Embedded Systems Lab, Fall 2025

Final Design Project

Model-Based PID Ball Balancing System

Description:

Implementing a Model-Based PID Ball Balancing System as shown in Figure 1. The system uses an Arduino board, a servo motor, an ultrasonic sensor, and a joystick module. The ultrasonic sensor continuously measures the position of a solid ball on a track and returns the ball to its predetermined position. The ultrasonic sensor feeds back the ball's immediate position. The sensor should be placed at the end of the track and directed to the ball's movement path. The servo motor is placed on a base under the track and is connected to it by a stick and flex joints to get the ability to move the track up and down until the ball returns to its position on the track. The ball's distance is fed into a proportional-integral-derivative (PID) controller, which has been programmed into the Arduino. The system shall compare this distance (ball's position) with a predetermined position to find the error. Thus, the PID should compensate for this error immediately. The Kp, Ki, and Kd are gains that should be tuned manually to operate the controller correctly. The tunning shall be done using the Joystick module shown in Figure 2. For more information, visit the link in [1].

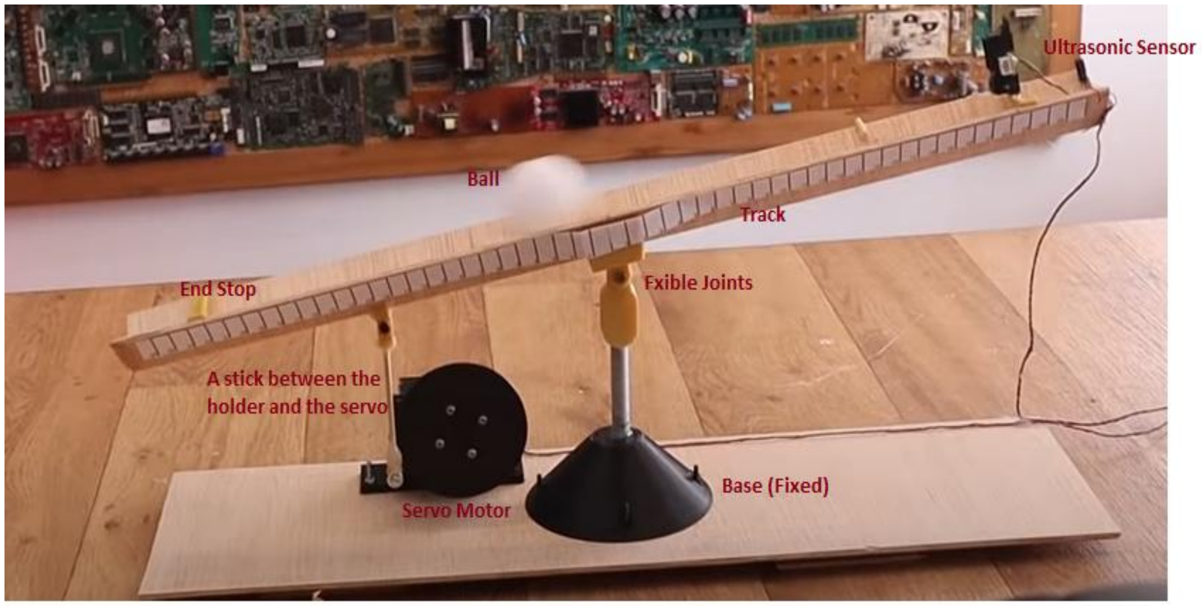
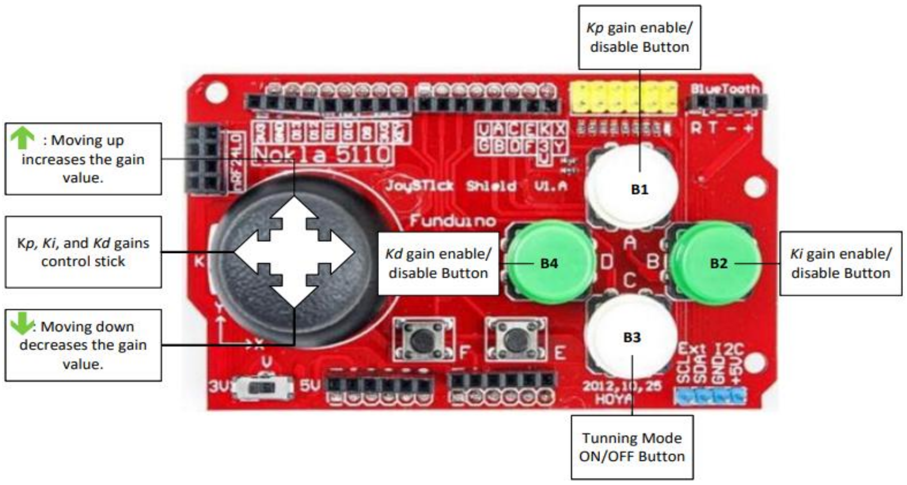


