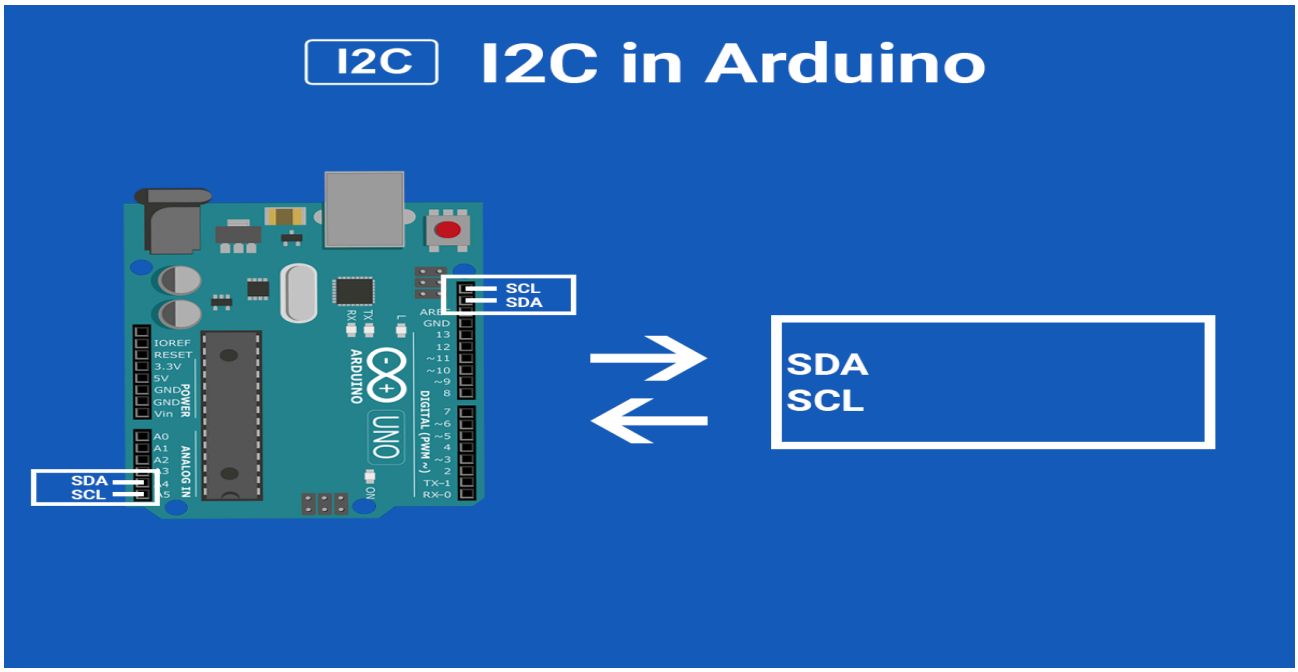
Arduino-Based Motor Control System with I2C Communication

1. Introduction

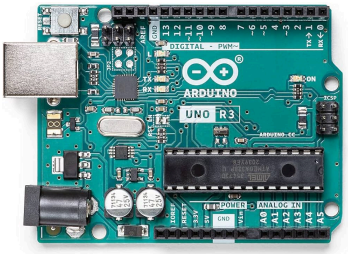
This project aims to control the speed and direction of a DC motor using two Arduino microcontrollers. Arduino 1 reads input from a potentiometer (mapped between -5V and +5V), and Arduino 2 uses this input to control a motor driver. Both Arduinos communicate via the I2C protocol to send and receive data. Arduino 2 is also connected to an LCD screen to display the input voltage and output voltage to the motor.

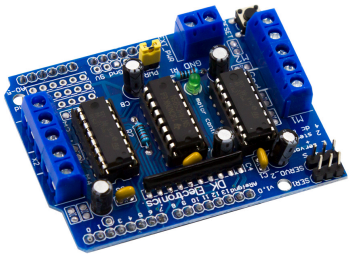


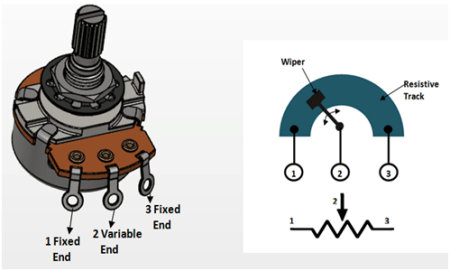
2. Circuit Diagram

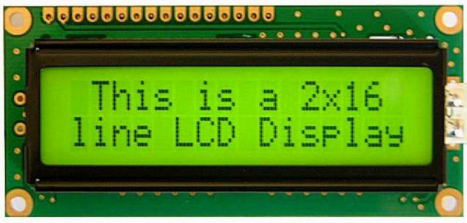
The following components are involved in the project:

