



AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 09

Experiment Title: Implementation of a motor control system using Arduino: Digital input, outputs, and PWM.

| | | | |
|-------------------------|---|----------------------------|-------------|
| Date of Perform: | 12 May 2025 | Date of Submission: | 19 May 2025 |
| Course Title: | Microprocessor and Embedded Systems Lab | | |
| Course Code: | EE4103 | Section: | P |
| Semester: | Spring 2024-25 | Degree Program: | BSc in CSE |
| Course Teacher: | Prof. Dr. Engr. Muhibul Haque Bhuyan | | |

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case Study is my/our original work; no part has been copied from any other student's work or any other source except where due acknowledgment is made.
3. No part of this Assignment/Case Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is acknowledged in the assignment.
4. I/we have not previously submitted or am submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived to detect plagiarism.
6. I/we permit a copy of my/our marked work to be retained by the Faculty Member for review by any internal/external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea, or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic, and visual forms, including electronic data, and oral presentations. Plagiarism occurs when the origin of the source is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or copy my/our work.

* Student(s) must complete all details except the faculty use part.

** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group # 01

| Sl No | Name | ID | PROGRAM | SIGNATURE |
|-------|---------------------------|------------|------------|-----------|
| 1 | Md. Saikot Hossain | 23-51242-1 | BSc in CSE | |
| 2 | Md. Mosharof Hossain Khan | 23-51259-1 | BSc in CSE | |
| 3 | Rimal Banik | 23-51260-1 | BSc in CSE | |
| 4 | Md. Rahidul Islam | 23-51269-1 | BSc in CSE | |
| 5 | Rahat Ahmed | 21-44911-2 | BSc in CSE | |

Faculty use only

| | | |
|------------------|----------------|--|
| FACULTY COMMENTS | Marks Obtained | |
| | Total Marks | |

Contents

| | |
|---|-------|
| Objectives | 3 |
| Apparatus | 3 |
| Circuit Diagram | 3 |
| Code Explanation | 4-5 |
| Hardware Implementation and Explanation | 6 |
| Experimental Output Results | 7-11 |
| Simulation Output Results | 12-15 |
| Answer to Question | 15-20 |
| Discussion | 21-22 |
| Conclusion | 22 |
| References | 22 |

Objectives:

The objectives of this experiment are to-

- Familiarize the students with the PWM signals generated by the Arduino.
- Control the speed of a DC motor using the PWM signals generated by the Arduino.
- Change the direction of rotation of a DC motor using the input push switch.

Apparatus:

1. Arduino IDE 2.3.5
2. Arduino UNO Microcontroller board
3. L298N Driver
4. 5 V High Torque DC Motor with Fan Blades connected to it
5. Potentiometer, push switch, and a resistor of 10 k Ω
6. A DC Power Supply
7. Breadboard
8. Jumper Wires

Circuit Diagram:

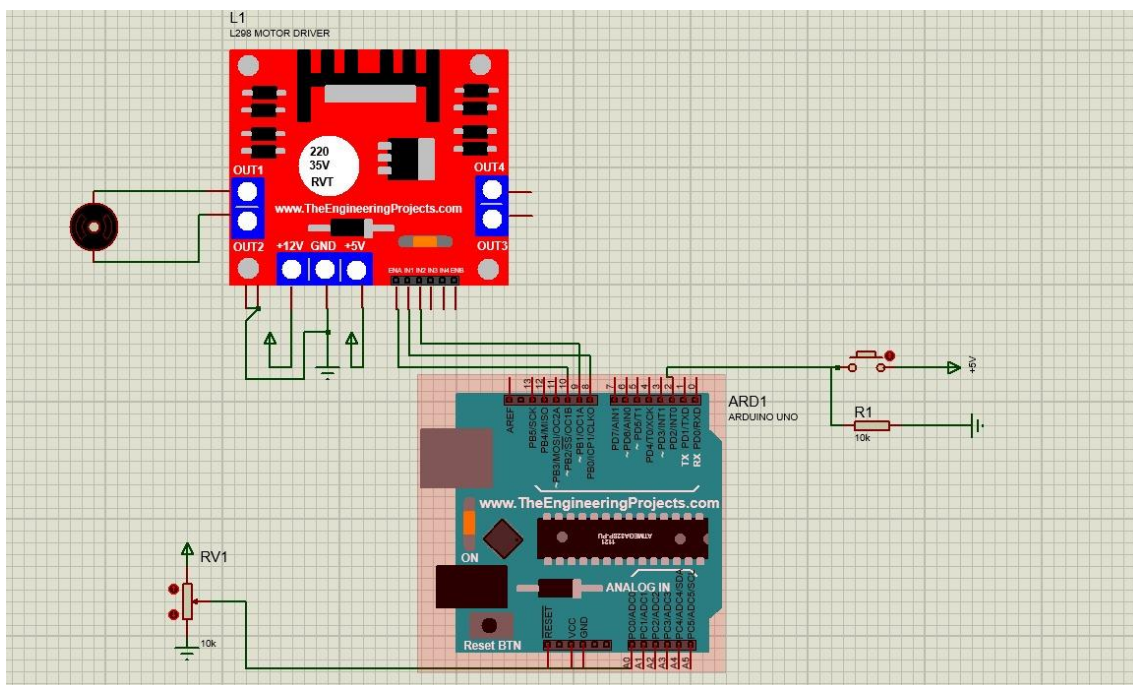


Figure 01: Arduino-Based DC Motor Control Using L298N Driver

Code Explanation:

```
int switchrotation = 2; // input pin to switch the direction of rotation
int in1 = 8; //Declaring where our module is wired
int in2 = 9;
int ConA = 10; // Don't forget this is a PWM DI/DO
int speed1;

void setup()
{
  Serial.begin(9600);
  pinMode(switchrotation, INPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(ConA, OUTPUT);
}

void TurnMotorA1()
{
  //A function to control the direction and speed in one direction
  digitalWrite(in1, LOW); //Switch between these HIGH and LOW states to change direction
  digitalWrite(in2, HIGH);
  float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
  int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
  // range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
  analogWrite(ConA, PWMvalue); // To activate the DC motor
  Serial.println("The motor is running in the clockwise direction."); // May need to change
  Serial.print("Digital Value = ");
  Serial.print(PWMvalue); //print digital value on serial monitor
  //convert digital value to analog voltage
  float analogVoltage = (PWMvalue * 5.00) / 255.00;
  Serial.print(" Analog Voltage = ");
  Serial.println(analogVoltage);
}

void TurnMotorA2()
{
  //A function to control the direction and speed in another direction
  digitalWrite(in1, HIGH); //Switch between these HIGH and LOW states to change direction
  digitalWrite(in2, LOW);
  float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
```

```

int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
// range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
analogWrite(ConA, PWMvalue); // To activate the DC motor
Serial.println("The motor is running in the anticlockwise direction."); // May need to change
Serial.print("Digital Value = ");
Serial.print(PWMvalue); //print digital value on serial monitor

//convert digital value to analog voltage
float analogVoltage = (PWMvalue *5.00)/255.00;
Serial.print(" Analog Voltage = ");

Serial.println(analogVoltage);
}
void loop()
{
if (digitalRead(switchrotation) == LOW)
{
TurnMotorA1();
// function that keeps looping to run the motor continuously.
// you can add another one to stop through the delay() function to run for a certain duration.
}
else if (digitalRead(switchrotation) == HIGH)
{
TurnMotorA2();
}
}

```

Explanation: This Arduino program controls the speed and direction of a DC motor using an L298N motor driver module. The system takes user input through a digital switch connected to **pin 2 (switchrotation)** and an analog voltage via the potentiometer connected to **A0**. Pins **8 and 9 (in1 and in2)** are configured as outputs to determine the motor's rotation direction, while pin **10 (ConA)** is a PWM-enabled pin used to regulate motor speed. In the **setup()** function, the pin modes are defined, and serial communication is initialized at **9600** baud to monitor the motor's status. The core logic lies in the **loop()** function, which continuously checks the state of the digital switch. If the switch is LOW, the **TurnMotorA1()** function runs the motor clockwise by setting in1 LOW and in2 HIGH. If the switch is HIGH, the **TurnMotorA2()** function runs it anticlockwise by reversing the signal levels. In both cases, the analog value from A0 is read, mapped to a range of 0–255 to generate an appropriate PWM signal, and written to ConA to control speed. The corresponding PWM value and its equivalent analog voltage are displayed in the Serial Monitor.

Hardware Implementation and Explanation:

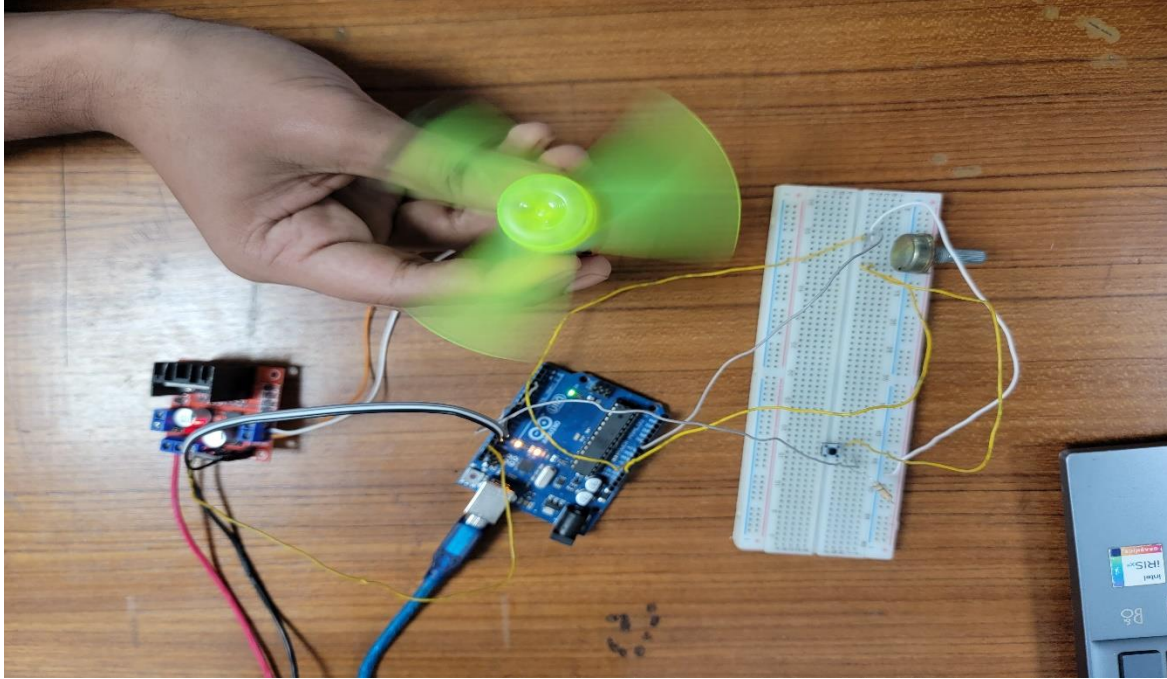


Figure 02: Hardware implementation of a DC Motor Control system

Explanation: In the circuit, an Arduino Uno board is used to control a DC motor. An L298N motor driver module is connected to the Arduino to handle the motor's direction and speed. The IN1 and IN2 pins of the L298N are connected to digital pins 8 and 9 of the Arduino respectively, allowing the Arduino to set the rotation direction. The ENA (Enable A) pin of the motor driver is connected to pin 10 on the Arduino, which is a PWM pin used to control motor speed. A DC motor is connected to the OUT1 and OUT2 terminals of the L298N module. The motor is powered by an external power source connected to the motor driver's 12V and GND input terminals. The GND of the motor driver is also connected to the GND of the Arduino to maintain a common ground. A push-button is connected to digital pin 2 of the Arduino to serve as an input signal to switch the motor's direction. One leg of the button is connected to pin 2, and the other leg is connected to GND through a pull-down resistor on a breadboard. Additionally, a potentiometer is connected to the A0 analog pin of the Arduino to adjust the motor speed. The middle terminal of the potentiometer is connected to A0, while the other two terminals are connected to 5V and GND respectively.

