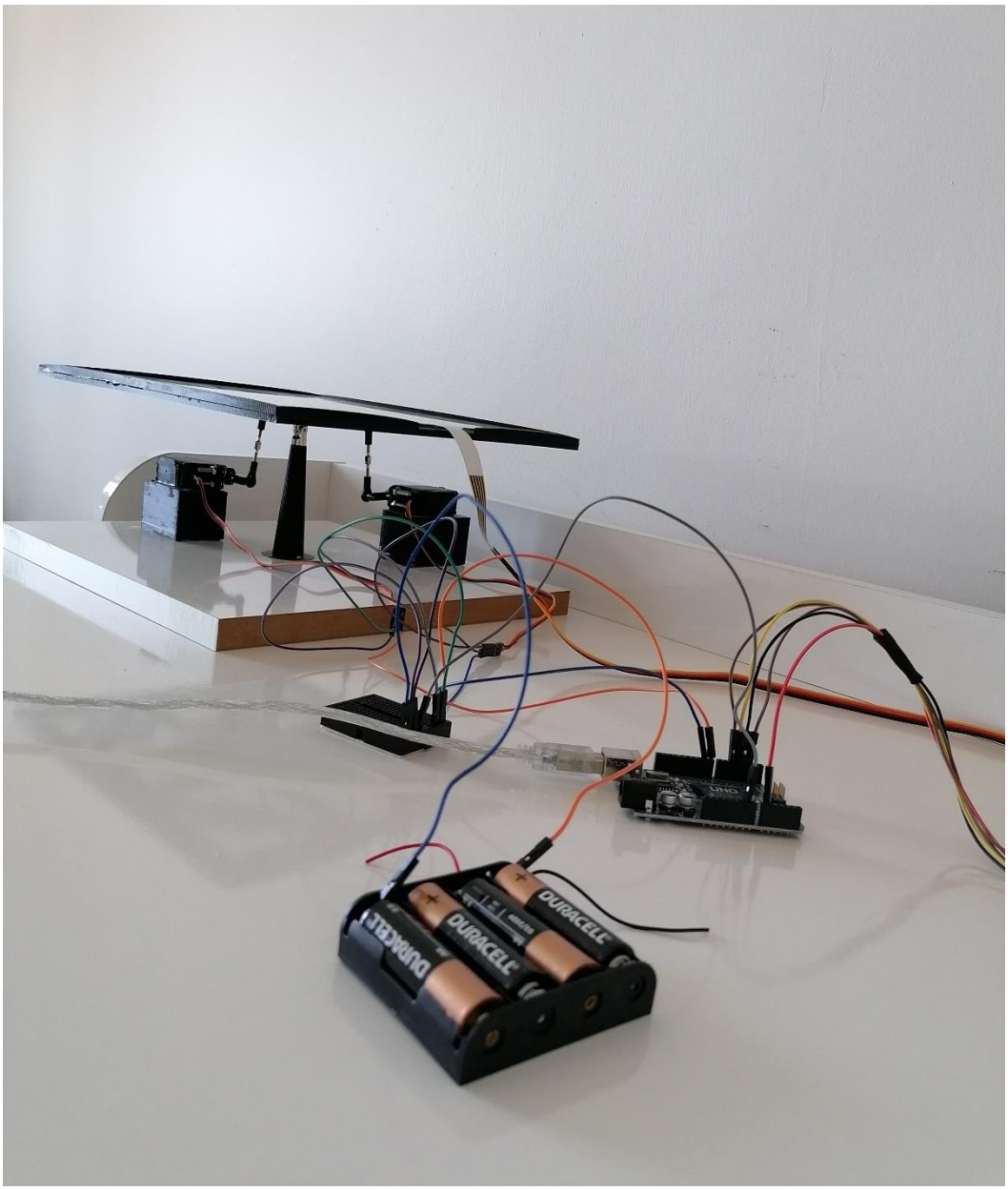

Figure C- 3 Whole system view.

