Lab Worksheet 6

CSE461: Introduction to Robotics

Department of Computer Science and Engineering

Lab 06: Interfacing a 5-Pin Analog Joystick with Arduino for Directional Control of DC Motors via Motor Driver

I. Topic Overview

This lab introduces the fundamentals of interfacing an analog joystick with an Arduino to control the directional movement of a DC motor using a motor driver (e.g., L298N). Students will learn how to read analog signals from a 5-pin joystick module and interpret its movements to generate motor control signals. The Arduino will serve as the main controller, converting joystick input into appropriate motor commands, enabling real-time directional control. By the end of this lab, students will develop a functional system that demonstrates analog input handling and motor actuation through PWM and digital outputs.

II. Learning Outcome

After this lab, students will be able to:

1. Interface a 5-pin analog joystick with an Arduino for analog signal reading.
2. Design and implement motor driver circuits to control DC motors.
3. Develop Arduino code to interpret joystick input for real-time motor control using digital I/O.
4. Diagnose and troubleshoot hardware and code-level issues in analog input and motor driver interfacing.

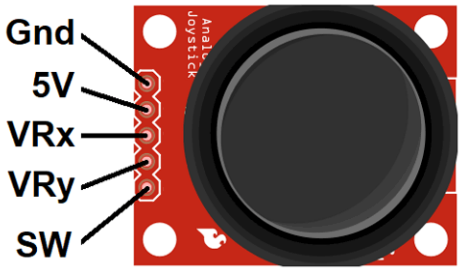
III. Materials

- Arduino Uno R3
- L298N Motor Driver Module
- 4-Pin Analog Joystick Module
- DC Motor (6V–9V, depending on power source)
- External Power Supply or Battery Pack (for motor)
- Breadboard
- Jumper Wires

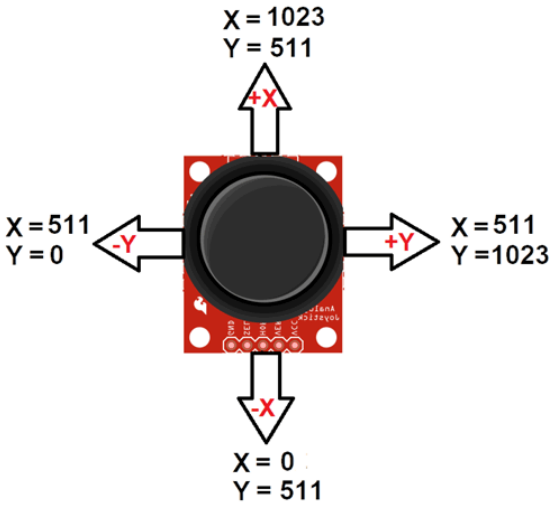
IV. Experiment 1 : Setting up the 5-pin Joystick module with Arduino

1. **Description of the components:**

The 5-pin thumb joystick module is a compact analog input device designed to sense two-dimensional directional movement. It functions similarly to analog sticks found in game controllers, offering X-axis and Y-axis control through **two potentiometers**. The module typically includes five pins:



- VCC – Power supply (3.3V or 5V)
- GND – Ground
- VRx – Analog output for X-axis movement
- VRy – Analog output for Y-axis movement
- SW – Digital output for the built-in push-button

