MAE106 Laboratory Exercises
Lab # 4 - P-type computer control of a motor

Goals
Understand how to create a P-type velocity controller using a DC motor coupled with a rotary encoder.

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
As a mechanical and/or aerospace designer, you will sometimes want to control actuators on a machine, such as a robot, car, plane, or space vehicle, to make the machine move like you wish. The purpose of this laboratory exercise is to learn how to use a computer (i.e. an Arduino microcontroller) to control a motor. One of the major benefits of using a computer for control is that you can express the control law and control gains in software. Then, changing the control law just involves typing in new software code. This will be really useful for your final project. Note that an alternate way to set up controllers is with analog circuit elements, such as op-amps. Such controllers respond more quickly than a computer, but changing parameters requires changing resistors or capacitors. With the decreasing cost and increasing speed of computers, it’s more common now to implement controllers with computers.

The basic idea of computer-based control is to (see Fig. 1):

1. Electronically measure the system performance (In this lab, the system performance in which we are interested, i.e. the thing we are trying to control, is the angular velocity of the motor shaft. We will measure this with a rotary encoder)
2. Read this measurement into the computer (In this lab, you will read the pulses from the encoder into two of the digital inputs of the Arduino using 2 interrupt service routines (ISRs). If the output of your sensor were analog, you could read it into one of the Arduino's Analog-to-Digital (A/D) converters; see Fig. 1 bottom).

3. Calculate an appropriate control law in a program running on the computer.

4. Generate a voltage output to the motor amplifier.

---

**Figure 1.** Block diagram of hardware elements needed for computer control with Arduino. The top row shows how to read in a digital sensor, and the bottom row shows how to read in an analog sensor. ISR = Interrupt Service Routine. PWM = Pulse Width Modulation.

---

**Part I: Reading position and velocity using a rotary encoder**

In this part of the lab we will be reading the output from the Hall effect sensor (i.e. rotary encoder) that is coupled to your DC motor. These readings allow us to compute the position and velocity of the motor. We will be looking at this output both in the oscilloscope and the Serial Monitor of the Arduino IDE.

**Rotary encoder**

A rotary encoder is a device that converts the rotation of its shaft into digital voltage pulses. Two common types of rotary encoders are optical and Hall effect encoders. Consider an optical encoder first (Figure 2), as they are easier to understand. An optical rotary encoder is made of a rotating disk with slits cut in it, and a light emitter shining light through the disk toward a light sensor. The disk is connected to the shaft of the encoder. As the shaft rotates, the disk alternately blocks or allows light to enter the light sensor, depending on whether there is currently a slit in front of the light emitter or not. Thus, each time a slit passes the light emitter, the light sensor outputs a pulse of voltage. For an “incremental encoder” or “relative encoder” such as the one we are using in lab, the computer measures the amount of rotation by counting the pulses being sent by the encoder. Incremental encoders typically have two light emitters and two light sensors, with two tracks of slits, which allows
them to determine the direction of motion using “quadrature encoding” (see http://letsmakerobots.com/node/24031).

Figure 2. Optical encoder (image from: http://www.ni.com/white-paper/4672/en/)

**Hall effect sensor**

For the motor in this lab and your final project, the rotary encoder is a Hall effect encoder, which produces voltage pulses just like an optical encoder (Figure 3). The difference is that the pulses are produced when a magnet on the shaft passes close to a Hall effect sensor. A Hall effect sensor make use of the Lorentz force law. When a magnetic field comes close to the sensor, it creates a Lorentz force on the charges flowing through the sensor, pushing them to one side of the sensor, thereby causing a small voltage difference in the direction perpendicular to current flow (see Figure 2). This voltage changes with the proximity and polarity of the magnetic field, so by putting magnets on a shaft, turning the shaft will cause the Hall effect sensor to output voltage pulses.

Figure 3. Hall effect sensor (image from http://www.akm.com/akm/en/product/add/magnetic_sensors/0029/)

For a detailed tutorial on using a Hall effect sensor see:
http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

For a detailed tutorial on using a quadrature encoder see:
Interrupts

Interrupts are hardware signals that controls designers use to temporarily stop a running program and allow a special program to run. Many computers, including the Arduino, have dedicated hardware that can continuously monitor a voltage on a pre-defined input line, called an “interrupt line” (your Seeeduino has 2 interrupt lines). When the voltage on the interrupt input line changes, this hardware then immediately tells the computer to stop what it is doing and run the “Interrupt Service Routine” (ISR), which is a piece of code that does something in response to the “interruption” indicated on the interrupt line. ISR's are useful when you have a device connected to a computer that provides rapidly changing information at random times, such as rotary encoders. Rather than having your main program periodically check the encoder (in which case it might miss a pulse if it isn't checking often enough), you set up an ISR to respond to the encoder immediately when it provides new information.

Reading from the encoder

The vendor of the DC motor provides instructions on using the encoder that is coupled to your DC motor (see section "Using the Encoder" in: [https://www.pololu.com/product/1442](https://www.pololu.com/product/1442)).

First, we will connect the encoder to the breadboard in order to look at its output with the oscilloscope. There are six wires coming out of the encoder. The Red and Black wires are used to power the motor and we will not be using those for this part. Wire the remaining four wires as follows:

- Green wire: to GND from the trainer kit,
- Blue wire: to 5V from the trainer kit,
- Yellow: to CH1 in the oscilloscope,
- White: to CH2 in the oscilloscope.

