ME 6705 / AE 6705 - Introduction to Mechatronics

Lab Assignment 6

Feedback control for DC motors

1 Objective

The objective of this lab is to learn how to implement a closed loop PID controller for a DC motor with encoder. You will characterize the step response and transfer function using experimental data to determine your PID gains.

2 Setup

This lab requires the following:

- A National Instruments myRIO microcontroller
- A breadboard
- A battery holder with 4 AA batteries
- A DC motor with encoder
- 4 x Diodes (1N5819 or similar)
- 1 H-Bridge motor driver (L293DNE or similar)
- You need to provide a custom load to attach to your motor. This can be 3D printed, laser-cut, etc.

3 Problem Statement

In this lab, you will build an electromechanical system using a DC motor with an encoder. This lab is divided into steps, which must be completed in order:

1. Build the motor driver circuit (Section 7)
2. Connect a load to the DC motor (Section 9)

3. Determine encoder steps with a custom LabVIEW program (Section 10)
4. Determine the motor model parameters (Section 11)
5. Calculate your PID values using the MATLAB script (Section 12)
6. Fine-tune your PID values (Section 13)
7. Create a custom LabVIEW program to set motor position (Section 14)

4 Deliverables and Grading

This lab is worth 200 points. This lab is done in a team of two. Demos must occur in-person during office or demo hours. You must demonstrate the following to the TAs or the instructor to the due date to receive credit:

1. Section 9 - Show your custom load to attach to the motor. (10 points)
2. Section 10 - Demonstrate your custom LabVIEW program to determine the number of encoder steps. (50 points)
3. Section 11 - Show your K_m and T_m values, and your measurement plots that justify those values. (20 points)
4. Section 14 - Demonstrate your custom LabVIEW program that sets motor position. (120 points)
5. Submit your code to Canvas. Your code should be a zip file of your LabVIEW project. This code will not be graded, but no submission will result in a zero for the lab.
6. Submit your peer review evaluation to Canvas.

5 The LabVIEW program specifications

This lab requires LabVIEW 2021 SP1 32-bit. You must use the Lab 6 LabVIEW template provided in Canvas.

6 Hardware

6.0.1 Brushed DC Motor

The brushed DC motor is a type of motor that operates on DC voltage and current. It is bidirectional, meaning if the voltage across the supply terminals are inverted, the motor turns in the opposite direction. An H-bridge helps in achieving this functionality. The DC motor (with gearbox and encoder already installed) used in this lab are manufactured by Digilent, SKU 290-006. Specifications for this motor can be found at: https://digilent.com/shop/dc-motor-gearbox-1-19-gear-ratio-custom-12v-motor-designed-for-digilent-robot-kits/. This motor is rated at 12V, with a stall torque at 2.2 kgf.cm, and a no-load speed of 789 rpm. Its stall current is 2.4A, and rated current is 620 mA. The Technical Datasheet for the motor is attached in Canvas. **You will be driving this motor using the 4-AA battery pack from prior labs, so the maximum voltage supplied to the motor will be 6V.**

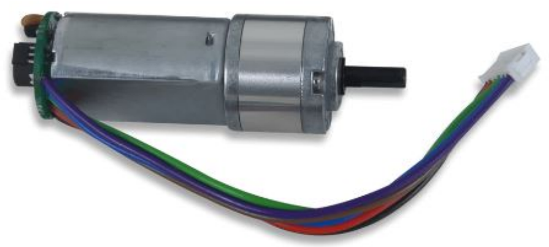

Figure 1: DC motor with encoder assembly (from Digilent)

For this lab assignment, you will connect the two terminals (Red and Black wires) of the DC motor to the H-bridge motor driver circuit. The other 4 pins of the motor are encoder signals, which will be connected to power, ground, and digital inputs on the myRIO.