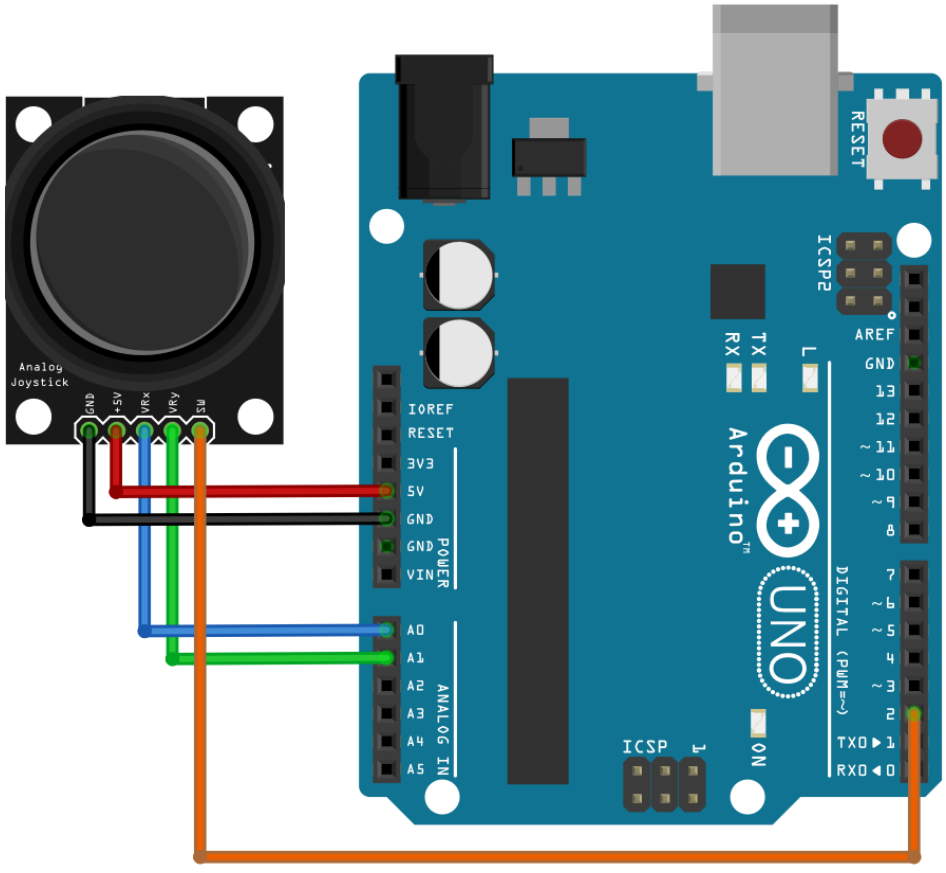


The analog outputs from VRx and VRy provide a 10-bit value ranging from 0 to 1023, where the center (neutral) position is typically around 512. Movement to either side from the center shifts the values toward 0 or 1023, depending on direction. The **SW** pin outputs a digital LOW (0V) when the joystick is pressed downward, and HIGH (VCC) when not pressed. This makes the joystick suitable for applications involving directional control, motion input, and button-based interaction in embedded systems.

2. **Setting up the circuit:**

- Connect **VCC** and **GND** of the joystick module to **5V** and **GND** on the Arduino.
- Connect **VRx and VRy** to Analog pin **A0** and **A1** of the Arduino respectively.
- Connect **SW** to **Digital pin 2** of the Arduino.

3. **Circuit diagram:**



4. **Arduino Code:**

```
#define SW 2   // Button pin
#define VRx A0   // X-axis
#define VRy A1   // Y-axis
#define threshold 200  // Sensitivity adjustment

void setup() {
  pinMode(SW, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  int x = analogRead(VRx);
  int y = analogRead(VRy);
  int button = digitalRead(SW);

  String direction = "Center";

  // X-axis directions
  if (x < 512 - threshold) direction = "Left";
  if (x > 512 + threshold) direction = "Right";

  // Y-axis directions (swap Back/Forward if reversed)
  if (y < 512 - threshold) direction = "Forward";
  if (y > 512 + threshold) direction = "Back";

  // Diagonal directions
  if (x < 512 - threshold && y > 512 + threshold) direction =
"Back-Left";
  if (x > 512 + threshold && y > 512 + threshold) direction =
"Back-Right";
  if (x < 512 - threshold && y < 512 - threshold) direction =
"Forward-Left";
  if (x > 512 + threshold && y < 512 - threshold) direction =
"Forward-Right";

  Serial.print("X: ");
  Serial.print(x);
  Serial.print("  Y: ");
  Serial.print(y);
  Serial.print("  Direction: ");
  Serial.print(direction);
  Serial.print("  Button: ");
  Serial.println(button ? "OFF" : "ON");

  delay(300);
}
```

How this works: *The joystick module uses two potentiometers—one for the X-axis and one for the Y-axis—to detect movement. Each potentiometer acts as a variable resistor that changes its resistance based on the position of the joystick. As the joystick is pushed in a direction, the wiper of the potentiometer moves, causing a change in the voltage output. This varying voltage is read by the Arduino's analog input pins and translated into a value between 0 and 1023. When the joystick is centered, the voltage is around*

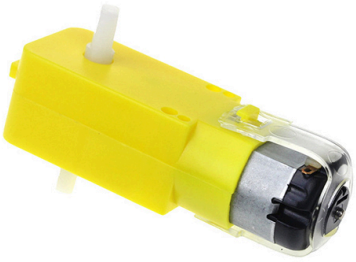
the midpoint (about 2.5V for a 5V supply), and pushing the joystick changes this voltage higher or lower depending on direction.

