ENGR 478 - LAB

Experiment #5:

Building Systems with External LED and Switch Interfaces

Edge-Triggered Interrupt on GPIO

Team name: **The Nucleo**

Team members:

- Nhan Nguyen
- Vi Gallegos

Date experiment performed: Apr. 08, 2025

Date report submitted: Feb. 24, 2025

Objective

- The objective of this lab is to explore GPIO control, timer-based delay, and external interrupt handling on the STM32L476 microcontroller through a three-part system:
- Part 1: Implement a basic LED blinking system where LEDs toggle every 0.5 seconds using the SysTick timer interrupt. This part demonstrates timer configuration and periodic task scheduling.
- Part 2: Configure two pushbuttons (SW1 and SW2) to toggle two LEDs (LED1 and LED2) independently using edge-triggered external interrupts (EXTI). Button presses are debounced using Timer 6, providing clean and reliable input detection.
- Extra Functionality: Extend the system into a 2-bit rotary counter, where SW1 increments and SW2 decrements the counter. The current counter value is displayed in binary via LED1 (LSB) and LED2 (MSB). The counter wraps around in the 2-bit range (0–3) and updates only on valid, debounced button presses.

Procedure:

⚙️ Hardware Wiring

SW1 → PC2:

- Connect one side of SW1 to VCC (3.3V)
- Connect the other side to PC2
- Enable internal pull-down on PC2
- Result:
  - Not pressed: PC2 is pulled LOW → 0
  - Pressed: PC2 is connected to VCC → 1

SW2 → PC3:

- Connect one side of SW2 to GND
- Connect the other side to PC3
- Enable internal pull-up on PC3
- Result:
  - Not pressed: PC3 is pulled HIGH → 1
  - Pressed: PC3 is connected to GND → 0

LED1 → PB4:

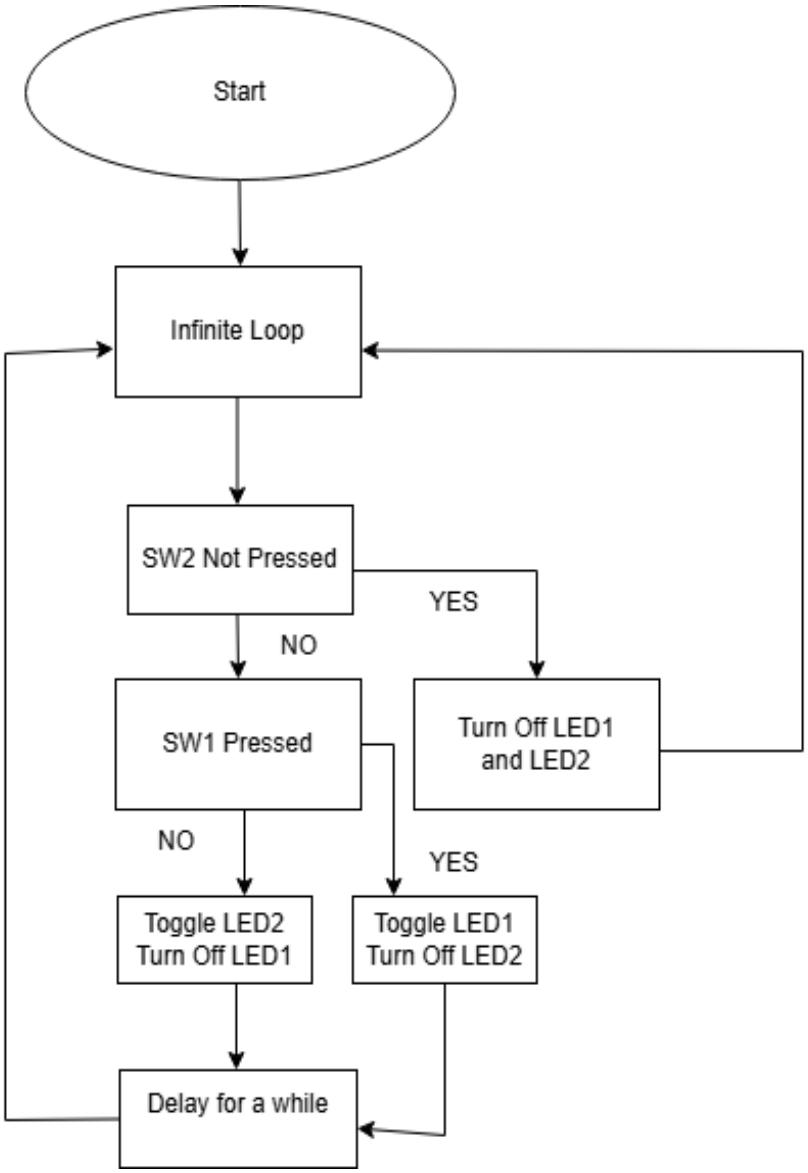
- Connect one side of LED1 to a resistor and then to GND
- Connect the other side to PB4

