PALESTINE POLYTECHNIC UNIVERSITY

College of Information Technology and Computer Engineering

Computer Engineering Department

Ball Balancing Robot (Ballbot)

Team Members

Lana Qawasmy - 181105

Hiba Edais - 187535

Supervisors

Dr. Alaa Halawani

and

Eng. Wael Takrouri

Hebron - Palestine

May, 2023

Certification and Anti-Plagiarism Declaration

This is to declare that the graduation project produced under the supervision of Dr. Alaa Halawani and Eng. Wael Takrouri having the title Ball Balancing Robot was prepared by students Lana Qawasmy and Hiba Edais in partial fulfillment of the requirements for the degree of Bachelor in Computer Systems Engineering and no part hereof has been reproduced illegally (in particular: cut and paste) which can be considered as Plagiarism.

All referenced parts have been used to support and argue the idea and have been cited properly. We certify that we will not commit any plagiarism, cheating, or any other academic integrity violation. We will be responsible and liable for any consequence if a violation of this declaration is proven.

Date: 23 May 2023

Graduation project group's students

Name: Lana Qawasmy

Signature

Lana

Name: Hiba Edais

Signature

[illegible]

ABSTRACT

One of the most challenging problems in a control system is balancing a system. This project presents the design and implementation of the ball-balancing robot. The system consists of a rigid plate with a ball rolling freely on the plate, two motors to control the plate, a camera to detect the position of the ball and give feedback to the system, and a Raspberry Pi that will run the algorithms. The aim is to maintain a stable ball position on the plate, rejecting position disturbances. The result is the ball is balanced at the center of the plate.

Keywords - Ball Balance, Balancing Robot, PID controller.

الملخص

واحدة من أكثر المشاكل صعوبة في نظام التحكم هي موازنة النظام. يقدم هذا المشروع تصميم وتنفيذ روبوت موازنة الكرة. يتكون النظام من لوحة صلبة مع كرة تتدحرج بحرية على اللوحة، ومحركين للتحكم في اللوحة، وكاميرا لاكتشاف موضع الكرة وإعطاء تغذية راجعة للنظام، وراسبيري باي الذي سيشغل الخوارزميات. الهدف هو الحفاظ على وضع الكرة المستقر على اللوحة، ورفض اضطرابات الموضع. والنتيجة هي أن الكرة متوازنة في منتصف اللوحة.

الكلمات المفتاحية : توازن الكرة، روبوت موازنة، المتحكم التناسبي التكاملي التفاضلي.

ACKNOWLEDGEMENTS

In the name of ”Allah”, the most beneficent and merciful who gave us strength, and knowledge, and inspired us with patience and diligence that enabled us to complete this project successfully.

We would like to express our sincere appreciation and gratitude to our graduation project supervisors: Eng. Wael Takrouri and Dr. Alaa Halawani for their insightful direction, continuous support, encouragement, patience, and being there to answer our questions and provide us with valuable suggestions and tips throughout the project’s duration, we have faced several obstacles, and without the help and support of our supervisors it wouldn’t be possible to overcome and pass them, we had extremely fortunate to have such advisors and mentors.

A lot of people have contributed to the completion of our graduation project directly or indirectly by supporting and encouraging us to finish the graduation project, so We would like to warmly thank all of them.

Not to forget the main reason that kept our standing, pushed us forward toward this success, and generously encouraged and continuously supported us throughout our life, which represents our parents, where none of this would be achieved without them so a special thanks to them. We do dedicate our dissertation to them. They filled us with love, care, and concern all the time.

Finally, thanks to Palestine Polytechnic University and its staff for all their teaching and guidance efforts.

Contents

List of Tables

List of Figures

List of Acronyms

Ballbot	Ball Balancing Robot
PID	Proportional Integral Derivative
RGB	Red, Green And Blue
CMYK	Cyan, Magenta, Yellow And Black
HSV	Hue, Saturation And Value
SP	Setpoint
PV	Process Variable
PWM	Pulse Width Modulation
USB	Universal Serial Bus
OpenCV	Open Source Computer Vision Library
IDE	Integrated Development Environment
CoG	Center Of Gravity
HD	High Definition
GB	Gigabyte
GHz	Gigahertz
MP	Megapixel
ARM	Advanced RISC Machine
GPIO	General Purpose Input/Output
RAM	Random Access Memory
CNC	Computer Numerical Control

HDMI	High Definition Multimedia Interface
VGA	Video Graphics Array
VNC	Virtual Network Computing
SD	Secure Digital
MB	Megabyte

Chapter 1

Introduction

1.1 Overview

One of the most challenging problems in a control system is balancing a system. There are many examples of control systems like inverted pendulums, ball and beam systems, and magnetic levitation. The challenge here is to balance these systems as desired. The ballbot system is a promoted version of the ball and beam system that is 1-dimensional, whereas the ballbot system is 2-dimensional. The challenge (the main aim of the robot) is to balance a ball on the plate.

There are two common classes of control systems: open-loop and closed-loop. In open-loop systems, the output is generated based on the inputs but in the closed-loop (also called a feedback system) the current output is taken into consideration and corrections are made based on feedback. The robot system is unstable if left as an open-loop system, so the ballbot uses a closed-loop system to be stable. That is; the ball will run away if the plate is not in the horizontal state unless the plate corrects its angle to the horizontal plane. The system includes a rigid plate with a ball rolling freely on the plate, two motors to control the plate, and a camera to detect the position of the ball and give feedback to the system.

1.2 Project Aims and Objectives

The main aim of this project is to design and implement a ball-balancing robot that can maintain a stable ball position on the plate, rejecting position disturbances. The initially-horizontal plate will lean along two horizontal axes in order to control the position of the ball. Each axis will be operated by a motor independently. Each motor will be controlled using a control algorithm. The position of the ball on the plate will be detected by a camera.

1.3 Problem Statement

When we put a ball on a surface, it falls, so we need to balance this ball. The importance of this is to prevent the fall of objects. It is helpful to those who want to balance something specific, such as the game of labyrinth, inverted pendulum problem, and many other applications. To solve the problem we will use a microcontroller, motors, a proportional integral derivative (PID) algorithm, and a camera.

1.4 Project Requirements

The Project requirement specifications are as followed:

- The system should keep the ball balanced.
- The ball-balancing robot is stationary (it has no wheels to move from one place to another).
- The plate should be able to tilt in the X, zero, and Y directions.
- The system should be a stand-alone or an autonomous (not connected to a computer or other systems and should take its decisions by itself).
- The system time is real-time (The processing time and response time are within one minute to allow centering of the ball).
- The ballbot system should be stable (rejecting position disturbances).

1.5 System Description

The ball-balancing robot system holds a ball in balance on a 2-axis tiltable plate. The system block diagram is shown in Figure 1.1. It consists of a camera that will be included in the system, connected to a controller for ball position detection. Finally, a controller actuates the pair of motors to stabilize the ball over the plate.

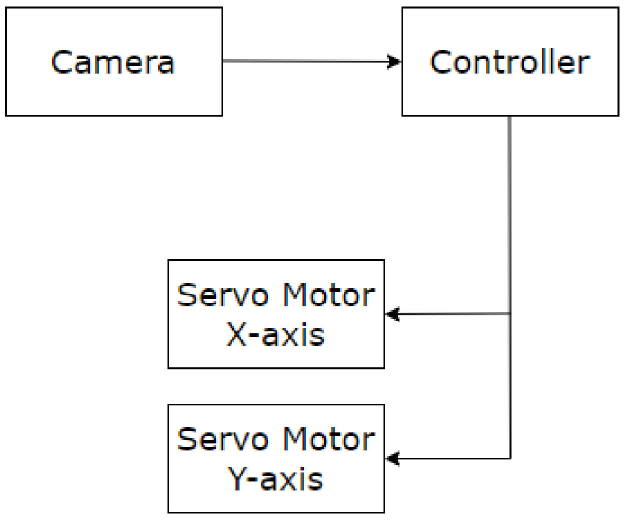


Figure 1.1: System block diagram

1.6 Project Limitations/constraints

The Project Limitations and constraints are as follows:

- The acrylic plate has a 30-centimeter length and 30-centimeter width, and the ping pong ball has 2.7 grams weight.
- The weight of the acrylic plate (325 grams) and the ball (2.7 grams) are within the load that the two motors can handle and move (each servo motor has 11 kg/cm torque).
- The diameter of the ping pong ball (40 mm) is smaller than the acrylic plate diameter (420 mm).
- The color of the ping pong ball (orange) is different from the color of the plate (white) to ease the detection of the ball.

1.7 Project Schedule

In Table 1.1 the project schedule is shown.

Table 1.1: Project schedule

Task / Number of weeks		**1-2**	**3-4**	**4-5**	**5-6**	**9-12**	**12-16**
T1	Project Idea		■				
T2	Project Analysis			■			
T3	System Design				■		
T5	Implementation phase					■	
T6	Testing	■					
T7	Documentation						■

1.8 Report Outline

This report is organized as follows: Chapter one introduced the problem statement and the main idea of our project, project aims and objectives, project requirements, system description, and project Limitations and constraints. Chapter two goes briefly over a preface, theoretical background, and introduces some literature review including available balancing robots and related projects. Chapter three will show the system design including the schematic diagram, a detailed block diagram, hardware, and software components of the system, design alternatives, and pseudocode for the algorithms that will be used in our project. And chapter four will be a conclusion and future work for our project.

Chapter 2

Theoretical Background

2.1 Preface

This chapter briefly describes the theoretical background and the literature review of the project.

2.2 Theories

This section introduces two separate theoretical areas of interest. The image processing models and PID controller.

2.2.1 Image processing

The color image in digital form is represented by a three-dimensional matrix of pixels where the dimensions correspond to the height and width of the image. Also, it can be represented using different color models such as red, green and blue (RGB), Gray; cyan, magenta, yellow, and black (CMYK), or hue, saturation and value (HSV). The values of pixels in the image vary depending on the color model used to represent it.

