TITLE AND DATE

Document name: *add duty cycle*
Document reference: *Bearden, Emily*.cmpe311.fall25.*project3*
Date of publication: *December 2, 2025*

LEAD ENGINEER: *Emily Bearden, 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 project adds a motor-speed controller ot the asynchronous embedded system developed in previous labs. A push button is used to cycle the motor (or LED) substitute) through a sequence of PWM duty-cycle levels. Pressing the button repretedly increases the duty cycle from OFF → 2/8 → 4/8 → 6/8 → 8/8, the decreases back down to OFF, repeating indefinitely.

The system uses:

1. A MOSFET-based motor driver circuit,
2. A PWM generator built using hand-written ATmega328P ISR code,
3. A software debouncer task,
4. A butter handler task, and
5. Integration with the asynchronous task manager from Project #2

The project demonstrates asynchronous tasking, PWM generation, hardware driver integration, and safe switching of inductive loads.

DESCRIPTION: The goal of this design was to create an embedded system capable of cycling the speed of a DC motor using a single push button. Because a DC motor's speed is proportional to the average applied voltage, the system varied the PWM duty cycle rather than the voltage source directly.

Before attaching the motor itself, an LED is used to emulate the PWM behavior for safe testing. When the LED's brightness correctly follows the stepping pattern, the motor driver is attached and tested using the MOSFET switching circuit.

The system must operate asynchronously with the LED-blinking tasks from earlier projects, ensuring that PWM generation, button debouncing, blinking lEDs, and system communications do not interfere with each other.

REFERENCES:

- ATmega328P Datasheet
- Project Add Duty Cycle Specification
- ELEGOO Uno R3 documentation

DEFINITIONS AND ABBREVIATIONS:

- 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.

- “PWM” - Pulse Width Modulation
- “MOSFET” - Metal-Oxide-Semiconductor Field-Effect Transistor
- “Duty Cycle” - % of time the PWM output is HIGH during one period
- “Debounce” - Software technique to eliminate false button transitions
- “Driver Circuit” - Circuit used to switch motor current without damaging the microcontroller

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

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 are started with “T.#”

CUSTOMER REQUIREMENTS:

C1. The User must be able to increase and decrease the speed of the motor by repeatedly pressing a push button
C2. The system must cycle though at least 5 discrete duty-cycle levels as defined in the project description.
C3. Button presses must be debounced so the motor/LED does not skip duty-cycle levels.
C4. The system must run on an Arduino Uno R3 compatible development board.
C5. The PWM signal must be generated via hand-coded ISR logic.
C6. The system must integrate asynchronously with the previously implemented LED blink tasks.

HIGH-LEVEL TECHNICAL REQUIREMENTS:

HL.1 The system must implement a timer-based PWM generator.
HL.2 The PWM generator must produce 5 levels of duty cycle: 0%, 25%, 50%, 75%, 100%.
HL.3 An asynchronous button handler task must detect presses without blocking other tasks.
HL.4 The button input must be debounced in software.
HL.5 The system must use a MOSFET motor driver circuit to safely control a DC motor.
HL.6 Pressing the button must update the PWM output within ≤1 cycle.
HL.7 PWM, button handling, and previously implemented Project #2 tasks must all run concurrently.

DESIGN:

Design prerequisites:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better
3. Wires
4. N-channel MOSFET
5. Flyback diode
6. Resistors
7. Pushbutton +10kohm resistor
8. DC motor
9. Breadboard

Development platform:

1. Arduino IDE 2.3.3 or better

Testing Platform:

1. ELEGOO Uno R3
2. Arduino IDE 2.3.3
3. USB-powered testbed
4. PWM output to LED
5. PWM output to MOSFET driver controlling the motor

Table 1: Required Test Dialog

User Action	**Expected System Output**	**Notes**
Initial State	Motor/LED off	Duty = 0/8
Push button	Duty = 2/8	LED slightly on
Push button	Duty = 4/8	LED medium brightness
Push button	Duty = 6/8	LED bright
Push button	Duty = 8/8	LED max
Push button	Duty = 6/8	Decreasing begins
Push button	Duty = 4/8	
Push button	Duty = 2/8	
Push button	Duty = 0/8	Off

This sequence must repeat indefinitely and operate asynchronously with all prior system tasks.

