Module 6: Control Systems

Engineering Disciplines: Systems and Control Engineering, Electrical Engineering

For the Part 1 Lab Report
Each team will submit:

- 1 .m file containing the code for Labs 1 & 2
- 1 PDF file (all in one file) containing typed answers to each to the Labs 1 & 2 discussion questions and the completed Project Management Task List

For the Part 2 Lab Report
Each team will submit:

- 1 .m file containing all code for this module
- 1 PDF file (all in one file) containing typed answers to each of the Lab 3 & Lab 4 discussion questions and the completed Project Management Task List

  Submit these files in accordance with the course assignment submission guide.

Labs 1 & 2: *PID* Simulation and Testing Components

Introduction

What is a controller and how is a controller characterized?

A controller is a set of processes that reliably produces a desired output. Control systems are used in all kinds of technology including heating/cooling, pharmaceutical manufacturing, and spacecraft. When designing a control system, we need to consider how our technology will respond to inputs over time. The time response to a change in input can be divided into two parts: the transient (short term) response, and the steady state (long term – if there are no changes in input) response. Imagine turning on a garden hose. Initially it is in a steady state with no water flowing out of it. As the spigot valve is turned, the water flow gradually increases during the transient state until it reaches a desired steady state with the water flowing at a constant rate. In this lab, we are going to examine how a system approaches steady state and why the properties of getting there are important. Look at the signal in Figure 1. The label “unit-step input” refers to a change in the system from one state to another, like turning a switch from OFF to ON. The graph of *y(t)* represents the system’s response to that change in input.

Key characteristics of the system response:

1. **Steady State Value (y_{ss}):** the long-term value of the system output. Hint: You may eyeball the time the curve takes to settle and take the average output value of these last several values.

E.g. the last 100 values seem to have the most minimal fluctuations, then you could calculate the mean value of these 100 values.

2. **Maximum Overshoot:** the maximum value of the system output minus the steady state value. This is often reported as a percentage:

$$y_{os,\%} = \frac{max(y) - y_{ss}}{y_{ss}} * 100\% \quad (1)$$

3. **Delay Time (t_d):** the time required for the response to reach 50% of its final value.
4. **Rise Time (t_r):** the amount of time to rise from 10% to 90% of the response's final value.
5. **Settling Time (t_s):** the time required for the response to decrease and stay within 5% of the final value. Note the dashed lines mark 1.05 and 0.95 when showing the t_s value.

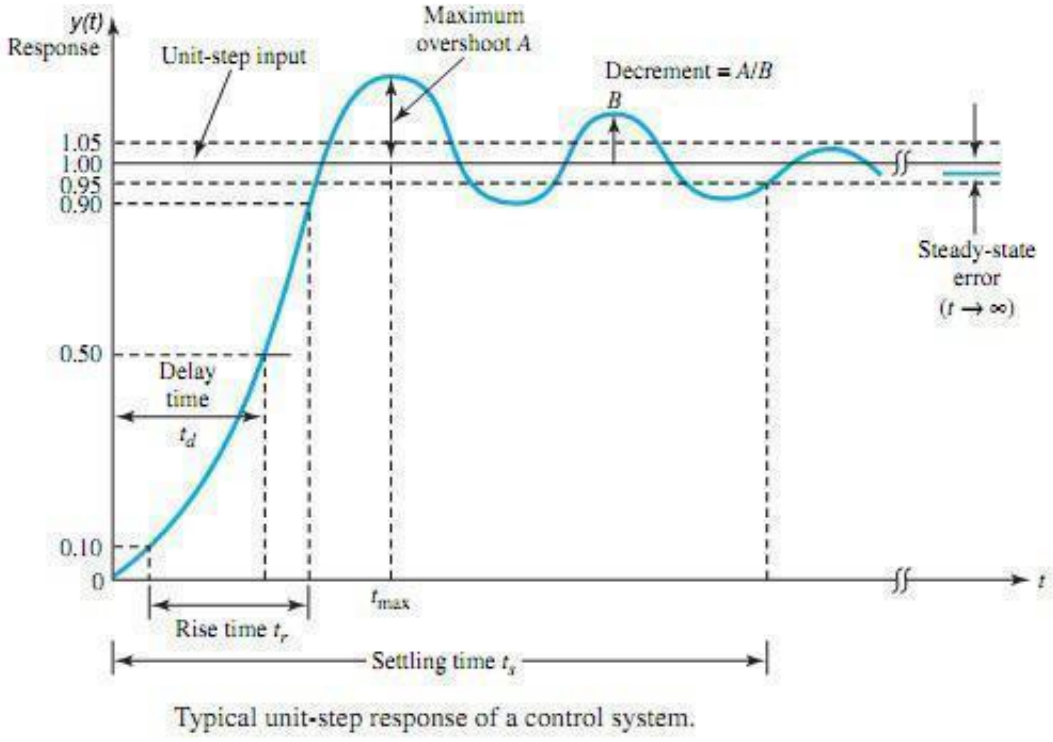

Figure 1: System response to unit-step input [source]

Control systems are built to optimize the system response. This is a balancing act. For example, choosing a low maximum overshoot percentage might increase the rise time. A systems engineer decides what gains to prioritize, as trade-offs must be made in the final system operation.