Experimental Output Results:

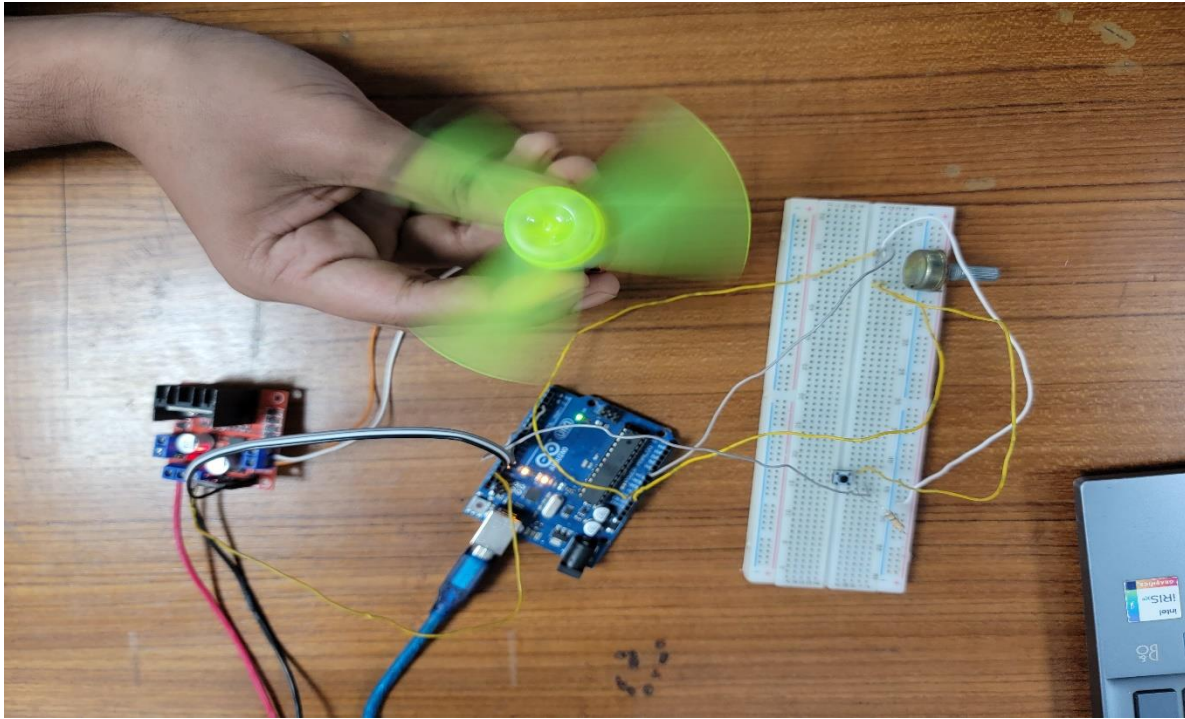


Figure 03: Clockwise Rotation of DC Motor(When switch is not pressed)

```
exp9 | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Arduino Uno
exp9.ino
1 int switchrotation = 2; //input pin to switch the direction of rotation
2 int in1 = 8; //declaring where our module is wired
3 int in2 = 9;
4 int conA = 10; //Don't forget this is a PWM pin/DI/DO
5 int speed1;
6 void setup() {
7   Serial.begin(9600);
8   pinMode(switchrotation, INPUT);
9   pinMode(in1, OUTPUT);
10  pinMode(in2, OUTPUT);
11  pinMode(conA, OUTPUT);
12 }
13 void TurnMotorA1() { //A function to control the direction and speed in one direction
14   digitalWrite(in1, LOW); //Switch between these HIGH and LOW states to change direction
15   digitalWrite(in2, HIGH);
16   float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
17   int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
18   // range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
19   analogWrite(conA, PWMvalue); // To activate the DC motor
20   Serial.println("The motor is running in the clockwise direction."); // May need to change
21   Serial.print("Digital Value = ");
22   Serial.print(analogvalue);
23 }
```

```
Output Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM4')
New Line 9600 baud
= 1.22
The motor is running in the clockwise direction.
Digital Value = 50 Analog Voltage = 0.98
The motor is running in the clockwise direction.
Digital Value = 38 Analog Voltage = 0.75
The motor is running in the clockwise direction.
Digital Value = 33 Analog Voltage = 0.65
The motor is running in the clockwise direction.
Digital Value = 34 Analog Voltage = 0.67
The motor is running in the clockwise direction.
Digital Value = 43 Analog Voltage = 0.84
```

Figure 04: Clockwise Rotation of DC Motor (Digital Value: Lower, Voltage: Lower)

The screenshot shows the Arduino IDE 2.3.6 interface. The code in `exp9.ino` is as follows:

```

1 int switchrotation = 2; // Input pin to switch the direction of rotation
2 int in1 = 8; // Declaring where our module is wired
3 int in2 = 9;
4 int conA = 10; // Don't forget this is a PWM pin/D0
5 int speed1;
6 void setup() {
7   Serial.begin(9600);
8   pinMode(switchrotation, INPUT);
9   pinMode(in1, OUTPUT);
10  pinMode(in2, OUTPUT);
11  pinMode(conA, OUTPUT);
12 }
13 void TurnMotorA1() { // A function to control the direction and speed in one direction
14   digitalWrite(in1, LOW); // Switch between these HIGH and LOW states to change direction
15   digitalWrite(in2, HIGH);
16   float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
17   int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
18   // range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
19   analogWrite(conA, PWMvalue); // To activate the DC motor
20   Serial.println("The motor is running in the clockwise direction."); // May need to change
21   Serial.print("Digital Value = ");
22   Serial.print(PWMvalue);
23 }

```

The Serial Monitor output shows the following data:

```

The motor is running in the clockwise direction.
Digital Value = 37 Analog Voltage = 0.73
The motor is running in the clockwise direction.
Digital Value = 57 Analog Voltage = 1.12
The motor is running in the clockwise direction.
Digital Value = 82 Analog Voltage = 1.61
The motor is running in the clockwise direction.
Digital Value = 92 Analog Voltage = 1.80
The motor is running in the clockwise direction.
Digital Value = 83 Analog Voltage = 1.63
The motor is running in the clockwise direction.

```

Figure 05: Clockwise Rotation of DC Motor (Digital Value: Medium, Voltage: Medium)

The screenshot shows the Arduino IDE 2.3.6 interface. The code in `exp9.ino` is identical to the previous figure:

```

1 int switchrotation = 2; // Input pin to switch the direction of rotation
2 int in1 = 8; // Declaring where our module is wired
3 int in2 = 9;
4 int conA = 10; // Don't forget this is a PWM pin/D0
5 int speed1;
6 void setup() {
7   Serial.begin(9600);
8   pinMode(switchrotation, INPUT);
9   pinMode(in1, OUTPUT);
10  pinMode(in2, OUTPUT);
11  pinMode(conA, OUTPUT);
12 }
13 void TurnMotorA1() { // A function to control the direction and speed in one direction
14   digitalWrite(in1, LOW); // Switch between these HIGH and LOW states to change direction
15   digitalWrite(in2, HIGH);
16   float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
17   int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
18   // range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
19   analogWrite(conA, PWMvalue); // To activate the DC motor
20   Serial.println("The motor is running in the clockwise direction."); // May need to change
21   Serial.print("Digital Value = ");
22   Serial.print(PWMvalue);
23 }

```

The Serial Monitor output shows the following data:

```

The motor is running in the clockwise direction.
Digital Value = 93 Analog Voltage = 1.82
The motor is running in the clockwise direction.
Digital Value = 202 Analog Voltage = 3.96
The motor is running in the clockwise direction.
Digital Value = 244 Analog Voltage = 4.78
The motor is running in the clockwise direction.
Digital Value = 240 Analog Voltage = 4.71
The motor is running in the clockwise direction.
Digital Value = 220 Analog Voltage = 4.31

```

Figure 06: Clockwise Rotation of DC Motor (Digital Value: High, Voltage: High)

Explanation: When the circuit is powered on and the push button remains unpressed, the DC motor operates in the clockwise direction as defined by the **TurnMotorA1()** function. In this function, the Arduino sets one input of the motor driver (in1) to LOW and the other (in2) to HIGH, which controls the direction of rotation. A potentiometer connected to analog pin A0 is used to vary the motor speed. The analog voltage from the potentiometer is read and mapped to a PWM (Pulse Width Modulation) range of 0 to 255. This mapped value is then sent to the motor driver's ConA input, allowing precise control over motor speed. As the potentiometer is adjusted, the analog voltage at pin A0 changes, which leads to a corresponding change in the PWM value and thus the motor speed. During the experiment, the Serial Monitor displays real-time updates showing the motor's direction and speed. It consistently prints the message "The motor is running in the clockwise direction," confirming the set direction. Alongside this, various PWM values and their corresponding analog voltages are displayed. For example, digital (PWM) values such as **33, 38, 43, 50, 82, 83, 91, 93, 220, and 244** were observed, corresponding to analog voltages of approximately **0.65V, 0.75V, 0.84V, 0.98V, 1.61V, 1.63V, 1.80V, 1.82V, 4.31V, and 4.78V**, respectively. These results clearly demonstrate the relationship between potentiometer position, analog voltage, and motor speed. At lower voltages, such as **0.65V**, the motor runs slowly with a PWM value of **33**. As the voltage increases, the PWM value rises accordingly, and the motor speeds up — for instance, reaching near maximum speed at **4.78V** with a PWM value of **244**. This confirms that the Arduino correctly reads the analog input, processes it into PWM output, and effectively controls both the direction and speed of the motor via the L298N driver. In the loop() function, the code continuously checks the state of the push button. When the button is **not pressed** the motor runs in the clockwise direction by calling the TurnMotorA1() function.

