ECEN 4213
Embedded Computer System Design

Lab 2: Feedback Control of a Ping-Pong Ball

Instructor: Dr. Weihua Sheng, weihua.sheng@okstate.edu

TA: Zhanjie Chen, zhanjie.chen@okstate.edu

Claudia Pauyac, cpauyac@okstate.edu

Fall 2025

I. Lab 1 Due Date and Submission

II. Lab 2 Introduction

III. Lab 2 Due Date and Submission

Due date

- Lab demonstration:
  - ✓ no later than 7: 20 pm, September 9, 2025 (Tuesday Session)
  - ✓ no later than 7: 20 pm, September 10, 2025 (Wednesday Session)
  - ✓ no later than 5: 20 pm, September 12, 2025 (Friday Session)

- Lab report:
  - ✓ no later than 11: 59 pm, September 9, 2025 (Tuesday Session)
  - ✓ no later than 11: 59 pm, September 10, 2025 (Wednesday Session)
  - ✓ no later than 11: 59 pm, September 12, 2025 (Friday Session)

Submission (in a ZIP file)

- Lab report (Word or PDF file): Follow the *ECEN4213 Lab Report Sample.docx*
- Your code: *Lab1EX3.cpp*, *Lab1EX4.cpp*, *Lab1EX5.cpp*