One of the best-known types of controllers is the proportional, integral, derivative, or PID, controller. It allows us to tune the speed and overshoot of a system to create fast and accurate responses. An electric motor is a simple device that we can control. We will model the rotational speed response of

an electric motor, measured in revolutions per minute (RPM) as a voltage is applied to it. We will step through the process of applying a P, I, and D controller, adding each type in one by one. The mathematical equation that describes a PID controller is:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad (2)$$

where $u(t)$ is the output of the controller. The output depends on the error at time t, $e(t)$, and the tuning parameters, $K_p, K_i, and\ K_d$. These tuning parameters represent the gains for the proportional, integral, and derivative terms.

In today's lab, you will investigate and analyze sample data from a fan motor. Based on the documentation provided by the manufacturer, the motor should provide an output speed of 1000 RPM when connected to a potential difference of 12 V.

Materials

- A copy of the function file "control_system.m"
- A copy of the file "MEASURED_RPM.mat"
- Control Systems Toolbox Add-on
- Breadboard
- Digital Multimeter (DMM)
- 10-kΩ potentiometers
- Jumper wires
- IRLZ14 Field effect transistor (FET)
- Power supply

Procedures

1. *Project planning and management*

Create your own Project Planning and Management Task List for all tasks in this module, following the instructions in the Project Planning and Management document. You will continue to edit your task list this week and submit your task list, current as of the submission date, as part of your Part I Report.

2. *Open loop control*

a. You should already have the "Control System Toolbox" Add-on installed in MATLAB. Verify by doing the following.

i. In the "Home" tab of MATLAB, in the "Environment" subheading, select the drop-down arrow beneath the "Add-Ons icon" (3 cubes) and select "Get Add-Ons". You should see the Control System Toolbox.
ii. If you do not see the Control System Toolbox, click the Add-Ons icon and search for "Control System Toolbox" to install the Add-On.

b. Load the fan speed data file (MEASURED_RPM.mat) into your script. There are two variables: `time` and `RPM`.

c. Plot RPM vs. time and use the `grid on` command to add a grid to your plot.

d. Add a horizontal line to your plot showing the target speed provided by the manufacturer of 1000 RPM. Adjust the line properties and limits of the plot so that the setpoint line is easy to see. Hint: Look up "yline" function.

3. Write Waveform Characteristics Function

a. Write an algorithm for a function that receives the RPM and time data as input arguments and outputs the final RPM value, max overshoot, rise time, and settling time.
b. Write the function from the algorithm in the previous step. Name it `getWaveformData`. Run it and fill in Table 1 below. Check your answers by visual inspection.

Table 1: Open loop control

Final Value (RPM)	Max Overshoot %	Rise Time (t_r)	Settling Time (t_s)

c. The data above should indicate that the response of this open-loop motor is not going to be acceptable because the motor speed does not reach the target RPM.

Recommended End Point for Lab 1

4. P controller

a. A motor control unit incorporates a sensor that measures RPM and a circuit that adjusts the voltage supplied to the motor. The file `control_system.m` is a specialized MATLAB function to simulate these RPM measurements and voltage adjustment.
   i. The input arguments for the `control_system` function are characters (case sensitive) for the Proportional (*P*), Integral (*I*), or Derivative (*D*) followed by the numerical value you would like to set for it.
   ii. If no value is given for a *P*, *I*, or *D*, the controller does not have that gain.

iii. To use just the *P* part of the controller, for example, the function call is `control_system('P', <num>)` and it returns two vectors of the format `[new_time, new_RPM]`.

iv. Similarly, to use just the P and D parts of the controller, the function looks like `control_system('P', <num1>, 'D', <num2>)` and will return two vectors of the format `[new_time, new_RPM]`.

b. There are algorithms to optimize gain parameters, but we are going to use an informed trial and error approach to find a good value for *P*. Try at least 5 values until you find one that reaches a steady state of around 1000 RPM quickly. For now, it is OK if it oscillates while getting to that steady state. The maximum available *P* value is 100. Hint: You do not need to write a loop for this since there are only 5 trials. The same goes with all 3 controllers.

c. Run the code with the different proportional gains and plot the controlled system on the same figure.

d. Fill in the rows of Table 2.

Table 2: *P* Control

P	**Final Value (RPM)**	**Max Overshoot %**	**Rise Time (t_r)**	**Settling Time (t_s)**

5. PD controller

a. Once you find a *P* gain that works well, the next step is to reduce the initial oscillations. The derivative gain will help do this by "predicting the future" and slowing down the motor input voltage as it approaches its target value. This will reduce the overshoot. Decreasing oscillation also reduces wear on physical components

b. Create another control system with the optimal *P* gain from Part 3 and try five values for the *D* gain until you find the value that removes most of the initial oscillations. The max *D* value is 20. At this point, the output of the controller should be critically damped,

meaning that it will complete one critical oscillation before reaching steady state.

c. Create a new figure and plot the system responses with the five *D* gains you tested in the same graph. Fill in the rows of Table 3.