LED2 → PB5:

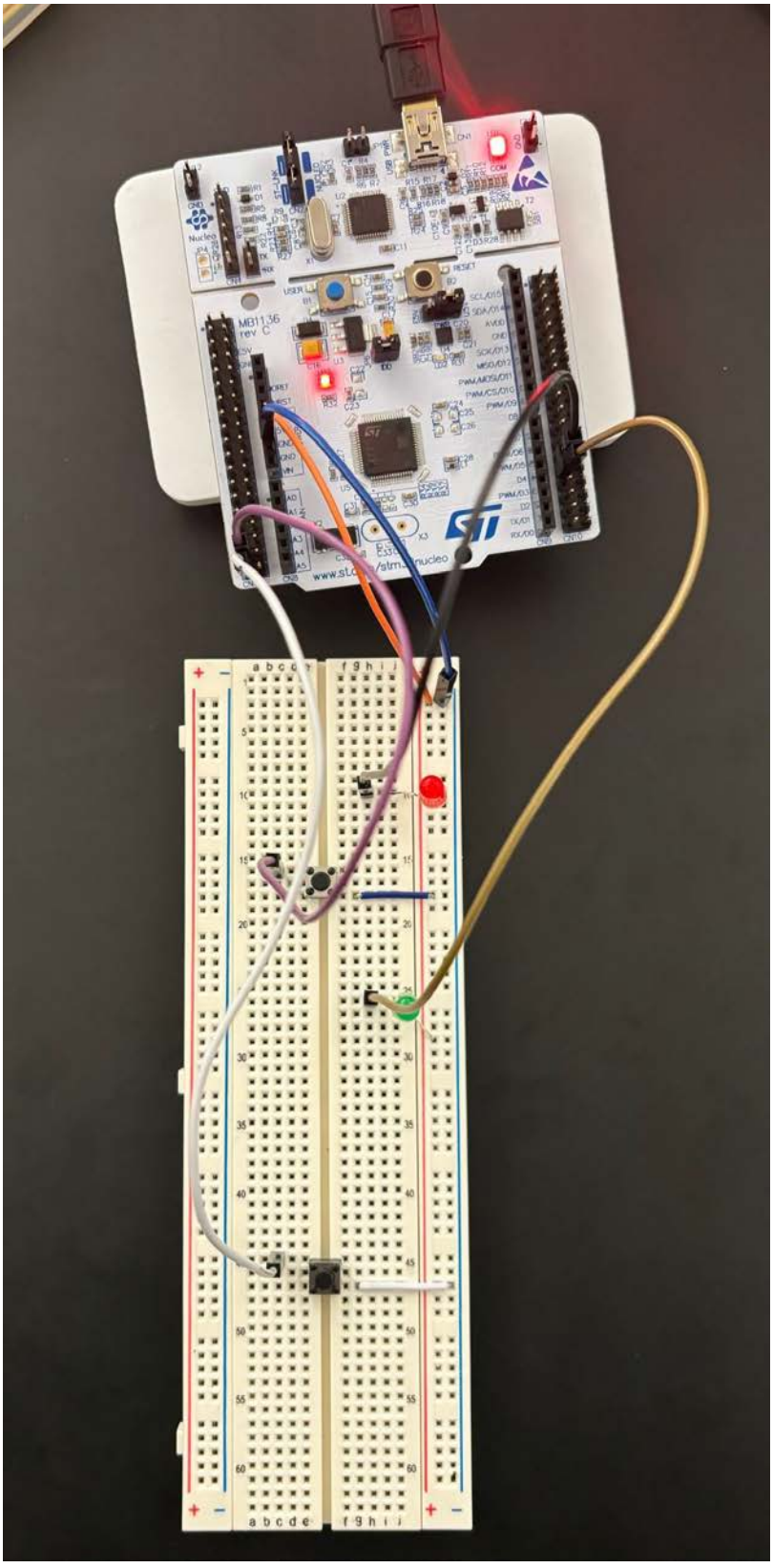
- Connect one side of LED1 to a resistor and then to 3.3V
- Connect the other side to PB5

Part A:

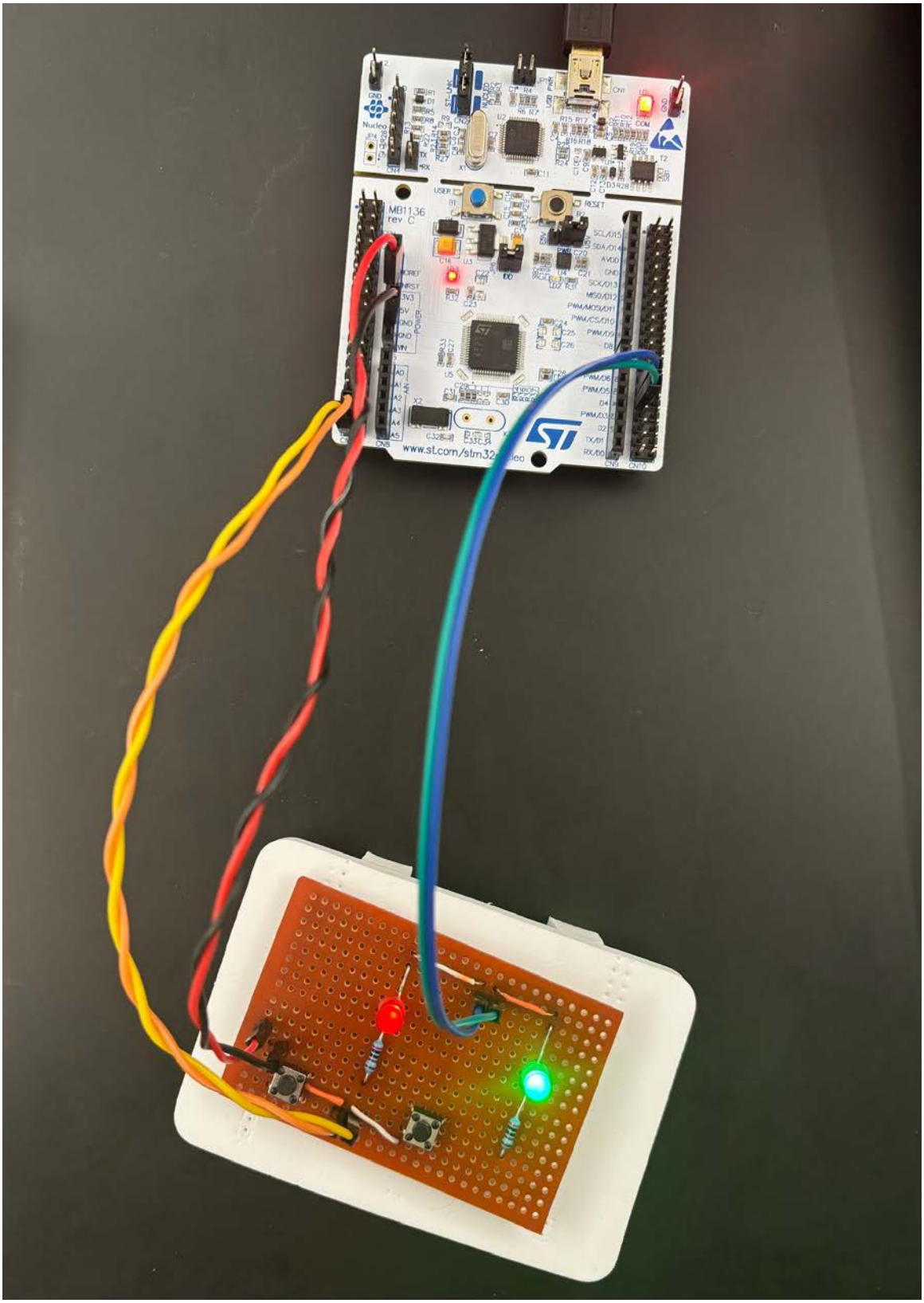
Flowchart:



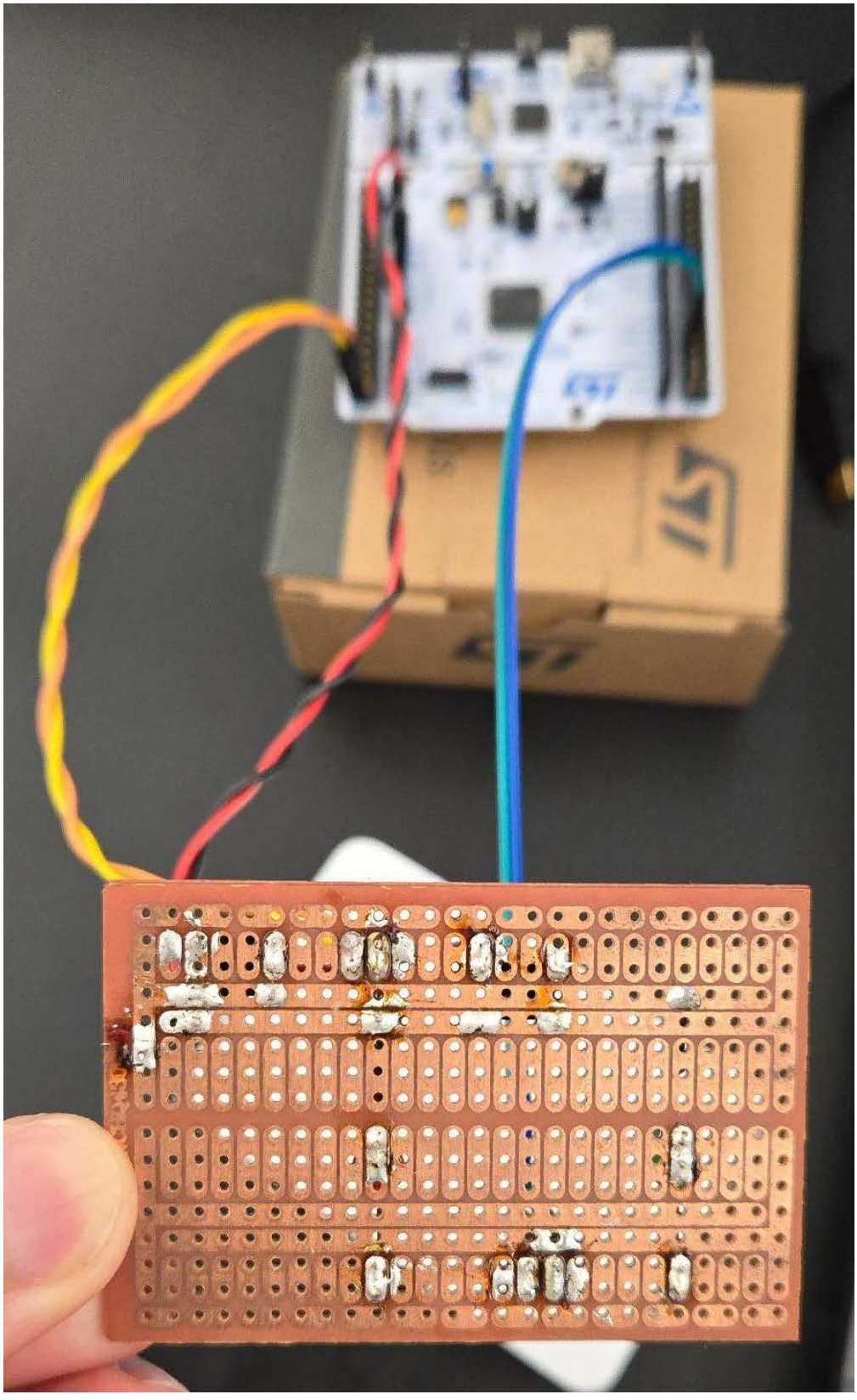
Circuit:



Solder circuit version (top view):



Solder circuit version (bottom view):

