**UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ**

**Colegio de Ciencias e Ingenierías**

# Reverse Engineering of a Ball and Plate System with Control Strategies for Stabilization.

**Emilia Villarroel Figueroa**

**Electrónica y Automatización**

Trabajo de fin de carrera presentado como requisito

para la obtención del título de

Ingeniera en Electrónica

Quito, Enero de 2024

# UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

**Colegio de Ciencias e Ingenierías**

**HOJA DE CALIFICACIÓN**

**DE TRABAJO DE FIN DE CARRERA**

**Reverse Engineering of a Ball and Plate System with Control Strategies for Stabilization.**

**Emilia Villarroel Figueroa**

Nombre del profesor Tutor: Óscar Camacho, Ph.D.

Quito, Enero de 2024

# © DERECHOS DE AUTOR



Nombres y apellidos: Emilia Villarroel Figueroa

Código: 00212093

C.I.: 1722489802

Fecha: Quito, Enero de 2024

# ACLARACIÓN PARA PUBLICACIÓN

**Nota:** El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en `http://bit.ly/COPETheses`

# UNPUBLISHED DOCUMENT

**Note:** The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on `http://bit.ly/COPETheses`

# RESUMEN

Este artículo detalla la ingeniería inversa de un sistema integrado ball-and-plate, haciendo hincapié en un análisis comparativo de las estrategias de control de la estabilización. El objetivo del controlador es mantener un movimiento preciso de la bola sobre una superficie plana, asegurando un posicionamiento consistente. El trabajo pretende diseñar un regulador Arduino versátil, que permita la adaptabilidad para esquemas de control avanzados. Esta adaptación es crucial debido a las limitaciones del regulador original, que dificulta cualquier capacidad de edición. La transición a un Arduino Uno comercial permite futuras modificaciones, ofreciendo la flexibilidad necesaria para implementar y mostrar diversas estrategias de control. Para evaluar el rendimiento del sistema en la nueva placa, se implementaron y compararon en la práctica dos estrategias de control de estabilización: el control proporcional derivativo (PD) clásico y el PD no lineal.

***Palabras clave*:** Ingeniería inversa, sistema embebido, control y estabilización, sistema Ball and Plate, Arduino Uno, estrategias de control, PD, PD no lineal.

# ABSTRACT

This work details the reverse engineering of an embedded ball-and-plate system, emphasizing a comparative analysis of stabilization control strategies. The controller's focus is maintaining precise ball motion on a flat surface, ensuring consistent positioning. The work aims to deploy a versatile Arduino regulator, allowing for adaptability for advanced control schemes. This adaptation is crucial due to the limitations of the original closed regulator, which hinders any editing capabilities. The transition to a commercial Arduino Uno enables future modifications, offering the flexibility to implement and showcase various control strategies. To assess system performance on the new board, two stabilization control strategies were practically implemented and compared: classical proportional derivative control (PD) and nonlinear PD.

***Keywords*:** Reverse engineering, embedded system, control and stabilization, Ball and Plate System, Arduino Uno, control strategies, PD, nonlinear PD.

# CONTENTS

# LIST OF FIGURES



# LIST OF TABLES

# DEDICATORIA

Dedico este logro a mi yo del pasado, quien no pensaba que iba a llegar a este punto tras tantas frustraciones y sentimientos de no poder más. También, a mis padres, quienes son mi principal fuente de apoyo y cariño, y estuvieron ahí para mí en cada situación de mi vida universitaria.

También lo dedico a todas las experiencias futuras.

# AGRADECIMIENTO

Quiero expresar mi agradecimiento a todas aquellas personas que me permitieron llegar a este punto de mi vida, en educación y vida. Sobre todo, a mis padres, quienes me apoyaron en cada momento, incluso en momentos de frustración y cansancio, me impulsaron a seguir avanzando y a aprender de mis errores.

También quiero agradecer al profesor de esta tesis, Oscar Camacho. Su pasión y dedicación sobre temas de control son lo que principalmente me llevaron a la realización de este proyecto, por lo que le tengo mucho aprecio.

# CHAPTER 1

## INTRODUCTION

Balancing systems are one of the most popular and challenging test platforms for control engineers. Such systems are the traditional cartpole system (inverted pendulum), the ball-and-beam system (BnB), the multiple inverted pendulums, the ball-and-plate system (BnP), etc (Mohajerin et al., 2010).

This work focuses on the ball and plate system, which is known to be highly unstable and nonlinear (Nuñez et al., 2020). It consists of two mechanically coupled servomotors on the axis of the platform to provide two degrees of freedom, which cause the movement of the sphere on the x and y axes in the plane of the platform (Nuñez et al., 2020). The aim of this system is to bring the ball rolling freely through the plate to a desired position; therefore, the investigations have focused on stabilizing the ball on the platform accurately and with a fast stabilization time.

The system has been the subject of numerous studies and experiments due to its complexity and challenge. A similar project was made by Sanchez Manzo who proposed the design and construction of a mechatronic ball-and-plate system of two degrees of freedom with the objective of controlling the position of a ball on a moving platform using Proportional Integral Derivative (PID) control techniques (Manzo, 2018). Another similar work is from (Castro-Mendoza et al., 2022) which explores the design and construction of a ball-and-plate system and implements classical controller techniques. In addition, it also highlights the accessibility and quality of the prototype, making it suitable for educational environments.

This work outlines the reverse engineering process (Rekoff, 1985) of an embedded ball-and-plate system, specifically focusing on a comparative examination of stabilization control strategies. The controller's primary objective is maintaining precise ball motion on a level surface, ensuring consistent positioning. This work aims to implement a versatile Arduino regulator that

provides adaptability for advanced control schemes. This adaptability becomes essential due to the constraints imposed by the original closed regulator, limiting any editing capabilities. The transition to a commercial Arduino Uno facilitates future modifications, granting the flexibility to implement and demonstrate various control strategies. To evaluate the system's performance on the new board, two stabilization control strategies, classical proportional derivative control (PD) and nonlinear PD, were practically implemented and compared.

This work is organized as follows. Chapter 2 presents the Fundamentals, explaining the mathematical modeling of a Ball and Plate System. Chapter 3 shows the Real Ball and Plate System considering all the elements obtained. Chapter 4 presents the experimental results of a new Arduino showing the performance of the new board for a classical PD and nonlinear PD. The conclusions are shown in Chapter 5.

# CHAPTER 2

# BALL AND PLATE SYSTEM FUNDAMENTALS

A Ball and Plate System is one of the most popular and important systems, a generalized version of the traditional ball and beam benchmark (Awtar et al., 2002). This system consists of a rigid plate that can be a resistive or capacitive touch panel, which measures the position of the ball with respect to the center or a predetermined point of the plate (Al-Haddad, 2020).

The system is unstable because any impulse or disturbance, no matter how small, causes the ball to move away from its desired point. In addition, it cannot be restored to equilibrium or a predetermined point on the plate without substituting force (Al-Haddad, 2020), which is why it is necessary to implement a controller whose objective is to balance a ball inside the plate at a specific point. If the ball moves from that point, the controller will cause the movement of the screen based on its coordinates to stabilize the ball at its designated location.

## 2.0.1 Mathematical Modeling

The mathematical model used for any mechanical system can be derived using two methods: 1- The Newton method, which is used for systems with multiple degrees of freedom. 2- The Euler-Lagrange method allows us to model the movement of the ball on the plate (Al-Haddad, 2020) (Bolívar-Vincenty and Beauchamp-Báez, 2014) (Morales et al., 2017).

The Lagrangian function is defined as:

$$\mathcal{L}(q, \dot{q}, t) = T(q, \dot{q}, t) - V(q, t) \tag{2.1}$$

The Euler-Lagrange equation is defined as:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = 0 \tag{2.2}$$

To obtain the mathematical model, a free body diagram of the Ball and Plate as shown in Fig. 2.1 is required, as also the following simplifications (Knuplez et al., 2004):

- The ball never loses contact with the plate.

- There is no sliding between the ball and the plate.

- All friction forces and rotational moments are neglected.