- **Arduino 1 (UNO)(Master)**: Reads the potentiometer input and sends the data via I2C.
- **Arduino 2 (Slave)**: Receives data, controls the motor driver, and displays the information on the LCD.
- **Motor Driver (L293D or BTS7960)**: Drives the DC motor.
- **Motor encoder** : Drives the DC motor.
- **Potentiometer(POT-HG)**: Provides the variable input voltage.
- **LCD Display (JHD-2X16-I2C)**: Displays input and output voltages.
- **Wires for I2C Communication**: SDA (Data) and SCL (Clock).



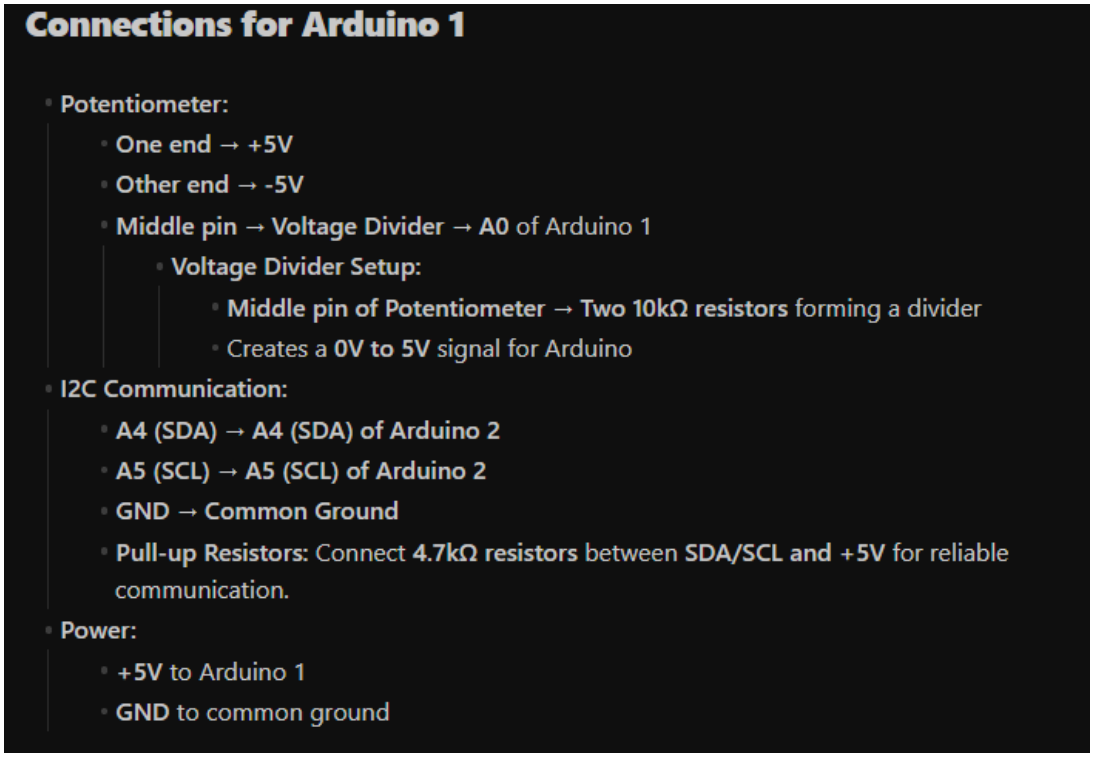






Key Connections:

