GEBZE TECHNICAL UNIVERSITY

ELECTRONICS ENGINEERING

ELM335

Microprocessors Laboratory

LAB 4 Experiment Report

Prepared by
151024008 - Abdürrahim Deniz KUMBARACI
171024050 - Abdül Samet Karapınar
171024008 - Yasin Özbek

Problem 1

Code

nucleo.h

```
/*
 * nucleo.h
 *
 *  Created on: Nov 29, 2021
 */

#ifndef NUCLEO_H_
#define NUCLEO_H_

// On-Board LED //

void nucleo_led_init();
void nucleo_led_set();
void nucleo_led_clear();
void nucleo_led_toggle();
void nucleo_ext_led_init();
void nucleo_ext_led_set();
void nucleo_ext_led_clear();
void nucleo_ext_led_toggle();

// Button Functions//

void nucleo_button_init();
int nucleo_button_read();


// Timer interrupts

void timer1_init();
void timer2_init();
void timer2_s();
void systic_init();
void systick_delay_ms(long unsigned int a);
void systick_delay_s(long unsigned int a);
void timer1_interrupt();
void timer1_statclear();
void timer1_s();
#endif /* NUCLEO_H_ */
```

nucleo.c

```
/*
 * nucleo.c
 *
 *  Created on: Nov 29, 2021
 */

#include "nucleo.h"
#include "stm32g0xx.h"

#define KILO    1000
#define MEGA    1000000
```

```c
void nucleo_led_init(void){
	RCC -> IOPENR |= (1U <<2 );
	GPIOC -> MODER &= ~ (3U << 2*6);
	GPIOC -> MODER |= (1U << 2*6);
	GPIOC -> BRR |= (1U << 6);

}

void nucleo_led_set(void){
	GPIOC -> ODR |= (1U << 6) ;
}

void nucleo_led_clear(void){
	GPIOC -> BRR |= (1U << 6);
}

void nucleo_led_toggle(void){
	GPIOC -> ODR ^= (1U << 6) ;
}

void nucleo_button_init(void){
	RCC -> IOPENR |= (1U << 5 );
	GPIOF -> MODER &= ~ (3U << 2*2);

}

int nucleo_button_read() {

	int a = ((GPIOF -> IDR >> 2 ) & 0x01);
		if (a) return 0;
		else return 1;

}

void nucleo_ext_led_init(){

	RCC -> IOPENR |= (1U <<0 );
	GPIOB -> MODER &= ~ (3U << 2*4);
	GPIOB -> MODER |= (1U << 2*4);
	GPIOB -> BRR |= (1U << 4);

}
void nucleo_ext_led_set(){
	GPIOB -> ODR |= (1U <<4 );
}
void nucleo_ext_led_clear(){
	GPIOB -> BRR |= (1U <<4 );
}
void nucleo_ext_led_toggle(){
	GPIOB -> ODR ^= (1U <<4 );
}
void timer1_init() {
	SystemCoreClockUpdate();
	RCC -> APBENR2 |= (1U << 11 );
	TIM1 -> CR1 = 0;
	TIM1 -> CR1 |= (1 << 7 );
	TIM1 -> CNT = 0;
```

