LAB REPORT 4

MECHATRONICS SYSTEM INTEGRATION

DR. ZULKIFLI BIN ZAINAL ABIDIN

SECTION 1

GROUP 6

MATRIC NUMBER	NAME
2312369	MUHAMMAD FAHIM BIN AHMAD NORISHAM
2312479	MUHAMMAD HAZIMUDDIN BIN FAIRUZ
2312451	MUHAMMAD ZAMARUL AZIM BIN ZULKHIBRI

DATE OF SUBMISSION

Wednesday, 12th November 2025

Abstract

This experiment dive into how to use the L298P Motor Driver Shield with an Arduino Uno to control the speed and direction of DC motor(s). The goal is to understand how GPIO (General Purpose Input/Output) pins and PWM (Pulse Width Modulation) signals can be used to adjust motor behavior through programming. The L298P shield acts as a connection bridge between the Arduino and the motors, allowing low-power control signals from the Arduino to drive higher-power motors safely and efficiently.

Different PWM values were tested to see how they affected motor speed, and the functions of the enable pins (ENA) were observed. The results showed that higher PWM values increased motor speed, while lower values reduced it. The experiment also demonstrated how switching the input pin logic could reverse the motor’s direction. Through this process, we gained a clearer understanding of how microcontrollers and motor drivers work together for motion control in robotics and automation.

***Keywords*:** L298P Motor Driver Shield, PWM, GPIO, DC Motor, Arduino Uno, Speed Control

Table of Contents

WEEK 5: L298P MOTOR DRIVER SHIELD WITH GPIO

1. Introduction

In modern mechatronic systems, electric motors play a crucial role in automation, robotics, and motion control. Since microcontrollers like the Arduino cannot directly drive motors due to current limitations, a motor driver circuit such as the L298P is required to interface between them. In this lab, we used the **L298P Motor Driver Shield** with an **Arduino Uno** to control DC motor(s) in both forward and reverse directions. We learned how PWM (Pulse Width Modulation) affects the motor's speed and how GPIO pins determine the direction of rotation.

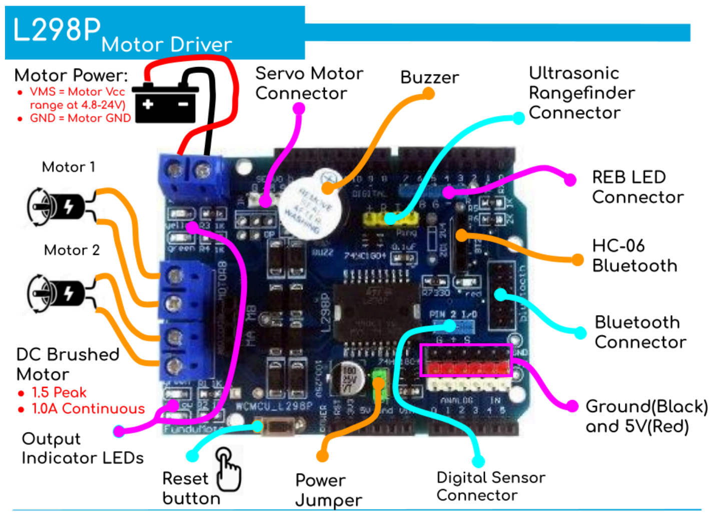


Figure 1.0 Shows the L298P Motor Driver Shield pin

Our main objectives in this experiment were:

- To understand the working principle of the **L298P Motor Driver Shield**.
- To **control the direction** of DC motors using **GPIO pins**.
- To **control the speed** of DC motors using **PWM (Pulse Width Modulation)** signals.
- To **observe the relationship** between PWM values and motor speed.

The L298P is an H-Bridge-based motor driver IC capable of controlling two DC motors simultaneously. It uses digital pins for direction control and PWM-capable pins (ENA and ENB) for speed regulation. The connection of the motor pins shows how the Arduino's digital outputs link to the shield's inputs for direction control, while PWM pins handle speed adjustments.

2. Materials and Equipments

Arduino UNO	1
L298P Motor Driver Shield	1
DC Motor	2
Computer with Arduino IDE	1
Jumper Wires	

3. Experimental Setup

The experiment setup involved **stacking the L298P Motor Driver Shield directly onto the Arduino Uno board**, which provided a simple and secure electrical connection between the two components. **DC motors** were then connected to the Motor A or Motor B terminals on the shield. To supply sufficient power for motor operation, the supply voltage from the laptop is sufficient (it might be different depending on the motor's operating voltage). This ensured that the motors received enough current for smooth and reliable movement without overloading the Arduino itself.