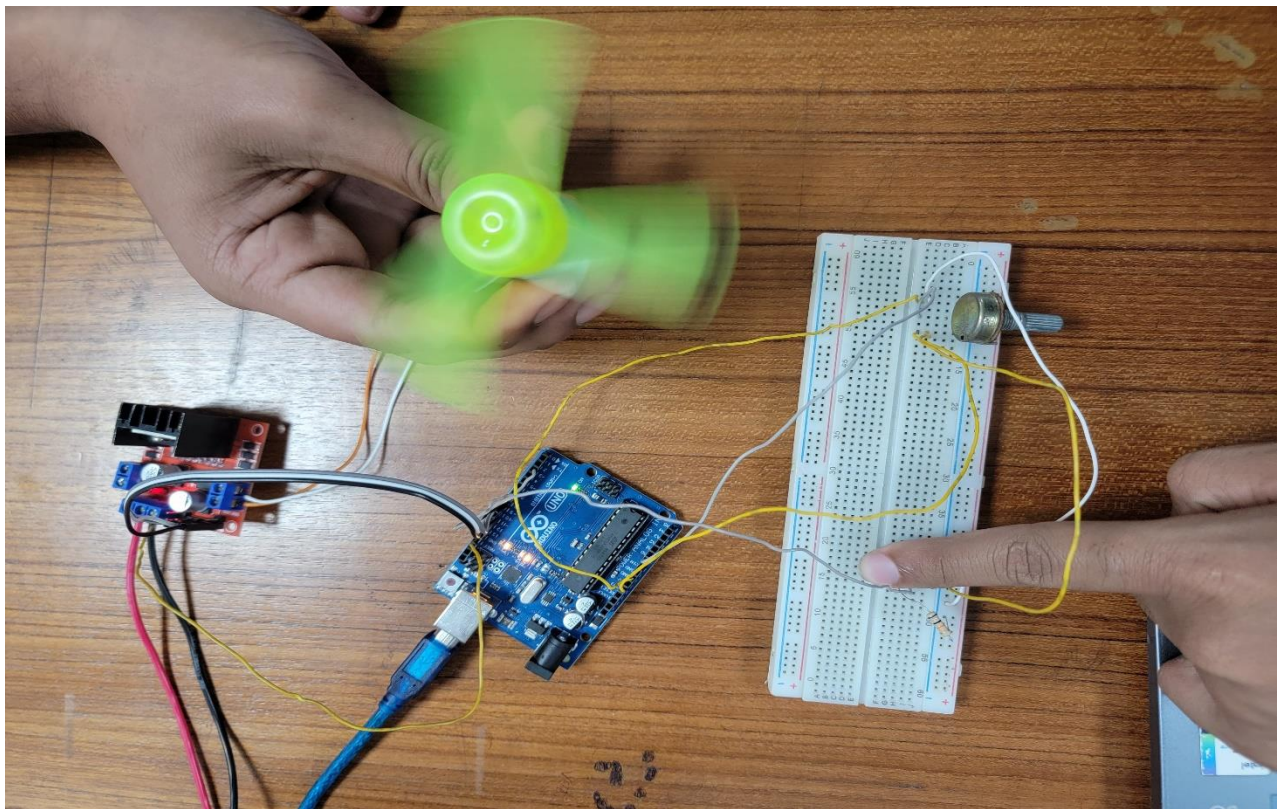


Figure 07: Anticlockwise Rotation of DC Motor (When switch is pressed)

The screenshot shows the Arduino IDE interface with the following code in `exp9.ino`:

```
1 int switchrotation = 2; // Input pin to switch the direction of rotation
2 int in1 = 8; // Declaring where our module is wired
3 int in2 = 9;
4 int conA = 10; // Don't forget this is a PWM pin/D0
5 int speed1;
6 void setup() {
7   Serial.begin(9600);
8   pinMode(switchrotation, INPUT);
9   pinMode(in1, OUTPUT);
10  pinMode(in2, OUTPUT);
11  pinMode(conA, OUTPUT);
12 }
13 void TurnMotorA1() { // A function to control the direction and speed in one direction
14   digitalWrite(in1, LOW); // Switch between these HIGH and LOW states to change direction
15   digitalWrite(in2, HIGH);
16   float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
17   int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
18   // range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
19   analogWrite(conA, PWMvalue); // To activate the DC motor
20   Serial.println("The motor is running in the clockwise direction."); // May need to change
21   Serial.print("Digital Value = ");
22   Serial.print(PWMvalue);
23 }
```

The Serial Monitor output shows the following data:

```
Message (Enter to send message to 'Arduino Uno' on 'COM4')
New Line 9600 baud

The motor is running in the anticlockwise direction.
Digital Value = 49 Analog Voltage = 0.96
The motor is running in the anticlockwise direction.
Digital Value = 48 Analog Voltage = 0.94
The motor is running in the anticlockwise direction.
Digital Value = 48 Analog Voltage = 0.94
The motor is running in the anticlockwise direction.
Digital Value = 51 Analog Voltage = 1.00
The motor is running in the anticlockwise direction.
Digital Value = 60 Analog Voltage = 1.18
```

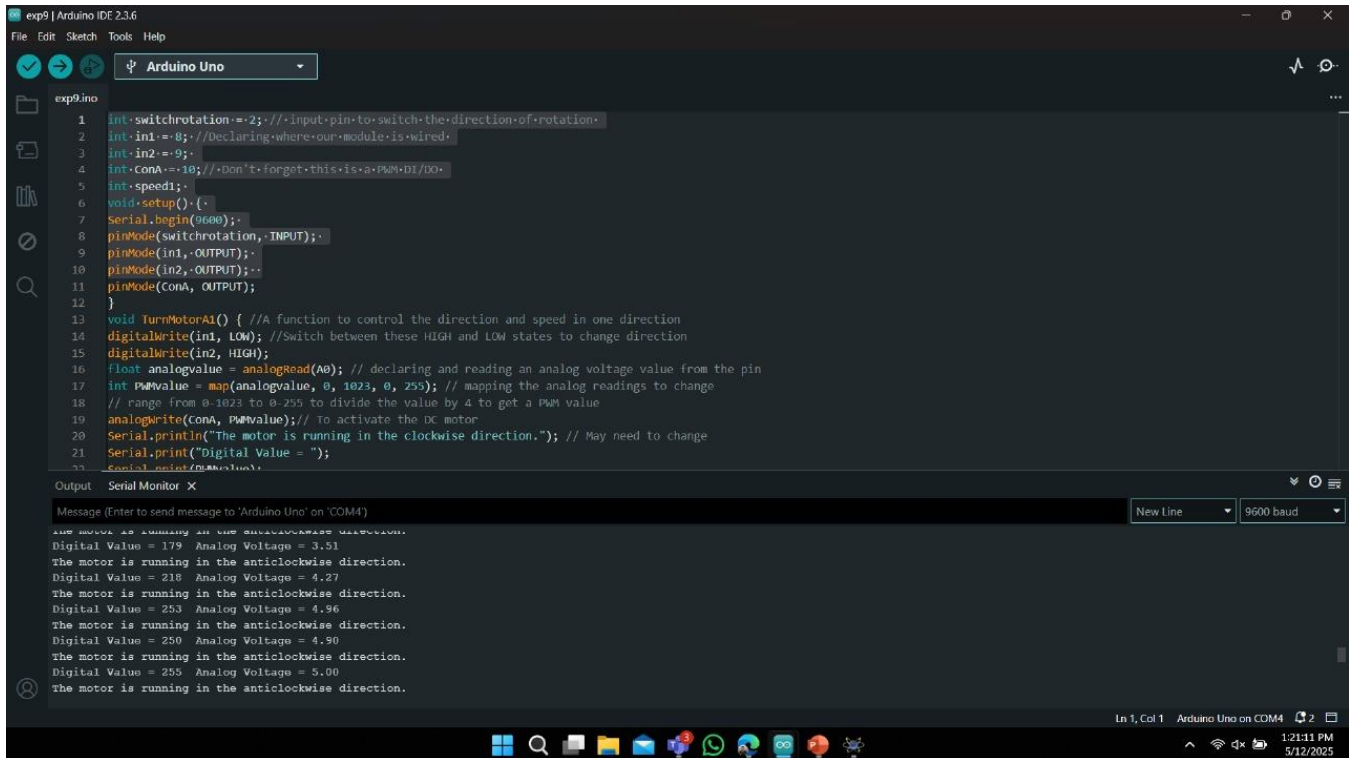
Figure 08: Anticlockwise Rotation of DC Motor (Digital Value: Lower, Voltage: Lower)

The screenshot shows the same Arduino IDE interface with the same code as Figure 08. The Serial Monitor output shows the following data:

```
Message (Enter to send message to 'Arduino Uno' on 'COM4')
New Line 9600 baud

The motor is running in the anticlockwise direction.
Digital Value = 86 Analog Voltage = 1.69
The motor is running in the anticlockwise direction.
Digital Value = 102 Analog Voltage = 2.00
The motor is running in the anticlockwise direction.
Digital Value = 115 Analog Voltage = 2.25
The motor is running in the anticlockwise direction.
Digital Value = 110 Analog Voltage = 2.16
The motor is running in the anticlockwise direction.
Digital Value = 104 Analog Voltage = 2.04
The motor is running in the anticlockwise direction.
```

Figure 09: Anticlockwise Rotation of DC Motor (Digital Value: Medium, Voltage: Medium)



The screenshot displays the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar shows icons for saving, compiling, uploading, and other functions. The 'Sketch' dropdown menu is open, showing 'Arduino Uno' selected. The main editor window shows the code for 'exp9.ino'. The code defines pins for direction control (in1, in2), speed control (ConA), and a potentiometer (A0). It includes a setup function for serial communication and pin modes, and a TurnMotorA2() function that controls the motor's direction and speed based on the potentiometer's position. The Serial Monitor at the bottom shows the output of the code, displaying the digital value and analog voltage for each step of the motor's rotation.

```
1 int switchrotation = 2; // input pin to switch the direction of rotation
2 int in1 = 8; // declaring where our module is wired
3 int in2 = 9;
4 int ConA = 10; // Don't forget this is a PWM pin
5 int speed1;
6 void setup() {
7   Serial.begin(9600);
8   pinMode(switchrotation, INPUT);
9   pinMode(in1, OUTPUT);
10  pinMode(in2, OUTPUT);
11  pinMode(ConA, OUTPUT);
12 }
13 void TurnMotorA2() { // A function to control the direction and speed in one direction
14   digitalWrite(in1, LOW); // Switch between these HIGH and LOW states to change direction
15   digitalWrite(in2, HIGH);
16   float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
17   int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
18   // range from 0.1023 to 0.255 to divide the value by 4 to get a PWM value
19   analogWrite(ConA, PWMvalue); // To activate the DC motor
20   Serial.println("The motor is running in the clockwise direction."); // May need to change
21   Serial.print("Digital Value = ");
22   Serial.print(digitalRead(switchrotation));
23 }
```