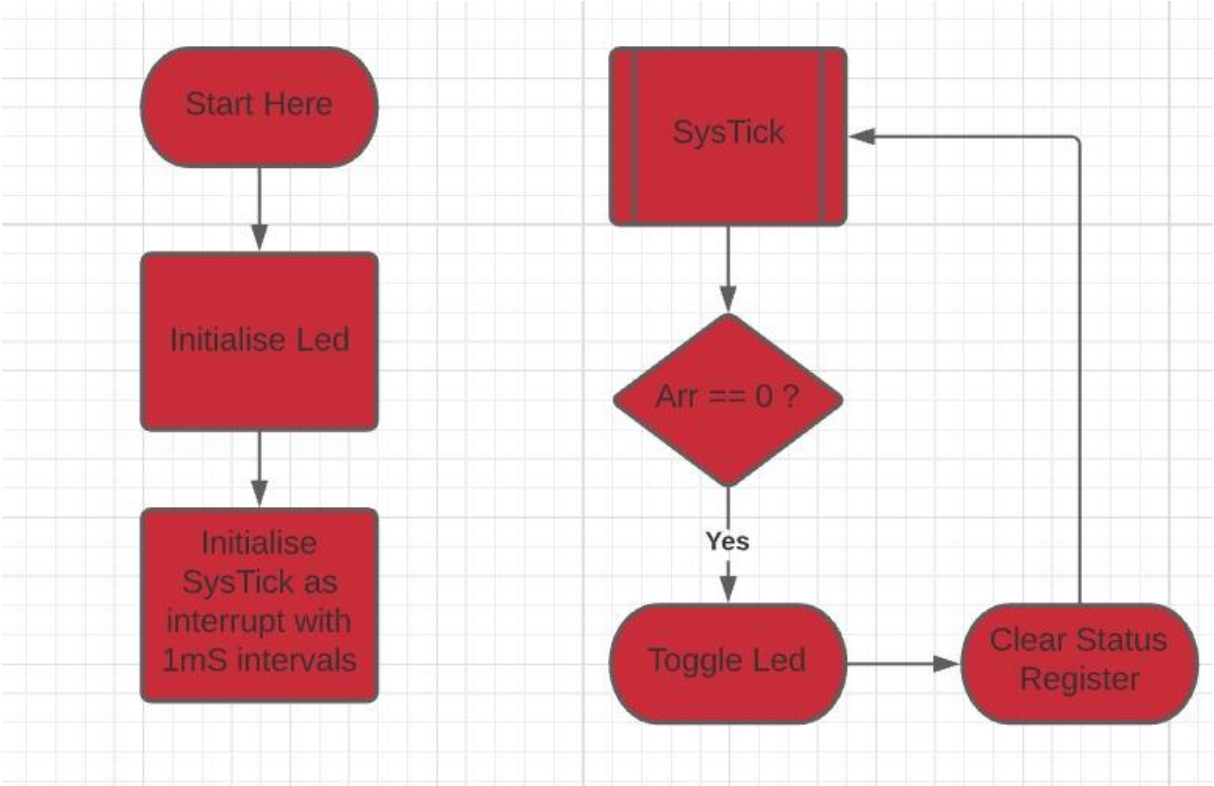
```
	TIM1 -> DIER |= (1 << 0 );
	TIM1 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn , 0);
	NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
}
void timer1_s(){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16000;
}
void timer2_init() {
	SystemCoreClockUpdate();
	RCC -> APBENR1 |= (1U << 0 );
	TIM2 -> CR1 = 0;
	TIM2 -> CR1 |= (1 << 7 );
	TIM2 -> CNT = 0;
	TIM2 -> DIER |= (1 << 0 );
	TIM2 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM2_IRQn  , 0);
	NVIC_EnableIRQ(TIM2_IRQn );
}
void timer2_s(){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 16000;
}


void systic_init(){
	SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;
	SysTick->VAL=0;
	SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;
	NVIC_EnableIRQ(SysTick_IRQn);
	NVIC_SetPriority (SysTick_IRQn,0);
}


void timer1_statclear(){
	TIM1 -> SR &= ~ (1U << 0);
}

void systick_delay_ms(long unsigned int a) {
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / KILO) * a);
}
void systick_delay_s(long unsigned int a){
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / MEGA) * a);
}
```