The Arduino IDE was used to write and upload the control program to the Arduino Uno. The program used **digital pins (IN1–IN2)** to determine the direction of motor rotation and **PWM pins (ENA)** to adjust the motor speed. By changing the PWM values in the code, the motors could spin at different speeds, while modifying the logic levels of the input pins controlled whether each motor rotated forward or backward.

This configuration provided a simple and effective way to study how electronic control systems manage both the direction and speed of DC motors using a motor driver shield and Arduino microcontroller. The schematic diagram of connection is shown below:

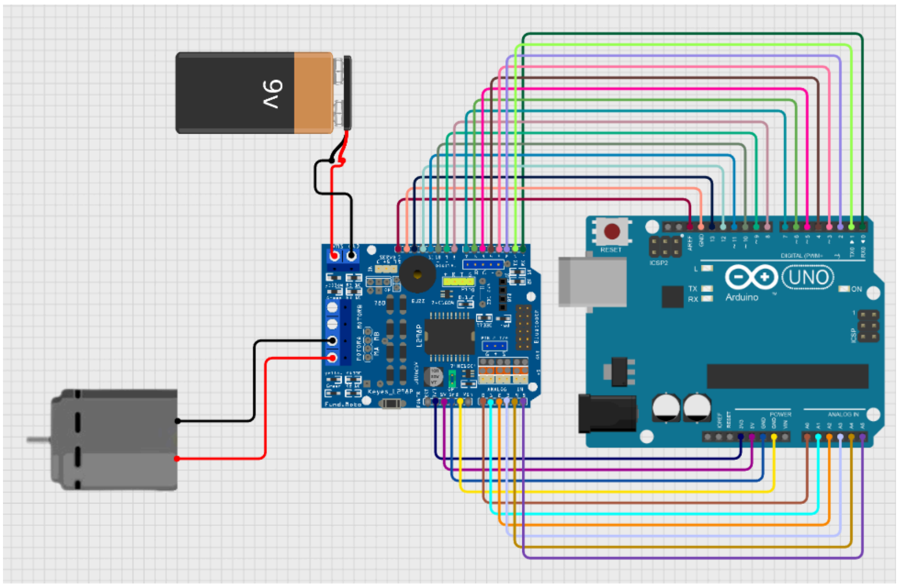


Figure 3.0 Shows the schematic diagram using Cirkit Designer

4. Methodology

4.1. Circuit Assembly

To set up the experiment, the L298P Motor Driver Shield was securely mounted on top of the Arduino Uno board, ensuring all the pins aligned correctly. DC motors were connected to the motor driver shield which 1 DC motor to the output terminal labeled Motor A or Motor B terminal. An external power supply is used from the laptop only since the motor can run with the voltage. The voltage of the DC motor is also important to ensure the motors receive enough current for smooth operation without overloading the Arduino.

The direction control pins (IN1–IN2) were automatically linked to the Arduino’s GPIO pins through the shield’s header connections, while the ENA pins were assigned as PWM outputs for controlling motor speed. This configuration allowed the Arduino to send both logic signals (for direction) and PWM signals (for speed) to the motor driver. A circuit schematic was used to visualize how each component was connected, showing the relationship between the Arduino pins, the motor driver shield, and the DC motor.

4.2. Serial Communication Setup

Once the circuit was fully assembled, the Arduino Uno was connected to a computer via USB cable. The Arduino IDE was opened to write, load, and upload the program controlling the motors. The provided code included commands to set the direction of rotation using the digital pins and to control motor speed using PWM values.

After uploading the code to the Arduino, the motors were observed as they spin clockwise, paused, and then reversed direction according to the programming code. To further analyze motor performance, the PWM values were modified to 64, 128, 200, and 255, which produced different rotation speeds and recorded in **Table 5.0** and **Table 7.0**.. As the PWM value increased, the motor rotated faster. Each observation such as motor direction, response time, and speed change was carefully recorded in a data table for later analysis and comparison.