HSV is a color model used to represent digital images, in which each pixel is represented using three values: hue, saturation, and value. Hue value is used to indicate the primary color whose value ranges from 0 to 360, whereas saturation is a measure of the degree of intensity of color ranges from 0 to 100, and value is a measure of the amount of brightness of the color whose value ranges from 0 to 100.

In our project, We will use image processing to locate the ball by taking a video of it. Each frame will be converted from the RGB color model into the HSV color model before being processed. The HSV is more robust to illumination changes of color than the RGB color model, making it a more attractive option in color thresholding applications compared to RGB [1].

2.2.2 PID controller

Is a feedback-based control loop mechanism that is frequently employed in industrial control systems and various other applications that call for constantly modulated control. With respect to the desired setpoint (SP) and a measured process variable (PV), a PID controller constantly calculates an error value e(t) as the difference between a desired setpoint SP = r(t) and a measured process variable PV = y(t), e(t) = r(t) - y(t), and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D, respectively), the controller attempts to minimize the error over time by adjustment of a control variable u(t). Figure 2.1 shows the block diagram of a PID controller in a feedback loop.

A PID controller continuously calculates an error and applies a corrective action to resolve the error; in this case, the error is the motor spinning at the wrong angle and the corrective action is changing the power to the motor, there are two motors so there are two angles. It is this continuous testing of the motor's angle and adjusting it to the correct angle which will make motors spin at the correct angle and go straight. It will calculate the difference (or error) between the target angle and the actual angle and apply an adjustment to the motor angle. If the adjustment overshoots the next time, a smaller opposite adjustment will be made. Over time, the adjustments will be equal, and the motors will run at a constant angle [2].

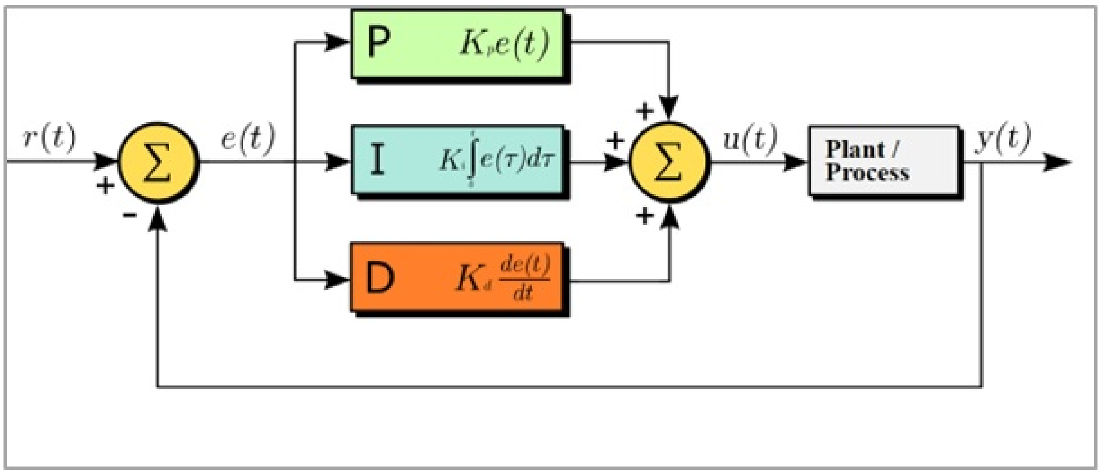


Figure 2.1: A block diagram of a PID controller in a feedback loop [3]

r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV).

2.3 Literature Review

Many studies were presented in the literature about the ball and plate system. These studies differ in the types of controls and the way of reading the ball position. However, some of the recent research on the ball and plate system is reviewed in this section.

1. Ball and Plate Balancing System (Anas Qasrawi and Yazeed Natsheh, 2018) [4]:

   They described the design and construction of a ball and plate system. Their system consists of a plate that leans along two horizontal axes, Arduino Uno, a resistive touch screen to measure the ball position on the plate and servo motors. They used the Matlab program and Simulink toolbox to evaluate the response of the closed-loop system and go from the non-linear equations to the linear model and simulation. The balancing mechanism in their system is that the touch screen will read the application of pressure by a steel ball and track it as it moves then sends the ball position to Arduino that run pid controller and send instructions to servos to make them move to keep the ball balanced, As feedback in their system, the ball position feedback is accomplished through the usage of the resistive touchscreen. The objective of the project was to control the position of the ball on the plate for static positions and build a system that is capable of correcting for disturbances in the ball position, and they achieved these goals.

2. Control of a Ball and Plate System Using Model-Based Controllers (Firas Al-haddad, 2020) [5]:

   In this thesis, a ball and plate system is developed, tested, and controlled. The system consists of a touch screen to determine the ball position, servo motors to change the angles of the plate, and the interfacing circuits that connect the touch screen and servo motors with the Arduino Uno controller. The aim was to compare the performance of the five different controllers for balancing a freely rolling ball in a specific position or moving it in a circle or square trajectory on the plate with the smallest settling time and the least possible error. With this ball and plate system, five different control strategies were applied. The balancing mechanism and feedback are the same as the ones in [4]. The controllers have successfully stabilized the ball on the center of the plate with different performances and some controllers track the predefined trajectory.

3. Design and development of a ball-plate balancing system with a smart phone human-machine interface [6]:

   This paper presents the design and mathematical model of two axes ball-plate balancing system. Their system consists of a plate, two servo motors that actuate to control the roll of the plate, a camera to read the ball position in real-time, Arduino UNO to issue the desired actuating signals to motors to make them move, and a dedicated mobile application forms an intuitive human-machine interface that is developed to control the operation of the balancing system and provides the operation via a smartphone from anywhere. They used SolidWorks software to build the computer-aided design of their system. The balancing mechanism in their system is that the web camera is connected to the computer through a universal serial bus (USB) focusing the plate from the top and reading the position of the ball on the plate. An interactive smartphone-based is developed to give the user input command, to set the ball on the plate at any desired location and also display the position of the ball on the plate at any time. A PID control algorithm is implemented to control the ball position on the plate, to the center of the plate by default, or to the desired user input position as read. The control signal for the required motor angle, to adjust the roll and pitch of the plate is sent to Arduino via serial communication.

4. Design and control of ball-on-plate balancing dynamic system using pid controller [7]:

   The main goal of this research is to keep a ball on a platform within a predetermined point by using a PID algorithm, They designed a dynamic model for their system using SolidWorks software, Their system consists of a touch screen, Aluminium Plate, Arduino linked with the LabVIEW simulator to display the response and control the system, two servo motors. As a balancing mechanism, they use an analog resistive touchscreen sensor that identifies the position of the ball on the plate and send the position to Arduino. The Arduino runs the system and applies the PID algorithm, Consequently, the servo motors as actuators will tilt the plate in the proper inclination to put the ball in the specified position, One is for the x-axis and another is for the y-axis, Each motor is controlled by a PID controller. The feedback of the ball position is accomplished through the analog resistive touchscreen. Their system was able to handle the disturbance and place the ball in the specified place in the shortest possible time with less error.

The differences between our project and other projects are shown in Table 2.1

Table 2.1: Comparison between ballbot and other projects

Project/ Comparison	Ball Detection Method	Processing node	Software	Class of control system
Ballbot	Using camera feedback	Raspberry Pi	Python and OpenCV	Closed-loop
Ball and Plate Balancing System [4]	Using resistive touch screen	Arduino Uno	Matlab and Simulink toolbox	Closed-loop
Control of a Ball and Plate System Using Model-Based Controllers [5]	Using resistive touch screen	Arduino Uno R3	Matlab and Simulink toolbox	Closed-loop
Design and development of a ball-plate balancing system with a smart phone human-machine interface [6]	Using camera	Mobile Application and Arduino UNO	SolidWorks	Closed-loop
Design and control of ball-on-plate balancing dynamic system using pid controller [7]	Using resistive Touch Screen Sensor	Arduino	SolidWorks and LabVIEW	Closed-loop

Our project will differ from other projects in the used software, the way of reading the ball position, and the type of used processing node. Our system has a common point with the system in [6] in the way of reading the ball position but it differs completely in the used processing node, our project has Raspberry Pi as a single processing node while the other project has two possessing nodes (mobile application and Arduino UNO).

2.4 Summary

Chapter two introduced the theoretical background in two areas of interest: The image processing models and PID controller; and some literature review including available balancing robots and related projects.

Chapter 3

System Design

3.1 Preface

This chapter contains a description of the hardware and software parts that are used in our project and discusses the overall design of the system and the way its components are integrated together, showing the detailed block diagram and schematic diagram for the design, in addition to some details about the algorithms we are going to use.

3.2 System components

This section contains a description of the hardware and software components used in ballbot.

3.2.1 Software components

- Open Source Computer Vision Library (OpenCV)

  It is an open-source library that contains over 2500 optimized algorithms in computer vision and machine learning and we will use one of these algorithms (HSV-based segmentation) to detect the ball position.

- Python Integrated Development Environment (IDE)

  It is an environment that will be used for writing code and uploading it to the controller.

3.2.2 Hardware components

Raspberry Pi 3 Model B

Raspberry Pi 3 Model B, shown in Figure 3.1, is a small, low-cost, programmable, single-board minicomputer. It is an enhanced motherboard and comes with all the critical features of the motherboard in an average computer without peripherals or internal storage. This minicomputer can connect with other peripheral hardware devices [8]. Raspberry Pi is the processing unit for our system and it will run the image processing algorithm and the PID controller and read/process data from both the camera and two motors.