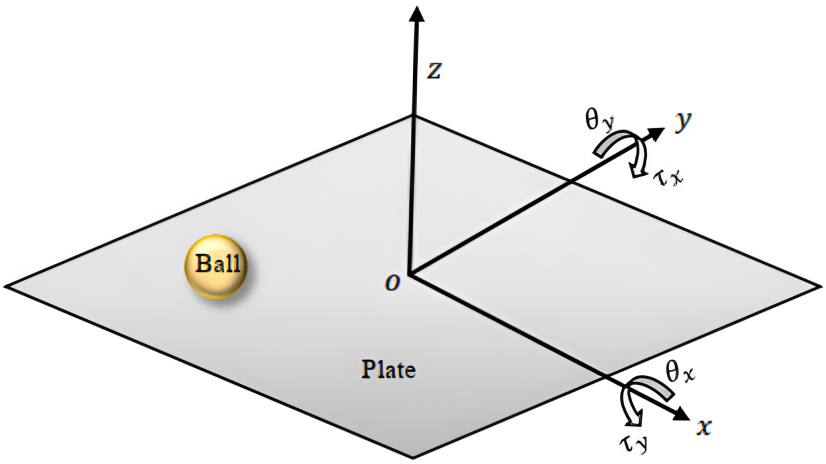


Figure 2.1: Free body diagram of Ball and Plate System.(Al-Haddad, 2020)

From the free-body diagram, it is obtained that the total kinetic energy of the system must be the sum of the kinetic energy of the ball and the kinetic energy of the plate. Then, we obtain the total energy of the system by substituting the total potential energy of the ball and the total kinetic energy of the system into the Eq. (2.1), then applying the Euler-Lagrange equation Eq. (2.2), these nonlinear differential equations are obtained:

$$\left(m + \frac{J_b}{r^2}\right)\ddot{x} - m\dot{\alpha}^2 x - m\dot{\alpha}y\dot{\beta} + mg\sin\alpha = 0 \tag{2.3}$$

$$\left(m + \frac{J_b}{r^2}\right)\ddot{y} - mx\dot{\alpha}\dot{\beta} - m\dot{\beta}^2 y + mg\sin\beta = 0 \tag{2.4}$$

The parameters and variables for (2.3) and (2.4) are shown in Table 2.1.

Table 2.1: Parameters and Variables of Mathematical Model

| **Parameters and Variables of Mathematical Model** | | |
|---|---|---|
| Symbol | Description | Units |
| $x$ | Ball position in the X-axis | cm |
| $y$ | Ball position in the Y-axis | cm |
| $\alpha$ | Angle of the plate in the X-axis | rad |
| $\dot{\alpha}$ | Angular velocity of the plate in the X-axis | rad/s |
| $\beta$ | Angle of the plate in the Y-axis | rad |
| $\dot{\beta}$ | Angular velocity of the plate in the Y-axis | rad/s |
| $g$ | Gravitational acceleration | m/s$^2$ |
| $m$ | Ball mass | kg |
| $J_b$ | Moment of inertia of the ball | kg cm$^2$ |
| $r$ | Radius of the ball | cm |

Taking into account that the movements of the system will be smooth, $\dot{\alpha}$ and $\dot{\beta}$ are assigned as: $\dot{\alpha} \cong 0$, $\dot{\beta} \cong 0$. With this idea and replacing equations (2.3) and (2.4), the following equations were obtained for the X-axis and the Y-axis.

$$\ddot{x} = -\frac{mg\sin(\alpha)}{(m + \frac{J_b}{r^2})} \tag{2.5}$$

$$\ddot{y} = -\frac{mg\sin(\beta)}{(m + \frac{J_b}{r^2})} \tag{2.6}$$

Equations (2.5) and (2.6) are linearized by the Taylor series when the plate of the system keeps the ball in equilibrium at its horizontal position at the operating point $\overline{\alpha} = 0$, $\overline{\beta} = 0$; giving:

$$\ddot{x} = -\frac{mg}{(m + \frac{J_b}{r^2})}\alpha \tag{2.7}$$

$$\ddot{y} = -\frac{mg}{(m + \frac{J_b}{r^2})}\beta \tag{2.8}$$

To solve equations (2.7) and (2.8), it must be taken into account that:

$$g = -980\,\text{cm/s}^2, \quad J_b = \frac{2}{5}mr^2$$

Also, consider the relationship between the angle of the plate and the rotation angles of the actuators (Awtar et al., 2002), for the X-axis and Y-axis:

$$\alpha = \frac{u_x}{10}, \quad \beta = \frac{u_y}{10}$$

# CHAPTER 3

# REAL BALL AND PLATE SYSTEM

The ball and plate system used for the development of this project is shown in Fig. 3.1.

Figure 3.1: Real ball and plate system.(HansaRobot Store, nd)

The acquired ball and plate system includes the following: an Arduino regulator, a two-degree-of-freedom motion platform, a ball, an OLED display, a 4-wire resistive touch screen of 12 inches, a PS2 wired handle, a USB program download cable, and a power supply.

This system also provides links to download original materials; this means that it has a full open source code including control code, coordinate extraction code, schematic, etc., and supports PID parameters online adjustment.

### 3.0.1 Mathematical Modeling

Given the equations outlined in the previous section, one can obtain the mathematical model of a ball-and-plate system, thus, the transfer function of the acquired system, knowing the mass (0.26 kg) and the radius (2 cm) of the system's ball. So, from Eqs. (7) and (8), the following is obtained:

$$\ddot{x} = 70u_x \tag{3.1}$$

$$\ddot{y} = 70u_y \tag{3.2}$$

Applying the Laplace transform in (7) and (8) to obtain the transfer function is shown in (9) and (10) considering null initial conditions:

$$\frac{X(s)}{U_x(s)} = \frac{70}{s^2} \tag{3.3}$$

$$\frac{Y(s)}{U_y(s)} = \frac{70}{s^2} \tag{3.4}$$

Where: $X(s)$, $Y(s)$ are the positions of the ball on the X axis and the Y axis, respectively, and $U_x(s)$, $U_y(s)$ are the rotation angles of the servomotors of the plant (Morales et al., 2017).

### 3.0.2 Elements system analysis

The aim is to comprehensively examine the components outlined in the preceding section, including the Arduino code, regulator, resistive screen, and schematics. Therefore, The objective is to formulate a versatile Arduino regulator conducive to subsequent modifications and integration of advanced control schemes. This imperative arises from the inherent limitation of the supplied initial regulator, which is closed and precludes any capacity for editing.

The initial Arduino regulator comprises an Arduino board and an adaptive overlaying board designed to control the 4-wire resistive touch screen and servo motors. The schematic of the Ball and Plate System in Fig. 3.2 identifies the primary board as an Arduino Uno. Similarly, the schematic in Fig. 3.3 was used to discern the design of the adaptive board.

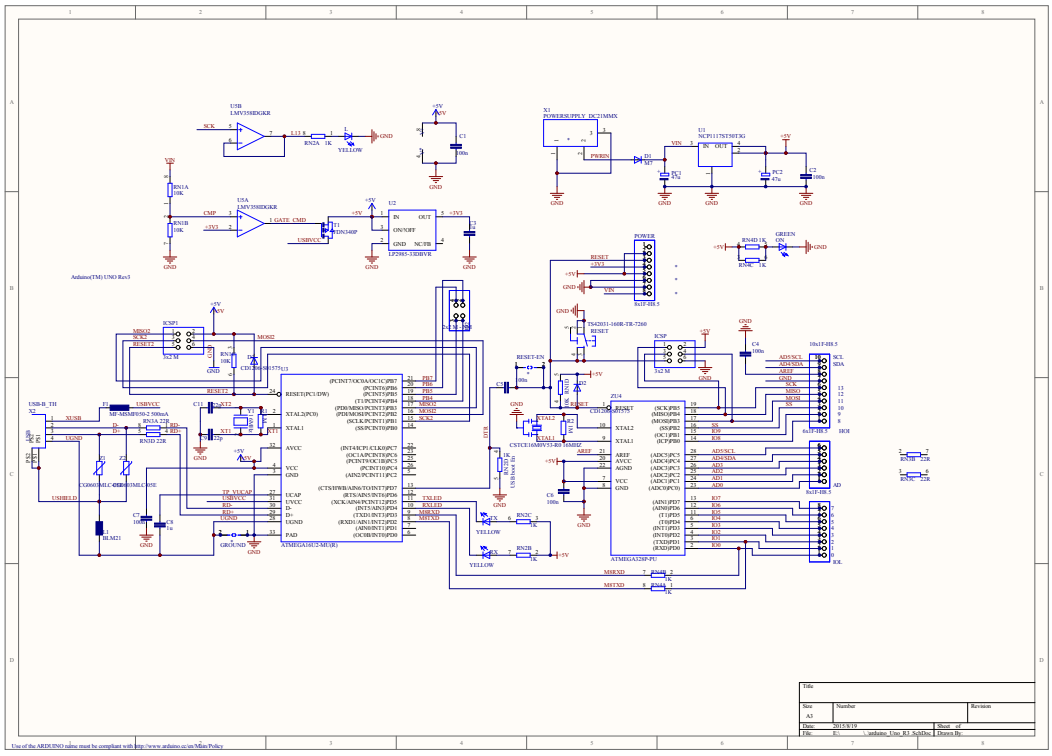

Figure 3.2: Arduino Uno Schematic.(HansaRobot Store, nd)

**Arduino code.**

Firstly, we upload the original Arduino code to a commercial Arduino Uno board. To achieve this, a comprehensive analysis and editing of the supplied code was necessary to ensure compatibility, including the installation of essential libraries such as Servo.h, allowing the control of multiple servo motors using a single timer (Servo-Arduino, 2024), and MsTimer2.h, facilitating an interrupt function through Timer2 (MsTimer2-Arduino, 2024).

