TITLE AND DATE

Document name: Project #3 PWM Motor Driver Lab Report
Document reference: kiddKid.cmpe311.fall25.project#3
Date of publication: November 30, 2025

LEAD ENGINEER: Kid Kidd, UMBC

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 project extends the previous embedded system by adding a DC motor whose speed is controlled through a push button. The system uses an Arduino Uno R3 along with a MOSFET-based motor driver, a flyback diode, and supporting resistors. Each button press steps the motor through defined PWM duty-cycle levels, cycling from off to full speed and back down. A software PWM generator, a debounced button-handler task, and a motor-control task were added and integrated into the existing cyclic-executive task manager so the motor control operates asynchronously alongside the earlier LED-blinking tasks. This document includes the updated requirements, system design, motor-driver schematic, resistor calculations, testing procedures, and results. The appendix contains the final code and video demonstrations.

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

REFERENCES AND GLOSSARY

REFERENCES:

- CMPE311 Project #3 – PWM Motor Driver Lab Report, Fall 2025
- 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.

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

gcc – front end for the GNU Compiler Collection

Github – A widely used distributed SVC (Software Version Control) system

LED – Light Emitting Diode

DESIGN

DESIGN PRE-REQUISITES:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better

DEVELOPMENT PLATFORM:

1. See DESIGN PRE-REQUISITES above

ANY ADDITIONAL DESIGN CONSIDERATIONS:

1. None

See Appendix A for the code executing on the Arduino Uno R3.
See Appendix B for the video of the test.