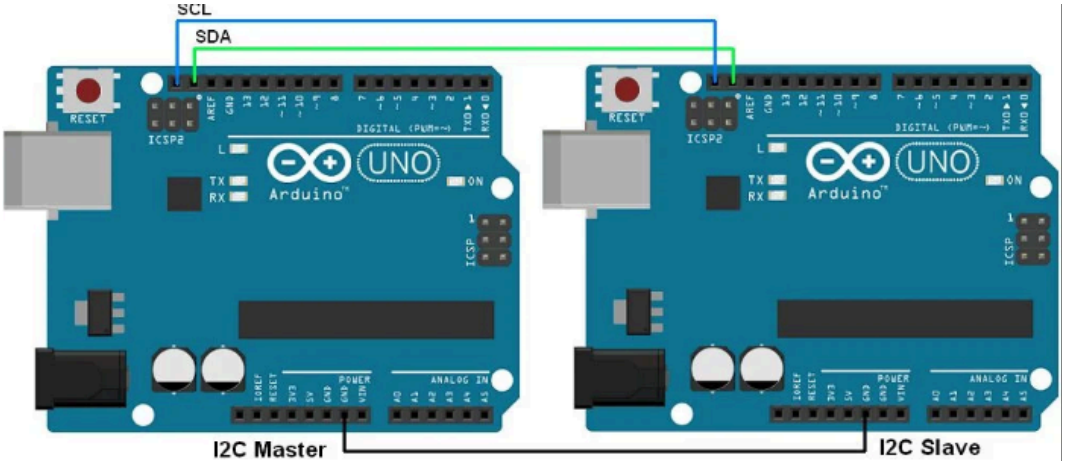
1. **Arduino 1:**
    - **Potentiometer** is connected to **A0**.
    - **SDA (Pin A4)** and **SCL (Pin A5)** are connected to **Arduino 2 (Slave)** via I2C communication.
    - **GND** is shared between Arduino 1 and Arduino 2.
2. **Arduino 2:**
    - **SDA and SCL** pins are connected to **Arduino 1** for I2C communication.
    - **Motor Driver** pins are connected to the motor and control pins (Motor Pins 1, 2, and PWM Pin for speed control).
    - **LCD Display** is connected via **SDA** and **SCL** for I2C communication.
3. **Motor Driver**:
    - The motor driver controls the direction and speed of the motor based on the voltage input from Arduino 2.
    - PWM controls the motor speed.
    - The direction is controlled based on the sign of the input voltage.