Certain adjustments were made, including the removal of the code associated with the OLED on the original adaptive board. This decision stemmed from editing constraints within the code library, as the commercial library did not allow a successful code compilation. Additionally, the code for manual system control via a PS2 interface was omitted, considering that such features would not be implemented on the new board. Following these modifications, the adapted Arduino code could be successfully uploaded to a commercial Arduino Uno board, seamlessly replacing the original one. The complete Arduino code used can be found at 6.1.

By connecting the commercial Arduino Uno board to the original adapter board, the system worked; however, values such as $SetpointX$ and $SetpointY$ that indicate the desired position

of the ball had to be modified, as shown in Table 3.1, because the ball was not in the center of the screen for this new Arduino, likewise, the value $K_d$ was modified.

Table 3.1: Changed values in Arduino code

| **Changed values in Arduino code** | | |
|---|---|---|
| Parameters | Original Values | New Values |
| $K_p$ | 70 | 70 |
| $K_d$ | 50 | 40 |
| $SetpointX$ | 133 | 75 |
| $SetpointY$ | 100 | 175 |

Thus, allowing the system to operate with the center as the desired position of the Ball and Plate System.

Also, from the Arduino code it was identified that the ball and plate system has a PD controller that is used to hold the ball in a desired position on the platform. The PD controller uses information on the current position of the ball and its velocity to calculate the force required to hold the ball in the desired position. The proportional controller (P) uses only the current ball position information, while the derivative controller (D) uses only the ball velocity information. The combination of both controllers (PD) allows a better stabilization of the system (Mijares et al., 2020).

**Arduino System Adapter Board.**

The original adaptive board placed in the Arduino allows the correct operation of the servomotors and the 4-wire resistive touch screen of the Ball and Plate System. Therefore, it was necessary to rebuild this board to perform the same function and not depend on the original. Also, knowledge of how to assemble a new one in case of failure is gained.

The operation of the adaptive plate was tested by assembling the schematic provided in Fig. 3.3.

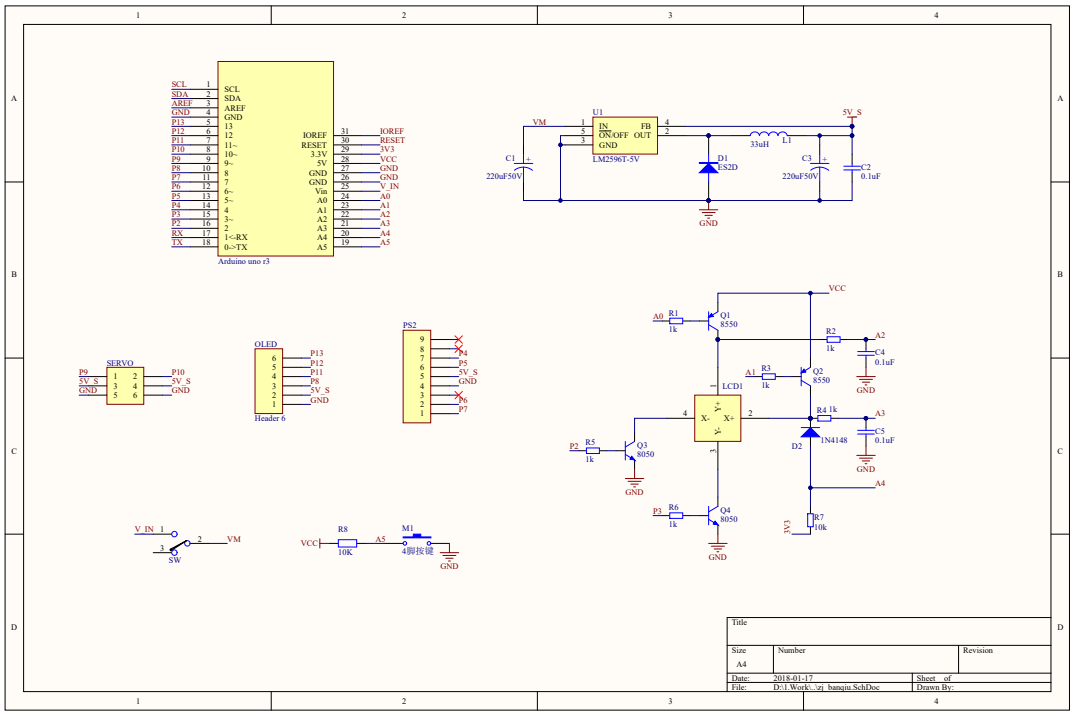


Figure 3.3: Adapter Board Schematic.(HansaRobot Store, nd)

Taking into account that the OLED and PS2 parts were omitted for the new circuit design, since as mentioned, these are not included in the new Arduino code. Another change was that a LM2596 module was used instead of the top circuit beside the Arduino Uno r3, since it is a voltage regulator, and was configured to output 5V.

The term LCD1 in the schematic refers to the 4-wire resistive touch screen that has the system, while M1 is a four-pin button.

The adaptive board circuit was built on a breadboard first to test its operation. But before the assembly, it was identified in the servo motors to which Arduino pin each one was connected. In addition, the 4-wire resistive touch screen identified which side corresponds to the X and Y axes, as well as knowing which to Arduino pin they were connected.

**Servo motors**

A servo motor is an electrical device that is used to rotate an object with the highest precision (Kumar et al., 2021). In this project all the position information detected by the resistive touch screen is used to control the servo motors. The Arduino reads the position data and sends control signals to the servos in this case so that if the ball moves from the desired position, the servo motors move to return the ball to that position.

In addition, the system makes use of two Servo motors shown in Fig. 3.4. By using the Arduino code it was identified which one corresponded to Servo 1 and Servo 2 in the system.


Figure 3.4: Servomotor identification in the Ball and Plate System.

From the wires that each servo has, as shown in Fig. 3.5, it was identified that the brown wire is GND, while the red wire is VCC and the orange wire is the signal which is connected to an Arduino pin.

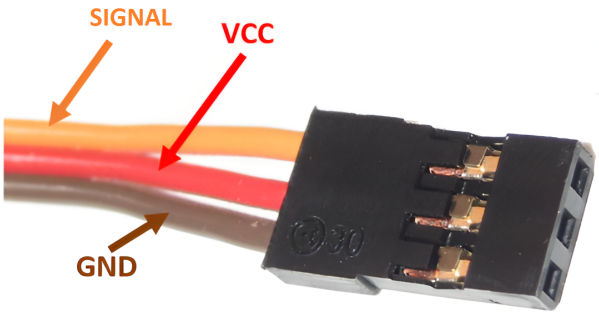


Figure 3.5: Servomotor wire.(Naylamp Mechatronics, nd)

In addition, measuring the continuity of the adapter board allowed to identify which Arduino pin is connected to each signal wire of each servo. All the information obtained is shown in Table 3.2 and this information was used to build the circuit related to the servomotors and the LM2596 according to the schematic in the Fig. 3.3.

Table 3.2: System servomotors

| **System servo motors** | |
|---|---|
| Servo Motors | Arduino pins |
| Servo Motor 1 | 9 (Analog) |
| Servo Motor 2 | 10 (Analog) |

**4-wire Resistive Touch Screen**

The Ball and Plate System consists of a 4-wire resistive touch screen that detects the position of some element, in this case, the ball of the system, using physical pressure, and this screen is managed around the X and Y axis. For the screen used in this system, the 4 wires to be considered were of the following colors: red, blue, green, and black, as shown in Fig. 3.6.