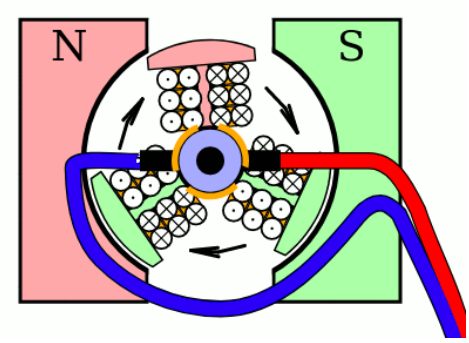
V. Experiment 2 : Setting up a DC motor with Arduino Uno via Motor Driver

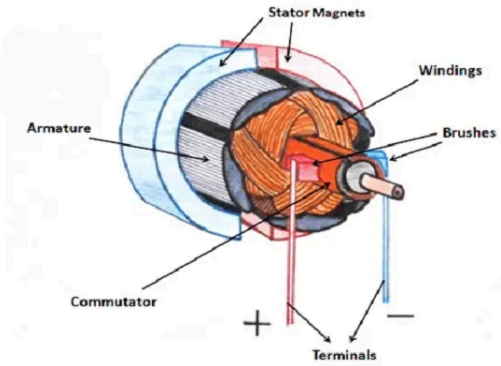
1. **Description of the components:**

- DC Motor

A DC motor operates by converting electrical energy into mechanical motion using the interaction of magnetic fields. Inside the motor, a rotor (armature) is placed within the magnetic field of a stator. When DC voltage is applied to the motor terminals, current flows through the armature windings, creating an electromagnetic field. This field interacts with the stator's magnetic field (either from permanent magnets or field windings), producing a torque that causes the rotor to turn.

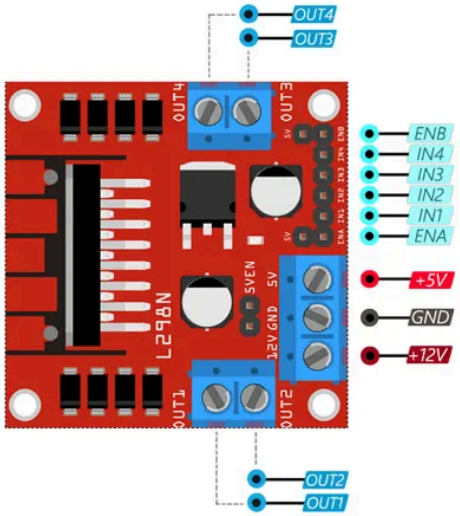






DC motors typically have two leads—one positive and one negative. When connected to a power source like a battery, the motor rotates; reversing the polarity of the connections causes the motor to spin in the opposite direction. The speed of the motor is proportional to the applied voltage, while the torque is proportional to the current. DC motors are widely used in embedded systems due to their simple speed control using PWM (Pulse Width Modulation) and ease of direction reversal using an H-bridge motor driver.

- Motor Driver



A motor driver is an electronic module that acts as an interface between a microcontroller (like an Arduino or Raspberry Pi) and a motor. Microcontrollers cannot supply the high current and voltage required to run motors directly, so a **motor driver takes low-power control signals and uses them to switch a higher-power supply to drive the motor**.

The **L298N motor driver** is a dual H-bridge motor driver based on the **L298N chip**, which can handle up to **46V** and **2A per channel**, making it suitable for low to medium-power motors in robotics and automation projects.

