GEBZE TECHNICAL UNIVERSITY

ELECTRONICS ENGINEERING

ELM335

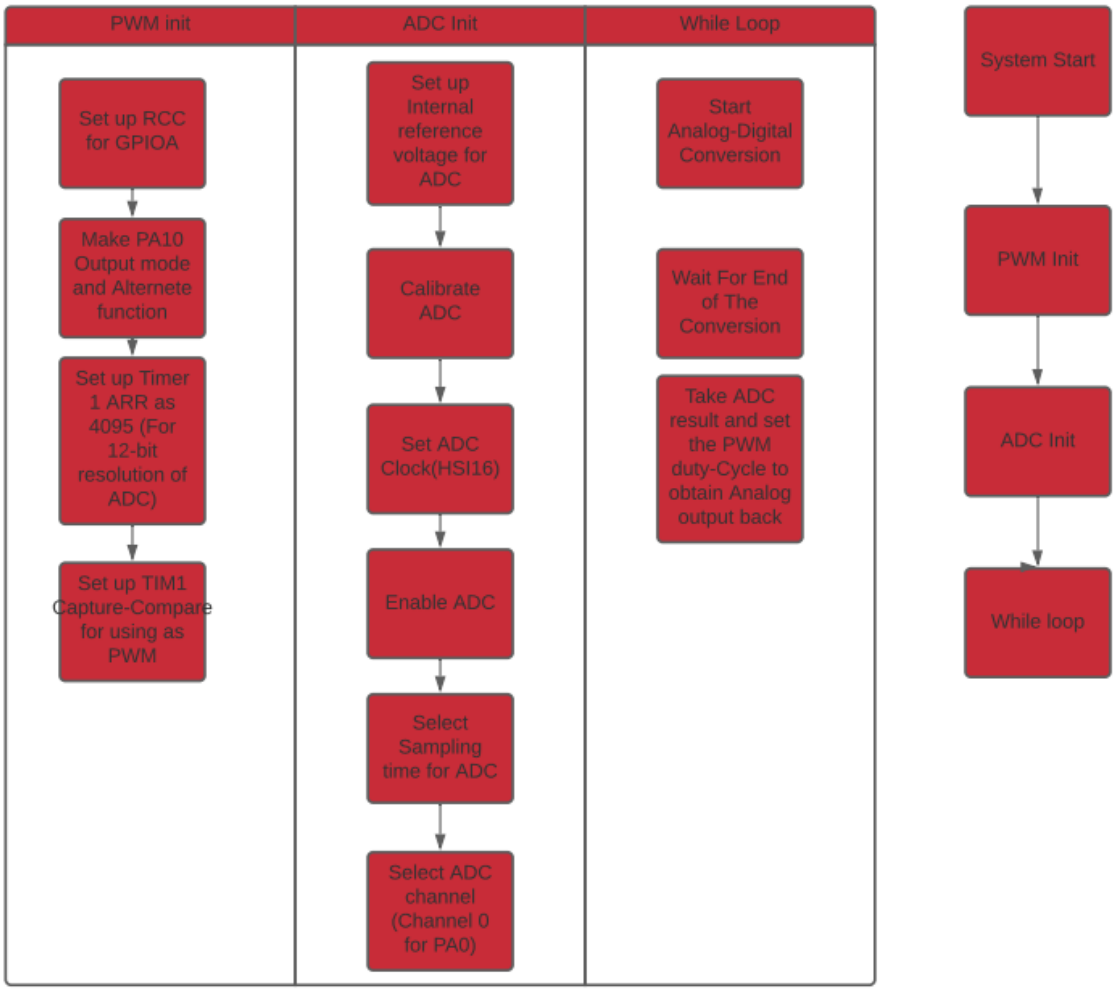
Microprocessors Laboratory

LAB 6 Experiment Report

