Princess Sumaya University for Technology
King Abdullah II Faculty of Engineering
Electrical Engineering Department

Princess Sumaya University for Technology
جامعة الأميرة سميّة للتكنولوجيا

# Embedded system lab Project:

# PID BALL BALANCING SYSTEM

*Authors:*

| | |
|---|---|
| Mariam Mourad | 20210798 |
| Omar Tubeileh | 20200322 |
| Yazan Alsharif | 20200970 |

*Supervisor:*

Eng. Raghad Al-Harasis

*Jan 1st , 2025*

## Abstract

The purpose of the PID Ball Balancing System is to maintain a ball in the middle of the platform and ensure that it stays there by continuously detecting the ball's location and modifying the motor's speed accordingly. The PID controller uses data from the ultrasonic sensor to figure out how far away the ball is from the center and adjust the angle accordingly. The PID controller has three parts - Proportional, Integral, and Derivative. Using all three parts helps the system react well to changes in position.


## Contents

## Introduction

Using a PID Ball Balancing System is a way to try and center a ball on a platform. It greatly helps in controlling the ball's movements. This new system builds a machine that can balance a ball using various components and tools. We will be using an Arduino Uno microcontroller, ultrasonic sensor, servo motor to help with the balancing, and a joystick to have control over the PID-controller.

## Design PID Ball Balancing System

### Software Design

Simulink is used to create the software part:

- Creating a representation of a system to better understand and analyze its behavior, structures, and processes.
- Build a model using Simulink's block diagram. Choose the right blocks and connect them in a way that matches the system's behavior.
- Add the ultrasonic sensor and servo motor to the Simulink design, Use the right Simulink building blocks to show how these parts work with the control system

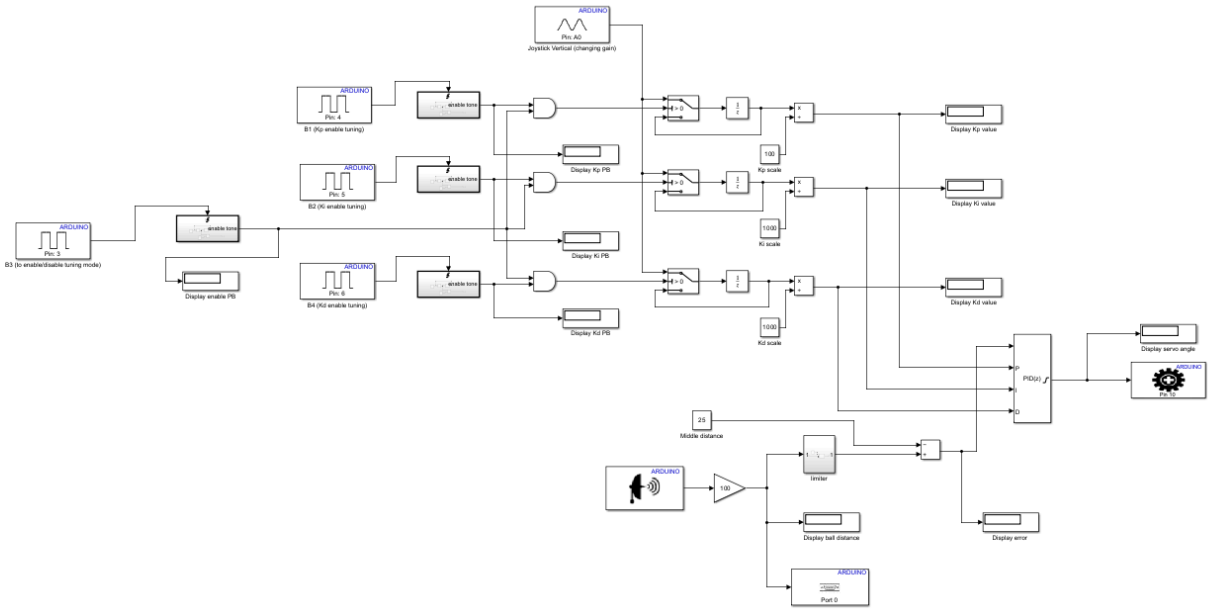

**Figure 1**: The Simulink Diagram


**Figure 2**: serial communication

## Mechanical and Electrical Components

- **Piece of Wood:**
  - The base of the system is a wooden plat; to support the system and stabilize it while it's trying to balance the ball.
- **Foam board:**
  - Two platforms connected at 90 degrees allowing the ball to move smoothly, and to give the ultrasonic a clear view of the track.
- **A ball:**
  - Any foam lightweight ball should work as long as it fits in the track
- **Servo motor**:
  - Servo motor is used to control the movement as the system is trying to reach balance; adjusting the track as the ball moves, based on the PWM signal received from the arduino and ultrasonic readings.