I. Lab 1 Due Date and Submission

II. Lab 2 Introduction

III. Lab 2 Due Date and Submission

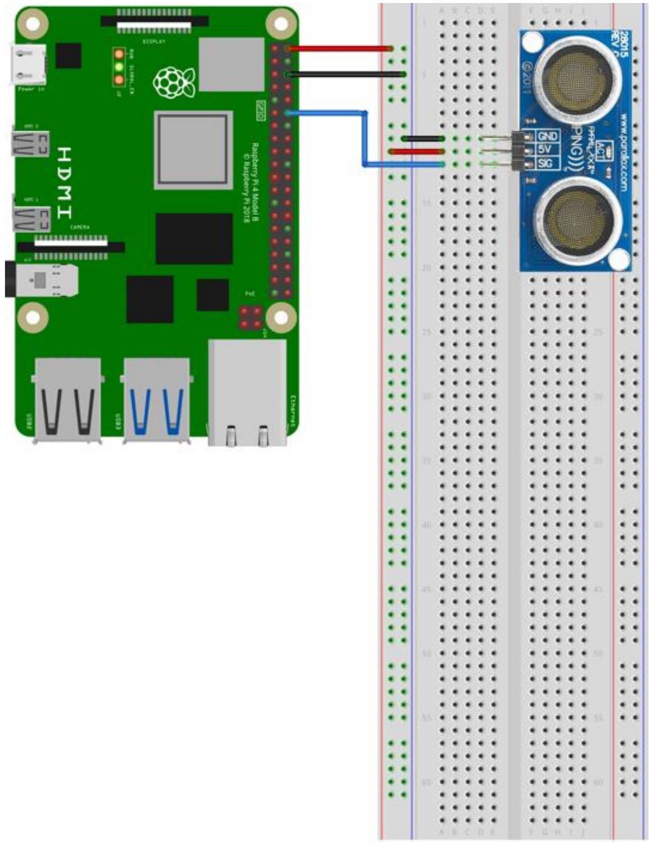
II. Lab 2 Introduction

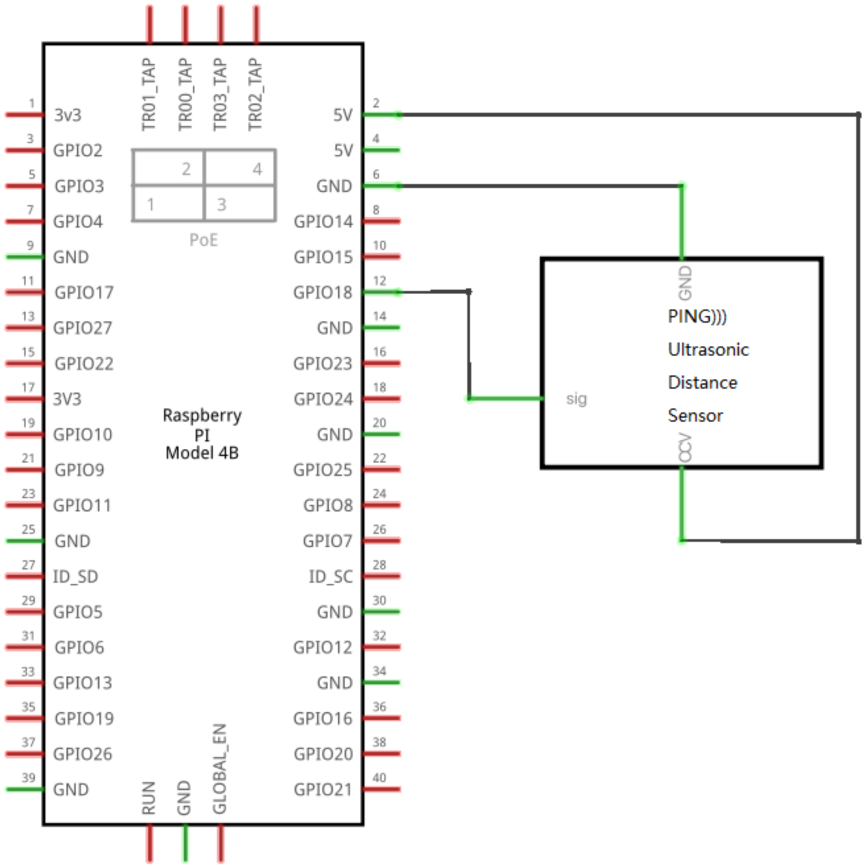
Lab 2 Objectives

- Interfacing the RPi with a sonar sensor to measure distance

- Using the PID algorithm to control the fan so that the Ping Pong ball can hover at a desired height in a tube which is indicated by a potentiometer

Exercise 1 – Measuring distance with sonar sensor



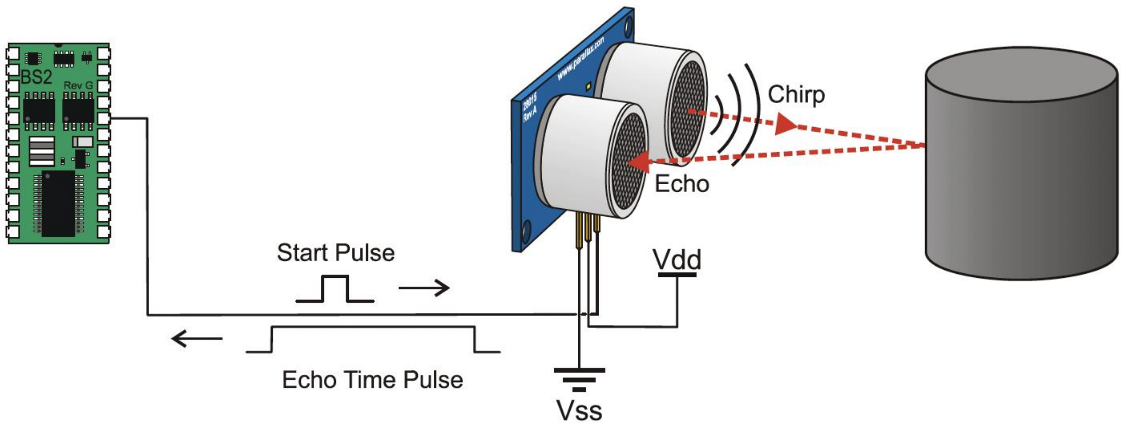


Read the distance measurement from the sensor

Operating principle of Ultrasonic sensors

- <u>Pulse emission</u>: When triggered, the sensor emits a $40kHz$ ultrasonic burst.
- <u>Echo reception</u>: If the burst hits an object, it bounces back to the sensor. The time taken to receive this echo is used to calculate distance.
- <u>Communication protocol</u>:

  1. Host (RPi) sends a trigger pulse (2-5us HIGH)
  2. Sensor emits a burst and waits.
  3. Sensor returns a pulse width corresponding to time-of-flight.
  4. Distance is computed using:



$$Distance\ (cm) = \frac{Echo\ time\ (us)}{58}$$

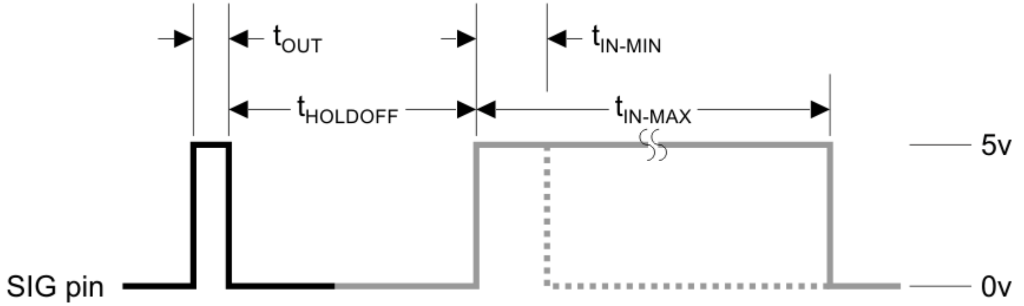
(Assuming speed of sound $\approx 340\ \frac{m}{s}$

<u>Supplemental document</u>: ***ParallaxPING-SonarSensor.pdf***

Ultrasonic sensor measurement sequence

Each cycle to measure distance includes:

1. Trigger signal:
   - Set pin as OUTPUT
   - Send LOW → wait 2us
   - Send HIGH → wait 5us
   - Send LOW again



▬ (black)	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬ (gray)	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	$t_{IN\text{-}MIN}$	115 µs
		Echo Return Pulse Maximum	$t_{IN\text{-}MAX}$	18.5 ms
		Delay before next measurement		200 µs

2. Echo holdoff:
   - Wait 750us for sensor processing

3. Read echo pulse:
   - Set pin as INPUT
   - Measure pulse duration (time the pin stays HIGH)

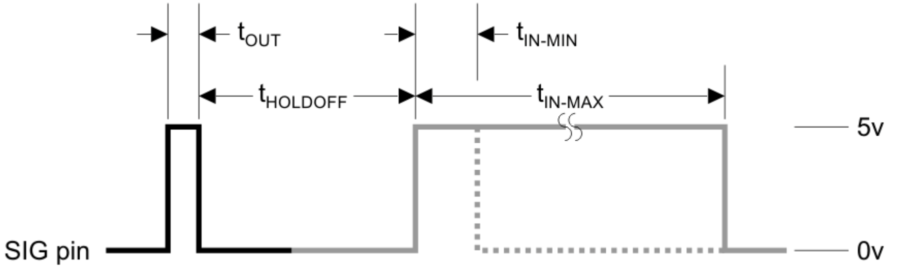
4. Compute distance:
   - Convert time to distance: $d = \frac{t_{echo}}{2} * Speed\ of\ Sound$

5. Print result and wait 200ms before next cycle

Measuring the Echo pulse duration

To accurately measure the echo return time, use high-resolution timing functions:

```
auto t1 = high_resolution_clock::now();
while (digitalRead(sig_pin)) {
   auto t2 = high_resolution_clock::now();
   auto pulse_width = chrono::duration_cast<chrono::microseconds>(t2 - t1).count();
   if (pulse_width >= MAX_ECHO_TIME)
      break;
}
```



▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

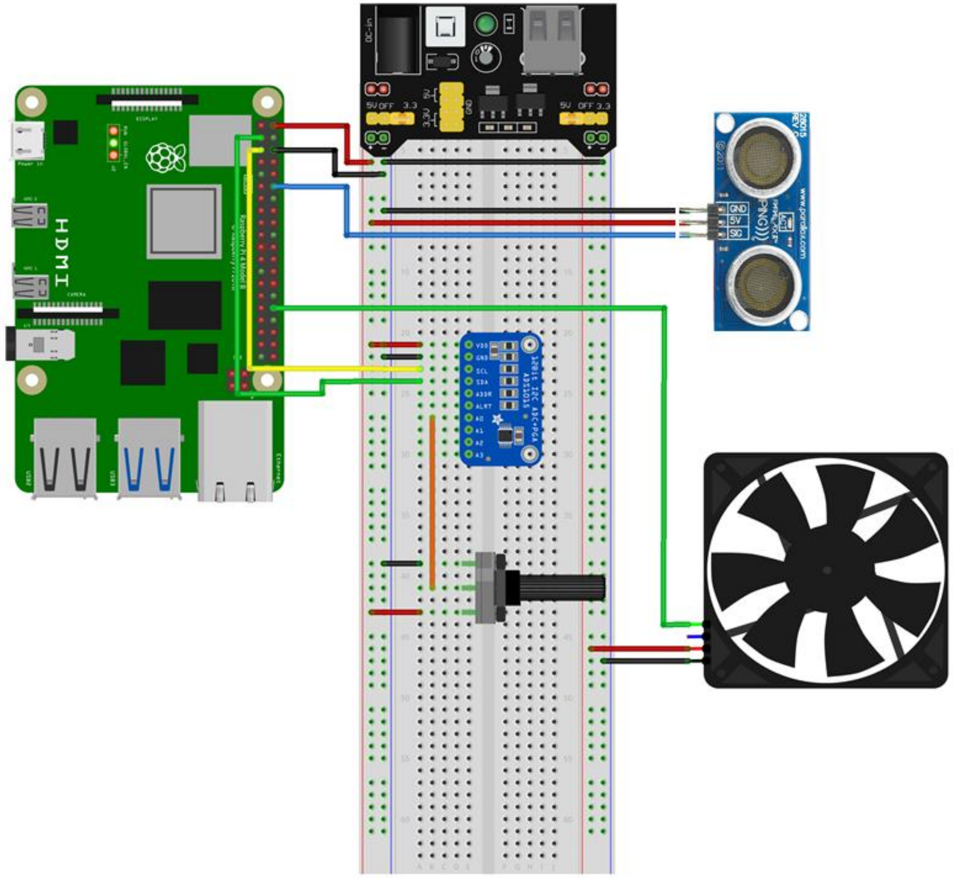
Explanation:

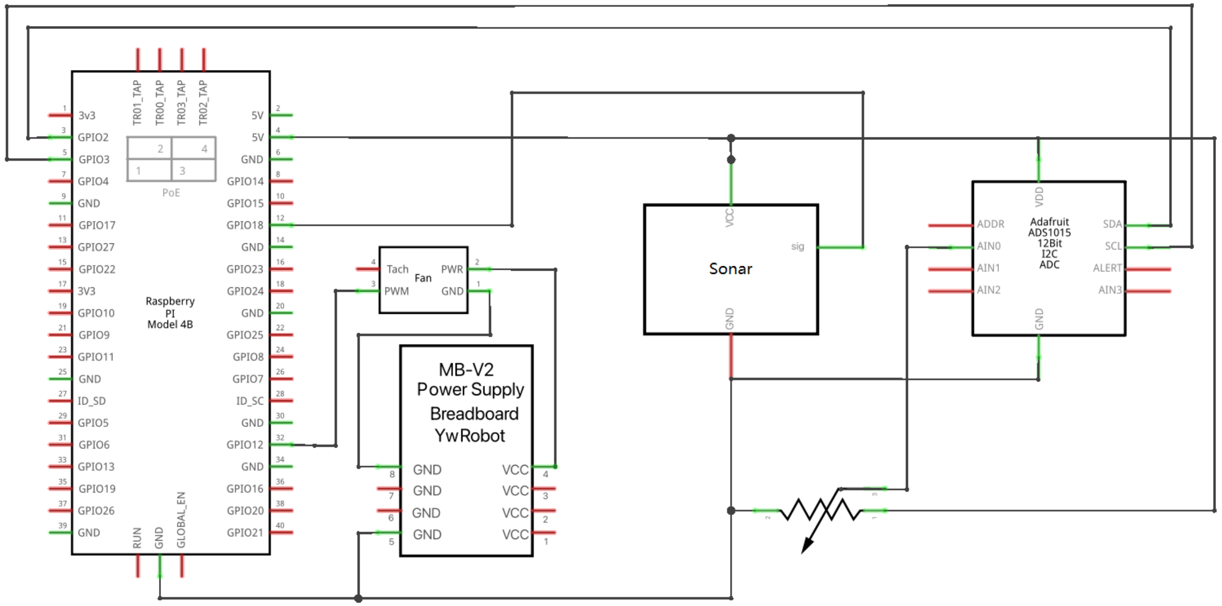
- Start timing when the echo pin goes HIGH
- Stop timing when it goes LOW
- The pulse_width gives the time taken by the burst to travel to the object and back
- Use this value to compute the distance

Note: Always consider a timeout (e.g., 18.5ms) to avoid infinite waiting when no object is detected