Serial Monitor Output:

```
Message (Enter to send message to 'Arduino Uno' on 'COM4')
The motor is running in the anticlockwise direction.
Digital Value = 179 Analog Voltage = 3.51
The motor is running in the anticlockwise direction.
Digital Value = 218 Analog Voltage = 4.27
The motor is running in the anticlockwise direction.
Digital Value = 253 Analog Voltage = 4.96
The motor is running in the anticlockwise direction.
Digital Value = 250 Analog Voltage = 4.90
The motor is running in the anticlockwise direction.
Digital Value = 255 Analog Voltage = 5.00
The motor is running in the anticlockwise direction.
```

Figure 10: Anticlockwise Rotation of DC Motor (Digital Value: High, Voltage: High)

Explanation: When the circuit is powered on and the push button is pressed, the DC motor rotates in the anticlockwise direction as defined by the **TurnMotorA2()** function. In this function, the Arduino sets input pins of the motor driver (in1 to HIGH and in2 to LOW), reversing the current through the motor and changing its direction. A potentiometer connected to analog pin A0 is used to control the motor's speed. The analog voltage from the potentiometer is read by the Arduino, mapped to a PWM (Pulse Width Modulation) range of 0 to 255, and sent to the motor driver's ConA pin. As the potentiometer is adjusted, the analog voltage at pin A0 varies, which in turn changes the PWM value and the motor speed. The Serial Monitor displays real-time updates that include both the motor's direction and speed. The message "The motor is running in the anticlockwise direction" is consistently shown, confirming the motor's direction. Additionally, various PWM values and their corresponding analog voltages are printed. For example, digital (PWM) values such as **48, 51, 60, 102, 110, 115, 250, 253, and 255** were observed, corresponding to analog voltages of approximately **0.94V, 1.00V, 1.18V, 2.00V, 2.16V, 2.25V, 4.90V, 4.96V, and 5.00V** respectively. These results demonstrate the relationship between the potentiometer's position, the analog voltage, and the motor's speed. At lower voltages, such as **0.94V**, the motor runs slowly with a PWM value of **48**. As the voltage increases, the PWM value increases, causing the motor to speed up. For instance, the motor reaches near maximum speed at **5.00V** with a PWM value of **255**. This shows that the Arduino is effectively reading the analog input, converting it to a PWM signal, and controlling the motor's direction and speed via the L298N driver. In the loop() function, the code continuously checks the state of the push button. When the button is pressed, the motor runs in the anticlockwise direction by calling the TurnMotorA2() function.

Simulation Output Results:

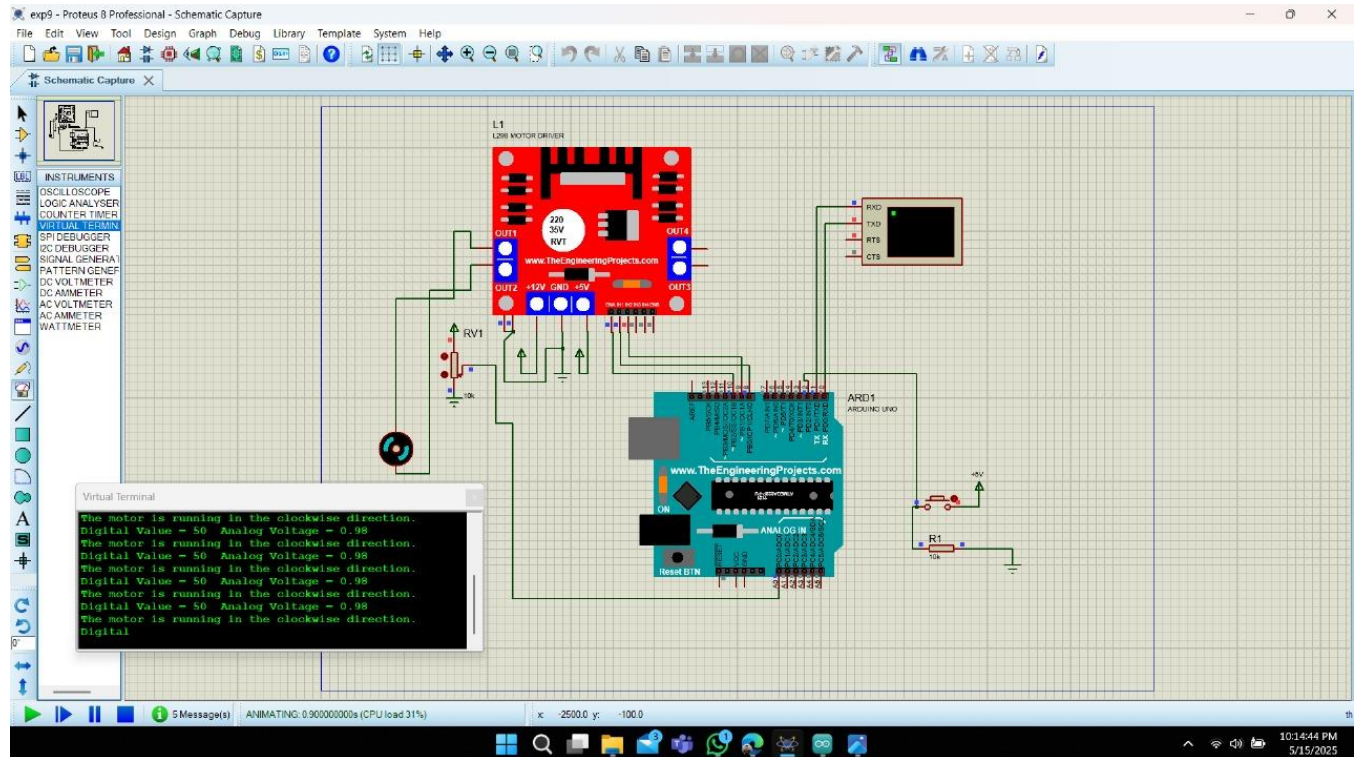


Figure 11: Simulation for Clockwise Rotation of DC Motor (Digital Value:50, Voltage: 0.98V)