4.3 Code Upload to Arduino IDE

The Arduino program was written and uploaded using the Arduino IDE. The coding is to control the direction and speed of DC motor through the L298P Motor Driver Shield. Direction control was managed using digital output pins (IN1–IN2), while PWM pins (ENA) were used to adjust motor speed.

```
// --- Motor A pins ---
int IN1 = 12;
int IN2 = 13;
int ENA = 10;

// --- Simulated RPM settings ---
const float maxRPM_12V = 10000.0;   // Motor rated speed at
12V
const float supplyVoltage = 5.0;     // Laptop USB output
voltage
const float ratedVoltage  = 12.0;
float maxRPM = (supplyVoltage / ratedVoltage) * maxRPM_12V;
// Scaled for 5V


void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  Serial.begin(9600);
  Serial.println("DC Motor (A) RPM Simulation Starting...");
  Serial.print("Estimated Max RPM at 5V: ");
  Serial.println(maxRPM);
}

void loop() {
  // === Forward motion ===
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 255); // full power
  printSimulatedRPM(255);
  delay(2000);
  // === Stop ===
  analogWrite(ENA, 0);
  printSimulatedRPM(0);
  delay(1000);

  // === Reverse motion ===
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, 255);
  printSimulatedRPM(255);
  delay(2000);

  // === Stop again ===
  analogWrite(ENA, 0);
  printSimulatedRPM(0);
  delay(1000);
}

void printSimulatedRPM(int pwmVal) {
  float rpm = (pwmVal / 255.0) * maxRPM;

  Serial.print("Motor A (simulated): ");

  Serial.print(rpm);

  Serial.println(" RPM");

}
```

5. Data Collection

During the experiment, data was collected to observe how varying the **PWM (Pulse Width Modulation)** values affected the rotation speed and direction of the DC motors. The Arduino program was modified several times to change the PWM values applied to both motors while keeping the direction control pins constant. The motors were allowed to run for a few seconds under each condition, and the estimated rotational speeds were observed visually and by comparing motion rates. The results were recorded in the table below.

PWM Value	Direction	Observed Speed	Observation
255	Forward	Very fast	Very fast rotation with strong torque and smooth motion
128	Forward	Medium	Medium speed with stable and consistent rotation
255	Reverse	Very Fast	Very fast rotation in reverse direction with similar torque
64	Reverse	Slow	Slow rotation; motor moved steadily but with less torque

Table 5.0 Shows the observation speed of DC motor

6. Data Analysis

From the observation table, it was clear that the **motor speed increased when the PWM value increased**. When the PWM value was set to 255, the motors spun at their fastest speed. At lower PWM values, such as 64, the motors turned much slower. This shows that changing the PWM value directly affects how fast the motors rotate.

The PWM (Pulse Width Modulation) signal plays a key role in controlling motor performance. Instead of continuously supplying a steady voltage, PWM rapidly switches the power on and off. The ratio of the "on" time to the total cycle time that is known as the duty cycle, **determines how much effective voltage the motor receives.** A higher duty cycle means the motor receives power for a longer portion of each cycle, resulting in faster motion. A lower duty cycle shortens the power pulse, reducing the motor's average voltage and slowing its speed.

The experiment also showed that changing the direction of the motor worked correctly. By switching the HIGH and LOW signals on the control pins, the motors were able to move in reverse. This proved that the L298P Motor Driver Shield's H-Bridge circuit was working properly and could make the motors spin in both directions.

Overall, the results matched what was expected, which is that higher PWM values made the motors faster, and changing the pin signals reversed their direction. This experiment helped us understand how the Arduino can control both motor speed and direction using simple programming and the motor driver shield.

7. Results

With the **constraints faced**, specifically the lack of measurement equipment such as a tachometer and the limited ability of the eyes and smartphone cameras to capture the motor's rotations per minute, the **ratio method was used,** and its formula was **implemented in the code**. The results were recorded in the table below.

PWM Value	**Direction**	**Speed from Serial Monitor (RPM)**	**Observation**
255	Forward	4166.67	Very fast rotation with strong torque and smooth motion
128	Forward	2091.50	Medium speed with stable and consistent rotation
255	Reverse	4166.67	Very fast rotation in reverse direction with similar torque
64	Reverse	1045.75	Slow rotation; motor moved steadily but with less torque

Table 7.0 Shows the observed speed of the DC motor

From the collected data shown in **Table 7.0**, it was clear that the **PWM value directly affected the motor speed**. A higher PWM duty cycle (closer to 255) provided more average voltage to the motor, resulting in faster rotation. Conversely, lower PWM values (like 64) reduced the average voltage, causing the motor to spin slower. The direction of rotation was reversed by swapping the logic levels of the digital direction pins.

The DC motor demonstrated consistent performance, showing that the **L298P Motor Driver Shield** effectively handled current distribution for bidirectional control. These results confirmed the theoretical relationship between PWM signal strength and motor speed, as well as the proper functionality of the H-Bridge configuration within the shield.