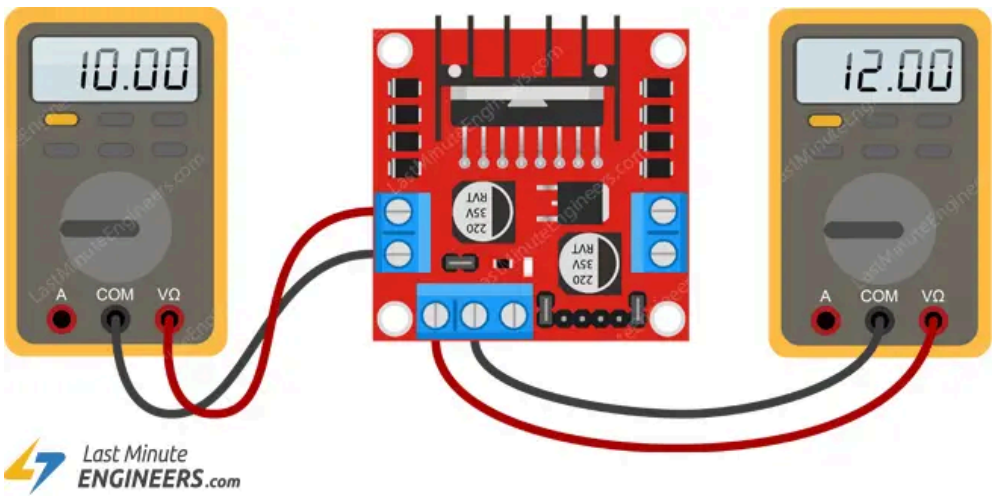
Pins and Connections:

- **INA, INB / IN1, IN2** : Control Motor A direction
- **INC, IND / IN3, IN4**: Control Motor B direction
- **ENA, ENB**: Enable pins (Usually shorted to 5v)

How is direction controlled?

Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

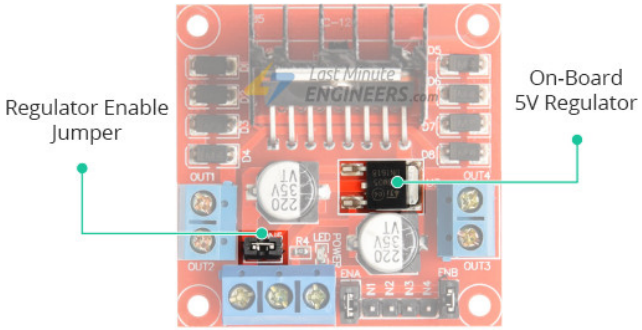
Voltage Drop of L298N:



The L298N motor driver has an **internal voltage drop of about 2V** due to the two forward-biased transistors in its H-bridge circuit. As a result, if we supply 12V to the driver, the motors will receive only about 10V. To ensure the motors run at their full rated speed, the input voltage should be approximately 2V higher than the motor's rated voltage. For example, we need to **use 7V for a 5V motor or 14V for a 12V motor.**

On-board 5V Regulator and Jumper:

The L298N module features a 78M05 voltage regulator that can be turned on or off using a jumper.