Supplemental document: ***High_Resolution_Clock_Reference.pdf***

Exercise 2 – Fan-Ball PID Control System





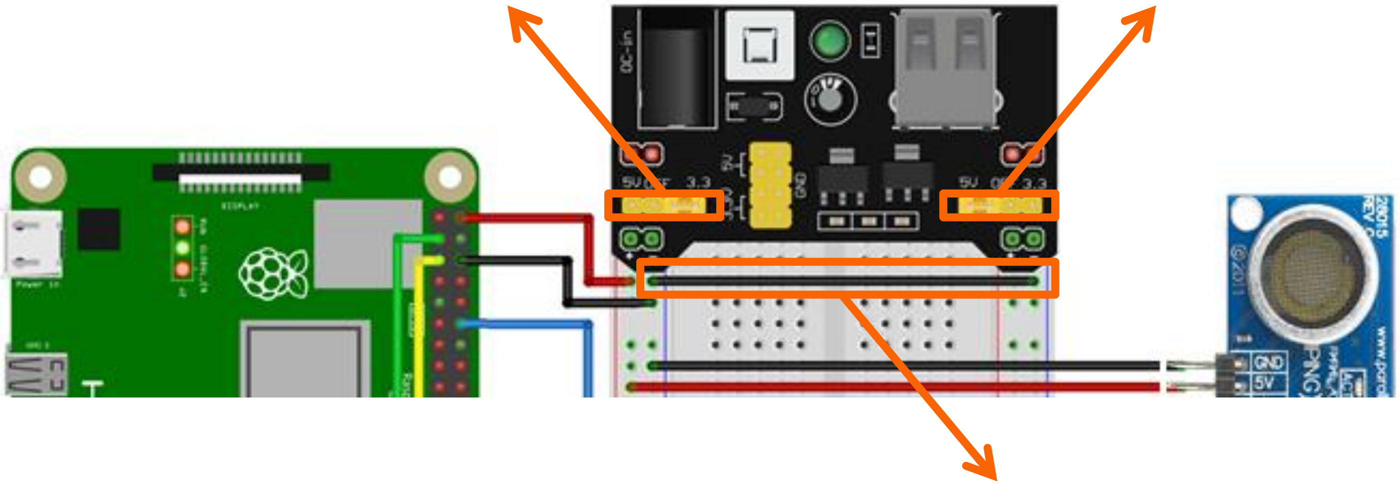
Use the Proportional Integral and Derivative (PID) algorithm to control the fan so that the Ping-Pong ball can hover at a desired height in a tube which is indicated by a potentiometer.

Setup warnings and connection adjustments

Before running your circuit:

❌ Remove the default hat connected to the breadboard power rails

✅ Move the right-side hat to align with the $5V$ power rail on the breadboard (very left side)

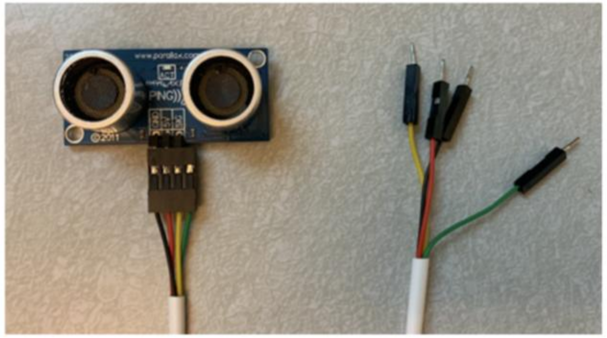


🔗 Don’t forget the connection between the ground rails to ensure all components share a common ground

<u>Note</u>: Incorrect jumper placement can prevent the fan or sensor from working properly.

