VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi,Karnataka - 590 018

A Project Report on

"Ball Balancing PID System using Image Processing"

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

by

Amoghavarsha S G USN:4JN18EC010
Kiran C N USN:4JN18EC041
Niranjana Jois H C USN:4JN18EC057

Under the Guidance of

Mr. Pradeepa S C

Assistant Professor,

Dept. of ECE,

JNNCE-577 204.

Department of Electronics and Communication Engineering

JNN College of Engineering, Shimoga - 577 204.

July 2022

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590 018.

JNN College of Engineering

Department of Electronics and Communication Engineering

Shimoga-577 204.

CERTIFICATE

This is to certify that the project work entitled **"Ball Balancing PID System using Image Processing"** is carried out by **Amoghavarsha S G (4JN18EC010), Kiran C N (4JN18EC041), Niranjana Jois H C (4JN18EC057)** , the bonafide students of JNN College of Engineering, Shimoga in partial fulfillment for the award of "Bachelor of Engineering" in department of "Electronics and Communication Engineering" of the Visvesvaraya Technological University, Belagavi, during the year 2021-2022. It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of the Guide
Mr. Pradeepa S C
Assistant Professor,
Dept. of ECE,
JNNCE, Shimoga.

Signature of the HOD
Dr. S.V. Sathyanarayana
Professor & HoD
Dept. of ECE,
JNNCE, Shimoga.

Signature of the Principal
Dr. K Nagendra Prasad
Principal
JNNCE, Shimoga.

External Viva

Name of the examiner — Signature with date

1.

2.

ABSTRACT

Control theory and its applications are crucial when operating within the area of dynamic systems. Compensating for disturbances and external actions imposed on a given system being inherently unstable or semi-stable. This project consists in the realization of a Ball-Plate system with 2 degree of freedom which will control the position of the ball by tilting the plate. Two servomotors will allow the table to be oriented with a certain angle of inclination to counterbalance the movements of the ball.

A Pi camera placed above the device will transmit live images of the plate to a Raspberry Pi which will be responsible for analyzing them to draw conclusions such as the position of the all, its speed and acceleration. A Python program will then be responsible for processing these data, feed it to designed PID controller and transmitting the result by USB to the Arduino which will control the rotation of both servos independently. All the data will be sent to the Python program running in another computer using MQTT protocol. And setpoint, error and pid v/s time graphs would be plotted.

The experiment is done by droping a ball on the plate and waiting for the system response to balance the ball in the center of the plate and to follow the set trajectories. The plate is made of wood to ensure smooth slipping of the ball without friction. Balancing experiments were done, following the set trajectories, and positioning the ball at the center of the plate. Through the use of multithreading, image processing at the rate of 90 FPS was achieved. And results were plotted.

In conclusion it should be emphasized that in control systems acceptable real time performance can be achieved by decentralizing the processing unit into several PUs. It is notable to mention that with small additional piece of software and control checkpoints PU redundancy feature can be added to the plant, so in the event of physical damage and PU failure the other PU can be notified and take over the system.

ACKNOWLEDGEMENTS

We consider it as a great privilege to express my gratitude and respect to all those who guided and inspired us in the completion of this seminar. It is difficult for us to express our sense of gratitude and appreciation for the help we have received in this endeavor. Our effort here is a feeble attempt to do so.

First of all, we acknowledge for the provision of the required infrastructure by our esteemed institute **J N N College of Engineering, Shivamogga and Department of Electronics and Communication Engineering.**

We thank to **Dr. K Nagendra Prasad**, Principal, JNNCE, Shivamogga for providing excellent academic climate.

We thank to **Dr. Manjunatha P**, Dean Academics, JNNCE, Shivamogga for giving facilities to undertake this project.

We would like to thank our head of the department **Dr. S V Sathyanarayana**., who stood as a guiding spirit and lending guidance to achieve the aim with added zeal.

Our special thanks to Assistant Professor **Mr. Pradeepa S C**, our project guide for providing all the inputs and corrections needed for the preparation of the report.

Lastly, we are thankful to our classmates, teaching and non-teaching staff and everyone who has helped us directly or indirectly for the successful completion of this project.

Amoghavarsha S G	**Kiran C N**	**Niranjana Jois H C**
4JN18EC010	**4JN18EC041**	**4JN18EC057**

Contents

List of Figures

List of Tables

Code Listings

Chapter 1

Introduction

1.1 General Introduction

Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing and remote sensing. One of the advantages of machine vision is the ability of sensing moving objects from distance and without contact with the object. However, the data in the image can be easily affected by the disturbance and changes like sunlight and shadow that occur in the surrounding area of the object being sensed. Such issue can be overcome by using digital image processing and advanced image filtering techniques to get rid of such noises and disturbances. The goal of this project is to develop a Ball-Plate balancing system which uses camera above it to get feedback on the position of the ball.

The Ball-Plate system has 2 degree of freedom (DOF) the goal is to balance the ball on a horizontal plane on a given setpoint. Position on of the ball on plate is represented by 2 coordinates system X-axis and Y-axis, once the position of the ball is known, error is calculated based on setpoint and developed PID controller is used to actuate the 2 servomotors independently, one servo is used to tilt the plate on X-axis and other to tilt the plate on Y-axis both will use separate PID controllers.

The camera above the pate will capture continuous frames that will be processed using Python in order to get the coordinates of the ball. The servomotors will be controlled by Arduino which will receive the angles of the servomotors from software developed using Python which is capable of image processing, designing a controller and serial communication.

1.2 Aim of the project

The aim of this project is to study the mathematical modelling of the ball plate system in order to design a good PID controller with visual position feedback so that the ball can be balanced at the centre of the plate or follow a trajectory.

1.3 Objectives

1. To develop the image processing system and PID controller using Python on Raspberry Pi.
2. To develop the servo motors control software using ARDUINO.
3. To select the appropriate actuators parameters and mechanical structure for the control of ball-plate system.
4. To Send position of ball to the computer using MQTT.
5. To Animate and plot the ball position using ball pose obtained from MQTT.

1.4 Methodology

1.4.1 Digital Image Processing

Image processing techniques like Filtering, colour detection, Morphological Transformations like erosion, dilation, thresholding is used to get the position of the ball on the plate.

1.4.1.1 RGB to HSV Conversion

While RGB representation is good for displays, it is not ideal for colourdetection since RGB values in image vary widely depending on lighting conditions, because R, G, and B components of an object's colour in a digitalimage is all correlated with the amount of light hitting the object, and therefore with each other, image descriptions in terms of those components make object discrimination difficult.

Descriptions in terms of hue/lightness/chroma or hue/lightness/saturation are often more relevant since HSV colour space abstracts colour (hue) by separating it from saturation and pseudo-

illumination. This makes it practical for real-world applications.

The formula for converting the RGB values to HSV is shown in Figure 1.1

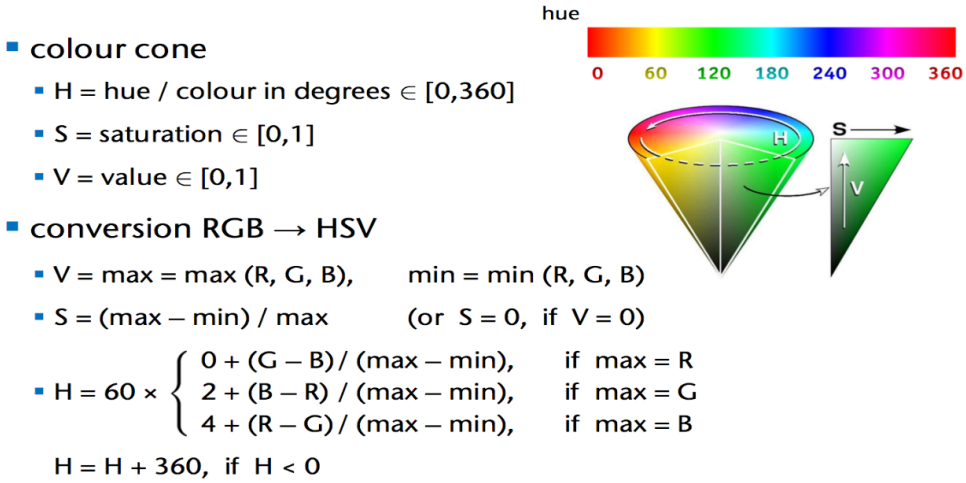
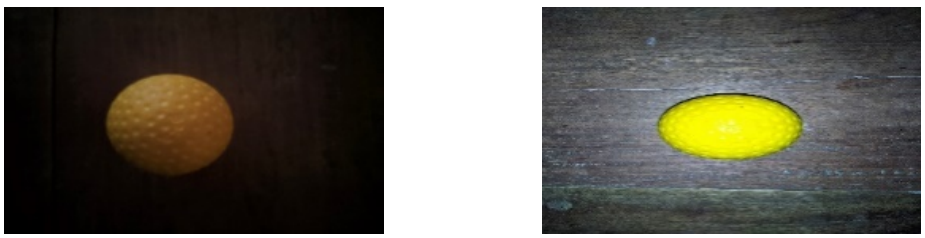


Figure 1.1: RGB and HSV color spaces

1.4.1.2 Object detection via colour-based image segmentation

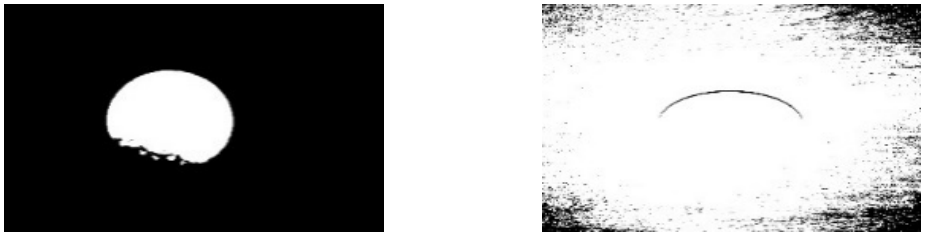
Once the image is in HSV colour space we can separate the object (in our case ball) from its surrounding by its colour using thresholding, which means if the pixel is not inside the defined range, then its value is made 0 else its value is made 1(255).

Once we have the binary image, we need to remove the noise in the image to do that we need to do some morphological operation called opening and closing. Opening will remove all the dots randomly popping here and there and closing will close the small holes that are present in the actual object. After this operation we find contours in the image and draw minimum enclosing circle around the counter which will give position and radius of the ball.



(a) Bad lighting (b) Good lighting

Figure 1.2: Sample images



(a) Bad lighting thresholding (b) Good lighting thresholding

Figure 1.3: RGB thresholding

(a) Bad lighting thresholding (b) Good lighting thresholding

Figure 1.4: HSV thresholding

1.4.1.3 Morphological Transformations:

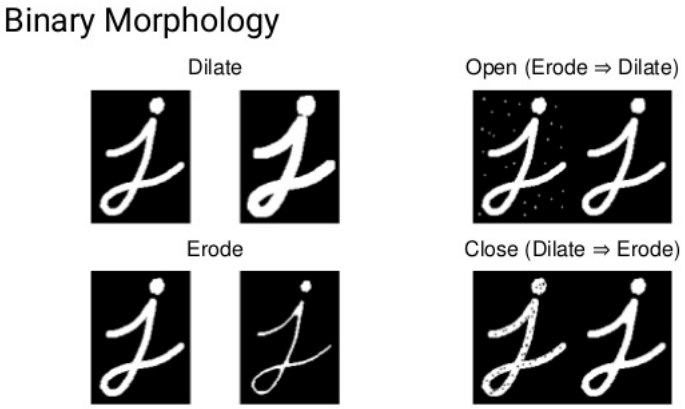


Figure 1.5: Erode and Dilate Operations

Morphological Transformations like erosion, dilation is usedto remove unwanted noise from the Binary Image. First performing erosionthen dilation is called opening an image, performing dilation then erosion is called closing an image.