8. Discussion

In this experiment, the Arduino Uno was programmed to control the direction and speed of the DC motor by sending signals to the L298P Motor Driver Shield. The direction of the motor was controlled using the digital output pins (IN1 and IN2), where one combination of HIGH and LOW caused the motor to rotate forward, and the opposite combination caused it to rotate in reverse. This confirmed the functionality of the H-Bridge circuit built into the L298P driver, which enables bidirectional control of current flow through the motor.

Speed control was achieved using PWM (Pulse Width Modulation) applied to the ENA or ENB pin. By adjusting the PWM value between 0 and 255, the duty cycle of the signal changed. A higher duty cycle means the motor receives power for a longer duration in each cycle, resulting in a faster speed. Conversely, a lower duty cycle provides less power, causing the motor to rotate slower. This was clearly observed in the experiment, where PWM values of 255, 128, and 64 produced noticeable changes in motor speed.

Since a tachometer was not available during the experiment, the motor's rotational speed (RPM) was **estimated using a scaling ratio** based on the difference between the motor's rated voltage and the experiment's supply voltage. The formula used in the code:

$$\text{maxRPM}= \left(\frac{supplyVoltage}{ratedVoltage}\right)\times \text{maxRPM\_12V}$$

allowed the maximum possible RPM at 5V voltage supply from laptop to be calculated. The actual RPM at each PWM level was then estimated proportionally using:

$$\text{estimatedRPM}= \left(\frac{PWM}{255}\right)\times \text{maxRPM}$$

This method provided consistent trends in the motor's estimated speed even though exact measured values could not be obtained.

Overall, the experiment successfully demonstrated the relationship between PWM duty cycle and motor speed, as well as how GPIO logic controls direction. The observed results matched theoretical expectations and confirmed that the L298P shield is effective for driving DC motors in mechatronic applications.

8.1 Questions

8.1.1. Function of ENA and ENB Pins:

The ENA and ENB pins are the enable pins used to control the speed of Motor A and Motor B respectively on the L298P Motor Driver Shield. Both pins are connected to PWM-capable pins on the Arduino, allowing the microcontroller to send PWM signals to vary the motor speed.

- ENA → Controls Motor A (speed)
- ENB → Controls Motor B (speed)

Even though this experiment focused mainly on Motor A at first, the shield is designed to control two motors simultaneously, which is why both ENA and ENB exist. When controlling two motors, each motor can have different speeds because ENA and ENB are adjusted independently using separate PWM values.

Pin	Controls	Usage
ENA	Motor A	`analogWrite(ENA, pmw Value)`
ENB	Motor B	`analogWrite(ENB, pmw Value)`

Thus, ENA and ENB **make it possible to drive two motors with independent speed control.**

8.1.2. Reason PWM is Used for Speed Control:

PWM (Pulse Width Modulation) is used for speed control because it allows the Arduino to **adjust the motor speed without changing the actual supply voltage.** The PWM signal rapidly switches the motor's power ON and OFF. The **duty cycle** (percentage of ON time) determines the effective voltage delivered to the motor.

- Higher duty cycle → more average voltage → faster speed
- Lower duty cycle → less average voltage → slower speed

PWM is **efficient**, prevents motor overheating, and allows **smooth and precise speed control**, which is why it is preferred over just lowering voltage.

8.1.3. Outcome When Both IN1 and IN2 Are Set HIGH:

When **IN1 and IN2 are both HIGH**, both ends of the motor receive the same voltage level. This means **there is no voltage difference across the motor terminals**, causing the motor to **stop instantly**. This condition is called **Active Braking** or **Short Brake.** Instead of slowing down gradually (coasting), the motor **halts abruptly** because current flow is forced to stop.

8.1.4. How Braking Can Be Implemented Using the L298P:

Braking on the L298P can be achieved by using the H-Bridge configuration to create an **electrical short across the motor terminals**, causing the motor to stop immediately. To do this, **both direction pins (IN1 and IN2) are set to the same logic level**, either both HIGH or both LOW. This removes the voltage difference across the motor and forces it to stop rapidly.

```
// === Active Braking (Instant Stop) ===
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, 0);
```

In contrast, simply setting:

```
  analogWrite(ENA, 0);
```

will only **cut the power**, causing the motor to **coast** and slow down gradually. This is **not** braking, it is **free spinning**.

