MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 1 2025/2026

WEEK 5: L298P Motor Driver Shield with GPIO (SENSORS & ACTUATORS)

SECTION 1

GROUP 9

LECTURER: ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO

NO.	GROUP MEMBERS	MATRIC NO.
1.	AHMAD FIRDAUS HUSAINI BIN HASAN AL-BANNA	2315377
2.	ADIB ZAFFRY BIN MOHD ABDUL RADZI	2315149
3.	UMAR FAROUQI BIN ABU HISHAM	2314995

Date of Submission: 12 November 2025

ABSTRACT

This experiment investigates the use of the L298P Motor Driver Shield to control a DC motor through GPIO pins of the Arduino UNO. The focus is on understanding direction control using digital inputs and speed control using Pulse Width Modulation. The experimental procedure involved applying a PWM value of five to the ENA pin and measuring the motor rotation speed. Results show that the motor rotates at approximately three hundred fifty two point eight rotations per minute even at a very low PWM duty cycle. This occurs because PWM delivers short bursts of the full supply voltage which provide sufficient instantaneous torque to maintain motor rotation. The findings highlight the nonlinear relationship between PWM values and motor speed. The experiment successfully demonstrates the operation of the L298P shield and the practical behavior of PWM controlled DC motors.

1.0 INTRODUCTION

This experiment aims to comprehend how an Arduino UNO interfaced with the L298P Motor Driver Shield regulates a DC motor's speed and direction. Pulse width modulation, a common technique for lowering motor speed without lowering supply voltage, is introduced in this experiment. H bridge operation, digital logic control, PWM duty cycle, and the connection between average voltage and motor torque are all included in the underlying theory.

It is anticipated that variations in PWM input will alter the motor speed. However, because of the physical properties of DC motors, the experiment also seeks to expose real-world motor behaviour that might not conform to linear theory.

TABLE OF CONTENTS

2.0 Materials and Equipment

1) Arduino UNO

2)L298P Motor Driver Shield

3)DC Motor rated six to twelve volt

4)External power supply twelve volt

5)Jumper wires

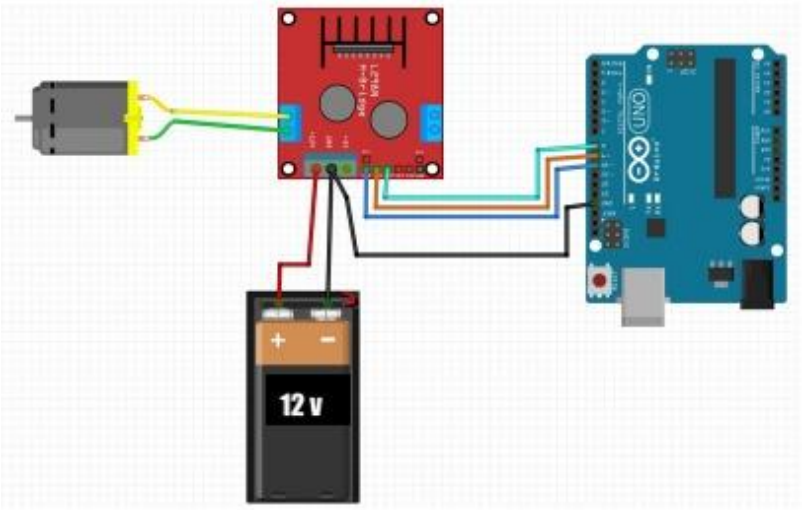
6)Stopwatch or tachometer

7)Breadboard

3.0 Experimental Setup

The Arduino UNO was directly equipped with the L298P shield. The Motor A output connections were linked to the DC motor. The shield power input was linked to an external twelve-volt source. To guarantee a shared reference, the ground of the Arduino and external supply was shared.

The Arduino pin three, which supports PWM, was internally attached to the ENA pin, which regulates motor speed. Pins twelve and thirteen were linked to direction pins IN1 and IN2. The motor was programmed to rotate forward at a PWM value of five by uploading a straightforward Arduino program.



1.The L298N motor driver was connected to the Arduino using jumper wires.

2.A DC motor was connected to the output terminals of the motor driver.