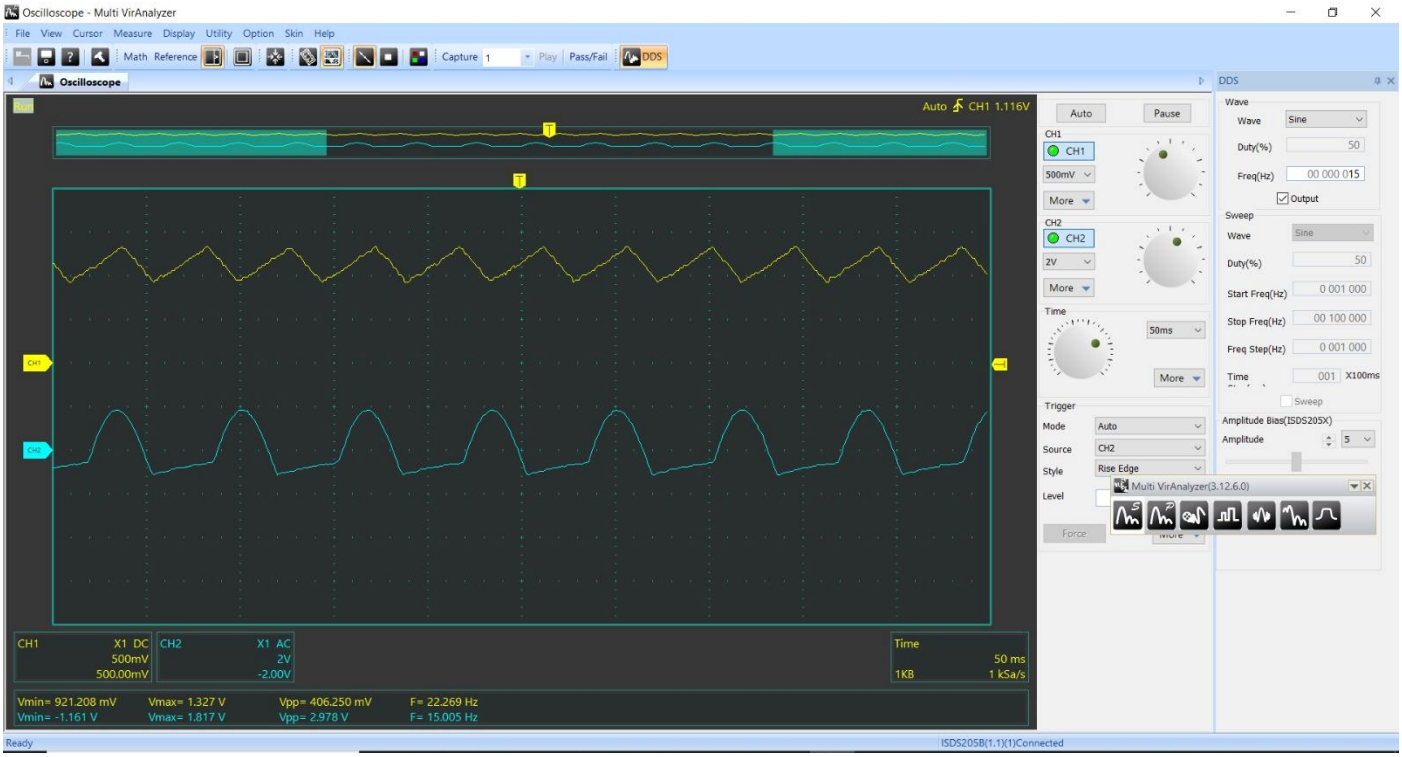
Prepared by
151024008 - Abdürrahim Deniz KUMBARACI
171024050 - Abdül Samet Karapınar
171024008 - Yasin Özbek