Therefore:

- **Coasting Stop** → Motor slows naturally
- **Active Braking** → Motor stops instantly using H-Bridge shorting

8.1.5. Modify the Code to Control Two DC Motors Simultaneously:

When controlling two motors, each motor requires its own direction pins and enable pin:

Motor	Direction Pins	Speed Pin
Motor A	IN1 & IN2	ENA
Motor B	IN3 & IN4	ENB

```
// Run 2 DC Motors (Motor A & B) in sync on L298P
shield

// Motor A pins
int IN1 = 12;
int IN2 = 13;
int ENA = 10;

// Motor B pins
int IN3 = 8;
int IN4 = 9;
int ENB = 11;

void setup() {
  // Set all pins as outputs
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);
}
void loop() {
  // === Forward ===
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  analogWrite(ENA, 255);  // Speed (0-255)
  analogWrite(ENB, 255);
  delay(2000);

  // === Stop ===
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  delay(1000);

  // === Reverse ===
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);

  analogWrite(ENA, 255);
  analogWrite(ENB, 255);
  delay(2000);

  // === Stop again ===
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  delay(1000);
}
```

9. Conclusion

In conclusion, this experiment provided practical understanding of how the L298P Motor Driver Shield interfaces with the Arduino Uno to control DC motors. The L298P module enabled safe current handling, while the Arduino supplied logic signals for direction and PWM signals for speed. By adjusting the PWM duty cycle, the speed of the motor increased or decreased accordingly, and by toggling the direction input pins, the motor was able to reverse its rotational direction.

Although a tachometer was not available, the use of a ratio-based RPM estimation method still allowed us to analyze speed trends relative to PWM input. The results clearly showed that higher PWM values resulted in faster rotational speeds and lower values produced slower speeds. The experiment achieved all objectives, demonstrating successful integration of microcontrollers and motor drivers for motion control in mechatronic systems.

10. Recommendations

1. Use a Digital Tachometer

In this experiment, the rotational speed of the motor was estimated using a mathematical ratio because a tachometer was not available. While the estimation method provided a reasonable indication of speed trends, it did not give the exact RPM. In future experiments, a laser tachometer or optical encoder should be used to obtain accurate and reliable RPM measurements. This would allow for better analysis and provide more precise validation between theoretical and experimental results.

2. External Power Supply for Higher Torque

The motor in this experiment was powered using the USB 5V supply from the laptop. Although this was sufficient to rotate the motor, it limited the maximum obtainable speed and torque, especially when load or friction increases. Connecting the L298P shield to a separate 9V or 12V DC supply would allow the motor to operate closer to its rated performance, producing stronger torque, smoother motion, and more stable speed control.

3. Implement Real-Time Speed Feedback Using a Rotary Encoder

Currently, the motor speed is controlled in open-loop mode, meaning there is no feedback to check whether the speed matches the intended value. By adding a rotary encoder to the motor shaft, the system can measure actual RPM in real-time and adjust the PWM accordingly. This enables closed-loop (feedback) speed control, which is commonly used in robotics, conveyor systems, and automation to maintain precise and consistent speed even under varying load conditions.

4. Add Soft-Start and Soft-Stop Control for Smoother Motor Operation

The motor speed changes instantly when different PWM values are applied, which may cause mechanical stress, jerky motion, and higher current spikes. Implementing a gradual increase or decrease in PWM (ramping) would allow the motor to start and stop more smoothly. This not only improves control performance but also extends the motor's operational lifespan.

5. References

Cirkit Design. “How to Use the L298P Drive Shield.” *Cirkitdesigner.com*, 2025, docs.cirkitdesigner.com/component/a25f6e70-294e-42fd-b77c-9e9349b76b9a/l298p-drive-shield. Accessed 6 Nov. 2025.

CytronTechnologies. “GitHub - CytronTechnologies/L298P_MotorDriverShield.” *GitHub*, 2025, github.com/CytronTechnologies/L298P_MotorDriverShield. Accessed 6 Nov. 2025.

Instructables. “Tutorial for L298 2Amp Motor Driver Shield for Arduino.” *Instructables*, 15 Nov.2017, www.instructables.com/Tutorial-for-L298-2Amp-Motor-Driver-Shield-for-Ard/. Accessed 7 Nov. 2025.

“Shield L298P Motor Driver with GPIO.” *Cytron Technologies Malaysia*, 2025, my.cytron.io/c-dc-motor-driver/ampp-shield-l298p-motor-driver-with-gpio?gad_campaignid=21520410146. Accessed 8 Nov. 2025.