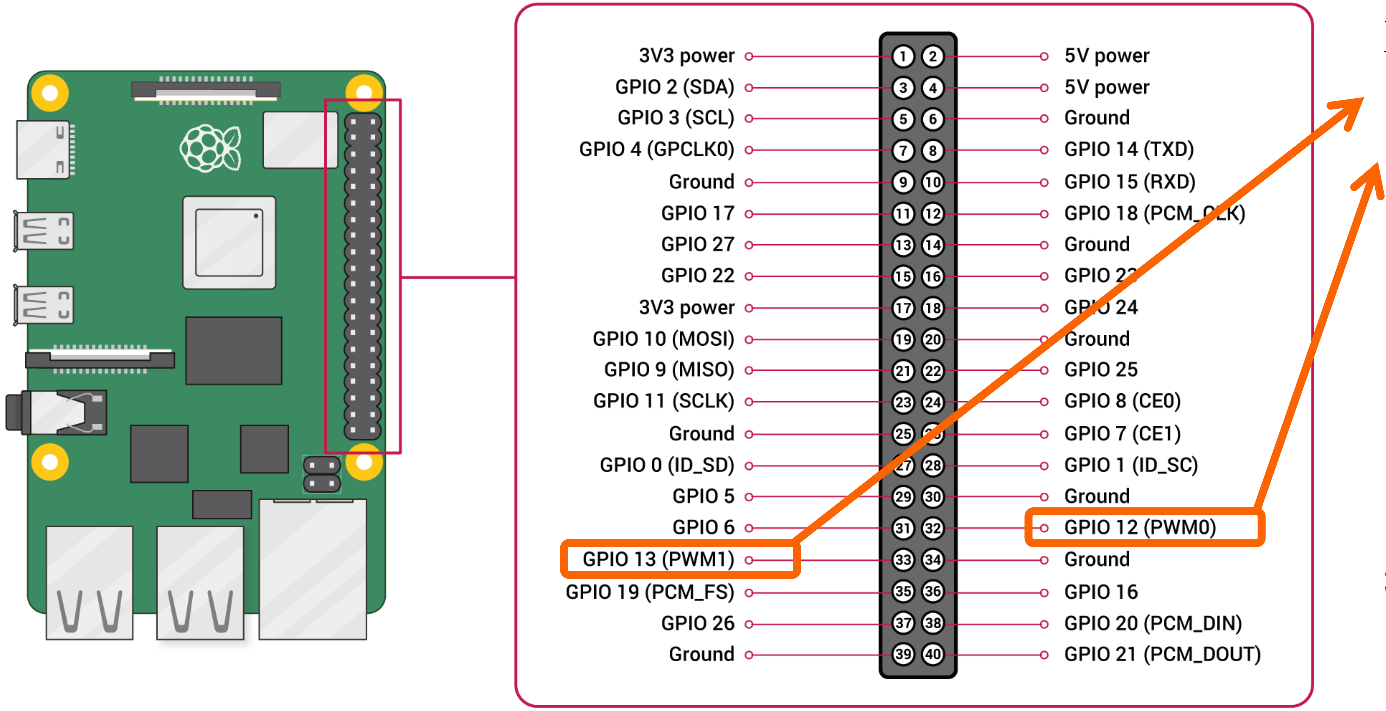
Sensor and hardware installation

- The ultrasonic sensor is mounted at the top of a vertical tube.
- The fan sits at the bottom and blows air upward.
- The ball floats in the airstream.
- Wires from the RPi run to:
    - The fan for PWM control,
    - The sonar sensor for position feedback,
    - The potentiometer for target height.



Controlling the Fan with PWM (Pulse Width Modulation)

To control the fan, we can use either Hardware PWM or Software PWM:



Hardware PWM

- Only works on specific pins (e.g., GPIO 12, 13)
- Uses *pwmWrite()* and *sudo*

```
pinMode(motor_pin, PWM_OUTPUT);
pwmWrite(motor_pin, value);
```

To run the code, you need *sudo*.
E.g., *sudo ./Lab2EX2*

Software PWM

- Works on any GPIO pin
- Uses *softPwmCreate()* and *softPwmWrite*

```
pinMode(fan_pin,OUTPUT);
softPwmCreate(fan_pin, 0, 100);
softPwmWrite(fan_pin, value);
```