Problem 1

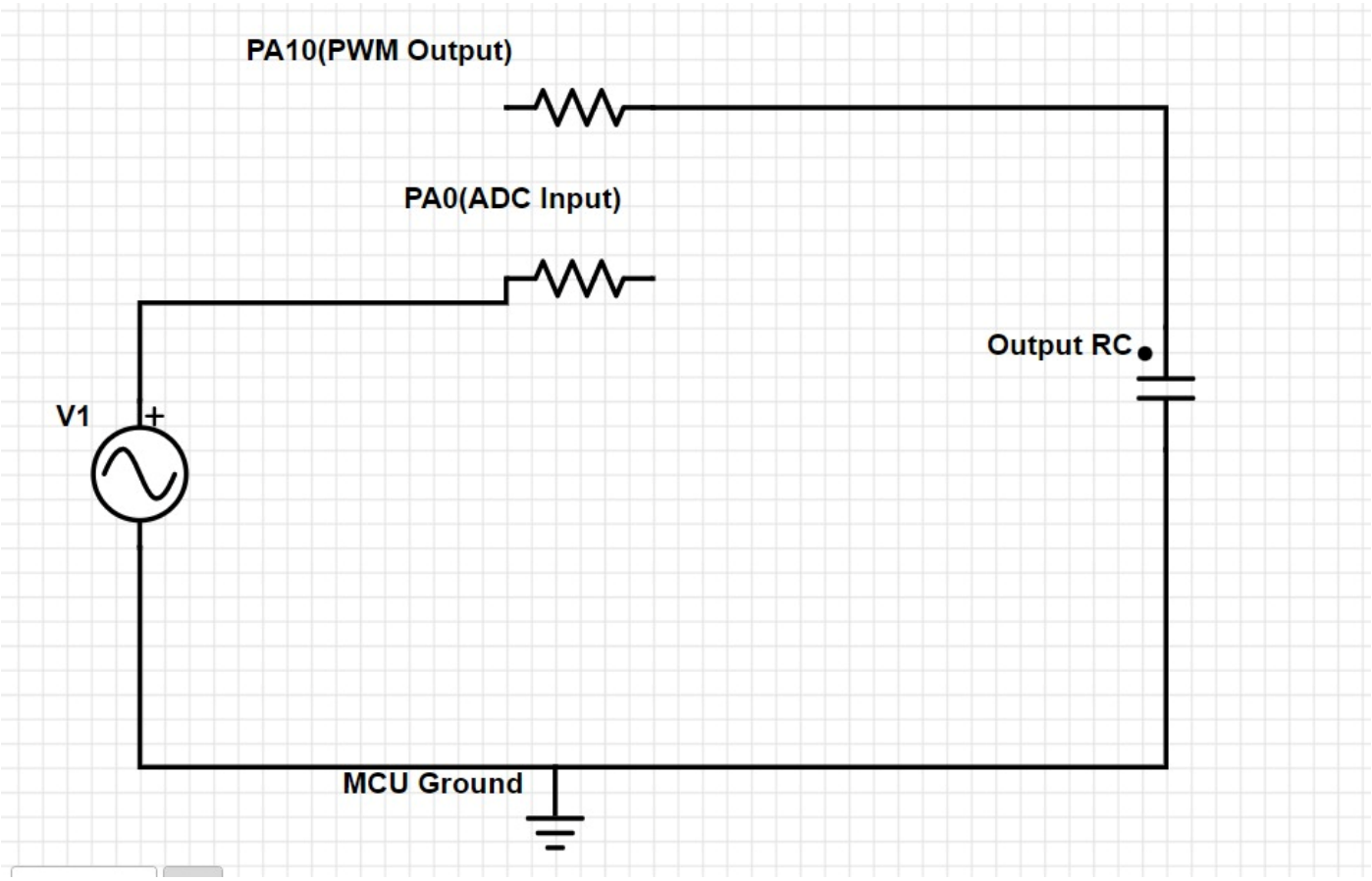
Flow Chart



Oscilloscope Image



Block Diagram



Code

main.c

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

void TIM1_BRK_UP_TRG_COM_IRQHandler(void){

		TIM1 -> SR &= ~ (1U << 0);
	}
int main(void) {
	RCC->APBENR2 |= RCC_APBENR2_TIM1EN;
	RCC->IOPENR |= RCC_IOPENR_GPIOAEN;
	GPIOA->MODER &= ~ GPIO_MODER_MODE10_0;
	GPIOA->MODER |= GPIO_MODER_MODE10_1;