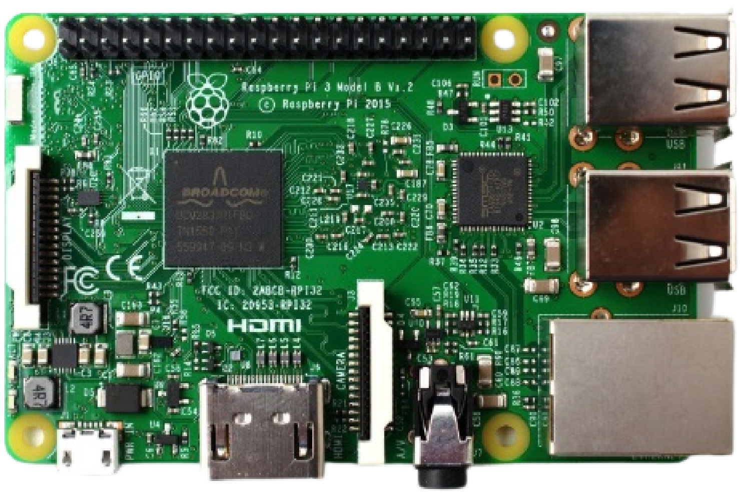


Figure 3.1: Raspberry pi 3 model B [9]

MG996R Servo Motor

MG996R, shown in Figure 3.2, is a metal gears servo motor with a maximum stall torque of 11 kilogram/centimeter. The motor rotates from 0 to 180 degrees based on the duty cycle value of the PWM pin [10]. We will use two motors of this type to tilt the plate in the proper inclination and make it leans along the x and y axes.

Figure 3.2: Mg996R servo motor [11]

Logitech QuickCam Pro 9000

Logitech QuickCam Pro 9000, shown in Figure 3.3, has a native 2-megapixel (MP) high-definition (HD) sensor that can capture HD 720-pixels video. Also has autofocus technology that allows getting as close as 10 centimeters to the camera, in addition to RightLight 2 technology that gives clear video record enhanced performance in a low-lit environment [12]. The camera will be used to detect the ball position on the plate and give feedback to the processing node.

Figure 3.3: Logitech quickcam pro 9000 camera [13]

3.3 Design options

This section contains a comparison between the hardware components that will be used in this project and alternatives (if available), it also contains a brief justification for certain choices.

Processing node:

Table 3.1 shows the differences between the Raspberry Pi 3 model B that will be used and its rivals (Banana Pi M3 and NanoPi Neo Plus2).

Table 3.1: Comparison between raspberry pi 3 and the rivals

Comparison	**Raspberry Pi 3 Model B [9]**	**Banana Pi M3 [14]**	**NanoPi neo plus2 [15]**
Random Access Memory (Ram) Size	1 Gigabyte (GB) LPDDR2-900 SDRAM	2GB LPDDR3 (shared with graphics processing unit)	1GB DDR3
CPU type / speed	Advanced RISC Machine (ARM) Cortex-A53 1.2 Gigahertz (GHz)	ARM Cortex-A7 1.8 GHz	ARM Cortex-A53 1.2GHz
USB	4 × USB 2.0 ports	2 × USB 2.0 PORT 1 × USB Outside Temperature Gauge	2 x Independent USB Host
Power	5V DC /2.5A via MicroUSB or General Purpose Input/Output (GPIO) header	5 volt /2A via DC Power and/or Micro USB (OTG)	DC 5V / 2A via MicroUSB or pin headers
GPIO	40 Pin extended GPIO	40 Pin	36 Pin

Raspberry pi 3 model B was our choice as a processing node because the clock speed and RAM size are sufficient to run the ball detection algorithm and PID, and contains 4 USB ports which we need in our project because we only have one processing node where the camera and two servo motors will connect with it, so we need this number of ports. Finally, it is available in the local market while others are not.

Actuator:

Table 3.2 shows the differences between the MG996R servo motor that will be used and its alternative (Stepper Motor Nema 17).

Table 3.2: Comparison between mg996r servo motor and its alternative

Comparison	MG996R Servo Motor [10]	Stepper Motor NEMA 17 [16]
Weight	55 grams	240 grams
Temperature range	0 ºC to 55 ºC	-10 °C to 55 °C
Current	2.5 A (6V)	1.2 A (4V)
Rotation	0°-180°	360º (1.8º angle per step)

MG996R Servo Motor was our actuator of choice since it doesn't need extra circuitry to run while the stepper needs a driving circuit moreover it has a lighter weight than the stepper and it has the rotation angle we need.

Camera:

Table 3.3 shows the differences between the logitech quickcam pro 9000 camera that will be used and its alternative (Astrum 720p hd usb webcam).

Table 3.3: Comparison between logitech quickcam pro 9000 camera and the rival

Comparison	Logitech QuickCam Pro 9000 [17]	Astrum 720p Hd USB Black Webcam [18]
Interface	USB 2.0	USB 2.0
Microphone	Built-in	External
Frame Rate	30 Hertz	30 Hertz
Video Resolution	1600 x 1200 Pixel	1920 x 1080 Pixel

We chose logitech quickcam pro 9000 camera because it has suitable Video Resolution.

3.4 Conceptual system description

The detailed block diagram of the system is shown in Figure 3.4. It gives an overview of the system's main components and how the system works. The camera detects the position of the ball on the movable plate and sends feedback to the Raspberry Pi that makes some processing and actuating the pair of motors therefore, the ball will be stabilized over the plate.

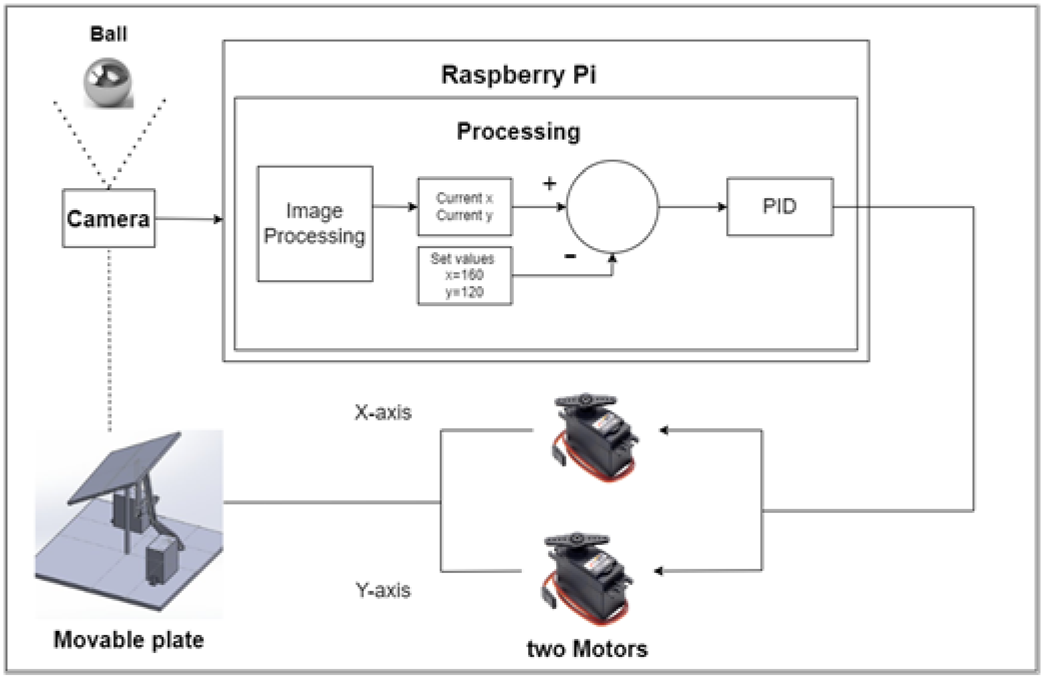


Figure 3.4: System detailed block diagram

3.5 Algorithms and Methodologies

Algorithm 1 shown below runs on the system. It contains a loop for capturing a video of the ball, after that the processor of the Raspberry Pi processes the image and detects the position of the ball, then the Raspberry pi controls the servo motors movement.

Algorithm 1 Ballbot system

```
 1: Setup Part
 2: INITIALIZE Raspberry pi GPIO pins
 3: INITIALIZE Camera
 4: INITIALIZE Two Servo motors and set the angle for both of them to zero
 5: Loop Part
 6: while capturing image do
 7:     PROCESS the image
 8:     LOCATING the ball position
 9:     MOVING the servo motors at a specific angle
10: end while
```

Algorithm 2 shown below is meant to detect the ball and its position. First, we convert the image from RGB to be represented in the HSV color model. Then apply a threshold to the image to isolate the ball from other objects in the environment based on its color (the threshold range is determined manually in setup mode attempting different values for the threshold's minimum and maximum values and selecting the best values). As a result of this process, we get a binary image (mask image). Then a Blob detection (used to determine the connected components in the image) is performed on this image, with the largest blob representing the object to be detected.

Therefore, in order to determine the ball position, we calculate the center of gravity (CoG) for the connected component by using the following equations:

$$X_c = \frac{\sum_{i=1}^{N} xi}{N}, Y_c = \frac{\sum_{i=1}^{N} yi}{N}$$

Where xi and yi are the coordinates of an image that have value 1, N is the number of white positions (pixels) in the image, and the point (Xc, Yc) represents the position of the ball in the coordinates.

Algorithm 2 Detect the ball and its position

```
1:  while capturing image from camera do
2:      CONVERT the image from RGB to HSV model
3:      THRESHOLD H and S :
4:      if H belong to [t1-t2] then
5:          Accept
6:      else
7:          Ignore
8:      end if
9:      if S belong to [t3-t4] then
10:         Accept
11:     else
12:         Ignore
13:     end if
14:     MERGE H and S after threshold
15:     CALCULATE the CoG ($X_c = \frac{\sum_{i=1}^{N} xi}{N}$, $Y_c = \frac{\sum_{i=1}^{N} yi}{N}$)
16:     DETECT the position of the ball
17: end while
```

t1 is the lower value of the ball color range within the hue value ranges.
t2 is the higher value of the ball color range within the hue value ranges.
t3 is the lower value of the ball color range within the saturation value ranges.
t4 is the higher value of the ball color range within the saturation value ranges.

Algorithm 3 shown below demonstrates how the PID algorithm works. The variable (e) represents the tracking error, the difference between the desired input value and the setpoint value. This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The output signal (u) just past the controller is now equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error [19].