6.0.2 Encoder

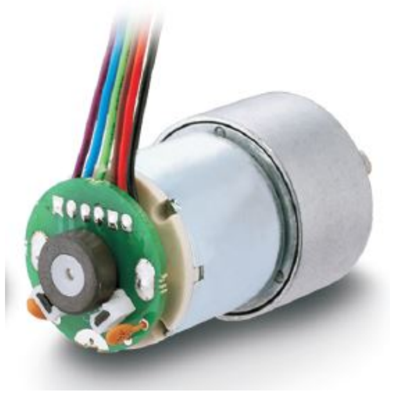

Figure 2: Encoder

An encoder is a sensor with digital outputs that relate to rotational speed. The DC motor used in this lab is equipped with an encoder board that senses the rotation of the magnetic disc and provides counts per revolution of the motor shaft. In the first step (Section 7), you will determine what that resolution is. The encoder board wires are as follows:

Color	Function
Black	-Motor
Red	+Motor
Brown	Hall Sensor Vcc
Green	Hall Sensor GND
Blue	Hall Sensor A Vout
Purple	Hall Sensor B Vout

6.0.3 H-bridge IC

The H-bridge IC that we use for this lab is the TI L293DNE. L293DNE is a quadruple half H-bridge driver IC. It has four half H-bridges that can be used either independently for motors with unidirectional applications, or in pairs for motors with bidirectional applications, such as our motor. The datasheet for L293DNE can be found on Canvas.

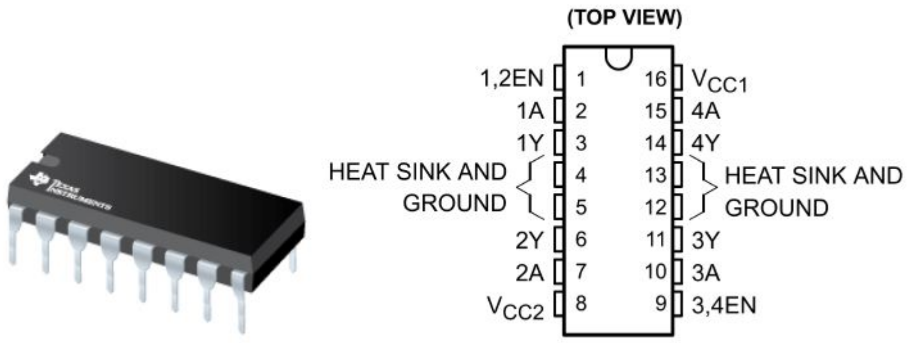

Figure 3: L293DNE pinout (from Texas Instruments)

We will be using channels 1 and 2 of the H-bridge driver IC as a pair to control the motor. This setup forms a full H-bridge which enables bidirectional operation of the DC motor.

Figure 4 shows the possible wiring configurations that can be used with the L293DNE Hbridge IC. Note that in the figure, the interfaces at ports 3 and 4 are shown for unidirectional operation of two motors. The left side of the diagram (ports 1 and 2) shows the wiring schematic for bidirectional operation of a motor.

Note: For this lab assignment, we will be using the **bidirectional configuration** in order to have bidirectional control of the DC motor.

Note that the L293DNE IC comes in a PDIP package, and has an identifier to help you locate Pin 1 on the chip: there is a Half Circle or Notch on the end of the chip, Pin 1 is to the left of the notch (Figure 3). IC pins are numbered in a counter clockwise fashion from Pin 1. The pins of the L293DNE H-bridge IC are described in the table below.