Comments

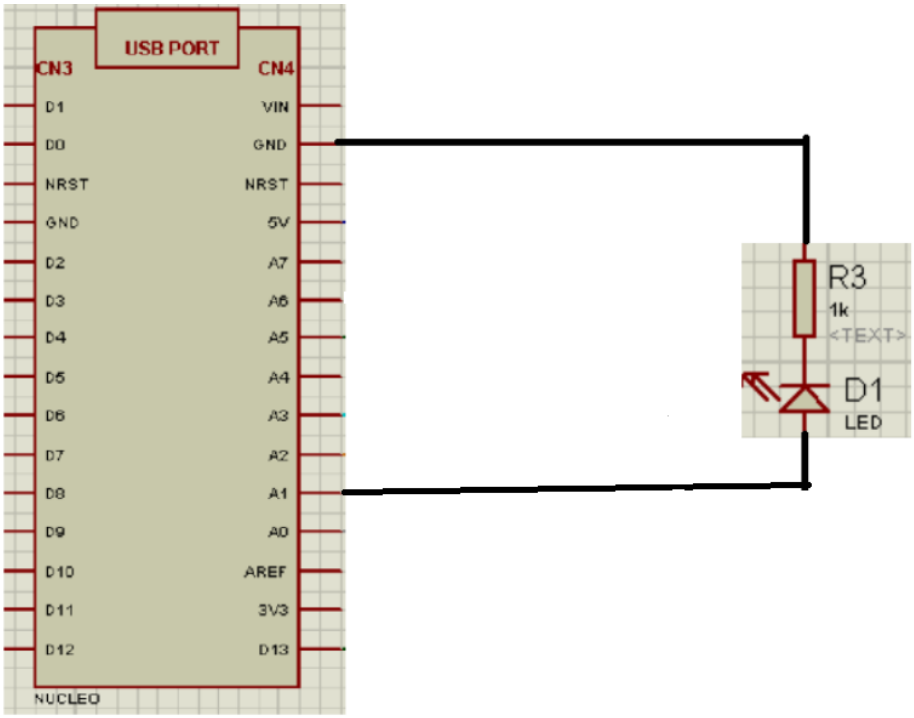
By creating a board support package, we started to make a scalable libary that will make our codes much more easier to implement in future. Also tasks with big codes will have sagnificantly lower execution time. Also this kind of approach makes it easier to troubleshoot the code in the future

Problem 2

Flow Chart



Block Diagram



Code

nucleo.h

```
/*
 * nucleo.h
 *
 *  Created on: Nov 29, 2021
 */

#ifndef NUCLEO_H_
#define NUCLEO_H_

// On-Board LED //

void nucleo_led_init();
void nucleo_led_set();
void nucleo_led_clear();
void nucleo_led_toggle();
void nucleo_ext_led_init();
void nucleo_ext_led_set();
void nucleo_ext_led_clear();
void nucleo_ext_led_toggle();

// Button Functions//

void nucleo_button_init();
int nucleo_button_read();


// Timer interrupts

void timer1_init();
void timer2_init();
void timer2_s();
void systic_init();
void systick_delay_ms(long unsigned int a);
void systick_delay_s(long unsigned int a);
void timer1_interrupt();
void timer1_statclear();
void timer1_s();
#endif /* NUCLEO_H_ */
```

