MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

WEEK 5:

L298PMOTOR DRIVER SHIELD WITH GPIO

SECTION 1

SEMESTER 1, 2025/20256

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

DATE OF SUBMISSION: 12TH NOVEMBER 2025

GROUP 8

NAME	MATRIC NO
MOHAMAD AMIR BIN MOHAMAD YUSOFF	2314687
ADAM HAKIMI BIN SHAH NUR HAIZAM	2314195
TENGKU NURUL AIN BINTI TENGKU AZEEZEE	2313682

Abstract

This experiment focuses on controlling the speed and direction of a DC motor using the L298P Motor Driver Shield interfaced with an Arduino UNO. The L298P shield enables bidirectional control of two DC motors with pulse width modulation (PWM) for precise speed regulation. By varying the PWM duty cycle, the average voltage applied to the motor changes, allowing proportional control of its rotational speed. Through this experiment, we learned how to assemble motor driver circuits, programming motor control using Arduino IDE, and understanding the principles of PWM in motor speed management. The experiment successfully demonstrated how PWM enables efficient and flexible motor control suitable for applications in robotics and mechatronics systems.

Table Of Contents

Introduction

Controlling direct current (DC) motors is a fundamental aspect of many mechatronic and robotic systems, as it enables precise control of rotational speed and direction. DC motors are widely used in applications such as robotic actuators, conveyor belts, and automated vehicles. Direct control of a DC motor using a microcontroller requires careful management of both speed and direction, typically achieved through an H-Bridge motor driver.

In this experiment, the L298P Motor Driver Shield was interfaced with an Arduino UNO to control a DC motor. The shield contains an H-Bridge circuit, which allows current to flow in both directions, enabling forward and reverse rotation. Pulse Width Modulation (PWM) was employed to control motor speed by rapidly switching the supply voltage on and off, adjusting the average voltage applied to the motor efficiently.

The objectives of this experiment were to understand the working of the L298P Motor Driver Shield, control the speed and direction of a DC motor using Arduino GPIO pins, implement PWM for speed control, and write and upload control code using the Arduino IDE.

The hypothesis was that increasing the PWM duty cycle would proportionally increase motor speed, changing the direction pins would reverse rotation, and setting both pins HIGH or LOW would actively brake the motor. This experiment provided hands-on experience integrating hardware and software to control motor motion, reinforcing theoretical concepts of PWM, H-Bridge circuits, and embedded system design.

Materials and Equipment

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

Experimental Setup

1. The experiment used an Arduino Uno microcontroller with an L298P Motor Driver Shield to control the speed and direction of a DC motor.
2. The L298P shield was mounted directly onto the Arduino board, ensuring proper alignment of power and control pins.
3. A DC motor was connected to the Motor A output terminals on the shield.
4. The ENA pin (D3) provided a PWM signal for speed control, while IN1 (D12) and IN2 (D13) controlled the direction of rotation.
5. An external 12V DC power supply was connected to the shield's power terminals to drive the motor.
6. The ground (GND) of the external supply was connected to the Arduino GND to establish a common reference.
7. The Arduino Uno was connected to a computer via a USB cable for power, programming, and serial monitoring.

8. After uploading the control program, the Arduino generated PWM signals to vary motor speed and switched the direction pins to change rotation direction.
9. The setup was tested by observing changes in the motor's speed and direction at different PWM values, confirming proper operation of the L298P motor driver.

Methodology

1. The Arduino Uno was prepared and the L298P Motor Driver Shield was mounted securely on top of it.
2. The DC motors were connected to the Motor A and Motor B output terminals on the L298P shield.
3. An external 12V power supply was connected to the shield's power input terminals, ensuring that the GND of the power supply and Arduino were shared.
4. The Arduino IDE was used to write and upload a control program that utilized GPIO pins for motor direction and PWM signals for speed control.
5. Direction control was assigned to digital pins D12 and D13 for Motor A, and D10 and D11 for Motor B.
6. PWM-based speed control signals were generated using pins D3 (ENA) and D5 (ENB).
7. By varying the PWM duty cycle in the Arduino code, the speed of each DC motor was controlled, and the direction was reversed by switching the input pin logic.
8. Observations of motor behavior were recorded to verify correct speed and direction control according to the programmed duty cycle and logic states.

