Project 3: 12/2/2025

Document name: PWM Motor Driver Project Report
Document reference: andrewOmer.cmpe311.fall25.project#3
Date of publication: December 2, 2025

LEAD ENGINEER: Andrew Omer, University of Maryland Baltimore County, Baltimore, 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 is the project document for Project#3 of the CMPE311 class. It contains customer, technical and testing requirements as well as the design and results of the validation testing.

DESCRIPTION: This design is to create a circuit on an Arduino Uno R3 development platform that allows the user of the program to change the speed of a motor through a button that changes the PWM and duty cycle. This will be done through multiple parts with their own validation of the steps of the design. Part 1 will test the PWM and duty cycle changes with the button and an LED. Part 2 will test the motor and MOSFET breadboard setup. Part 3 will integrate both parts to control the PWM duty cycle for the motor with the button.

RESULT SUMMARY: The project was a success, the embedded system design meeting all testing and high-level requirements.

REFERENCES AND GLOSSARY

REFERENCES:

- CE Task Manager Project Report – Prior CMPE 311 project which the asynchronous software is based off of
- PWM Motor Driver Project Report.pdf– The definition of the project this document addresses
- Arduino UNO R3 Product Reference Manual SKU A000066, 12/03/2024

***DEFINITIONS*:**

"The User" – The person operating (not programming) the embedded system
"The System" – The embedded system being operated by The User
"The Customer" – The person(s) paying for the embedded system being designed and built
"The Developer" – The person(s) designing and building the System
"The Evaluator" – The person(s) that determine whether or not The System satisfies The Customer-requirements.
"The Customer-requirements" – The requirements defined by The Customer as satisfying The Contract.
"The Requirements" – The System's high-level technical requirements derived from The Customer-requirements.
"The Educational-constraints" – Requirements imposed by the instructor unrelated to the embedded system that allow The System to be evaluated.
"The Company" – The organization The Customer has contracted with to build The System.
"The Contract" – The business document that legally binds The Company to provide some service or product to The Customer.
"serial-monitor" – The serial port used by the Arduino IDE to communicate with The User.
"The Reference-platform" – The configuration of The System used by The Developer to test and validate The System. For this class, The System is the Arduino compatible ELEGOO Uno R3 development board.
"PROJECT-ASYNC" – The previous project upon which this project is based

ACRONYMS 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
Github – A widely used distributed SVC (Software Version Control) system
LED – Light Emitting Diode
CE – Cyclic Executive
PWM – Pulse Width modulation
Duty cycle – ratio of pulse width to period of rectangular waveform

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:

C.1 User must be able to change motor speed with button
C.2 Product must contain embedded system and MOSFET circuit
C.3 Duty cycle changes must follow figure A
C.4 Embedded system must contain Arduino uno R3 compatible board
C.5 Embedded system must use materials from Table A

HIGH-LEVEL TECHNICAL REQUIREMENTS:

HL.1 Arduino must run tasks asynchronously
HL.2 User must be able to control PWM with the button
HL.3 Embedded system must use LED for circuit in part 1
HL.4 Embedded system must use Arduino IDE compatible board

EDUCATIONAL REQUIREMENTS:

E.1 Circuit must contain provided materials (MOSFET, diode, motor, etc)
E.2 PWM must be changed with an ISR
E.3 Project report must follow professional standards

DESIGN:

DESIGN PRE-REQUISITES:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better
3. Materials used in each part

Part 1:

Using the asynchronous scheduler from project 2, set up timer to trigger an ISR which triggers the update for the duty cycle. The main loop contains the button debouncing logic to prevent multiple inputs. PWM is on pin 9 which is used for the Timer 1 ISR. There 5 different duty cycles used from 0/800 to 800/800. Pin 2 is used for the button input in pullup mode so that it is inverted.

Part 2:

See Figure B below for the Simple motor driver used in this design. Note that the MOSFET used was an IRF Z34N, the 12V source was created with 8 1.5V AA batteries, and the 150 ohm resistor was swapped out for a 330 ohm resistor. This was assembled on a breadboard