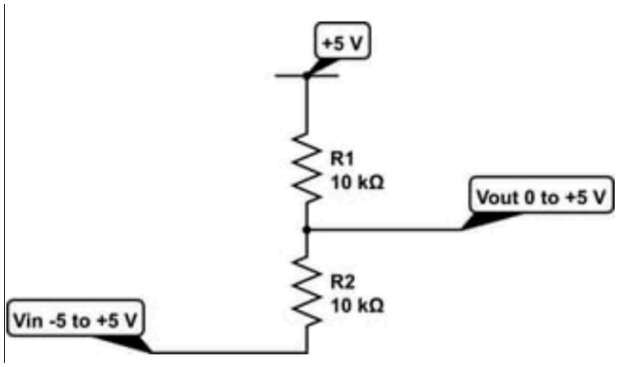


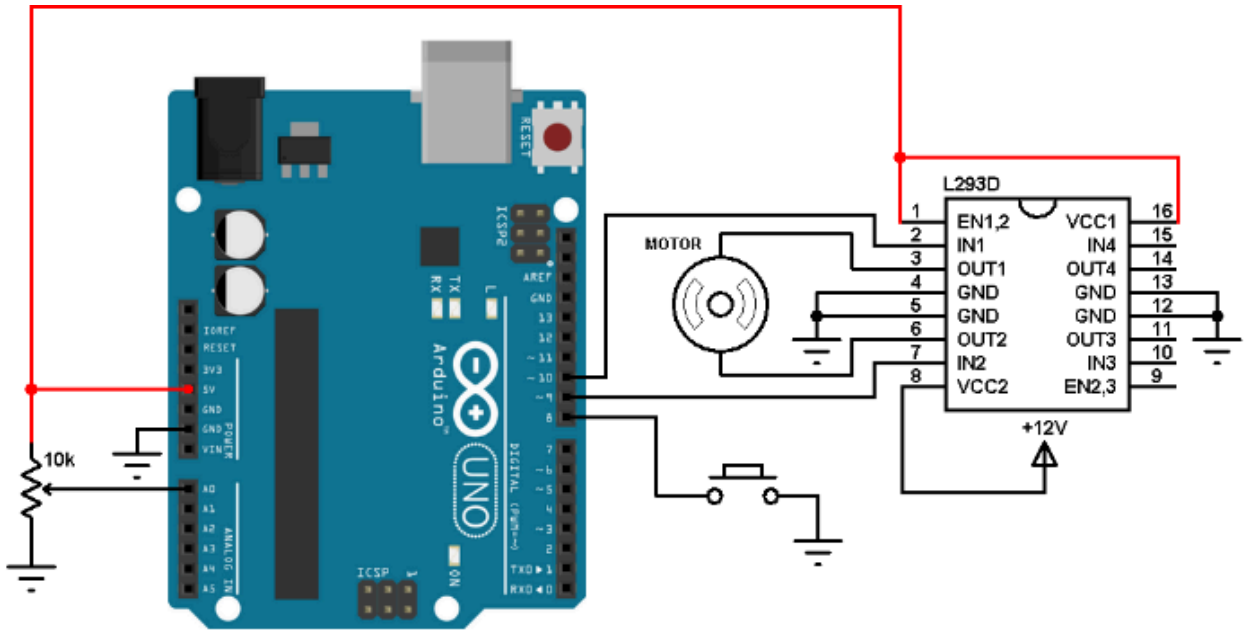
Connections for Arduino 2

- **I2C Communication:**
  - **A4 (SDA) → A4 (SDA) of Arduino 1**
  - **A5 (SCL) → A5 (SCL) of Arduino 1**
  - **GND → Common Ground**
- **L293D Motor Driver:**
  - **Pin 1 (Enable 1,2) → PWM Pin 9 of Arduino 2** (for speed control)
  - **Pin 2 (Input 1) → Digital Pin 13 of Arduino 2** (direction control)
  - **Pin 7 (Input 2) → Digital Pin 12 of Arduino 2** (direction control)
  - **Pin 3,6 (OUT1, OUT2) → Motor Encoder A, B Terminals**
  - **Middle Terminal of Motor Encoder → GND of L293D**
  - **Pin 16 (VSS) → +5V** (Arduino power for logic)
  - **Pin 8 (VS) → +12V** (External power for motor)
  - **Pin 4, 5, 12, 13 (Grounds) → Common Ground**
  - **12V Power Supply Ground → Common Ground** (Important for proper motor operation)
- **Display (Optional - I2C LCD):(JHD-2X16-I2C)**
  - **VCC** OR VDD → **5V**
  - **GND OR VSS → Common Ground**
  - **SDA → A4** of Arduino 2
  - **SCL → A5** of Arduino 2
  - **Library:** Use **LiquidCrystal_I2C.h** for Arduino

