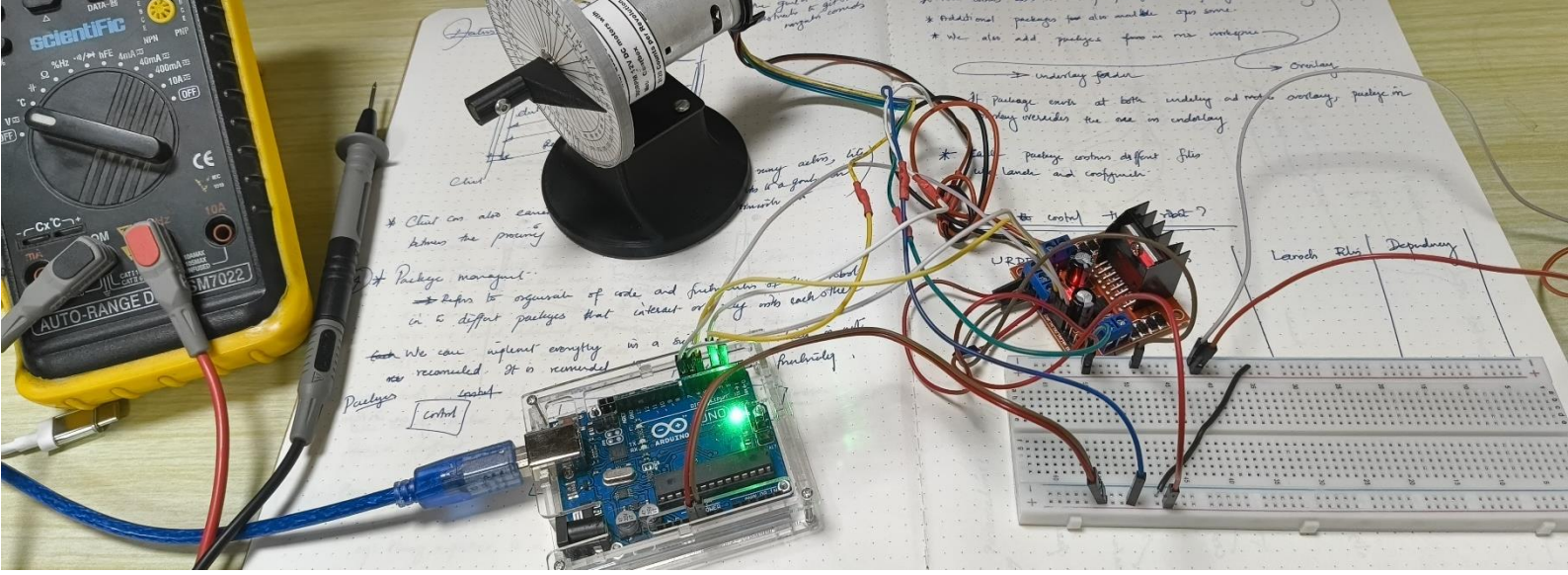


Experiment 4

Actuators: DC motor with encoder

Position and velocity control of a DC motor with encoder

Precautions

1. As we are introducing external voltage, turn on the circuit only after verifying it with a TA. Please double check the connections or ask in case of doubt. Do not blow up the electronic components.
2. Please upload all your files clearly name in the format ***ME381_E4_Name_RollNumber_GroupNumber_.pdf***.
3. **Please tidy up after your work. Return the equipment and clean your workspace.**

Materials Required

Please ensure that the kit provided to you have the following components

1. Arduino Uno
2. Motor Driver L298N
3. DC motor with its datasheet
4. Power supply module/battery
5. Connecting wires and breadboard

Parts of the Experiment

1. Reading encoder signal from the Arduino (Time: 15 mins)
2. Calculating the velocity of the motor in rpm (Time: 30 mins)
3. Controlling the motor using PID techniques (Time: 40 mins)

Setting up the Serial Plotter

See page 10 before starting the experiment

Part 1: Reading the encoder motion in Arduino IDE

1. Make the connections to the encoder from the Arduino as shown in the figure below.
    - Encoder A output (Yellow) → DO pin 2
    - Encoder B output (White)→ DO pin 3
    - Encoder ground (Green) → Arduino GND
    - Encoder VCC (Blue) → Arduino 5V