Table 3: *PD* Control

D	Final Value (RPM)	Max Overshoot %	Rise Time (t_r)	Settling Time (t_s)

6. PID controller

a. At this point, the initial response should look decent. In fact, if a *PD* controller is much easier or more cost effective to implement in a system, many engineers stop here, but we won't. The integral, *I*, piece of the controller fixes something that might be a little difficult to notice. Did you check to make sure your final value is 1000 RPM? The integral gain can remove the steady state error.

b. Using the previously selected *P* and *D* gains, finish out your control system by incorporating an *I* value. Again, try out five values to see which value restores the setpoint the most accurately. The max *I* value is 20.

c. Create a new figure and plot the system responses with the five *I* values you tested in the same graph. Fill the rows of Table 4 of your selected *I* gains.

Table 4: PID Control

I	Final Value (RPM)	Max Overshoot %	Rise Time (t_r)	Settling Time (t_s)

7. Test Electrical Components

a. Use the power supply to test the blower fan.

i. Set channel 1 (CH 1) on the power supply to 12 V and limit the current to 0.75 A.

ii. Connect the power supply cable to the blower fan. Turn on the power supply output and the fan should run.

b. Test the FET.

i. The FET (Figure 4) we will use in this lab has the following pinout:

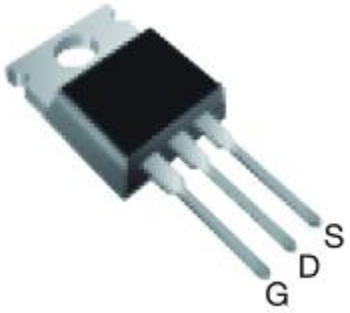

Figure 4: FET pinout

ii. Where G is the gate, D is the drain, and S is the source (using the terminology common to FETs). The basic operation of a FET utilizes the fact that a voltage applied to the gate will allow current to flow through the drain and source pins.

iii. Set the DMM to test continuity (Fig. 5) and touch the conductive part of the test leads together to make sure the meter is set correctly (the meter should indicate an electrical path between the test leads with an audible alarm).

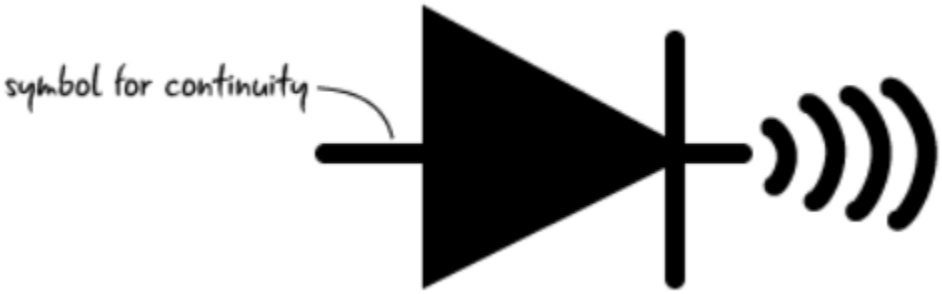

Figure 5: DMM symbol for continuity

iv. Build the circuit shown in Figure 6. Set channel 3 on the power supply to 5 V and limit the current to 0.1 A. Observe the continuity through the FET with the power supply off, and with the power supply providing 5 V to the gate. Discuss with your team whether the result of your test with the DMM is consistent with the operation of the FET described above.

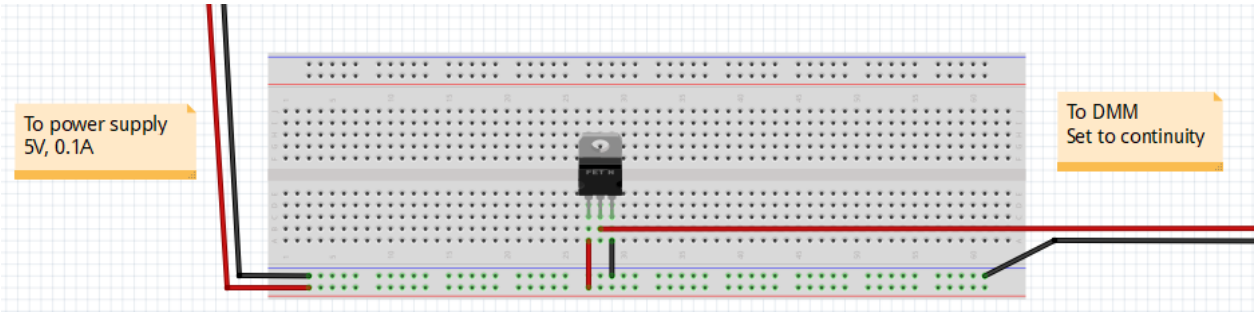

Figure 6: Circuit to test the FET

Questions

1. Report your group's filled in Table 1, Table 2, Table 3, and Table 4.
2. What waveform characteristic does the *P* gain of the controller affect most?
3. How does the *D* gain help the controller achieve our goal?
4. What was your deviation from the setpoint without the *I* gain? Is it beneficial to use the *I* gain in this scenario?
5. Can you think of any other devices where a *PID* controller is critical to use?