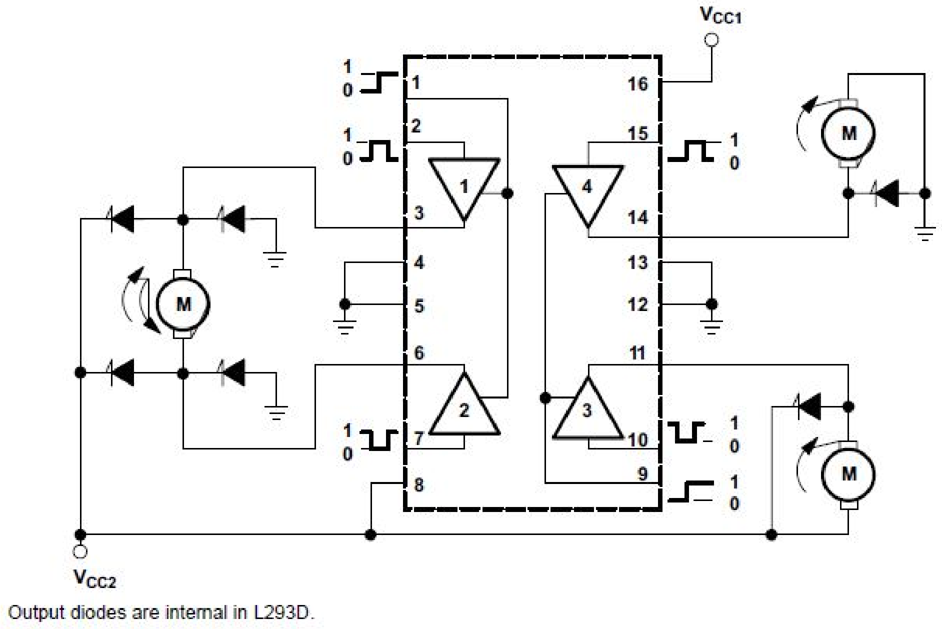

Figure 4: L293DNE functional block diagram (from Texas Instruments)

Pin	Pin Name	Pin Function
1	1,2EN	Enable signal for driver channels 1 and 2 (active high input)
2	1A	Input PWM signal for driver channel 1
3	1Y	Output signal to the motor for driver channel
4	Heat Sink/GND	Connect to Ground
5	Heat Sink/GND	Connect to Ground
6	2Y	Output signal to the motor for driver channel
7	2A	Input PWM signal for driver channel
8	VCC2	Power supply for motors (connect to +6V)
9	3,4EN	Enable signal for driver channels 3 and 4 (active high input)
10	3A	Input PWM signal for driver channel 3
11	3Y	Output signal for driver channel 3
12	Heat Sink/GND	Connect to Ground
13	Heat Sink/GND	Connect to Ground
14	4Y	Output signal for driver channel 4
15	4A	Input PWM signal for driver channel 4
16	VCC1	Power supply for internal logic (connect to 5V)

To understand exactly what the input control PWM signals do, refer to Figure 4. Setting pin 2 of L293DNE high and simultaneously setting pin 7 of L293DNE low will turn the motor (attached to the output pins 3 and 6) in one direction. To turn the motor in the other direction, pin 2 should be low and pin 7 of L293DNE should be high. Note that these input pins can also accept PWM signals from the myRIO. In order to operate the motor in either direction, a PWM signal should be applied to one of these two pins while maintaining the other pin low. Setting the enable signals to high will enable the associated driver channels. To make sure that the driver channels are enabled, we manually connect the enable pins to 3.3V, as will be shown in the circuit schematic in Figure 6.

6.0.4 Diodes

Diodes are semiconductor devices used to control the direction of current flow. They are used often in mechatronic devices. In this lab, diodes will be used as flyback diodes for the motor driver, just like in Lab 2. Flyback diodes provide alternative path for current in motor coil and mitigates arcing/excessive current through transistors, to avoid damaging the transistors.

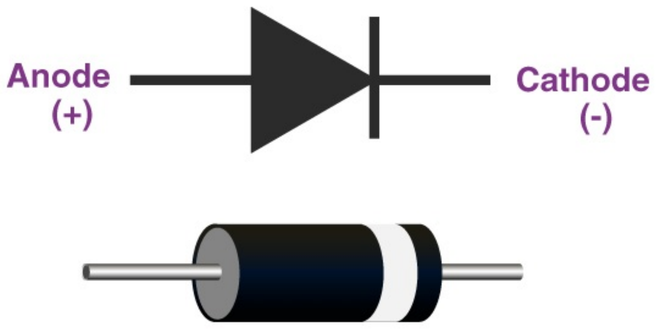

Figure 5: The diode circuit symbol, and the physical shape

7 Circuit Schematic

Figure 6 shows the circuit we use to interface the brushed DC motor with the myRIO. The motor is powered via four AA batteries, which provide a voltage in the range of 4.8V to 6V. We recommend to connect the positive wire of the battery holder to the (+) power rail of the breadboard, and then use the power rail for all the pins connected to +6V. Similarly, the negative wire of the battery holder and the GND pin of the myRIO can be connected to the negative power rail of the breadboard, so that all the pins connected to GND can share that power rail. The +5V required for the VCC1 pin of L293DNE (pin 16) can be directly taken from the 5V pin of the myRIO using a jumper wire. The flyback diodes provide paths for current to flow when the drivers are switched off. Pay extra attention to the direction of the diodes when implementing them on the breadboard.