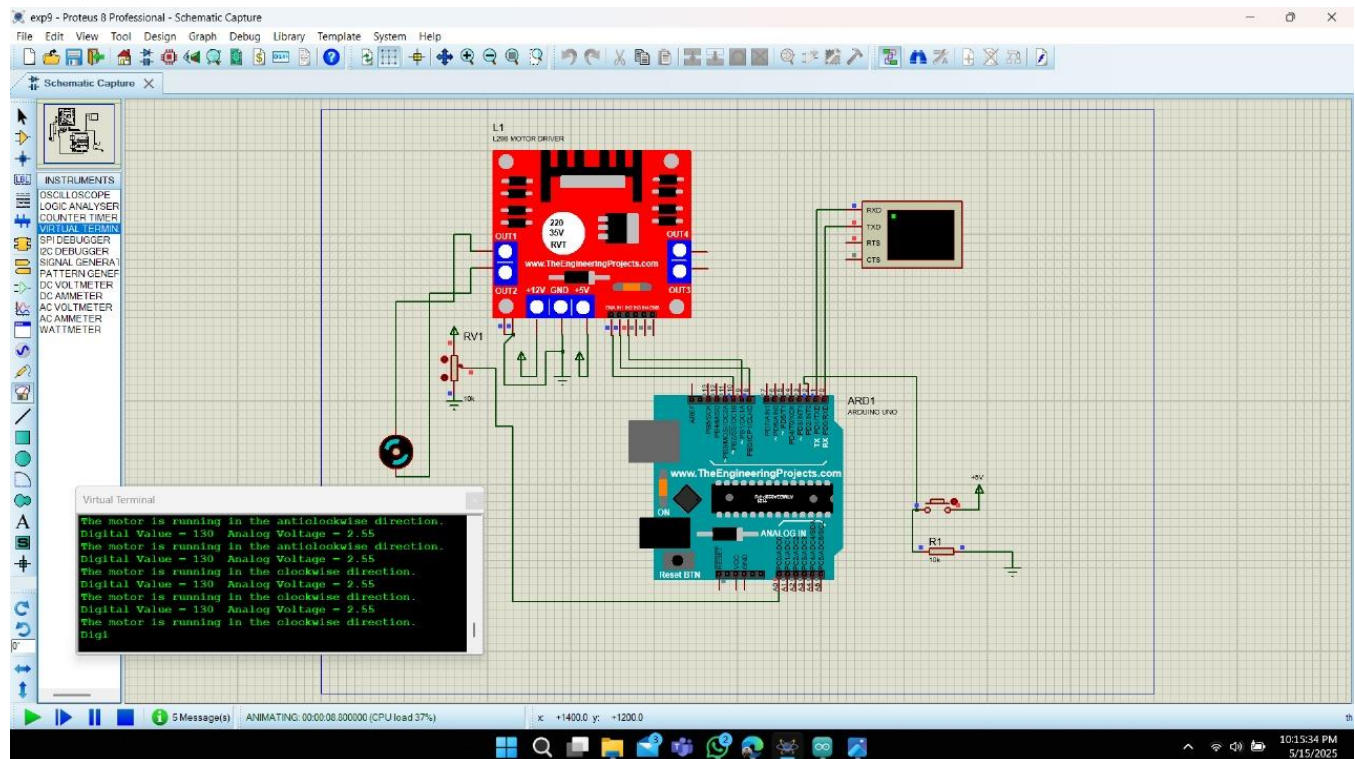
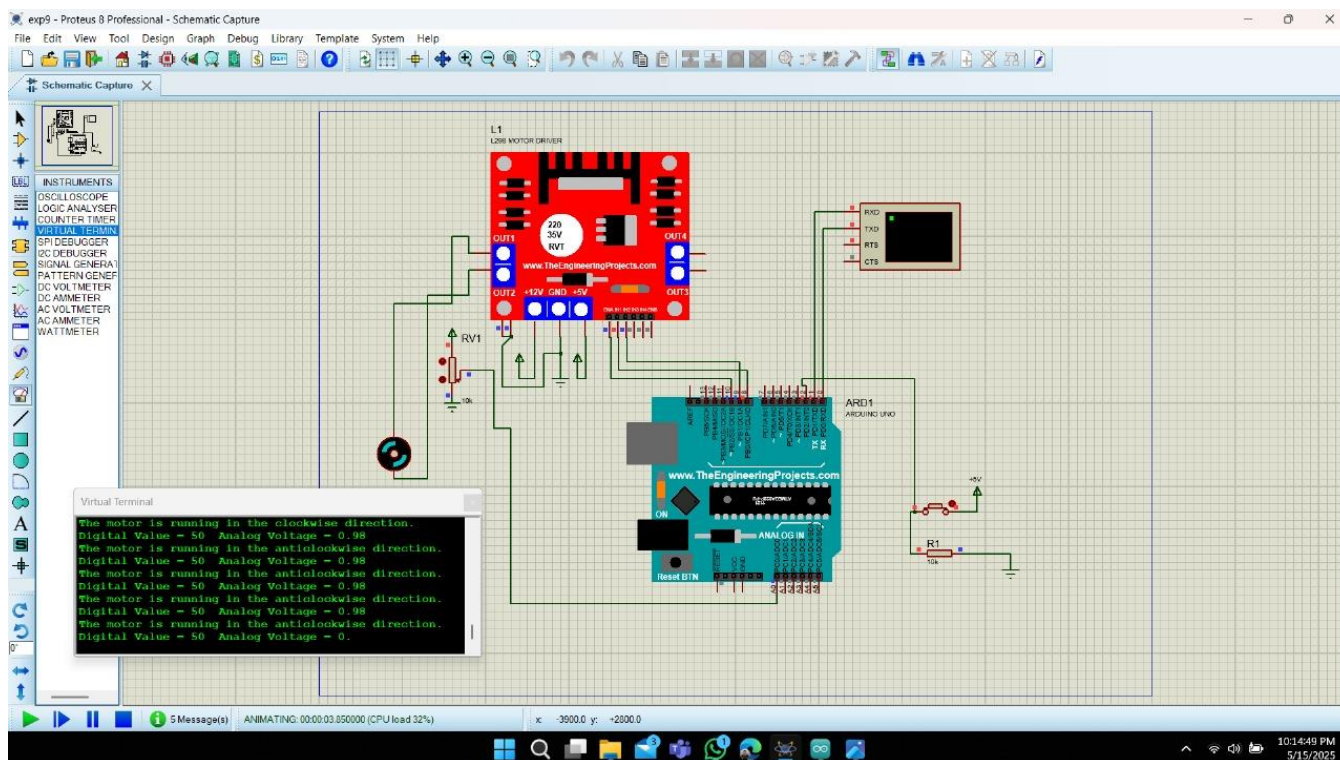
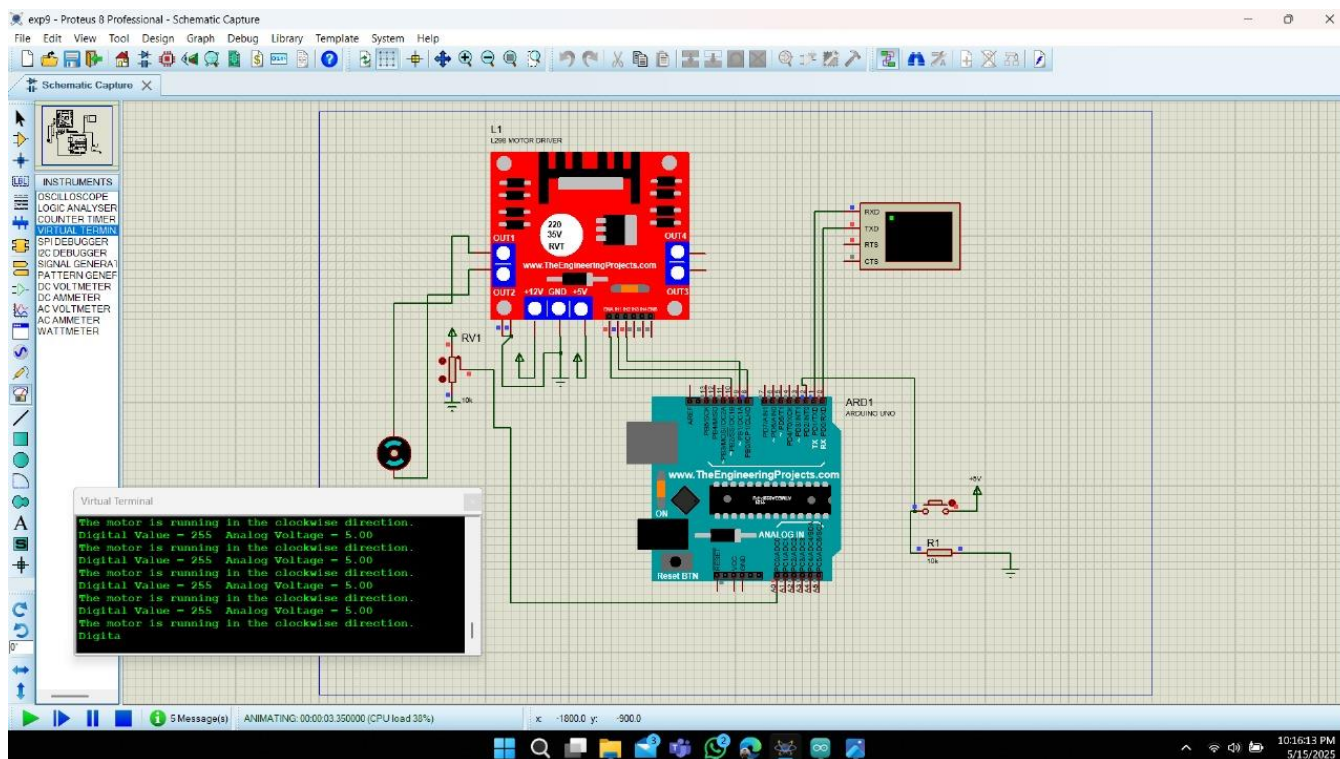


Figure 12: Simulation for Clockwise Rotation of DC Motor (Digital Value: 130, Voltage: 2.55V)



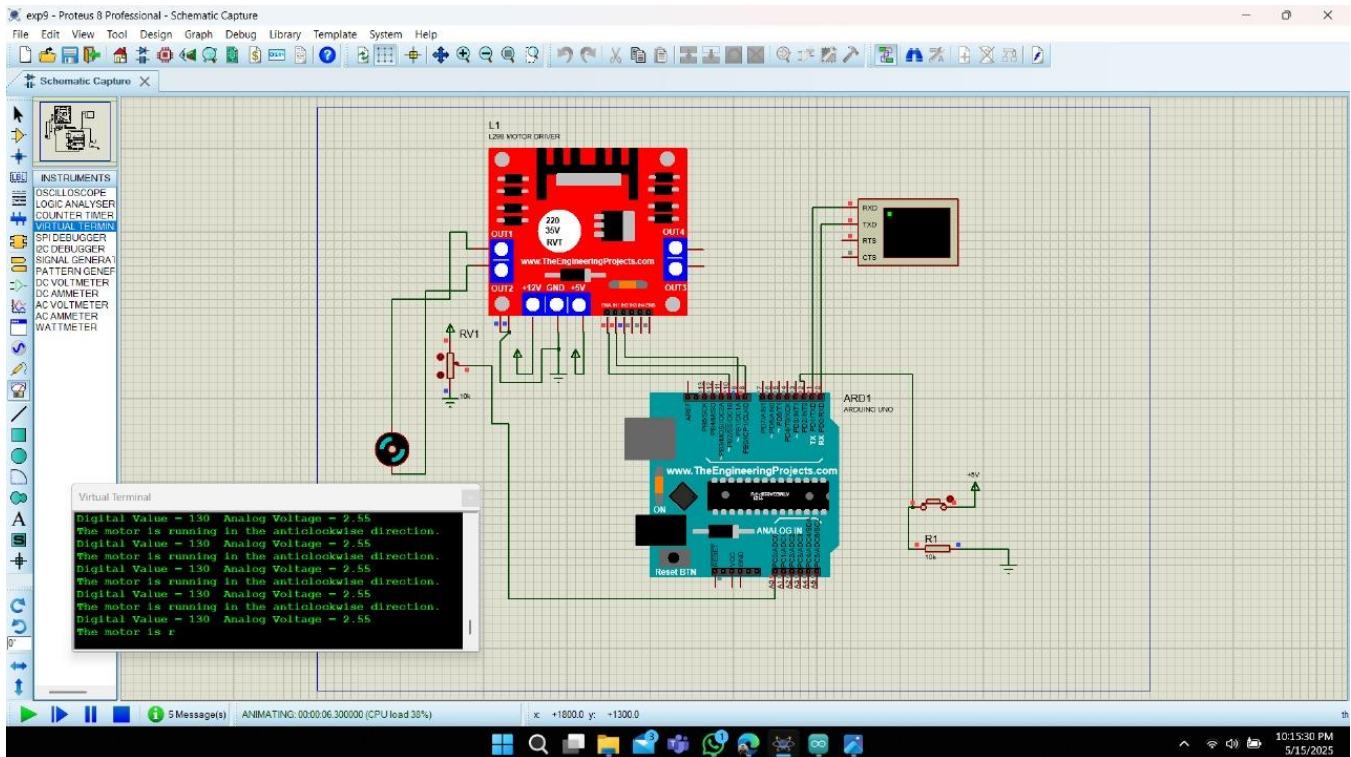


Figure 15: Simulation for Anticlockwise Rotation of DC Motor (Digital Value: 130, Voltage: 2.55V)

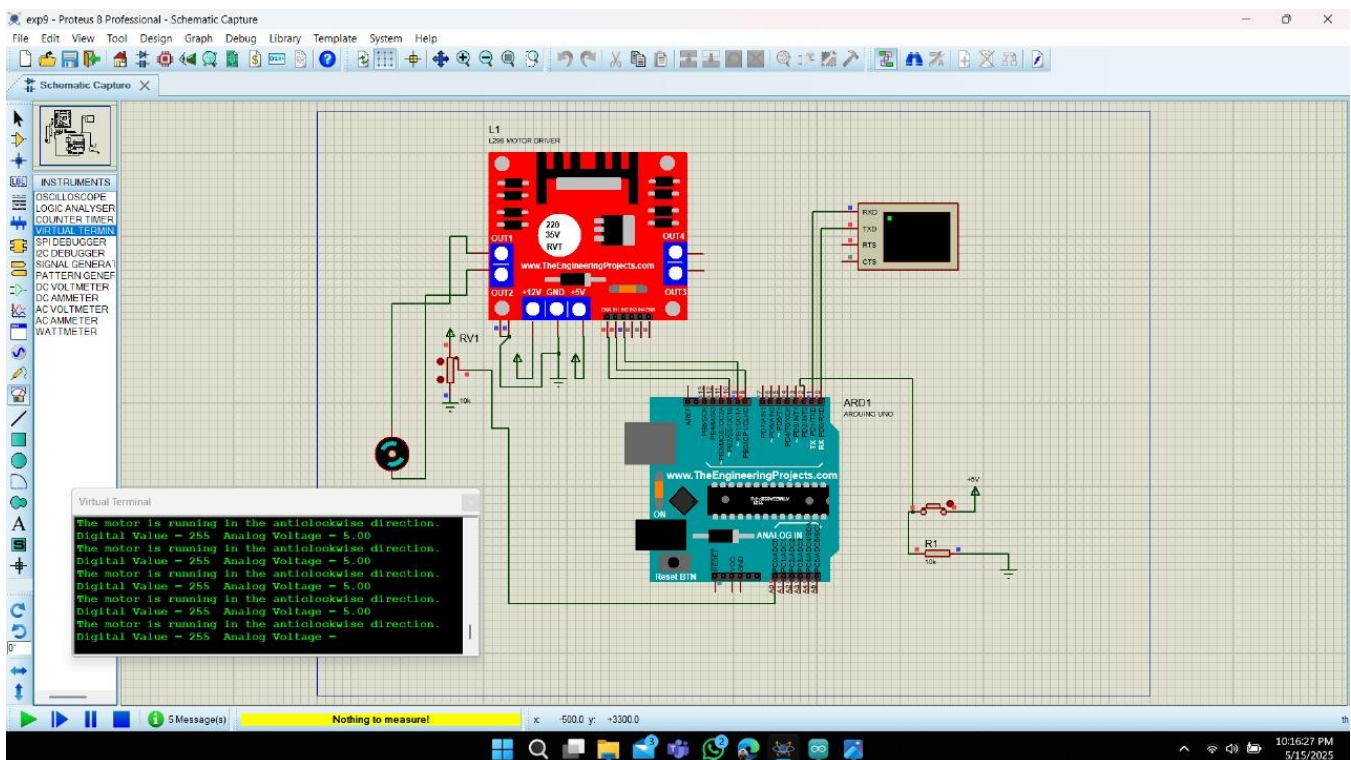


Figure 16: Simulation for Anticlockwise Rotation of DC Motor (Digital Value:255, Voltage:5V)