1.4.1.4 Minimum circle enclosing Algorithm

The task is to find the center and the radius of the minimum enclosing circle (MEC). A minimum enclosing circle is a circle in which all the points lie either inside the circle or on its boundaries. Then coordinate other ball is sent to the PID controller.

1.4.2 PID Controller Design

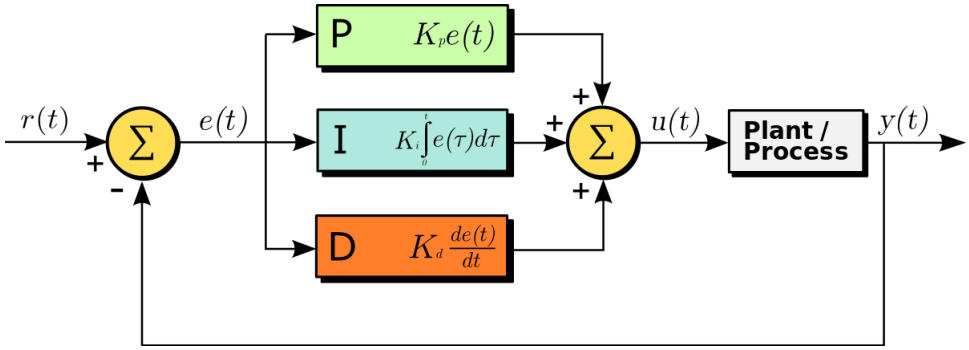


Figure 1.6: controller block diagram

A PID controller is proposed to control the system. The general PID control formula is as follows

$$u(t) = K_p e(t) + K_i \int e(t)\mathrm{d}t + K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{1.1}$$

Where,

K_p: proportional gain that is proportional to the error

K_i: integral gain that interprets for past errors.

K_d: derivative gain that interprets for future errors.

1.4.3 Mechanical structure

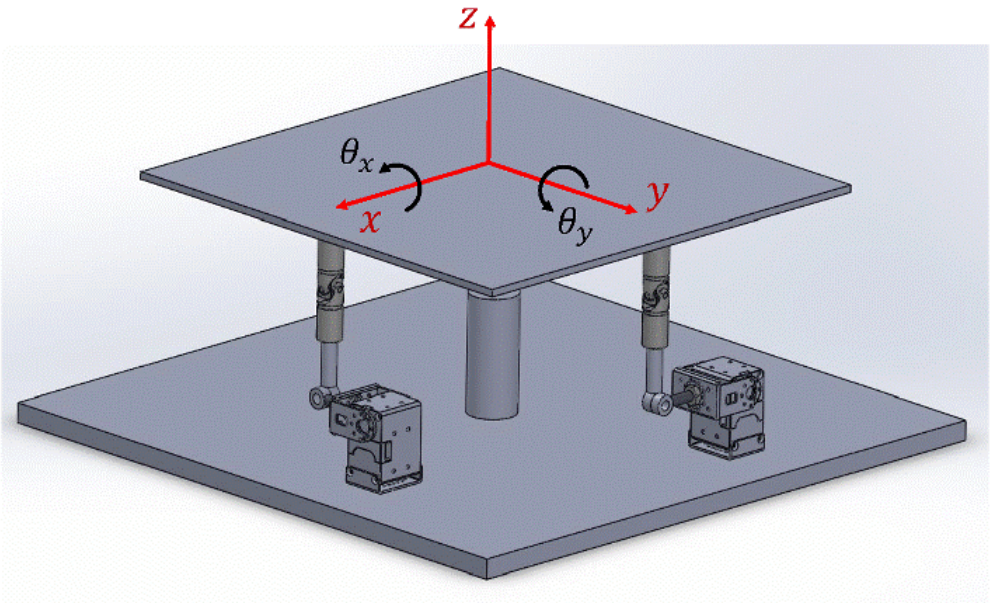


Figure 1.7: CAD Model of the project

The ball-plate system will be constructed with wood. A 30 cm × 40 cm × 1.0 cm wood fixed to a universal joint in the middle of the plate. The system will be built using wood to ensure getting weightless system and also to get frictionless movement of the ball. Each servo motor will be connected through an extended arm that will tilt the plate in the X and Y direction. The camera will be located on the top of the plate with support at a height of 60 cm and positioned to get the centre of the frame perpendicular to the centre of the plate.

1.4.4 Sending Coordinate of the ball using MQTT

Raspberry Pi will be MQTT broker and the client. The Raspberry Pi will publish the ball position and subscribe to the setpoint commands where as the laptop will subscribe to the ball position and publish the setpoint command.

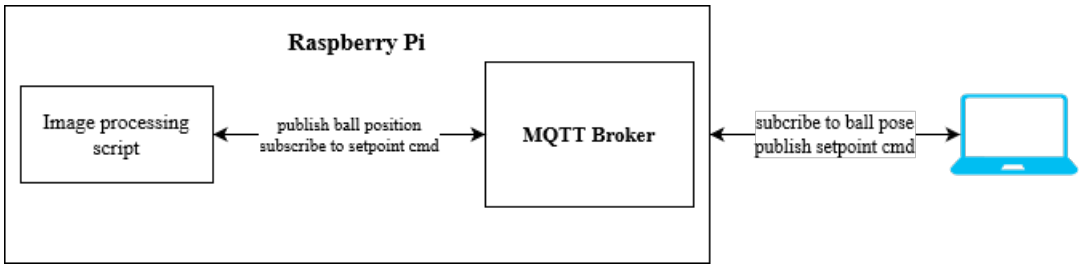


Figure 1.8: MQTT configuration used

1.5 Scope of the project

The scope of this document is to explain the design of a ball-balancing platform both in hardware and software, the given requirements, the process of operation, as well as the methods of implementation. This document is meant to be used to understand principles of functionality and hardware generalities. This document includes several charts, figures, and graphics to help better explain and visualize aforementioned details.

For research purpose the aforementioned method of control design will be implemented, monitored, and analyzed through the use of a physical construction, a ball balancing platform. The objective is to balance a ball on a flat surface and adjust for ball displacement caused by e.g., human interaction or environmental disturbances.

To actualize, generate data and acquire desired motion output of the construction, three types of hardware will be utilized apart from construction materials. The pi camera, to get the position of the ball on the plate of the ball, calculating required servo angles to move to a specific setpoint using PID controller, transceiving required servo angles to a microcontroller. The microcontroller sending control signals to a pair of servo motors. The servo motors, responding to signal from the micro-controller, adjusting in accordance with the control signal generating an angular displacement. The acquired data from the actual physical model will then be used for the analysis. Testing will be conducted through manual ball position disturbance and observed through steady state stability. Ultimately the project aims to answer the following research questions:

- Can one design and construct a ball balancing platform with satisfactory performance solely practicing linear control?
- What are the performance limitations when stabilizing a naturally unstable or semi-stable system and how do these differ from the theoretical expectations?

1.6 Limitations

- Play in the servo motors and arm joints and links.
- Vibrations in the whole structure due to shattering in the servo motors
- Geometrical construction errors and tolerance inaccuracies
- Mounting errors of the platform allowing play
- Deformation and give in solid parts.

Chapter 2

Theoretical Background

2.1 Literature survey

We referred many papers among which some will be discussed in later section we have gone through many concepts by referring these papers, many old papers and articles were also referred but latest papers had many improvements over the previous ones. New methodologies and approaches were discussed.We referred many latest papers related to this project out which five of them are discussed below.

[1]. Lefrouni, K., Moubachir, Y. and Taibi, S., 2021. Control of Ball-Plate System with Observer-Based State-Feedback and Geometric Considerations. International Journal of Difference Equations (IJDE), 16(1), pp.35-46.

- Linear model of the ball and plate system was developed by applying Lagrange's method.
- Ball-plate system was studied using a frequency domain approach.
- The developed controller is based on a state feedback controller and Luenberger observer.
- Choice of observer gains within the stability regions is difficult

Conclusions:

- The stability of the closed loop system is consequently proven
- Simulations and experimental results of demonstrations were not so excellent in tracking performance and control design
- Back stepping is classic method to solve the control design problem cascaded under-actuated system by transforming the control design problem of entire system into several lower order system.
- Model was done using load gear and driver gear which was introducing more friction component than usual.

[2]. Ali,E.,and Aphiratskan,N.(2015, December).AU ball on plate balancing robot. IEEE. International Conference on Robotics and Biomimetics(pp. 2031-2034).

- PC-Based Controller, Linux
- More than one PUs.
- 2 axis PID control
- Matlab/Simulink-based
- Raspberry Pi 2
- Slower FPS due to single threading

Conclusions:

- In system with high number of active actions the derivation of mathematical model becomes somewhat impossible, therefore adopting a model free approach using feedback to calculate the responses is the only rescue.
- For control mechanism raspberry pi 2 is used which equips 700Mhz ArM 11762P-S processor solely for image processing task and a STM 3219407 board the includes 168MHz ARM Processor solely for control of the servos.

[3]. A. Gustavo and L. Juan and J. B. Jovani, “Control of a ball-and-plate system using a State-feedback controller”. Ingeniare. 28. 6-15. 10.4067/S0718-33052020000100006, 2020

- Mechanical constraints with which the central ball joint (central piece) was designed only allows inclinations of ± 15º in the x-y plane.
- State feedback controller using the pole assignment method was designed to maintain a desired position of the sphere over the plane
- Phenomenological analysis to obtain a state-space representation of the system

Conclusions:

- Actuation mechanism is based on RC servos motion which are connected to the beneath of plate through arms and sphere joints.
- The ball position detection is done by capturing frames with a single Raspberry Pi camera module that is placed on top of the plate.

- Two processing units Raspberry Pi and STM32F407.
- Main disadvantage is that using such a system with high quality images leads to decrease in FPS.
- This gives a period of 66ms between each ball coordinate being send to second PU.
- This was not enough to let the second PU to calculate the PID values while giving the system acceptable precision in real time.
- PID tuning was manually by Ziegler-Nicholas method.
- Due to this the ball dropped on plate came to rest at about 28 secounds.

[4]. Mohsin, M., T. (2015). Development of a Ball and Plate System.122nd ASEE Anual Conference and Exposition.Washignton.

- Three-degree-of-freedom system.
- Simulations using Simulink/MATLAB™.
- Touch panel platform.
- The control system utilizes Euler-Lagrange equations.
- An irregularity of the system performance may result from the sensitivity of the touch screen and control algorithm.