PIN DIAGRAMS







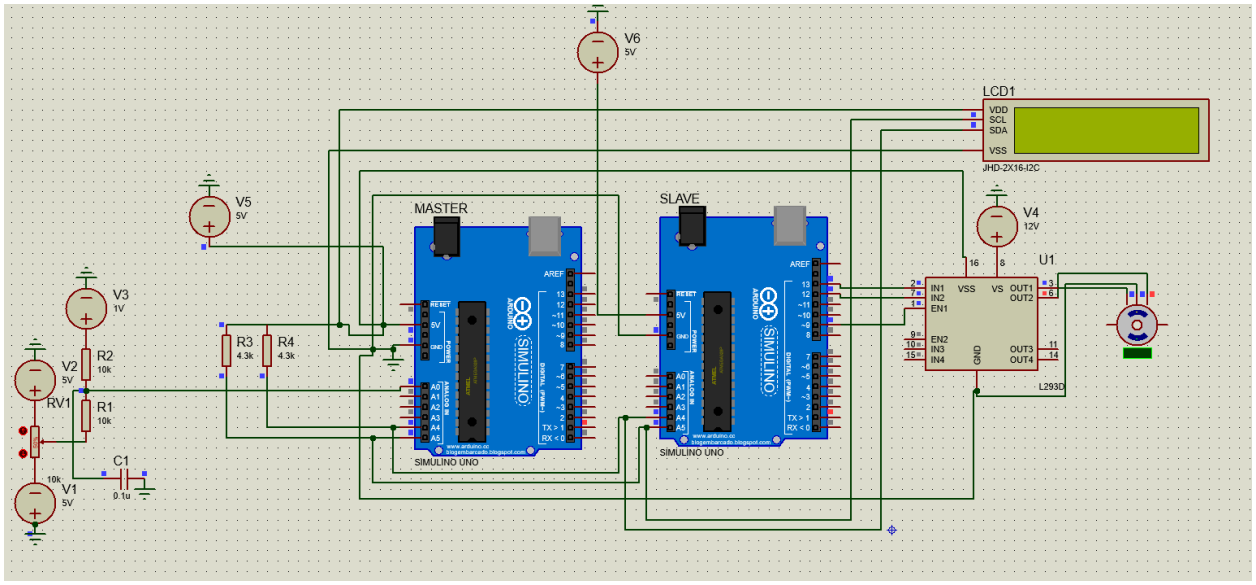
3. Circuitry Explanation

- **Potentiometer (Arduino 1)**: The potentiometer is used as an analog input to Arduino 1. It provides a voltage that varies from -5V to +5V (achieved through a voltage divider or an external power supply). The Arduino maps the analog input to a range of -5V to +5V.

- **Motor Driver**: The motor driver (e.g., **L293D** or **BTS7960**) is used to drive the motor with voltage levels that the Arduino cannot directly generate. It allows the motor to run at higher voltages (e.g., 12V), and the Arduino can control the speed and direction using PWM signals and logic high/low inputs.

- **I2C Protocol**:

    - **SDA (Serial Data)** and **SCL (Serial Clock)** lines are used for communication. The **Master (Arduino 1)** sends the potentiometer voltage data to the **Slave (Arduino 2)** using these lines.
    - **I2C** is a two-wire serial communication protocol, with **SDA** carrying the data and **SCL** providing the clock signal.
    - The data is transmitted in **packets** between devices. In this case, Arduino 1 sends the input voltage to Arduino 2, which then processes the data and controls the motor and updates the LCD.



4. I2C Protocol - Hardware Level Understanding

Overview:

I2C (Inter-Integrated Circuit) is a serial communication protocol that allows multiple devices to communicate over just two wires: **SDA** and **SCL**. It uses **master-slave** communication, where the master device controls the clock line (SCL) and initiates data transmission, while the slave responds to the master's requests.

Wires and Pinout:

1. **SDA (Serial Data)**: Carries the data bits. Each device on the bus listens to this line and can transmit data when the master requests.
2. **SCL (Serial Clock)**: Carries the clock signal generated by the master. It synchronizes data transfer.

Communication Flow:

- **Start Condition**: The master sends a start condition to initiate communication (pulls SDA low while SCL is high).
- **Addressing**: Each I2C device on the bus has a unique address. The master sends a 7-bit or 10-bit address to select the target device.
- **Data Transfer**: Data is transmitted in 8-bit bytes, with the least significant bit (LSB) sent first. The master controls the clock (SCL), and the slave devices send or receive data in sync with the clock signal.
- **Stop Condition**: Once the communication is complete, the master sends a stop condition to release the bus.

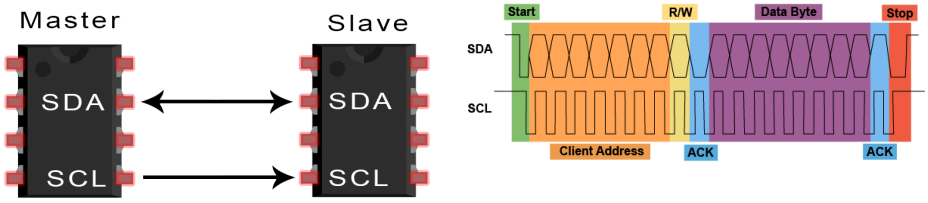
Electrical Considerations:

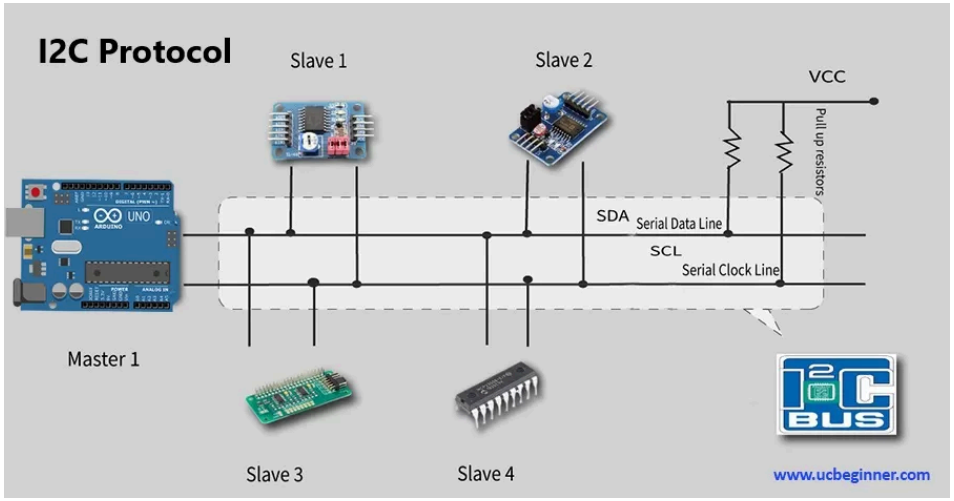
- **Pull-up Resistors**: The SDA and SCL lines require pull-up resistors (typically 4.7kΩ) to ensure they remain at a high logic level when not actively driven low by a device. These resistors are needed because I2C is an open-drain communication protocol.
- **Power Supply**: Both the master and slave devices should share the same **VCC** and **GND** to establish a common reference.

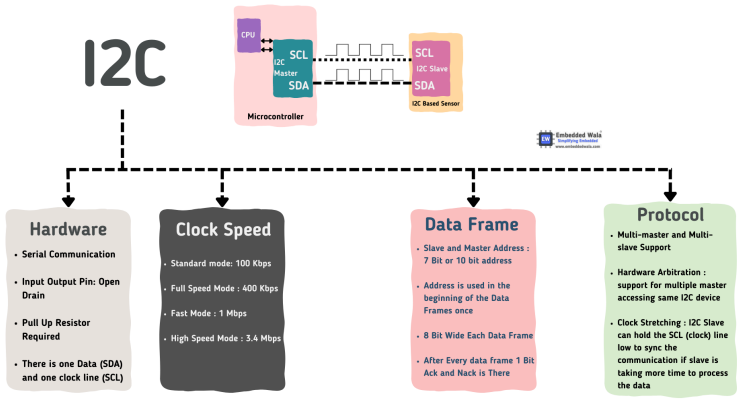
I2C Addressing:

Each I2C device has a **unique 7-bit address**. The master communicates with the slave device by specifying the slave's address. In your setup:

- **Arduino 1 (Master)** sends the data to **Arduino 2 (Slave)** with the address `0x08`.







5. Software Overview

- **Arduino 1 (Master)** reads the potentiometer, processes the analog signal, and sends the mapped voltage value over I2C to Arduino 2.
- **Arduino 2 (Slave)** receives the input voltage via I2C and uses this data to control the motor driver (L293D/BTS7960). It also updates the LCD with the input and output voltages.

6.CODE
(FOR MASTER)

```
// 📌 Include Wire library for I2C communication
#include <Wire.h>

// 📌 Define the analog input pin where the potentiometer is connected
const int potentiometerPin = A0;

// 📌 Variable to store the processed voltage in integer form
int voltageToSend = 0;

void setup() {
  // 📡 Initialize I2C communication as a Master
  Wire.begin();

  // 🖥️ Start Serial communication for debugging (9600 baud rate)
  Serial.begin(9600);
}

void loop() {
  // 📌 Read the raw analog value from the potentiometer (0-1023)
  int sensorValue = analogRead(potentiometerPin);

  // 🔄 Convert the raw sensor value into a voltage range (-5V to +5V)
  float inputVoltage = ((sensorValue / 1023.0) * 10.0) - 5.0;

  // 📝 Convert the float voltage into an integer (scaled by 100) for accurate I2C
transmission
  voltageToSend = (int)(inputVoltage * 100);

  // 📡 Start I2C transmission to Slave device with address 8
  Wire.beginTransmission(8);

  // ✉️ Send the **high byte** of the voltage value
  Wire.write((voltageToSend >> 8) & 0xFF);

  // ✉️ Send the **low byte** of the voltage value
  Wire.write(voltageToSend & 0xFF);

  // 🔚 End transmission to Slave
  Wire.endTransmission();

  // 🖥️ Print the voltage value for debugging purposes
  Serial.print("Input Voltage: ");
  Serial.println(inputVoltage);

  // 🕒 Small delay to prevent buffer overflow and CPU overuse in Proteus
  delay(50);
}
```