Note: Remember to connect a common ground between the myRIO and the driver circuit!

Warning: While driving the motor, check the temperature of the H-bridge occasionally. If it is getting hot, disconnect immediately and debug the circuit.

Warning: A DC motor should never receive PWM signals of opposite spin directions at the same time. This could potentially cause the motor to burn out. Therefore, before you connect the motor terminals to the circuit, verify the functionality of using the digital out pins as ON/OFF switches to ensure that the motor receives only one PWM signal at a time. Only when you are confident of the functionality of your code should you connect the terminals of the DC motor.

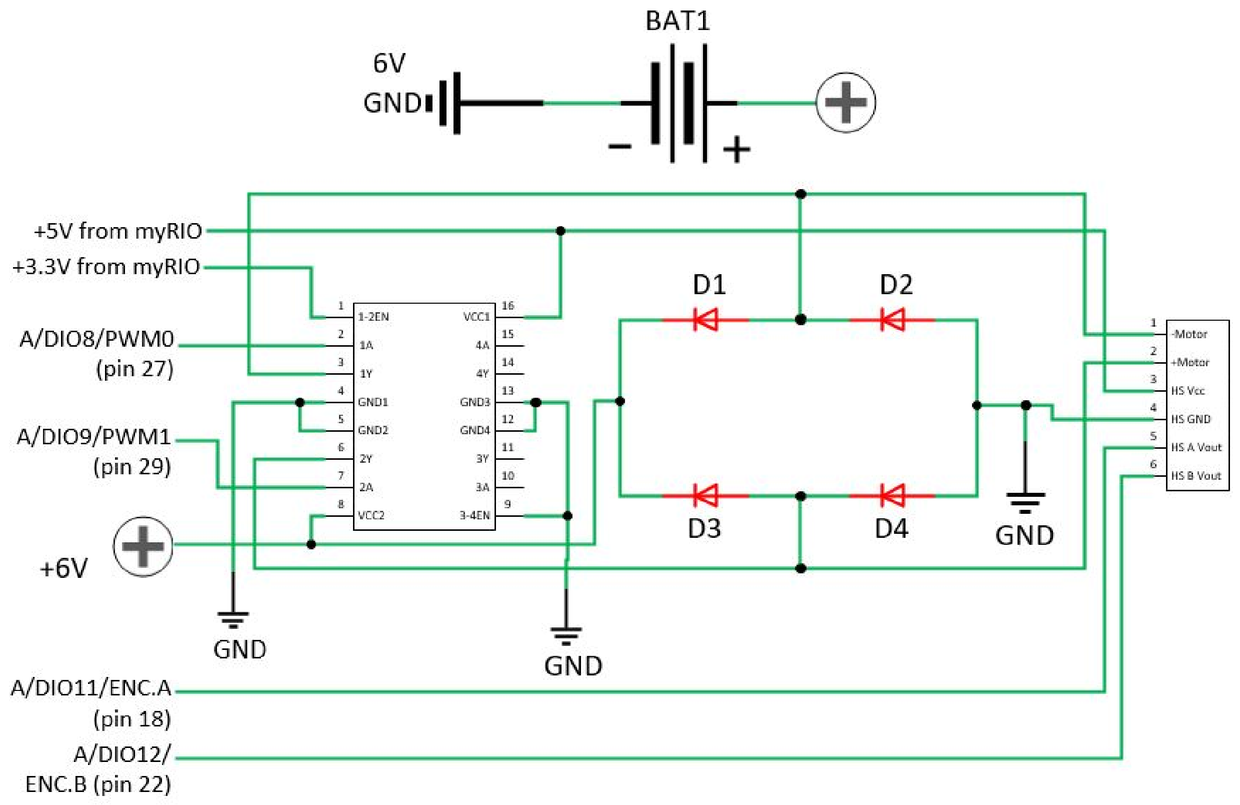

Figure 6: Circuit schematic for interfacing stepper motor to myRIO

8 LabVIEW program details

Open up "Main.vi" in the project "Lab 6.lvproj". This project uses the LabVIEW Control and Simulation package, already pre-installed on the myRIO. In the main VI, you have 4 tabs to work with. Below are descriptions of each tab.