Conclusions:

- In this paper mainly four stages of engineering analysed and designs were addressed.
- 1st state corresponds to the phenomenological analyses of the movement of ball on plate from which its representation in space state is obtained.
- In 2nd stage with the previous results a state feedback controller using the pole assignment method is developed and the establishment time and
- Percentage overshoot as the main design parameter are considered.
- In 3rd stage simulations are performed in Simulink in order to verify the fulfilment of desired parameters.
- In the final stage, ball plate system controlled and implementation is experimentally validated

[5]. M. Yaseen, J. A. Haider, "Modeling and control for a magnetic levitation system based on SIMLAB platform in real time", Results in Physics, Volume 8, Pages 153-159, ISSN 2211-3797, 2018.

- Maglev system model.
- 4 electromagnets as actuators for applying magnetic forces to achieve levitation and position control.
- LQR based controller, PID control and Lead Compensation.
- Real-time control Simulink feature of (SIMLAB) microcontroller

Conclusions:

- As the work use that touch panel for the main point locator a sensor mount was fabricated for the touch panel to fit within the platform to eliminate any sliding, so the risk of damaging of the costly sensor was a bit less.
- This 4-wire resistive feather touch panel obtains the ball location very easily.
- It consisted of two layers, as the top layer was flexible it causes the surface to deform when a force is applied the operation force range is around 0.02 to 0.3 Newton, which in turn produces current as an indication.
- The project also used linear actuators to change the angle of the plate with respect to ground.
- The main delivers due to the linear actuators only.
- The ball location was to updated within milliseconds but the movements of the linear actuators wasn't up to the required level.

Chapter 3

Design and Implementation

3.1 Introduction

A ball-and-plate platform is a common example of the dynamical system used to apply modeling and control concepts studied in some engineering courses. In this kind of systems, a spherical body is positioned on a mobile plate with two degrees of freedom. The system is operated by two servo motors that cause displacement of the sphere in the x and y axis in the plane of the platform. It is known that such a system is highly unstable and non-linear so, different studies have been interested in this system to develop new control methods and algorithms.

3.2 System Design

The Ball-Plate system has 2 degree of freedom (DOF) the goal is to balance the ball on a horizontal plane on a given setpoint. Position on of the ball on plate is represented by 2 coordinates system X-axis and Y-axis, once the position of the ball is known; error is calculated based on setpoint and developed PID controller is used to actuate the 2 servomotors independently, one servo is used to tilt the plate on X-axis and other to tilt the plate on Y-axis both will use separate PID controllers.

The camera above the plate will capture continuous frames that will be processed using Python on the Raspberry Pi in order to get the coordinates of the ball, these coordinates are sent to the PID controller which is also running on the Raspberry Pi and the computed servo angles are sent to the Arduino. The Arduino then receives the angles of the from the Raspberry Pi and change the servomotors angle accordingly.

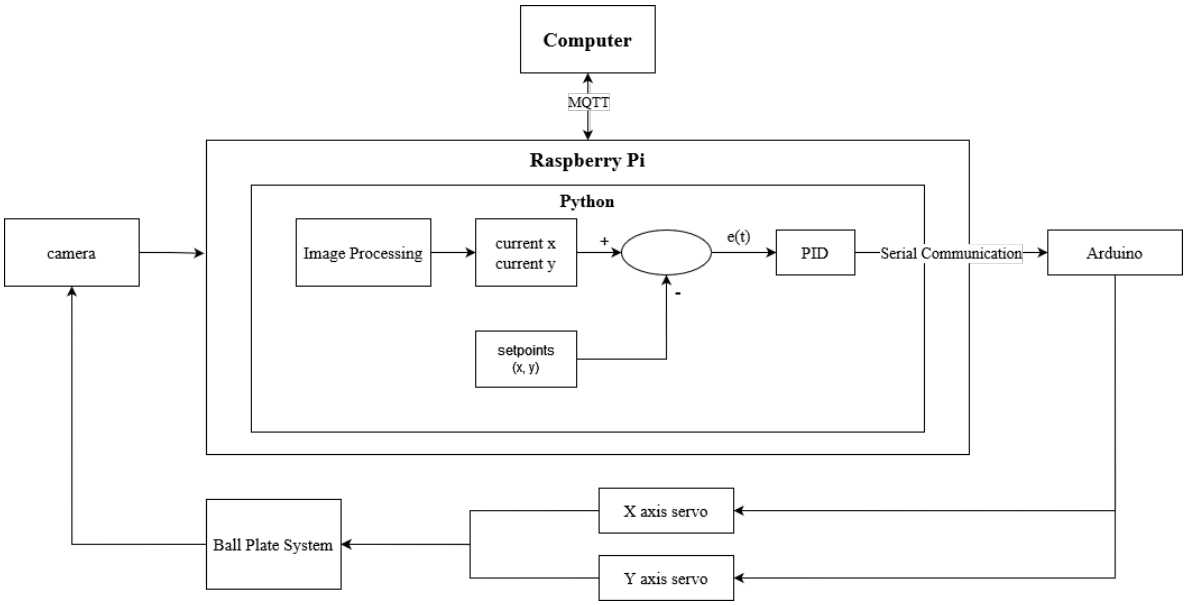


Figure 3.1: Block Diagram

3.3 Circuit Description

As follows below in Figure 3.2 is the electrical circuit and the respective components.Besides wiring the components are the Arduino Uno microprocessor, two servo motors, a small bread-board and a voltage step down component.

The Arduino and the servo motors are powered from an external power source stepped down to 5V. The capacitors are used to prevent temporary power deficit and assure continuity in the power supply to the servo motors.

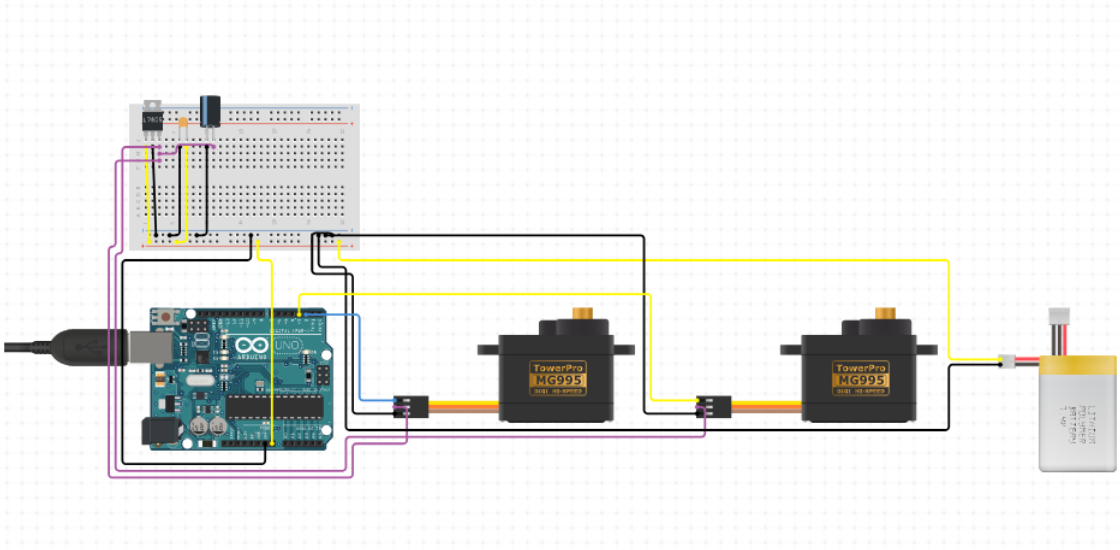


Figure 3.2: Servo Circuit Diagram

3.4 Implementation

3.4.1 To develop the image processing system and PID controller

3.4.1.1 Installation of Raspberry Pi OS and libraries on Raspberry Pi.

We flashed Raspbian image onto the 8GB SD card. Since we did not have any external monitor, we used Raspberry pi in headless mode. To use it in headless mode we mounted the SD card and created `WPA_SUPPLICANT` file with Wi-Fi router info and an SSH file in order to SSH into Raspberry pi. We inserted the SD card into the Raspberry Pi and SSHed into it using the default password which we changed it soon after.

In order to install opencv we tried it the traditional way i.e., `pip3 install opencv-python`. But since we needed aruco marker detection we used `opencv-contrib-python`: `pip3 install-contrib-python`. But it failed with error "could not find a version that satisfies the requirement opencv-contrib-python". Then we did some research and we tried different versions many of which failed with error "unable to build wheel". Then we found the version that worked `3.4.4.19`: `pip3 install opencv-contrib-python==3.4.4.19`

3.4.1.2 Developing multithreaded image processing system

First stage is to get frames from Pi camera. We use picamera library to get frames from pi camera in python. Since we needed to get frames at 90 FPS resolution should be set to 640×480 and frame rate should be 90 and video port should be set to `True`.

```
camera.resolution =(640,480)
camera.framerate=90
camera.capture_sequence(find_table(aruco_marker_order), use_video_port=True
                                        )
output = ProcessOutput(num_image_processors)
camera.start_recording(output, format='mjpeg')
while not done:
    camera.wait_recording(1)
    camera.stop_recording()
```

Code Listing 3.1: Getting frames from the camera

In the above code `find_table()` is a generator `yield`'s a new `io.BytesIO()` byte stream and `output` is a object of class `ProcessOutput` which has a method called "write"

which receives the image buffer from the camera and writes it to the different image processing threads which are ready and sends them to an "event" indicating that new frame is ready to be processed. When the next frame is ready from the picamera the write method does not need to wait for image processing thread to complete, it writes to `io.BytesIO` stream from pool of image processes which are ready.

```
def write(self, buf):
    if buf.startswith(b'\xff\xd8'):
        with self.lock:
            if self.pool:
                self.processor = self.pool.pop()
            else:
                self.times_starved += 1
                self.processor = None
            if self.processor:
                self.processor.stream.write(buf)
                self.processor.id = perf_counter()
                self.processor.event.set()
```

Code Listing 3.2: Write method of output object

Where `self.pool` list contains 4 image processing objects. `self.processor.stream` is an `ByteIO()`. Another problem arises when we use for image processing threads in parallel that is one thread may finish processing the image faster than the other.

ex: suppose say at $t = 0ms$ a frame arrives from Pi camera, the `write` method of `output` object writes the buffer to the 4th image processing thread's `stream` attribute and the 4th thread begins to process the image after getting the processor event. If another frame arrives at $t = 10ms$ and `write` method of `output` object writes the buffer to the 3rd image processing thread's `stream` attribute and it finishes its processing at $t = 15ms$. But if 4th thread finishes processing at $t = 20ms$. Since we send the commands to PID controller soon after the processing is done, we end up sending past data after the present data this will lead to instability of the system.

In order to prevent this issue, we add another attribute to processor object that is `id`. `id` attribute holds time at which frame was provided so that if the above situation occurs, the PID controller can discard the data which arrives after the data that is latest. PID controller can make use of below method in order to decide whether or not to consider the data.

```
def check_id(self, current_id):
    if self.last_id > current_id:
        return False
    self.last_id = current_id
    return True
```

Code Listing 3.3: check_id method to validate the frame

3.4.1.3 Finding the corners of the table

Since we use the video port the frames we get will be encoded and we are not in numpy array format so we need to convert it to numpy array format that is of shape (x, y, 3) y is number of rows, x is number of columns and 3 represents colour channels (R, G, B) in opencv (B, G, R).