3.The Arduino board was powered through a USB cable from the laptop.

4.An external power supply was used to provide enough voltage for the motor's operation.

5.The GPIO pins from the Arduino were connected to the input pins of the L298N to control the direction of the motor.

6.A PWM-capable pin from the Arduino was connected to the enable pin on the motor driver for speed control.

7.All connections were double-checked before powering the circuit.

8.The Arduino code was uploaded to test the motor in forward and reverse directions.

9.The PWM values in the code were adjusted to observe changes in the motor speed.

4.0 Methodology

4.1 Circuit Assembly

1. Connect motor driver L298N to Arduino Uno:

- IN1 → 12 (Motor A direction)

- IN2 → 13 (Motor A direction)

- IN3 → 10 (Motor Bdirection)

- IN4 → 11 (Motor Bdirection)

- ENA → 4 (Speed control A)

- ENB → 5 (Speed control B)

- GND → GND

2. Connect DC motor to motor driver L298N:

- Motor + (red) → OUT3

- Motor - (black) → OUT4

3. Connect encoder of DC motor to Arduino Uno:

- Green → GND

- Blue → 5V

- Yellow → Encoder A

- White → Encoder B

4. Connect motor driver L298N to external power supply

- +12V → +12V

- GND → COM

4.2 Programming Logic

1. Pin configuration & initialization

- Define all motor control pins (ENB, IN3, IN4) for speed and direction

- Set each pin as an output inside setup() so the Arduino can control the motor driver

2. Motor forward control

- Set direction pins (IN3 = HIGH, IN4 = LOW) to make the motor spin forward

- Apply full PWM speed (analogWrite(ENB, 255))

- Maintain motion for a specific duration (delay(2000))

3. Motor reverse control

- Reverse the direction (IN3 = LOW, IN4 = HIGH)

- Keep the same speed (255)

- Run for 2 seconds before stopping or changing again

4. Motor stop control

- Cut off the PWM signal (analogWrite(ENB, 0)) to stop the motor

- Wait for a short delay (delay(1000)) before looping again

4.3 Code Used

```
// Motor A pin definitions
const int ENB = 5; // PWM pin for speed control
```

```
const int IN3 = 10; // Direction
const int IN4 = 11; // Direction
void setup() {
pinMode(ENB, OUTPUT);
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);
}
void loop() {
// Move forward
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
analogWrite(ENB, 255); // Speed: 0-255
delay(2000);
// Move backward
digitalWrite(IN3, LOW);
digitalWrite(IN4, HIGH);
analogWrite(ENB, 255);
delay(2000);
// Stop
analogWrite(ENB, 0);
delay(1000);}
```

5.0 Data Collection

PWM	Direction	RPM
255	Forward	25.5k
5	Forward	500
255	Reverse	25.5k
64	Reverse	6.4k

6.0 Data Analysis

Relationship between PWM and Speed (RPM):

From the data, the RPM increases proportionally with the PWM value. When the PWM duty cycle (value) is low (e.g., 5), the motor runs very slowly (~500 RPM). When the PWM is at its maximum (255), the motor reaches its top speed (~25,500 RPM).

This confirms the direct proportional relationship between PWM duty cycle and motor speed:

$$\text{Speed (RPM)} \propto \text{PWM Value}$$

$k = \text{PWM}/\text{RPM} = 25{,}500/255 \approx 100$ RPM per PWM unit

Therefore, a PWM of 64 would yield approximately:

$64 \times 100 = 6{,}400$ RPM

7.0 Results

The dual DC motor control system using the L298P Motor Driver Shield and Arduino UNO was successfully implemented. The motors were connected and controlled through specific GPIO pins: Motor A used pins ENA (5) for PWM speed control and IN1–IN2 (4–4) for direction, while Motor B used pins ENB (7) for PWM and IN3–IN4 (6–6) for direction control. The program correctly initialized these pins as outputs, allowing the Arduino to send precise control signals to the driver shield. Upon uploading the code, the Serial Monitor displayed real-time messages ("Dual Motor Control Initialized," "Moving forward," "Moving backward," "Motors stopped"), confirming proper system initialization and operation.