Algorithm 3 PID controller work

```
1:  Setup Part for PID controller
2:  INITIALIZE Kp
3:  INITIALIZE Ki
4:  INITIALIZE Kd
5:  Loop Part
6:  while read the input do
7:      GET the setpoint value
8:      GET the input value
9:      CALCULATE the error $e(t) = setpoint - input$
10:     CALCULATE $P = Kpe(t)$
11:     CALCULATE $I = Ki \int_0^t e(\tau)d(\tau)$
12:     CALCULATE $D = Kd\frac{de(t)}{dt}$
13:     CALCULATE $u = P + I + D$
14: end while
```

3.6 Schematic diagrams

The schematic diagram shown in Figure 3.5 represents the components of the system and their connection with the Raspberry pi via analog output pins. The Raspberry pi 3 model B is providing PWM signal to the servo motors to control the angle changes.

The connection of the schematic diagram shown in Figure 3.5 will be as the following:

1. Using an AC-to-DC 5V adapter with a micro USB connector to power Raspberry pi.
2. The PWM lines for Servo motors 1 and 2 are connected to pins GPIO18 and GPIO12.
3. The servo motor's power supply input will be connected to a 5V power supply.
4. The servo motor's ground input will be connected to the negative pole of the power supply.
5. The USB cable of the camera will be connected to USB port in Raspberry pi.

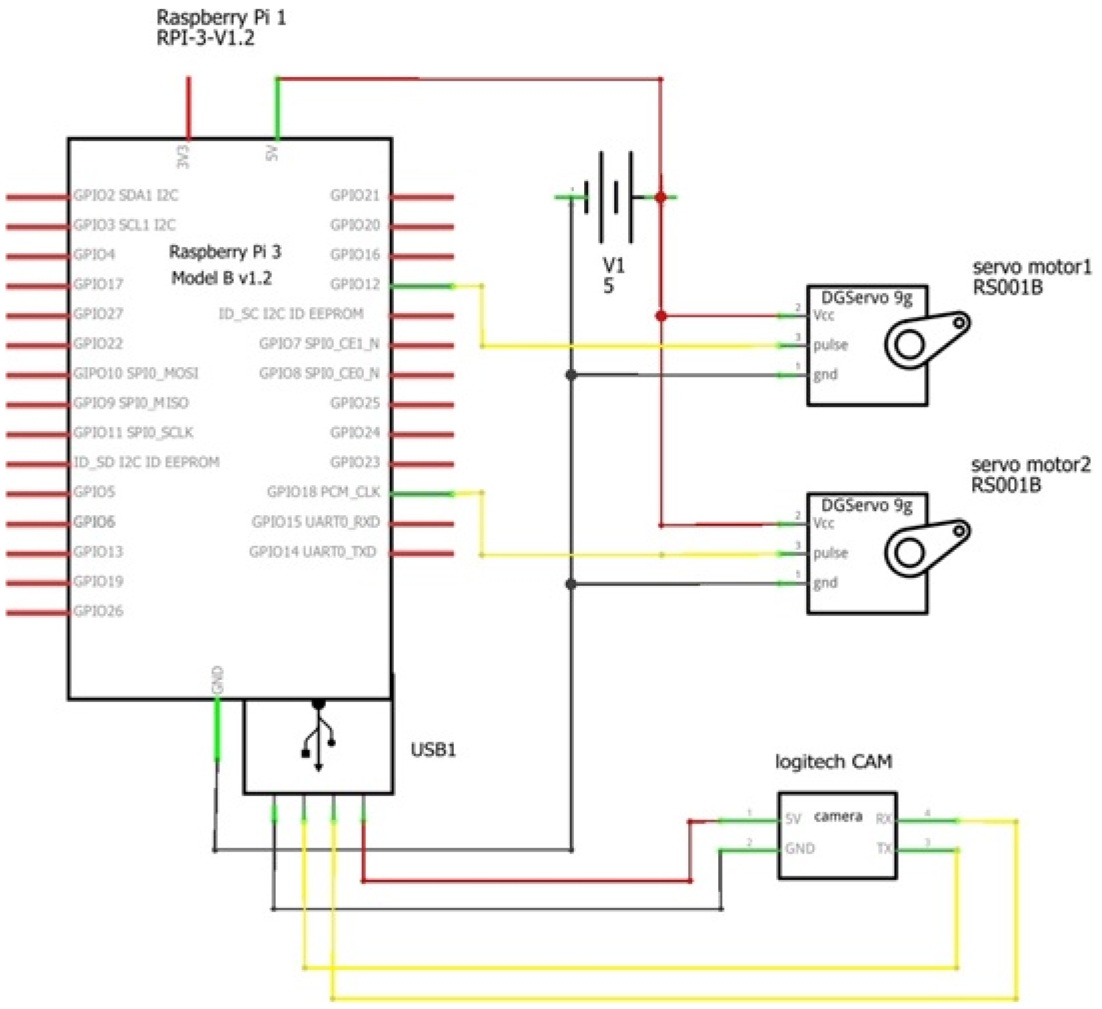


Figure 3.5: The schematic diagram of the system

3.7 Summary

Chapter three introduced the system components in two areas: Hardware and software; it discussed the design options, showing the diagrams including a detailed block diagram and schematic diagram, in addition to some details about the algorithms used.

Chapter 4

Implementation

4.1 Preface

This chapter describes the implementation part of our project in more detail. It delves into the different hardware components of the system and its software with all of its modules.

4.2 Hardware Implementation

The steps for the hardware implementation are as follows:

- We printed a 3D “ ball and socket joint ” shown in Figure 4.1, which consisted of two pieces, one that was fixed to the plate, and the other that contained the ball was fixed to the base of a wooden stand, so that the plate moved smoothly, Figure 4.2 shows how the joint linked with the plate.

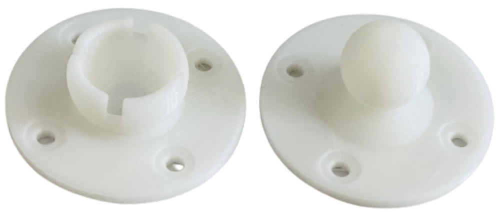


Figure 4.1: Ball and socket joint

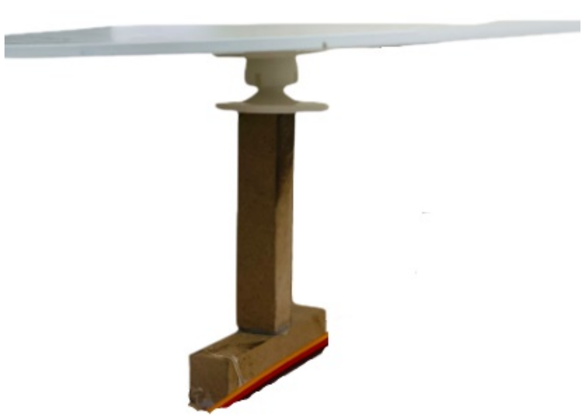


Figure 4.2: The link between the plate and the ” ball and socket ” joint

- Installation of each of the servo motors:

  First, the servo motors bracket was printed using a 3D printer as shown in Figure 4.3 to contain the servos and prevent friction with the base, one of the brackets was fixed to the base on the x-axis, and the other on the y-axis.

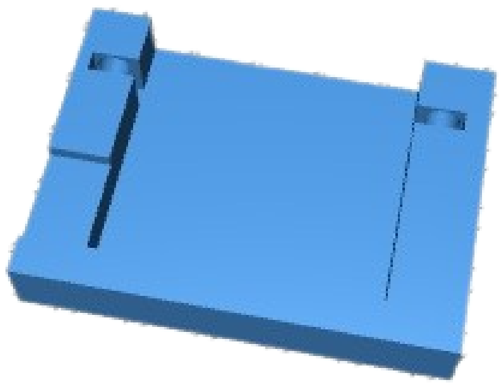


Figure 4.3: Servo motors bracket [20]

  Second, the servo motor’s arms were 3D printed for each motor to install with the plate as shown in Figure 4.4.

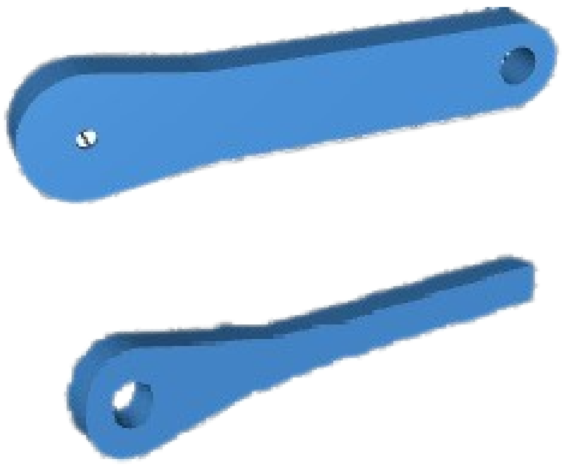


Figure 4.4: Servo motor's arms [20]

Finally, by using the universal joint with a 3-millimeter inner diameter as shown in Figure 4.5, the servo motor's arms were fixed with the plate as shown in Figure 4.6.

Figure 4.5: Universal joint with a 3-millimeter inner diameter

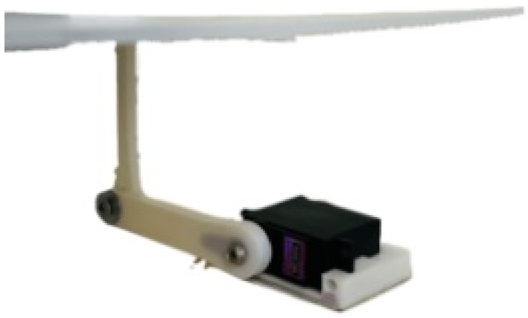


Figure 4.6: The link between the plate and the servo motor

- The logitech quickcam pro 9000 camera was installed on the stand in a suitable position, where the camera was placed at a distance of 60 cm toward the center of the movable plate as shown in Figure 4.7, which allows the plate to be completely in the camera's field of view, also it was not placed at a longer distance to not capture the surrounding objects.

