TITLE AND DATE

Document name: PWM Motor Driver
Document reference: *Davenport, Robert*.cmpe311.fall25.*Lab-PWM-Motor-Driver*
Date of publication: 12*/1/2025*

LEAD ENGINEER: *Robert Davenport, UMBC CSEE, Catonsville, Maryland, USA*

STAKEHOLDERS:
Prof Kidd, Instructor, University of Maryland Baltimore County, Baltimore, Maryland, USA
MD Safwan Zaman, TA, University of Maryland Baltimore County, Baltimore, Maryland, USA

HIGH-LEVEL DESCRIPTION: This document outlines requirements for project #4. These include customer, technical, and testing requirements

DESCRIPTION: *Design and implement an embedded system using an Arduino UNO compatible development board. System must drive a motor using Pulse-Width Modulation. System must also be able to adjust the duty cycle of the motor based on button presses by the user. System must finally have an asynchronous button task manager that handles button debounces and communicates with motor driving circuit.*

RESULT SUMMARY: The project was successful, and the design proposed meets all requirements outlined

REFERENCES AND GLOSSARY

REFERENCES:

- projectAddDutyCycle.pdf – *Original project outline*
- Example_Project_Report.docx – *Further clarification on report outline*
- https://docs.arduino.cc/language-reference/en/functions/time/millis - How to use millis() built in function
- Arduino UNO R3 Product Reference Manual SKU A000066 12/03/2024
- https://www.alldatasheet.com/datasheet-pdf/pdf/22400/STMICROELECTRONICS/IRF740.html - IRF740 Mosfet Datasheet
- driveMotorFromDigitalPin.pdf – Sample circuit for simple PWM motor driver and calculations

DEFINITIONS AND ABBREVIATIONS:
Arduino – an Italian open-source hardware and software company; also refers to a development board created by the company
arduino.h – header for a library of convenience functions specific to the Arduino development platform
AVR – A family of microcontrollers, originally developed by Atmel, and currently owned by

Microchip Technology
ELEGOO – A Chinese company that develops and markets 3D printers and accessories
IDE – Integrated Development Environment
gcc – front end for the GNU Compiler Collection
Github – A widely used distributed SVC (Software Version Control) system
LED – Light Emitting Diode
Msec – Millisecond time interval used by IDE

REQUIREMENTS

CONVENTIONS:

Must, shall or will – your design must satisfy the requirement
May – your design may satisfy the requirement but doesn't have to
Informative – the intent of the following description is to make the requirement more understandable
All customer requirements are started with "C.#".
All high-level requirements are started with "HL.#".
All testing/validation requirements ar5e started with "T.#"

CUSTOMER REQUIREMENTS:

C1. The User must be able to adjust the duty cycle of the motor with a button push
C2. The User must be able to update the duty cycle with subsequent button pushes
C3. The System must run upon an Arduino Uno R3 compatible development board

HIGH-LEVEL TECHNICAL REQUIREMENTS:

HL.1 The System must a limiting resistor and a pull-down resistor
HL.2 The System must include the IRF740 Mosfet, flyback diode as well as a provided motor
HL.3 The System must use a standard Arduino compatible development board (e.g. the provided ELEGOO Uno R3)
HL.4 The System must interpret user button inputs asynchronously, and send updates using the PWM pin on the Arduino (or equivalent board)
HL.5 The User must be able to adjust the duty cycle of the motor with subsequent button presses
HL.6 The set duty cycle must stay constant until the user input is detected

TESTING AND VALIDATION REQUIREMENTS:

T.1 Motor RPM is at 0 initially
T.2 Motor runs 2/8 duty cycle with 1 button push
    T.2.1 Subsequent button pushes increase duty cycle 2/8, up to its maximum duty cycle
T.3 After Motor reaches maximum duty cycle, subsequent button pushes will decrease duty cycle, back down to 0
T.4 System must be able to increase and decrease duty cycle automatically as user continues to push button

DESIGN AND TESTING

DESIGN

Materials:

- Arduino Uno R3 or similar development board
- Arduino IDE 2.3.3 or newer
- Cables & Breadboard
- 1 flyback diode
- 1 IRF740 Mosfet
- An DC electric motor
- 2 resistors, 1 limiting and one pull down
- 9V battery

Schematic: Located under Appendix A.

Design code: Located under Appendix B.