Figure 1: PID Ball Balancing System

The Joystick module is an input device that pivots on a base and conveys its angle or direction to a control system. It commonly includes push buttons that the controller can read to determine their state. The B1 button turns on or off, controlling the Kp gain, and the B2 and B4 buttons do the same for the Ki and Kd gains, respectively. The B3 button turns the tunning mode on or off. This button should be enabled first to enable the other buttons. The stick S increases or decreases the enabled gain by moving it up or down.



Kp gain enable/ disable Button
: Moving up increases the gain value.
Kp, Ki, and Kd gains control stick
: Moving down decreases the gain value.
Kd gain enable/ disable Button
Ki gain enable/ disable Button
B1
B2
B3
B4
Tunning Mode ON/OFF Button

Figure 2: A Joystick Module.

PID Controller Explained:

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]

PID Controller Terms:

A. **Set Point:** The set point is normally a user-entered value, in cruise speed control it would be the set speed, or for a heating system, it would be the set temperature. In our project, it is the set distance of the ball on the track (hardcoded position in cm by the user).
B. **Process Value:** The process value is the value that is being controlled. In cruise speed control, it would be the actual vehicle speed, or in a heating system, it would be the current temperature of the system. In ball balancing system, this would be the immediate position of the ball (read by the ultrasonic sensor).
C. **Output:** The output is the controlled value of a PID controller. In cruise control, the output would be the throttle valve, in a heating system, the output might be a

3 way valve in a heating loop, or the amount of fuel applied to a boiler. In the ball balancing system it would the angle of the servo motor that is driven by a PWM signal. (you would need to scale the PID output to the pulse width of the PWM signal controlling the servo motor)

D. **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 the process value or the actual value). Error = Set point – Process Value

E. **Gain:** Gain is the term used for "multiplication factor". By adjusting the gain settings (or multiplication factor) of the proportional, the integral and the derivative, the user can control how much effect the PID controller has on the output, and how the controller will react to different changes in the process value.

- **P or Proportional:** The Proportional is calculated by multiplying the P-Gain by the error. The purpose of the proportional, is to have a large immediate reaction on the output to bring the process value close to the set point as shown in figure 3. As the error becomes less, the influence of the proportional value on the output becomes less. The Proportional math looks like this:

$$P = kP \times Err$$

- **I or Integral**: The Integral is calculated by multiplying the I-Gain, by the error, then multiplying this by the cycle time of the controller (how often the controller performs the PID calculation) and continuously accumulating this value as the "total integral". The integral looks into the past of the system. Adding an integral command will correct the static error induced by a proportional command set alone as shown in Figure 3.

$$I = kI \times Err \times dt$$

$$It = It + I$$

- **D or Derivative:** The derivative is calculated by multiplying the D-Gain by the ramp rate of the process value. The purpose of the derivative is to "predict" where the process value is going, and bias the output in the opposite direction of the proportional and integral, to hopefully prevent the controller from over-shooting the set point if the ramp rate is too fast as shown in Figure 3. Explained a bit simpler, if the process value is approaching the set point to fast, the derivative will limit the output to prevent the process value from.