In order to do that we make use of two important functions namely `np.frombuffer()` and `cv2.imdecode`. Before reading from streams(stream,read())we have to seek the starting byte `stream.seek(0)`. And after converting it to numpy array format we clear the stream in order to get ready for next frame from camera `stream.truncate()`. Since the lighting condition changes, we loop over all the threshold values and keep track of the ID we identified and we exit once we get all the specified ID so that we can crop the table out from the frame.

Figure 3.3: Table sketch

3.4.1.4 Finding the position of the ball on the table

First as discussed earlier we don't get the image in numpy array format so we have to decode it using np.from buffer and `cv2.imdecode` functions. After getting the frame in BGR format we call `self.find.ball` method.

```
def find_ball(self):
    hsvFrame = cv2.cvtColor(self.frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsvFrame, self.lower_limits, self.upper_limits)
    mask_open = cv2.morphologyEx(mask, cv2.MORPH_OPEN, self.kernel_open)
    mask_close = cv2.morphologyEx(mask_open, cv2.MORPH_CLOSE, self.
                                        kernel_close)
    contours, _ = cv2.findContours(mask_close, cv2.RETR_TREE, cv2.
                                        CHAIN_APPROX_SIMPLE)[-2:]
    if contours:
        ball = max(contours, key = cv2.contourArea)
        if cv2.contourArea(ball) <= 0:
            self.ball_pose = ()
            return False
        ((x,y), radius) = cv2.minEnclosingCircle(ball)
        if radius < 5 or radius > 40:
            self.ball_pose = ()
            return False
        self.ball_pose = ((int(x), int(y)), int(radius))
        return True
    self.ball_pose = ()
    return False
```

Code Listing 3.4: find_ball method to detect the ball

We can find the ball by performing color segmentation.Which can be done in RGB or BGR or HSV colour space. The problem we found in RGB colour space is that whenever there is a small change in lighting conditions in the Room all the values in RGB would change i.e., R & G & B differently, which was making it harder and inaccurate ball detection.

So first we convert the BGR frame to HSV using the function `cv2.cvtcolor`. Then we threshold the image using lower limit and upper limit in HSV space which we found using experimentation. `cv2.inRange`. Then in order to reduce the noise we do morphological transformations like open and close. Then find all the contours in the frame and pick the contour with the maximum area. Then we set the ball-pose to be the position and radius

obtained which will be used in PID controller.

3.4.1.5 Building the PID Controller

We need some kind of controller in order to make use of the ball positions data we get from the image processing threads and bring the ball to the requested setpoint.
We choose PID controller since it is widely sed in industrial systems and it is easy to implement and also provides very good results as we found out in our previous sem mini project.

PID stands for Proportional-Integral-Derivative which is represented by Equation 1.1. As we can observe in the equation Kp is an proportional constant meaning $K_p e(t)$ increases proportionally as $e(t)$ (error) increases. This will help the ball to get closer to the setpoint however when $e(t) = 0$, $K_p e(t) = 0$, so the plate is flat but $e(t-1) \neq 0$. So the ball will have momentum so the ball will move past the setpoint and $K_p e(t)$ will account for that again the same process will continue thus inducing oscillations.

Kd is the derivative constant and $K_d \frac{\mathrm{d}}{\mathrm{d}t}$ is a derivative form as the ball approaches the setpoint $K_p e(t)$ decreases but derivative term $K_d \frac{\mathrm{de}}{\mathrm{d}t}$ does not since $\frac{\mathrm{de}}{\mathrm{d}t}$ is the velocity of the ball hence derivative term increases which tilts the table opposite to the motion of the ball trajectory hence reducing the oscillations.

Ki is an integral constant and $Ki \int e(t)\mathrm{d}t$ is an integral term. This helps us to reduce the static error i.e., when the ball is very close to the setpoint the proportional term maybe very less and since the ball will be stationary. Derivative term will also be 0. But the integral term adds the error in each iteration thereby reducing the static error.

3.4.1.6 Deciding the layout so as to build the PID controller

Things to be keep in mind while designing the table servo mounting

1. Increasing the servo angle should move the table up.
2. Decreasing the servo angle should move the table down.

With these two things in mind, we define `error = setpoint - input`. We can see that when `setpoint > input`, `error > 0`. So, when the ball is at top and setpoint is at down the table tilts up and the same thing with X coordinate. The PID controller `__call__`

method can be seen in Code Listing A.5.

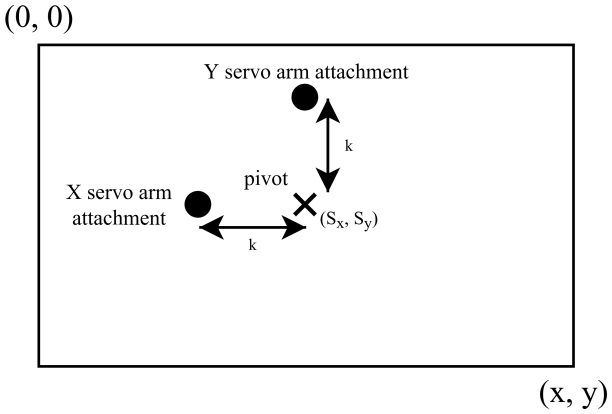


Figure 3.4: Table dimensions

3.4.2 To develop a servo motor control software using Arduino

Arduino used: Arduino UNO

In order to receive the PID values from the raspberry pi, serial communication between ARDUINO and py must be established. We can achieve this by using inbuilt serial.bigin method and setting the same baud rate between two devices. Baud rate used 115200

The servo motor is controlled through electrical pulses with pulse width as a variable. The PWM does further more regulate the orientation of the angular change, that is clockwise or counter clockwise. We can change the pwm on time thus the angle of the server arm by using servo library the server library provides the function `writeMicroseconds` which takes microseconds as an argument. We get the x and y axis microseconds (thus angle) from the raspberry pi in the format: `"{x_servo_us}{sperator}{y_servo_us}{stopByte}"` where separator is # and stop byte is $.

The ARDUINO then separates the x and y axis microseconds and converts it into integer by using the function `|atoi|` and then applies it using the function `writeMicroseconds` The servo control code can be seen in Code Listing A.1.

3.4.3 To select appropriate actuators parameters and mechanical structure for the control of ball-plate system

While implementing the PID controller layout of the table was decided Figure 3.4 and now the distance from the centre of the table to the server armed attachments was decided to be 8 cm i.e., `K=8cm`. And the height at which the pi camera is mounded is 75 cm from the base of the platform.

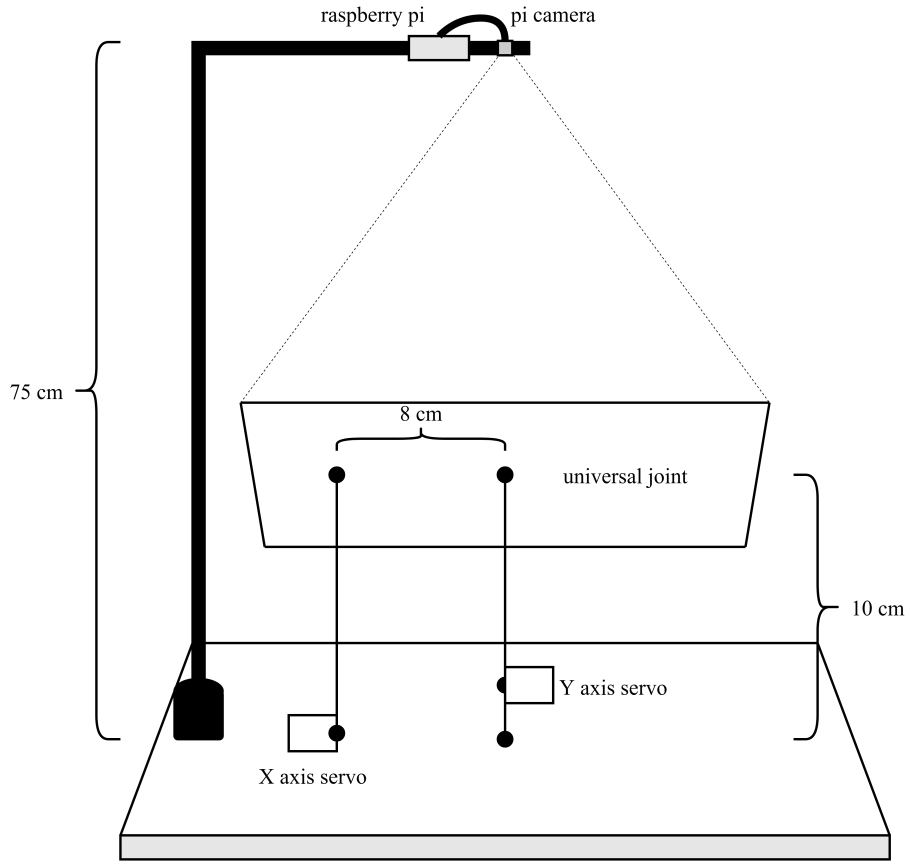


Figure 3.5: Designed Mechanical Structure

3.4.4 To send position of ball to the computer using MQTT

MQTT broker used: Mosquito

By default MQTT broker listens on port 1883. In order to communicate with MQTT broker from Python we installed a library called paho mqtt `pip install paho_mqtt`.

In order to send messages we need to first establish connection to MQTT broker. We can connect to MQTT broker by using a method called connect.

```
import paho.mqtt.client as mqtt
client = mqtt.Client()
client.connect(broker_url, broker_port)
```

Code Listing 3.5: Connecting to MQTT broker

Since we are connecting to broker from raspberry pi where MQTT broker is also installed `brocker_url = "localhost"` and `brocker_port = 1883` Once we are connected to broker we can subscribe to a topic by using the method `client.subscribe` We can publish by using the method `client.publish` Since we use these methods multiple times we made ourselves a library called iot.py. And we created a class called MQTT in it, which will be imported by the image processing script and by interface script in Objective 5 which can be seen in Code Listings A.3.

3.4.5 To animate and plot the ball position using ball pose obtained from MQTT

We used the iot library A.3 made in Objective 4, in order to receive the ball position, set point and PID values from the raspberry pi. In order to receive the messages we first have to connect to the mosquito MQTT broker on raspberry pi in order to do that both raspberry pi and the computers has to be on the same network.

3.4.5.1 Multicast DNS (MDNS)

In order to connect to the MQTT broker on raspberry pi we need its IP address but since DHCP server assigns different IP every time the device is reconnected every time thereby we would have to change the code multiple times. We could use static IP but if we use a different network that uses different subnet mask then again we need to change the configuration. That is where MDNS is helpful

MDNS is used to locate a device or service by name on a small local Network without using pre configured name server. So we can connect to the raspberry pi using `"Hostname".local` In our case `raspberrypi.local`. When the computer require to know IP address of above domain it sends DNS query to a DNS server using a unicast udp message on port 53. All MDNS host see this query in our case raspberry pi and the host that has its network name responds to the query using a multicast message that contains its IP address.