D. Arduino Code for PID Controller

```
#include <Servo.h>
#include <PID_v1.h>
const int pin1UL = 2;// Pin 1 at touchs screen (Upper Left)// BEYAZ MOR
KAHVERENGİ MAVİ SİYAH - SİYAHTAN SARIYA
const int pin2UR = 3;// Pin 2 at touchs screen (Upper Right)
const int pin4LR = 4;// Pin 4 at touchs screen (Lower Right)
const int pin5LL = 5;// Pin 5 at touchs screen (Lower Left)
const int sense = A0; // Pin 3 at touchs screen (COMMON)
double SetpointX, InputX, OutputX; //for X axis
double SetpointY, InputY, OutputY; //for Y axis
Servo servoX; //X axis
Servo servoY; //Y axis
int Ts =25;//Time Sample
float KpX = 1.95;
float KiX = 0;
float KdX = 0.5;
float KpY = 1.95;
float KiY = 0;
float KdY = 0.5;
double Xoutput,Youtput;
PID PIDX(&InputX, &OutputX, &SetpointX, KpX, KiX, KdX, DIRECT);
PID PIDY(&InputY, &OutputY, &SetpointY,KpY,KiY,KdY, DIRECT);

void setup() {
SetpointX=16;
SetpointY=11;
servoX.attach(8);
servoY.attach(9);
servoX.write(99);//Balance position
servoY.write(80);//Balance position
pinMode(pin1UL,OUTPUT);
pinMode(pin2UR,OUTPUT);
pinMode(pin4LR,OUTPUT);
pinMode(pin5LL,OUTPUT);
Serial.begin(9600);
PIDX.SetMode(AUTOMATIC);
PIDX.SetOutputLimits(-60,30);//Değişecek.
PIDY.SetMode(AUTOMATIC);
PIDY.SetOutputLimits(-30, 30);//Değişecek.
PIDX.SetSampleTime(Ts);
PIDY.SetSampleTime(Ts);
delay(100);
}

void loop() {
SetpointX=16;
SetpointY=11;
InputX= Xcoordinates(); // read and convert X coordinate
InputY= Ycoordinates(); // read and convert Y coordinate
PIDX.Compute(); //action control X compute
PIDY.Compute(); // action control Y compute
Xoutput = 99 + OutputX;//Değişecek
Youtput = 80 + OutputY;//Değişecek
//Gereksiz Değişecek
Serial.print(InputX);
```

```
Serial.print("\t");
Serial.print(InputY);
Serial.print("\t");

Serial.print(OutputX);
Serial.print("\t");
Serial.println(OutputY);
Serial.print("\t");
if(InputX > SetpointX + 0.1 && InputX < SetpointX - 0.1 && InputY > SetpointY
+ 0.1 && InputY < SetpointY - 0.1 )
{
servoX.write(90);//Balance position
servoY.write(80);//Balance position
delay(20);
}
else
{
servoX.write(Xoutput);//Controll position
servoY.write(Youtput);//Controll position
delay(20);
}
}
double Xcoordinates()
{
double xVal;
digitalWrite(pin1UL,HIGH);
digitalWrite(pin2UR,LOW);
digitalWrite(pin4LR,LOW);
digitalWrite(pin5LL,HIGH);
delay(10);
xVal = analogRead(sense);


xVal = map(xVal,250,781,0,320);
xVal=xVal/10;//mm to cm
return xVal;
}
double Ycoordinates()
{
double yVal;
digitalWrite(pin1UL,LOW);
digitalWrite(pin2UR,LOW);
digitalWrite(pin4LR,HIGH);
digitalWrite(pin5LL,HIGH);
delay(10);
yVal = analogRead(sense);


yVal = map(yVal,232,754,0,240);
yVal=yVal/10;//mm to cm
return yVal;
}
```

E. Arduino Code for Fuzzy Logic Controller

```
#include <Servo.h>
#include <Fuzzy.h>

const int pin1UL = 2;// Pin 1 at touchs screen (Upper Left)
const int pin2UR = 3;// Pin 2 at touchs screen (Upper Right)
const int pin4LR = 4;// Pin 4 at touchs screen (Lower Right)
const int pin5LL = 5;// Pin 5 at touchs screen (Lower Left)
const int sense = A0; // Pin 3 at touchs screen (COMMON)
float SetpointX, InputX, Input2X, velX, OutputX ,Xoutput; //for X axis
float SetpointY, InputY, Input2Y, velY, OutputY ,Youtput; //for Y axis
Servo servoX; //X axis
