# ELEC335 Lab4

November 30, 2020


The objective of this lab is to give you a thorough understanding of timers. You will use C language for the problems unless some parts require *inline assembly*. Use *blinky* project from stm32g0 repo as the starting point for your problems.

## Submission

You should submit the following items organized in a folder:

- **Lab report** - Written in English. Proper cover page, intro, problems, flow charts, block diagrams, schematic diagrams, comments, and theoretical, mathematical work, simulation vs.
- **Source files** - should have proper comments and pin connections if there are any external components.
- **Video link** - A link to video demonstration of your lab (this video also should show yourself.)
- **Elf files** - generated elf file with **debug information**.

Compress the folder to a zip file, and submit that way. Each problem should be in a separate folder. Example folder structure is given below.

```
name_lastname_number_labX/
  name_lastname_report_labX.pdf
  video_link.txt
  problem1/
    problem1.s/c
    problem1.elf
    ...
  problem2/
    problem2.s/c
    problem2.elf
    ...
```

## Problems

### Problem 1

In this problem, you will work on creating an accurate delay function using *SysTick exception*. Create a SysTick exception with 1 millisecond interrupt intervals. Then create a `delay_ms(..)` function that will accurately wait for (blocking) given number of milliseconds.

- How would you measure the accuracy of your delay using software methods?
- How would you measure the accuracy of your delay using hardware methods?

Explain each case, and (if possible) implement your solution.

### Problem 2

In this problem, you will work with general purpose timers. Set up a timer with lowest priority that will be used to toggle on-board LED at 1 second intervals. Change the blinking speed using an external button. Each button press should increase the blinking speed by 1 second up to a maximum of 10 seconds. Next button press after 10 should revert it back to 1 second. All the functionality should be inside your interrupts.

## Problem 3

In this problem, connect your seven segment display and implement a count up timer. It should sit at zero, once the external button is pressed, it should count up to the max value (i.e. 9999) then once it overflows to 0, it should stop, and light up an LED (on-board or external). Pressing the button again should clear the LED and count again. For this problem, your main loop should not have anything. All the functionality should be inside your interrupts.

Note: Arrange it so that the LSD of the number increments in 1 second intervals. (i.e it should take 10 seconds to go from 0000 to 9999)

## Problem 4

In this problem, you will work with watchdog timers. Setup either *window* or *indepdentent watchdog timer* and observe its behavior in the simple blinky example from the repo. Calculate the appropriate reset time and implement it. Add the necessary handler routine for reseting the device.

## Problem 5

In this problem, you will implement your watchdog timer in Problem 3. Figure out a way to properly incorporate it to your code when it all works with timers. Explain your solution and implementation and make sure you covered all possible scenarios.