END OF LABS 1 & 2

Lab 3: Manual Control

Introduction

In this lab, you will manually control a blower fan to levitate a puck at a specified height. In the following labs, you will use *PID* to automatically control the system.

Pulse Width Modulation (PWM)

To control the blower fan that levitates the puck, we will rapidly cycle voltage to the fan, turning it on and off. If you measured the voltage using an oscilloscope, you would see a square-shaped signal like those in Figure 1.

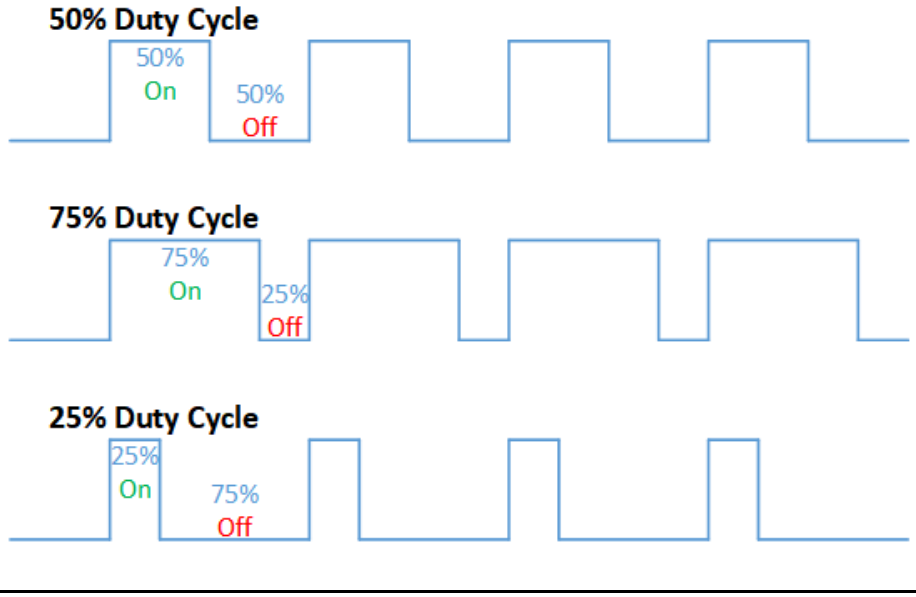

Figure 1: Pulse width modulation signals [source]

To make the motor on the fan go faster, increase the amount of time the voltage is turned on, relative to the time the voltage is turned off. As you increase the amount of time the voltage is turned on, the width of the “pulse” increases, corresponding to the time that the voltage is on. Therefore, we can change the speed of the fan by modulating or changing the width of the pulse, hence the term Pulse Width Modulation (PWM).

PWM is often described quantitatively using the duty cycle, which is the proportion of time that the voltage is turned on. At 25% duty cycle, the fan is turned on 25% of the total time. Look at Figure 1 to see the relationship between the pulse width and the duty cycle.

Serial communication

The Arduino can generate a PWM signal and perform other real-time tasks. However, communication through MATLAB is slow. For this lab, you will run a pre-written C++ program on the Arduino and send commands to it using the serial communication functions in MATLAB. This offloads the basic tasks to the Arduino and allows it to respond faster.

Materials

- Oscilloscope
- Oscilloscope probe
- 3D printed puck
- Blower fan
- 3D printed cap with Infrared (IR) sensor
- Clear tube with ruler attached
- Wiring harness
- Perfboard
- Breadboard
- Digital Multimeter (DMM)
- Arduino with manual control firmware (marked with white tape)
- I2C LCD
- 10-kΩ potentiometer
- Jumper wires
- IRLZ14 Field effect transistor (FE

Procedures

1. Build the Arduino control circuit

a. Build the control circuit according to Figures 2 and 3. When making connection A, make sure that the red wire is connected to the positive rail and the black wire is connected to the negative rail. Pin B can be inserted into any of the four holes marked “B”. When making connection C, make sure that the lines on the labels align.

b. Do not connect the Arduino to your laptop for this part. After the rest of the circuit has been assembled, connect the Arduino (which should be marked with white tape) to the USB cable at your lab station. The LCD should show two numbers for the fan and sensor values. The first number is the duty cycle of the motor as a percentage. The second number is the sensor's voltage output.

c. Verify that turning the knob on the potentiometer closest to connection “C” changes the fan speed on the display and changes the speed of the motor.

d. Use the oscilloscope to verify that the duty cycle shown on the display matches the actual output:
   i. First, run the probe calibration as always before using an oscilloscope.
   ii. Check that the oscilloscope is set to “normal” acquisition mode under the “acquire” button’s menu. **Hint:** using the “Default Setup” and “Autoscale” buttons can be helpful to quickly acquire the signal.
   iii. Connect the ground clamp of the probe to the ground rail on the breadboard

iv. Connect the probe to digital output 3 on the Arduino by loosening the jumper wire occupying digital output 3 just enough to hook the probe onto the wire without losing connection.

e. Observe the signal to verify that the duty cycle on the oscilloscope approximately matches the fan duty cycle on the LCD. Record the frequency of the signal for several duty cycles to answer Question 1.