If you now spin the motor's shaft by hand, you should be able to see the pulses in the oscilloscope (similar to those in Figure 4). Make sure that you understand what happens when you spin the motor in both directions.

Once you are sure that the encoder is connected properly, you can now connect the Red and Black wires to the power supply. Start with a voltage of 3V and work your way up to 6V, what happens to the length of the pulses as you increase the speed of the motor?
Practical Exam I:
In the section "Using the Encoder" of the Pololu page there is an image showing the output of encoders A and B in an oscilloscope (see below).

Show your TA that you can replicate the output shown in Figure 5 in the oscilloscope as you spin the motor. Also, show your TA that you can compute the velocity of the motor using the microcontroller by printing it to the Serial Monitor of the Arduino IDE.
**Part II: P-type control of the velocity of the DC motor**

Now that we can read the position of the motor, we will use it to compute the velocity of the motor and use it as a feedback signal to control the motor's velocity.

Use the pseudocode that you created for problem 1 in the pre-lab assignment and the code provided here:

http://gram.eng.uci.edu/~dreiken/MAE106/labs/Lab4Pvelocity.ino

to control the velocity of the motor.

**Do not simply copy and paste the code.** You will need to understand how the code works by going through it and relating them to the steps mentioned below. Note that there are many lines that have been commented to ensure that you go through the code and understand each line.

Make sure you understand wherer your getVelocity() function should be implemented.

To implement the P-type (proportional-type) velocity feedback controller. The program needs to:

1. read the signals from the encoders,
2. compute the velocity of the motor,
3. calculate the control law, and
4. send the control signal to the motor driver.

Since we already know how to compute the velocity of the motor, the next step is to implement the control law and send the appropriate signal to the motor driver.

The equation that describes the control law is:

\[ u = K_p e \]

where:

\[ e = error = \omega - \omega_d, \]

\[ \omega = motor\ velocity, \]

\[ \omega_d = desired\ motor\ velocity, \]

\[ \tau = motor\ torque\ (control\ signal\ to\ the\ motor\ driver) \]

The basic idea of this controller is to measure the velocity errors of the motor shaft, and then to apply torques proportional (hence the name proportional-type controller) to those errors that correct those errors, so that the motor shaft turns to
the desired angle. Note that if the error is zero, the correction torque to be applied is zero; if the error is large, the correction torque to be applied is large.

You will wire the motor and its encoder as shown in Figure 6. Note that we have provided a diagram using a generic motor driver, as such, we expect that you will take the time to understand where you should be connecting the wires for the encoder and the motor output.

![Figure 6. Motor and encoder connections to motor driver](image)

Note that you must follow the pin assignment exactly as described above because digital pins 2 and 3 are the pins connected to the ATmega328 external interrupts (see http://www.seeedstudio.com/wiki/Seeeduino_v3.0).

Once the circuit is wired you can now start tuning the code and the controller to get the motor to spin at a specific velocity. For this week's write-up you will need to gather the response of your system for at least 5 proportional gains ranging in value from 5 to 20 (you must study the code to see where the controller parameters are set).

**Practical Exam II:**
Show your TA that you can measure the velocity of the motor and that you can control the velocity of the motor so that it spins at 60rpm. Prove that the motor is running at this speed by printing the velocity of the motor on the Serial Monitor of the Arduino IDE. Be ready to answer what happens to the controller when there is a disturbance (i.e. what happens to $u$ when you apply a load to the shaft?).
Pre-lab assignment

Answer the following questions and turn them in to your TA when you come into lab. Make sure to label your answers.

Q1. The vendor of the DC motor provides instructions on using the encoder that is coupled to your DC motor (see section "Using the Encoder" in: https://www.pololu.com/product/1442).

Write the pseudocode for a function called “getVelocity()” that computes the velocity (both magnitude and direction) of the DC motor. This function will be called inside the loop() function of the arduino and should have the following input:

- current_time
- previous_time
- current_position
- previous_position

That is, the function should be of the form:

```
getVelocity(current_time,previous_time,current_position,previous_position)
```

What should this function return?

Q2. Watch the video title “Encoder demo” located at:

http://gram.eng.uci.edu/~dreinken/MAE106/res/motorEncoderDemoMAE106.mp4

and answer the following questions:

a) Why do you need two encoders to determine the direction at which the motor is spinning?

b) What happens to the length of the pulse as the motor speeds up?

Q3. Based on the control law given in Part II, imagine that the current speed of the motor is 10rpm and the desired speed is 17rpm. For a gain value, K, of 3, what is the magnitude of the control input to the motor driver?

Write-Up

Create a plot of the response of your system to at least 5 different proportional gains with values between 5 and 20. Also, answer the following questions:

1. What happens as you increase the gain?
2. Is your system able to perfectly follow the desired velocity?
**Last hour of lab**

Electrical fabrication and controls team members need to stay in the lab and perform the following exercise.

1. Wire a potentiometer to one of the microcontroller’s analog inputs.
2. Use the signal from the potentiometer to control the velocity of the motor using the closed-loop routine that you implemented in lab.

*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.*