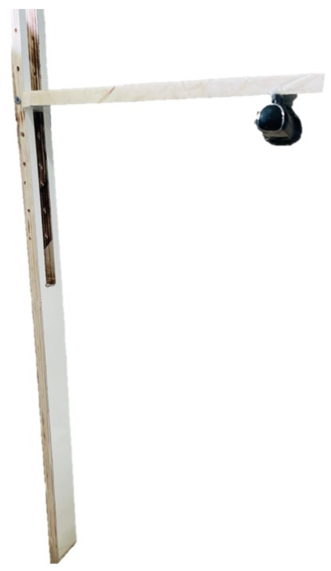


Figure 4.7: The logitech quickcam pro 9000 camera installed on the stand

- The final step was to connect the circuit we need to control the camera, and the servo motors through the raspberry pi 3 model b. Figure 4.8 shows the main circuit of the system.The camera was connected to the raspberry pi through a USB cable. For the ground and the power of the servo motors, we connected them to a breadboard, and then we connected the power of the breadboard to pin 2 in the raspberry pi and the ground to pin 6. The signal of each of the two servo motors was connected to PWM pins 12 and 32 in raspberry pi.

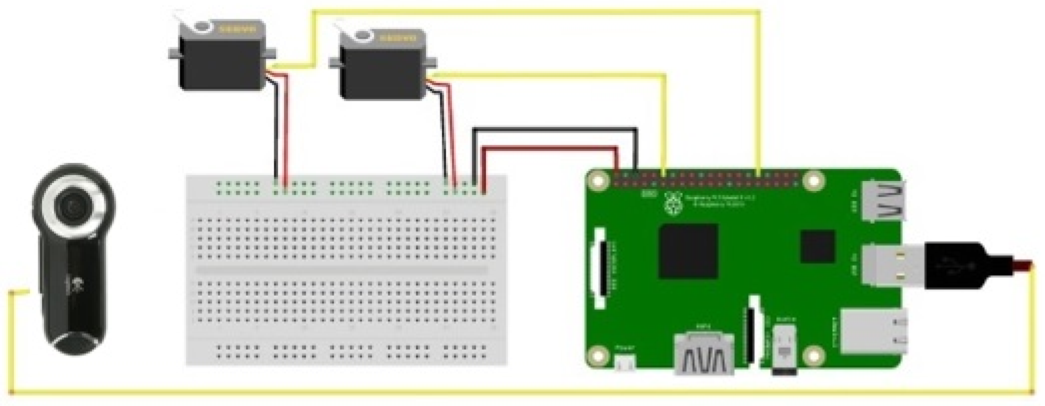


Figure 4.8: The main circuit of ballbot system

The system after the overall hardware implementation is shown in Figure 4.9.

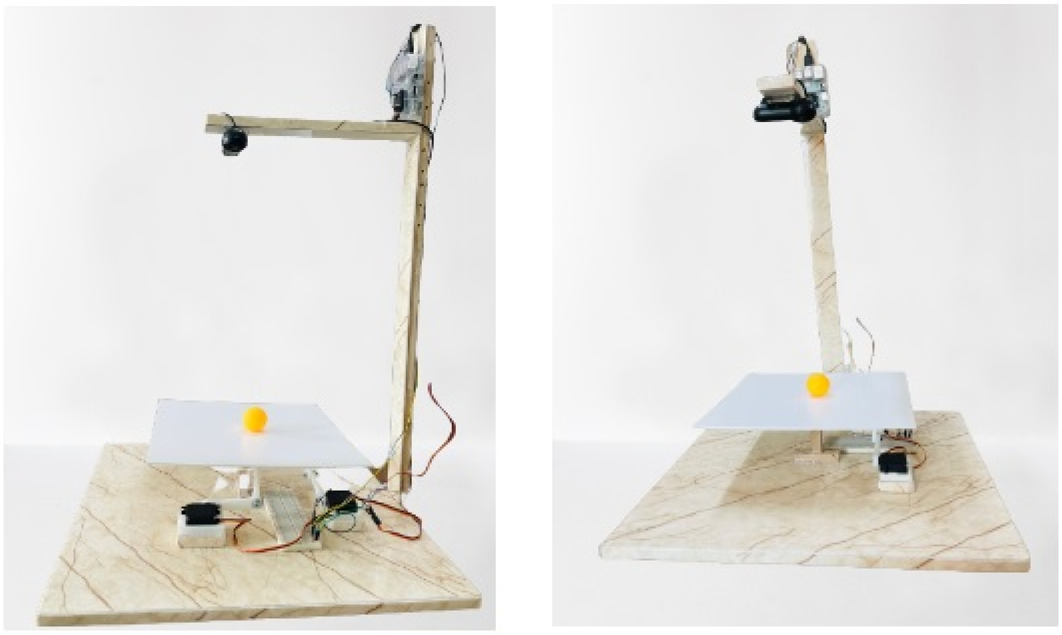


Figure 4.9: The system after the overall hardware implementation

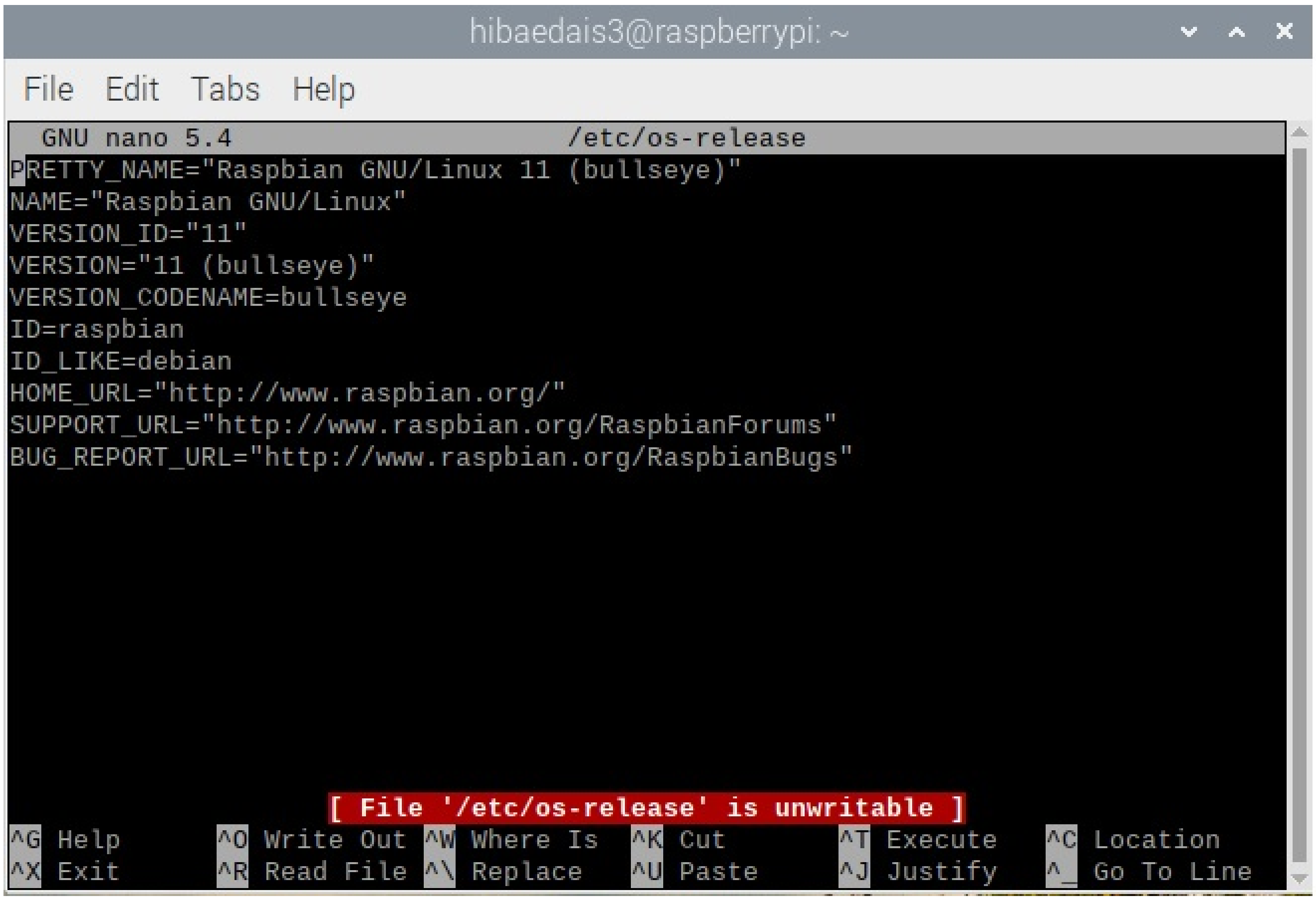
4.3 Software Implementation

This section describes the implementation details of the software components of the system.

4.3.1 Raspberry pi operating system installation

We downloaded an operating system to the SD card and then installed it on the Raspberry Pi by following these steps:

- Downloading the operating system image file and extracted it.
- Took an SD card holder, inserted it into the computer, formatted it, and ran the imager.
- Inserte the SD card into the Raspberry Pi and turned it on.
- We have seen the version shown in Figure 4.10 of the operating system by entering the command ” /etc/os-release ” in the terminal



```
  GNU nano 5.4                          /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 11 (bullseye)"
NAME="Raspbian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"

                    [ File '/etc/os-release' is unwritable ]
^G Help         ^O Write Out ^W Where Is  ^K Cut         ^T Execute   ^C Location
^X Exit         ^R Read File ^\ Replace   ^U Paste       ^J Justify   ^_ Go To Line
```

Figure 4.10: Operating system version

4.3.2 Camera configuration

We configured the camera on the Raspberry Pi by following these steps:

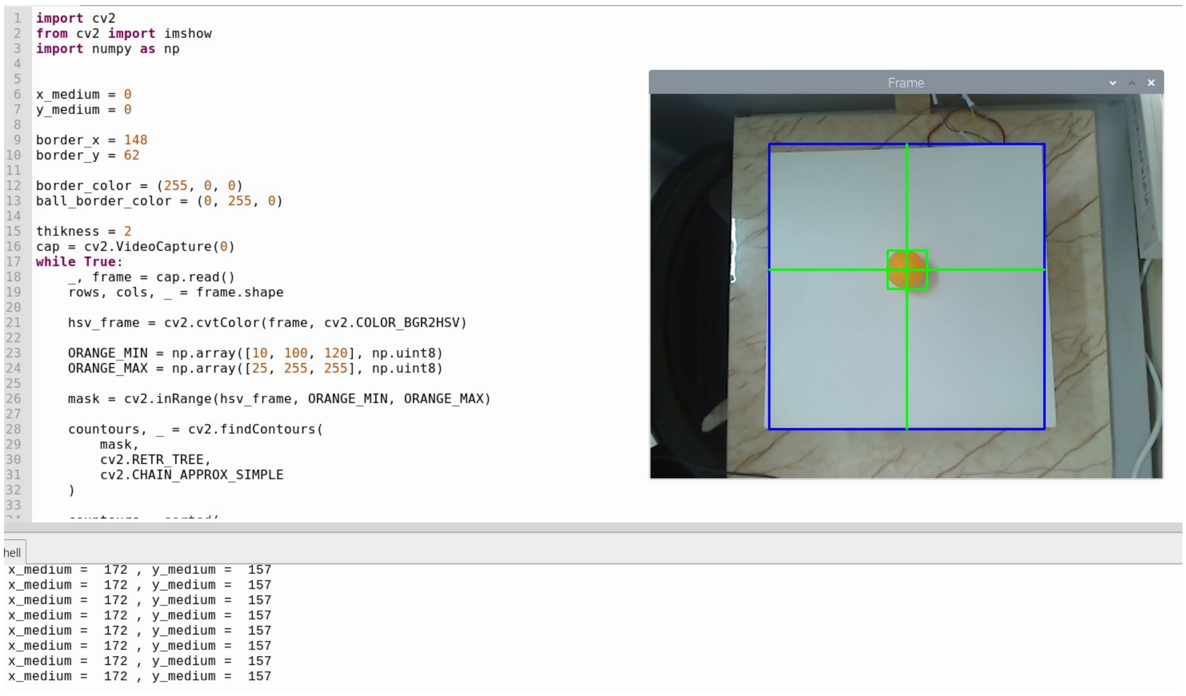
1. We have made sure that the camera is connected to the Raspberry Pi.
2. We opened terminal and entered the command ”sudo raspi-config”.
3. We selected ”Interface Options”.
4. We selected ”CameraLegacy” and press Enter.
5. We selected ”Yes” to confirm.
6. Then we clicked on “Finish”.

4.3.3 HSV implementation

- We installed the opencv library using the command “ sudo install opencv-python”.
- Then imported the opencv library into our code using the ’import cv2’ command.
- A video was captured with the command “cap = cv2.VideoCapture(0)”.
  The image was loaded into the HSV color space with the command “hsv_frame= cv2.cvtColor(frame, lcv2.COLOR_BGR2HSV)”.

- Setting the required range for the orange ball color by setting the HSV boundary values. We did this by creating a NumPy array containing the minimum and maximum HSV values for the orange color using the command “ORANGE_MIN = np.array([10, 100, 120], np.uint8) “ and “ORANGE_MAX = np.array([25, 255, 255], np.uint8)”.

- Using the command ”cv2.inRange(hsv_frame, ORANGE_MIN, ORANGE_MAX)” to get the final image containing the selected colors, And we used the command “cv2.findContours” and “cv2.boundingRect( )” to determine the center of the ball. Figure 4.11 shows the HSV algorithm after implementation.



```python
import cv2
from cv2 import imshow
import numpy as np


x_medium = 0
y_medium = 0

border_x = 148
border_y = 62

border_color = (255, 0, 0)
ball_border_color = (0, 255, 0)

thikness = 2
cap = cv2.VideoCapture(0)
while True:
    _, frame = cap.read()
    rows, cols, _ = frame.shape

    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    ORANGE_MIN = np.array([10, 100, 120], np.uint8)
    ORANGE_MAX = np.array([25, 255, 255], np.uint8)

    mask = cv2.inRange(hsv_frame, ORANGE_MIN, ORANGE_MAX)

    countours, _ = cv2.findContours(
        mask,
        cv2.RETR_TREE,
        cv2.CHAIN_APPROX_SIMPLE
    )

```

```
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
x_medium =  172 , y_medium =  157
```

Figure 4.11: The hsv algorithm after implementation

4.3.4 Mathematical system modelling

The dynamical system model 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 [21]. Newtonian mechanics will be used in this project.

In order to achieve the equations of motion for a dynamical system consisting of a ball on a platform the following have to be assumed:

- No friction is considered.
- The geometry of the ball is perfectly spherical and homogeneous.
- The ball is rolling and not slipping on the platform.

Equations of motion

The equation for the absolute acceleration of the ball is given by [22].

$$a_a = w * r + w * (w * r) + 2w * v_{rel} + a_{rel} \tag{4.1}$$

Where w is the ball's angular velocity vector, r is the radius of the ball, a is the absolute acceleration of the ball over the x- and y-axis, and v is the velocity. The above can be rewritten as follows for one of the two-dimensional ball-on-beam systems [23].

$$a_1 = \alpha_1 e_{k1} * x_p e_{i1} + \alpha_1 e_{k1} * (\alpha_1 e_{k1} * x_p e_{i1}) + 2\alpha_1 e_{k1} * x_p e_{i1} + x_p e_{i1} \tag{4.2}$$

As shown in Figure 4.12, α_1 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 [23].

$$a_1 = (x_p - x_p {\alpha_1}^2)e_{i1} + (x_p \alpha_1 + 2\alpha_1 x_p)e_{j1} \tag{4.3}$$

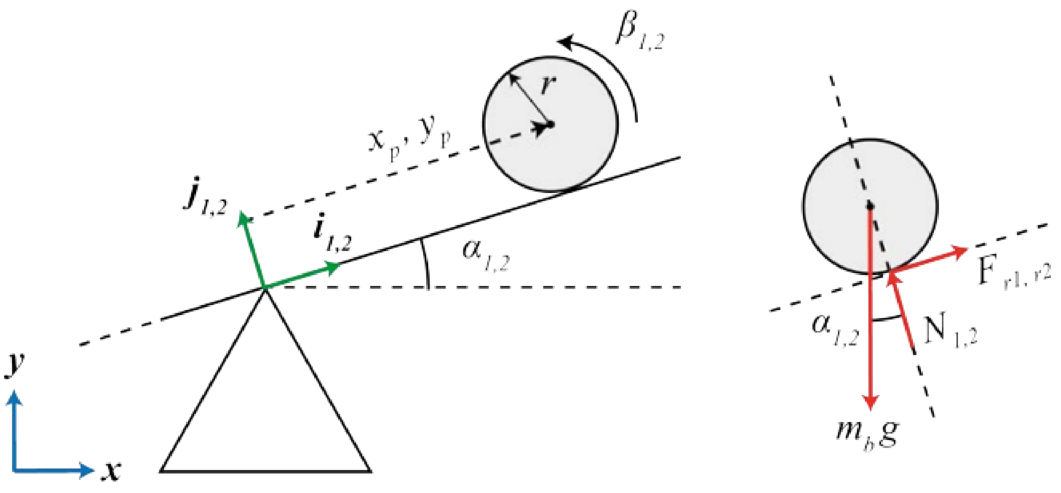


Figure 4.12: Two-dimensional representation of the system and free body diagram of the ball [23]

From the equilibrium of torques in the free body diagram shown in Figure 4.12 it is possible to derive the remaining forces on the ball [23].

$$I_b \beta_1 = F_{r1} r \tag{4.4}$$

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 regard to the assumption of no slip on the platform, the relative angle β can be defined by the position as follows [23]:

$$\beta_1 = -\frac{x_p}{r} \tag{4.5}$$

In order to solve (4.4) for F_r, the second time derivative of equation (4.5) is combined with (4.4). Resulting in equation (4.6) as stated below [23].

$$F_r = -\frac{I_b x_p}{r^2} \tag{4.6}$$

The equilibrium of forces exerted on and by the ball parallel to the platform given by the acceleration in equation (4.3) and the force in (4.6) results in the two principal equations of motion of the dynamic system [23].

$$(\frac{I_b}{r^2} + m_b)x_p + m_b g \sin\alpha_1 - m_b x_p {\alpha_1}^2 = 0 \tag{4.7}$$

$$(\frac{I_b}{r^2} + m_b)y_p + m_b g \sin\alpha_2 - m_b y_p {\alpha_2}^2 = 0 \tag{4.8}$$

Where m_b is the mass of the ball, y_p is the position of the ball relative origin on the y-axis, and g is the gravitational constant. By rearranging the equations of motion the following equations suitable for Laplace transformation is achieved [23].

$$x = \frac{m_b {r_b}^2 (x_p \alpha_1^2 - g \sin\alpha_1)}{m_b {r_b}^2 + I_b} \tag{4.9}$$

$$y = \frac{m_b {r_b}^2 (y_p \alpha_2^2 - g \sin\alpha_2)}{m_b {r_b}^2 + I_b} \tag{4.10}$$

Linearization

The dynamical ball on the platform system can thereafter be described by two nonlinear differential equations, one for each axis. the system needs to be linearized around a working point. The desired working point of the apparatus will be based in the center of the platform. Hence, equations (4.9) and (4.10) 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 [23].

$$x = \frac{m_b g \alpha_1 r^2}{m_b {r_b}^2 + I_b} \tag{4.11}$$

$$y = \frac{m_b g \alpha_2 r^2}{m_b {r_b}^2 + I_b} \tag{4.12}$$

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 the radius of the ball [23].

$$x = \frac{3}{5} g \alpha_1 \tag{4.13}$$

$$y = \frac{3}{5} g \alpha_2 \tag{4.14}$$

4.3.5 PID implementation

After writing the PID equations based on the mathematical model we have implemented a specially designed graphical user interface screen to control the PID parameters. First, we select the motor number then choose the PID parameters value using the sliders, and choose the error rate initially for each motor as shown in Figure 4.13.