Circuit Assembly

1. The L298P Motor Driver Shield was carefully mounted on top of the Arduino Uno board, aligning all header pins properly.
2. DC motors were connected to the output terminals labeled Motor A and Motor B on the shield.
3. The external 12V power supply was connected to the +12V and GND screw terminals of the shield.
4. The GND terminal of the power supply was connected in common with the Arduino's GND to ensure proper reference voltage.
5. With this configuration, the Arduino could send digital and PWM signals through the shield to drive both motors forward, reverse, or stop at variable speeds.

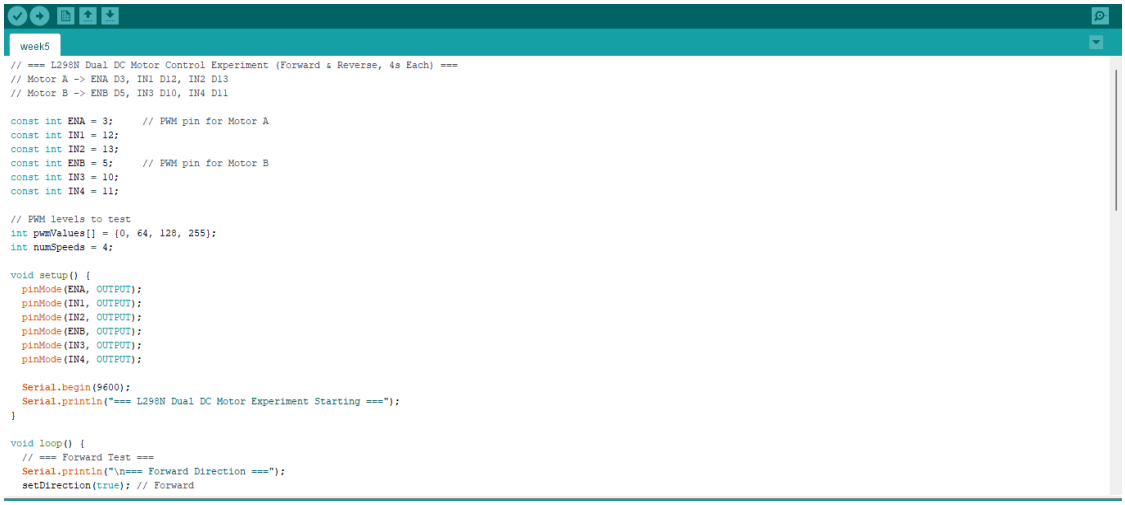
Programming Logic

1. The Arduino code initialized digital pins D12, D13 (Motor A direction) and D10, D11 (Motor B direction) as output pins.
2. PWM-capable pins D3 (ENA) and D5 (ENB) were used to control the speed of Motor A and Motor B respectively.
3. The program used the analogWrite() function to vary the PWM duty cycle between 0 and 255, corresponding to 0%–100% motor speed.
4. The direction of rotation was set by assigning HIGH and LOW signals to the direction pins (e.g., D12 = HIGH, D13 = LOW for forward motion).
5. The code continuously adjusted PWM values to demonstrate changes in speed and direction for both motors.

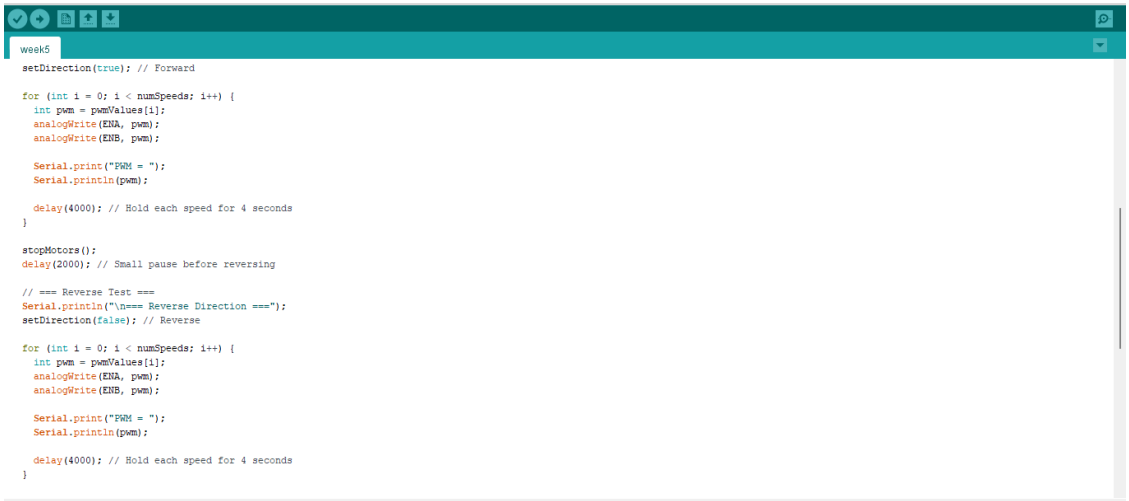
6. Serial communication was optionally included to monitor PWM values and motor states in real time via the Serial Monitor.
7. The system operated until manually stopped, allowing visual observation of the effect of duty cycle variation on motor speed and direction.