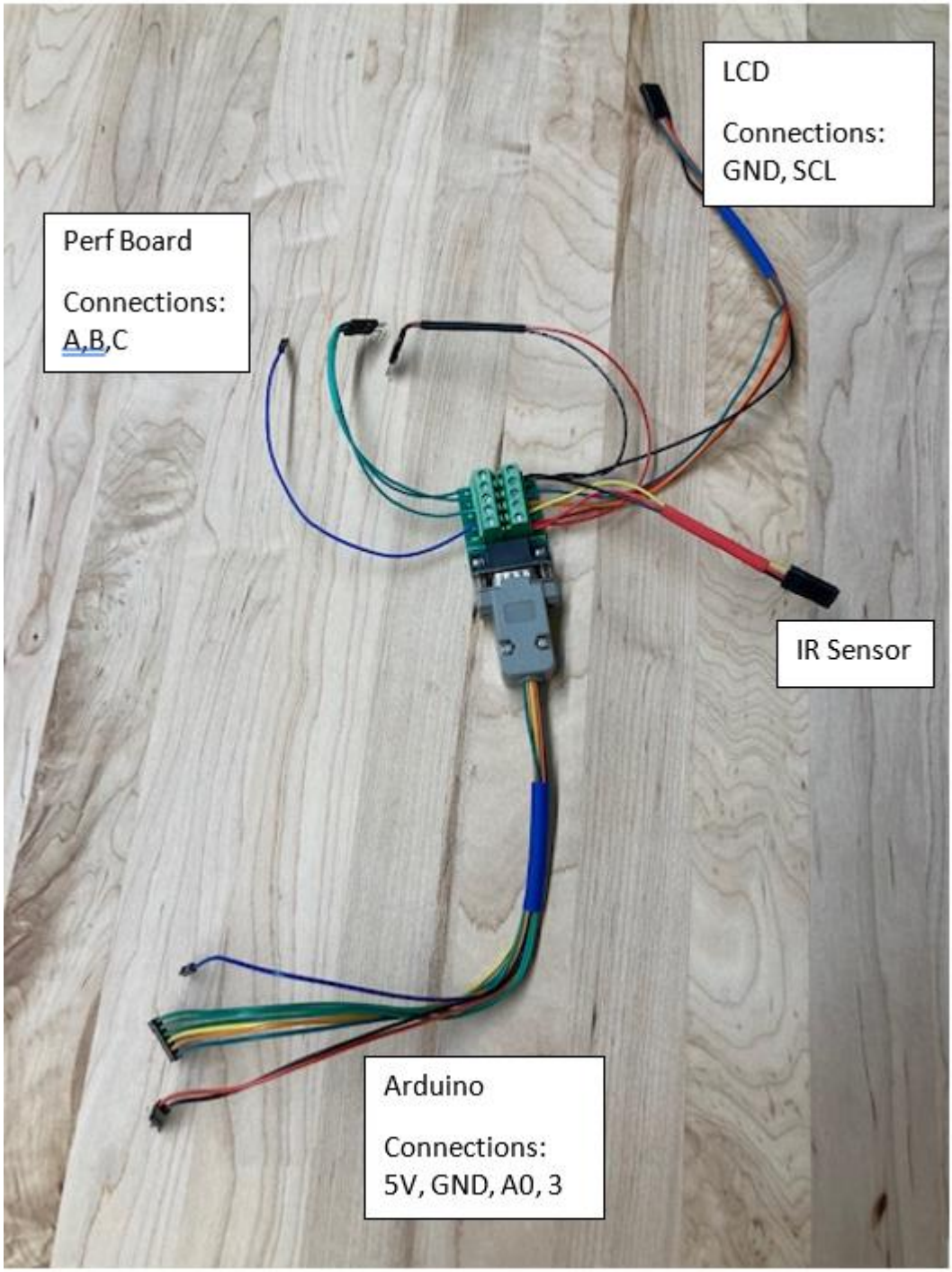

Figure 2: PWM Wiring Harness and Connections

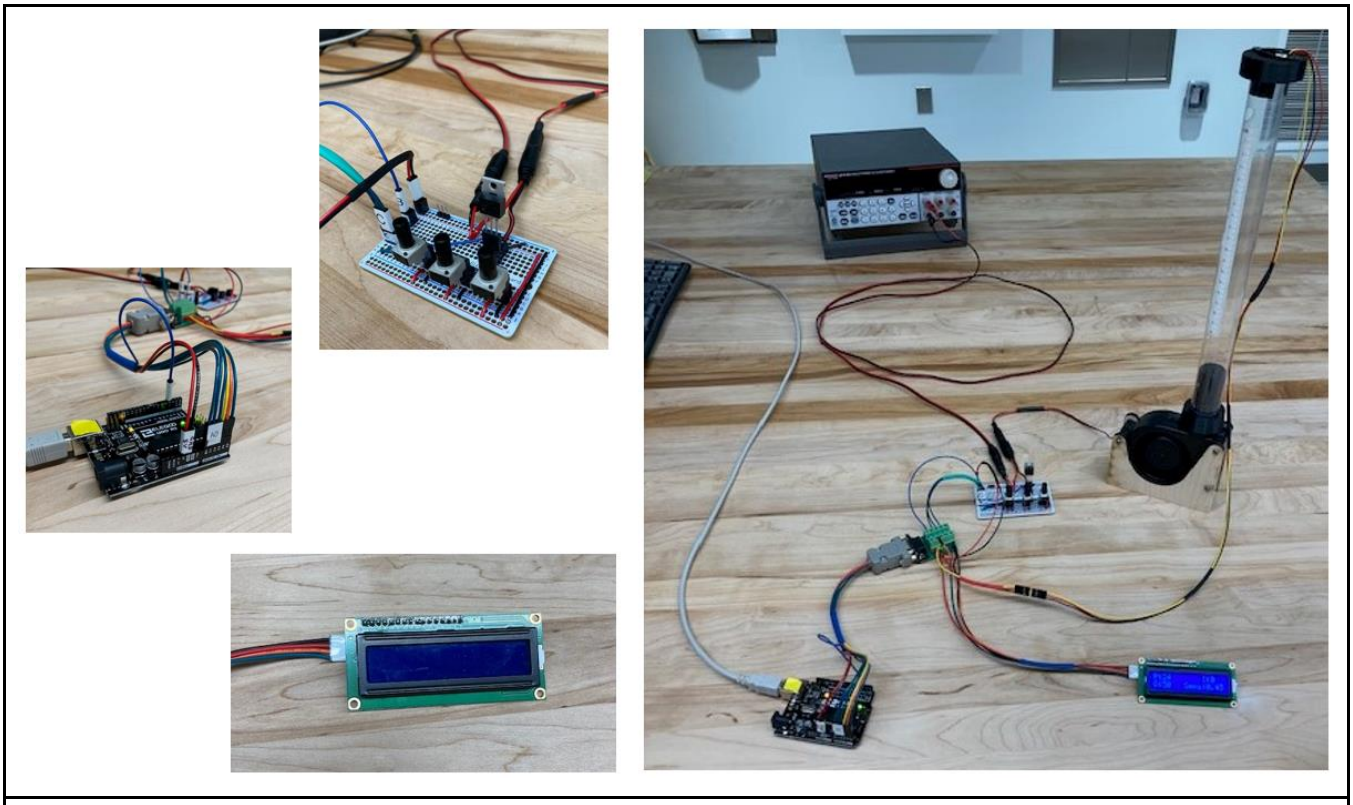

Figure 3: Assembled Circuit

2. *Calibrate the infrared sensor*

a. Wave your hand in front of the infrared (IR) sensor to verify that the sensor voltage displayed on the LCD changes. Then reattach it to the top of the plastic tube.

b. Move the puck to four different positions along the length of the tube between 6 and 16 cm away from the sensor. Record the sensor voltage for the distance at these four points. You might observe that the sensor readings change erratically outside this range.

c. Use these data to create a function that takes in the sensor reading (voltage) and outputs the corresponding distance (cm). **Hint:** Try `polyfit` and `polyval`.

3. *Build puck levitation system*

a. Make sure the power supply output is turned off.

b. Assemble the tube, blower fan, and sensor (Fig. 3).

c. Connect the sensor and motor back to the circuit. You will need to use jumper wires to extend the sensor wires to reach the breadboard.

4. Manual control

a. Now, you will act as the controller to float the puck at the desired height (aka the setpoint). You will use the oscilloscope to collect the sensor values until the system reaches steady state.

b. First, set up the oscilloscope to collect data from the sensor:
   i. Move the oscilloscope probe to the output on the sensor (yellow wire).
   ii. Use the Acquire menu to change time mode to “Roll”.
   iii. Use the Acquire menu to change acquisition mode to high resolution. This will help eliminate some of the noise and result in a cleaner signal.
   iv. Set your vertical scale to 1 V/div and horizontal scale to 2 s/div.

c. Turn on the power supply output.

d. Practice turning the knob on the potentiometer to float the puck at different heights and observe how the sensor voltage changes on the oscilloscope. Adjust the display as needed.

e. Once you feel confident in your ability to bring the puck to a desired height quickly, get ready for a timed run. You will save the controller’s (i.e., your) response to a changing setpoint. This part is best done with one group member recording data on the oscilloscope and one group member controlling the fan speed.
   i. Turn the potentiometer all the way down to stop the fan. The data on the oscilloscope should stabilize at a low value.
   ii. When ready, adjust the potentiometer to levitate the puck so that the bottom is lined up with the 10-cm mark on the scale. Once the puck reaches the setpoint and stays there, press the “Start/Stop” button on the scope to stop recording data. The signal should look like the simulated control signals from Lab 1.
   iii. Make sure the data on the scope shows everything from your initial response until the steady state. Adjust the time scale and retry if necessary.
   iv. Save the data from the scope as a .csv and a .png file.

