GEBZE TECHNICAL UNIVERSITY

ENGINEERING FACULTY

ELECTRONICS ENGINEERING

ELEC 335

MICROPROCESSORS

LAB 04

Name - Surname	Yağmur DERYA
Student ID	171024011

1. Introduction

In this lab, the goal is to get a thorough understanding of timers. C language is used for the problems.

2. Problems

2.1. Problem 1

In this problem, a delay_ms() function is created using SysTick Exception. First a SysTick exception which has 1 millisecond interrupt intervals is created and then the delay_ms() function which accurately wait for given number of milliseconds is created.

```
#include "stm32g0xx.h"

void SysTick_Handler(void);
void delay_ms(uint32_t);

volatile uint32_t TimeDelay; // global volatile variable

int main(void) {

       /* Enable GPIOC clock */
       RCC->IOPENR |= (1U << 2); // 100

       /* Setup PC6 as output */
       GPIOC->MODER &= ~(3U << 2*6); // clear bits ~(11) = 00
       GPIOC->MODER |= (1U << 2*6); // set PC6 as output : 01

       GPIOC->ODR |= (1U << 6); // turn on the LED on PC6

       SystemCoreClockUpdate(); // to update processor clock speed
       SysTick_Config(SystemCoreClock/1000); // 1 ms interrupt interval =
16MHz/1ms = 16 000 = SystemCoreClock/1000

    while(1){

       delay_ms(500);
       GPIOC->ODR ^= (1U << 6); // toggle the LED on PC6
    }
    return 0;
}
```

```
/* SysTick Handler */
void SysTick_Handler(void){
       if(TimeDelay !=0)
              TimeDelay --; // decrease TimeDelay until it equals to 0
}
```

```
/* function to wait for given number of milliseconds */
void delay_ms(uint32_t time){
       TimeDelay = time;
       while(TimeDelay != 0); // waits until TimeDelay = 0 by Systick_Handler
}
```

To measure the accuracy of my delay by using hardware methods, I toggled on-board LED with 500 milliseconds delay and controlled it.

To measure the accuracy of my delay by using software methods,

2.2. Problem 2

In this problem, I set up a timer with lowest priority and an external button with highest priority. When the button pressed, blinking number is increased by 1 per second.

```c
#include "stm32g0xx.h"
volatile uint32_t bcount ;

int main(void);
void TIM1_BRK_UP_TRG_COM_IRQHandler(void);
void EXTI2_3_IRQHandler(void);
void init_timer1(void);
void init_button(void);
```

```c
int main(void) {

      /* Enable GPIOB and GPIOC clock */
      RCC->IOPENR |= (3U << 1); // 110

      /* Setup PC6 as output */
      GPIOC->MODER &= ~(3U << 2*6); // clear bits ~(11) = 00
      GPIOC->MODER |= (1U << 2*6); // set PC6 as output : 01

      /* Setup PB3 as input */
      GPIOB->MODER &= ~(3U << 2*3);
      bcount = 0;

      init_button();
      init_timer1();

      while(1){
      }

      return 0;
}
```

```c
void TIM1_BRK_UP_TRG_COM_IRQHandler(void){

       GPIOC->ODR ^= (1U << 6); // toggle the LED on PC6
       TIM1->SR &= ~(1U << 0); // clear update status register
}
```

```c
void EXTI2_3_IRQHandler(void){

       bcount ++;

       if(bcount > 0 && bcount < 11) {
              TIM1->ARR = (16000 / (bcount + 1)); // to speed up the toggle
              //TIM->ARR = (16000 * (bcount + 1)); // to slow down the toggle
       } else{
              bcount = 0;
              TIM1->ARR = 16000;
       }

       EXTI->RPR1 |= (1U << 3); // clear pending
}
```

```c
void init_timer1(void){
	RCC->APBENR2 |= (1U << 11); // TIM1  clock enable

	TIM1->CR1 = 0; // control register reset
	TIM1->CR1 |= (1 << 7); // ARPE
	TIM1->CNT = 0; // zero out counter

	/* to 1 second interrupt interval */
	TIM1->PSC = 999;
	TIM1->ARR = 16000;

	TIM1->DIER |= (1 << 0); // update interrupt enable
	TIM1->CR1 |= (1 << 0); // TIM1 enable

	NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn, 0xC0); // lowest priority
	NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
}
```

```
void init_button(void){

	EXTI->RTSR1 |= (1U <<3 );
	EXTI->EXTICR[0] |= (1U << 8*3);
	EXTI->IMR1 |= (1U << 3);

	NVIC_SetPriority(EXTI2_3_IRQn, 0); // highest priority
	NVIC_EnableIRQ(EXTI2_3_IRQn);
}
```

2.3. Problem 3

In this problem, it is asked to connect a SSD and implement a count up timer.