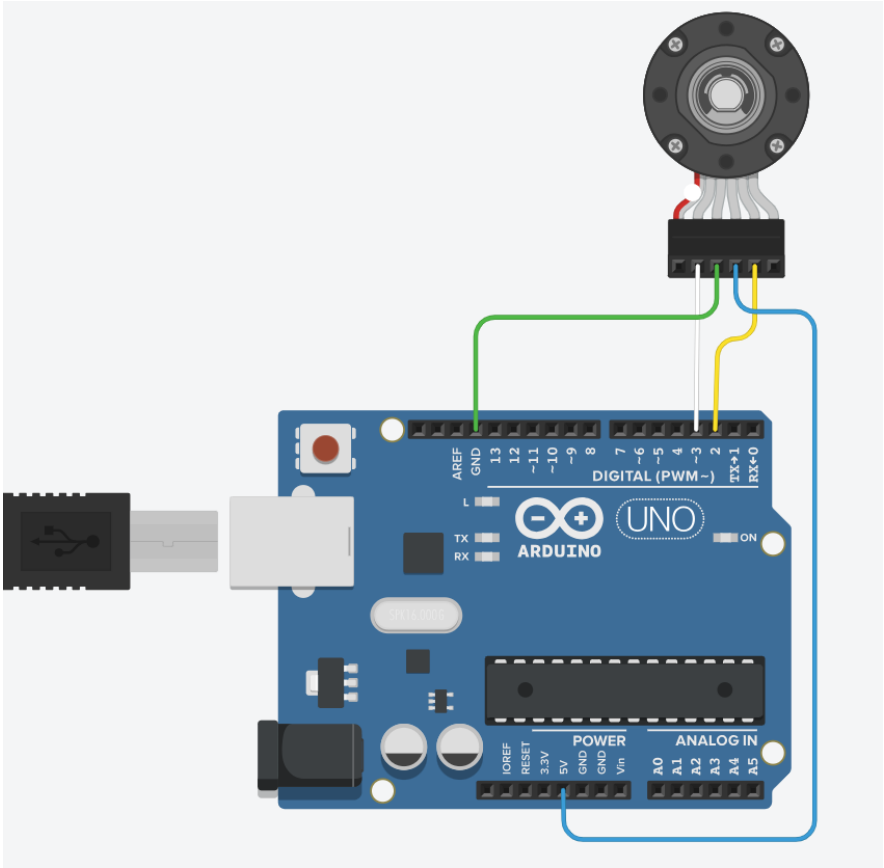


Figure: Connections to the Arduino for taking encoder measurements

2. Use the Part 1 code in Arduino IDE. Upload the code to the microcontroller.

3. Open the rear cap of the DC motor and rotate the encoder as shown in CW and CCW directions. Open the serial plotter to view the output (Note that the output in code is multiplied by 5 and 10 for better visibility).

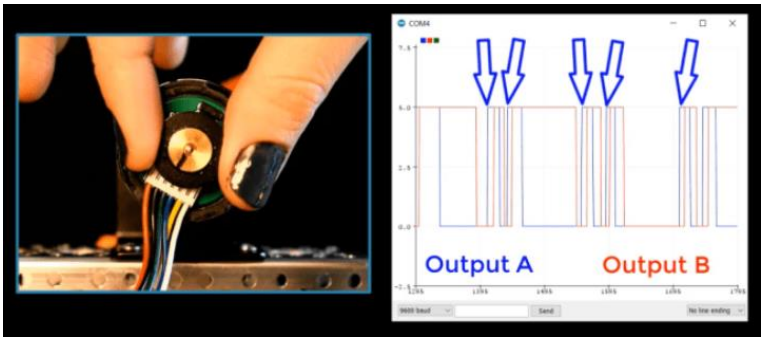


Figure: Rotate the encoder and observe the serial plotter

4. Rotate the encoder in CW and CCW directions.

For Report

Take the photograph of the output from the serial plotter.