$$D = kD \times (Err - pErr) / dt$$

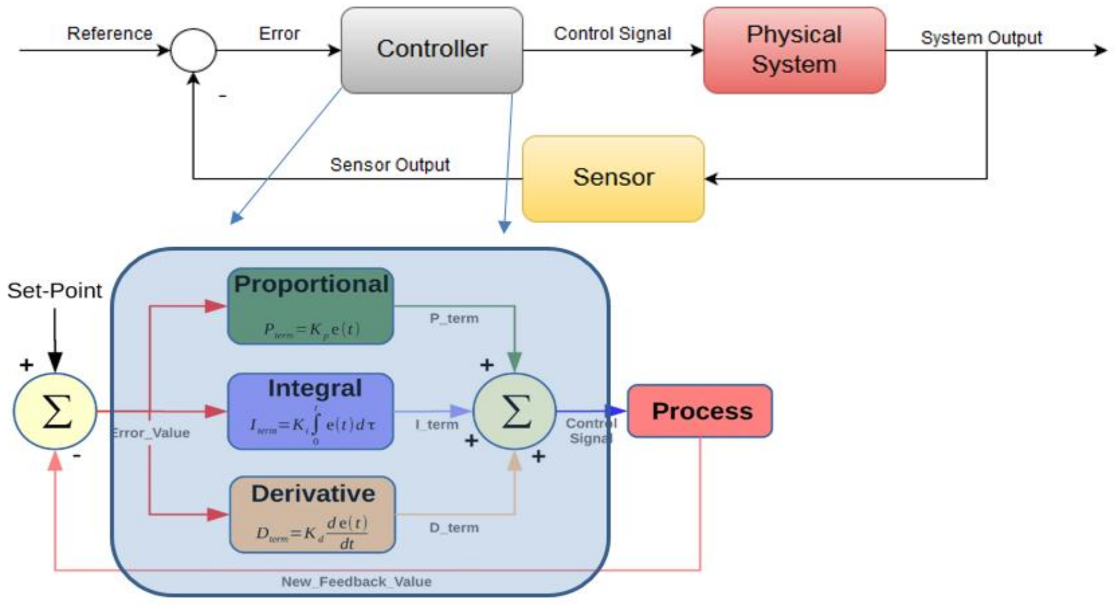


Figure 3: PID control block diagram

F. **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

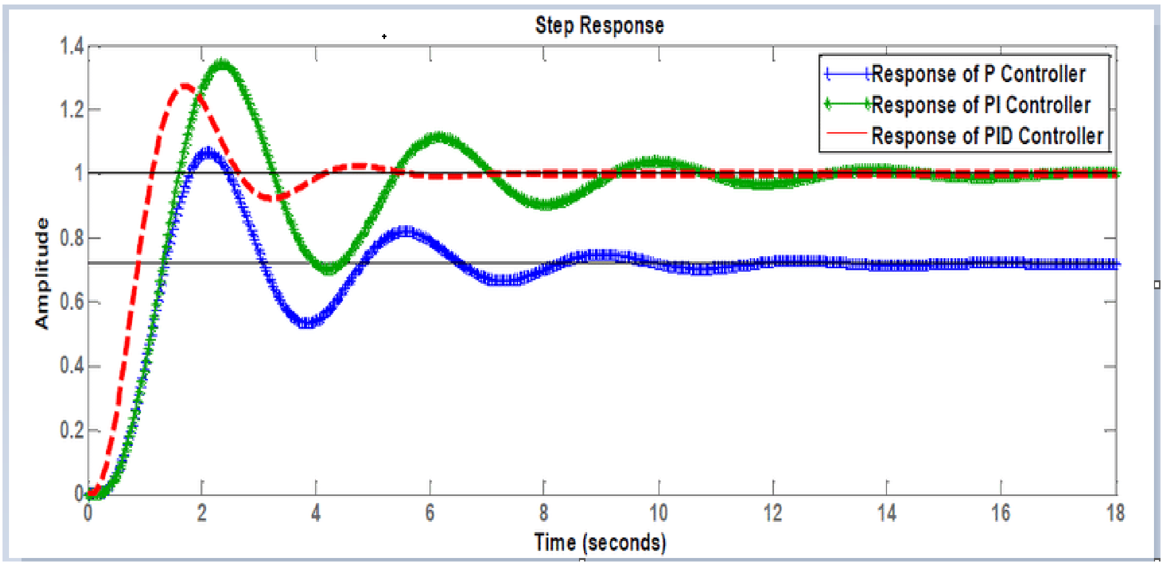


Figure 4: The Response of each term of the PID Controller on the system.

Embedded Hardware part description:

1. The system consists of a track, a flexible holder, and a base. In addition to a servo motor an ultrasonic sensor, and a Joystick module.
2. The holder of the track is flexible, can move left or right using flexible joints.
3. The ball moves through the track. The system should return the ball to its position on the track.
4. The position of the ball will be measured using the ultrasonic sensor. The sensor should be placed at the beginning of the track.
5. A servo motor will be placed on the base and holding a stick connected to the track.
6. The servo should move the track up and down until the ball returns to its position on the track.
7. At the end of the track, an end stop should be placed to protect the ball from falling.
8. You may use any type of Arduino boards.
9. Some of the required hardware parts will be printed by the 3D printer and provided.

Embedded Software part description:

1. The system should be implemented using Simulink
2. The model should measure the position of the ball continuously (i.e. immediate distance).
3. Depending on the current and the predetermined position, the error should be found and then shall be inserted into the PID controller.
4. The PID controller compensates for the error using the three factors (Proportional, Integral, and the Derivative).
5. The feedback comes from the measurements of the ultrasonic sensor.
6. The system should be deployed in a stand-alone mode.
7. The gains values and the immediate distance shall be displayed on the Simulink “Display” Block, and the system’s response should be displayed on the “Scope” block in Simulink.
8. Using any HyperTerminal program, the model should send the immediate distance serially to the PC.