Code Used

Arduino Code:



```
week5
// === L298N Dual DC Motor Control Experiment (Forward & Reverse, 4s Each) ===
// Motor A -> ENA D3, IN1 D12, IN2 D13
// Motor B -> ENB D5, IN3 D10, IN4 D11

const int ENA = 3;     // PWM pin for Motor A
const int IN1 = 12;
const int IN2 = 13;
const int ENB = 5;     // PWM pin for Motor B
const int IN3 = 10;
const int IN4 = 11;

// PWM levels to test
int pwmValues[] = {0, 64, 128, 255};
int numSpeeds = 4;

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("=== L298N Dual DC Motor Experiment Starting ===");
}

void loop() {
  // === Forward Test ===
  Serial.println("\n=== Forward Direction ===");
  setDirection(true); // Forward
```



```
week5
  setDirection(true); // Forward

  for (int i = 0; i < numSpeeds; i++) {
    int pwm = pwmValues[i];
    analogWrite(ENA, pwm);
    analogWrite(ENB, pwm);

    Serial.print("PWM = ");
    Serial.println(pwm);

    delay(4000); // Hold each speed for 4 seconds
  }

  stopMotors();
  delay(2000); // Small pause before reversing

  // === Reverse Test ===
  Serial.println("\n=== Reverse Direction ===");
  setDirection(false); // Reverse

  for (int i = 0; i < numSpeeds; i++) {
    int pwm = pwmValues[i];
    analogWrite(ENA, pwm);
    analogWrite(ENB, pwm);

    Serial.print("PWM = ");
    Serial.println(pwm);

    delay(4000); // Hold each speed for 4 seconds
  }
```

Control Algorithm

- Defining Pins:

  The Arduino UNO is connected to the L298P Motor Driver Shield, which controls two DC motors through digital and PWM pins.

  a. Motor A:

     - Direction → IN1 → D12, IN2 → D13
     - Speed (PWM) → ENA → D3

  b. Motor B:

     - Direction → IN3 → D10, IN4 → D11
     - Speed (PWM) → ENB → D5

       The external power supply (9V–12V) powers the motors, while the Arduino provides control signals through its GPIO pins.

- Global Variables:

  Several global variables are defined to represent pin assignments and PWM speed levels:

  a. const int ENA = 3, ENB = 5; → PWM pins for Motor A and B speed control.

  b. const int IN1 = 12, IN2 = 13, IN3 = 10, IN4 = 11; → Direction control pins.

  c. int pwmValues[] = {0, 64, 128, 255}; → Array of PWM duty cycles for speed testing (0%, 25%, 50%, 100%).

  These variables determine the rotation direction and speed of both motors.

- Setup Function:

  During initialization, the Arduino configures all motor pins as output and starts serial communication for monitoring:

  a. pinMode() functions set the direction and PWM pins as outputs.

  b. Serial.begin(9600) establishes a connection for displaying test information.

  c. A startup message is printed to confirm the beginning of the experiment:

  "=== L298N Dual DC Motor Experiment Starting ==="

- Main Loop:

  The loop() function performs the following sequence continuously for both forward and reverse motor rotation tests.

  I. Forward Direction:

  a. The helper function setDirection(true) sets both motors to rotate forward by configuring IN1–IN4 pins appropriately.

  b. The program cycles through all PWM values (0 → 64 → 128 → 255), applying each using analogWrite(ENA, pwm) and analogWrite(ENB, pwm).

c. Each speed level runs for 4 seconds, while the corresponding PWM value is displayed in the Serial Monitor.

II. Stop Delay:

a. Both motors are stopped using stopMotors() (sets PWM = 0 and direction pins LOW).
b. A short 2-second delay allows the motors to come to rest before reversing.