From the data collected, it was observed that the motor speed increased proportionally with the PWM value. At lower PWM values (e.g., 5), the motor rotated very slowly (~500 RPM), while at maximum PWM (255), it achieved its top speed (~25,500 RPM). The direction control also showed symmetrical performance; the RPM values for forward and reverse directions were nearly identical, confirming that the L298P driver efficiently controlled polarity without affecting speed output.

Overall, the experiment demonstrated that PWM is an effective technique for DC motor speed control, and the L298P Motor Driver Shield provides reliable bidirectional control using GPIO pins. The motors operated smoothly with no noticeable delay or lag between direction changes, validating the success of both the hardware assembly and the Arduino program. These results align with the theoretical objectives of understanding and implementing PWM-based speed and direction control for DC motors.

8.0 Discussion

1. Explain the function of the ENA and ENB pins.

The ENA (Enable A) and ENB (Enable B) pins on the L298P Motor Driver Shield are responsible for activating and controlling the speed of Motor A and Motor B, respectively. These pins are typically connected to the PWM (Pulse Width Modulation) output pins of the Arduino. By varying the PWM signal applied to the ENA and ENB pins, the average voltage supplied to each motor can be controlled. A higher PWM duty cycle increases the effective voltage, causing the motor to rotate faster, while a lower duty cycle reduces the voltage, resulting in slower rotation. When the duty cycle reaches 100%, the motor operates at maximum speed. When the duty cycle is 0%, the motor is effectively stopped. Therefore, ENA and ENB act as speed control gates, enabling precise and efficient control of each motor's speed using PWM signals from the Arduino.

2. Describe the reason PWM is used for speed control.

Pulse Width Modulation (PWM) is used to control the speed of a DC motor because it allows efficient adjustment of the average voltage delivered to the motor without wasting power. The technique works by switching the motor's power ON and OFF at a very high frequency. The duty cycle of this signal, which represents the proportion of time the signal stays ON within one cycle, determines the motor's speed. A higher duty cycle means the motor receives more average voltage, causing it to spin faster, while a lower duty cycle supplies less voltage, resulting slower rotation. Unlike analog voltage control,PWM maintains high efficiency because the transistor is either fully ON or fully OFF, minimizing energy loss as heat. This method provides precise and energy-efficient speed control, making it ideal for microcontroller motor applications.

3. Describe the outcome when both IN1 and IN2 are set HIGH.

When both IN1 and IN2 inputs for a motor channel are set to HIGH, and the ENA pin is also active, the motor enters an active braking or dynamic braking state. In this configuration, both terminals of the DC motor are connected to the same voltage level, meaning there is no potential difference across the motor. However, as the motor is spinning, it behaves like a generator and produces a counter-electromotive force back EMF. Since both motor terminals are shorted to the same voltage, the back EMF generates a reverse current that creates a strong opposing torque, bringing the motor to a quick and abrupt stop. This method provides a faster stop compared to simply disabling the motor of setting ENA LOW, which would allow it to coast to a stop gradually.

4. Explain how braking can be implemented using the L298P.

Braking on the L298P Motor Driver can be achieved by setting both input pins (IN1 and IN2) of a motor channel to the same logic level either both HIGH or both LOW. This causes the motor terminals to have equal voltage potential, eliminating any voltage difference and stopping current flow through the motor. The result is an active braking effect, where the motor's kinetic energy is quickly dissipated as heat within the circuit, causing the motor to stop rapidly.

Example Configuration for Motor A:

digitalWrite(IN1, HIGH); digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH); digitalWrite(IN2, LOW);

Or

Both configurations engage the braking mode, ensuring that the motor stops faster than when the ENA pin is simply turned off. This method is particularly useful in applications that require precise motion control, such as robotic arms or automated systems.