Screenshots of the Circuit Setup

Figure 1. LED Brightness Control

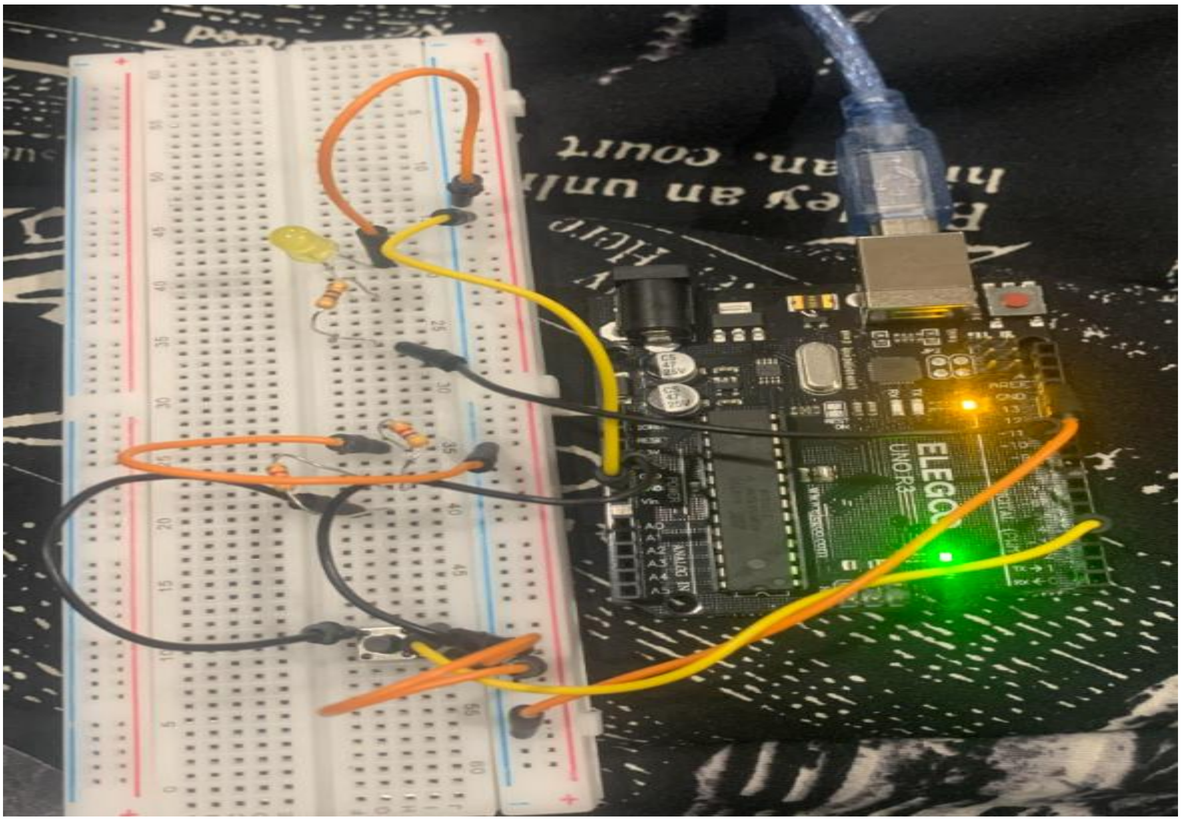
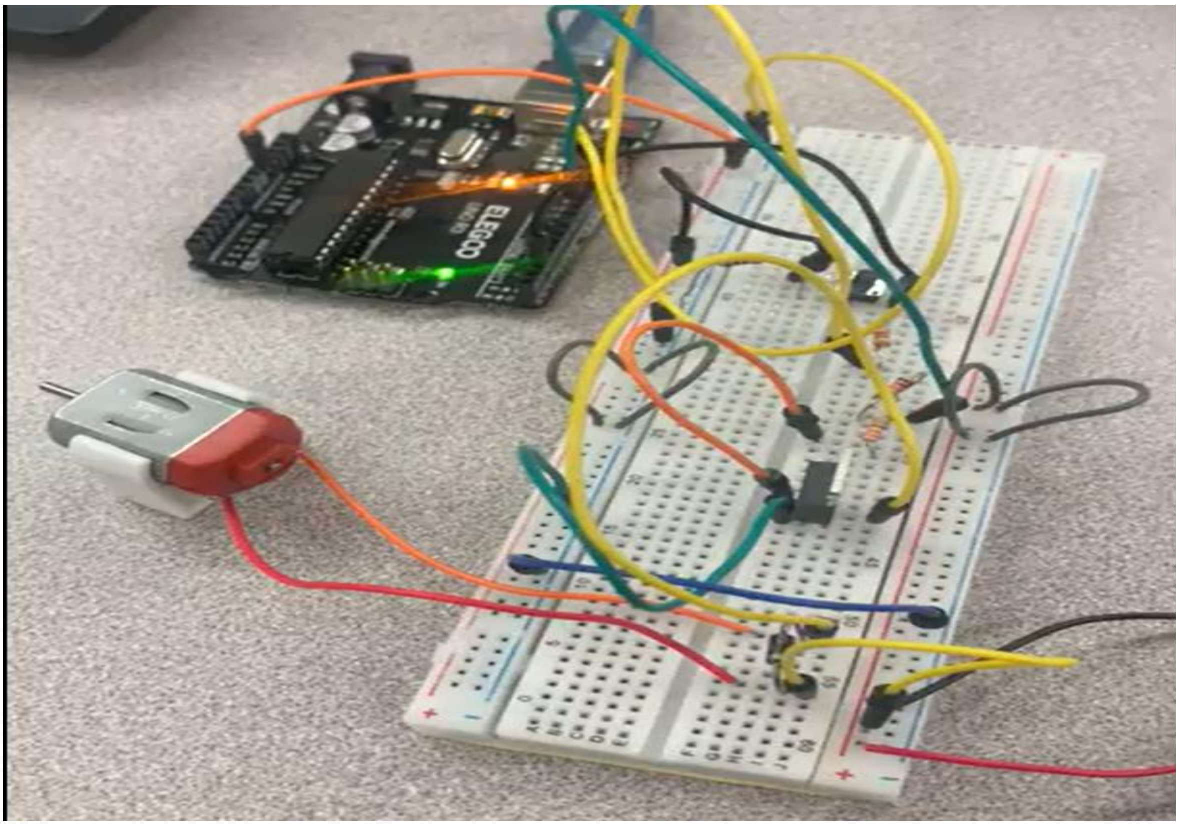


Figure 2. Motor Speed Control



Motor Driver Component Calculations and Justification

R1 – Current Limiting Resistor
R1 is used to limit the current going into the MOSFET gate in case of a fault.
I based the calculation on 10 mA as the maximum safe current for the Arduino pin:

$$R = \frac{5V}{0.01A} = 500\Omega$$

The closest resistor I had was 470Ω, so I used that. It's close enough to the calculated value and keeps the gate current within a safe range.

R2 – Pull-Down Resistor
R2 keeps the MOSFET gate at 0V when the Arduino isn't driving it.
I didn't have a 10kΩ resistor, so I used two 4.7kΩ resistors in series, which gives roughly:

$$4700 + 4700 = 9400\Omega \approx 10k\Omega$$

This works fine as a pull-down and prevents the motor from randomly turning on.