3.4.5.2 Receiving the setpoint and ball position

In order to receive the ball position we need to subscribe to the topic `ball_set_pose` we can do that by importing our library A.3 and using `subscribe_thread_start` method.

```
from iot import MQTT
broker_url = "raspberrypi.local"
pose_topic = "ball_set_pose"
mqtt = MQTT(broker_url=broker_url)
mqtt.subscribe_thread_start(pose_callback, pose_topic, 0)
```

Code Listing 3.6: Subscribing to a topic

The payload we receive will be a JSON string of format:

`"{"setpoint":[x,y],"ball_pose":[[x,y], r],"pid_values":[x,y]}"`

We can convert it back to Python directory by using a method `json.load` from JSON library. In case of animating the ball we need only set point and ball pose so following call back function is used.

```
def pose_callback(self, client, userdata, msg):
    pose = json.loads(msg.payload.decode('UTF-8'))
    self.ball_pose = pose['ball_pose']
    self.setpoint = pose['setpoint']
```

Code Listing 3.7: Position callback function for animating the ball

In case of plotting the graph we need all three values so following call back function is used

```
def pose_callback(client, userdata, msg):
    pose = json.loads(msg.payload.decode('UTF-8'))
    ball_pose = pose['ball_pose']
    pid_values = pose['pid_values']
    if ball_pose:
        ball_pose_plot[:] = ball_pose[0]
    if pid_values:
        pid_plot[:] = pid_values
    setpoint_plot[:] = pose['setpoint']
```

Code Listing 3.8: Position callback function for plotting the graphs

where `ball_pose_plot`, `PID_plot` and `setpoint_plot` are array objects from multiprocessing library which offers multithreading protection.

```
from multiprocessing import Array, Value

setpoint_plot = Array('i', [0, 0])
ball_pose_plot = Array('i', [0, 0])
pid_plot = Array('d', [0, 0])
```

Code Listing 3.9: Importing and initializing Array objects

3.4.5.3 Animating the ball position

In order to animate the ball position first we take a static table or platform image and add the ball image of size $2 * r \times 2 * r$ at the position received from MQTT where 'r' is the radius of the ball received from MQTT. Since the radius of the ball changes and resizing the image is computation intensive, this would result in decrease in FPS which would make animation look choppy. Thus we resize the ball image from 2×2 till 140×140 and made a dictionary with keys as radius of ball and value as images.

```
{
    1: img_2x2,
    2: img_4x4,
        .
        .
        .
    70: img_140x140
}
```

Code Listing 3.10: Ball images dictionary with radius as key

And we stored it using pickle library. So now we can just use pickle to load the dictionary into memory and there is no need to compute and resize the image. We can just get the required image by passing the radius of ball as key. The animate function which can be seen in Code Listing A.4 will be continuously called in a loop with ball position and radius given as argument.

3.4.5.4 Sending the setpoint command

We can send the set point command by clicking anywhere on the animated window or pressing on a specific key. Again we can send the set point by using our iot library on topic `key_cmd` in format `"S{x},{y}"`

3.4.5.5 Sending the PID tuning parameters

We can send the PID tuning parameters namely `kp_x, kp_y, kd_x, kd_y, ki_x` and `ki_y` by moving the sliders provided in the animation window. We can send the tuning parameters using our iot library on topic `key_cmd` in format `"T{parameter_name}{value}"` where the raspberry pi will be subscribed to the topic `key_cmd` and determines if it is a set point command or tuning parameter using the first letter.

3.4.5.6 Plotting the graphs

Graph plotter will run as a separate process so it again uses iot library A.3 in order to get ball position, setpoint and PID value data from MQTT. In order to plot any live data we need to have a buffer which stores the data and when new data is received we need to delete oldest data and insert the new data at the start so that it gives us that rolling effect.

We use double ended queue data structure for this purpose. Where we insert the new data at the end and remove old data from index zero.

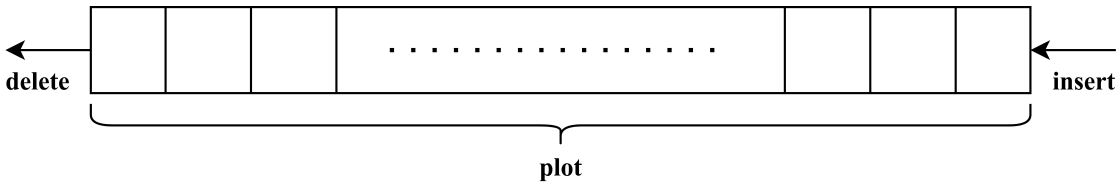


Figure 3.6: Dequeue Operation

First we used matplotlib in order to plot the graph but it was too slow so we switched to by py qtgraph instead which uses Qt. Then we used timer from Qtimer in order to call graph plotter function which plots the data in dqueue every 1/800 second. The dqueue will be updated in parallel in a separate thread when ever the data comes from the MQTT.

In order to make it thread stay safe we used array from multiprocessing library for dqueue to update. 3 separate graphs will be plotted in the same window i.e., set point vs time and ball position vs time, error vs time and PID vs time all three will be in separate sections. The `qt_plotter` function can be seen in Code Listing A.2.

3.5 Summary

To summarize, the camera above the table will capture continuous frames that will be processed using Python in order to get the coordinates of the ball. And PID controller was designed using Python which will calculate the PID values and is sent to the Arduino using serialcommunication.

The Arduino then recceives these values and changes the servo angle accordingly. The platform is designed and the pi camera is at the height of 75 cm from the base of the platform which captures the whole view of the platform then required ROI is cropped using Aruco markers present at the edges of the platform.

Chapter 4

RESULTS AND DISCUSSIONS

4.1 Need of Raspberry PI 4B with 4GB of RAM

We intend to do real time image processing at 90 FPS which a Pi camera can do at 640 x 480 resolution with minimal delay. But in order to get 90 FPS, IF we were to use single threading, the processing of single frame would have to be $\frac{1}{90}$ seconds, which for any computer is hard. So, we use multithreading that is one thread is used just for fetching the frames from the camera and other four threads to do image processing. For this we need more than 1 crore which only 4B processor has while 3B+ variants have single core processor.

4.2 Developed the image processing system and PID controller using Python on Raspberry Pi

- Installed Raspbian OS on Raspberry pi.
- Interfacing the PiCamera.
- Wrote efficient multithreaded image processing code in python which
    1. Finds the corners of the table.
    2. Finds the position of the ball on the table.
    3. Process the frames at 90FPS
- Wrote Python Code for PID Control which will be used by next objective.
- Decided the layout so as to build the PID controller.

Figure 4.1: Raspberry Pi and PiCamera

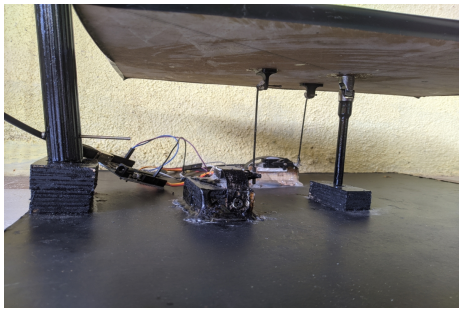
4.3 Developed the servo motors control software using Arduino

- UART protocol was used to interface Arduino UNO with Raspberry Pi
- Baud rate of 115200 was used
- PWM on time in μs calculated by the PID Controllers is received through UART
- Using `writeMicroseconds` method in Servo library, PWM Signal is generated.
- This PWM signal is given to the Servos in order to change the angle of the platform.

4.4 Developed mechanical structure for the control of ball-plate system

- 30cmx40cmx1cm wooden slab is used as Platform on which the ball will be balanced.
- 4 Aruco Markers are placed at each corners so that the perimeter of the platform could be calculated as shown in Figure 4.3.

(a) Platform



(b) Servo Placement

Figure 4.2: Base Design

Figure 4.3: Overall Mechanical Structure

- Servo Specifications:
  1. **Model:** MG90S
  2. **Weight:** 13 gm
  3. **Operating voltage:** 4.8V-6.6V
  4. **Stall torque @5V:** 2kg-cm
- 3D printed control horns are used to connect the servo rods to the platform.
- The distance from the control horn to the centre of the platform is 8cm
- Universal joint is attached at the centre of the platform to allow 2 degree of freedom as shown in Figure 4.2b.

- Steel pipe bent at 90 degree is used to hold Raspberry Pi and Pi Camera.
- Pi camera is mounted at the height of 67cm from the platform at the centre as shown in Figure 4.2a.

4.5 Send position of ball using MQTT and animate and plot the ball position

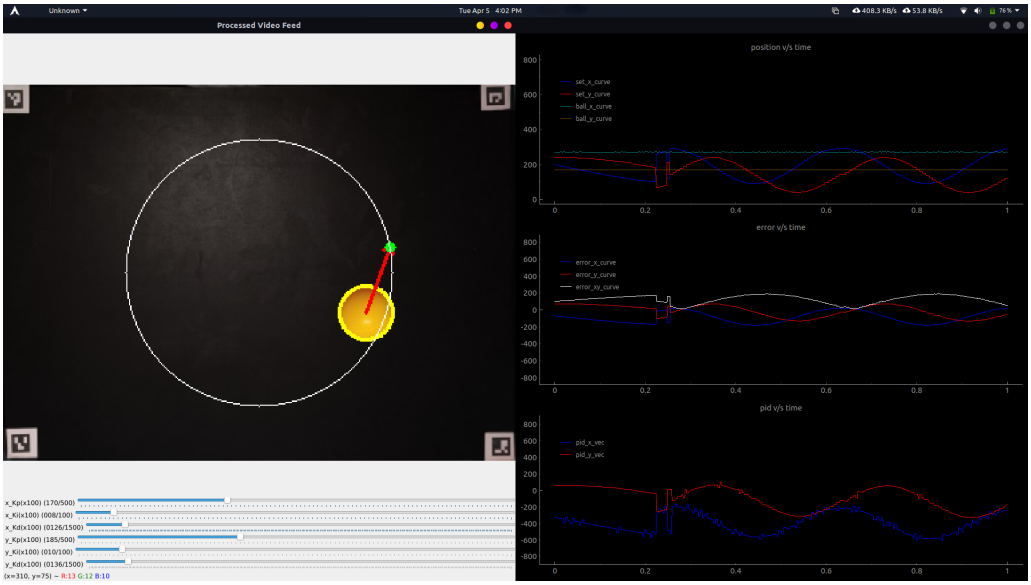


Figure 4.4: Interface

- Mosquito MQTT broker is installed on Raspberry Pi through which position of the ball, setpoint and the PID values is sent to the computer
- Using these values ball position is animated using existing background image
- And setpoint, error and PID vs time graphs are plotted.

4.6 Issues Faced

4.6.1 Raspberry PI Overheating

- At this point we noticed that the pi was overheating and this issue resulted in loss of performance.

- The CPU temperature was reaching over 80°Celsius just for updating the system. This was basically verified by CPU temperature command:

  `/opt/vc/bin/vcgencmd measure_temp`

- We researched a little and found that we needed some passive or active cooling.

- So, we cut an existing large heat sink to the size of Pi processor and put some thermal paste and secured it to the processor using some threads.

- We also provided some active cooling by using an 5V USB computer fan which draws its power from Raspberry Pi itself and the resulting temperature was under 40° Celsius at high loads.

4.6.2 False Positives while detecting the ball

- The issue we had here is that when ball is not present small noise which resemble the colour of the ball and would be detected, it was random so we check the radius to be in certain threshold (5-40).

4.6.3 Integral windup

- If just define integral = $K_p \times error \times \mathrm{d}t$ the integral term would just keep on increasing and goes beyond the range the servos can go so we clamp the integral term preventing integral windup.