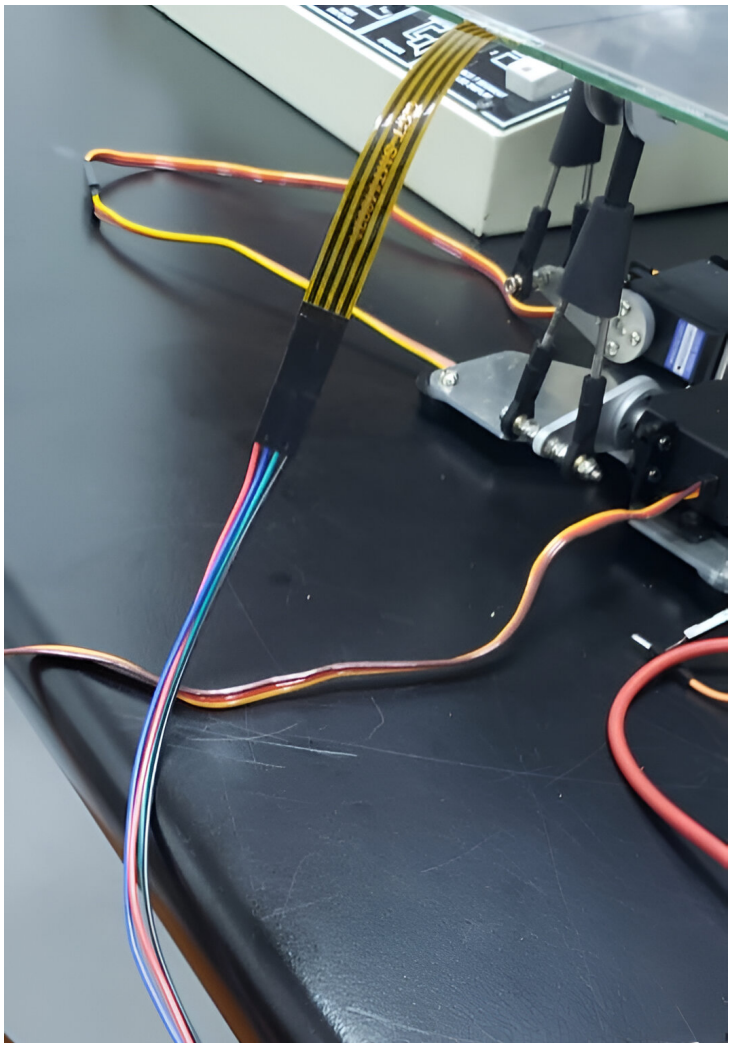

Figure 3.6: Resistive Touch screen Wires

In this resistive touch screen, two wires correspond to the X-axis (X + and X-), and the other two correspond to the Y axis (Y + and Y-). First, it was necessary to check which wires were connected, so the resistance between the wires was measured, and it was identified that the blue and black wires were connected while the red and green wires corresponded to the other junction.

5V was placed in each connection to know which wire junction corresponded to each axis. First, the blue and black wires received the 5V and the measurement was made between the black and green wires; second, the red and green wires received the 5V and the measurement was made between the red and blue wires. In each case, the ball was placed on the screen to observe with a multimeter how the voltage varied and if this variation was in the X or Y axis.

In addition, by measuring the continuity of the adapter board it was possible to identify to which Arduino pin each wire is connected. All the information obtained is shown in Table 3.3, and this information was used to assemble the circuit related to the resistive display (LCD1) according to the schematic of the Fig. 3.3.

Table 3.3: System touchscreen connection

| **System touchscreen connection** | | | |
|---|---|---|---|
| Wire | Arduino pins | Sequence | Axis |
| Red | 2 (Digital) | 4 | X- |
| Blue | 3 (Digital) | 3 | Y- |
| Green | A1 (Analog) | 2 | X+ |
| Black | A0 (Analog) | 1 | Y+ |

# CHAPTER 4

# EXPERIMENTAL RESULTS

### 4.0.1 Implementation of new adapter system

With the necessary understanding of the interconnections and operational principles of both servo motors and the 4-wire resistive touch screen, we proceed to implement the newly embedded Arduino regulator in this part.

#### Adapter Board in a breadboard

Firstly, it assembles the circuit of Fig. 3.3, which is the adapter board of the ball and plate system on a breadboard, and connects this circuit to the Arduino Uno board as shown in Fig. 4.1 to check the operation of the system.

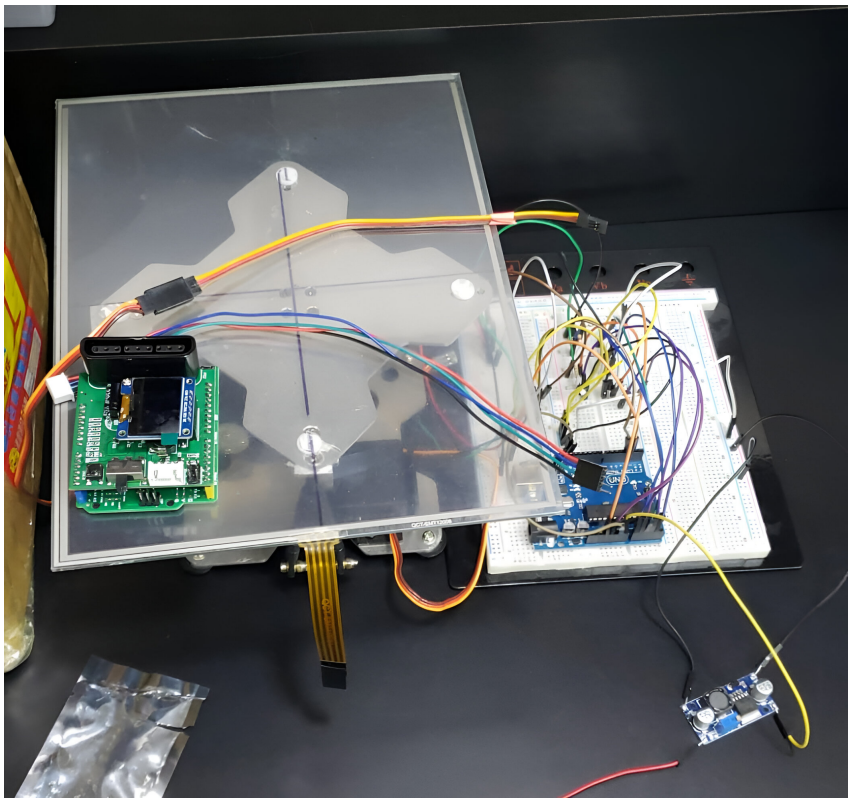

Figure 4.1: The adapter board on a breadboard and connected to the system.

The elements used to build the circuit of the adapter board are described in Table 4.1.

Table 4.1: Electronic Elements used in the circuit

| **Electronic Elements used in the circuit** | |
|---|---|
| Electronic Element | Quantity |
| 1k ohm Resistance | 6 |
| 8050 Transistor | 2 |
| 8550 Transistor | 2 |
| 10k ohm Resistance | 2 |
| 1N4148 Diode | 1 |
| 0.1uF Capacitor | 2 |
| 4-pin push button | 1 |
| LM2596 Module | 1 |

Connecting the breadboard circuit to the commercial Arduino board, the system worked while keeping the same values of $SetpointX$, $SetpointY$, and $Kd$ from Table 3.1.

**Adapter Board - Proteus Design**

With the correct operation of the adapter board in a breadboard, based on Fig. 3.3, the system adapter board schematic was made in Proteus as shown in Fig. 4.2.