Explanation: In this simulation, an Arduino Uno, L298N motor driver, push button, potentiometer (RV1), and a DC motor were used to control motor direction and speed. The Arduino code was written in IDE version 2.3.5, and the HEX file was loaded into Proteus for simulation. The potentiometer sets the PWM signal from 0 to 255, controlling the motor's speed, while the push button toggles the rotation direction. When the button is not pressed, the motor runs in the clockwise direction using the TurnMotorA1() function. As the potentiometer is adjusted, the motor responds with varying speeds: at a digital value of 50, the analog voltage is 0.98V; at 125, it's 2.55V; and at 255, it reaches 5V, corresponding to low, medium, and maximum speeds respectively. When the button is pressed, the direction changes to anticlockwise via TurnMotorA2(), with the same voltage-speed relationship. The virtual terminal continuously displays the motor's direction and PWM values, confirming the system works correctly. The simulation results closely match real hardware behavior, verifying the design and code logic for reliable bidirectional speed control of a DC motor using Arduino and L298N.

Answer to Question:

3) My ID no:23-51242-1. Based on my ID number, the four digits **5, 1, 2, and 4** correspond to the control pins as follows: **In1 = 5, In2 = 1, switchRotation = 2, and ConA = 4**. According to the question, these pins should be used directly. However, **pin 1** is the **TX pin** of the Arduino, which is used for serial communication. Using it as a motor control pin creates a conflict and disrupts the serial monitor functionality. To resolve this, I replaced pin **1** with **pin 6** for **In2**. Similarly, **ConA** was supposed to be assigned to **pin 4**, but pin 4 is not a PWM-enabled pin, which is essential for analog voltage control in motor speed regulation. Therefore, I used **pin 9** instead of 4 for ConA.

According to question $P=5(\text{In1})$, $Q= \frac{1}{6}(\text{In2})$ and $A=2(\text{switchRotation})$, $B= \frac{4}{9}(\text{ConA})$

Code:

```
int switchrotation = 2; // input pin to switch the direction of rotation
int in1 = 5; //Declaring where our module is wired
int in2 = 6;
int ConA = 9; // Don't forget this is a PWM DI/DO
int speed1;
void setup()
{
  Serial.begin(9600);
  pinMode(switchrotation, INPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(ConA, OUTPUT);
}
```

```

void TurnMotorA1()
{
//A function to control the direction and speed in one direction
digitalWrite(in1, LOW); //Switch between these HIGH and LOW states to change direction
digitalWrite(in2, HIGH);
float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
// range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
analogWrite(ConA, PWMvalue); // To activate the DC motor
Serial.println("The motor is running in the clockwise direction."); // May need to change
Serial.print("Digital Value = ");
Serial.print(PWMvalue);
    //print digital value on serial monitor
//convert digital value to analog voltage
float analogVoltage = (PWMvalue * 5.00)/255.00;
Serial.print(" Analog Voltage = ");
Serial.println(analogVoltage);
}
void TurnMotorA2() {
//A function to control the direction and speed in another direction
digitalWrite(in1, HIGH); //Switch between these HIGH and LOW states to change direction
digitalWrite(in2, LOW);
float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
// range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
analogWrite(ConA, PWMvalue); // To activate the DC motor
Serial.println("The motor is running in the anticlockwise direction."); // May need to change
Serial.print("Digital Value = ");
Serial.print(PWMvalue); //print digital value on serial monitor

//convert digital value to analog voltage
float analogVoltage = (PWMvalue * 5.00)/255.00;
Serial.print(" Analog Voltage = ");
Serial.println(analogVoltage); }
void loop() {
if (digitalRead(switchrotation) == LOW) {
TurnMotorA1();
// function that keeps looping to run the motor continuously.
// you can add another one to stop through the delay() function to run for a certain duration.
}
else if (digitalRead(switchrotation) == HIGH) {
TurnMotorA2(); }
}

```

Explanation: This Arduino program controls the speed and direction of a DC motor using an L298N motor driver module. The system takes user input through a digital switch connected to **pin 2 (switchrotation)** and an analog voltage via the potentiometer connected to **A0**. Pins **5 and 6 (in1 and in2)** are configured as outputs to determine the motor's rotation direction, while pin **9 (ConA)** is a PWM-enabled pin used to regulate motor speed. In the **setup()** function, the pin modes are defined, and serial communication is initialized at **9600** baud to monitor the motor's status. The core logic lies in the **loop()** function, which continuously checks the state of the digital switch. If the switch is LOW, the **TurnMotorA1()** function runs the motor clockwise by setting in1 LOW and in2 HIGH. If the switch is HIGH, the **TurnMotorA2()** function runs it anticlockwise by reversing the signal levels. In both cases, the analog value from A0 is read, mapped to a range of 0–255 to generate an appropriate PWM signal, and written to ConA to control speed. The corresponding PWM value and its equivalent analog voltage are displayed in the Serial Monitor.

Simulation:

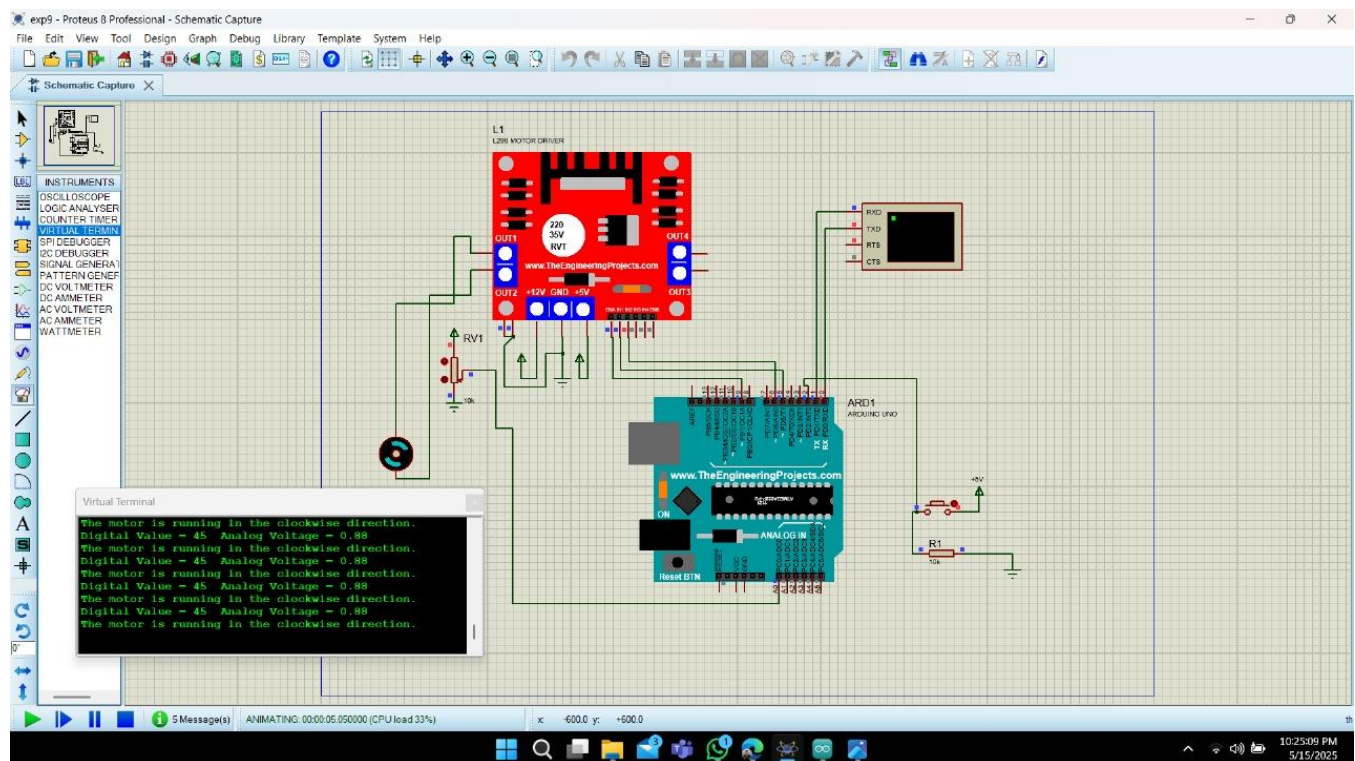


Figure 17: Simulation for Clockwise Rotation of DC Motor (Digital Value:45, Voltage:0.88V)