4.6.4 Projected error instead of actual error

Another issue is that whenever the table tilts, we will not be getting the true error since we will be looking at the projection of the table. Hence in order to correct this we wrote another function called error-map.

we see that,

$$d = r\sin(\theta_s)$$

$$d = K\sin(\theta_t)$$

$$r\sin(\theta_s) = K\sin(\theta_t)$$

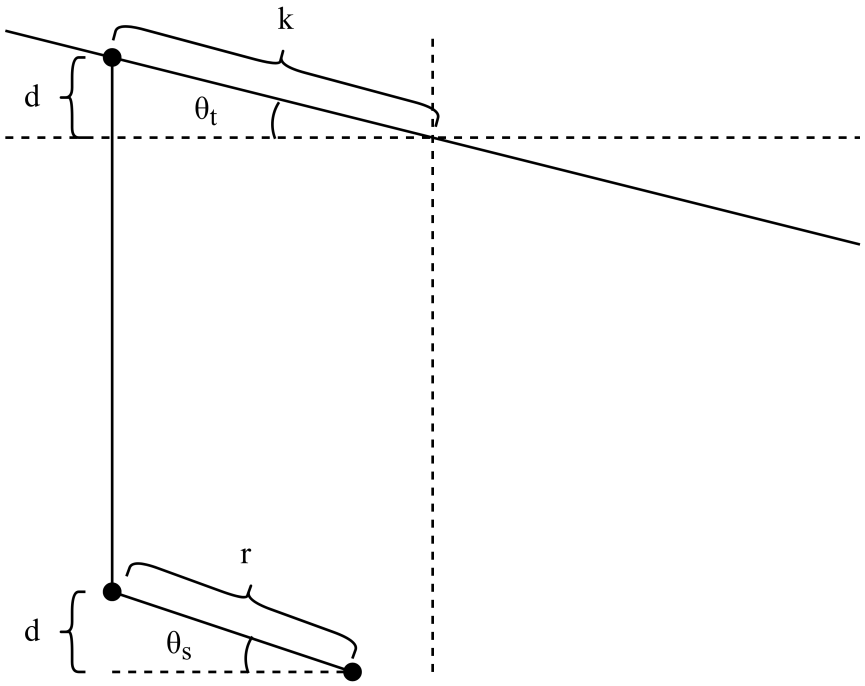


Figure 4.5: Servo table angle relation

$$\theta_t = \sin^{-1}(\frac{rsin(\theta_s)}{k}) \tag{4.1}$$

Once we have the table angle we can get the real error from projected error by

$$r\_error = \frac{error}{\cos(\theta_t)} \tag{4.2}$$

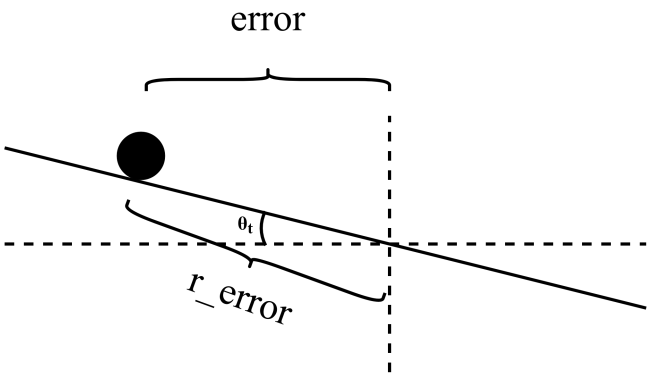


Figure 4.6: Actual error from projected error

As we can see when $\theta_t = 0, \quad r\_error = error$

Now we can take the calculated PID value and send it to the Arduino.

4.7 PID and Serial communication parameters used

Parameter	Value
kp_x	1.70
ki_x	0.08
kd_x	1.26
x_offset	1535
exp_filter_alpha_x	0.85

Table 4.1: X axis PID tuning parameters

Parameter	Value
kp_y	1.85
ki_y	0.1
kd_y	1.36
y_offset	1410
exp_filter_alpha_y	0.9

Table 4.2: Y axis PID tuning parameters

Parameter	Value
port	/dev/ttyACM0
baudrate	115200
write_timeout	30 ms
seperator	'#'
stopByte	'$'

Table 4.3: Serial communication parameters

Chapter 5

Conclusions & Future Scope

5.1 Conclusions

The overall system will consist of two distinct processing units, one controls the servo motors while the other runs the PID control and the image processing task to track the ball position. Both PUs will be coupled via a serial UART communication channel. The control mechanism is a free-model approach based on PID control. The PID tunning will be done manually by applying Ziegler-Nichols method. The experiment will be conducted by dropping a ball on the plate and waiting for the ball equilibrium around the center of the plate. A disturbance will be introduced after having the ball stabilized and the motion of servo arms and ball coordinates will be recorded and sent using MQTT and plotted on another computing system. The image processing rate 90 FPS is achieved.

In conclusion it should be emphasized that in control systems acceptable real time performance can be achieved by decentralizing the processing unit into several PUs. It is notable to mention that with small additional piece of software and control checkpoints PU redundancy feature can be added to the plant, so in the event of physical damage and PU failure the other PU can be notified and take over the system.

5.2 Future Scope

There is a bit of play in the servo motors and arm joints and links in our construction, we could address this by using better precession 3D printed parts. And there is also vibrations in the whole structure due to shattering in the servo motors and we wolud like to address this by better shock absorbing material at the bottom of the mechanical structure than the double sided tape we are using now. And we would also like to address mounting errors of the platform allowing play and Deformation and give in solid parts. And investing in better servos will also increase precession and will reduce the vibrations in the structure.

Chapter 6

References

[1] Ali, E., & Aphiratsakun, N. (2015, December). AU ball on plate balancing robot. IEEE. International Conference on Robotics and Biomimetics (pp.2031-2034). https://ieeexplore.ieee.org/document/7419072

[2] Liu, A., Peng, C., and Chang, S. (1997). Wavelet analysis of satellite images for coastal watch IEEE Journal of Oceanic Engineering (Vol.22, pp. 9–17).

[3] A. Gustavo & L. Juan & J. B. Jovani, "Control of a ball-and-plate system using a State-feedback controller". Ingeniare.28.6-15. 10.4067/S0718-33052020000100006, 2020

[4] M. Yaseen, J. A. Haider, "Modeling and control for a magnetic levitation system based on SIMLAB platform in real time", Results in Physics, Volume 8, Pages 153-159, ISSN 2211-3797, 2018.

[5] Kar Anirban. OpenCV object tracking by colour detection in python.

[6] Adrian Rosebrock. Ball tracking with OpenCV. https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv

[7] Araki, M. "PID Control". http://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf

[8] Serial Communication between Python and Arduino.

[9] OpenCV tutorial on moments. https://docs.opencv.org/3.4/d0/d49/tutorial_moments.html

[10] OpenCV documentation on Morphological Transformations.

[11] Mohsin, M., T. (2015). Development of a Ball and Plate System.122nd ASEE Anual Conference & Exposition.Washignton.

Appendix A

Appendix

International Journal of Difference Equations (IJDE).
ISSN 0973-6069 Volume 16, Number 1 (2021), pp. 35-46

https://dx.doi.org/10.37622/IJDE/16.1.2021.35-46

Control of Ball-Plate System with Observer-Based State-Feedback and Geometric Considerations

Khalid Lefrouni
Mohammed V University in Rabat, Morocco.

Younes Moubachir
Mohammed V University in Rabat, Morocco.

Saoudi Taibi
Mohammed V University in Rabat, Morocco.

Abstract

The ball and plate system is a very unstable and non-linear system. So different studies have been interested in this system to develop new control methods and algorithms. In this article, we propose a state feedback controller based on the Luenberger observer that provides an estimate of the system state. To this end, first, considering the feedback delay in the control loop, we determine the state space model of the Ball-Plate System, and secondly, based on the geometric method and the state feedback control developed in the previous work [7], we synthesized an observer. Finally, we tested the effectiveness of our methodology through simulation.

Keywords: Luenberger Observer, Frequency-Domain, Ball-Plate System, State Estimation.

INTRODUCTION

Like the magnetic levitation system [5] and the ball-Beam system [4], the ball and plate system has complex dynamics due to its instability and nonlinearity, so it has attracted special interest in teaching engineering and research. The main advantages of this type of system are primarily the facility of its implementation and the opportunity for experimental verification of theoretical knowledge in modeling, control, and other engineering disciplines [2].

Several works have been reported in literature. In [6], the ball and plate system uses two magnetic actuators for two degrees of freedom. For this system, a controller of ball position using Lyapunov function is designed. The LQR control method is proposed

Proceedings of the 2015
IEEE Conference on Robotics and Biomimetics
Zhuhai, China, December 6-9, 2015

AU Ball on Plate Balancing Robot*

Ehsan Ali [1] and Narong Aphiratsakun [2]

***Abstract*— The ball on plate system is the extension of traditional ball on beam balancing problem in control theory. In this paper the implementation of a proportional-integral-derivative controller (PID controller) to balance a ball on a plate has been demonstrated. To increase the system response time and accuracy multiple controllers are piped through a simple custom serial protocol to boost the processing power, and overall performance. A single HD camera module is used as a sensor to detect the ball's position and two RC servo motors are used to tilt the plate to balance the ball. The result shows that by implementing multiple PUs (Processing Units) redundancy and high resolution can be achieved in real-time control systems.**

I. INTRODUCTION

There are two major paths to tackle the problems in control systems: (1) Derive the mathematical model of the system, and formulate the response accordingly. (2) Adopt a model-free approach, and use the feedback mechanism to calculate the response. In systems with high number of active actors the derivation of mathematical model becomes cumbersome, and in some cases impossible, therefore the second approach comes to rescue. Similar ball on plate papers were studied and the observation shows that many of them have used the mathematical model [1], [2], [3], [4] due to the fact that although ball and plate is a non-linear system but derivation of its mathematical model is very straightforward.

PID algorithm usually is run on a single micro-controller to control a given system. There are cases that computational power of a single micro-controller is not enough to cope with the real-time demands of the system. This paper attempt to demonstrated how the computational demand of various system parts can be distributed between separate processing units through simple serial UART communication pathways.

A. Related Previous Works

1) Ball Position Detection Mechanism:

- Single Camera on top of the plate, and then calculate the position by image processing techniques [3], [8].
- Two Cameras on top of the plate, and then calculate the position by image processing techniques [4], [7].
- Grid of infrared sensors.
- Attach an infrared-ultrasonic sensor to the ball and track the ball in 3D dimension.
- Resistive or capacitive touchscreen grid [2].
- Photo-transistor sensors triggered by monochromatic sharp beams of laser light [5].

*This work was supported by Assumption University of Thailand

[1]Ehsan Ali is with the Department of Computer Engineering, Assumption University, Thailand ehssan.aali@gmail.com

[2]Narong Aphiratsakun with Faculty of Mechatronics Engineering, Assumption University, Thailand nott.notty@gmail.com

2) Actuation Mechanism:

- DC motors attached to cables and pulleys.
- Servo motors attached to arms [2], [3], [5], [8].
- Linear actuators.
- Industrial Robot Arm [4], [7].

3) Control Mechanism:

- PID control on a single micro-controller [5].
- Matlab/Simulink-based real-time control prototyping application on a single micro-controller [1], [2], [3].
- PC-Based Controller, RT Linux [7].
- More than one PUs [8].

