SEM 1 25/26

SECTION 1

COURSE CODE: MCTA 3202

LECTURER NAME:
ZULKIFLI BIN ZAINAL ABIDIN

WEEK 4: CONTROLLING A DC MOTOR USING L298P MOTOR DRIVER SHIELD AND GPIO

PREPARED BY:
GROUP 1

NAME	MATRIC NUMBER
AMIR MUIZZUDDIN BIN NORAIDIL RAZMAN	2129579
MUHAMMAD HAIKAL HANIF BIN ABDUL RAZAK	2213297
AHMAD ILHAM BIN ABDUL AZIZ	2112109

Introduction

This task introduces us to the L298P motor driver shield. The purpose of this task is to gain an in-depth understanding of the usefulness and ways to use the L298P motor driver shield. We were also tasked to control the speed and direction of a DC motor using GPIO pins, and implement PWM (Pulse Width Modulation) for speed. All of this needs to be written and uploaded through the code using Arduino IDE. By the end of this task, we hope to learn how to control the motor speed using codes in Arduino IDE and understand the way to use the L298P motor driver shield.

Material

- Arduino UNO
- L298P Motor Driver Shield
- DC Motor (6V–12V) 1–2
- External Power Supply (9V/12V)
- Jumper Wires
- Breadboard

Experimental Setup and Results

Setup Procedure:

1. Mount the L298P shield on top of the Arduino UNO.
2. Connect Motor A and Motor B to the L298P terminals.
3. Ensure that the GND of the external power supply and Arduino are common.
4. Upload the provided Arduino program to the board using Arduino IDE.
5. Observe the direction and speed of both motors as the code executes.

Pin Configuration :

Function	Arduino Pin	Description
ENA	5	PWM pin for Motor A speed
IN1	4	Direction pin 1 for Motor A
IN2	4	Direction pin 2 for Motor A
ENB	7	PWM pin for Motor B speed
IN3	6	Direction pin 1 for Motor B
IN4	6	Direction pin 2 for Motor B

Arduino Code :

```
// --- Mechatronic System Integration (MCTA3203) ---

// Controlling Two DC Motors using L298P Motor Driver Shield and GPIO


// Motor A pins

const int ENA = 5;    // PWM pin for Motor A speed

const int IN1 = 4;    // Direction pin 1 for Motor A

const int IN2 = 4;    // Direction pin 2 for Motor A


// Motor B pins

const int ENB = 7;    // PWM pin for Motor B speed

const int IN3 = 6;    // Direction pin 1 for Motor B

const int IN4 = 6;    // Direction pin 2 for Motor B


void setup() {

  pinMode(ENA, OUTPUT);

  pinMode(IN1, OUTPUT);

  pinMode(IN2, OUTPUT);
```

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

  Serial.begin(9600);
  Serial.println("Dual Motor Control Initialized.");
}

void loop() {
  moveMotor(ENA, IN1, IN2, 200, true);
  moveMotor(ENB, IN3, IN4, 200, true);
  Serial.println("Both motors moving forward...");
  delay(2000);

  moveMotor(ENA, IN1, IN2, 200, false);
  moveMotor(ENB, IN3, IN4, 200, false);
  Serial.println("Both motors moving backward...");
  delay(2000);

  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  Serial.println("Motors stopped.");
  delay(1000);
```

```
}

void moveMotor(int EN, int IN1, int IN2, int speed, bool forward) {
  if (forward) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
  }
  analogWrite(EN, speed);
}
```

METHODOLOGY

An Arduino UNO fitted with an L298P Motor Driver Shield was used in the experiment to simultaneously operate two DC motors. The Arduino's PWM-capable pins (5 and 7) were linked to the ENA and ENB pins, enabling variable voltage control for Motor A and Motor B, respectively. The IN1–IN4 pins were set up as digital outputs so that HIGH and LOW logic signals could be applied to each motor to identify whether it was rotating forward or backward.

The analogWrite() method in the program produced PWM signals with a duty cycle of roughly 78% (200/255). This resulted in a medium operating speed since each motor was powered for 78% of each PWM cycle. These PWM pulses were supplied to the enable pins by the Arduino while concurrentlyIn order to specify the desired rotation, the Arduino concurrently configured direction pins and transferred these PWM signals to the enable pins.

The code's loop function was designed to:

1. For two seconds, turn both motors forward.
2. For an additional two seconds, reverse the direction of both motors.
3. Set the PWM values of both motors to zero to stop them for one second.

In order to show the current status of operation—whether the motors were moving forward, backward, or stopped—messages were transmitted to the Serial Monitor (via Serial.println()) during execution. The synchronisation and proper operation of the program logic and motor responses were guaranteed by this feedback.

CONCLUSION:

In conclusion, the experiment successfully demonstrated the use of the L298P Motor Driver Shield to control the speed and direction of a DC motor using the Arduino UNO. By utilizing GPIO pins for direction (IN1, IN2) and PWM signals for speed control (ENA/ENB), the system achieved smooth and efficient motor operation.

The results confirmed that:

- Increasing the **PWM value** (duty cycle) increases the **motor speed**, and decreasing it slows the motor down.
- Changing the **logic state of IN1 and IN2** reverses the **motor's rotation direction**.
- Setting both IN1 and IN2 to **HIGH** engages **braking**, immediately stopping the motor.

This experiment deepened the understanding of **H-bridge motor control**, **PWM speed regulation**, and **digital logic direction control**, which are fundamental in mechatronics systems such as mobile robots, conveyors, and automated actuators.

QUESTIONS:

1. What is the function of the ENA and ENB pins?

The ENA and ENB pins are the **enable or speed control pins** for Motor A and Motor B. They receive **PWM signals** from the Arduino to regulate the average voltage and therefore control the motor's **speed**.

2. Why is PWM used for speed control?
PWM provides an efficient and precise way to vary motor speed by adjusting the **duty cycle** of a digital signal instead of changing the supply voltage. A higher duty cycle delivers more average power, increasing the motor speed.

3. What happens when both IN1 and IN2 are set HIGH?
When both pins are HIGH, both motor terminals are connected to the same voltage

level, resulting in **active braking**. The motor stops quickly because the back EMF is shorted and energy is dissipated as heat.

4. How can braking be implemented using the L298P?
Braking is implemented by setting both input pins (IN1 and IN2, or IN3 and IN4) to the **same logic level** (either HIGH or LOW). This configuration stops the motor by creating a short circuit across its terminals.

5. How can two DC motors be controlled simultaneously?

By defining another set of control pins (IN3, IN4, ENB) for Motor B and applying similar control logic as Motor A. Using separate PWM outputs for ENA and ENB allows **independent speed and direction control** for both motors.