1. How many pulses of A and B are seen for one rotation of the encoder?
2. When the encoder is seen from the back (as shown above) and rotated CW, which is true?
    a. Blue leads Red
    b. Red leads Blue

Part 2: Velocity Measurement in rpm

1. Connect the motor driver and power supply as below and **remove the regulator enabling jumper and keep it safely in the box**
    - Motor terminal 1 → Motor driver output 1
    - Motor terminal 2 → Motor driver output 2
    - Power supply → Motor driver Vin/GND
    - Motor driver GND → Arduino GND
    - Motor driver PWMA input (ENA) → Arduino pin 5 [PWM]
    - Motor driver IN1 input → Arduino pin 7
    - Motor driver IN2 input →Arduino pin 6
    - Motor 5V → Arduino 5V

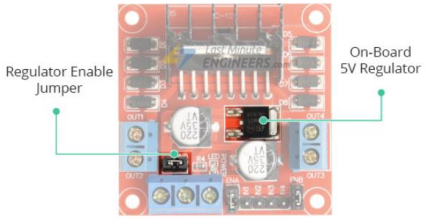


Figure: L298N motor driver

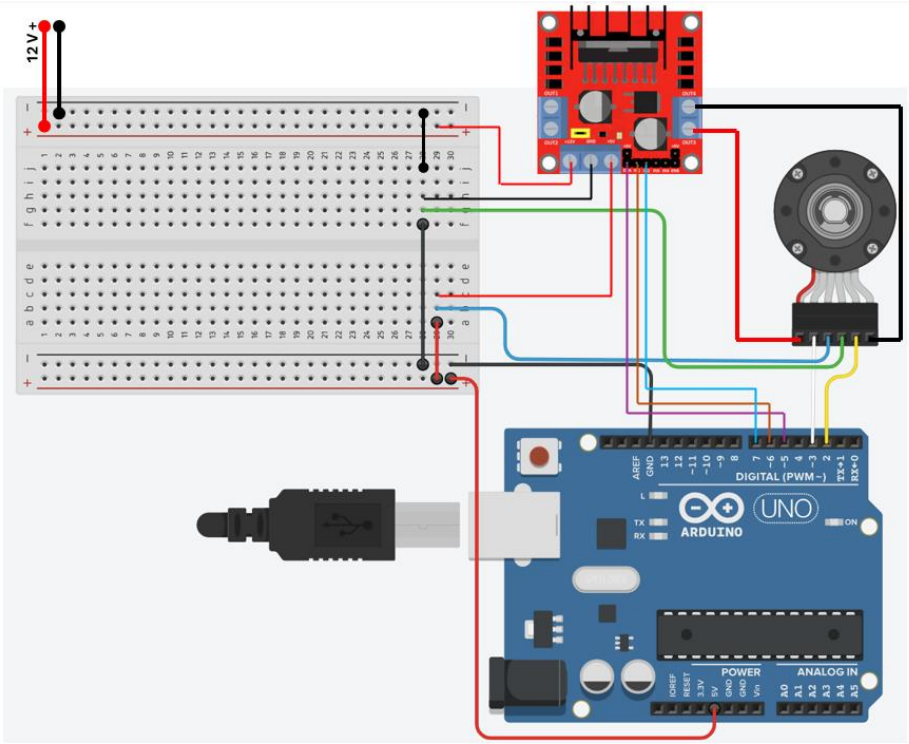


Figure: Schematic for the connections

2. Use the DC Motor Velocity Measurement code

   Explanation of the code:

   - Instead of reading the signal from the digital pin, we need to be able to find the position of the rotor from the encoder motion. For this, everytime there is an change in encoder position, the same needs to be added or substracted to the previous position value.
   - For this, we use the interrupt function. Every time there is an interrupt over the pin (a CHANGE in this case), a particular function is run, which in the code provided is readEncoder function.
   - An additional library ATOMIC_BLOCK can be used to avoid misreading of the variable 'posi' as it is being read by two functions, in loop and in the interrupt operation.
   - The function setMotor is used to drive the motor. The functions takes the input as direction, PWM pin, PWM value to be written to the pin, digital input pins for direction control.
   - The function filterVelocity is used to filter the velocity calculated from encoder measurements. As encoder measurements are discrete in nature, switching between discrete states can create high frequency signals. We use a low pass filter to avoid these high frequencies.

For Report

Provide the input in the setMotor function and observe the output velocity. Plot the output in the report (Question 2) and join the lines for PWM values 0, 25, 75, 50, 100, 125, 150,175, 200, 225, 255.

Part 3: PID Control of the motor

Moving from Velocity Control to Position Control

Initially, we used PWM (Pulse Width Modulation) to control the motor's speed. But what if we want the motor to reach a specific position instead of just watching the motor spin at some rpm? In this scenario, we need to continuously measure the motor's current position and compare it to the target position. The difference between these two is called the error.