The section 2 explains the details of our system, the mechanical specifications, and Solidworks models. The section 3 covers the control method and the details of PUs involved, system flowcharts, and the specifications of the serial UART custom protocol. The section 4 provides the outcome of conducted experiments and graph results. Finally the section 5 concludes the paper.

II. AU BALL ON PLATE BALANCING ROBOT

Meanwhile there are numerous papers which have demonstrated the PID control [5], [6]. There are basically three major challenges needed to be addressed to have a functional system:

- A method to find the ball position in real time.
- The actuation mechanism to tilt the plate accordingly.
- The controller itself that sits between sensors and actuators.

First, the summarization of the methods used in the previous works to face the above challenges are provided. Second, the complete details of our own work is laid out.

A. Our System

For control mechanism a Raspberry Pi 2 board is used which equips an 700 MHz ARM1176JZF-S processor solely for image processing task and a STM32F407 board that includes an 168 MHz ARM processor solely to control the servo motors. The PID control and all other algorithms are implemented in pure C/C++ programming language which has maximized the performance of the system.

Actuation mechanism is based on RC servo motors which are connected to the beneath of the plate through arms and ball-sphere joints.

The ball position detection is done by capturing frames with a single Raspberry Pi camera module that is placed on top of the plate. Image processing techniques are used to detect the ball position per frame and send appropriate control signals to drive the servos accordingly.

978-1-4673-9675-2/15/$31.00 © 2015 IEEE

Ingeniare. Revista chilena de ingeniería, vol. 28 Nº 1, 2020, pp. 6-15

Control of a ball-and-plate system using a State-feedback controller

Control de un sistema bola-placa utilizando un controlador por realimentación de estados

David Núñez[1] Gustavo Acosta[1]* Jovani Jiménez[2]

Recibido 17 de noviembre de 2017, aceptado 08 de octubre de 2018
Received: November 17, 2017 Accepted: October 08, 2018

ABSTRACT

A ball-and-plate platform is a common example of the dynamical system used to apply modeling and control concepts studied in some engineering courses. In this kind of systems, a spherical body is positioned on a mobile plate with two degrees of freedom. The system is operated by two servo motors that cause displacement of the sphere in the x and y axis in the plane of the platform. It is known that such a system is highly unstable and non-linear. Initially, from a phenomenological analysis we obtained a state-space representation of the system. Then, a state feedback controller using the pole assignment method was designed to maintain a desired position of the sphere over the plane. A computer vision technique allowed us measuring the two-dimensional position of the sphere over the plane in real time. Model of the plant under study and controller design for sphere positioning were experimentally validated considering different set-points. Obtained results let us know that our plant-controller is stable and responds satisfactorily to position disturbances around a reference point.

Keywords: Ball and beam, ball and plate, transfer function, state feedback, state observer.

RESUMEN

Una plataforma bola-placa es un ejemplo común de sistema dinámico utilizado para aplicar los conceptos de modelado y control estudiados en algunos cursos de ingeniería. En este tipo de sistemas, un cuerpo esférico se coloca sobre una placa móvil con dos grados de libertad. El sistema es accionado por dos servomotores que provocan el desplazamiento de la esfera en los ejes x e y en el plano de la plataforma. Se sabe que dicho sistema es altamente inestable y no lineal. Inicialmente, a partir de un análisis fenomenológico se obtuvo una representación del sistema en el espacio de estado. A continuación, un controlador de realimentación de estado basado en el método de asignación de polos fue diseñado para mantener una posición deseada de la esfera sobre el plano. Una técnica de visión por computador fue usada para medir la posición bidimensional de la esfera sobre el plano en tiempo real. El modelado de la planta en estudio y el diseño del controlador para el posicionamiento de la esfera se validaron experimentalmente considerando diferentes puntos de ajuste. Los resultados obtenidos mostraron que el controlador diseñado es estable y responde satisfactoriamente a las perturbaciones de posición de la esfera alrededor del punto de referencia.

Palabras clave: Sistema barra-esfera, sistema Plano-Esfera, función de transferencia, realimentación de estados, observador de estados.

[1] Grupo de Investigación ICARO. Facultad de Ingeniería. Politécnico Colombiano Jaime Isaza Cadavid. Cra 48 Nº 7-151. Medellín, Colombia. E-mail: juan_nunez04091@elpoli.edu.co; gaacosta@elpoli.edu.co
[2] GIDIA: Grupo de investigación y desarrollo en inteligencia artificial. Facultad de Minas. Universidad Nacional de Colombia. Cl. 80 #65-223. Medellín, Colombia. Email: jajimen1@unal.edu.co
* Autor de correspondencia: gaacosta@elpoli.edu.co

122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Paper ID #12313

Development of a Ball and Plate System

Dr. Chan Ham, Kennesaw State University

He is an Associate Professor in Mechatronics Engineering at the Kennesaw State University. He has over fifteen year experience in Mechatronics education and research.

Mohsin Mohammad Taufiq, Mechatronics Engineering

Mohsin Mohammad Taufiq is a senior year undergraduate student at Southern Polytechnic State University perusing a B.S. in Mechatronics Engineering. Currently he is assisting Ph.D. Chan Ham of Mechatronics department at the Southern Polytechnic State University in research. Email: mohsin.taufiq@gmail.com

Results in Physics 8 (2018) 153–159

Contents lists available at ScienceDirect

Results in Physics

journal homepage: www.journals.elsevier.com/results-in-physics

Modeling and control for a magnetic levitation system based on SIMLAB platform in real time

Mundher H.A. Yaseen [a,*], Haider J. Abd [b]

[a] Gaziantep University, Electrical & Electronics Engineering Department, Gaziantep, Turkey
[b] Babylon University, College of Engineering, Department of Electrical Engineering, Iraq

ARTICLE INFO

Article history:
Received 19 October 2017
Received in revised form 21 November 2017
Accepted 21 November 2017
Available online 5 December 2017

Keywords:
Magnetic levitation system
Linear Quadratic Regulator (LQR)
PID control
Lead compensation

ABSTRACT

Magnetic Levitation system becomes a hot topic of study due to the minimum friction and low energy consumption which regards as very important issues. This paper proposed a new magnetic levitation system using real-time control simulink feature of (SIMLAB) microcontroller. The control system of the maglev transportation system is verified by simulations with experimental results, and its superiority is indicated in comparison with previous literature and conventional control strategies. In addition, the proposed system was implemented under effect of three controller types which are Linear–quadratic regulator (LQR), proportional–integral–derivative controller (PID) and Lead compensation. As well, the controller system performance was compared in term of three parameters Peak overshoot, Settling time and Rise time. The findings prove the agreement of simulation with experimental results obtained. Moreover, the LQR controller produced a great stability and homogeneous response than other controllers used. For experimental results, the LQR brought a 14.6%, 0.199 and 0.064 for peak overshoot, Setting time and Rise time respectively.

Introduction

Magnetic levitation technology is a perfect solution to achieve better performance for many motion systems, e.g., precision positioning, manipulation, suspension, and haptic interaction due to its non-contact, non-contamination, multi-Degrees-Of-Freedom (DOF), and long-stroke characteristics [1–4]. One of the features of maglev systems is reduction of imagination, which makes them enjoyable in the field of real-life applications [5], which are transportation systems [6], wind tunnel levitation [7], magnetic bearing systems and anti-vibration table [8]. These systems are inherently nonlinear and unstable as well. Therefore, the maglev system is also an interesting issue to confirm the performance of control schemes. However, many techniques were used to control these systems [9]. In [10] a real maglev system is controlled using PID. In [11], a feed forward multilayer neural network was used to model the system, in which learning and control is done simultaneously. As well as some works were done based on neural networks. Active neural networks for the pattern recognition are designed and implemented in [12]. Also, efficient techniques of adaptive controllers are investigated in [13,14]. In [15], stable neural controllers of nonlinear systems are designed. In [16], new iterative adaptive dynamic programming based optimum controllers are suggested and tested. Various methods for PI controller are design and have been tested [17–20].

However, the magnetic levitation system has unstable nonlinear dynamics which should be taken in count. Most of the contributions require measurements of position, velocity and electric current, and thus state observers should be synthesized to estimate the unavailable signals of the nonlinear dynamical system. Furthermore, it needs to design complex systems and some of these systems are costly. Considering the mention study, a controller to stabilize this system should be of great interest.

In this study, a model and a controller are introduced foran active method to control the maglev system based on SIMLAB platform. The maglev system used was verified experimentally and with simulations as well .The control systems were compared under various parameters. The findings showed an agreement for both simulation and experiment results.

Maglev system model

In this section, the construction of the magnetic levitation system and its components will be explained.

* Corresponding author.
E-mail addresses: mundheryaseen@gmail.com, my13436@mail2.gantep.edu.tr (M.H.A. Yaseen).

https://doi.org/10.1016/j.rinp.2017.11.026
2211-3797/

MINI PROJECT COMPETITION

National Education Society ®

J N N COLLEGE OF ENGINEERING (JNNCE)

SHIVAMOGGA-577204

(Approved by AICTE, New Delhi,Certified by UGC 2f & 12B,Accredited by NAAC -'B', UG programs CE, ME, EEE,ECE,CSE,ISE, TCE Accredited by NBA: 1.7.2020 TO 30.6.2022, Recognized by Govt. of Karnataka and Affiliated to VTU, Belagavi)

Department of Electronics and Communication Engineering

MINI PROJECT COMPETITION

Certificate

This is to certify that the project titled "Ball Balancing PID System Using Image processing" undertaken by 6th Semester Electronics & Communication Engineering department project team B7 - Amoghavarsha S G (4JN18EC010), Kiran C N (4JN18EC041) and Niranjana Jois H C (4JN18EC057) has been conferred the VLB Best Project Award in the Mini Project Competition held at J. N. N. College of Engineering, Shivamogga, Karnataka on 21.8.2021.

Dr. S V Sathyanarayana
Head of the Department

Dr. Manjunatha P
Principal, JNNCE

INNOVATIVE PROJECT EXHIBITION

National Education Society (R.)