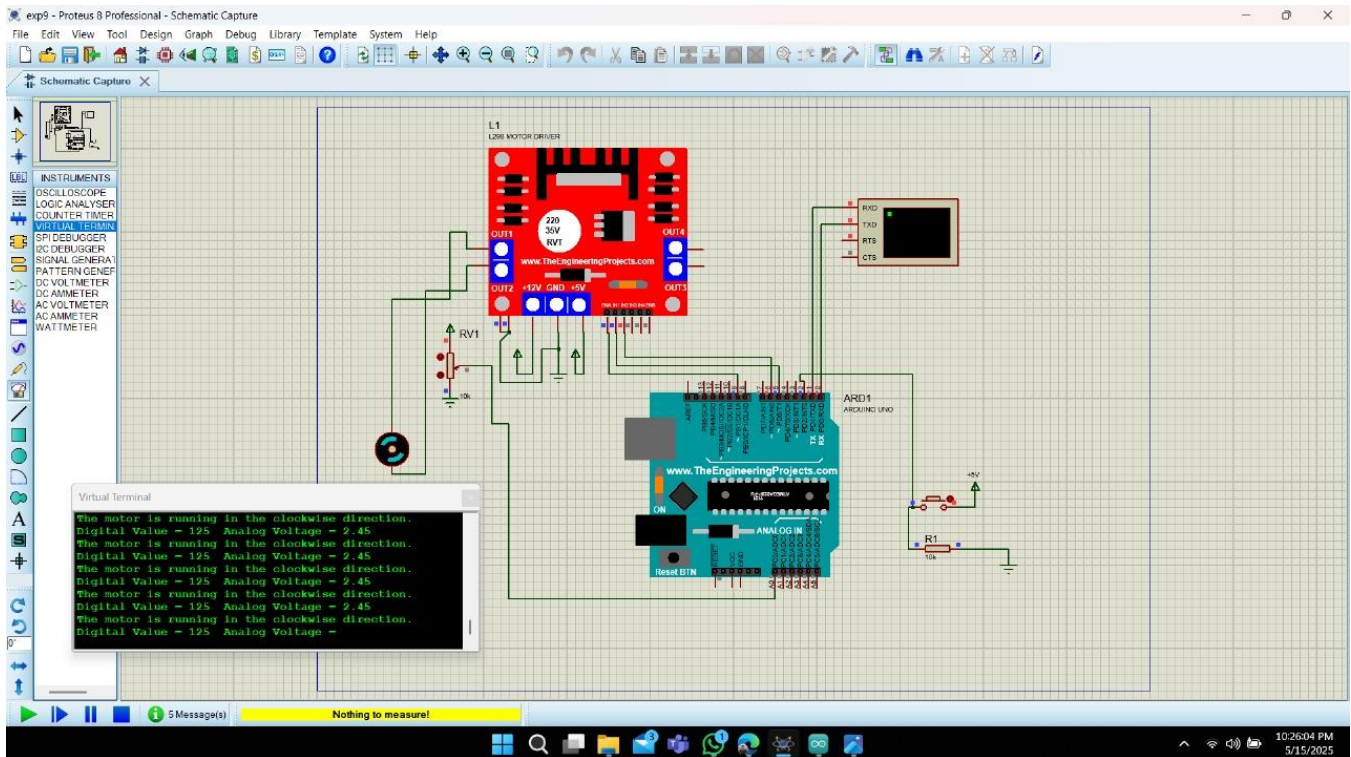


Figure 18: Simulation for Clockwise Rotation of DC Motor (Digital Value: 125, Voltage: 2.45V)

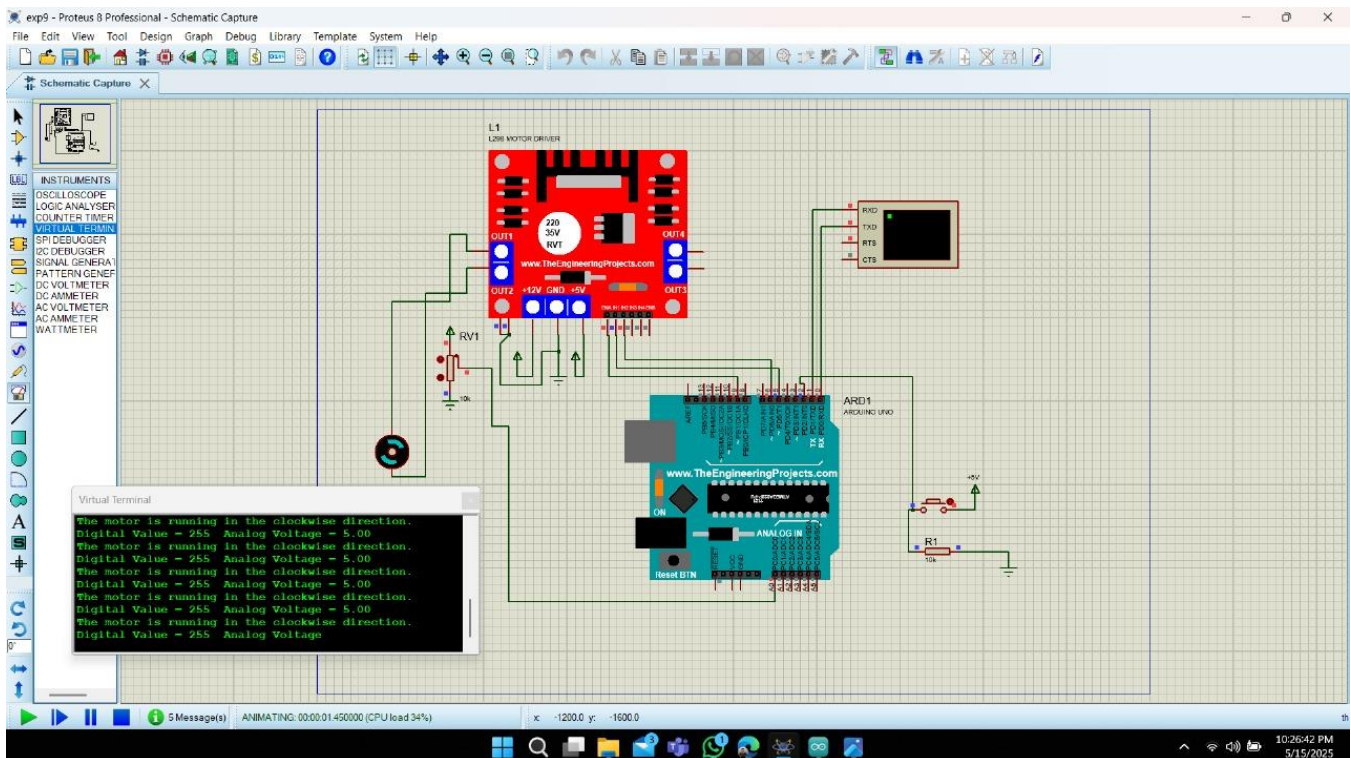


Figure 19: Simulation for Clockwise Rotation of DC Motor (Digital Value: 255, Voltage: 5V)

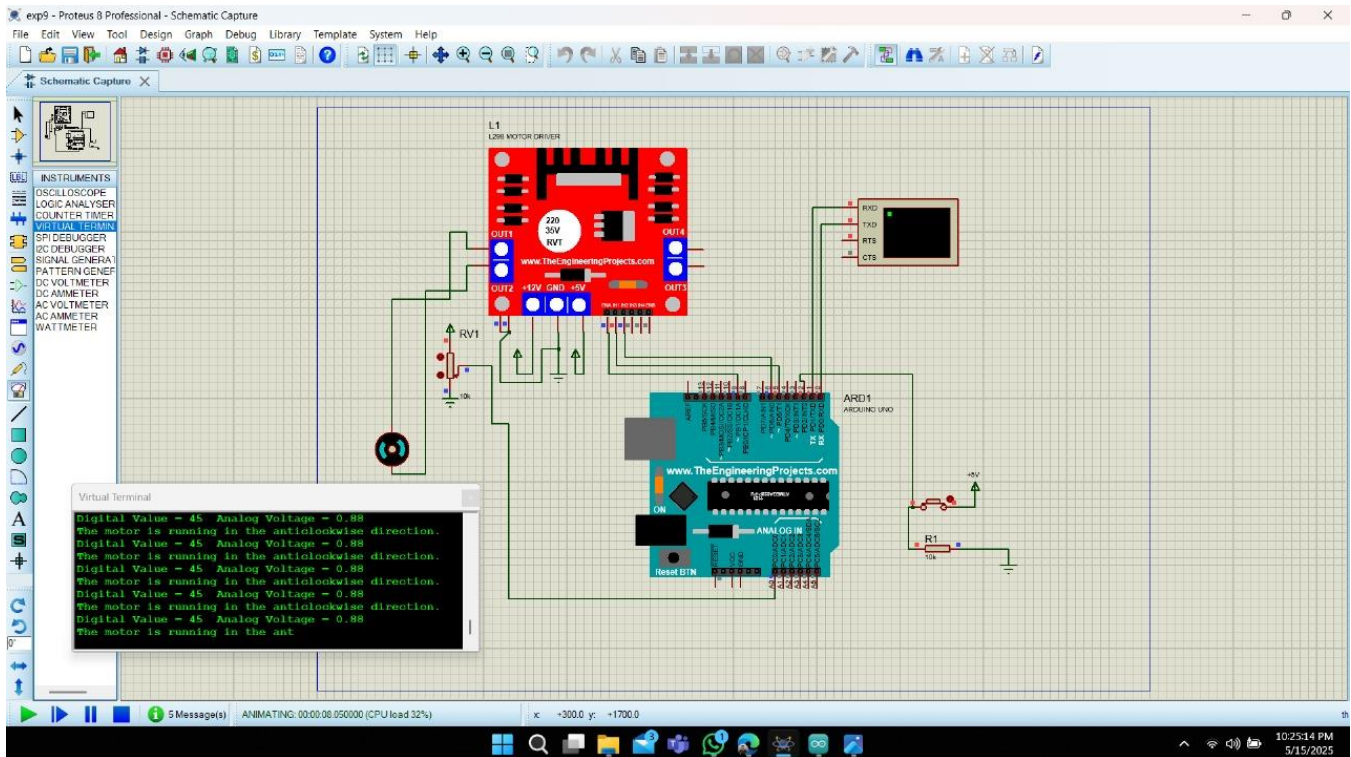


Figure 20: Simulation for Anticlockwise Rotation of DC Motor (Digital Value:45, Voltage:0.88V)