nucleo.c

```
/*
 * nucleo.c
 *
 *  Created on: Nov 29, 2021
 */

#include "nucleo.h"
#include "stm32g0xx.h"

#define KILO    1000
#define MEGA    1000000


void nucleo_led_init(void){
```

```
	RCC -> IOPENR |= (1U <<2 );
	GPIOC -> MODER &= ~ (3U << 2*6);
	GPIOC -> MODER |= (1U << 2*6);
	GPIOC -> BRR |= (1U << 6);

}

void nucleo_led_set(void){
	GPIOC -> ODR |= (1U << 6) ;
}

void nucleo_led_clear(void){
	GPIOC -> BRR |= (1U << 6);
}

void nucleo_led_toggle(void){
	GPIOC -> ODR ^= (1U << 6) ;
}

void nucleo_button_init(void){
	RCC -> IOPENR |= (1U << 5 );
	GPIOF -> MODER &= ~ (3U << 2*2);

}

int nucleo_button_read() {

	int a = ((GPIOF -> IDR >> 2 ) & 0x01);
		if (a) return 0;
		else return 1;

}

void nucleo_ext_led_init(){

	RCC -> IOPENR |= (1U <<0 );
	GPIOB -> MODER &= ~ (3U << 2*4);
	GPIOB -> MODER |= (1U << 2*4);
	GPIOB -> BRR |= (1U << 4);

}
void nucleo_ext_led_set(){
	GPIOB -> ODR |= (1U <<4 );
}
void nucleo_ext_led_clear(){
	GPIOB -> BRR |= (1U <<4 );
}
void nucleo_ext_led_toggle(){
	GPIOB -> ODR ^= (1U <<4 );
}
void timer1_init() {
	SystemCoreClockUpdate();
	RCC -> APBENR2 |= (1U << 11 );
	TIM1 -> CR1 = 0;
	TIM1 -> CR1 |= (1 << 7 );
	TIM1 -> CNT = 0;
	TIM1 -> DIER |= (1 << 0 );
	TIM1 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn , 0);
```

```
	NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
}
void timer1_s(){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16000;
}
void timer2_init() {
	SystemCoreClockUpdate();
	RCC -> APBENR1 |= (1U << 0 );
	TIM2 -> CR1 = 0;
	TIM2 -> CR1 |= (1 << 7 );
	TIM2 -> CNT = 0;
	TIM2 -> DIER |= (1 << 0 );
	TIM2 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM2_IRQn  , 0);
	NVIC_EnableIRQ(TIM2_IRQn );
}
void timer2_s(){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 16000;
}


void systic_init(){
	SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;
	SysTick->VAL=0;
	SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;
	NVIC_EnableIRQ(SysTick_IRQn);
	NVIC_SetPriority (SysTick_IRQn,0);
}


void timer1_statclear(){
	TIM1 -> SR &= ~ (1U << 0);
}

void systick_delay_ms(long unsigned int a) {
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / KILO) * a);
}
void systick_delay_s(long unsigned int a){
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / MEGA) * a);
}
```

main.c

```
#include "stm32g0xx.h"
#include "nucleo.h"
 long unsigned int a= 1;
void SysTick_IRQHandler(void){
	void nucleo_led_toggle();


}

 int main(void){

	 void nucleo_led_init();
	void systic_init();
```

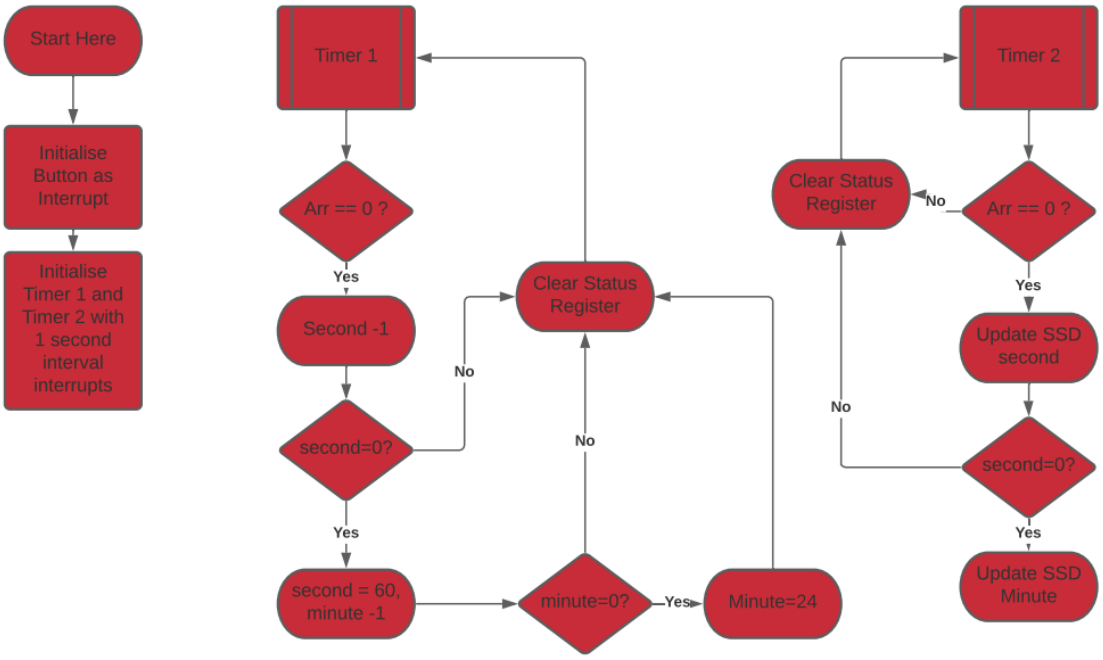
```
        void systick_delay_ms();

        for(;;);

        return 0;
}
```