Important Notes:

1. You may build your own PID controller or use the “Discrete PID Controller” block in the Simulink Libraries.
    A. In the case of using the “Discrete PID Controller” block, you need to change the block’s mode to external mode to enable manual adjustment of the P, I, and D factors, as shown in Figure 5.

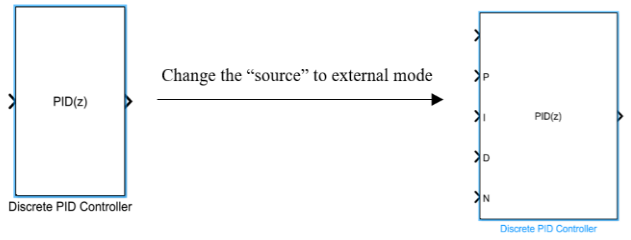


Figure 5: Discrete PID Controller block changing mode.

    B. To build your own PID, you may use the following blocks shown in Figure 6:

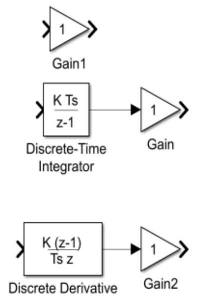


Figure 6: P, I, and D Factors Blocks in Simulink.

2. Make sure when using any block to be in the Z domain (not the S domain).
3. Tuning can be performed using either the "Ziegler and Nichols" method or through trial and error. Ultimately, the selected factors should control the system's response, which should be fast, stable, and have minimal steady-state error.

Deliverables:

1. **Oral Exam/Presentation**:
    - **Date**: Tuesday, January 7th, 2025.
    - All deliverables must be uploaded to eLearning by **midnight on January 6th, 2025.**

The project deliverables for the oral exam are as follows:

a. **PowerPoint Presentation**: A presentation that explains the design of the project.
b. **Report**: Detailed project documentation, including precise figures showing the PID control response.
c. **Short Video**: A video demonstrating the system in operation.

2. **Written Exam/Short Quiz**:
   - This will take place after the oral exam for each group.
   - The quiz will assess how well each team member understands and contributed to the project.
   - The quiz duration is at most 10 minutes.

Project Requirements:

1. **Design Requirements and Constraints:**
   a. The PID ball balancing system shall be microcontroller-based.
   b. The used microcontroller shall be any module of Arduino boards.
   c. The Embedded Software shall be Model-Based developed using Matlab Simulink.

Exclusions from Evaluation Criteria:

The following cases will **not** be evaluated as part of the project criteria and will be directed to the misconduct committee:

1. **False Claims of Contribution**: Any student who claims to have contributed to the project but is found not to have done so will be excluded from the evaluation.
2. **Plagiarism or Cheating:** Students who copy any required deliverable (e.g., code, report, presentation, video) from previous samples or use AI tools to generate these deliverables will be excluded.
3. **Failure Submission:** the final project is considered a final exam, any student or group who does not submit the project by the deadline will receive an "incomplete" mark.

References:
[1]: https://www.youtube.com/watch?v=JFTJ2SS4xyA&t=709s
[2]: PID Controller Explained • PID Explained

Project Evaluation:

Student Name:	**Student ID:**	**Quiz (6%):**	**Project (30%):**

Evaluation Policy			**100%**
Task	**Criteria**	**Grade**	**Total**
Mechanical Setup Finishing	The finishing of the physical implementation of the ball balancing system.		**10**
Electrical Design	Electrical Circuits implementation, wiring, and finishing. - Rigid hardware, loose-less wiring connection, sensors, actuators positioning and installation, etc.)		**10**
Model-Based Embedded Software Design	The Software design is plausible and well-implemented. - Input/Output Interfacing, Serial Communication, Scaling (if required), Ultrasonic readings precision, etc.		**10**
PID Tuning	How well the PID controller is tuned and doing its job:		**40**
	- Keeping the ball at the setpoint and how well and fast it responds to misplaced positions with minimal oscillation in the presence of external noise.		**10**
	- How fast it brings the ball to the required position (Response time) i.e., how well the P controller is doing		**10**

	- The amount of overshoot/undershoot when reaching the set point. i.e., How well the D controller is doing		**10**
	- Reaching the actual set point with no offset. i.e., steady-state error is controlled by the I controller.		**10**
Report	How well the report is written in terms of structure, completeness, and language.		**15**
Presentation	Oral presentation of the project to the audience		**10**
Teamwork	How did the team distribute the tasks and communicate with each other to complete the project.		**5**
		Total (100%):	
		(30%):	