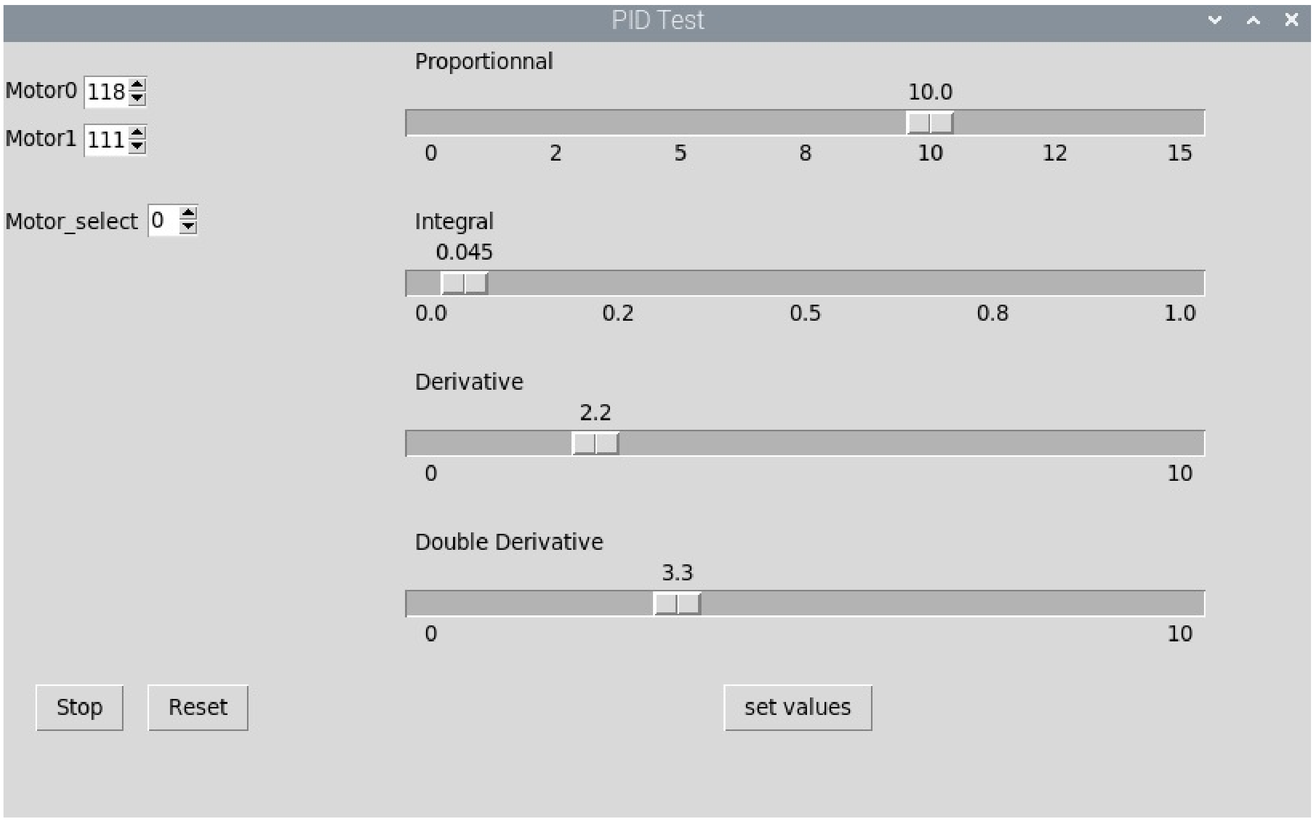


Figure 4.13: Pid implementation

4.3.6 Coding

In this section, we'll describe libraries' Installation, set-up, and hardware functions on the Raspberry Pi.

Libraries installation

In order for the system to start working, we needed several libraries to be installed on the operating system of the raspberry pi. The image we used had every necessary library preinstalled, including OpenCV, and the Pigpio library for the servos.

Opencv library was installed to use the HSV algorithm by the command shown below :
```
sudo apt-get install python-opencv
```

Pigpio library was installed to reduce the servo motors' jitter by the command shown below :
```
sudo apt-get install python3-pigpio
```

We used the pigpiod utility of the pigpio library that runs it as a daemon. It allows the library to run in the background, so before we run the exe program, the daemon should already be running.

This requires sudo privileges [24].

Set-up

- Initialize the two servos.
- Initialize the capture.
- Connect to the pigpio daemon.

Loop

- In the loop function, The camera starts capturing the video (30 frames per second) and then the process of localizing the ball is done using the HSV color space to determine the position of the ball on the plate. This is done by converting the video frame into an HSV color space. The loop is used to find the largest rectangle containing the sphere, calculate the center of the sphere, and determine its desired position. This process is repeated for each frame to keep track of the ball's position and to continually update the ball's position on the screen.
- The PID loop is then used to determine the angle of the motors required to keep the ball balanced based on its position. This is done by measuring the angle of the existing motors and calculating the error between them and the required angle to center the ball. The loop is used to update the signal sent to the actuators based on the measured error values and to control the PID parameters to achieve the desired balance. This process is repeated every frame to keep the ball continuously balanced.

4.4 Implementation Issues and Challenges

The issues and challenges that we faced in the implementation phase are as follows:

- The Raspberry Pi 4 model B 8GB that we are supposed to use is not available in the local market and has a high price, so we used Raspberry Pi 3 model B v1.2 instead.
- The universal joints that are 3 millimeters inner diameter are not available in the local market, we bought them from online website, but they took about three months to arrive, along these three months we faced a problem in the link between the plate and the servo motors, the links were taken off when the servo motors moved.
- The acrylic board is neither available in 30 × 30 centimeters size nor colored board, and it is not easy to cut it. It should be cut using a computer numerical control (CNC) machine. We searched in every library but it was not available. Finally, we found a shop that accepted to cut it for us, but the board is transparent, we solved the problem by finding a carpenter that accepted to spray it with white color.
- Raspberry Pi should be connected with a High-Definition Multimedia Interface (HDMI) cable, but we don't have a computer screen with hdmi or video graphics array (vga) cable in our homes, we tried to connect it with the laptop, it connected successfully using virtual network computing (vnc) but the camera didn't work so we used the television screen for a period of time but when we connected the whole system we couldn't able to use it because we need a screen on a table so we used a computer screen from the university lab.

- Logitech camera c905 has a problem in that Raspberry Pi doesn't recognize it every time, so we used Logitech QuickCam Pro 9000 camera instead of it.
- The pi camera has a short cable and we need a longer cable to put the camera above the system, but the taller cable is not available in the local market so we didn't use it.
- The Raspberry pi gave a notification of low voltage, we changed the charger, the power supply, and the raspberry pi itself but the problem is not solved.
- We tried a Logitech camera c905, a Logitech QuickCam Pro 9000 camera, and a pi camera, but all of them have latency when using a secure digital (sd) card with 16 GB capacity and 80 Megabyte (MB) /second read-write speed.
- Raspberry pi 3 model b was very slow and had a high temperature, we attached a fan, but the problem is not solved, the cameras latency, low voltage notification, and slow speed of Raspberry Pi were solved by using an SD card that has 64 GB capacity, and 170 MB/second read-write speed.

4.5 Summary

In this chapter, the ballbot implementation details are presented, the hardware and software aspects of the project were covered, and how to design and assemble the basic parts of the ballbot were explained, the algorithms used in the project which include the hsv algorithm and pid controller were analyzed and explained. How to program the ballbot using the python programming language is explained. The challenges and issues that we faced in the project and how we dealt with them were discussed.

Chapter 5

Testing and Results

5.1 Preface

In this chapter, we will discuss the testing of all components of the system and the results obtained. We tested all the parts to ensure that all of the functions work as expected and without errors.

5.2 Hardware Testing

This section discusses the testing process of each of our hardware components.

5.2.1 Testing the Raspberry Pi 3

We started testing the Raspberry pi by connecting to it remotely from our laptop via VNC, after that we connected it to a monitor to control it directly.

5.2.2 Testing the logitech quickcam pro 9000 camera

Testing the camera is very easy, so we connected it to a Raspberry Pi and made sure it works correctly by running this code :

```
cam= cv2.VideoCapture(0)
ret,image = cam.read()
cv2.imshow('Image Test',image)
```

5.2.3 Testing Servo motors

We have two servo motors and both of them are connected to the Raspberry Pi. We run many tests on each of our servo motors to make sure they are working properly. We ran the sweep motion code to move the motors from angle 0 to 180, then modified the calibration so that the servo motors were moved to precisely defined angles and the necessary range of motion was determined, The servo motors were shaking “jittering”, and we solved the problem by using the “pigpio” library.