Flyback Diode
A flyback diode is needed because the motor is an inductive load and creates voltage spikes when it turns off.
I placed the diode across the motor terminals. This protects the MOSFET by safely absorbing the spike.

Appendix A. Design Code

LED Brightness Control

```
#include <Arduino.h>

#define PWM_PIN 9
#define BUTTON_PIN 4

const int NUM_LEVELS = 6;
int dutyLevels[NUM_LEVELS] = {0, 51, 102, 153, 204, 255};
int currentLevel = 0;

bool lastButton = HIGH;
unsigned long lastDebounce = 0;

volatile int pwmCounter = 0;
volatile int pwmDuty = 0;

void ledTask();
void buttonTask();
void pwmTask();

typedef void (*TaskFn)();
TaskFn tasks[] = { ledTask, buttonTask, pwmTask };
const int NUM_TASKS = 3;

void setupTimer1() {
  cli();
  TCCR1A = 0;
  TCCR1B = 0;
  TCCR1B |= (1 << WGM12);
  OCR1A = 1999;
  TIMSK1 |= (1 << OCIE1A);
  TCCR1B |= (1 << CS10);
  sei();
}

ISR(TIMER1_COMPA_vect) {
  pwmCounter++;
  if (pwmCounter < pwmDuty) digitalWrite(PWM_PIN, HIGH);
  else digitalWrite(PWM_PIN, LOW);
  if (pwmCounter >= 255) pwmCounter = 0;
}

void setup() {
  Serial.begin(9600);
  pinMode(PWM_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  setupTimer1();
  Serial.println("System Ready. Press button to change brightness.");
}

void loop() {
  for (int i = 0; i < NUM_TASKS; i++) tasks[i]();
}

void ledTask() {
  pwmDuty = dutyLevels[currentLevel];
}

void buttonTask() {
  bool reading = digitalRead(BUTTON_PIN);
  if (reading == LOW && lastButton == HIGH && (millis() - lastDebounce) > 50) {
    currentLevel = (currentLevel + 1) % NUM_LEVELS;
    Serial.print("Brightness level: ");
    Serial.print((currentLevel * 100) / (NUM_LEVELS - 1));
    Serial.println("%");
    lastDebounce = millis();
  }
  lastButton = reading;
}

void pwmTask() {
}
```

Motor Speed Control

```
#include <Arduino.h>

#define BUTTON_PIN 4
#define PWM_PIN 9

volatile int pwmCounter = 0;

const int NUM_STEPS = 8;
int dutySteps[NUM_STEPS] = {0, 2, 4, 6, 8, 6, 4, 2};
int dutyIndex = 0;

bool lastButton = HIGH;
unsigned long lastDebounce = 0;

void taskButton();
void taskMotorPWM();
void taskEmptyPWM();

void (*taskQueue[])(void) = {
  taskButton,
  taskMotorPWM,
  taskEmptyPWM,
  NULL
};

void setupTimer1() {

  cli();

  TCCR1A = 0;
  TCCR1B = 0;
  TCCR1B |= (1 << WGM12);

  OCR1A = 1999;

  TIMSK1 |= (1 << OCIE1A);
  TCCR1B |= (1 << CS10);

  sei();
}

ISR(TIMER1_COMPA_vect) {

  pwmCounter++;

  if (pwmCounter < dutySteps[dutyIndex])
    digitalWrite(PWM_PIN, HIGH);
  else
    digitalWrite(PWM_PIN, LOW);

  if (pwmCounter >= 8)
    pwmCounter = 0;
}

void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(PWM_PIN, OUTPUT);

  setupTimer1();

  Serial.println("Project 3 Ready – Press button to change motor speed.");
}

void loop() {
  int i = 0;
  while (taskQueue[i] != NULL) {
    (*taskQueue[i])();
    i++;
  }
}

void taskButton() {
  bool reading = digitalRead(BUTTON_PIN);

  if (reading == LOW && lastButton == HIGH && (millis() - lastDebounce) > 50) {

    dutyIndex++;
    if (dutyIndex >= NUM_STEPS) dutyIndex = 0;

    Serial.print("Duty step: ");
    Serial.print(dutySteps[dutyIndex]);
    Serial.println("/8");

    lastDebounce = millis();
  }

  lastButton = reading;
}

void taskMotorPWM() {
}

void taskEmptyPWM() {
}
```

Appendix B. Video

LED Brightness Control

https://www.youtube.com/shorts/EaOELL9fFKk

Motor Speed Control

https://www.youtube.com/shorts/AJfdlWkYg2s