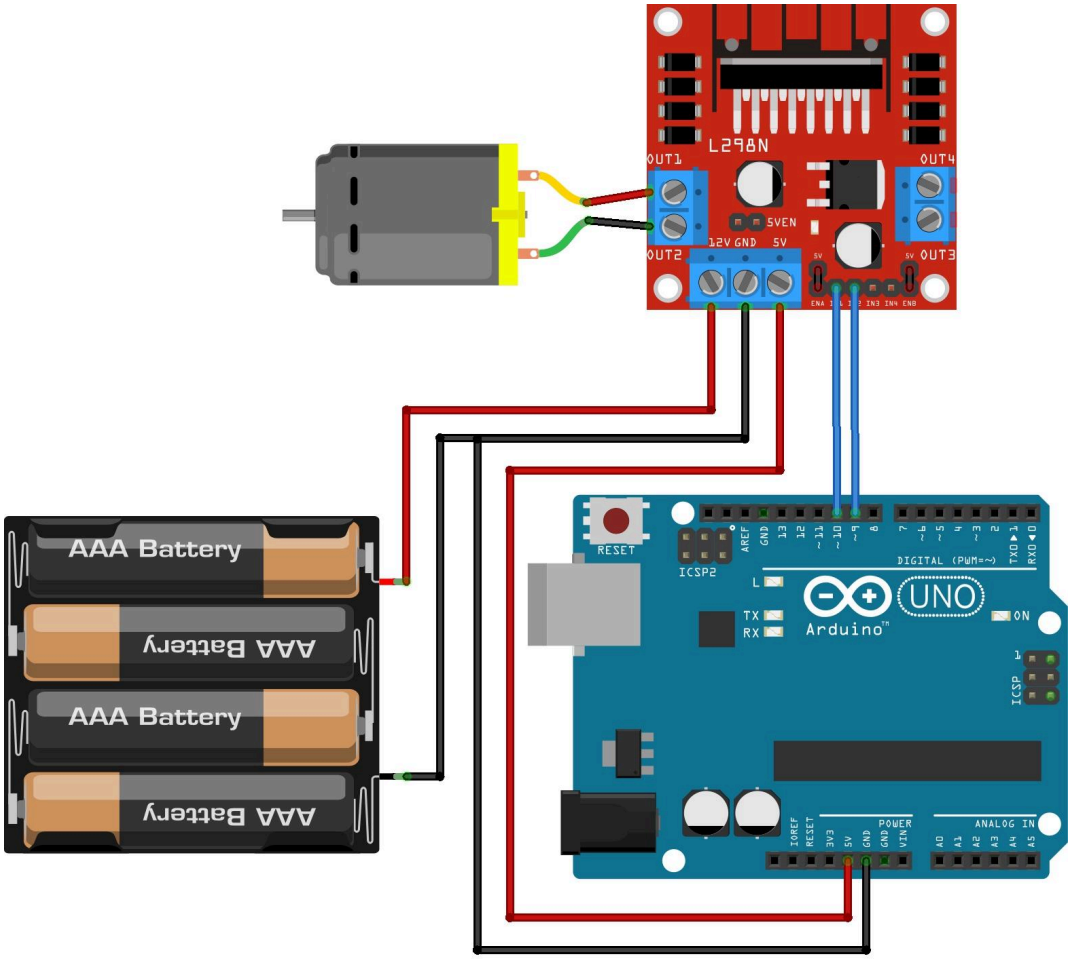
When the jumper is in place, the regulator is enabled, and the module uses the motor power supply (VCC) to generate a 5V logic supply internally. In this mode, **the 5V pin acts as an output**, providing up to 5V at 0.5A, which can be used to **power an Arduino**.

2. **Setting up the circuit:**

- Connect **OUTA** and **OUTB** of the motor driver to the **two terminals of the DC motor**.
- Connect **INA** and **INB** of the motor driver to **Digital pins 10 and 9** of the Arduino.
- Connect **ENA** (Enable A) is shorted to 5V so that it remains enabled.
- Connect **VCC (12V)** on the motor driver to an **external power supply** matching our motor's rated voltage (e.g., 12V, 9V, etc).
- Connect **GND** of the motor driver to **both the power supply ground** and the **Arduino GND**.
- Connect **5v** on the motor driver to **5v** of the Arduino.

Do not connect 5V from Arduino to the driver's 5V pin if the 5V regulator is in enabled and jumper is in place

3. **Circuit diagram:**



4. **Code:**

Arduino Code:

```
#define IN1 10
#define IN2 9

void setup() {
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    // Rotate Motor Clockwise
    Serial.println("Forward");
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    delay(2000);
```