Servo servoY; //Y axis

Fuzzy *fuzzy = new Fuzzy();

// FuzzyInput analog1
FuzzySet *verylow1          = new FuzzySet(-15, -15, -15, -7.5);
FuzzySet *low1          = new FuzzySet(-15, -7.5, -7.5,0);
FuzzySet *mid1          = new FuzzySet(-7.5, 0, 0, 7.5);
FuzzySet *high          = new FuzzySet(0, 7.5, 7.5, 15);
FuzzySet *veryhigh          = new FuzzySet(7.5, 15, 15, 15);

// FuzzyInput analog2
FuzzySet *back2          = new FuzzySet(-1.5,-1.5, -1.5, 0);
FuzzySet *middle          = new FuzzySet(-1.5, 0, 0, 1.5);
FuzzySet *forward2          = new FuzzySet(0, 1.5, 1.5, 1.5);

// FuzzyOutput
FuzzySet *verylowout1            = new FuzzySet(-30, -30, -30, -20);
FuzzySet *midlowout1             = new FuzzySet(-30, -20, -20, -10);
FuzzySet *lowout1                = new FuzzySet(-20, -10, -10, 0);
FuzzySet *midout1                = new FuzzySet(-10, 0, 0, 10);
FuzzySet *highout1               = new FuzzySet(0, 10, 10, 20);
FuzzySet *midhighout1            = new FuzzySet(10, 20, 20, 30);
FuzzySet *veryhighout1           = new FuzzySet(20, 30, 30, 30);


void setup()
{
  SetpointX=16;
  SetpointY=11;
  servoX.attach(8);
  servoY.attach(9);
  servoX.write(99);//Balance position
  servoY.write(80);//Balance position
  pinMode(pin1UL,OUTPUT);
  pinMode(pin2UR,OUTPUT);
  pinMode(pin4LR,OUTPUT);
  pinMode(pin5LL,OUTPUT);
  Serial.begin(9600);

  //--------------------------------------
  // FuzzyInput
  FuzzyInput *analog1 = new FuzzyInput(1);
```

```
  analog1->addFuzzySet(verylow1);
  analog1->addFuzzySet(low1);
  analog1->addFuzzySet(mid1);
  analog1->addFuzzySet(high);
  analog1->addFuzzySet(veryhigh);
  fuzzy->addFuzzyInput(analog1);

  // FuzzyInput
  FuzzyInput *analog2 = new FuzzyInput(2);

  analog2->addFuzzySet(back2);
  analog2->addFuzzySet(middle);
  analog2->addFuzzySet(forward2);
  fuzzy->addFuzzyInput(analog2);

  // FuzzyOutput
  FuzzyOutput *led = new FuzzyOutput(1);

  led->addFuzzySet(verylowout1);
  led->addFuzzySet(midlowout1);
  led->addFuzzySet(lowout1);
  led->addFuzzySet(midout1);
  led->addFuzzySet(highout1);
  led->addFuzzySet(midhighout1);
  led->addFuzzySet(veryhighout1);
  fuzzy->addFuzzyOutput(led);
  //--------------------------------------

  // Building
FuzzyRule////////////////////////////////////////////////////////////////// 1
  FuzzyRuleAntecedent *verylow1_back2 = new FuzzyRuleAntecedent();
  verylow1_back2->joinWithAND(verylow1, back2);

  FuzzyRuleConsequent *led_verylowout1 = new FuzzyRuleConsequent();
  led_verylowout1->addOutput(verylowout1);

  FuzzyRule *fuzzyRule1 = new FuzzyRule(1, verylow1_back2, led_verylowout1);
  fuzzy->addFuzzyRule(fuzzyRule1);

  // Building
FuzzyRule////////////////////////////////////////////////////////////////// 2
  FuzzyRuleAntecedent *verylow1_middle = new FuzzyRuleAntecedent();
  verylow1_middle->joinWithAND(verylow1, middle);

  FuzzyRuleConsequent *led_midlowout1 = new FuzzyRuleConsequent();
  led_midlowout1->addOutput(midlowout1);

  FuzzyRule *fuzzyRule2 = new FuzzyRule(2, verylow1_middle, led_midlowout1);
  fuzzy->addFuzzyRule(fuzzyRule2);

  // Building
FuzzyRule////////////////////////////////////////////////////////////////// 3