5. Analyzing your manual control of puck height

a. Import the .csv to MATLAB and convert the voltage data to distance (cm) using your function.
b. Using your MATLAB function from Lab 1, calculate the final value (cm), max overshoot (%), rise time (s), and settling time (s) of your best attempt at stabilizing the ball.

Questions

1. What is the frequency in Hz of the signal driving the motor? How does this frequency change with the duty cycle?
2. Attach the .png file of the data you saved in Part 4.e.iv. and fill in the values describing your control into Table 1 as shown below.

Table 1: Manual Control Parameters of Levitating Ball.

Final Value (cm)	Max Overshoot %	Rise Time (t_r)	Settle Time (t_s)

3. How difficult was it to control the ball's position using the potentiometer? Do you think you could drive a car with this control, with practice?
4. Before Spring 2023, the circuit in Fig. 3 was built on a breadboard (Fig. 4). Explain the benefits and drawbacks of using a breadboard vs. the wiring harness for circuitry that you used in the current lab.

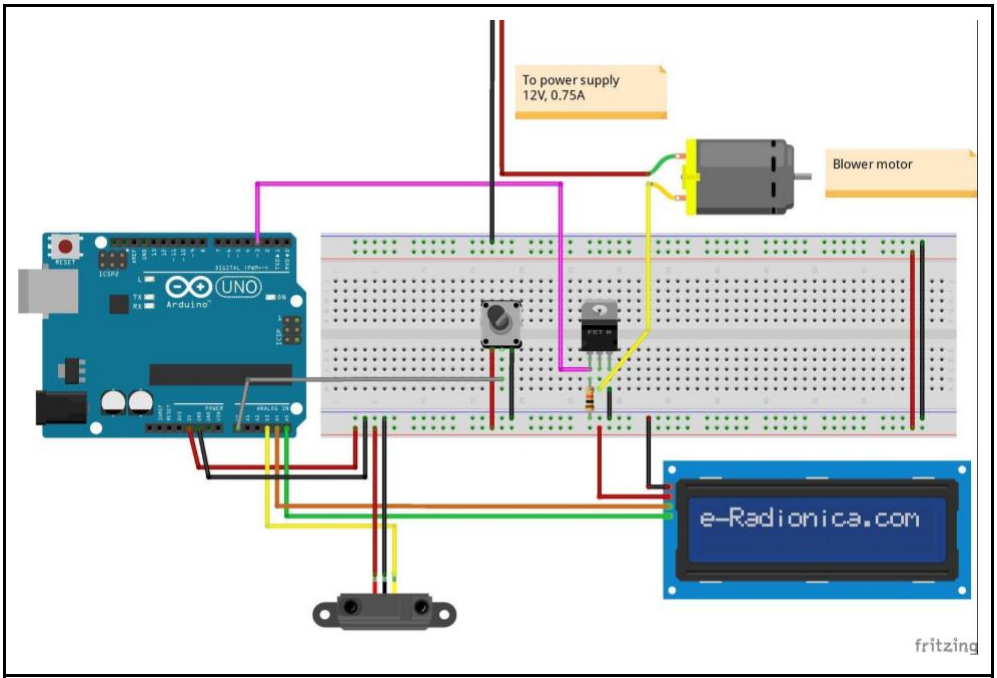

Figure 4: PWM control circuit

END OF LAB 3

Lab 4: PID Tuning

Introduction

You worked with a simulated *PID* controller in Labs 1 & 2. In this lab, you will apply your code to a real system, the floating puck from Lab 3, and analyze the system's response using the function developed in Labs 1 & 2. The *PID* algorithm needs to be tuned to specific applications by adjusting the gains (K_p, K_i, K_d) for the three terms in Equation 2 from Labs 1 & 2.

Materials

- Levitating puck setup from Lab 3
- Arduino with PID control firmware (marked with yellow tape)

Procedures

1. Build the control circuit from Lab 3

a. Replace the Arduino with an Arduino that has the PID control firmware. The board should be marked with yellow tape for this lab.
b. Connect the Arduino to the USB cable provided at your lab station. The display should turn on and show three values for each of the three control gains. Turn the potentiometer knobs to verify that the numbers change when you adjust the knobs.
c. The Arduino is pre-programmed to maintain a sensor reading setpoint of 2.6 V. Use your sensor calibration function from Lab 3 to determine the corresponding height within the tube and make note of this value.

