# CMPE 311 Project 5 – Duty Cycle Motor Control System Free RTOS

Document name: CMPE 311 Lab Report 5

Document reference: Hoover,Ian.cmpe311.fall25.project#5

Date of publication: 12/10/25

**LEAD ENGINEER:**
Ian Hoover, UMBC

**STAKEHOLDERS:**
Prof. Kidd, Instructor, UMBC

MD Safwan Zaman, TA, UMBC

## HIGH-LEVEL DESCRIPTION:

This project requires the design and implementation of a PWM-controlled motor system using an Arduino Uno R3 compatible development board. The system implements a PWM-controlled DC motor using an Arduino Uno R3 and FreeRTOS. Independent RTOS tasks manage motor control, button input, and LED blinking concurrently. A binary semaphore is used to synchronize button events with motor duty-cycle updates, ensuring non-blocking operation.

## DESCRIPTION:

The document defines customer requirements, technical requirements, and testing requirements. Software uses FreeRTOS task scheduling with Timer2 interrupts for software-generated PWM. Button input is debounced within a dedicated RTOS task and synchronized with the motor control task using a binary semaphore. This architecture replaces the cyclic executive used in project 4.

## REFERENCES:

- How to do Multitasking with the Arduino
- The Arduino Handbook – Learn Microcontrollers for Embedded Systems
- sec04b-engineeringProcess_pt1.pdf
- sec04b-engineeringProcess_pt2.pdf
- CMPE310 asynchronous Arduino lab
- Atmega328p Datasheet

## DEFINITIONS AND ABBREVIATIONS:

- “Customer requirements” - Requirements defined by The Customer.
- “The Requirements” - High-level technical requirements.
- “Serial monitor” - Arduino IDE serial interface.

- “PWM” - Pulse Width Modulation.
- “ISR” - Interrupt Service Routine.
- “MOSFET” - Metal-oxide-semiconductor field-effect transistor. The transistor used to drive the motor.
- “LED” - Light-emitting diode
- “IDE” - Integrated development environment
- “Arduino” - An Italian open-source hardware and software company; also refers to a development board created by the company
- “gcc” - front end for the GNU compiler collection
- “FreeRTOS” – A real-time operating system used to schedule concurrent tasks.
- “Task” – An independently scheduled thread of execution managed by FreeRTOS.
- “Semaphore” – A synchronization primitive used to signal events between tasks.

**CONVENTIONS:**

- Must, shall, or will – mandatory
- May - optional.
- All Customer requirements are started with “C.#”
- All technical requirements are started with “HL.#”
- All testing requirements are started with “T.#.”

**CUSTOMER REQUIREMENTS:**

C1. The User must be able to control the motor duty cycle via a hardware push button.

C2. The System must cycle through nine duty-cycle levels.

C3. The System must not block LED-blinking logic.

C4. The System must run on an Arduino Uno R3.

C5. The System must visually reflect PWM output changes.

**HIGH-LEVEL TECHNICAL REQUIREMENTS:**

HL.1 PWM must be implemented using Timer2 interrupts.

HL.2 Button input must be debounced in software.

HL.3 Duty-cycle updates must be printed to the Serial Monitor.

HL.4 The System must use FreeRTOS to manage concurrent tasks for motor control, button input, and LED blinking.

HL.5 MOSFET circuit must include a flyback diode.

HL.6 The System must use an Arduino Uno R3 compatible board.

HL.7 Task synchronization between the button input and motor control shall be implemented using a FreeRTOS semaphore.

**TESTING AND VALIDATION REQUIREMENTS:**
T.1 Confirm power and IDE recognition.

T.2 Verify nine-step duty-cycle sequence.

T.3 Serial output must reflect correct duty levels.

T.4 LEDs must blink independently during PWM operation.

T.5 Motor must accelerate/decelerate through all levels.

T.6 Oscilloscope verification optional but recommended.

**DESIGN:**
The circuit includes two LEDs on pins 8 and 9, a push button on D2 using INPUT_PULLUP, and a MOSFET motor driver connected to D3. A flyback diode protects the MOSFET. PWM is generated in a Timer2 compare-match interrupt service routine. Motor duty-cycle updates are performed in a dedicated FreeRTOS task, which is triggered by a binary semaphore released by the button input task.