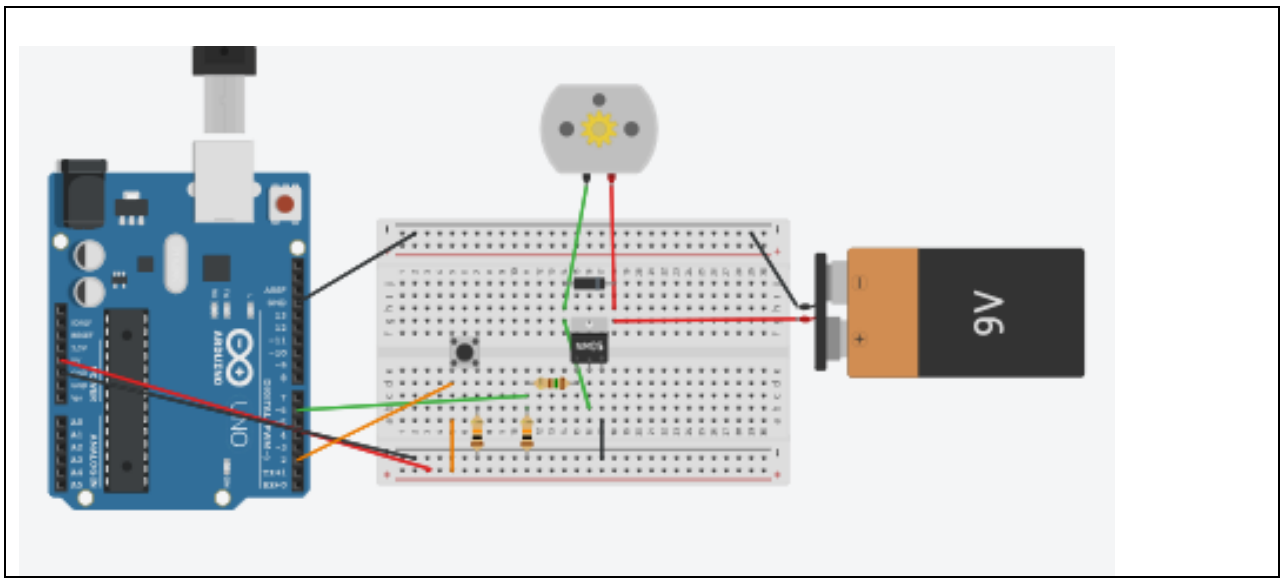
Solutions: Located under Appendix C.

TESTING RESULTS:

Test # (See Testing and Validation requirements)	Result (See video of test)
T.1	Passed
T.2	Passed
T.2.1	Passed
T.3	Passed
T.4	Passed

Appendix A. Schematic

Testbed Schematic. PWM pin at 6, Button input at pin 2.



Appendix B. Design Code

```
// Pin definitions
const int button = 2;  // Button input pin
const int pwm = 6;     // PWM output pin for motor control

// Button state tracking variables
int buttonState = LOW;      // Current stable button state (after
debounce)
int lastButtonState = LOW;  // Previous button reading from last loop
iteration

// Motor speed state variables
int state_count = 0;        // Current speed state (0, 2, 4, 6, 8)
const int maxDuty = 255;    // Maximum PWM duty cycle value
bool decend = false;        // Direction flag: false = ascending, true =
descending

// Debounce timing variables
unsigned long lastDebounceTime = 0;  // Timestamp of last button state
change
unsigned long debounceDelay = 50;    // Debounce delay in milliseconds

void setup() {
  // Initialize pins
  pinMode(pwm, OUTPUT);      // Set PWM pin as output
  pinMode(button, INPUT);    // Set button pin as input
  analogWrite(pwm, 0);       // Start with motor off
}

void loop() {
  // Read current button state
  int buttonRead = digitalRead(button);
```

```
  // Check if button reading has changed
  if (buttonRead != lastButtonState) {
    lastDebounceTime = millis();  // Reset debounce timer
  }

  // Check if debounce is complete)
  if ((millis() - lastDebounceTime) > debounceDelay) {
    // Check if the stable reading is different from current button
state
    if (buttonRead != buttonState) {
      buttonState = buttonRead;  // Update button state to new stable
value

      // Only update motor speed on button PRESS (transition from LOW to
HIGH)
      if (buttonState == HIGH) {
        int duty;  // Variable to hold calculated PWM duty cycle

        // Ascending phase: increase speed from 0 to 8
        if (decend == false) {
          state_count = state_count + 2;  // Increment speed state by 2
          if (state_count >= 8) {         // Check if maximum reached
            state_count = 8;              // Cap at maximum
            decend = true;                // Switch to descending mode
          }
        }
        // Descending phase: decrease speed from 8 to 0
        else {
          state_count = state_count - 2;  // Decrement speed state by 2
          if (state_count <= 0) {         // Check if minimum reached
            state_count = 0;              // Cap at minimum
            decend = false;               // Switch back to ascending
mode
          }
        }

        // Calculate PWM duty cycle based on state_count
        // Map state_count to duty cycle (0-255)
        duty = (maxDuty * state_count) / 8;

        // Write duty cycle to the motor
        analogWrite(pwm, duty);
      }
    }
  }

  // Save current reading for next loop iteration
  lastButtonState = buttonRead;

  // Small delay to slow down loop and make serial output readable
  delay(100);
}
```

Appendix C. Solutions

PROBLEM#1: Calculate the value of the resistor R1 to limit the current through the gate to the maximum acceptable for an Arduino Uno digital pin. (Note that R1 only comes into play if there the MOSFET has failed resulting in a short.)

$$R_1 = \frac{V}{I} = \frac{5}{0.04} = 125\Omega$$

PROBLEM#2: What constraints must you be aware of in determining an acceptable value for R2?

Some constraints we must be aware of when choosing an acceptable value for R2 are that it is large enough to not waste too much current, but small enough that the MOSFET is able to quickly discharge its gate capacitance. Additionally, it must not reduce the voltage for R1, meaning that it must still allow for enough gate voltage across R1.

PROBLEM#3: What is the minimum response time theoretically possible by the ATmega328p running at 16MHz?

$$T = \frac{1}{f} = \frac{1}{16MHz} = 62.6ns$$

END OF DOCUMENT

Appendix C. Solved Problems

END OF DOCUMENT