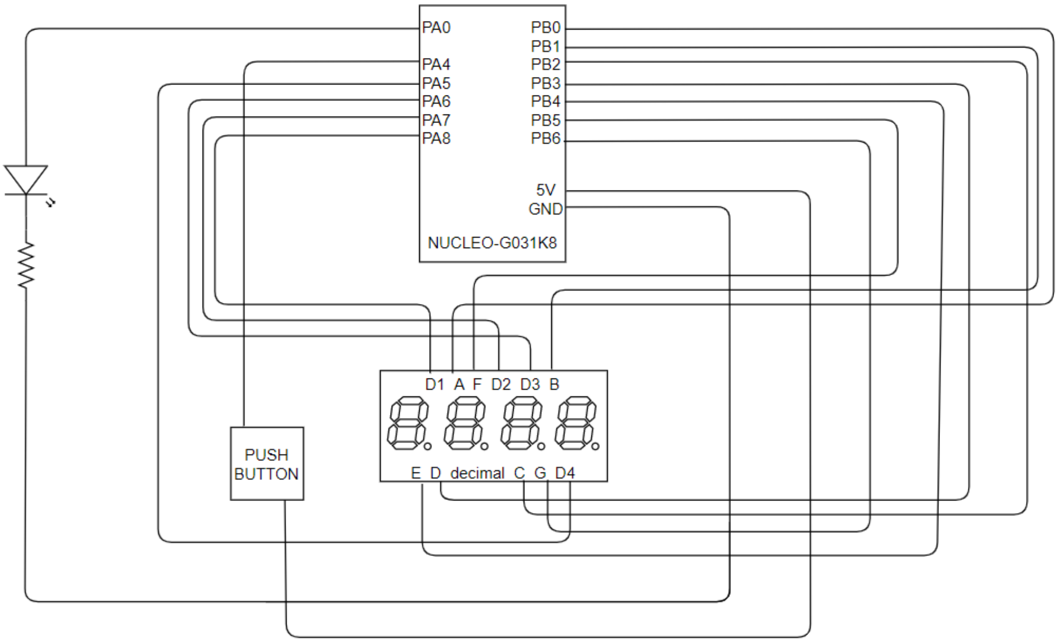

Figure 1. block diagram of the set-up.

2.4. Problem 4

In this problem, I setup independent watchdog timer to observe its behaviour but could not toggle the LED.

```
/*
 *
 * Yagmur Derya
 *
 *
 */

#include "stm32g0xx.h"

#define LEDDELAY    1600000

void delay(volatile uint32_t);

void init_IWDG(void){
	IWDG->KR = 0xCCCC; // enable IWDG
	IWDG->KR = 0x5555; // enable configuring peripheral
	IWDG->PR = 0x06; // 256
	IWDG->RLR = 0xFFFF;

}

int main(void) {

    /* Enable GPIOC clock */
    RCC->IOPENR |= (1U << 2);

    /* Setup PC6 as output */
    GPIOC->MODER &= ~(3U << 2*6);
    GPIOC->MODER |= (1U << 2*6);

    init_IWDG();

    while(1) {
        /* Toggle LED */
        GPIOC->ODR ^= (1U << 6);
        IWDG->KR = 0xAAAA;
    }

    return 0;
}

void delay(volatile uint32_t s) {
    for(; s>0; s--);
}
```

2.5.Problem 5

3. References

[1] RM0444 Reference manuel

[2] https://github.com/fcayci/stm32g0

[3] https://electronics-homemade.com/STM32F4-LED-Toggle-Systick.html

[4] https://www.mcu-turkey.com/stm8s-iwdgindependent-watchdog-modulu-kullanimi/