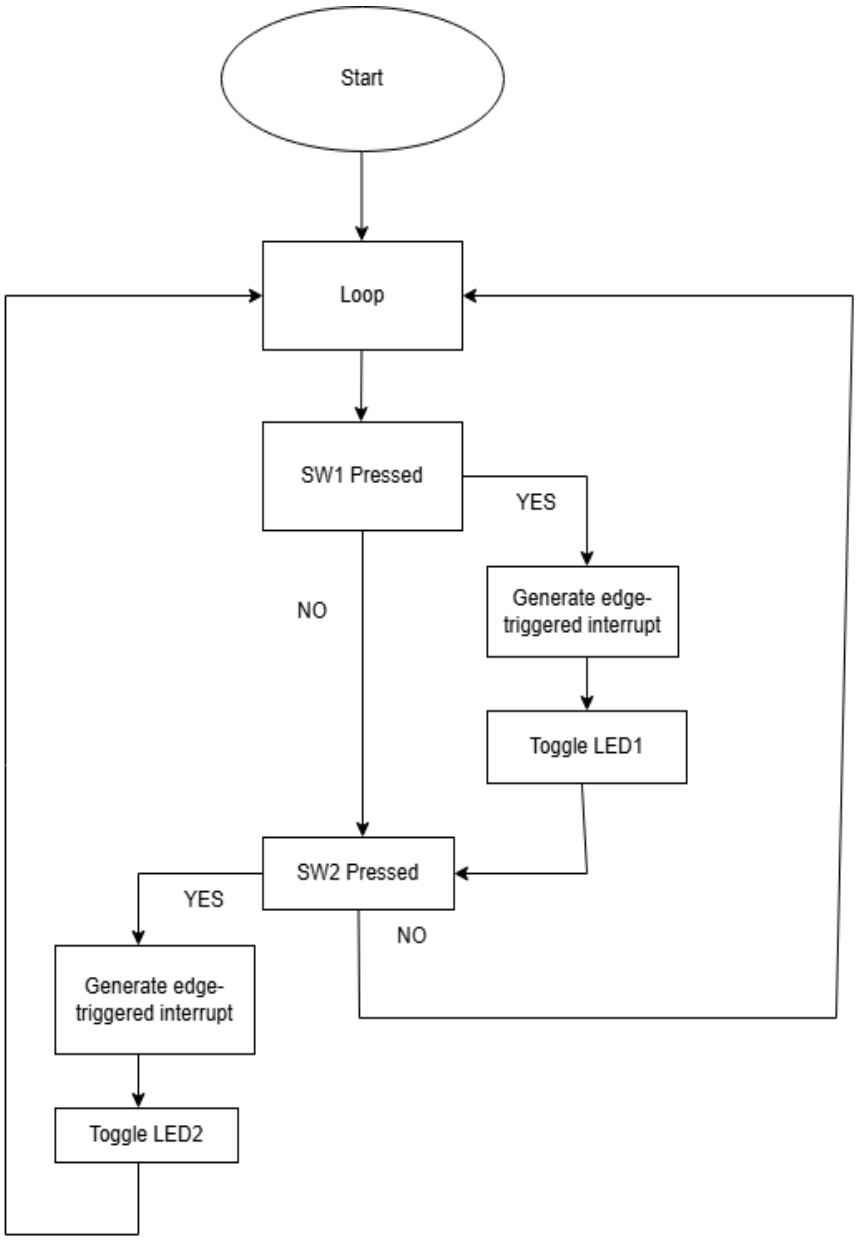


Results:

Based on our program testing, the system worked as expected with no issue. When testing it, SW1 being either pressed or not toggled either LED or in the case of SW2 keeped both LEDs off.

Part B:

Flowchart:



Circuit:

Circuit is unchanged from Part A, only the code is changed.

Results:

The program worked as expected and successfully demonstrated the use of edge-triggered external interrupts to control two independent LEDs based on button input. When SW1 was pressed, the system generated a rising-edge interrupt that triggered a debounced toggle of LED1. Similarly, pressing SW2 caused a falling-edge interrupt, resulting in a debounced toggle of LED2.

Do Something Cool:

The objective of this part is to develop a 2-bit rotary counter controlled by two push-buttons (SW1 and SW2) and displayed using two LEDs (LED1 and LED2). The system uses edge-triggered external interrupts to detect button presses, Timer 6 for input debouncing, and bitwise logic to update and display the counter value in binary form. This program highlights real-time responsiveness, state control, and modular embedded C programming.

Two LEDs (LED1 and LED2) display the current binary value of the counter, with LED1 representing the least significant bit (LSB) and LED2 representing the most significant bit (MSB). The counter operates cyclically within a 2-bit range (values 00 to 11 or 0 to 3).

When the counter starts at 00, pressing the increment switch (SW1) changes the state to 01, turning on only LED1. Pressing it again sets the counter to 10, turning on only LED2. A third press sets the counter to 11, lighting up both LEDs. This binary output provides immediate visual feedback for each state transition.

All push-button inputs are detected using edge-triggered external interrupts, and Timer 6 is used to debounce each press, ensuring reliable and noise-free transitions between states.

Reflection:

This lab reinforced my understanding of embedded system fundamentals by giving me the opportunity to apply key concepts such as GPIO configuration, timer usage, and interrupt handling on a real microcontroller platform. Working directly with the STM32L476 allowed me to see how hardware and software interact at a low level, and how precise control over timing and input detection is essential in building reliable embedded applications.

Through the process, I gained practical experience in modular C programming, debugging timing-sensitive logic, and managing asynchronous events like button presses. I also learned the importance of debouncing in physical input systems and how hardware timers can be used to solve real-world problems in a clean and efficient way.

Overall, the lab deepened my confidence in designing responsive, real-time systems and strengthened my ability to translate system requirements into structured, maintainable embedded code. It was a meaningful step toward building more complex and dependable microcontroller-based projects.

Appendix:

All the code for this lab can be found at: https://github.com/nhannguyensf/lab5-engr478