5. Modify the code to control two DC motors simultaneously.

```
// ======== Motor A Connections ========
const int ENA = 5;      // Speed control (PWM)
const int IN1 = 4;   // Direction 1
const int IN2 = 4;   // Direction 2

// ======== Motor B Connections ========
const int ENB = 7;      // Speed control (PWM)
const int IN3 = 6;   // Direction 1
const int IN4 = 6;   // Direction 2
```

```
void setup() {
  // Set all motor pins as outputs
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {
```

```
  // --- Move both motors forward ---
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 200);   // Motor A speed (0–255)
  analogWrite(ENB, 200);   // Motor B speed (0–255)
  delay(2000);

  // --- Move both motors backward ---
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, 200);
  analogWrite(ENB, 200);
  delay(2000);

  // --- Apply braking (both motors stop instantly) ---
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  delay(1000);

  // --- Stop (coast stop – motors free spin) ---
  digitalWrite(IN1, LOW);
```

```
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  delay(1000);

  // --- Forward again at different speeds (speed test) ---
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 128);  // Half speed
  analogWrite(ENB, 255);  // Full speed
  delay(2000);

  // Repeat
}
```

9.0 Conclusion

In conclusion, the experiment successfully fulfilled its objectives of controlling both the speed and direction of a DC motor using the L298N motor driver module in conjunction with the Arduino UNO. The practical results clearly demonstrated that the L298N driver can efficiently operate a DC motor in forward and reverse directions by adjusting the logic states of the input pins, while the PWM (Pulse Width Modulation) signal applied to the enable pins provided accurate and smooth speed control. Furthermore, it was observed that setting both input pins to the same logic level (either HIGH or LOW) effectively activated the braking mode, allowing the motor to stop quickly when required.

The overall performance of the system validated the theoretical concepts of PWM control and H-bridge operation. As the PWM duty cycle increased, the motor speed increased proportionally, confirming the linear relationship between duty cycle and average voltage. Similarly, by reversing the input pin configuration, the motor direction could be switched seamlessly without any hardware modification. These results highlighted the importance of understanding logic control and power modulation in DC motor applications.

Moreover, this experiment provided valuable hands-on experience in integrating hardware components and developing control programs for real-world mechatronic systems. It reinforced the significance of combining electronics, programming, and control theory to design functional and efficient motor control systems. Such practical skills are essential in robotics, automation, and embedded systems engineering, where precise motion control is a fundamental requirement.

Overall, the experiment demonstrated the effectiveness, reliability, and versatility of the L298N motor driver when interfaced with the Arduino platform, providing a strong foundation for more advanced motion control and automation projects in future studies.