  FuzzyRuleAntecedent *verylow1_forward2 = new FuzzyRuleAntecedent();
  verylow1_forward2->joinWithAND(verylow1, forward2);

  FuzzyRuleConsequent *led_lowout1 = new FuzzyRuleConsequent();
  led_lowout1->addOutput(lowout1);
```

```
  FuzzyRule *fuzzyRule3 = new FuzzyRule(3, verylow1_forward2, led_lowout1);
  fuzzy->addFuzzyRule(fuzzyRule3);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 4
  FuzzyRuleAntecedent *low1_back2 = new FuzzyRuleAntecedent();
  low1_back2->joinWithAND(low1, back2);


  FuzzyRule *fuzzyRule4 = new FuzzyRule(4, low1_back2, led_midlowout1);
  fuzzy->addFuzzyRule(fuzzyRule4);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 5
  FuzzyRuleAntecedent *low1_middle = new FuzzyRuleAntecedent();
  low1_middle->joinWithAND(low1, middle);

  FuzzyRule *fuzzyRule5 = new FuzzyRule(5, low1_middle, led_lowout1);
  fuzzy->addFuzzyRule(fuzzyRule5);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 6
  FuzzyRuleAntecedent *low1_forward2 = new FuzzyRuleAntecedent();
  low1_forward2->joinWithAND(low1, forward2);

  FuzzyRuleConsequent *led_midout1 = new FuzzyRuleConsequent();
  led_midout1->addOutput(midout1);

  FuzzyRule *fuzzyRule6 = new FuzzyRule(6, low1_forward2, led_midout1);
  fuzzy->addFuzzyRule(fuzzyRule6);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 7
  FuzzyRuleAntecedent *mid1_back2 = new FuzzyRuleAntecedent();
  mid1_back2->joinWithAND(mid1, back2);

  FuzzyRule *fuzzyRule7 = new FuzzyRule(7, mid1_back2, led_lowout1);
  fuzzy->addFuzzyRule(fuzzyRule7);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 8
  FuzzyRuleAntecedent *mid1_middle = new FuzzyRuleAntecedent();
  mid1_middle->joinWithAND(mid1, middle);

  FuzzyRule *fuzzyRule8 = new FuzzyRule(8, mid1_middle, led_midout1);
  fuzzy->addFuzzyRule(fuzzyRule8);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 9
  FuzzyRuleAntecedent *mid1_forward2 = new FuzzyRuleAntecedent();
  mid1_forward2->joinWithAND(mid1, forward2);

  FuzzyRuleConsequent *led_highout1 = new FuzzyRuleConsequent();
  led_highout1->addOutput(highout1);

  FuzzyRule *fuzzyRule9 = new FuzzyRule(9, mid1_forward2, led_highout1);
```

```
  fuzzy->addFuzzyRule(fuzzyRule9);
  // Building
FuzzyRule////////////////////////////////////////////////////////////// 10
  FuzzyRuleAntecedent *high_back2 = new FuzzyRuleAntecedent();
  high_back2->joinWithAND(high, back2);

  FuzzyRule *fuzzyRule10 = new FuzzyRule(10, high_back2, led_midout1);
  fuzzy->addFuzzyRule(fuzzyRule10);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 11
  FuzzyRuleAntecedent *high_middle = new FuzzyRuleAntecedent();
  high_middle->joinWithAND(high, middle);

  FuzzyRule *fuzzyRule11 = new FuzzyRule(11, high_middle, led_highout1);
  fuzzy->addFuzzyRule(fuzzyRule11);

  // Building
FuzzyRule////////////////////////////////////////////////////////////// 12
  FuzzyRuleAntecedent *high_forward2 = new FuzzyRuleAntecedent();
  high_forward2->joinWithAND(high, forward2);

  FuzzyRuleConsequent *led_midhighout1 = new FuzzyRuleConsequent();
  led_midhighout1->addOutput(midhighout1);

  FuzzyRule *fuzzyRule12 = new FuzzyRule(12, high_forward2, led_midhighout1);
  fuzzy->addFuzzyRule(fuzzyRule12);

    // Building
