Goals
Understand how to implement and tune a PD controller to control the position of a DC motor.

Explore the frequency response of the PD controller by testing how it responds to sinusoidal inputs of different frequencies.

Parts & equipment

<table>
<thead>
<tr>
<th>Qty</th>
<th>Part/Equipment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seeeduino board</td>
</tr>
<tr>
<td>1</td>
<td>Motor driver</td>
</tr>
<tr>
<td>1</td>
<td>DC motor with encoder</td>
</tr>
</tbody>
</table>

Introduction
The most common controller used by engineering designers to control the movement of a motorized part is the PD (proportional-derivative) controller (sometimes and integral control term is added, to create a PID controller, but we will not explore I control in this lab). In this lab you will implement a PD position controller. You can use the position controller code from this lab for your steering system for your final project. Such controllers are also used robot arms, radars, numerically controlled milling machines, manufacturing systems, and control surfaces on aerospace vehicles. The PD control law is:

\[ \tau = -K_p(\theta - \theta_d) - K_d(\dot{\theta} - \dot{\theta}_d) \]

where:
- \( \theta_d \) desired motor angular position
- \( \theta \) actual motor angular position
- \( \dot{\theta}_d \) desired motor angular velocity
- \( \dot{\theta} \) actual motor angular velocity
- \( \tau \) desired motor torque
- \( K_p \) proportional gain
- \( K_d \) derivative gain
Note that the controller has two terms – one proportional to the position error (the “P” part), and one proportional to the derivative of position (i.e. velocity, the “D” part). Thus, it is called a “PD” controller. The implementation of this controller for a DC motor with inertia, J, is shown in Figure 2.

![Block diagram of the PD controller](image)

Figure 1. Block diagram that you will implement to make the PD controller for the motor. J is the inertia of the motor shaft.

**Part I: Step response of the actual system**

Construct the Arduino circuit in Figure 2. Download and run the code for this lab from the web site and run it on the Arduino. As in previous labs, when you open the Serial Port Monitor (or when you press the 'reset' button on the Arduino), the code will re-initialize, then run a step response and send time, position, and desired position of the motor to the monitor for the response. You can copy and paste this into Excel to display the step response. *Use the template provided on the class website for Excel so that you can quickly see the motor’s response.*

To understand how the actual system behaves we need to first understand its dynamics. First, let’s look at the dynamical equation that describes how \( \theta \) evolves with time when the controller is attached to the motor.

Dynamics of the motor and shaft:

\[
\tau = J \dot{\theta}
\]

Dynamics of the controller system:

\[
\tau = J \ddot{\theta} = -K_p(\theta - \theta_d) - K_d(\dot{\theta} - \dot{\theta}_d)
\]

Re-writing to make input-output clear:

\[
J \ddot{\theta} + K_d \dot{\theta} + K_p \theta = K_a \theta_d + K_p \theta_d
\]
This differential equation has similar dynamics to a mass-spring-damper system with Force as the input and Position as the output. That is, it follows the same equations of motion. This allows us to use our intuition about mass-spring-damper systems when designing and tuning a PD controller.

Recall that the differential equation of motion for a mass-spring-damper system is given by:

\[ m\ddot{x} + B\dot{x} + Kx = F \]

and thus using the analogy to the PD controller we have that:

<table>
<thead>
<tr>
<th>mass-spring-damper system</th>
<th>PD controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>mass</td>
</tr>
<tr>
<td>B</td>
<td>damper</td>
</tr>
<tr>
<td>K</td>
<td>spring</td>
</tr>
<tr>
<td>F</td>
<td>input force</td>
</tr>
</tbody>
</table>

In a mechanical system, if you wanted to the system to respond more quickly, you would increase the natural frequency \( (\omega_n) \) by picking a stiffer spring (higher \( K_p \)). Which variable would you change in your differential equation for the PD system to make your system respond more quickly (i.e. increase its natural frequency)?

*Note: there is a limit to how big you can make this variable because of the time delays in this sampled data system.*

By adding the derivative gain (\( K_d \)) to control the position of the motor we must now take into account the concept of damping when designing and implementing the controller. With damping in the controller we can have four types of behaviors:

- **Undamped** (i.e. zero damping): The system oscillates at its natural frequency. These oscillations are a function of the controller's \( K_p \) gain.
- **Underdamped**: The system will move to its desired position and oscillate about this position with oscillations gradually decreasing to zero.
- **Critically damped**: The system will move to its desired position as quickly as possible without oscillating.
- **Overdamped**: The system will move asymptotically towards its desired position without oscillating.