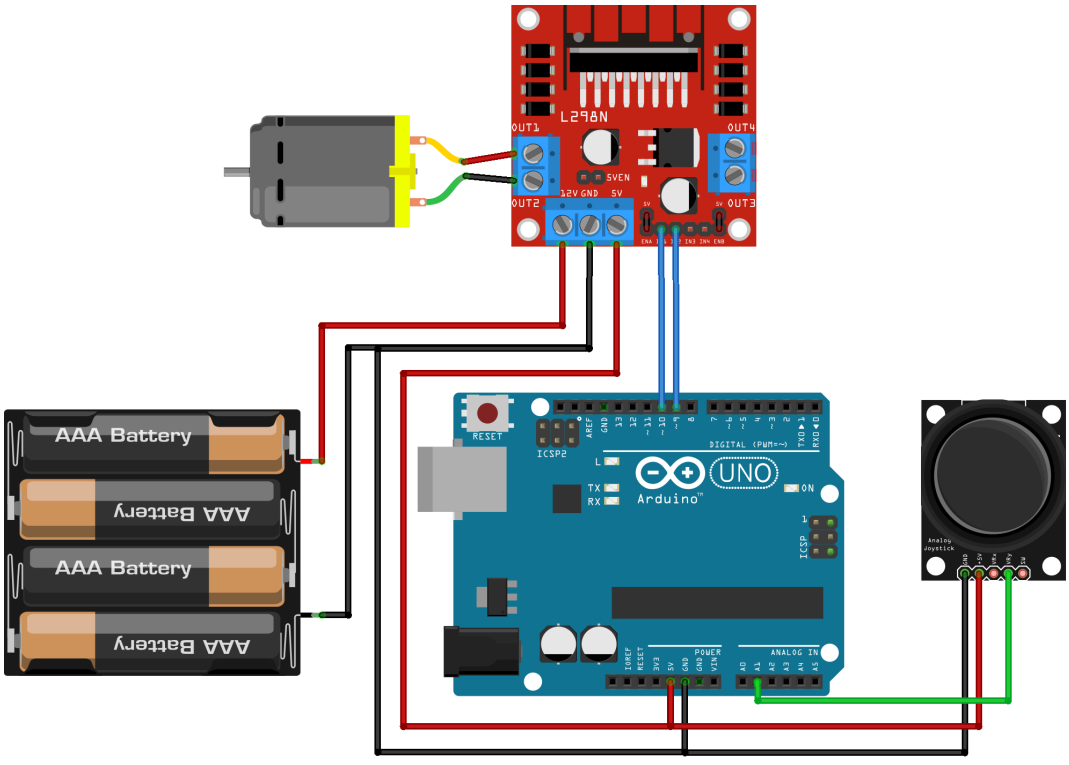
```
    // Stop Motor
    Serial.println("STOP");
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    delay(1000);

    // Rotate Motor Counterclockwise
    Serial.println("Backward");
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    delay(2000);
    // Stop Motor
    Serial.println("STOP");
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    delay(1000);
}
```

How this works: *At hardware level, the Arduino sends low-power control signals to pins INA and INB of L298N. When INA is HIGH and INB is LOW, the H-bridge switches on internal transistors to allow current to flow from the external power supply to the motor, spinning it forward. Reversing the signals makes the current flow in the opposite direction, spinning it backward. Setting both LOW or HIGH stops the motor. The Arduino does not power the motor directly—it only controls the logic, while the motor draws power from the external supply connected to the motor driver*

VI. Experiment 3 : Directional control of DC motor using a Joystick

1. **Circuit diagram:**



2. **Code:**

Arduino Code:

```
#define VRy A1
#define threshold 200
#define IN1 10
#define IN2 9

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int y = analogRead(VRy);

  if (y < 512 - threshold) {       // Forward
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    Serial.println("Forward");
  }
  else if (y > 512 + threshold) { // Backward
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    Serial.println("Backward");
  }
  else {                           // Stop
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
  }

  delay(300);
}
```

How this works: *This code uses the joystick's Y-axis to control a DC motor's direction. The Arduino reads analog values from pin A1; if the value is below 312, the motor spins forward, and if above 712, it spins in reverse. When centered (around 512), the motor stops. The Arduino sends control signals to the motor driver through pins IN1 and IN2, while the motor draws power from an external source connected to the driver.*

VIII. Lab Task: Control a DC motor's direction using the joystick's Y-axis, while using LEDs to visually indicate the direction based on joystick movement

Explanation: Use 3 LEDs

- Motor moves forward, Forward LED ON, other LEDs OFF
- Motor moves backward, Left and Right LEDs ON
- All LEDs turn OFF otherwise

Deliverables:

- Circuit diagram
- Arduino code
- Demonstration of the working circuit

IX. References:

1. Arduino Official Documentation: https://www.arduino.cc/en/Guide
2. L298N Motor Driver: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/