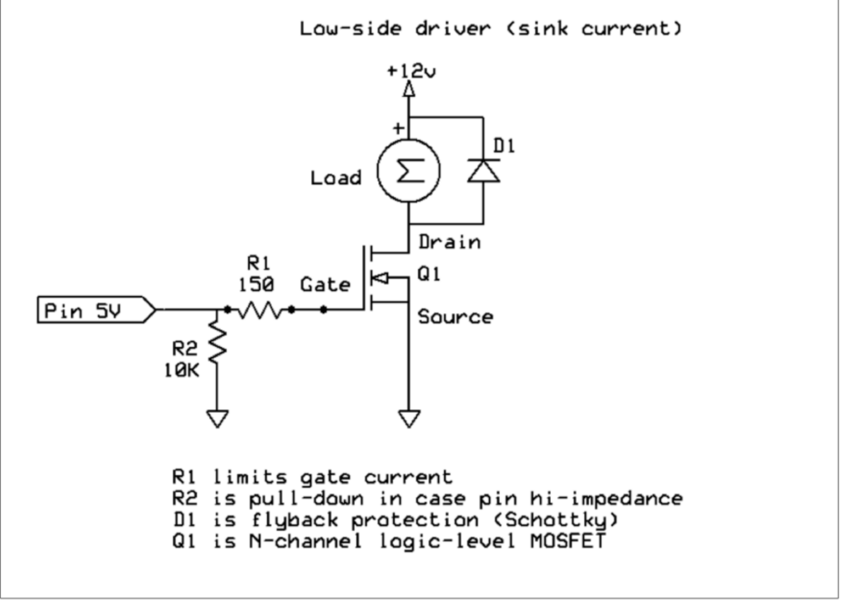
Low-side driver (sink current)
+12v
+
D1
Load
Drain
R1
150
Gate
Q1
Pin 5V
Source
R2
10K
R1 limits gate current
R2 is pull-down in case pin hi-impedance
D1 is flyback protection (Schottky)
Q1 is N-channel logic-level MOSFET

Figure B. Simple motor driver using a N-Channel MOSFET.
Note: modify the source voltage to match the voltage of the motor, 6Vdc.

Part 3:

Combined both part one and two on the same breadboard by substituting the 5V pin with pin 9 (PWM pin). By hitting the button, the software will update the duty cycle, rotating through it's values.

TESTING AND VALIDATION

Part 1:

T.1 Each button press changes the LED brightness by changing the duty cycle
T.2 The Serial interface is updated with relevant information per button press
T.3 Multiple button inputs are protected by debouncing software

The video shows the LED getting brighter with each button press until it resets to 0. On each press the serial monitor is updated to display the current duty cycle. The software

Video Link: https://drive.google.com/file/d/1ohbm10I6w-dmbe5tOt_nYQ2FmNK2wFS1/view?usp=sharing

Part 2:

T.1 Breadboard uses MOSFET Circuit from Figure B
T.2 Motor runs

The requirements are simple, when connected to power and ground the motor turns on.

Video Link:
https://drive.google.com/file/d/1g1jdGbDvCwMVEoS36bIEai6v87OKsoCT/view?usp=sharing

Part 3:

T.1 Speed of Motor is updated by changing the duty cycle when button is pressed
T.2 Button presses only change the speed once.
T.3 Serial monitor is updated with motor speed information on button press.

The video shows the code running on the Arduino, which updates the motor speed once on button press. The button presses trigger the duty cycle being output to the serial monitor, and not accept multiple duty cycle changes per button press.

Video Link:
https://drive.google.com/file/d/1nTIuEoyPrcziFaSkXYq06oGGzQAGudgu/view?usp=sharing

APPENDIX A. DESIGN CODE

```
#define buttonPin 2
#define PWM_Pin 9
```

```
// Global PWM duty variable (0–ICR1 for 16-bit PWM)
volatile uint16_t duty = 0;

// Button state tracking
bool volatile lastButton = HIGH;
unsigned long lastDebounce = 0;
const unsigned long debounceDelay = 1;


//should have ISR as fast as possible
ISR(TIMER1_COMPA_vect)
{
    //updates the duty cycle
    OCR1A = duty;
}


void setup() {
  Serial.begin(9600);

  pinMode(PWM_Pin, OUTPUT);
  pinMode(buttonPin,INPUT_PULLUP);

  TCCR1A = 0;
  TCCR1B = 0;


  //COM1A1/COM1B1 == 1 ; COM1A0/COM1B0 == 0 it is non-inverting !
  TCCR1A |= (1 << COM1A1);


  //sets to mode 14, fast PWM
  TCCR1A |= (1 << WGM11);
  TCCR1B |= (1 << WGM12) | (1 << WGM13);

  ICR1 = 800;      // sets PWM frequency (~20 kHz with prescaler 1)

  OCR1A = duty;    // initial duty cycle

  TCCR1B |= (1 << CS10);    // prescaler = 1 (full clock speed)

  // Enable compare match interrupt, OCIE1A is the 4th bit of TIMESK1
  TIMSK1 |= (1 << OCIE1A);


  Serial.print("Duty Cycle set to ");
  Serial.print(duty);
  Serial.println("/800 ");
}


void loop() {
  bool reading = digitalRead(buttonPin);

  if (reading != lastButton) {
      lastDebounce = millis();
  }

  //software debouncing here to prevent button from triggering mult times per
press
  if ((millis() - lastDebounce) > debounceDelay) {
      // needs to check incase user is holding the button down.
      if (reading == LOW && lastButton == HIGH) {
        // ---------- Increment duty ----------
        duty += 200;     // step size (adjustable)

        if (duty > ICR1) duty = 0;   // wrap around

        //prints the duty cycle
        Serial.print("Duty Cycle set to ");
        Serial.print(duty);
        Serial.println("/800");
      }
      lastButton = reading;
  }


}
```

END OF DOCUMENT