8.1 Find Encoder Step Count

In this tab, you will write a custom LabVIEW code to experimentally measure the encoder step count. See section 10 for details.

8.2 Step Test

In this tab, you can provide a voltage to your motor and see the measured rotational velocity from the encoder. Your plot time can be altered with ”Step File Time (s)”. To save data, right click on the plot, and go to ”Export”. Copy the data, then paste into Excel. You will use this data to calculate your motor model parameters.

8.3 Sine Test

In this tab, you can determine the motor model parameters in the frequency domain (see lecture slide on Inputs and Outputs in the Frequency Domain). This method will provide you a more accurate model for the motor time constant T_m. Use of this tab is **not required** for this lab, it is completely optional.

8.4 Pulse Controller

In this tab, you can prescribe a square wave, or pulse, to your motor. This will allow you to look at (and collect data for) controller performance over many cycles. You will also set your preliminary K_p, K_d, and K_i values based on the MATLAB program. To save data, right click on the plot, and go to ”Export”. Copy the data, then paste into Excel. You will use this data to ”fine-tune” your PID controller using the trial-and-error method as described in lecture.

8.5 Set Position

In this tab, you will write your own program that will use your final, tuned PID values. Your motor will turn to an angle, as specified by the control dial.

9 Connect a load to the DC motor

Go make a load that can be attached to the 4mm D-shaft on the DC motor. This can be any load, such as a 3D-printed wheel. You will need a way to keep track of 360 degrees of rotation, for the next section ”Determine Encoder Steps”.

10 Determine encoder steps

Read the encoder data sheet. Generate a hypothesis for the step count of this motor (motor diameter 20.3 mm). Using the tab ”Find Encoder Step Count” in the Lab 6 LabVIEW program, create custom LabVIEW code that will help you verify your hypothesis by answering the following:

- How many encoder steps are there for every rotation of the gearbox shaft? This is the D-shaft where you would attach loads. Determine this by creating a custom LabVIEW code.

11 Determining the motor model parameters K_m, T_m

As discussed in class, the motor can be represented with the first order transfer function between angular velocity and input voltage in the Laplace domain:

$$\frac{\Omega(s)}{V_{in}(s)} = \frac{K_m}{T_m s + 1} \tag{1}$$

In order to use this simplified equation, our assumptions are that friction effects are Newtonian, armature inductance is negligible, etc. The two motor constants we look to solve for are the **motor gain** K_m and the **motor time** T_m constants. These values will be measured from empirical data through evaluation of a step response. Applying a step response ($E_i(s) = \frac{V_{in}}{s}$) to Equation 1 will result with:

$$\Omega(s) = \frac{K_m}{T_m s + 1}\left(\frac{V_{in}}{s}\right) \tag{2}$$

We will then apply an inverse Laplace transform $\mathscr{L}^{-1}(\Omega(s))$ to Equation 2 to revert back to the time domain:

$$\omega(t) = V_{in}K_m\left(1 - \exp\left(-\frac{t}{T_m}\right)\right) \tag{3}$$

Now, we will evaluate Equation 3 experimentally.

11.1 Determining the motor gain constant K_m

If we evaluate Equation 3 at time $t = \infty$ (when the motor reaches a steady-state rotational velocity), the term $\exp(-\infty)$ reduces to 1. Therefore:

$$\omega(t) = V_{in}K_m \tag{4}$$

Perform the following procedure using the LabVIEW template to find K_m:

1. Set a voltage V_{in} and let the motor spin until it reaches a steady state rotational velocity ω_{ss}
2. Measure the steady state rotational velocity ω_{ss}
3. Increment the input voltage V_{in}. Repeat steps 1 and 2 to form a ω_{ss} vs V_{in} plot. Ensure you have at least 6 data points (the more data points, the more accurate your regression)
4. Perform a linear fit on your ω_{ss} vs V_{in} plot. The slope of the curve will be your motor gain K_m.

