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

|  |
| --- |
| **Embedded Lab Systems**  **PID Ball Balancing System** |

|  |  |  |  |
| --- | --- | --- | --- |
| *Authors:* |  |  | *Supervisor:* |
| Lujain Amro  Katrin Zagha  Leen Bilto | 20210738  20210858  20210310 | IoT Engineering  NIS Engineering  IoT Engineering | Eng. Abdulghani Aljundi |

*June 1, 2025*

**Abstract**

This project implements a model-based PID ball balancing system using an Arduino microcontroller and Simulink. An ultrasonic sensor measures the ball’s position on a tilting track, and a servo motor adjusts the track to maintain the ball at a predefined setpoint. A PID controller processes the error between the current and target positions, with gains manually tuned via potentiometers. The system demonstrates real-time feedback control, integrating embedded hardware and software to achieve stable and responsive performance.

Contents

[**Introduction** 4](#_TOC_250006)

**Design PID Ball Balancing System** 4

**Software Design** 4

[**Mechanical and Electrical Components** 5](#_TOC_250005)

[**PID Ball Balance System ( Proportional – Integral – Derivative)** 8](#_TOC_250004)

[**Prototype** 8](#_TOC_250003)

[**Testing the Components** 8](#_TOC_250002)

[**Conclusion** 10](#_TOC_250001)

[**References** 10](#_TOC_250000)

**Introduction**

Ball balancing systems are widely used to demonstrate principles of feedback control, particularly Proportional-Integral-Derivative (PID) control. These systems provide a practical platform for understanding dynamic system behavior, real-time sensing, and actuator response. In this project, a model-based PID controller is implemented to stabilize a ball on a track by adjusting the track’s angle using a servo motor.

The setup includes an Arduino microcontroller, an ultrasonic sensor for position feedback, and three potentiometers for manual tuning of the PID gains. The embedded software is developed using Simulink, which facilitates real-time modeling, simulation, and code generation for hardware deployment. The system continuously measures the ball's position, computes the error from a predefined setpoint, and adjusts the track accordingly to bring the ball back to equilibrium.

This project highlights the integration of hardware and model-based design to solve control problems and illustrates the effectiveness of PID control in maintaining system stability under varying conditions.

**Design PID Ball Balancing System Software Design**

The software component of the system is developed using Simulink, which enables model-based design and simulation:

* A system-level model is created to facilitate a deeper understanding of its dynamics, structure, and operation.
* Simulink's block diagram environment is used to construct the control model, selecting appropriate functional blocks and interconnecting them to replicate the behavior of the physical system.
* The ultrasonic sensor and servo motor are integrated into the model using corresponding Simulink blocks, accurately representing their interaction with the PID control loop.

A diagram of a circuit

AI-generated content may be incorrect.

**Figure 1**: The Simulink Diagram

**Mechanical and Electrical Components**

The ball balancing system combines basic mechanical materials with essential electrical components to form a functional control setup:

* **Piece of Wood:**
* A flat wooden platform serves as the base of the system. It provides structural support and stability, ensuring the entire setup remains steady while the system operates to balance the ball.
* **Wood board:**
* The system uses two wooden platforms joined at a 90-degree angle. This configuration allows the ball to roll smoothly along the track while also ensuring the ultrasonic sensor has a clear and direct line of sight to accurately measure the ball’s position.
* **A ball:**
  + A small solid ball (such as a rubber or plastic ball) is used as the object to be balanced. Its movement along the wooden track is controlled by the servo mechanism.
  + **Servo motor**:
    - Mounted beneath the track and connected via a stick, the servo motor adjusts the inclination of the track to guide the ball back to its target position based on PID output.



**Figure 3**: The Servo Motor

* **Ultrasonic sensor**
* The ultrasonic sensor measures the ball’s distance by emitting ultrasonic waves (trigger signals) and calculating the time it takes for the reflected waves (echo signals) to return. By using the time of flight of the signal and knowing the total length of the track, the system accurately determines the ball’s current position relative to the starting point.



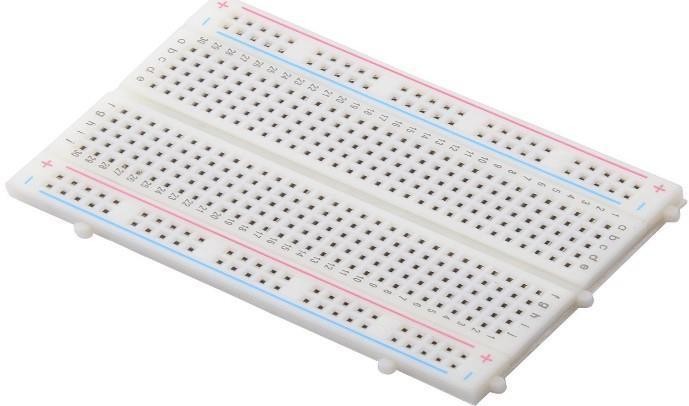
**Figure 4:** The Ultrasonic Sensor

* **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 4:** The Arduino Uno

* **Breadboard:**
* Used to prototype the electrical circuit. It allows easy connections between the Arduino, potentiometers, ultrasonic sensor, and servo motor without soldering.