Testing and Validation:

Table 2: Testing and validation requirements

Requirement	Description
T.0	All testing must be performed on the physical testbed in Figure 1
T.1	Each button press must advance the duty cycle exactly one step
T.1.1	The motor speed must change without interfering with blink tasks.
T.2	The button must be correctly debounced.
T.3	Duty cycle must match the expected PWM values.
T.4	The PWM output must be generated by ISR code only.
T.5	The cycle must work continuously for at least 20 consecutive button presses
T.6	When the motor driver circuit receives the 5v. the motor must turn on; at 0v it must turn off.
T.7	The integrated motor driver must operate without damaging the microcontroller.

Table 3: Results of testing

Test Performed	**Result**
T.1	Satisfied - each press changed one duty step.
T.1.1	Satisfied - LEDs from Project #2 continues blinking
T.2	Satisfied - debouncer eliminated bounce-induced double steps.
T.3	Satisfied - PWM waveform observed an oscilloscope
T.4	Satisfied - no Arduino analogWrite() used
T.5	Satisfied - passes 20-press stability test.
T.6	Satisfied - verification with motor driver
T.7	Satisfied - MOSFET driver operated correctly with flyback diode protection

Appendix A:

Github code link

Appendix B:

Answers to problems:

Problem 1 - Calculate R1 to limit current through the gate to the maximum acceptable fo ran Arduino Uno digital pin

Assumptions/facts used:

- Arduino Uno recommended DC output drive per I/O pin = 20 mA
- MCU output voltage when HIGH = 5v
- R1 only carries current if the MOSFET gate is effectively shorted to ground or another low-voltage node

Calculation: $R = \frac{V}{I} = \frac{5v}{0.02A} = 250\Omega$

If a 250Ω resistor is not available, use a 330Ω resistor to safely operate below amp limit or a 220 Ω resistor to be slightly above the recommended amp limit.

Problem 2 - What constraints must you be aware of in determining an acceptable value for R2 (gate pulldown)?

R2 is the resistor from MOSFET gate to ground that keeps the gate low when the MCU pin is high-impedance. Constraints and trade-offs include:

1. Hold ability (must keep gate low/noise immunity):
    a. R2 must be low enough that environmental nouse or leakage currents won't

raise the gate to an intermediate voltage and partially turn the MOSFET on. If the gate capacitance is C_g and there is a small leakage/noise current I_{leak}, the DC gate voltage is $V_g = I_{leak} * R2$. Smaller R2 → better hold.

2. Turn-off time when the MCU goes high-z
    a. Gate discharge time constant $\tau = R2 * C$. Smaller R2 → faster turn-off. If you want the gate to discharge to <5% in t_{off}, choose $R2 \leq t_{off}/(3C_g)$
3. Voltage divider effect with R1
    a. When the MCU drives the gate HIGH through R1, R1 and R2 form a divider. To ensure the gate sees 5V, R2 should be much larger than R1. Approximate gate voltage when driver high is $V_{gate} \approx 5 \times \frac{R2}{R1+R2}$ with R1 = 330Ω and choosing R2 = 10kΩ yields $V_{gate} \approx 4.84$
4. Power dissipation when MCU is driving HIGH
    a. Current through R1→R2 when MCU outputs HIGH is $I = \frac{5}{R1+R2}$. Large R2 reduces continuous current draw.
5. Practical gate capacitance:
    a. Small MOSFETs: C_g typically 1-10nF. Larger MOSFETs can be larger. This strongly affects needed R2 for desired switching speed.

Best choice: R2 = 10kΩ.

Problem 3 - What is the minimum response time theoretically possible by the ATmega329P running at 16 MHz?

Clock period: $T_{clk} = \frac{1}{16MHz} = 62.5ns$