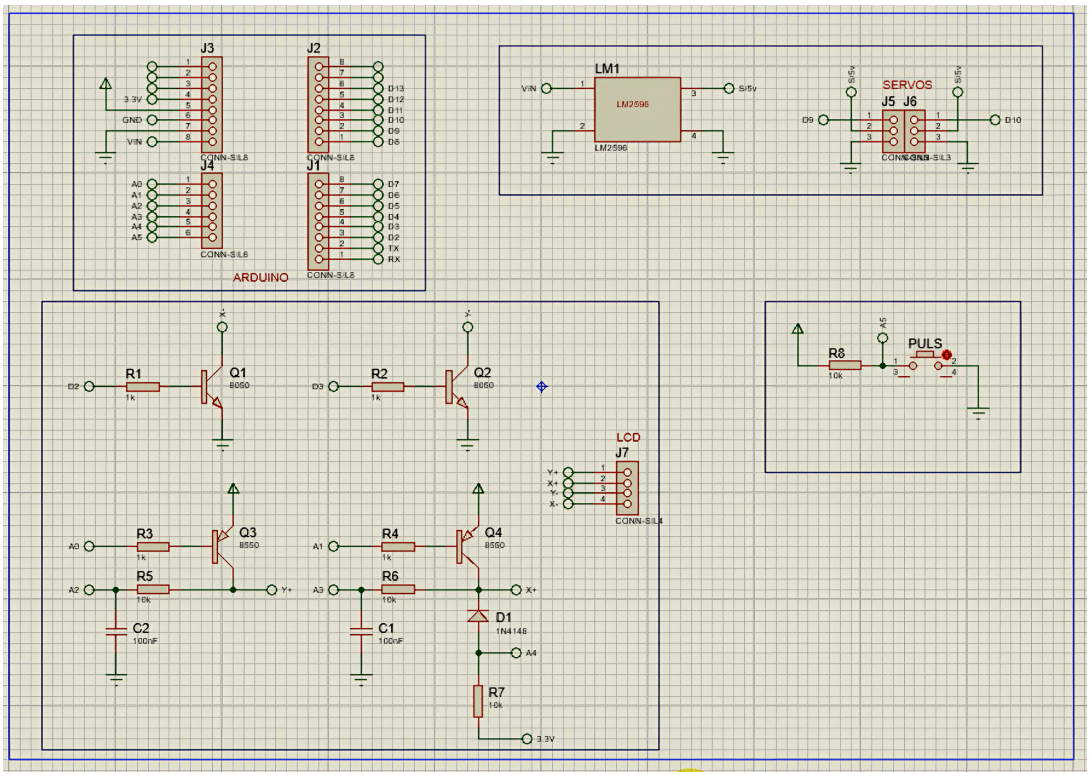


Figure 4.2: Schematic of the Adapter Board in Proteus.

With the schematic already elaborated, the connections and location of the elements on the board were designed as shown in Fig. 4.3, additional Fig. 4.4 shows how the board would look in 3D with the elements already welded.

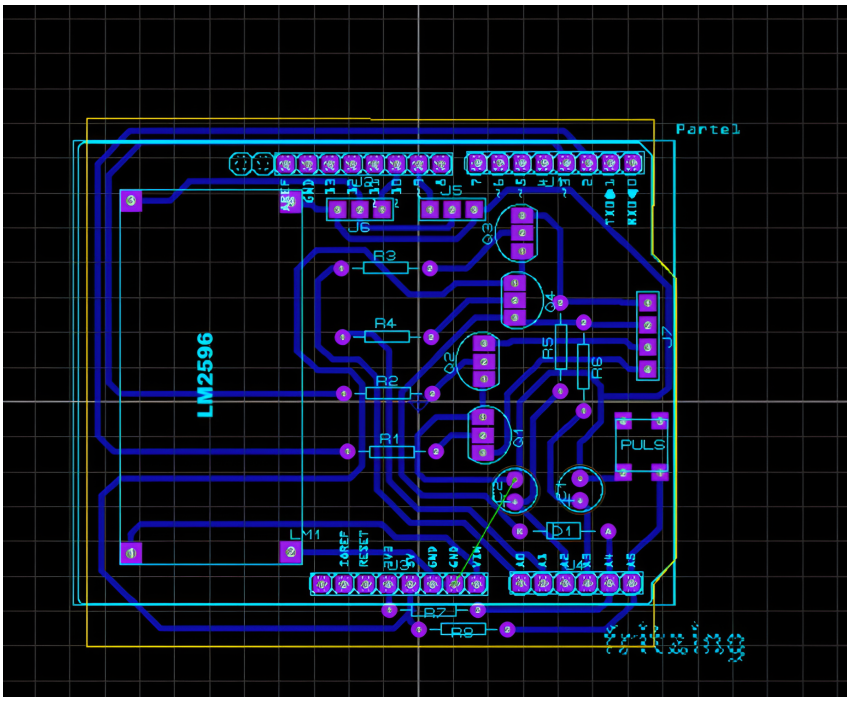


Figure 4.3: Connections of the Adapter Board in Proteus.


Figure 4.4: 3D model of the Adapter Board.

Once the two elements (schematic and connections) were ready, the board design was printed and the elements were soldered together, resulting in what is shown in Fig. 4.5.

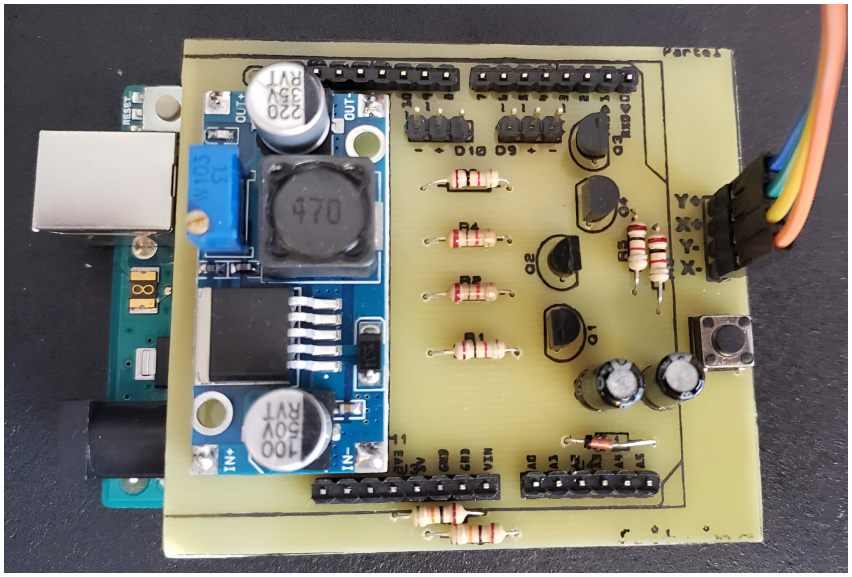


Figure 4.5: New Adapter Board.

In addition to the elements mentioned in Table 4.1. Male headers were placed for the connection of the adapter board with the Arduino Uno and also were placed for the connection of the 4-wire resistive touch screen and the two servo motors to the adapter board.

Fig. 4.6, shows the ball and plate System with the new Arduino board.

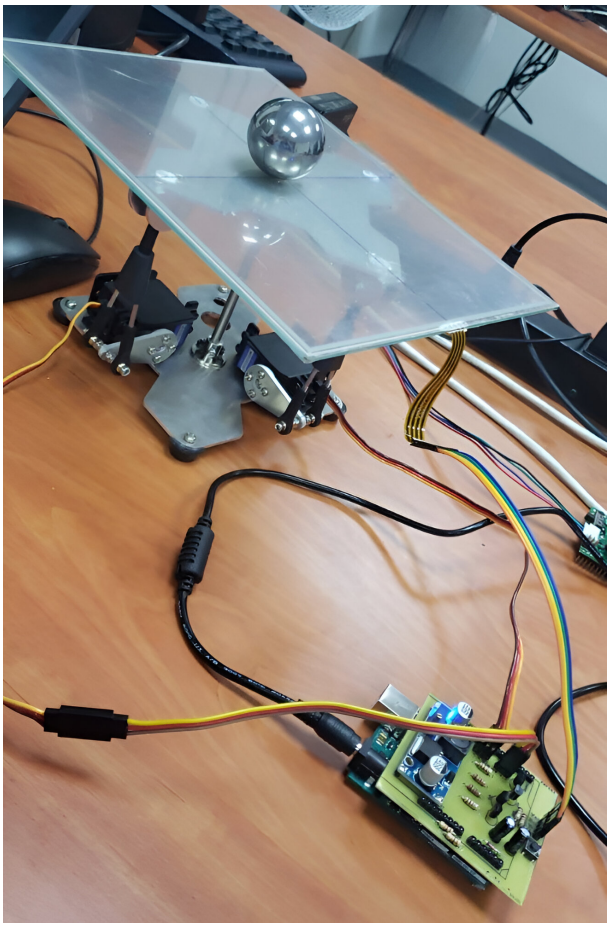

Figure 4.6: New ball and plate system.