AMRUTHA MAHOTSAVA CELEBRATION

J N N COLLEGE OF ENGINEERING, SHIVAMOGGA

JNNCE - Tech Anveshan - 2022

Innovative Project Exhibition

CERTIFICATE

This certificate is awarded to

Mr/Mrs Amoghavarsha SG

for outstanding participation in JNNCE-Tech Anveshan 2022 on 27th and 28th May, 2022 at JNNCE, Shivamogga

Dr. Santhosha
Event Coordinator
Asst. Prof., MBA, JNNCE

Mr. G. Suresh
Event Coordinator
Placement Officer, JNNCE

Dr. Manjunatha P.
Dean Academic, JNNCE, Shivamogga

Dr. K. Nagendra Prasad
Principal, JNNCE, Shivamogga

Arduino Servo control code

```
#include <Servo.h>

Servo xServo;
Servo yServo;

int xServoVal;
int yServoVal;
int cmd_len = 0;
char cmd[11];
int i;

void setup() {
    Serial.begin(115200);
    xServo.attach(5);
    yServo.attach(6);
}

void loop() {
    if (Serial.available() > 0){
        cmd_len = Serial.readBytesUntil('$', cmd, 10);
        cmd[cmd_len] = '\0';
        // find index of separator '#'
        for(i=0; cmd[i] != '#' && cmd[i] != '\0' && i<=cmd_len; i++);
        // make data at separator '#' as '\0'
        cmd[i] = '\0';
        // atoi only reads till first '\0' character
        xServoVal = atoi(cmd);
        // atoi reads from first '\0' till secound '\0' which is at index `
                                        cmd_len`
        yServoVal = atoi(cmd + i + 1);
        if (xServoVal >= 500 && xServoVal <= 2500)
        xServo.writeMicroseconds(xServoVal);
        if (yServoVal >= 500 && yServoVal <= 2500)
        yServo.writeMicroseconds(yServoVal);
    }
}
```

Code Listing A.1: Arduino Servo control code

qt_plotter function

```
import sys, signal
import numpy as np
from collections import deque
from pyqtgraph.Qt import QtGui, QtCore
import pyqtgraph as pg

def qt_plotter(setpoint_plot, ball_pose_plot, pid_plot, plot_terminate,
                                        max_limit=800, size=800, identifier='
                                        '):

    app = QtGui.QApplication([])
    timer = QtCore.QTimer()

    win = pg.GraphicsWindow(title=identifier, size=(1000, 900))
    t_vec = np.linspace(0, 1, size)

    graph1 = win.addPlot(title='position v/s time')
    graph1.setRange(yRange=[0, max_limit])
    graph1.addLegend()
    set_x_vec = deque(np.zeros(size, dtype='uint8'))
    set_y_vec = deque(np.zeros(size, dtype='uint8'))
    set_x_curve = graph1.plot(t_vec, set_x_vec, pen=(0, 0, 255, 255), name=
                                        "set_x_curve")
    set_y_curve = graph1.plot(t_vec, set_y_vec, pen=(255, 0, 0, 255), name=
                                        "set_y_curve")

    ball_x_vec = deque(np.zeros(size, dtype='uint8'))
    ball_y_vec = deque(np.zeros(size, dtype='uint8'))
    ball_x_curve = graph1.plot(t_vec, ball_x_vec, pen=(0, 255, 255, 128),
                                        name="ball_x_curve")
    ball_y_curve = graph1.plot(t_vec, ball_y_vec, pen=(255, 150, 0, 128),
                                        name="ball_y_curve")

    win.nextRow()
    graph2 = win.addPlot(title='error v/s time')
    graph2.setRange(yRange=[-max_limit, max_limit])
    graph2.addLegend()
    error_x_vec = deque(np.zeros(size, dtype='uint8'))
    error_y_vec = deque(np.zeros(size, dtype='uint8'))
```

```
    error_xy_vec = deque(np.zeros(size, dtype='uint8'))

    error_x_curve = graph2.plot(t_vec, error_x_vec, pen='b', name="
                                        error_x_curve")
    error_y_curve = graph2.plot(t_vec, error_y_vec, pen='r', name="
                                        error_y_curve")
    error_xy_curve = graph2.plot(t_vec, error_xy_vec, pen='w', name="
                                        error_xy_curve")

    win.nextRow()
    graph3 = win.addPlot(title='pid v/s time')
    graph3.setRange(yRange=[-max_limit, max_limit])
    graph3.addLegend()
    pid_x_vec = deque(np.zeros(size, dtype='uint8'))
    pid_y_vec = deque(np.zeros(size, dtype='uint8'))

    pid_x_curve = graph3.plot(t_vec, pid_x_vec, pen='b', name="pid_x_curve"
                                        )
    pid_y_curve = graph3.plot(t_vec, pid_y_vec, pen='r', name="pid_y_curve"
                                        )

    def updateGraph():
        try:
            set_pose_x, set_pose_y = setpoint_plot
            ball_pose_x, ball_pose_y = ball_pose_plot
            pid_x_value, pid_y_value = pid_plot
            error_x = set_pose_x - ball_pose_x
            error_y = set_pose_y - ball_pose_y

            set_x_vec.popleft()
            set_x_vec.append(set_pose_x)
            set_y_vec.popleft()
            set_y_vec.append(set_pose_y)
            set_x_curve.setData(t_vec, set_x_vec)
            set_y_curve.setData(t_vec, set_y_vec)

            ball_x_vec.popleft()
            ball_x_vec.append(ball_pose_x)
            ball_y_vec.popleft()
            ball_y_vec.append(ball_pose_y)
```

```
            ball_x_curve.setData(t_vec, ball_x_vec)
            ball_y_curve.setData(t_vec, ball_y_vec)

            error_x_vec.popleft()
            error_x_vec.append(error_x)
            error_y_vec.popleft()
            error_y_vec.append(error_y)
            error_xy_vec.popleft()
            error_xy_vec.append(((error_y)**2 + (error_x)**2)**0.5)
            error_x_curve.setData(t_vec, error_x_vec)
            error_y_curve.setData(t_vec, error_y_vec)
            error_xy_curve.setData(t_vec, error_xy_vec)

            pid_x_vec.popleft()
            pid_x_vec.append(pid_x_value)
            pid_y_vec.popleft()
            pid_y_vec.append(pid_y_value)
            pid_x_curve.setData(t_vec, pid_x_vec)
            pid_y_curve.setData(t_vec, pid_y_vec)


            app.processEvents()

            if plot_terminate.value:
                timer.stop()
                win.close()
        except KeyboardInterrupt:
                plot_terminate.value = 1
                timer.stop()
                win.close()

    timer.timeout.connect(updateGraph)
    timer.start(t_vec[1])
    signal.signal(signal.SIGINT, signal.SIG_DFL)
    sys.exit(app.exec_())
```

Code Listing A.2: qt_plotter function

iot.py library

```
import threading
import paho.mqtt.client as mqtt
from time import sleep

class MQTT:
    def __init__(self, broker_url, broker_port=1883):
        self.broker_url = broker_url
        self.broker_port = broker_port
        self.pub_client_exits = False
        self.lock = threading.Lock()

    @staticmethod
    def on_connect(client, userdata, flags, rc, sub_topic, qos):
        print(f"Connected With Result Code {rc}")
        (rc, mid) = client.subscribe(sub_topic, qos=qos)
        if not rc:
            print(f'subscribed to topic: {sub_topic}, qos: {qos}')
        return rc

    @staticmethod
    def sub_on_connect(func, sub_topic, qos):
        def wrapper(client, userdata, flags, rc):
            return func(client, userdata, flags, rc, sub_topic, qos)
        return wrapper

    def subscribe_thread_start(self, on_message, sub_topic, qos):
        try:
            client = mqtt.Client()
            on_connect = MQTT.sub_on_connect(MQTT.on_connect, sub_topic,
                                             qos)
            client.on_connect = on_connect
            client.on_message = on_message
            client.connect(self.broker_url, self.broker_port)
            client.loop_start()
            return 0
        except Exception as e:
            print(e)
            return -1

```

```
    def publish_reuse_client(self, pub_topic, payload, qos, timeout=None):
        with self.lock:
            try:
                if not self.pub_client_exits:
                    self.pub_client = mqtt.Client()
                    self.pub_client.connect(self.broker_url, self.
                                            broker_port)
                    self.pub_client.loop_start()
                    self.pub_client_exits = True
                pub_info = self.pub_client.publish(pub_topic, payload, qos=
                                            qos, retain=False)
                if timeout:
                    sleep(timeout)
                else:
                    pub_info.wait_for_publish()
                return pub_info[0]

            except Exception as e:
                print(e)
                return -1

    def publish(self, pub_topic, payload, qos):
        try:
            client = mqtt.Client()
            client.connect(self.broker_url, self.broker_port)
            client.loop_start()
            pub_info = client.publish(pub_topic, payload, qos=qos, retain=
                                        False)
            pub_info.wait_for_publish()
            client.loop_stop()
            client.disconnect()
            return pub_info[0]
        except Exception as e:
            print(e)
            return -1
```

Code Listing A.3: iot.py library

Animate method

```
def animate(self, ball_pose, radius=15):
    st_time = perf_counter()
    ball_img = self.ball_imgs[radius]
    ball_height, ball_width = ball_img.shape[:2]
    tmin_x = ball_pose[0] - radius
    tmin_y = ball_pose[1] - radius
    tmax_x = ball_pose[0] + radius
    tmax_y = ball_pose[1] + radius
    bmin_x, bmin_y, bmax_x, bmax_y = 0, 0, ball_width, ball_height
    if (tmin_y > - ball_height) and (tmax_y < self.table_height +
                                     ball_height) and (tmin_x > -
                                     ball_width) and (tmax_x < self.
                                     table_width + ball_width):
        if tmin_x < 0:
            bmin_x = abs(tmin_x)
            tmin_x = 0
        if tmax_x > self.table_width:
            bmax_x = ball_width - (tmax_x - self.table_width)
            tmax_x = self.table_width
        if tmin_y < 0:
            bmin_y = abs(tmin_y)
            tmin_y = 0
        if tmax_y > self.table_height:
            bmax_y = ball_height - (tmax_y - self.table_height)
            tmax_y = self.table_height
        self.return_img = np.array(self.table_img)
        return_img_roi = self.return_img[tmin_y:tmax_y, tmin_x:tmax_x]
        cv2.add(return_img_roi, ball_img[bmin_y:bmax_y, bmin_x:bmax_x], dst
                                     =return_img_roi)
    else:
        self.return_img = np.array(self.table_img)
    fps.ptime_update((perf_counter() - st_time)*1000)
    fps.fps_update()
    if fps.ready():
        print(f"\rFPS: {fps.get_fps()} | Ptime: {fps.get_ptime()} ms".ljust
                                     (65), end='')
        fps.reset()
```

Code Listing A.4: Animate method

PID __call__ method

```
def __call__(self, input_):

    if not self.auto_mode:
        return self._last_output

    now = self._current_time()

    dt = now - self._last_time if (now - self._last_time) else 1e-16
    if dt < 0:
        print(f'dt has negative value {dt}')

    if self.sample_time is not None and dt < self.sample_time and self.
                                        _last_output is not None:
        # Only update every sample_time seconds
        return self._last_output

    if(self._last_input is not None):
        # y(t) = (alpha * x(t)) + (beta * y(t - 1))
        input_ = int((self.exp_filter_alpha * input_) + (self.
                                        exp_filter_beta * self._last_input))

    # Compute error terms
    error = self.setpoint - input_
    d_input = input_ - (self._last_input if (self._last_input is not None)
                                        else input_)

    # Check if must map the error
    if self.error_map is not None:
        error = self.error_map(error, self.axis)

    # Compute the proportional term
    if not self.proportional_on_measurement:
        # Regular proportional-on-error, simply set the proportional term
        self._proportional = self.Kp * error
    else:
        # Add the proportional error on measurement to error_sum
        self._proportional -= self.Kp * d_input

    # Compute integral and derivative terms
```

```
37      self._integral += self.Ki * error * dt
38      self._integral = self.clamp(self._integral)  # Avoid integral windup
39
40      self._derivative = -self.Kd * d_input / dt
41
42      # Compute final output
43      output = self._proportional + self._integral + self._derivative
44      output = self.clamp(output)
45
46      # Keep track of state
47      self._last_output = output
48      self._last_input = input_
49      self._last_time = now
50
51      return output
```

Code Listing A.5: PID __call__ method