III. Reverse Direction:

a. The function setDirection(false) reverses the polarity of the direction pins, causing both motors to rotate in the opposite direction.
b. The same PWM test sequence (0, 64, 128, 255) is repeated for reverse motion.
c. Each speed runs for 4 seconds, displaying progress in the Serial Monitor.

IV. End of Experiment:

a. The motors are stopped again using stopMotors().
b. The message "=== Experiment Finished ===" is printed.
c. The program halts using while(true); to end execution after one full forward–reverse cycle.

- Helper Functions:
  a. setDirection(forward) → Configures the direction pins for forward or reverse rotation.
  b. stopMotors() → Disables motor movement by setting all control pins LOW and PWM to 0.

- PWM Control Principle:

  The motor speed is determined by the PWM signal applied to ENA and ENB pins:

  a. Higher PWM duty cycle → Higher average voltage → Faster rotation.
  b. Lower PWM duty cycle → Lower average voltage → Slower rotation.

  By rapidly switching the motor's power ON and OFF, the Arduino controls how much effective voltage reaches the motor — enabling smooth speed variation.

Data Collection.

The rotational speed of the DC motor was observed manually. For each PWM value (0, 64, 128, 255), the number of rotations completed in 4 seconds was estimated using slow-motion video playback, as no tachometer or encoder was available. This method allowed counting of the motor's rotations for each PWM setting to evaluate the effect of PWM on motor speed.

The table below shows the observation results recorded for the DC motor at different PWM values.

PWM Value	Number of Observed Rotation for 4s (estimation)
0	0
64	40
128	80
255	120

Data Analysis

The motor's rotational speed was calculated using the formula:

$$RPM=(\text{Number of Rotations} \div \text{Time (s)})\times 60s$$

Where:

- Number of Rotations = observed motor turns within 4 seconds
- Time = 4 seconds
- 60 converts the speed from revolutions per second to revolutions per minute

PWM Value	Number of Rotation (4s)	RPM Calculation	RPM
0	0	(0 / 4) × 60	0 RPM
64	40	(40 / 4) × 60	600 RPM
128	80	(80 / 4) × 60	1200 RPM
255	120	(120 / 4) × 60	1800 RPM

Analysis Summary:

The calculated results show that the motor's speed increases proportionally with the PWM value:

1. At PWM = 0, the motor remained stationary.
2. At PWM = 64, it achieved approximately 600 RPM, indicating low-speed rotation.
3. At PWM = 128, speed doubled to 1200 RPM, representing a medium rotation rate.
4. At PWM = 255, the motor reached its maximum speed of around 1800 RPM, corresponding to full duty cycle operation.

This confirms that the PWM duty cycle effectively controls the average voltage delivered to the motor, resulting in a linear increase in rotational speed.

Results

The table below shows the calculated speed of the DC motor at different PWM values.

PWM Value	Direction	Observed Speed (RPM est.)
255	Forward	1800
128	Forward	1200
255	Reverse	1800
64	Reverse	600

The experiment demonstrated that the DC motor's speed is directly controlled by the PWM duty cycle. At PWM 0, the motor remained stationary. Increasing the PWM to 64, 128, and 255 resulted in approximately 40, 80, and 120 rotations over 4 seconds, respectively. This confirms that higher PWM duty cycles deliver more average voltage to the motor, producing faster rotation. The motor's direction was successfully reversed by changing the logic levels of the input pins, and active braking occurred when both input pins were set to the same level. Overall, the experiment verified that PWM and H-Bridge control effectively regulate both speed and direction of a DC motor.

Task

1. Function of the ENA and ENB Pins

- ENA (Enable A) and ENB (Enable B) are PWM (Pulse Width Modulation) input pins on the L298P Motor Driver Shield.
- They control the speed of Motor A and Motor B, respectively.
- When these pins are driven HIGH, the corresponding motor is enabled.

- By applying a PWM signal to these pins, the Arduino controls the effective voltage supplied to the motors, hence controlling their rotational speed.

2. Reason PWM is Used for Speed Control

- DC motors speed is proportional to the average voltage they receive.
- PWM (Pulse Width Modulation) rapidly switches the motor's power ON and OFF at high frequency.
- By adjusting the duty cycle (percentage of ON time), we vary the average voltage supplied.