Appendix A - Circuit Diagram for 2 DC Motors

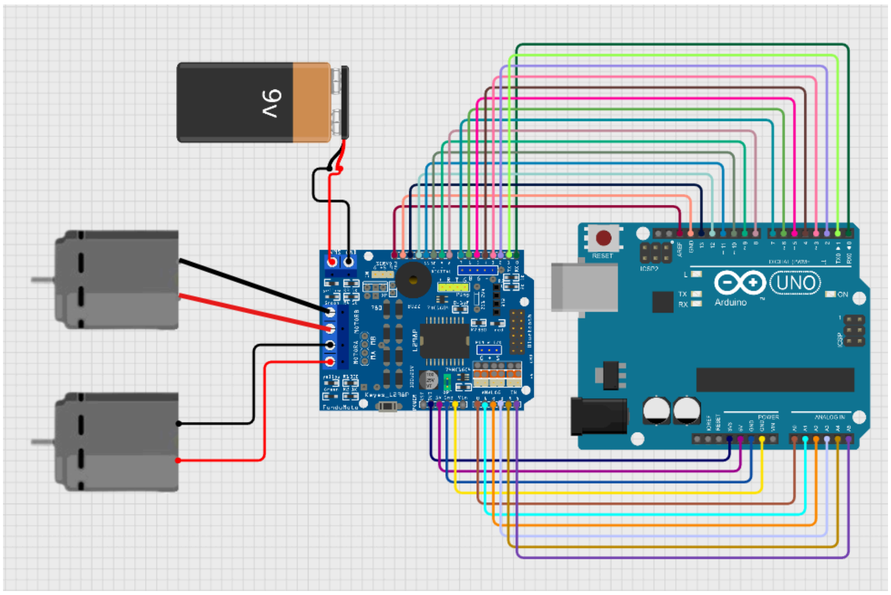


Figure A1 Shows the connection of circuit configuration by using Cirkit Designer

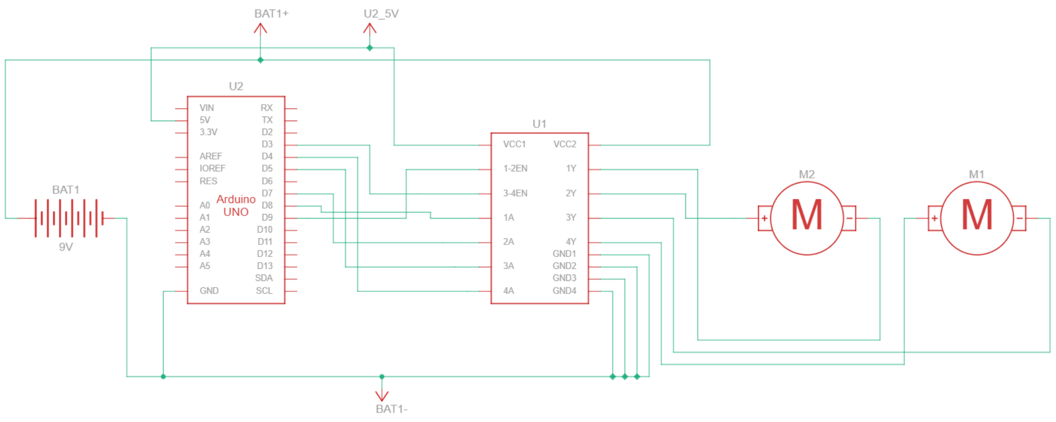


Figure A2 Shows the schematic diagram by using Tinkercad

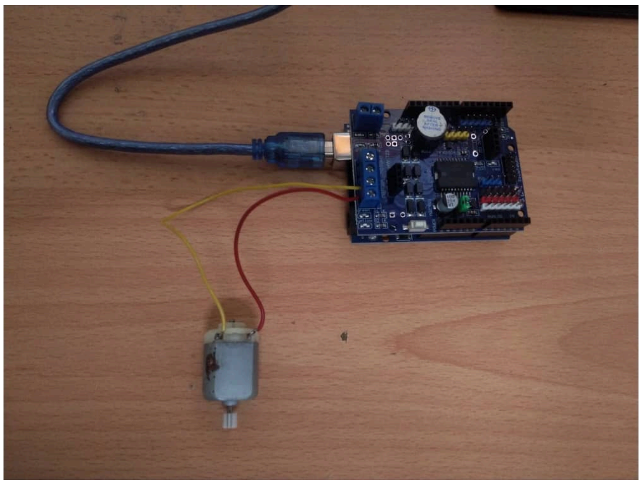


Figure A3 Shows the connection for 1 motor

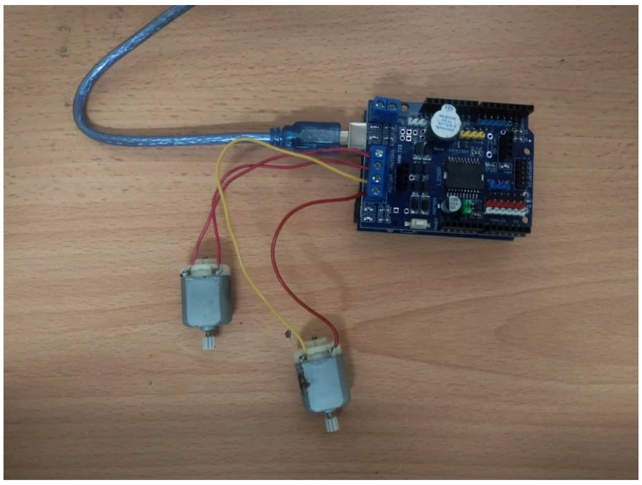


Figure A4 Shows the connection for 2 motors

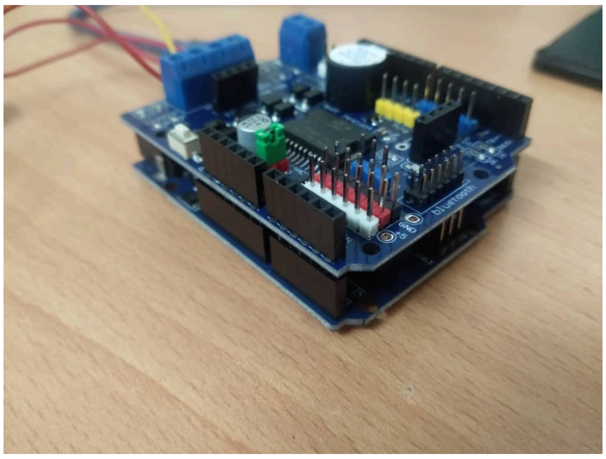


Figure A4 Shows the motor driver attached on top of the Arduino UNO

Appendix B - Coding Snippets

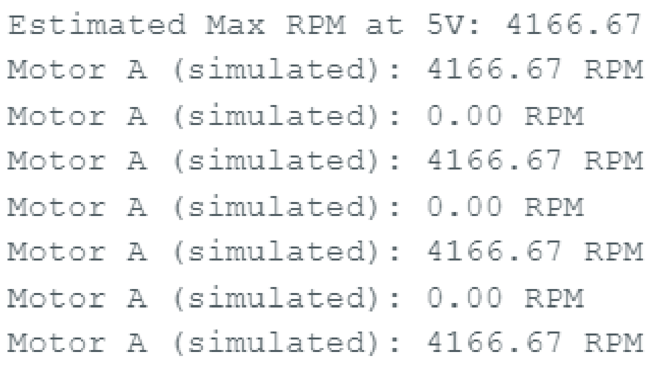
B1 Arduino IDE Coding for controlling 1 DC motor

```
// --- Motor A pins ---
int IN1 = 12;
int IN2 = 13;
int ENA = 10;

// --- Simulated RPM settings ---
const float maxRPM_12V = 10000.0;   // Motor rated speed at
12V
const float supplyVoltage = 5.0;     // Laptop USB output
voltage
const float ratedVoltage  = 12.0;
float maxRPM = (supplyVoltage / ratedVoltage) * maxRPM_12V;
// Scaled for 5V


void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  Serial.begin(9600);
  Serial.println("DC Motor (A) RPM Simulation Starting...");
  Serial.print("Estimated Max RPM at 5V: ");
  Serial.println(maxRPM);
}


void loop() {
  // === Forward motion ===
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 255); // full power
  printSimulatedRPM(255);
  delay(2000);
  // === Stop ===
  analogWrite(ENA, 0);
  printSimulatedRPM(0);
  delay(1000);

  // === Reverse motion ===
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, 255);
  printSimulatedRPM(255);
  delay(2000);

  // === Stop again ===
  analogWrite(ENA, 0);
  printSimulatedRPM(0);
  delay(1000);
}

void printSimulatedRPM(int pwmVal) {
  float rpm = (pwmVal / 255.0) * maxRPM;

  Serial.print("Motor A (simulated): ");

  Serial.print(rpm);

  Serial.println(" RPM");

}
```