5.3 Software Testing

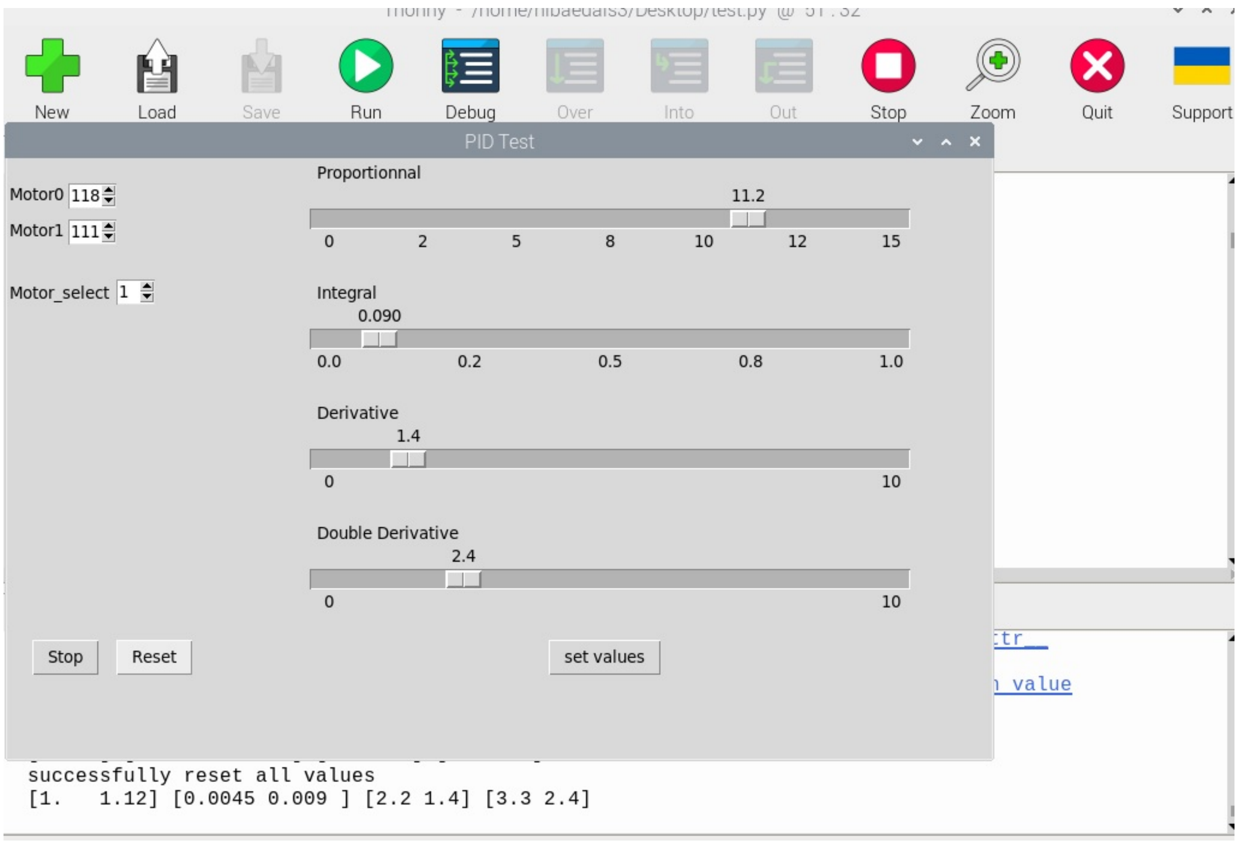
In this section, we will discuss testing of HSV algorithms and PID controllers.

5.3.1 Testing HSV algorithm

We used the HSV algorithm to locate the ball on the plate, where we installed the opencv library in Python. For our ability to use the HSV algorithm, we then ran the code to locate the ball.
We determined the HSV values for the orange color, which is the color of the ball used, by continuing to change the range of h, s, and v until we got the required values, so the range for h was [10-25], for s it was [100-255], and for v it was [120-255].
We also continued to change the boundaries of the X and Y frames of the camera until they fit the boundaries of the plate.

5.3.2 Testing PID controller

The PID controller is tuned by the trial-and-error method using the graphical user interface to achieve the desired response. The gain values for the system (Kp, Ki, and Kd) are so chosen that leads to reduced oscillation so as to maintain stability and lesser settling time. The tuned PID controller gain values for both the x and y axes are shown in Figure 5.1.



Motor0 118
Motor1 111
Motor_select 1

Proportionnal 11.2

Integral 0.090

Derivative 1.4

Double Derivative 2.4

```
successfully reset all values
[1.   1.12] [0.0045 0.009 ] [2.2 1.4] [3.3 2.4]
```

Figure 5.1: Pid controller values for both the x and y axes

5.3.3 Testing the System Altogether

We tested the whole system by operating the system and moving the ball in different directions. The location of the ball was determined using the camera. We changed the different PID values to improve the performance of the system and achieve a better balance for the ball. After completing the test, the results can be displayed and analyzed to improve system performance and achieve better ball balance.

5.4 Results

Satisfactory results were achieved after implementing the ballbot system to keep the ball balanced on the center of the plate using a camera, two servo motors, and a PID algorithm. The system uses the camera to locate the ball and sends the signals to the raspberry pi, which uses the pid algorithm to determine the appropriate signals to send to the servo motors. The position of the ball is updated continuously and the servo motors are controlled with high precision to keep the ball balanced.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this project, we built a ballbot that was capable of balancing the ball on the center of the plate, and we successfully achieved the main goal of the project. The HSV algorithm was used to determine the position of the ball on the plate, and appropriate PID parameters were used to control the servo motors and achieve the best possible performance. A graphical user interface (GUI) was designed to facilitate the control of the motors.

6.2 Future Works

The future works for the ballbot project are as follows:

- Develop the ballbot to have wheels to move from one place to another.
- Develop a ballbot that keeps the ball balanced at any point or at any desired location the user wants on the plate.
- Make a ballbot that can be put under the car to keep it balanced when an accident happens.
- Enhance the ballot stability by using more than 2 servo motors.
- Improving the ballbot capabilities to handle one ball or more with different weights, colors, and sizes.
- Ballbot can play a labyrinth game that moves the object from the start point to the endpoint as shown in Figure 6.1.

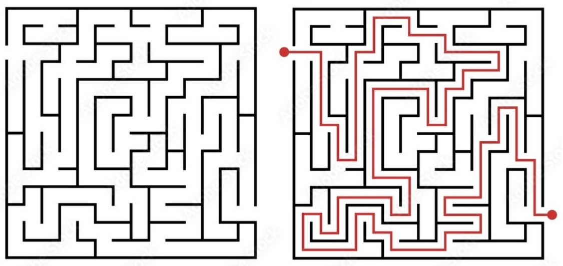


Figure 6.1: Labyrinth game

References

[1] Gonzalez, Rafael; Richard Woods. Basic Edge Detection, “Digital Image Processing”, (3rd ed.), 2008.

[2] Going Straight with PID.Projects.raspberrypi.org. Available at: https://projects.raspberrypi.org/en/projects/robotPID/3 (Accessed: November 11, 2022).

[3] Wikipedia, ”PID controller,” 2022. [Online]. Available: https://en.wikipedia.org/wiki/PID_controller. [Accessed 2022].

[4] Anas Qasrawi and Yazeed Natsheh, ”Ball and Plate Balancing System,” Palestine Polytechnic University, 2018.

[5] F. Al-haddad, ”Control of a Ball and Plate System,” Palestine Polytechnic University, 2020.

[6] G Madhumitha, S Suresh Kumarand A Gurupriya, ”Design and development of a ball-plate balancing system with a smartphone human-machine interface,” Journal of Physics: Conference Series, vol. 1969, no. 012057, 2021.

[7] Aboajela Kajaman,Ezuldeen Aldeeb and Mohamed Aljayar, ”Design And Control Of Ball-On-Plate Balancing Dynamic System Using Pid Controller,” in Conference: International Conference on Technical Sciences, LIBYA, 2019.

[8] C. BasuMallick, ”What Is Raspberry Pi? Models, Features, and Uses,” spiceworks [Online]. Available:https://www.spiceworks.com/tech/networking/articles/what-is-raspberry pi. [Accessed 2022].

[9] M. Rundle, ”Raspberry Pi 3 is the first with built-in Wi-Fi,” WIRED, 29 2 2016. [Online]. Available:https://www.wired.co.uk/article/raspberry-pi-three-wifi-bluetooth-release-price-cost. [Accessed 2023].

[10] Components101, ”MG996R Servo Motor,” 3 April 2019. [Online]. Available: https://components101.com/motors/mg996r-servo-motor-datasheet. [Accessed 2022]

[11] Aliexpress, ”Mitoot MG996R Metal Gears Digital RC Servo Motor High Torque for Rc Airplane Helicopter Car Boat,” [Online]. Available: https://www.aliexpress.com/i/4000075435453.html. [Accessed 2022].

[12] B. photo-video-audio, ”Logitech QuickCam Pro 9000 USB 2.0 Webcam - 2MP,”[Online]. Available:https://www.bhphotovideo.com/c/product/505399-REG/Logitech_960_000048_QuickCam_Pro_9000_USB.html. [Accessed 2023].

[13] newegg, ”Logitech QuickCam Pro 9000 2.0 M Effective Pixels USB 2.0 WebCam,” [Online].Available:https://www.newegg.com/logitech-quickcam-pro-9000/p/N82E16826104074. [Accessed 2023].

[14] Wikipedia, ”Banana Pi,” [Online]. Available: https://en.wikipedia.org/ wiki /Banana _Pi. [Accessed 2022].

[15] F. ELEC, ”NanoPi NEO Plus2,” 30 May 2017. [Online]. Available: https://wiki.friendlyelec.com/wiki/index.php/NanoPi_NEO_Plus2. [Accessed 2022].

[16] M. Industries, ”Stepper Motor Specifications - NEMA 17 1.8 degree 200 steps-per-revolution four-phase unipolar permanent-magnet stepper-motor,” [Online]. Available: http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/stepper-motors/specifications. [Accessed 2022].

[17] ebay, ”Logitech QuickCam Pro 9000 2mp Autofocus Carl Zeiss USB Webcam,” [Online]. Available: https://www.ebay.com/p/1601490290?iid=374266852755. [Accessed 2023].

[18] comx-computers, ”Specifications for WEBCAM-720P - Astrum 720p Hd USB Black Webcam With Mic,” [Online]. Available: https://www.comx-computers.co.za/WEBCAM-720P-specifications-246904.htm. [Accessed 2022].

[19] ”PID Tutorial,” THE UNIVERSITY OF MICHGAN, [Online]. Available: https://www3.diism.unisi.it/ control/ctm/PID/PID.html. [Accessed 2022].

[20] N. Hammje, ”STL_Files,” 2023. [Online]. Available: https://github.com/nicohmje/PID-ballonplate/tree/main/STL_Files. [Accessed 2023].

[21] C. G. Bolivar-Vincenty and G. Beauchamp-Baez, “Modelling the ball-and-beam system from newtonian mechanics and from lagrange methods,” 12th Latin American and Caribbean Conference for Engineering and Technology, 2014.

[22] N. Apazidis, Mekanik II, Partikelsystem, stel kropp och analytisk mekanik,vol. 1:3. Studentlitteratur AB, 2017.

[23] A. HASP FRANK and M. TJERNSTRÖM, ”Construction and theoretical study of a ball balancing platform: Limitations when stabilizing dynamic systems through implementation of automatic control theory,” KTH, School of Industrial Engineering and Management (ITM), STOCKHOLM, SWEDEN, 2019.

[24] “pigpio library,” abyz.me.uk. [Online]. Available: https://abyz.me.uk/rpi/pigpio/ cif.html [Accessed 2023].