Course: CMPE311-Project 3
DATE: Dec 1, 2025
TO: Professor Kidd
FROM: Abhinna Das
SUBJECT: Duty cycle project

Part 1

Part I: (PWM) Build the embedded system to change the motor RPM as described above. For this part, instead of driving an actual motor, use the motor digital control signal to control an LED's intensity instead of the speed of the motor.

Part 2

Part II: (Driver) The embedded system (i.e. the Arduino) is not needed for this part. Build and test a MOSFET-based motor driver circuit. See Table A for the DC motor's specification. Test thecircuit by connecting the motor and setting the control signal input voltage to 0Vdc (0 RPM) and two 1.5Vdc batteries in series. Use one to verify that the motor runs. Make sure you place a resistor in series to limit the current draw of the motor.

Part 3

(Integration) Connect your embedded system to the driver circuit and demonstrate its operation.
Rubric (Demonstration of successful operation): (1) Pressing the button cycles the speed (RPMs) of the motor and fan from off to maximum and back down. (2) This operation operates asynchronously with the blinking LEDs from Project #2.

Question answering

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.)

Answer: The max current output for the Arduino pin is 40 mA howeve,r it is recommended to only output 20 mA as such the resistor to limit the current to the gate needs to be at least 40mA as such using ohms law we have V = IR since where V = 5 V and I = 40 mA for the max and 20 mA for the recommended. As such, the minimum resistance for R1 is 125 ohms; anything higher would still protect the Arduino.

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

Course: CMPE311-Project 3
DATE: Dec 1, 2025
TO: Professor Kidd
FROM: Abhinna Das
SUBJECT: Duty cycle project

Answer: You want to make sure that the resistor value is not big enough to limit the draw a lot of current from the Arduino.

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

Video:
https://drive.google.com/file/d/1AdBm6QapT5gWINPT-9TEk6rVkz4kZFi1/view?usp=sharing

Code section:

```
// --- Hardware Definitions ---
const int PWM_PIN = 9;
const int BUTTON_PIN = 2;

// --- PWM Constants ---
const unsigned long PWM_PERIOD_MS = 1000; // 1000ms = 1 Hz (1 Second)

// --- Logic Constants ---
const unsigned long DEBOUNCE_DELAY = 50;

// --- Global Variables ---
unsigned long pwmOnTimeMs = 0;  // How long pin stays HIGH per cycle
unsigned long lastCycleStart = 0; // Timestamp of when the current second
started

unsigned long lastDebounceTime = 0;
int buttonState = LOW;
int lastButtonState = LOW;
int sequenceIndex = 0;

const float dutyCycleSequence[] = {0.0, 0.25, 0.50, 0.75, 1.0, 0.75, 0.50,
0.25};
const int SEQUENCE_LENGTH = 8;

void setup() {
  pinMode(PWM_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
```

Course: CMPE311-Project 3
DATE: Dec 1, 2025
TO: Professor Kidd
FROM: Abhinna Das
SUBJECT: Duty cycle project

```
  Serial.begin(9600);
  Serial.println("System Starting (Software PWM Mode - 1 Hz)...");

  // Start with 0% duty
  setDutyCycle(0.0);
}

void loop() {
  handleButton();
  runSoftwarePWM(); // <--- This function replaces the ISRs
}

// --- The New PWM Function ---
// This replaces the ISRs. It checks the clock and toggles the pin.
void runSoftwarePWM() {
  unsigned long currentMillis = millis();

  // 1. Check if the 1-second cycle has finished. If so, restart the
timer.
  // (This replaces ISR(TIMER1_OVF_vect))
  if (currentMillis - lastCycleStart >= PWM_PERIOD_MS) {
    lastCycleStart = currentMillis;
  }

  // 2. Determine if we are in the "ON" part or "OFF" part of the cycle.
  // (This replaces the logic of ISR(TIMER1_COMPA_vect))
  unsigned long elapsedTime = currentMillis - lastCycleStart;

  if (elapsedTime < pwmOnTimeMs) {
    digitalWrite(PWM_PIN, HIGH);
  } else {
    digitalWrite(PWM_PIN, LOW);
  }
}

// --- Logic to Set the Duty Cycle ---
void setDutyCycle(float ratio) {
```

Course: CMPE311-Project 3
DATE: Dec 1, 2025
TO: Professor Kidd
FROM: Abhinna Das
SUBJECT: Duty cycle project

```
  // Calculate how many milliseconds the LED should be ON
  pwmOnTimeMs = (unsigned long)(PWM_PERIOD_MS * ratio);
}

// --- Button Logic (Same as before) ---
void handleButton() {
  int reading = digitalRead(BUTTON_PIN);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == HIGH) {
        advanceSequence();
      }
    }
  }
  lastButtonState = reading;
}

void advanceSequence() {
  sequenceIndex++;
  if (sequenceIndex >= SEQUENCE_LENGTH) {
    sequenceIndex = 0;
  }

  float currentRatio = dutyCycleSequence[sequenceIndex];

  Serial.print("Button Pressed. Duty: ");
  Serial.print(currentRatio * 100);
  Serial.println("%");

  setDutyCycle(currentRatio);
```

Course: CMPE311-Project 3
DATE: Dec 1, 2025
TO: Professor Kidd
FROM: Abhinna Das
SUBJECT: Duty cycle project

```
}
```