B2 Arduino IDE Coding for controlling 2 DC motor

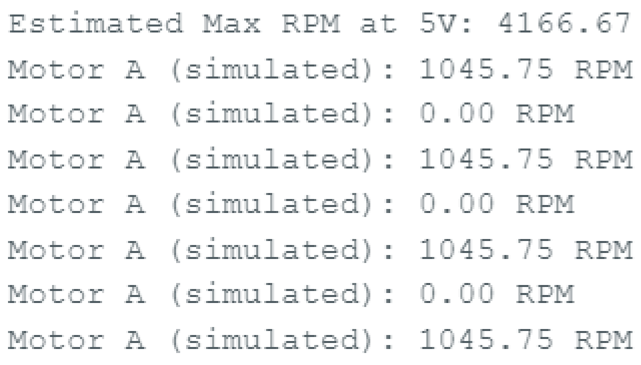
```
// Run 2 DC Motors (Motor A & B) in sync on L298P shield

// Motor A pins
int IN1 = 12;
int IN2 = 13;
int ENA = 10;

// Motor B pins
int IN3 = 8;
int IN4 = 9;
int ENB = 11;

void setup() {
  // Set all pins as outputs
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);
}

void loop() {
  // === Forward ===
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  analogWrite(ENA, 255);  // Speed (0-255)
  analogWrite(ENB, 255);
  delay(2000);

  // === Stop ===
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  delay(1000);

  // === Reverse ===
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);

  analogWrite(ENA, 255);
  analogWrite(ENB, 255);
  delay(2000);

  // === Stop again ===
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  delay(1000);
}
```

Appendix C - Results



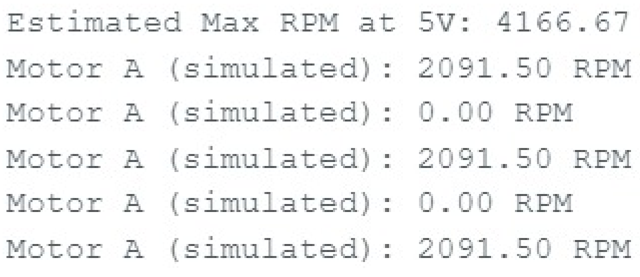
```
Estimated Max RPM at 5V: 4166.67
Motor A (simulated): 4166.67 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 4166.67 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 4166.67 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 4166.67 RPM
```

Figure C1 Shows the output at serial monitor for PWM set at 255



```
Estimated Max RPM at 5V: 4166.67
Motor A (simulated): 1045.75 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 1045.75 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 1045.75 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 1045.75 RPM
```

Figure C2 Shows the output at serial monitor for PWM set at 128



```
Estimated Max RPM at 5V: 4166.67
Motor A (simulated): 2091.50 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 2091.50 RPM
Motor A (simulated): 0.00 RPM
Motor A (simulated): 2091.50 RPM
```

Figure C3 Shows the output at serial monitor for PWM set at 64

Acknowledgments

We would like to thank our lecturer and lab instructor for their kind guidance and support throughout this experiment. Their clear explanations about how to use the L298P Motor Driver Shield and Arduino Uno helped us understand how to control the speed and direction of DC motors. They also gave us helpful advice during the lab session, which made it easier to complete the circuit connection and run the program correctly.

We are also very thankful to the lab assistants for helping us set up the equipment and check our wiring before testing. Their help made sure that everything was connected safely and worked as expected.

We would like to thank all our team members for their teamwork and effort in completing this experiment. Everyone took part in assembling the circuit, writing the code, testing the motors, and collecting the data. Working together made the experiment smoother and more enjoyable.

Lastly, we want to thank our classmates for sharing ideas, components, and helping each other during the lab session. The cooperation and teamwork among everyone made this experiment successful. Overall, we are truly grateful to our lecturer, lab assistants, teammates, and classmates for their support and guidance throughout this lab.

Certificate of Originality and Authenticity

This is to certify that we are **responsible** for the work submitted in this report, that **the original work** is our own except as specific in references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or a person.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us**.

Signature: [illegible]		Read	/
Name: Muhammad Fahim Bin Ahmad Norisham		Understand	/
Matric Number: 2312369		Agree	/

Signature: [illegible]		Read	/
Name: Muhammad Zamarul Azim Bin Zulkhibri		Understand	/
Matric Number: 2312451		Agree	/

Signature: [illegible]		Read	/
Name: Muhammad Hazimuddin Bin Fairuz		Understand	/
Matric Number: 2312479		Agree	/