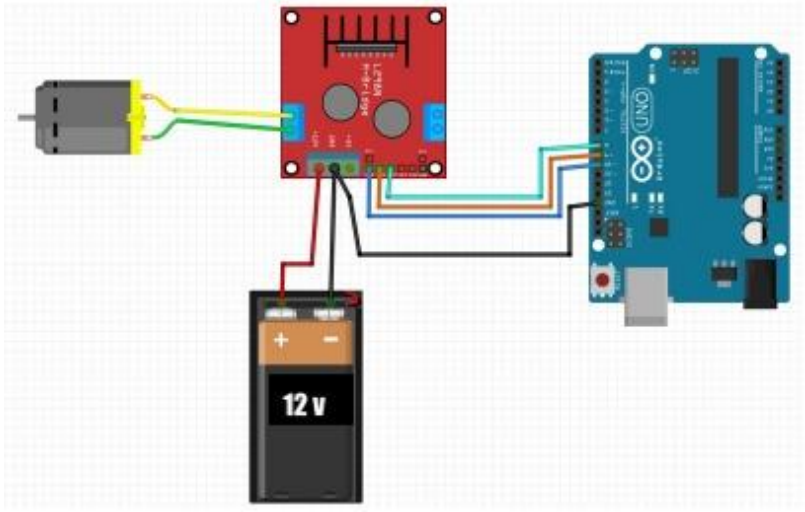
10.0 Recommendation

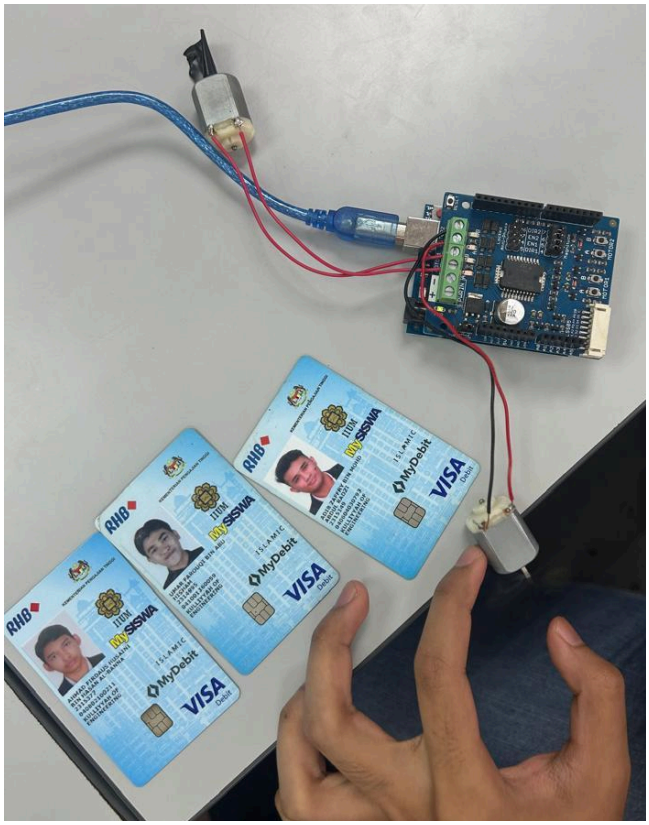
To enhance the performance, reliability, and efficiency of the DC motor control system, several improvements can be implemented. On the software side, using the Serial Monitor to observe real-time PWM values and motor responses can improve speed calibration, enable faster debugging, and allow more precise control. Structuring the program with dedicated functions for speed, direction, and braking would also improve code readability and maintainability. From a hardware perspective, ensuring firm and clean connections between the Arduino, L298P driver, and motor terminals can reduce electrical noise and improve system stability. It is also important to use an appropriate and stable power supply to prevent underpowering or overheating the motor and driver. Incorporating protective components, such as flyback diodes or capacitor filters, can suppress voltage spikes generated by the motor's inductive load, enhancing the durability of the circuit. For future development, integrating a closed-loop feedback system using a rotary encoder could allow precise monitoring of motor RPM and automatic speed adjustments, while adding wireless modules like Bluetooth or Wi-Fi would enable remote control and data logging. Additionally, advanced control strategies, such as PID algorithms or microcontroller timers, could optimize response time, reduce overshoot, and increase overall system efficiency. Implementing these improvements would make the system more precise, versatile, and suitable for advanced robotics, automation, and embedded motor control applications.

11.0 References

1. Arduino, “PWM (Pulse Width Modulation),” [Online]. Available: https://www.arduino.cc/en/Tutorial/PWM
2. Cytron Technologies Sdn Bhd, “0.8Amp 5 V-26 V DC Motor Driver Shield for Arduino (2 Channels),” [Online]. Available: https://my.cytron.io/p-0.8amp-5v-26v-dc-motor-driver-shield-for-arduino-2-channels
3. Arduino, “Arduino UNO Rev3,” [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3

12.0 Appendices





13.0 Acknowledgement

We would like to express our sincere gratitude to Dr. Zulkifli Bin Zainal Abidin and Dr. Wahju Sediono for their continuous support and for providing the essential resources and facilities required to carry out this project. We also extend our heartfelt appreciation to everyone who contributed to the success of this lab report through their valuable insights, assistance, and feedback.

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 specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons. 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 that no further improvement on the reports is needed from any of the individual contributors to the report.

Signature:

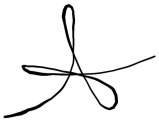
Read [/]

Name: AHMAD FIRDAUS HUSAINI BIN HASAN AL-BANNA
Matric Number: 2315377

Understand [/]
Agree [/]

Signature:

Read [/]



Name: ADIB ZAFFRY BIN MOHD ABDUL RADZI
Matric Number: 2315149

Understand [/]
Agree [/]

Signature:

Read [/]

Name: UMAR FAROUQI BIN ABU HISHAM
Matric Number: 2314995

Understand [/]
Agree [/]