Comments

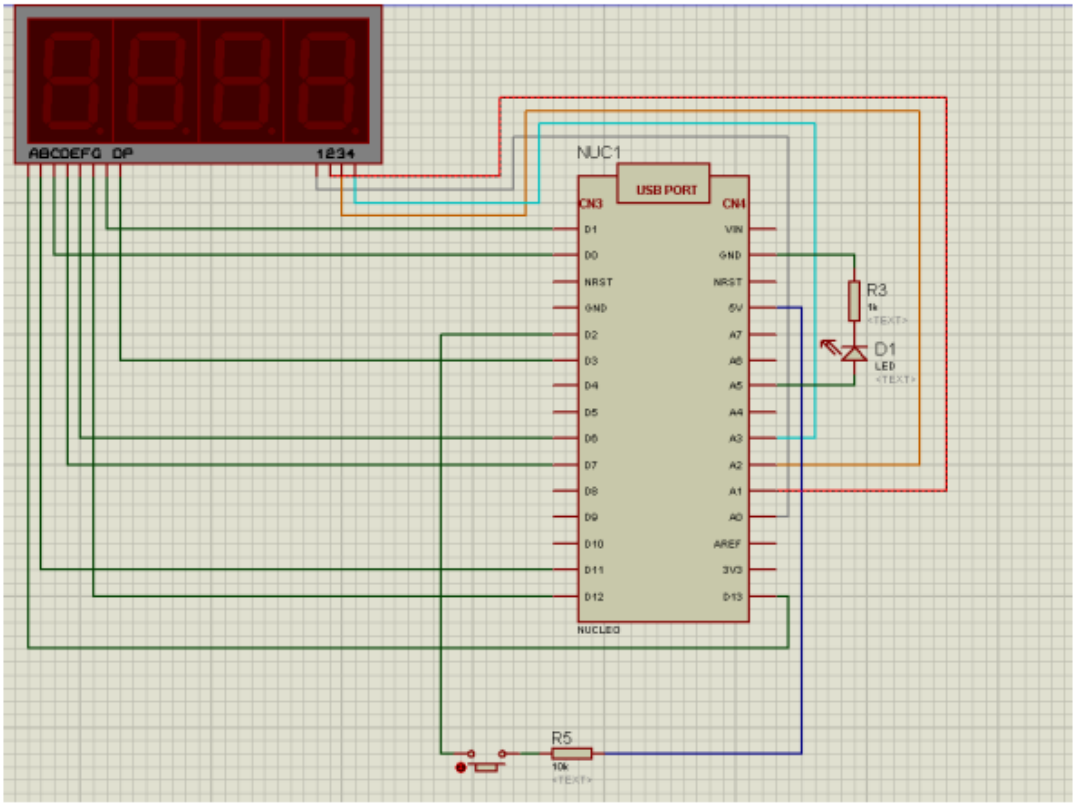
By using header files,the coding process was much cleaner and easier to implement.Also while we were running user generated delay functions, we were roughly estimating the delay times but using internal timers are much more accurate