**Figure 3**: The Servo motor

- **Ultrasonic sensor:**
  - Used to measure the distance of the ball by sending waves (trigger signals) and then measuring the time it takes for it to come back (received echo signals). With the time travel of the trigger wave, it calculates the distance of the ball and its position related to the track (knowing the length of the track in advance)

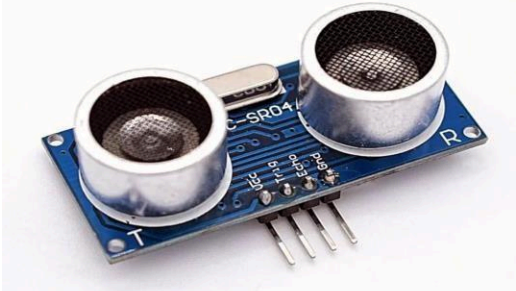


**Figure 4:** The Ultrasonic sensor

- **Joystick shield :**
  - Using the joystick, we were able to tune and monitor the values of P-I-D until balance is reached.
  - The enable button must be pressed first to allow the other buttons to function. Three other buttons are left for Kp, Ki and Kd. if any is pressed and using the vertical axis of the stick the gain for the corresponding PID-value will be changing

**Figure 5**: Joystick shield

- **Arduino uno:**
  - Arduino is an open-source platform, which will be programmed using Simulink. It has an 8-bit AVR architecture which serves as its central component. It requires a 16 MHz clock speed to run.

  - The board has a set of digital I/O pins, which include PWM, Analog pins and Digital pins.

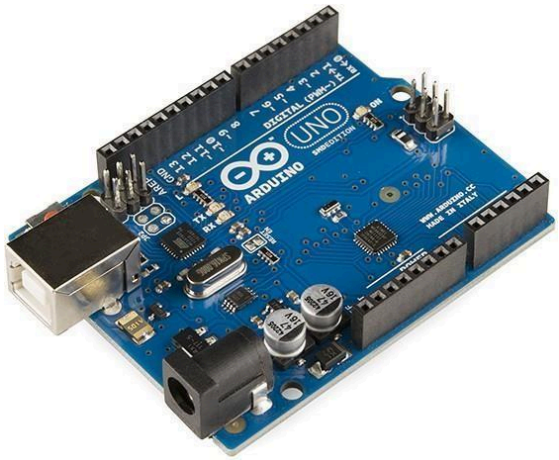


**Figure 6:** The Arduino Uno

- **Breadboard:**
  - Breadboard is used to connect the whole system together; Joystick, Servo Motor and Ultrasonic sensor are all connected together to the Arduino via the breadboard.

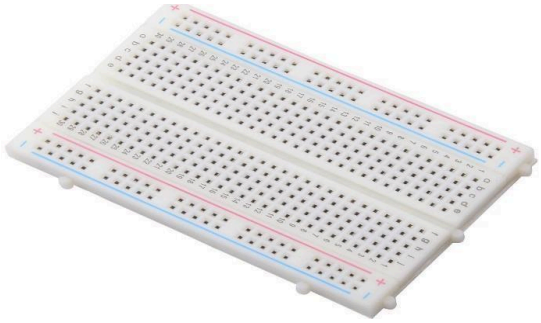

**Figure 7:** Breadboard

## PID Ball Balance System ( Proportional – Integral – Derivative)

A PID controller continuously monitors the error between the desired (set point) and the feedback (Actual) values of the controlled parameter (Distance, for example. Using this value (error), the PID controller calculates the proportional, integral, and derivative values. The controller then adds these three values together to create the output. [2]

### Proportional:

The proportional term is determined by multiplying the P-Gain with the error. Its main purpose is to provide a strong and immediate response to adjust the output, bringing the process value closer to the set point. As the error decreases, the proportional term's effect on the output also diminishes.

**P = kP x Err**

### Integral:

The integral term is calculated by multiplying the I-Gain with the error, then further multiplying it by the controller's cycle time (the interval at which the PID calculation is performed). This value is continuously summed to form the "total integral." The integral term accounts for the system's past behavior. Including an integral term helps correct the steady-state error that may occur when using only the proportional term.