2. Tune the PID controller

The controller should now be tuned. This is like the tuning exercise from Lab 1, but the simulated motor is replaced by a physical system. We want the system to do two things: bring the puck quickly to the desired height and keep it there with minimal overshoot and oscillation. We will use a methodical approach to study how adjustment of gains achieves this.

a. Turn on the power supply.

b. Connect the oscilloscope probe to the output on the sensor (yellow wire).

c. Connect the ground clamp to the ground rail on the breadboard.

d. Adjust the oscilloscope to display the system response:
   i. Use the Acquire menu to change time mode to "Roll".
   ii. Use the Acquire menu to change acquisition mode to high resolution. (This will help eliminate some of the noise and result in a cleaner signal.)

iii. Set the controls so that the scale is 0.5 V/div, the ground offset is near the bottom of the screen, and the time scale is set to around 3-5 s/div. Adjust these settings as needed so that the system's response (since pressing the reset button) is displayed until it reaches steady state (Fig. 1).

iv. Add a Y1 cursor to help put the response in context. Press the Cursors button and choose Cursors → Y1. Adjust the Y1 cursor (a dashed horizontal yellow line) to show the setpoint voltage of 2.6 V on the oscilloscope screen.

e. Adjust the potentiometers on analog inputs 1 and 2 so that *I* and *D* gains are set to zero.

f. Adjust the *P* gain to try to control the height of the levitating puck. What do you observe? If your puck sticks in the tube or adjusting the controller does not seem to be changing the system, press the red reset button on the Arduino to restart the controller.

g. Now adjust the *P*, *I*, and *D* gains to get a controller that gives a satisfactory response. Record all three coefficients.

3. Collect and plot data using the Oscilloscope

a. With satisfactory *P, I,* and *D* gains, record the sensor voltage on the oscilloscope.

i. Press the red button on the Arduino to restart the controller, and the oscilloscope will record the sensor voltage.

b. Once the voltage stabilizes, press the "Start/Stop" button on the oscilloscope to stop recording data. The signal should resemble the plot in Fig. 1. Once you are satisfied with the captured sensor voltage on the oscilloscope, save the data from the scope as a .csv file. You will need to unplug the Arduino to save the files to a USB.

c. Import the .csv to MATLAB.

d. You should now have a plot that shows a voltage value on the vertical axis and the number of the data point on the horizontal axis. It is more useful to show this information in terms of the quantities we are interested in, distance (cm) and time (s):

i. Using the sensor calibration function from Lab 3, convert the voltage values into distance.

ii. Add a horizontal line corresponding to the setpoint voltage of 2.6 V.

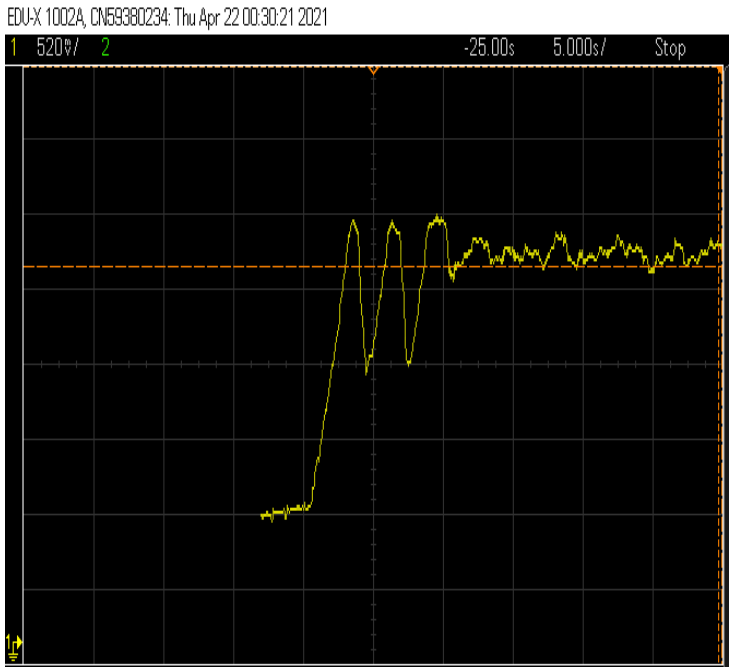

Figure 1: Height of the puck over time measured by the oscilloscope; note that the steady state still shows a small oscillation

e. Finally, you will use your function from Lab 1 to output the values required for Table 5 below. The signal oscillates around a steady state (which is why we need a controller in the first place!) To find the steady state in this real-life system response, modify your function so that it averages the last 100 values from the signal to determine the steady state value.

Table 5: Final PID response.

Final Value (cm)	Max Overshoot %	Rise Time (t_r, s)	Settle Time (t_s, s)

4. Project Planning Task List

Complete your final Project Planning and Management Task List. You will turn in a pdf of your final task list with your Part II Lab Report.

Questions

1. Which *P*, *I*, and *D* gains worked best for your control? Give numerical values and explain what criteria you used to define the “best” control system.
2. Overall, did the manual or *PID* control work better?

3. What are some major differences between the simulated control system from Lab 1 and the real system we worked with? Think about the difference between the physics of the two systems and how that changes the controller's response.
4. You might notice there is a lot of noise in the signal. What are some ways you can reduce the noise and get a more accurate reading of the position?
5. What could be improved about the system?

END OF LAB 4