FuzzyRule////////////////////////////////////////////////////////////// 13
  FuzzyRuleAntecedent *veryhigh_back2 = new FuzzyRuleAntecedent();
  veryhigh_back2->joinWithAND(veryhigh, back2);

  FuzzyRule *fuzzyRule13 = new FuzzyRule(13, veryhigh_back2, led_highout1);
  fuzzy->addFuzzyRule(fuzzyRule13);

    // Building
FuzzyRule////////////////////////////////////////////////////////////// 14
  FuzzyRuleAntecedent *veryhigh_middle = new FuzzyRuleAntecedent();
  veryhigh_middle->joinWithAND(veryhigh, middle);

  FuzzyRule *fuzzyRule14 = new FuzzyRule(14, veryhigh_middle,
led_midhighout1);
  fuzzy->addFuzzyRule(fuzzyRule14);

    // Building
FuzzyRule////////////////////////////////////////////////////////////// 15
  FuzzyRuleAntecedent *veryhigh_forward2 = new FuzzyRuleAntecedent();
  veryhigh_forward2->joinWithAND(veryhigh, forward2);

  FuzzyRuleConsequent *led_veryhighout1 = new FuzzyRuleConsequent();
  led_veryhighout1->addOutput(veryhighout1);

  FuzzyRule *fuzzyRule15 = new FuzzyRule(15, veryhigh_forward2,
led_veryhighout1);
  fuzzy->addFuzzyRule(fuzzyRule15);
  delay(100);
```

```
}


void loop()
{
  SetpointX=16;
  SetpointY=11;
  InputX= Xcoordinates(); // read and convert X coordinate
  InputY= Ycoordinates(); // read and convert Y coordinate
  Input2X= Xcoordinates(); // read and convert X coordinate
  Input2Y= Ycoordinates(); // read and convert Y coordinate
  velX=(Input2X-InputX)/0.615;
  velY=(Input2Y-InputY)/0.615;
if (velX<-1.5)
{
velX= -1.5;
}
if (velX>1.5)
{
  velX= 1.5;
}

if (velY<-1.5)
{
  velY= -1.5;
}

if (velY>1.5)
{
  velY= 1.5;
}


float in_analog1 =InputX-SetpointX;
float in_analog2 =velX;

  //--------------------------------------

  fuzzy->setInput(1, in_analog1);
  fuzzy->setInput(2, in_analog2);
  fuzzy->fuzzify();

  float out_X = fuzzy->defuzzify(1);
  float in_analog1y =InputY-SetpointY;
  float in_analog2y =velY;
   fuzzy->setInput(1, in_analog1y);
  fuzzy->setInput(2, in_analog2y);
  fuzzy->fuzzify();
float out_Y = fuzzy->defuzzify(1);
Xoutput=99-1*out_X;
Youtput=80-1*out_Y;

if(InputX > SetpointX + 0.1 && InputX < SetpointX - 0.1 && InputY > SetpointY
+ 0.1 && InputY < SetpointY - 0.1 )
{
servoX.write(99);//Balance position
servoY.write(80);//Balance position
```

```
delay(20);
}
else
{
servoX.write(Xoutput);//Controll position
servoY.write(Youtput);//Controll position
delay(20);
}

  //--------------------------------------
  Serial.print(' ');
  Serial.print(in_analog1);
   Serial.print(' ');
  Serial.print(in_analog2);
  Serial.print(' ');
  Serial.print(out_X);
  Serial.print(' ');
  Serial.print(in_analog1y);
   Serial.print(' ');
  Serial.print(in_analog2y);
  Serial.print(' ');
  Serial.println(out_Y);

}
double Xcoordinates()
{
double xVal;
digitalWrite(pin1UL,HIGH);
digitalWrite(pin2UR,LOW);
digitalWrite(pin4LR,LOW);
digitalWrite(pin5LL,HIGH);
delay(10);
xVal = analogRead(sense);


xVal = map(xVal,250,781,0,320);
xVal=xVal/10;//mm to cm
return xVal;
}
double Ycoordinates()
{
double yVal;
digitalWrite(pin1UL,LOW);
digitalWrite(pin2UR,LOW);
digitalWrite(pin4LR,HIGH);
digitalWrite(pin5LL,HIGH);
delay(10);
yVal = analogRead(sense);


yVal = map(yVal,232,754,0,240);
yVal=yVal/10;//mm to cm
return yVal;
}
```