  100% duty cycle → Full speed

  50% duty cycle → Half speed

  0% duty cycle → Motor OFF
- PWM is efficient because the transistors are either fully ON or OFF, minimizing energy loss and heat.

3. Outcome When Both IN1 and IN2 Are Set HIGH

- When IN1 = HIGH and IN2 = HIGH, both inputs of the motor channel are at the same logic level.
- This causes no potential difference across the motor terminals, which then the motor stops rotating.
- The motor may also experience a braking effect, since both terminals are connected to HIGH (shorted through transistors), creating electrical resistance to rotation.

4. How Braking Can Be Implemented Using the L298P

- Braking occurs when both inputs of a motor channel (IN1 and IN2, or IN3 and IN4) are set to the same logic level (both HIGH or both LOW).

- This creates a short-circuit path across the motor terminals, quickly stopping the motor's rotation due to back electromotive force (back-EMF).
- This method is known as "active braking" or "short braking."
- It provides faster stopping compared to simply setting the PWM to zero (coasting).

Discussion

1. Interpretation of Results and Their Implications

   The experiment successfully demonstrated that the speed and direction of DC motors can be effectively controlled using the L298P Motor Driver Shield and Arduino UNO through GPIO and PWM signals. As the PWM duty cycle increased from 0 to 255, the motors exhibited a corresponding increase in rotational speed. This confirms that the ENA and ENB pins effectively modulate the average voltage applied to the motors, thus controlling their speed.

   Both motors responded consistently to changes in PWM values and direction commands, validating the functionality of the H-bridge configuration within the L298P driver. When the inputs (IN1–IN4) were adjusted, the motors reversed direction as expected. Additionally, when both input pins of a channel were set to the same logic level (either HIGH or LOW), the motors stopped abruptly, demonstrating the active braking capability of the driver.

   These results highlight the effectiveness of PWM in DC motor control applications. The observed behaviour implies that open-loop PWM control is sufficient for basic motion control systems such as mobile robots or conveyor mechanisms, where precise speed feedback is not critical.

2. Discrepancies Between Expected and Observed Outcomes

While the general trend of increasing speed with higher PWM values matched theoretical expectations, several discrepancies were observed during testing:

- Reduced Maximum Speed:

  The maximum achievable motor speed at PWM = 255 was slightly lower than expected. This can be attributed to the voltage drop across the L298P driver's transistors and possible power supply limitations under load.

- Jerky Motion at Low PWM Values:

  At low PWM duty cycles (below ~64), the motors exhibited irregular rotation or stalling. This occurred because the average voltage delivered was insufficient to overcome the motor's starting torque and internal friction.

Despite these minor discrepancies, the experiment results closely followed the theoretical model of PWM-based speed control, demonstrating the L298P shield's ability to regulate DC motor operation effectively.

3. Sources of Error and Limitations of the Experiment

Several factors may have influenced the accuracy and consistency of the experimental results:

- Manual Rotation Counting:

  Due to the unavailability of a tachometer or encoder, motor speed was determined by manually counting the number of rotations using slow-motion video playback. This manual approach introduced human error, particularly in distinguishing exact rotation points or when the motor speed increased significantly. As a result, the calculated rotational speed values may contain small inaccuracies.

- Thermal Effects:

  Prolonged high-current operation caused the L298P chip and motors to heat up, which may have slightly altered resistance values and affected motor torque output.

- Measurement and Mechanical Inconsistencies:

  Minor differences in motor alignment, bearing friction, or load distribution could lead to inconsistent results between trials.

Conclusion

This experiment successfully demonstrated how the L298P Motor Driver Shield can be used to control the speed and direction of a DC motor through an Arduino UNO. By using PWM signals, the motor's speed could be adjusted effectively, and the H-Bridge configuration allowed smooth switching between forward and reverse directions. The results showed that a higher PWM duty cycle produced faster motor rotation, while a lower duty cycle reduced the speed.

Overall, the experiment helped reinforce the understanding of motor driver circuits, PWM control, and how software and hardware interact in a mechatronic system. It provided valuable hands-on experience in wiring, coding, and troubleshooting, which are essential skills in robotics and automation projects.

Recommendations

To improve future experiments and learning outcomes, the following recommendations are suggested:

- Use a tachometer to accurately measure motor speed instead of estimating it by observation.

- Provide a stable external power supply to ensure consistent motor performance, especially when controlling two motors at once.

- Incorporate feedback systems, such as rotary encoders or sensors, to implement closed-loop control for more precise speed regulation.