Problem 3

Flow Chart



Block Diagram



We try to solve problem 3 with 2 different code

Code 1

```
#include "stm32g0xx.h"
#include "nucleo.h"
 long unsigned int a= 1;
int minutes = 24;
int seconds = 60;
 enum TimeStates{
       State0,
       State1,
       State2,
       State3,
       State4,
       State5

};
enum TimeStates Scale = State0 ;

 void EXTI0_1_IRQHandler(void){
         Scale = Scale + 1;

        void nucleo_PA0_button_statclear();
}

 void TIM1_BRK_UP_TRG_COM_IRQHandler(void){
        // Fill Here
        seconds = seconds-1;
        if (seconds < 1){
```

```
                minutes = minutes -1;
                        seconds = 60;
                        if(minutes == 0){
                                if(seconds == 0){
                                        minutes = 24;
                                        seconds = 60;
                                }
                        }
        }
        void timer1_statclear();

}

void TIM2_IRQHandler(void){
        //Fill Here
        void timer2_statclear();
}

int main(void){
        void nucleo_PA0_button_init();
        void nucleo_PA0_button_INT();
        void nucleo_led_init();
        while(1){
                if (Scale == State0){
                        //do mode 0

                }
                if (Scale == State1){
                        //do mode 1
                        void timer1_s();
                        void timer2_s();
                        void timer1_init();
                        void timer2_init();
                }
                if (Scale == State2){
                        //do mode 2
                        void timer1_s2();
                        void timer2_s2();
                        void timer1_init();
                        void timer2_init();
                }
                if (Scale == State3){
                        //do mode 3
                        void timer1_s3();
                        void timer2_s3();
                        void timer1_init();
                        void timer2_init();


                }
                if (Scale == State4){
                        //do mode 4
                        void timer1_s4();
                        void timer2_s4();
                        void timer1_init();
                        void timer2_init();

                }
                if (Scale == State5){
```

```
                        //do mode 5
                        void timer1_s5();
                        void timer2_s5();
                        void timer1_init();
                        void timer2_init();

                }
                if (Scale > State5){
                        Scale = State0;
                }

        return 0;
}
 }
```