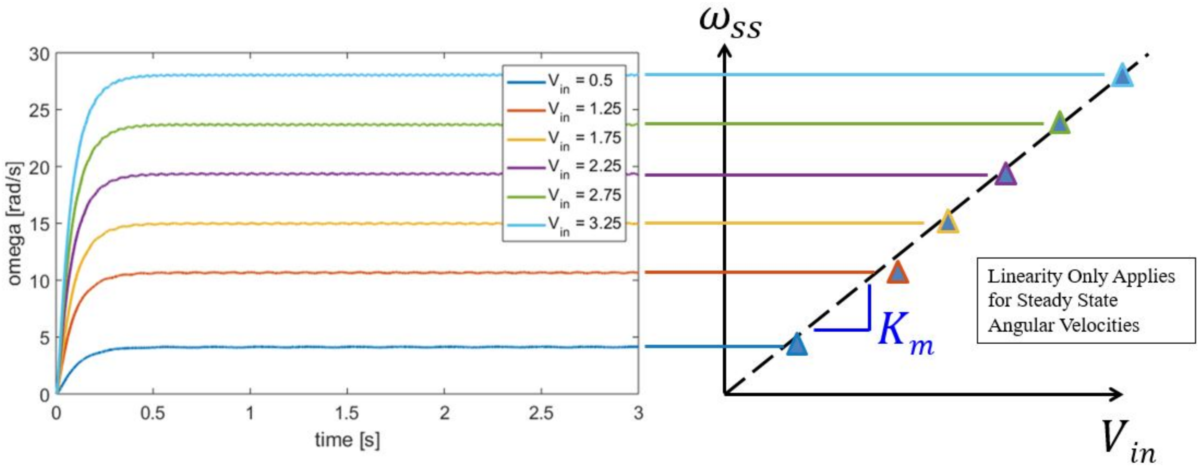

Figure 7: A representation of the process to find K_m

11.2 Determining the motor time constant T_m

If we evaluate Equation 3 at time $t = T_m$, or $\frac{t}{T_m} = 1$, the term $\exp(-1)$ reduces to the value 0.632. Therefore:

$$\omega^*_{t=T_m} = V_{in}K_m(0.632) \tag{5}$$

Where ω^* is the rotational velocity at time $t = T_m$. The motor time constant T_m is independent of voltage, and can be found using the following procedure:

1. For each voltage V_{in} vs rotational velocity ω_{ss} curve measured, find T_m using Equation 5.
2. Average all values of T_m from step 1.

12 Calculate your PID values using the MATLAB script

Using your measured K_m and T_m values, calculate your PID gains (K_p, K_d, K_i) using the MATLAB script "PID-gains-calculator.m", which you can find in Canvas. You will use the following additional inputs:

- M_p is the desired max overshoot (see slide 26 from Lecture 14 for visual)
- t_s is the desired settling time (see slide 26 from Lecture 14 for visual)
- N is the iteration of the program. Set $N = 1$.

13 Fine-tune your PID values

In main.vi, using the tab "Pulse Controller", observe how your PID gains control your system. Using trial-and-error, adjust your PID gains until you have the desired response of your system.

14 Create a custom LabVIEW program to set motor position

In main.vi, navigate to the tab ”Set Position”. In this tab, you will create a custom program that uses your final, tuned PID gains. Your program should do the following:

- Use the driver gauge to control the motor position
- The driver gauge should change the motor position on-demand (i.e. when you click and drag the arrow, the motor should turn with the arrow)
- Plot the desired and actual position of the motor (note: the position should be in degrees)
- Input the actual position measured (from the encoder) into the Encoder gauge. As you move the driver gauge, you will see the encoder gauge needle move with lag behind you.