Suppose you didn’t want your motor to oscillate too much. *This is an important issue!* PD controllers are used in many applications such as NC milling machines, plotters, and for your final project steering. You usually want your motor to go to a desired value quickly and accurately without oscillating. Change the derivative gain \( K_d \) until you get the system to be critically damped.
To control your motor you will have to go through a process of ‘tuning’ the gains of the controller. You tune a PD controller based on the response that you want to get from your system (i.e. how fast do you want it to reach its desired position? Do you want to allow oscillations around the desired position?) You can follow a procedure such as this one:

1. Set the derivative gain, Kd, to zero and only change the proportional gain Kp. Change Kp so that you have an acceptable response in regards to how fast your system reaches its desired position. Note that since Kd is zero you will get oscillations around the desired position (undamped system). For your motor a good place to start is with a Kp value of 100. Then start increasing it until you get a rise time (defined in Figure 3) close to 0.038 seconds.

2. Once you have an acceptable response in terms of how fast your system reaches its desired position, you can now begin to add damping so that the oscillations will decrease (underdamped system). Because you are now adding damping to your controller, you can expect your system to react slower.

3. You can keep increasing the derivative gain until your system no longer oscillates as it reaches its desired position. This will give you a critically damped system. What happens if you keep increasing the damping gain past this point?

For your code to control the motor you will need to write the control law, u, in the Arduino program provided.
Practical Exam I

Show your TA that your system is critically damped and that it has a rise time close to 0.038 seconds (your system should respond very similar to Figure 4).

- What is the time constant at these settings?
- What is the value of Kp and Kd that you used?

Figure 3. Underdamped system. The rise time is the time it takes for your system to go from 10% to 90% of the desired position. This figure was made using values derived from the code that was provided to you (i.e. you should be able to obtain a similar figure).

Figure 4. Expected output for the critically damped system.
Part II: Frequency response of the actual system

The goal of this part of the lab is to characterize the frequency response of the system. In particular, you will explore how well the system tracks the desired input position when the input is a sinusoid, across a range of frequencies. Remember, you can view linear systems such as this one as “filters”. The PD controller acts like a low-pass filter, although it has a resonant peak if the damping is not great enough.

To characterize the frequency response of the system, follow the steps below.

**REMEMBER: DO NOT LET YOUR MOTOR OVERHEAT.**

1. Set Kp equal to the value that you found in Part I for a rise time close to 0.038 seconds and Kd = 0. Also set the variable delayToTakeData = 2.0 (this will allow the controller to run for 2 seconds before saving data to the virtual oscilloscope in the code, allowing initial transients to die out).

2. Change the Arduino code to input a sinusoid. To do this, go to the function “desiredPosition()” and comment the lines that define a square wave input and uncomment those for a sine wave. Record the output amplitude and phase shift at 1, 2, 4, 8, 12, 16Hz *(you will need these for your write-up)*
   a. Does the system have a resonant frequency? Note, you may have to search between the frequencies suggested above to find it. What should the phase difference between desired and actual position be at the resonant frequency? (Answer: 90 degrees).
   b. Which signal should be larger at the resonant frequency? (Answer: The actual position of the motor).
   c. What is the resonant frequency? Do you notice any high frequency oscillations in your output signal? What do you think might be causing those?

**Practical Exam II:**
Show your TA that you can find the resonant frequency of your system. What is the frequency, in Hz, at which resonance occurs?

**Pre-lab assignment**

1. What is the control law that you will implement in this week's lab? How does this control law related to the differential equation of motion of a mass-spring-damper system?

2. Draw the behavior for an underdamped, critically damped, and overdamped system as you change the position of the motor from 0 degrees to 90 degrees. How do you expect that the time constant will change as you move from underdamped to critically damped, and finally to overdamped?
Write-Up

1. Following the procedure in Part I for tuning the PD gains, make 3 plots of your system for when it is Underdamped, Critically Damped, and Overdamped. Label the values for Kp and Kd that you used to obtain these behaviors.

2. Make two Bode plots of the frequency response of the system:
   a. Set $K_d = 0$
   b. Set $K_d = \text{value you found that made the system critically damped.}$

To make each Bode plot you will plot:

- $20\log(\text{output amplitude/input amplitude})$ vs. input frequency (on a log scale), and
- phase shift vs. input frequency (on a log scale).

Last hour of lab

Controls team members need to stay in the lab and perform the following exercise.

1. Make the LED onboard the Seeeduino board blink at a rate of 1Hz while your motor follows a sinusoidal input (as in Part II) of 1Hz and an amplitude of 90 degrees.

This is also your time to work one-on-one with the TAs to get better at programming the microcontroller. If you have questions about programming this is your opportunity to receive feedback and instruction on its use.