(FOR SLAVE)

```
// 📌 Include Wire library for I2C communication
#include <Wire.h>

// 📌 Include LCD library for displaying voltage values
#include <LiquidCrystal_I2C.h>

// 📺 Initialize LCD at I2C address 0x27 (16x2 display)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// ⚙️ Define motor control pins (for H-Bridge or motor driver)
const int motorPin1 = 13;  // Motor Direction Pin 1
const int motorPin2 = 12;  // Motor Direction Pin 2
const int pwmPin = 9;      // PWM pin to control motor speed

// 📌 Variable to store the received voltage from the Master
float inputVoltage = 0;

void setup() {
  // 📡 Initialize I2C communication as a Slave (Address: 8)
  Wire.begin(8);

  // 🔄 Register a function that gets triggered when data is received from Master
  Wire.onReceive(receiveData);

  // ⚙️ Set motor control pins as OUTPUT
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(pwmPin, OUTPUT);

  // 🖥️ Start Serial communication for debugging (9600 baud rate)
  Serial.begin(9600);

  // 📺 Initialize the LCD display
  lcd.init();
  lcd.backlight();

  // 📺 Display startup message
  lcd.setCursor(0, 0);
  lcd.print("Waiting for");
  lcd.setCursor(0, 1);
  lcd.print("input voltage");
}

void loop() {
  // 🔄 Convert received input voltage (-5V to +5V) into a motor voltage range (0V to 12V)
  float outputVoltage = (inputVoltage / 5.0) * 12.0;

  // 🔄 Convert absolute output voltage to a PWM duty cycle (0 to 255)
  int pwmValue = map(abs(outputVoltage), 0, 12, 0, 255);
```

```
  // 🔄 Motor Direction Control:
  if (outputVoltage > 0) {
    // ➡️ If output voltage is positive, rotate motor forward
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    analogWrite(pwmPin, pwmValue);
  } else if (outputVoltage < 0) {
    // ⬅️ If output voltage is negative, rotate motor in reverse
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, HIGH);
    analogWrite(pwmPin, pwmValue);
  } else {
    // ⏹️ Stop the motor if the voltage is zero
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
  }

  // 🖥️ Print voltage values for debugging
  Serial.print("Input Voltage: ");
  Serial.print(inputVoltage);
  Serial.print(" V, Output Voltage: ");
  Serial.print(outputVoltage);
  Serial.println(" V");

  // 📺 Display input and output voltages on LCD screen
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("In: ");
  lcd.print(inputVoltage, 2);
  lcd.setCursor(0, 1);
  lcd.print("Out: ");
  lcd.print(outputVoltage, 2);

  // 🕒 Small delay for stable operation
  delay(50);
}

// 📡 Function to handle received data from the Master
void receiveData(int byteCount) {
  if (byteCount == 2) {  // ✅ Expecting 2 bytes (high + low)
    int receivedValue = (Wire.read() << 8) | Wire.read();  // 🔄 Reconstruct integer value
    inputVoltage = receivedValue / 100.0;  // 🔄 Convert back to float (-5V to +5V)
  }
}
```

NOW PASTE THESE COMPILED HEX FILE FROM ARDUINO IDE INTO RESPECTIVE ARDUINO IN PROTEUS

7. Conclusion

This project demonstrates a fundamental use of the I2C protocol to enable communication between two Arduino boards. Arduino 1 reads an analog input (potentiometer) and sends this information over I2C to Arduino 2, which then uses the data to control a motor. The use of PWM and I2C ensures efficient motor control while maintaining precision in voltage mapping. Additionally, the LCD provides a user-friendly interface to display real-time data.

---

8. References

1. Arduino Documentation on I2C: https://www.arduino.cc/en/Tutorial/MasterReader
2. L293D Motor Driver Datasheet: https://www.tescaglobal.com/pdf/l293d.pdf
3. JHD-2X16-I2C LCD Datasheet: (Search online for the datasheet for this specific module, such as "JHD 2x16 I2C LCD datasheet")
4. I2C Protocol Overview: https://www.i2c-bus.org/