Royal Military College of Canada
EEE307 Computer Interfacing Techniques
Assignment #1
Winter 2025

Course Instructor: Dr. Hussein Ammar **Due Date:** Monday Feb 3, 2025

Question: Pulse Width Modulation

You need to write an embedded C program to configure the STM32G491RE microcontroller to control four LED signaling lights at Kingston Airport. The lights are used to signal the takeoff of an airplane through a specific sequence of dimming, which provides visual guidance to pilots, ground crew, and controllers. At the beginning all LEDs are initially OFF. Then, the following sequence starts:

1. LED 1 gradually brightens from **low dim to full brightness** using timing pattern D1 defined below and remains ON.
2. LED 2 gradually brightens from low dim to full brightness using dimming pattern D1 defined below and remains ON.
3. LED 3 gradually brightens from low dim to full brightness using dimming pattern D1 defined below and remains ON.
4. LED 4 gradually brightens from low dim to full brightness using dimming pattern D1 defined below and remains ON.
5. Repeat the whole steps again

You will need to generate a single PWM signal that will control each LED one at a time. The PWM signal will use channel 2 of timer 3, i.e., TIM3_CH2, and it will control each LED through the sequence described above. The LEDs will be connected as follows:

- LED1 connected to PA4 which can use TIM3_CH2 through alternate function AF2
- LED2 connected to PA7 which can use TIM3_CH2 through alternate function AF2
- LED3 connected to PB5 which can use TIM3_CH2 through alternate function AF2
- LED4 connected to PC7 which can use TIM3_CH2 through alternate function AF2

Dimming pattern D1:

- The period of the PWM signal will be 20 ms, with a duty cycle that starts at 5% then increments by 0.5% (simply increment CCR2 by 1) **every timer overflow** till it reaches a duty cycle of 100%. This will produce the dimming effect for the LED till it lights brightly at the end when the duty cycle reaches 100%.

Your main() method will look as follows:

```c
int main(void) {
    GPIO_Config();
    TIM_Config();

    while (1) {
        // Main loop does nothing, PWM is generated by hardware
    }
}
```

a) Implement the function GPIO_config(). This function should configure the pins connected to the LEDs as outputs except for PA4 which will be connected to TIM3_CH2. After doing the

initialization, output LOGIC HIGH to the output pins to turn OFF LEDs 2, 3, and 4. <u>Note:</u> The LEDs will turn OFF when 1 is outputted on the pins and ON when 0 is outputted. ***(20 pts)***

b) **For the PWM signal, using PSC = 1599 how much should be ARR to produce the needed signal period? What should be the value of CCR2 for duty cycles 5%, 10%, 40%, 70% and 100%? <u>Note:</u> write down your value for ARR and show your calculations.** ***(20 pts)***

c) **The LEDs need to gradually brighten from low dim to full brightness. How this can be done using PWM? By considering that the LEDs turn ON when logic low (0) is outputted on the controlling pin, which PWM mode should you use?** ***(10 pts)***

d) **Implement the function TIM_config() which will set the starting configuration for the CCR1 and the other needed configurations for TIM3_CH2.** ***(20 pts)***

To move between each step in the LEDs sequence, you will need to create a variable **led_number**, which is a flag that will indicate the LED being controlled by PWM. Define the variable as a global variable (outside main() method) as follows:

```
uint32_t led_number = 1;
```

This variable needs to be updated in the interrupt subroutine (ISR), where after LED1 reaches a duty cycle 100%, it will stay ON (pin redefined in the ISR as output), and then LED2 will be connected to TIM3_CH2. Then, when LED2 reaches a duty cycle of 100%, it will stay ON, and then LED3 will be connected to TIM3_CH2, and so on until you go through the whole sequence. At each cycle, **led_number** will be updated and used to track your sequence. At the end, you will reset **led_number** back to 1.

e) **Implement the ISR to handle LEDs dimming. Follow the hints below to complete your implementation of the ISR.** ***(30 pts)***
Hint 1: Your ISR should be executed every timer overflow. Make sure you check and clear the correct interrupt flag inside the ISR.
Hint 2: Inside the ISR, increment the value of CCR2 by 1 after each PWM period to gradually increase the duty cycle. Continue this process until the duty cycle reaches 100%. At that point:
- Reset CCR2 to the minimum duty cycle value.
- Increment the **led_number** to indicate that you will proceed to the next LED.
- Reconfigure the output modes and alternate functions for the new LED that will be dimmed using TIM3_CH2.

Hint 3: When you are done dimming all the LEDs, ensure all necessary variables are reset to their initial states so the sequence can start over seamlessly.

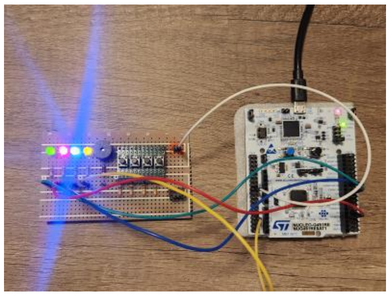
Figure 1: You can use the external LEDs board to test your program.