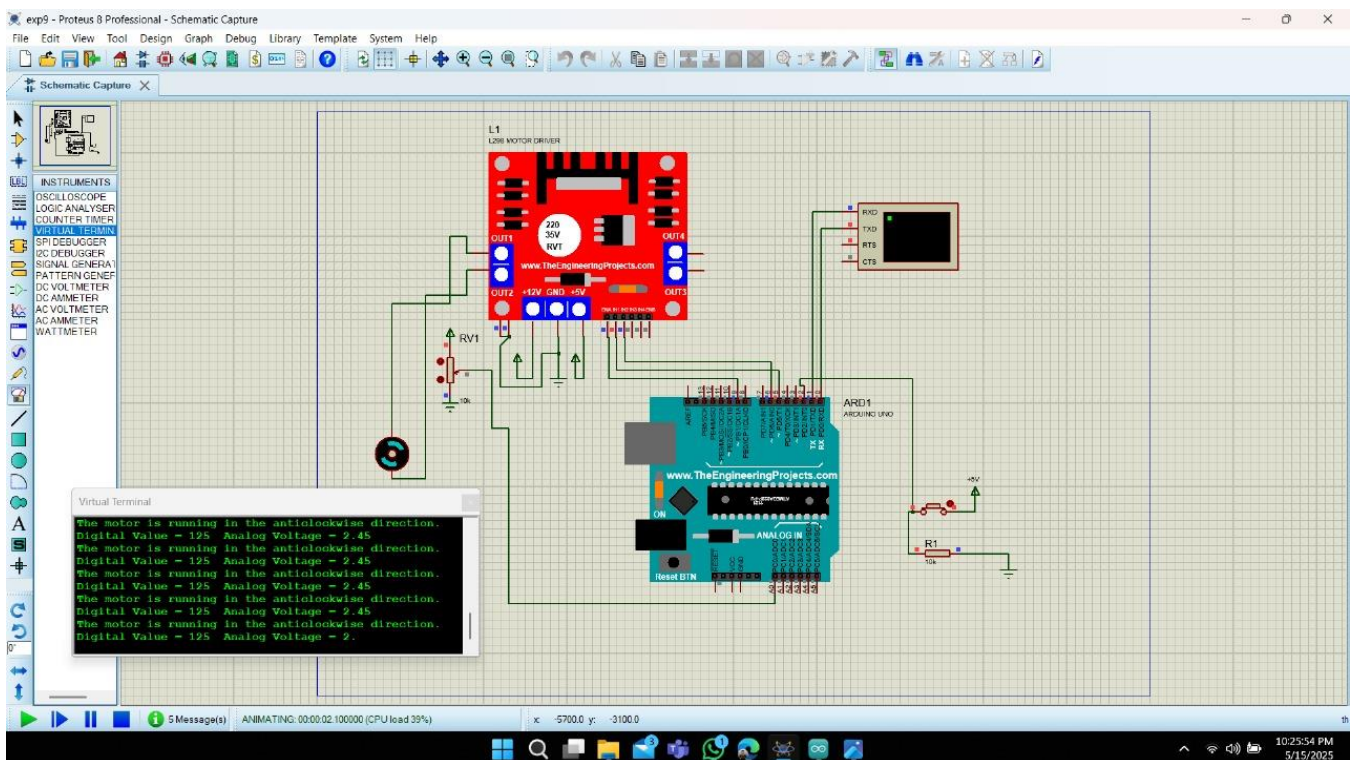


Figure 21: Simulation for Anticlockwise Rotation of DC Motor (Digital Value: 125, Voltage: 2.45V)

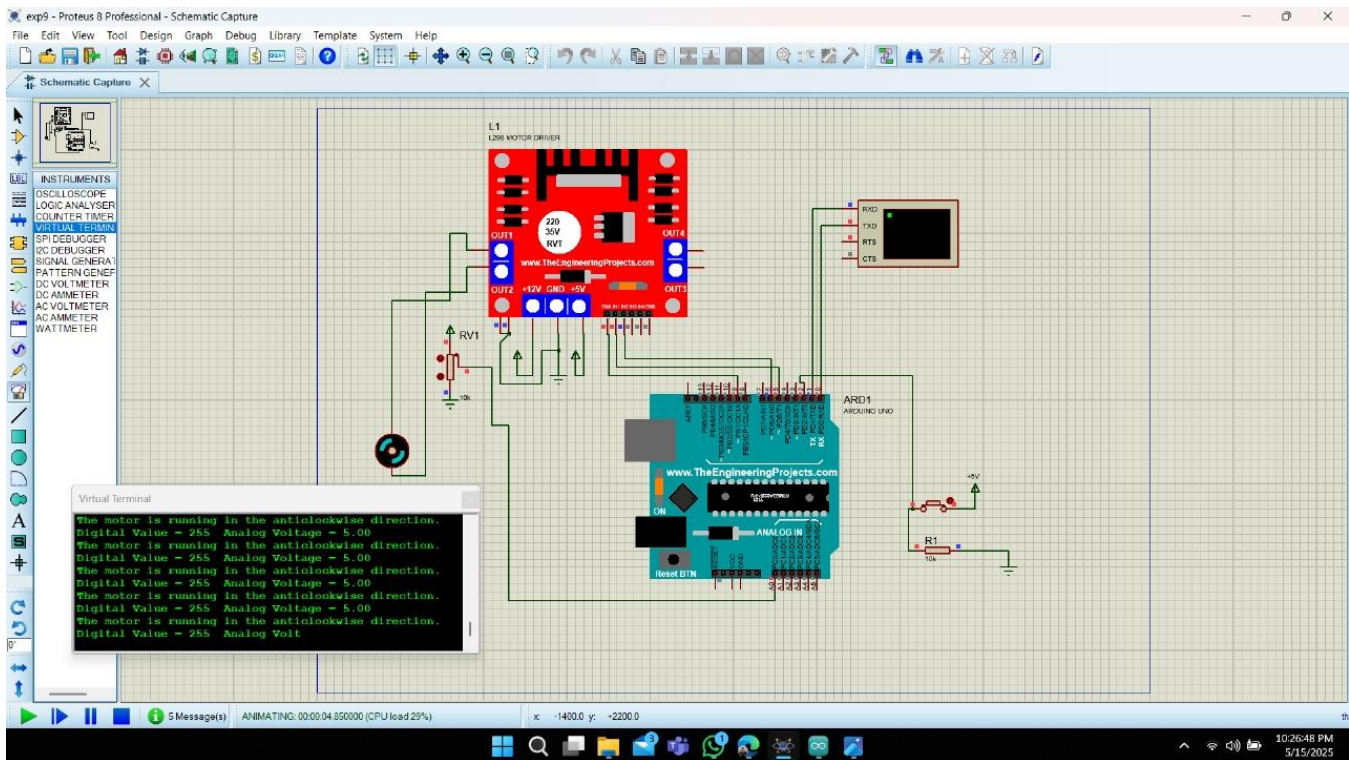


Figure 22: Simulation for Anticlockwise Rotation of DC Motor (Digital Value:255, Voltage:5V)

Explanation: In this simulation, an Arduino Uno, L298N motor driver, push button, potentiometer (RV1), and a DC motor were used to control motor direction and speed. The Arduino code was written in IDE version 2.3.5, and the HEX file was loaded into Proteus for simulation. The potentiometer sets the PWM signal from 0 to 255, controlling the motor's speed, while the push button toggles the rotation direction. When the button is not pressed, the motor runs in the clockwise direction using the `TurnMotorA1()` function. As the potentiometer is adjusted, the motor responds with varying speeds: at a digital value of 45, the analog voltage is 0.88V; at 125, it's 2.45V; and at 255, it reaches 5V, corresponding to low, medium, and maximum speeds respectively. When the button is pressed, the direction changes to anticlockwise via `TurnMotorA2()`, with the same voltage-speed relationship. The virtual terminal continuously displays the motor's direction and PWM values, confirming the system works correctly. The simulation results closely match real hardware behavior, verifying the design and code logic for reliable bidirectional speed control of a DC motor using Arduino and L298N.

Discussion:

This experiment focused on understanding how DC motor control can be achieved using an Arduino by utilizing digital inputs, outputs, and Pulse Width Modulation (PWM). The main objective was to learn how direction and speed of a DC motor can be controlled through programmed logic and variable voltage signals generated by the microcontroller. For this purpose, a simple motor driver circuit was interfaced with the Arduino Uno, which took user input from a digital switch and controlled the motor using two digital output pins and one PWM-enabled output pin.

In the setup, digital pins were used to control the rotation direction of the motor by toggling logic states (HIGH/LOW) at the IN1 and IN2 terminals of the motor driver. PWM was applied to the control pin (ConA), allowing the Arduino to simulate variable voltage output by adjusting the duty cycle, thus varying the motor speed. The system also included an analog voltage input from a potentiometer connected to A0, which was read and converted to a PWM value using the `map()` function. This approach enabled dynamic speed control depending on the analog input. A serial monitor was used to display real-time information including digital PWM values and their corresponding analog voltages, helping to visualize the changes during operation. Additionally, a button connected to a digital input pin acted as a switch to change the rotation direction of the motor.

This hands-on implementation highlighted the importance of PWM in embedded systems where analog-like control is needed using digital hardware. It also demonstrated how digital input signals can be processed in real-time to control output devices like motors effectively. The experiment emphasizes the role of Arduino in building simple, low-cost automation systems that mimic real industrial controls.

Here are some real-life scenarios and applications of this experiment:

1. Robotics and Remote-Controlled Vehicles: DC motors with PWM control allow smooth and efficient movement of wheels or robotic arms.

Example: A line-following robot uses motor direction and speed control to navigate based on sensor input from the environment.

2. Drone Flight Control: Drones use PWM signals to control the speed of the motors that spin the propellers. By adjusting PWM values, the drone can change its altitude, speed, and direction.

Example: A drone adjusts its motor speeds via PWM to maintain stability during flight or to respond to user commands for movement.

3. Water Pumping Systems: PWM is used to control the speed of water pumps in irrigation or water treatment systems, allowing for more energy-efficient operation by adjusting the flow rate as needed.

Example: An irrigation system uses PWM-controlled water pumps to regulate water flow depending on soil moisture levels, ensuring efficient watering.

4. Electric Vehicles: PWM is used in controlling the speed of motors in electric vehicles. By adjusting the duty cycle of the PWM signal, the motor's speed is adjusted, providing smooth acceleration and deceleration.

Example: An electric scooter uses PWM-controlled motors to regulate speed based on user input or road conditions.

5.Smart Fans or Cooling Systems: Temperature-controlled fan systems often use PWM to regulate motor speed based on room or component heat levels.

Example: A CPU cooling fan controlled by Arduino increases speed as temperature rises, maintaining optimal system performance.

6Automatic Window Blinds: PWM-controlled motors can adjust the position of window blinds based on user preferences or time of day, using light sensors to control the angle of the blinds.

Example: A smart home system uses PWM to adjust the angle of window blinds depending on the intensity of sunlight entering the room.

7.Elevators and Lifts: DC motors in elevators use PWM for speed control. This allows for smooth acceleration and deceleration, improving user comfort and reducing mechanical stress.

Example: An elevator adjusts its motor speed using PWM to ensure a smooth start and stop while transporting passengers between floors.

Conclusion:

In conclusion, this experiment demonstrated the fundamental principles of DC motor control using an Arduino, focusing on controlling direction and speed through digital inputs, outputs, and Pulse Width Modulation (PWM). By applying these techniques, we successfully simulated variable voltage control, allowing dynamic adjustments to motor speed based on user input. The integration of a motor driver circuit with the Arduino enabled us to control rotation direction using digital pins and adjust motor speed through PWM signals, offering a practical solution for efficient motor control. This experiment emphasized the versatility and effectiveness of PWM in embedded systems, where precise motor control is essential. It also highlighted real-world applications in fields like automation, robotics, and industrial control, where PWM is widely used for efficient operation. Overall, this hands-on approach reinforced Arduino's value as a powerful, low-cost tool for creating cost-effective and efficient motor control solutions in a variety of practical scenarios.

References:

- [1] <https://www.arduino.cc/>.
- [2] <https://www.educba.com>
- [3] <https://www.researchgate.net/publication/>
- [4] <https://www.geeksforgeeks.org>