### 4.0.2 Controllers comparison for stabilization

With the system already in operation using the new Arduino regulator, two control strategies were applied to evaluate stabilization performance: classical PD vs. nonlinear PD.

The mathematical expression for the PD controller is given by:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} \tag{4.1}$$

Classical PD control is a common methodology used in control systems, where it employs proportional and derivative terms to formulate control signals. The proportional component is the existing error, while the derivative component is the error's rate of change. The objective is to provide stability and reduce overshoot in the control system (Ogata, 2010).

The mathematical expression for the nonlinear PD controller is given by:

$$u(t) = K_p' e(t) + K_d' \frac{de(t)}{dt} \tag{4.2}$$

Where:

$$K_p' = K_p(1 + \alpha|e|)$$

$$K_d' = T_d K_p'$$

And:

$$0 < \alpha < 1$$

According to (Song, 2018), nonlinear PD control introduces a nonlinear element into the control law. This modification is implemented in situations where linear control strategies may fail to improve the response of the system. Nonlinear PD controllers are designed to adapt to specific system characteristics and provide a more dynamic response.

These strategies allow to prove the functionality of the new commercial board and analyze the system's response to perturbations.

Figs. 4.7 and 4.8 show the position of the ball on the resistive touch screen, both with perturbations in the same direction, to compare which controller performs better in this system, and also show that the new commercial board designed functions correctly.

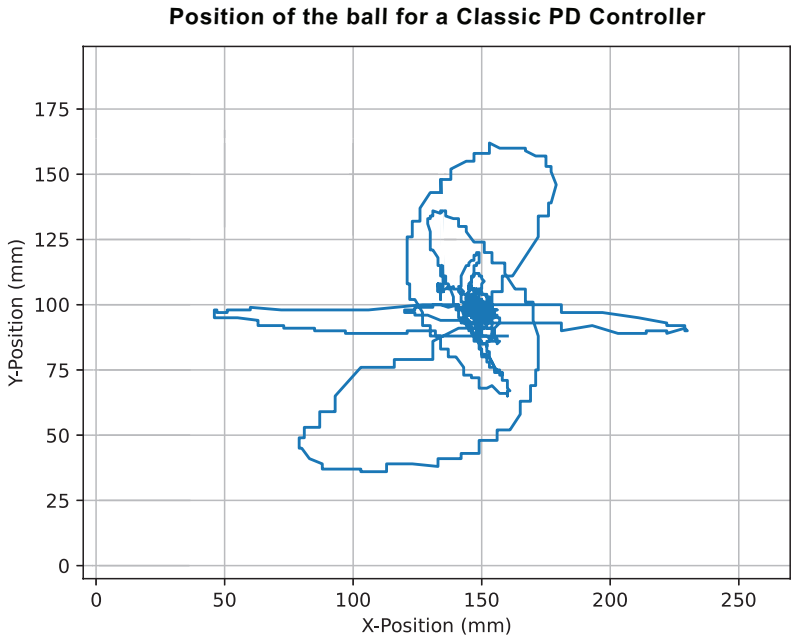


Figure 4.7: Position of the ball for a Classic PD Controller.

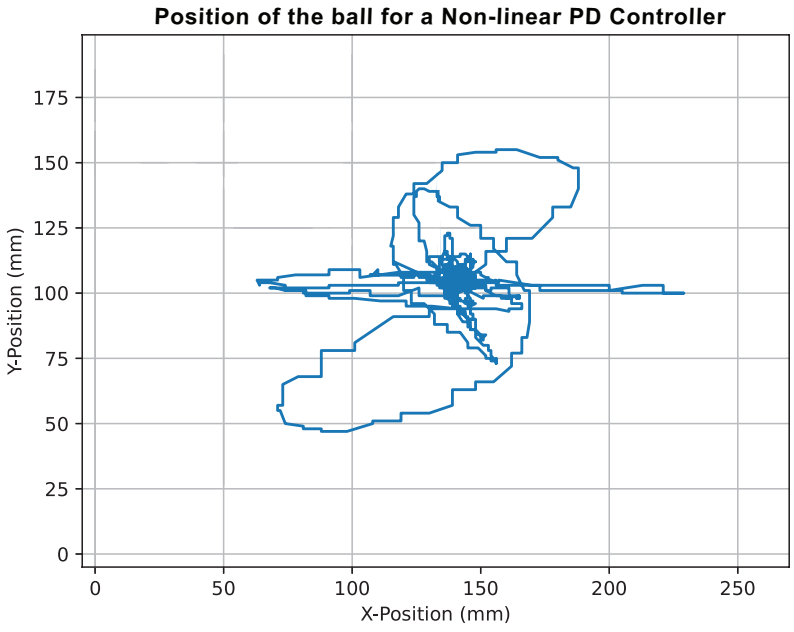


Figure 4.8: Position of the ball for a Non-linear PD Controller.

Analyzing both Fig. 4.7 and Fig. 4.8, we notice that both graphs present similar figures in terms of the trajectory of the ball, which allows us to identify which controller had a better performance in the system.

It is shown that the nonlinear PD Controller reacts in a better way to the perturbations since as soon as it feels the movement of the ball is observed, an immediate reaction of the system;

this is shown more clearly in the Y-Position in the value of 150mm for both graphs. For the nonlinear PD, the ball in its trajectory does not move so far away from the 150mm, which shows that the system reacts avoiding that the ball goes farther, while for the Classic PD, the trajectory of the ball is farther away from the 150mm, showing that the system takes a little more time to notice the change in the system and to correct it. Also measuring the time it takes to stabilize when moved to the right side for the classical PD it takes 7.47 seconds to stabilize, while for a nonlinear PD it takes 6.49 seconds.

Similarly, analyzing the entire trajectory of the ball in each controller, the nonlinear PD is still the best for this system, comparing only both controllers. This also shows that the new Arduino Uno board works correctly in the system.

# CHAPTER 5

## CONCLUSIONS

This project involved reverse engineering of an Arduino regulator for a ball-and-plate system. The primary aim was to develop a flexible and editable Arduino regulator that could facilitate modifications. To achieve this, all obtained documents, including Arduino code and schematics, were thoroughly analyzed to comprehensively understand the original system's operation.

The newly designed regulator facilitated the implementation of two control systems: a classical proportional derivative (PD) controller and a more effective nonlinear PD controller. The nonlinear PD controller shows in performance a superior responsiveness through experimentation, enhancing system efficiency when the ball move from the desired position.

Ultimately, the versatility of the new Arduino regulator enables testing with various control strategies, potentially optimizing the Ball-and-Plate System in the future. This project not only made the Arduino board more accessible for editing and further studies but also validated its correct operation by successfully implementing two controllers, showcasing improved system performance through the incorporation of a nonlinear PD controller.

# BIBLIOGRAPHY

Al-Haddad, F. (2020). Control of a ball and plate system using model-based controllers.

Awtar, S., Bernard, C., Boklund, N., Master, A., Ueda, D., and Craig, K. (2002). Mechatronic design of a ball-on-plate balancing system. *Mechatronics*, 12(2):217–228.

Bolívar-Vincenty, C. G. and Beauchamp-Báez, G. (2014). Modelling the ball-and-beam system: From newtonian mechanics and from lagrange methods. In *Twelfth LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2014)*.

Castro-Mendoza, J. C., Cabrera-Velázquez, C. J., Ríos-Montiel, E., Pérez-Silva, D., and Villafuerte-Segura, R. (2022). Sistema bola–viga: construcción y aplicación de técnicas de control. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 10(Especial 6):107–116.

HansaRobot Store (n.d.). Ball and board control system, pid resistive display for arduino stm32, open source. `https://n9.cl/ballandboardcontrolsystem`.

Knuplez, A., Chowdhury, A., and Svečko, R. (2004). Modeling and control design for the ball and plate system. pages 1064–1067 Vol.2.

Kumar, R. K., Thilagaraj, M., Arulkumar, P., Balraj, S., Pradeepa, G., Srinidhi, S., and Vivitha, V. (2021). Library management system using arduino. *Annals of the Romanian Society for Cell Biology*, pages 4425–4432.

Manzo, J. M. S. (2018). Diseño y construcción de un sistema mecatrónico de bola y plato (ball and plate). Tech. rep., Instituto Tecnológico De Tuxtla Gutiérrez.