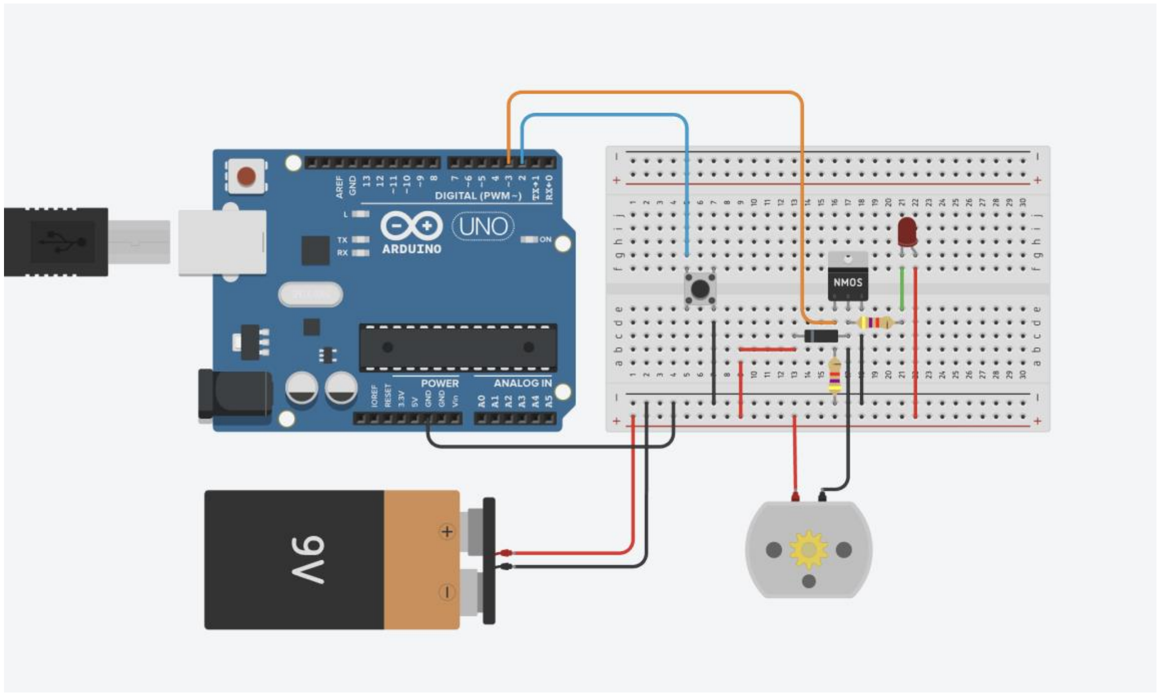

**Figure 1: System wiring diagram. The motor, diode, push button, battery, and MOSFET can be seen above. From pins from left to right on the MOSFET are the gate, drain, and source, respectively.**

The Voltage that comes from each pin of the Arduino is 5V. The Arduino also has a recommended current of between 20-40 mA for each pin. To calculate the resistance, $R_1$, we simply take $\boldsymbol{I} = \frac{\boldsymbol{V}}{\boldsymbol{R}}$, or $\mathbf{20}\ \boldsymbol{mA} = \frac{\mathbf{5}\boldsymbol{V}}{\boldsymbol{R_1}}$, giving a value of **250 Ω or greater for $R_1$**. Thus, the **470 Ω resistor** given to us by Dr. Kidd is an appropriate value for $R_1$. The second resistor's value can be calculated in a similar way. We don't want its value to be so large that the gate of the MOSFET discharges very slowly. We also need it to be small enough to not overload the pin when it receives a HIGH voltage. Assuming 1 mA of current is an acceptable amount, we can now do the same calculation as with $R_1$. $\mathbf{1}\ \boldsymbol{mA} = \frac{\mathbf{5}\boldsymbol{V}}{\boldsymbol{R_2}}$, or **$R_2$ = 4.7 k Ω**, which is what we were given to do the project. Unlike the previous project, motor-control is event-driven using a FreeRTOS semaphore, ensuring that each button press results in exactly one duty-cycle transition regardless of task scheduling.

**TESTING:**

| **Serial Port I/O** | **Notes** |
|---|---|
| Motor is initially off | Initial state after the circuit is powered |
| Button press -> "Motor operates at 2/8 duty cycle" | First increase in speed, LED turns on |
| Button press -> "Motor operates at 4/8 duty cycle" | Second increase in speed, LED brightens |
| Button press -> "Motor operates at 6/8 duty cycle" | Third increase in speed, LED brightens again |
| Button press -> "Motor operates at 8/8 duty cycle" | Final increase in speed, LED is at its brightest |
| Button press -> "Motor operates at 6/8 duty cycle" | First decrease in speed, LED dims slightly |
| Button press -> "Motor operates at 4/8 duty cycle" | Second decrease in speed, LED dims again |
| Button press -> "Motor operates at 2/8 duty cycle" | Third decrease in speed, LED dims again |
| Button press -> "Motor is off" | Motor is powered off, LED is off, and we are back to initial state |

**Table 1: Results from testing**

Serial output during testing confirms correct transitions. The testing requirements, T.1-T.5, are satisfied since each button press corresponds with both a change in brightness in the LED and an increase or decrease in speed of the motor. The serial monitor also receives output that tells the user which duty level is currently being used. Additionally, the LED operates at the same time as the motor, and motor control, button input, and LED blinking are implemented as independent FreeRTOS tasks. The only test yet to be verified is T.7, which could be done in an on-campus lab if I had access.

**END OF DOCUMENT**