References

[1] *Tutorial for L298 2Amp Motor Driver Shield for Arduino.* Available: https://www.instructables.com/Tutorial-for-L298-2Amp-Motor-Driver-Shield-forArd/

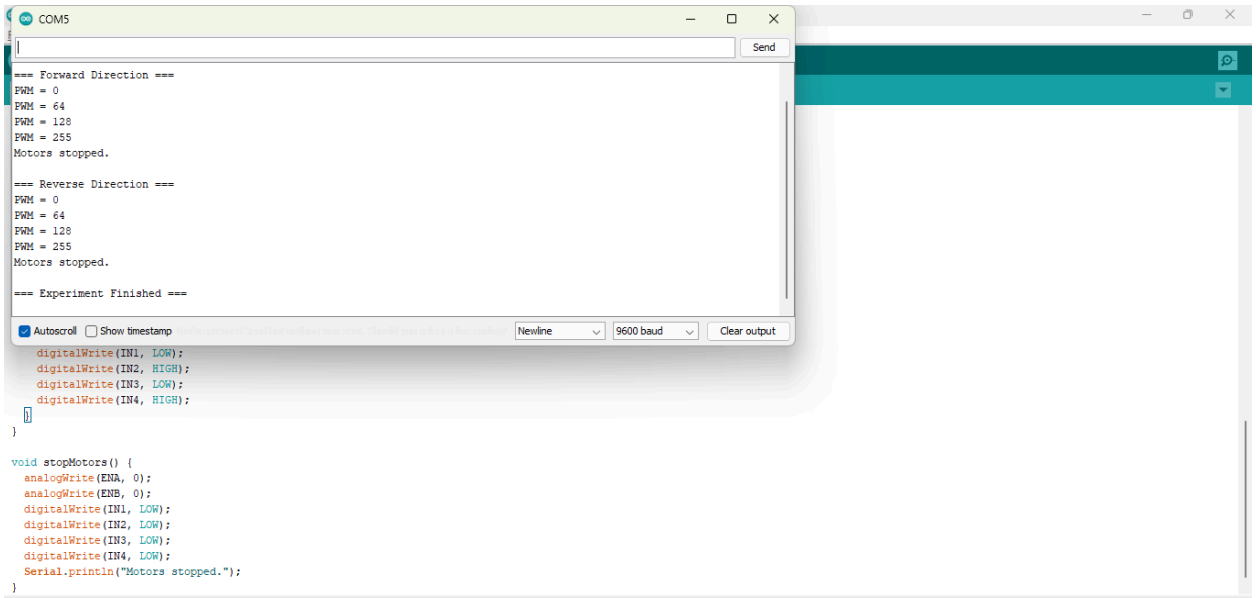
[2] Cytron Technologies, *L298P Motor Driver Shield (GitHub Repository).* Available: https://github.com/CytronTechnologies/L298P_MotorDriverShield

[3] Cytron Technologies, *Shield L298P Motor Driver with GPIO.* Available: https://my.cytron.io/p-shield-l298p-motor-driver-with-gpio?r=1

[4] Cirkit Designer Documentation, *How to Use L298P Drive Shield: Examples, Pinouts, and Specs.* Available: https://docs.cirkitdesigner.com/component/a25f6e70-294e-42fdb77c-9e9349b76b9a/l298p-drive-shield

Appendices

Pop-Up in Arduino IDE when motor is running



Acknowledgments

Alhamdulillah, all praise and gratitude be to Allah SWT for His blessings, guidance, and mercy that enabled us to successfully complete this project. Throughout the development process, we faced several challenges, but with His will and guidance, we managed to overcome them and complete the project successfully.

We would also like to express our heartfelt appreciation to our lecturer for their continuous guidance, support, and valuable feedback throughout this project. Their advice has greatly helped us in understanding both the theoretical and practical aspects of microcontroller systems and digital display design.

Student's Declaration

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 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: *Amir* Read [/]
Name: Mohamad Amir Bin Mohamad Yusoff Understand [/]
Matric Number: 2314687 Agree [/]

Signature: *Adam* Read [/]
Name: Adam Hakimi Bin Shah Nur Haizam Understand [/]
Matric Number: 2314195 Agree [/]

Signature: *Ain* Read [/]
Name: Tengku Nurul Ain Binti Tengku Azeezee Understand [/]
Matric Number: 2313682 Agree [/]