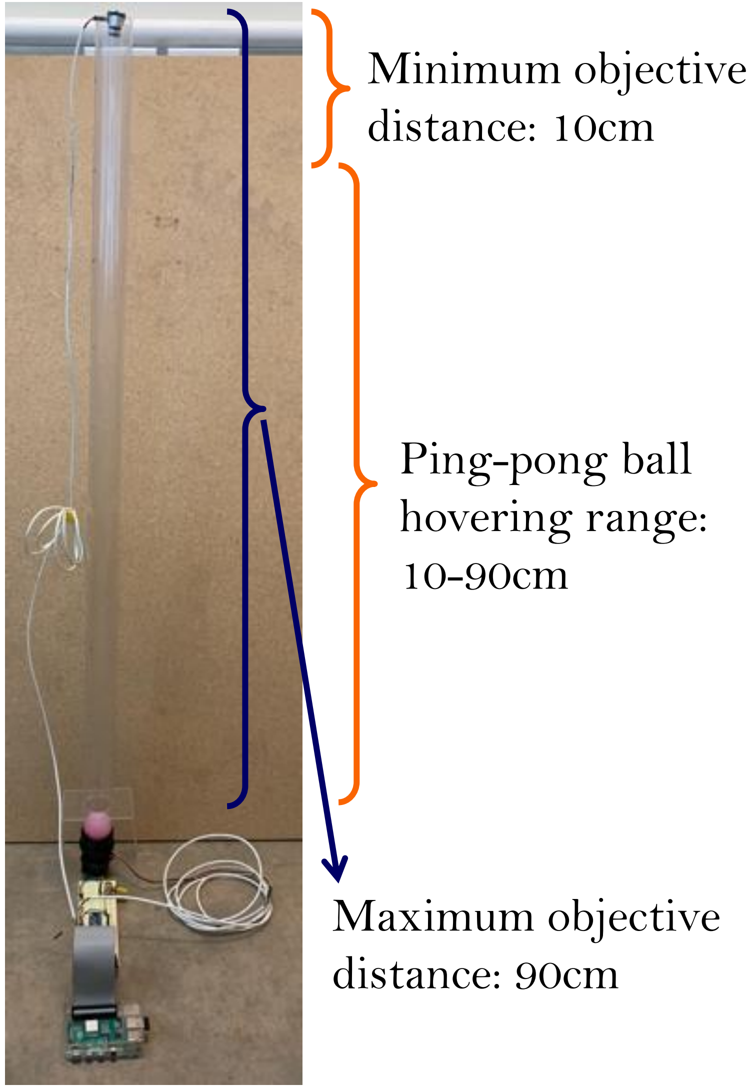
Tuning the PID for Fan control

- Target hover range: 10-90cm
- Use this 6-step tuning strategy:

    1. Start with all PID gains set to 0
    2. Gradually increase P until you get oscillations
    3. Increase D to reduce or eliminate oscillation
    4. Repeat steps 2 and 3 to find a stable setting
    5. Lock in P and D
    6. Increase I slowly to eliminate steady-state error

Link for reference: https://robotics.stackexchange.com/questions/167/what-are-good-strategies-for-tuning-pid-loops

Note: Tuning PIDs is dependent on the characteristics of the system. You don't have to follow the above steps strictly. You are encouraged to explore a better tuning strategy.



PID Control logic – Code walkthrough

- Measured Distance (MD): From sonar
- Target Distance (TD): From potentiometer
- Error = MD - TD

The PID control is implemented as:

Error = MD – TD;
*cumError += Error *elapsedTime;*
rateError = (Error - lastError)/ elapsedTime;
*PID_total = Kp * Error + Ki * cumError + Kd * rateError;*

Supporting variables:

- *cumError* tracks accumulated error (for integral term)
- rateError calculates change in error (for derivative term)
- *elapsedTime* is fixed (e.g., 50ms)

```
void PID(float kp, float ki, float kd){
    /*read the objective position/distance of the ball*/

    /*read the measured position/distance of the ball*/

    /*calculate the distance error between the obj and measured distance */

    /*calculate the proportional, integral and derivative output */
    PID_p = ;
    PID_i = ;
    PID_d = ;
    PID_total = PID_p + PID_d + PID_i;

    /*assign distance_error to distance_previous_error*/


    /*use PID_total to control your fan*/

}


/* use a sonar sensor to measure the position of the Ping-Pang ball. you may reuse
your code in EX1.*/
float read_sonar()
{

}

/* use a potentiometer to set an objective position (10 - 90 cm) of the Ping-Pang ball,
can change the objective distance. you may reuse your code in Lab 1.*/
float read_potentionmeter()
{
```