Mijares, J. S., Villalobos, F. C., Molina, M., and Mendoza, K. (2020). Sistema de control difuso ballplate mediante pdi. 1:81–87.

Mohajerin, N., Menhaj, M. B., and Doustmohammadi, A. (2010). A reinforcement learning fuzzy controller for the ball and plate system. In *International Conference on Fuzzy Systems*, pages 1–8.

Morales, L., Gordón, M., Camacho, O., Rosales, A., and Pozo, D. (2017). A comparative analysis among different controllers applied to the experimental ball and plate system. In *2017 International Conference on Information Systems and Computer Science (INCISCOS)*, pages 108–114.

MsTimer2-Arduino (2024). *MsTimer2-Arduino Reference*. Arduino.

Naylamp Mechatronics (n.d.). Tutorial uso de servomotores con arduino. `https://naylampmechatronics.com/blog/33_tutorial-uso-de-servomotores-con-arduino.html`.

Nuñez, D., Acosta, G., and Jimenez, J. (2020). Control of a ball-and-plate system using a state-feedback controller. *Ingeniare. Revista chilena de ingeniería*, 28:6–15.

Ogata, K. (2010). *Ingeniería de control moderna*. 5ta edition.

Rekoff, M. G. (1985). On reverse engineering. *IEEE Transactions on Systems, Man, and Cybernetics*, 2:244–252.

Servo-Arduino (2024). *Servo-Arduino Reference*. Arduino.

Song, Y.-D. (2018). *Control of Nonlinear Systems via PI, PD and PID: Stability and Performance*. CRC Press, 1st edition. [Online]. Available: https://doi.org/10.1201/9780429455070.