**I = kI x Err x**
**dtIt = It + I**

### Derivative:

The derivative term is calculated by multiplying the D-Gain with the rate of change (ramp rate) of the process value. Its purpose is to "predict" the future behavior of the process value and adjust the output in the opposite direction of the proportional and integral terms. This helps prevent the controller from overshooting the set point when the process value changes too quickly. Simply put, if the process value approaches the set point too rapidly, the derivative reduces the output to slow it down and maintain control

**D = kD x (Err – pErr) / dt**

### Output:

The PID controller output is calculated by simply adding the Proportional, the Integral and the Derivative. Depending on the gain setting of these three values, will determine how much effect they will have on the output. Figure 4. Illustrates the gains' contribution to the system's response**.**

**PID Controller Output Math: P + It + D**

## Prototype:

### Testing the components:

- To ensure that all of the parts function flawlessly, we test the joystick, servo, and ultrasonic sensor independently.

- We computed the PID equation and made some necessary adjustments, initially by connecting constant blocks to the PID-module input; from there we started tuning the gain values before reconnecting the joystick inputs back to the module.

- The primary goal is to have the ball in the designated spot and to react quickly to move it to the needed location.

- The servo motor was moving so quickly and unsteadily when we put all the parts together, as we adjusted the gain values we regained control over the system

- We link the servo motor to the joystick Kp,Ki and Kd values and calibrate them to be dependable.

- We double the distance measured by the ultrasonic sensor by 100 to convert from meter to centimeter.

- The ultrasonic sensor continuously measures the position of a solid ball on a track and returns the ball to its predetermined position. It feeds back the ball's immediate position. The system shall compare this distance (ball's position) with a predetermined position to find the error.

- The error value is the value used by the PID controller to determine how to manipulate the output to bring the process value to the set point (difference between the set point and the feedback value which is the process (actual) value). **Error = Set point – Process Value.** Thus, the PID should compensate for this error immediately


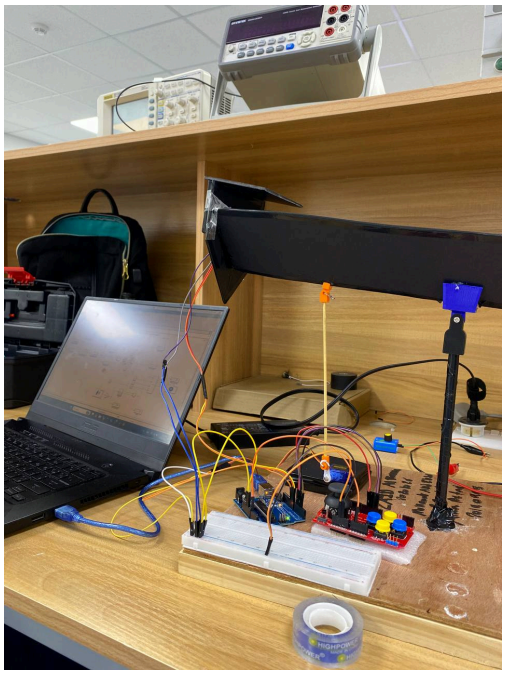

**Figure 8:** show the Prototype of PID ball Balance

## Conclusion:

The Arduino Uno functions similarly to a brain. In order to keep track of what's happening, it can link to equipment and sensors like Sharp IR sensors. The system monitors the ball tilt using the PID control algorithm, adjusting the potentiometers in response to any issues. It accomplishes this by determining the appropriate control signal by combining derivative, integral, and proportional terms. In this manner, the ball remains in the middle, where it should always be.

## References


[1] https://www.google.com/search?q=pid+controller&oq=pid&aqs=chrome.0.69i59l2j69i57j69i5 9j0i512l5j0i10i512.386j0j15&sourceid=chrome&ie=UTF-8

[2] https://www.hackster.io/nihalsuri/self-balancing-ball-and-beam-using-pid-control-2bfb86

[3] https://www.youtube.com/watch?v=JFTJ2SS4xyA

[4] https://reference.wolfram.com/language/MicrocontrollerKit/workflow/BallAndBeamControl

[5] https://www.instructables.com/Self-Balancing-Robot-Using-PID-Algorithm-STM-MC/