Applying PWM Based on Error: The Proportional term

To reduce this error, we adjust the PWM value. If the error is large, meaning the motor is far from the target, we apply a higher PWM value to move the motor faster. If the error is small, indicating the motor is close to the target, we apply a lower PWM value. However, because the PWM is directly proportional to the error, the motor never perfectly reaches the target without some error remaining. This means that in order to generate some PWM, there should always be an error and the motor oscillates and settles around the target value unable to reduce the error below a certain value.

Smoothing the Response: The Derivative Term

To reduce these oscillations, we introduce the derivative term. This term looks at how quickly the position is changing and adjusts the PWM value accordingly. It's like adding a damper in a spring-mass system, which helps to smooth out the movement and reduce the back-and-forth oscillations of the motor.

Eliminating Steady-State Error: The Integral Term

Even after reducing the oscillations, there might still be a small error left, because the proportional term always needs some error to generate a control signal. To remove this error, we introduce the integral term. The integral term sums up all the past errors and applies a correction based on the accumulated error, pushing the motor to the exact target position.

Applying PID to Velocity Control

If we use this same PID control method to maintain a specific speed instead of position, it's called velocity control. Here, the system would monitor the speed, calculate the error, and adjust the PWM using the proportional, integral, and derivative terms to keep the motor running at the desired speed.

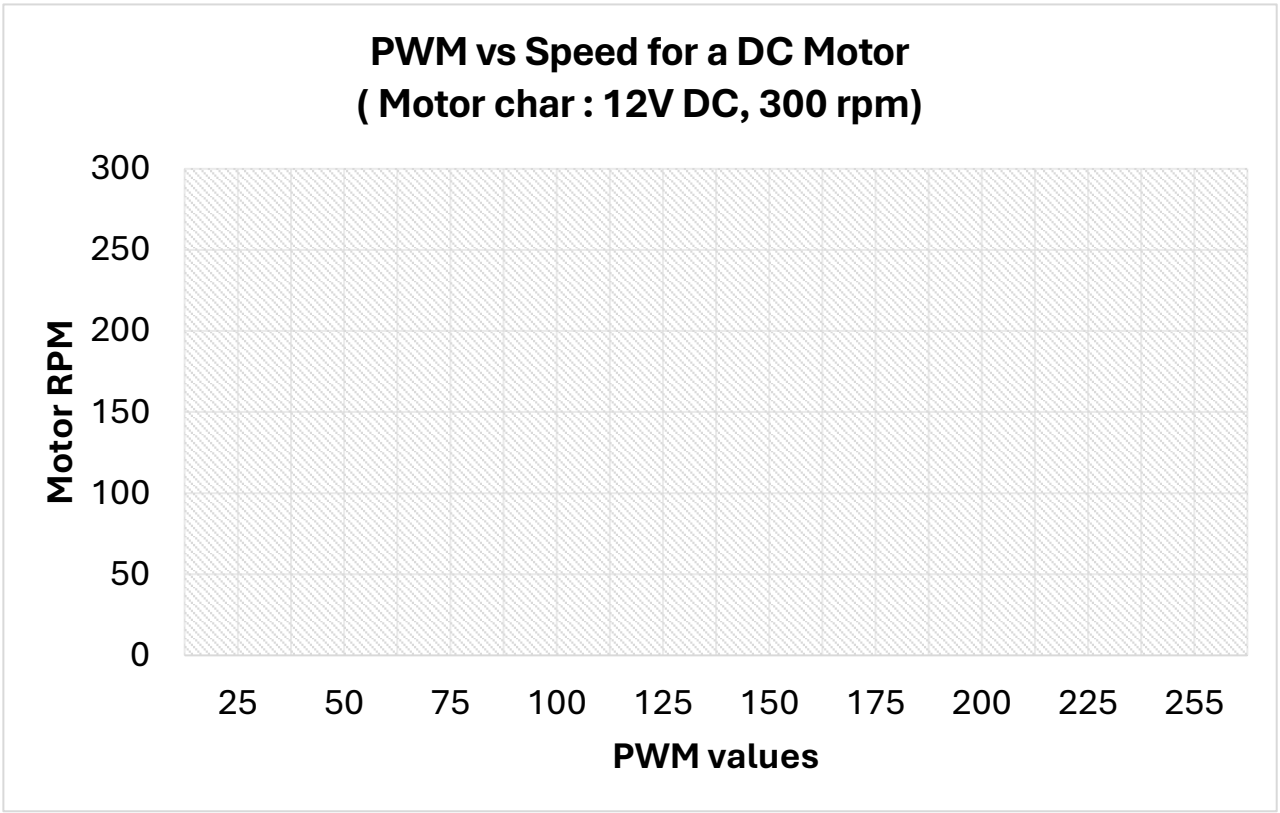
1. Aim of this experiment is to provide a target position and velocity of the motor that can be tracked.
2. Use the codes for position and velocity control. Change the PID values in both codes as per the table given in report and observe the curve on the serial plotter.
3. Set a low target velocity like 10 RPM. Check the response curve with and without using filtered velocity (lines 93 and 94 in the code). Observe the motion of the motor in the two cases as well.