**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.

**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:**

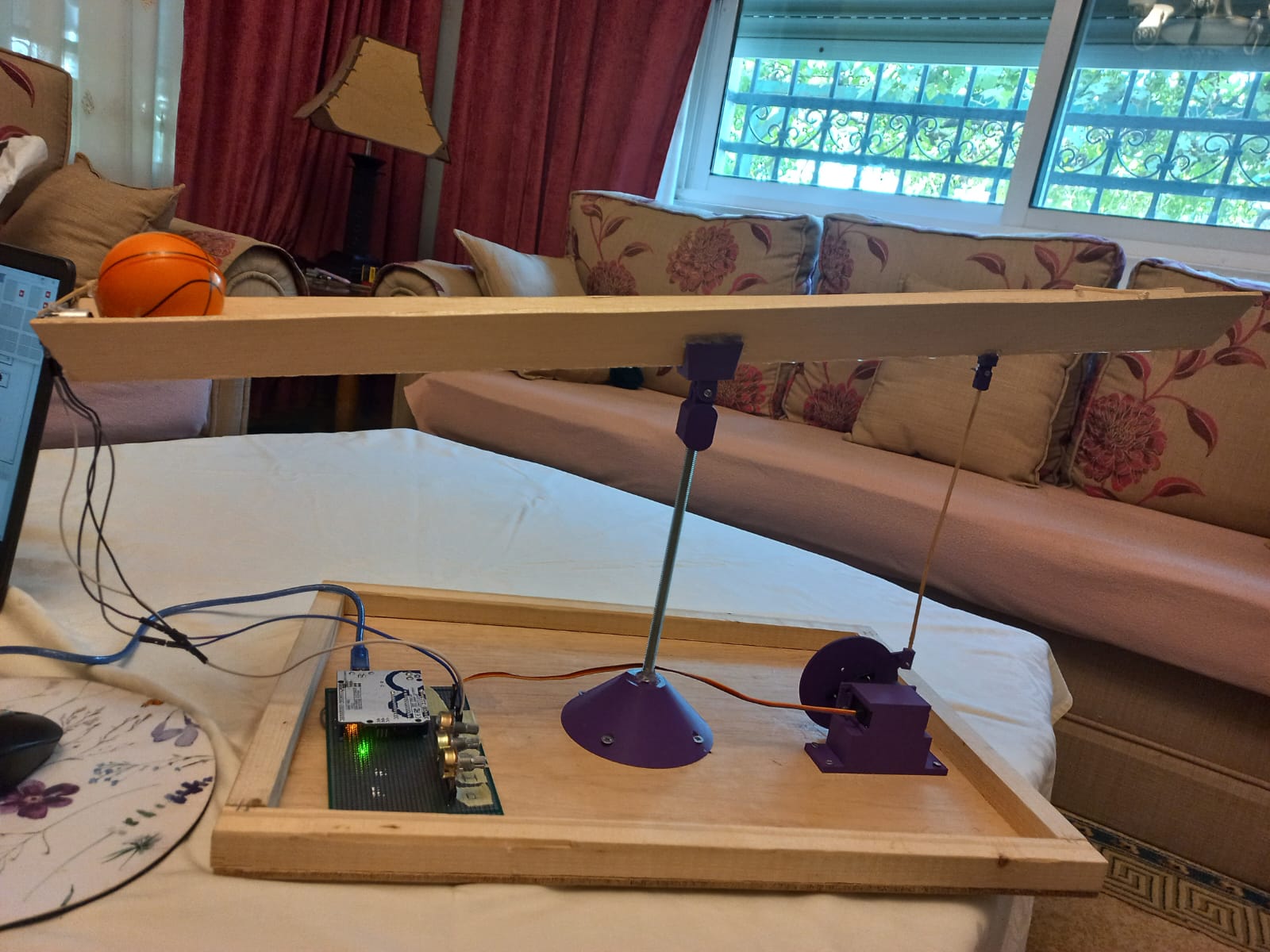
The PID controller output is calculated by simply adding the Proportional, the Integral and the Derivative. Depending on the gain setting of these three values, will determine how much effect they will have on the output. Figure 4. Illustrates the gains’ contribution to the system’s response**.**

**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:** 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.0.69i59l2j69i5](https://www.google.com/search?q=pid%2Bcontroller&oq=pid&aqs=chrome.0.69i59l2j69i57j69i59j0i512l5j0i10i512.386j0j15&sourceid=chrome&ie=UTF-8) [7j69i](https://www.google.com/search?q=pid%2Bcontroller&oq=pid&aqs=chrome.0.69i59l2j69i57j69i59j0i512l5j0i10i512.386j0j15&sourceid=chrome&ie=UTF-8)5 [9j0i512l5j0i10i512.386j0j15&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=pid%2Bcontroller&oq=pid&aqs=chrome.0.69i59l2j69i57j69i59j0i512l5j0i10i512.386j0j15&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>