Code 2

```
#include "stm32g0xx.h"

volatile  uint16_t
numbers[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
volatile  uint16_t digits[4]={0x01,0x02,0x10,0x20};

volatile uint32_t msTicks = 0;                                  /* Variable to store
millisecond ticks */
volatile uint32_t pressed = 0;                                 //Used as a trigger to
start counting
volatile uint32_t loops[4] = {2,3,5,9};                        //Used to store the
current cycle of each digit

void clean(){       //clear the shown number and open digit to prevent shadows
/silhouettes .
        GPIOB->ODR &= ~(255U<<0);
        GPIOA->ODR &= ~(51U<<0);
}
void reset_loops(){       //Clears the loops , effectively making them to show all
0
        loops[3] = 9;
        loops[2] = 5;
        loops[1] = 3;
        loops[0] = 2;
        GPIOC->ODR ^= (1U << 6);         //Toggles led. //Turning on
}
void disp_num(int digit){
        clean();     //Turns off the activated digit and old number value to prevent
shadows appearing between digit switches
         GPIOA->ODR |=  digits[digit];  //Turn on the specific digit by which the
function is called
         GPIOB->ODR |=  numbers[loops[digit]]; //loops[digit]  chooses the current
cycle for the specific digit

        //numbers[loops[digit]] chooses the correct hex value to show at the given
digit
}



void display(){           //Display the current values at each digit. Digits are
arrayed in mixed order to prevent one side being too bright
```

```
                                //lights each digit 40 times to create a non
flickering display
      volatile  uint32_t i=40;
for(;i>0;--i){
      disp_num(1);
      disp_num(3);
      disp_num(2);
      disp_num(0);
      }

}

void adjust_loops(){
      loops[3]--;
      if(   loops[3] <= 0      ){
            loops[3] = 9;
            loops[2]--;
            if(   loops[2] <= 0      ){
                  loops[2] = 5;
                  loops[1]--;
                  if(   loops[1] <= 0      ){
                        loops[1] = 9;
                        loops[0]--;
                        if(   loops[0] <= 0      ){
                              loops[0] = 0;
                              reset_loops();
                              pressed = 0;
                                                    }
                                                }
                                          }
                                    }

}

void TIM3_IRQHandler(){
      TIM3->ARR =160; //ADJUSTIG Auto reload to change the delay time with each
press. ARR*pressed changes the time interval between interrupts .
      TIM3->SR &=  ~(1U << 0); //Reset
      display();          //display current values
      if(pressed   ==    1){   //if button is pressed or if it is still counting
call adjust_loops
            adjust_loops();

      }
//If not pressed or not counting do nothing
}

void init_timer3(){ //Set to create exception each 0.0001 second
      RCC->APBENR1 |= (1U << 1);      //Enabling TIM3
      TIM3->CR1 = 0;                  //RESET TIM3_CR1 register
      TIM3->CR1 |= (1 << 7); //AUTO RELOAD ENABLED
      TIM3->DIER |=(1 << 0); //UPDATE INTERRUPT ENABLED
      TIM3->CNT = 0 ;           //RESET COUNTER

      TIM3->PSC = 9;            //PRESCALER SET to 9
      TIM3->ARR =160;           //AUTORELOAD VALUE
(PSC+1*ARR)/SystemCoreClock=0.0001
      TIM3->CR1 |= (1 << 0);    //Counter enabled
```

```
      NVIC_SetPriority(TIM3_IRQn , 4); //Set to the lowest priority level
      NVIC_EnableIRQ(TIM3_IRQn);             //Enable interrupt

}

void init_pa15(){ //Button interrupt

    EXTI->EXTICR[3] |= (0U << 8*3);    //Third mux is selected for Pin 12-15 , 0U
is used to select PA 8*3 is used to select pin 15
    EXTI->RTSR1 |= (1U << 15);         //Rising edge triggered
    EXTI->IMR1  |= (1U << 15);         //Mask 1

    NVIC_SetPriority(EXTI4_15_IRQn , 0);     //Set to the highest priority level
    NVIC_EnableIRQ(EXTI4_15_IRQn);                 //Enable interrupt
}

void init_initialize(){
            SysTick_Config(SystemCoreClock / 10000);
        /* Enable GPIOC and GPIOA clock */
          RCC->IOPENR |= (0x7);
          /* Setup PC6 as output */
          GPIOC->MODER &= ~(3U << 2*6);
          GPIOC->MODER |= (1U << 2*6);
          /*Setup PA15 as pulldown input */
          GPIOA->MODER &= ~(3U << 2*15);
          GPIOA->PUPDR |= (2U << 2*15);
          /* Setup PA pins as output */
          GPIOA->MODER &= ~(12586767U);
          GPIOA->MODER |= (0x400505);
          /* Setup PB0-7 pins as output */
          GPIOB->MODER &= ~(65535U);
          GPIOB->MODER |= (0x5555);
          GPIOC->ODR ^= (1U << 6);    //Open led
}



int main(void) { //Empty main function it only calls init_xxx functions then rest
at infinite loop

init_initialize();
init_timer3();
init_pa15();
while(1){}

}



void SysTick_Handler(void)  {                              /* SysTick interrupt
Handler. */
  msTicks++;/* See startup file startup_LPC17xx.s for SysTick vector */
}

void EXTI4_15_IRQHandler(void){
        pressed = 1;                  //Set button pressed value to 1. This
variable will trigger adjust_loops function
        GPIOC->ODR ^= (1U << 6);      //Toggle onboard led
```

```
        EXTI->RPR1 |= (1U << 15);      //Set hardware raised flag to zero by
software
}
```

Comments

Compared to our last lab, implementation was much more straight forward and more like a upper levell(further from computer language) coding experience. Hence the scalability is much higher for this kind of application.Also using of timers gave us much more presice timing operations.