For Report

Fill the table with the generated PID values for each target

Lab Report (Upload as ***ME381_E4_Name_RollNumber_GroupNumber_.pdf***)

1. Number of pulses of A and B per rotation is _____. When the encoder is seen from the back and rotated CW, _____ leads _____.
2.



3.

Target	P	I	D	Image of the response curve in serial plotter
Position Control	3	0	0	
	3	0	0.5	
	3	0.05	0.5	

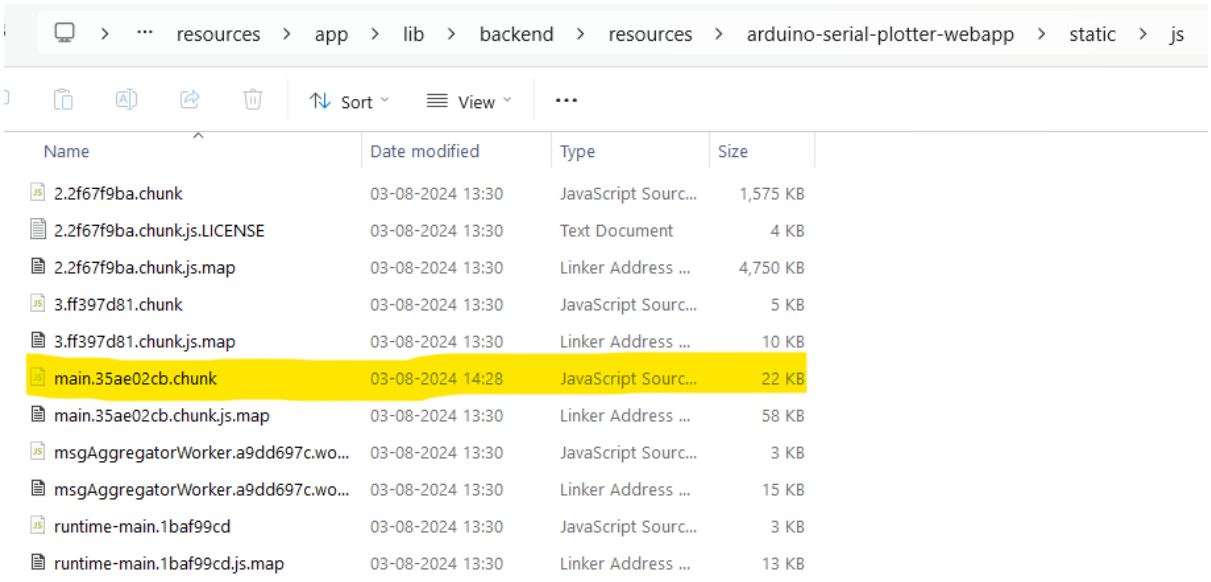
Velocity control	4	0	0	
	4	0	0.2	
	4	5	0.2	

Feedback on this experiment

Setting up the Arduino Serial Plotter environment

1. Since the serial plotter plots only for the last 50 instances, the PID curves could be difficult to understand. To increase the number of points plotted on the X axis, follow below instructions.

2. **Go to the corresponding path in your Arduino installation** "E:\Softwares\arduino-ide_nightly-20240630_Windows_64bit\resources\app\lib\backend\resources\arduino-serial-plotter-webapp\static\js\main.35ae02cb.chunk.js". PLEASE NOTE THAT THE PATH COULD BE DIFFERENT. SEARCH FOR THE FILENAME IN THE INSTALLATION FOLDER.



3. Open the file main.35ae02cb.chunk.js in notepad.

4. Search for “useState)(50)” and change 50 to 3000 and save the file.