# CHAPTER 6

# APPENDIX

```
#include <Servo.h>
#include <MsTimer2.h> //interrupcion cronometrada
//////////pin de control de pantalla tactil/////////////////
#define YL    2   //2
#define YH    15 //15
#define XL    3 //3
#define XH    14 //14
#define KEY 19 //Alfiler de boton
Servo myservo1,myservo2; //crear 2 objetos de control de servo
int Flag_Stop=1,Flag_Show,Flag_Move; //banderas relacionadas
float SetpointX=75,SetpointY=175,Ang_X,Ang_Y; //Valor objetivo y
   cantidad de control en las direcciones X e Y
int Position_X,Position_Y, Vx, Vy; //Mediciones en las direcciones X
    e Y
float PS2_KEY; //Variante clave de PS2
float Balance_Kp=70,Balance_Kd=40;//parametros PID
int PS2_LX=128,PS2_LY=128,PS2_RX=128,PS2_RY=128; //Relacionado con
   el control remoto de PS2
int error = 0; //Una variable utilizada por PS2 para identificar si
   PS2 est enchufada
void (* resetFunc) (void) = 0;// Reset func
/**********************************************************
Funcin funcin: movimiento de trayectoria especfica
```

```
Parmetros de entrada: Ninguno
Valor devuelto: Ninguno
*******************************************************/
void Setting_Move(void)
{
    static float count; //variable de conteo
    count++; //auto-incremento
   if(Flag_Move==1) //controla el movimiento de la pelota a lo largo
       de la trayectoria del tringulo
   {
          if(count<40) SetpointY++;
      else if(count<80) SetpointY-=2,SetpointX-=1.5;
      else if(count<120)SetpointX+=3;
      else if(count<160)SetpointY+=1,SetpointX-=1.5,Flag_Move=0,
         count=0;//El ltimo paso, volver al origen y poner a cero el
          contador
   }
   else if(Flag_Move==2) //Controlar el movimiento de la pelota a lo
       largo de la trayectoria de la pelota
   {
          if(count<40) SetpointY++;
      //else if(count<40+PI*40)SetpointY=SetpointY+40*cos((count-40)
         /20),SetpointX=SetpointX+40*sin((count-40)/20);
      else if(count<210)SetpointY--,Flag_Move=0,count=0;//El ltimo
         paso, volver al origen y poner a cero el contador
   }
   else if(Flag_Move==4) //Controlar el movimiento de la pelota a lo
       largo de la trayectoria de la horquilla (X)
   {
          if(count<40) SetpointY++,SetpointX--;
      else if(count<120)SetpointY--,SetpointX++;
```

```
      else if(count<160)SetpointY++,SetpointX--;
      else if(count<200)SetpointY++,SetpointX++;
      else if(count<280)SetpointY--,SetpointX--;
      else if(count<320)SetpointY++,SetpointX++,Flag_Move=0,count=0;
         //El ltimo paso, volver al origen y poner a cero el
         contador
   }
   else if(Flag_Move==8) //Controla la pelota para que se mueva a lo
       largo de la trayectoria del cuadrado.
   {
          if(count<40) SetpointY++;
      else if(count<80) SetpointX++;
      else if(count<160)SetpointY--;
      else if(count<240)SetpointX--;
      else if(count<320)SetpointY++;
      else if(count<360)SetpointX++;
      else if(count<400)SetpointY--,Flag_Move=0,count=0;//El ltimo
         paso, volver al origen y poner a cero el contador
   }
}
/*************************************************************

Funcin funcin: ajuste de parmetros PID
Parmetros de entrada: Ninguno
Valor devuelto: Ninguno
*************************************************************/
void Adjust(void)
{
   int X_temp,Y_temp,Threshold=100;//limite
   X_temp=PS2_RX-128; //Actualizacin de variable temporal de
      desviacin de direccin X
```

```
    Y_temp=PS2_RY-128; //Actualizacin de variable temporal de
       desviacin de direccin X
    if(PS2_KEY==16||PS2_KEY==32||PS2_KEY==64||PS2_KEY==128)//
                        4PID                 Presione cualquiera de
       los 4 botones a la izquierda para ajustar los parmetros PID
       para evitar un toque accidental
    {
    if(X_temp>Threshold) Balance_Kp++; //Aumento parametro KP
    if(X_temp<-Threshold)Balance_Kp--; //Reduccin de parmetros KP
    if(Y_temp>Threshold) Balance_Kd--; //Disminuye el parmetro KD
    if(Y_temp<-Threshold)Balance_Kd++; //Aumento del parmetro KD
    }
}
/**************************************************
Funcin funcin: programa de control del mecanismo de direccin
Parmetros de entrada: volumen de control del servo
Valor devuelto: Ninguno
**************************************************/
 void Control_servo(float ang_x,float ang_y)
{
    myservo1.write(90-ang_x); // Especifica el ngulo de direccin del
       mecanismo de direccin.
    myservo2.write(90-ang_y); // Especifica el ngulo de direccin del
       mecanismo de direccin.
}
/**************************************************
Funcin funcin: escaneo de teclas
Parmetros de entrada: Ninguno
Valor de retorno: estado del botn 0: ninguna accin 1: un solo clic 2:
    doble clic
**************************************************/
```

```
u8 click_N_Double (u8 time)
{
   static unsigned char flag_key,count_key,double_key;
   static unsigned int count_single,Forever_count;
    if(digitalRead(KEY)==0) Forever_count++; //El indicador de
       pulsacin larga no est establecido en 1
    else     Forever_count=0;
   if(digitalRead(KEY)==0&&0==flag_key) flag_key=1; //botn
      presionado
   if(0==count_key)
   {
      if(flag_key==1)
      {
        double_key++;
        count_key=1;
      }
      if(double_key==2)
      {
        double_key=0;
        count_single=0;
        return 2;//Haga doble clic para ejecutar el comando
      }
   }
   if(1==digitalRead(KEY)) flag_key=0,count_key=0;

   if(1==double_key)
   {
     count_single++;
     if(count_single>time&&Forever_count<time)
     {
     double_key=0;
```

```
      count_single=0;
      return 1;//Haga clic para ejecutar el comando
      }
      if(Forever_count>time)
      {
      double_key=0;
      count_single=0;
      }
    }
    return 0;//ninguna accin
}
/**************************************************************
Funcin funcin: control de PD de equilibrio de direccin X
Parmetro de entrada: ngulo
Valor de retorno: control vertical PWM
Autor: hogar del coche de equilibrio //cambiaron para pd no lineal
**************************************************************/
//PD NO LINEAL
int balanceX(float pos )
{
  float Differential,error;//Definir variables de diferencia y
     sesgos
  static float Last_error; //ltima desviacin
  int balance;// valor de retorno equilibrado
  error=pos-SetpointX; //===Encuentre el ngulo medio de equilibrio y
      la correlacin mecnica
  Differential=error-Last_error; //Encuentre la tasa de cambio de la
      desviacin.
  balance=Balance_Kp*(1+0.001*abs(error))*error/500+Balance_Kd
     *(1+0.001*abs(error))*Differential/50; //=== Clculo del
     mecanismo de direccin de control de balance PWM Control PD kp
```

```
      es el coeficiente P kd es el coeficiente D
  Last_error=error; //Guardar ltima desviacin
  return balance; //valor de retorno
}

//Para PD CLASICO
/*
int balanceX(float pos )
{
  float Differential,error;//Definir variables de diferencia y
      sesgos
  static float Last_error; //ltima desviacin
  int balance;//valor de retorno equilibrado
  error=pos-SetpointX; //===Encuentre el ngulo medio de equilibrio y
       la correlacin mecnica
  Differential=error-Last_error; //Encuentre la tasa de cambio de la
       desviacin.
  Vx = Differential;
  balance=Balance_Kp*error/500+Balance_Kd*Differential/50; // ===
      Clculo del mecanismo de direccin de control de balance PWM
      Control PD kp es el coeficiente P kd es el coeficiente D
  Last_error=error; //Guardar ltima desviacin
  return balance; //valor de retorno
}*/
/**************************************************
Funcin funcin: control de PD de equilibrio de direccin Y
Parmetro de entrada: ngulo
Valor de retorno: control vertical PWM
Autor: hogar del coche de equilibrio
**************************************************/
//PD NO LINEAL
```

```
int balanceY(float pos )
{
  float Differential,error;//Definir variables de diferencia y
     sesgos
  static float Last_error; //ltima desviacin
  int balance;//valor de retorno equilibrado
  error=pos-SetpointY; // ===Encuentre el ngulo medio de equilibrio
     y la correlacin mecnica
  Differential=error-Last_error; //Encuentre la tasa de cambio de la
      desviacin.
  balance=Balance_Kp*(1+0.001*abs(error))*error/500+Balance_Kd
     *(1+0.001*abs(error))*Differential/50; //=== Clculo del
     mecanismo de direccin de control de balance PWM Control PD kp
     es el coeficiente P kd es el coeficiente D
  Last_error=error; //Guardar ltima desviacin
  return balance; //valor de retorno
}
//PD LINEAL
/*
int balanceY(float pos )
{
  float Differential,error;//Definir variables de diferencia y
     sesgos
  static float Last_error;//ltima desviacin
  int balance;//valor de retorno equilibrado
  error=pos-SetpointY; //===Encuentre el ngulo medio de equilibrio y
      la correlacin mecnica
  Differential=error-Last_error; //Encuentre la tasa de cambio de la
      desviacin.
  Vy = Differential;
  balance=Balance_Kp*error/500+Balance_Kd*Differential/50; //===
```

```
      Clculo del mecanismo de direccin de control de balance PWM
      Control PD kp es el coeficiente P kd es el coeficiente D
   Last_error=error; //Guardar ltima desviacin
   return balance;//valor de retorno
}*/
/**************************************************
Funcin funcin: cdigo central de la funcin de control de 5 ms Autor:
   balance car home
Parmetros de entrada: Ninguno
Valor devuelto: Ninguno
**************************************************/
void control(){
   char Key;  //variable clave
   static u8 Flag_Target,Max_Ang=30;
   sei();     //interrupcin global habilitada
   Key=click_N_Double(50);//Escanear cambios clave
   if(Key==1)   Flag_Stop=!Flag_Stop; //Haga clic para controlar el
      estado del servo
   else if(Key==2) Flag_Show=!Flag_Show; // Haga clic para controlar
       el estado de la computadora superior
   if(++Flag_Target>4) Flag_Target=0; //Procesamiento por divisin de
       frecuencia
    if(Flag_Target==1) //Los primeros 10 ms recopilan datos del eje
       Y
   {
     digitalWrite(YL, LOW); //Dar la direccin X + voltaje de 3.3V
     digitalWrite(YH, HIGH);
     digitalWrite(XL, HIGH);
     digitalWrite(XH, LOW);
     Position_Y=analogRead(3)/5; //Medir las coordenadas en la
        direccin Y
```

```
  Vy= analogRead(3);
  digitalWrite(YL, HIGH);
  digitalWrite(YH, LOW);
  digitalWrite(XL, LOW);
  digitalWrite(XH, HIGH);
}
  else if(Flag_Target==2) //Los segundos 10 ms reciben el control
      remoto de PS2 y ajustan los parmetros PID
{
   if(PS2_KEY==4096||PS2_KEY==8192||PS2_KEY==16384||PS2_KEY
      ==32768)Flag_Move=PS2_KEY/4096; //Juzgar si se cumplen las
      condiciones de ejecucin de la accin establecidas
   //if(Flag_Move==0)Get_RC(); //Direccin remota de PS2
   else Setting_Move();   //establecer accin
   Adjust(); //Ajuste de parmetros PID
}
 else if(Flag_Target==3) //Los terceros 10 ms recopilan datos del
     eje X
{
  digitalWrite(YL, HIGH); //Dar la direccin Y voltaje de +3.3V //
      D3
  digitalWrite(YH, LOW); // A0
  digitalWrite(XL, LOW); // D2
  digitalWrite(XH, HIGH); // A1
  Position_X= analogRead(2)*4/15; //Medicin de la direccin X
  Vx=analogRead(2);
  digitalWrite(YL, LOW); //
  digitalWrite(YH, HIGH); // A0
  digitalWrite(XL, HIGH); // D2
  digitalWrite(XH, LOW); // A1
 }
```

```
    else if(Flag_Target==4) // El cuarto control PID de 10ms
   {
    Ang_X=-balanceX(Position_X); //Controlador PID en direccin X
    Ang_Y=-balanceY(Position_Y); //Controlador PID en direccin Y
    if(Ang_X<-Max_Ang) Ang_X=-Max_Ang; //El ngulo de control mximo
       del mecanismo de direccin en la direccin X
    if(Ang_X>Max_Ang) Ang_X=Max_Ang; //El ngulo de control mximo del
        mecanismo de direccin en la direccin X
    if(Ang_Y<-Max_Ang) Ang_Y=-Max_Ang; //El ngulo de control mximo
       del servo en la direccin Y
    if(Ang_Y>Max_Ang) Ang_Y=Max_Ang; //El ngulo de control mximo del
        servo en la direccin Y
    if(Flag_Stop==0)Control_servo(Ang_X,Ang_Y); //No hay anormalidad
       , controle el mecanismo de direccin.
   }
 }
/**************************************************************
Funcin funcin: la inicializacin es equivalente a la funcin principal
    en STM32 Autor: inicio del coche de equilibrio
Parmetros de entrada: Ninguno
Valor devuelto: Ninguno
**************************************************************/
void setup() {

  Serial.begin(9600);    //abrir puerto serie CAMBIADO
  delay(200);             //Retraso en espera de que se complete la
     inicializacin
   pinMode(XL, OUTPUT);   //pasador de control de motor
   pinMode(XH, OUTPUT);   //pasador de control de motores,
   pinMode(YL, OUTPUT);   //Pasador de control de velocidad del motor
   pinMode(YH, OUTPUT);   //Pasador de control de velocidad del motor
```

```
  myservo1.attach(9);    // Inicializar cada servo
  myservo2.attach(10);    // Inicializar cada servo
  MsTimer2::set(10, control); //Use Timer2 para configurar la
     interrupcin temporizada
  MsTimer2::start();     //habilitar interrupcin
}
/*********************************************************
Funcin funcin: cuerpo del programa de bucle principal
Parmetros de entrada: Ninguno
Valor devuelto: Ninguno
*********************************************************/
void loop() {
  //Serial.print("/*");
  Serial.print(Position_X); // prints a label
  Serial.print(",");
  Serial.print(Position_Y); // prints a label
  Serial.print(",");
  Serial.print(SetpointX); // prints a label
  Serial.print(",");
  Serial.print(SetpointY); // prints a label
  Serial.print(",");
  Serial.print(Vx); // prints a label
  Serial.print(",");
  Serial.println(Vy); // prints a label
  //Serial.print("*/");

  delay(10);
  //Serial.println();
}
```

Listing 6.1: Arduino code for ball and plate system.