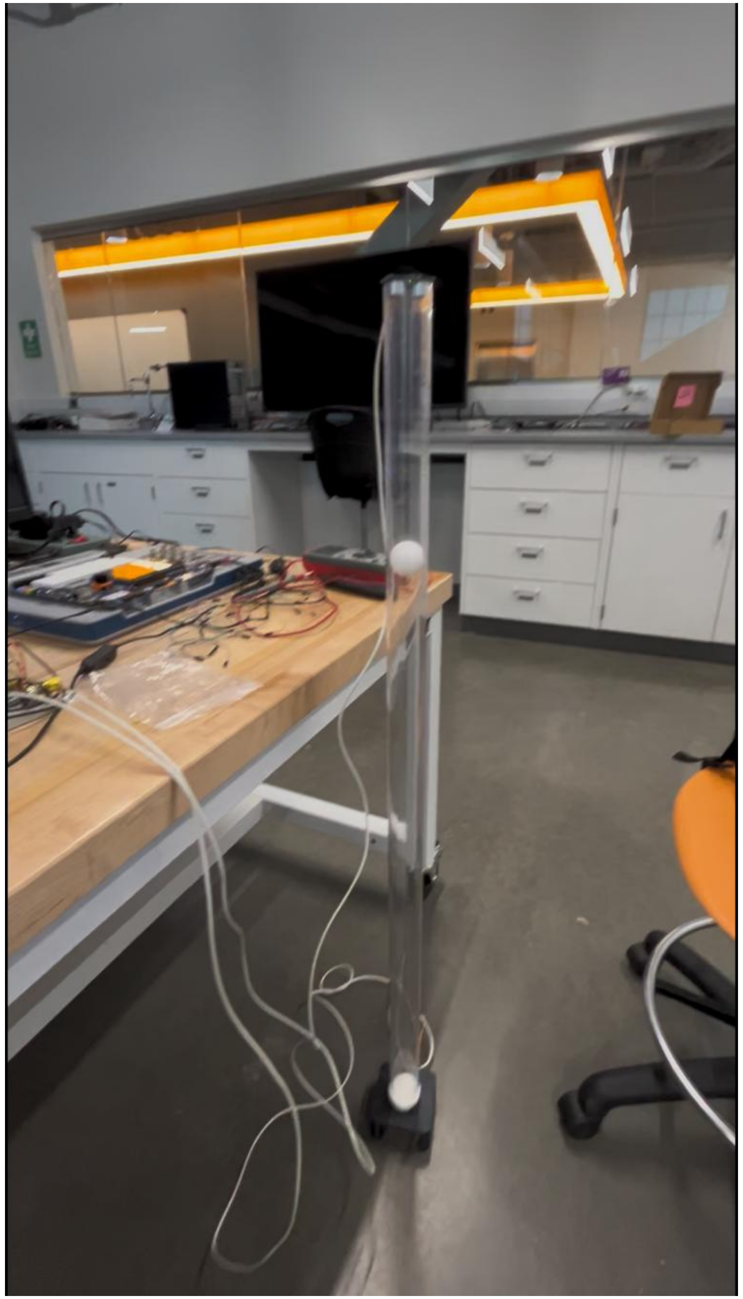
Example PID tuning values

Refer to this table for starting points:

Case	Kp	Ki	Kd
1	25	5	8000
2	7	0.0035	1.3
3	25	3	7
4	10	5	8600
5	0.3	0.04	500

🧠 **Keep in mind:** Optimal PID values depend heavily on your fan power, sensor accuracy, and tube geometry.

Experiment demo



I. Lab 1 Due Date and Submission

II. Lab 2 Introduction

III. Lab 2 Due Date and Submission

Due date (Two weeks)

- Lab demonstration:
  - ✓ no later than 7: 20 pm, September 23, 2025 (Tuesday Session)
  - ✓ no later than 7: 20 pm, September 24, 2025 (Wednesday Session)
  - ✓ no later than 5: 20 pm, September 26, 2025 (Friday Session)

- Lab report:
  - ✓ no later than 11: 59 pm, September 23, 2025 (Tuesday Session)
  - ✓ no later than 11: 59 pm, September 24, 2025 (Wednesday Session)
  - ✓ no later than 11: 59 pm, September 26, 2025 (Friday Session)

What to submit?

A ZIP file that includes:

- Lab report (Word or PDF file)
  - Supplemental questions
  - Screenshots of your results
  - Pictures of the circuits
- Your code
  - Lab2EX1.cpp
  - Lab2EX2.cpp

Note: One group, one lab report.

Grading Criteria

The grading criteria is same as listed on the handout; however, if you don't demonstrate your code to TA, then 50% of maximum points are reduced directly.

Office hours

- Tuesday : 4:30 pm – 5:30 pm, Endeavor 350
- Wednesday: 4:30 pm – 5:30 pm, Endeavor 350
- Friday: 3:30 pm – 4:30 pm, Endeavor 350