	GPIOA->AFR[1] = (2U<<8);
	timer1_init();
	TIM1 ->PSC =3;
	TIM1->ARR = 4095;
	TIM1->CCR3 = 2000;
	TIM1->CCMR2 |= TIM_CCMR2_OC3M_1 | TIM_CCMR2_OC3M_2 |TIM_CCMR2_OC3PE;
	TIM1->CCER |= TIM_CCER_CC3E;
	TIM1->BDTR |= TIM_BDTR_MOE;
	TIM1->CR1 |= TIM_CR1_CEN;
	TIM1 -> EGR |= TIM_EGR_UG;

	ADC1->CR |= (1U << 28);                          /* ENABLE ADC VOLTAGE REGULATOR */

	ADC1->CR &= ~ ADC_CR_ADEN;
	ADC1->CR |= ADC_CR_ADCAL;
	while((ADC1->CR & ADC_CR_ADCAL )!= 0){

	}
	RCC->APBENR2 |= RCC_APBENR2_ADCEN;
```

```
        RCC->CR |= RCC_CR_HSION;
        while((RCC->CR & RCC_CR_HSIRDY)==0){

        }
        ADC1->CFGR2 |= ADC_CFGR2_CKMODE;

        ADC1->CR |= ADC_CR_ADEN;
        while((ADC1->ISR & ADC_ISR_ADRDY)==0){

        }
        ADC1->SMPR |= ADC_SMPR_SMP1_0 |ADC_SMPR_SMP1_1 |ADC_SMPR_SMP1_2;
        ADC1->CHSELR |= ADC_CHSELR_CHSEL1;




    while(1) {
        ADC1->CR |= ADC_CR_ADSTART;
        while((ADC1->ISR & ADC_ISR_EOC )==0){

        }
        TIM1->CCR3 = ADC1->DR;


    }

    return 0;
}
```

nucleo.h

```
#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();
void nucleo_PA0_button_init();
int nucleo_PA0_button_read();
void nucleo_PA0_button_INT();
void nucleo_PA0_button_statclear();
//
// Timer interrupts

void timer1_init();
void timer2_init();
void timer2_s();
void systic_init();
void systick_delay_ms();
void systick_delay_s();
void timer1_interrupt();
void timer1_statclear();
void timer2_statclear(void);
void timer1_s();
void timer1_s2();
void timer1_s3();
void timer1_s4();
void timer1_s5();
```

```
// Project functions

// GPIO Functions
// Input Initialise
void Init_PA0_Input();
void Init_PA1_Input();
void Init_PA2_Input();
void Init_PA3_Input();
void Init_PA4_Input();
void Init_PA5_Input();
void Init_PA6_Input();
void Init_PA7_Input();
void Init_PA8_Input();
void Init_PA9_Input();
void Init_PA10_Input();
void Init_PA11_Input();
void Init_PA12_Input();
void Init_PA13_Input();
void Init_PA14_Input();
void Init_PA15_Input();
void Init_PB0_Input();
void Init_PB1_Input();
void Init_PB2_Input();
void Init_PB3_Input();
void Init_PB4_Input();
void Init_PB5_Input();
void Init_PB6_Input();
void Init_PB7_Input();
void Init_PB8_Input();
void Init_PB9_Input();
void Init_PB10_Input();
void Init_PB11_Input();
void Init_PB12_Input();
void Init_PB13_Input();
void Init_PB14_Input();
void Init_PB15_Input();
// Output Initialise
void Init_PB0_Output();
void Init_PB1_Output();
void Init_PB2_Output();
void Init_PB3_Output();
void Init_PB4_Output();
void Init_PB5_Output();
void Init_PB6_Output();
void Init_PB7_Output();
void Init_PB8_Output();
void Init_PB9_Output();
void Init_PB10_Output();
void Init_PB11_Output();
void Init_PB12_Output();
void Init_PB13_Output();
void Init_PB14_Output();
void Init_PB15_Output();
void Init_PA0_Output();
void Init_PA1_Output();
void Init_PA2_Output();
void Init_PA3_Output();
void Init_PA4_Output();
void Init_PA5_Output();
void Init_PA6_Output();
void Init_PA7_Output();
void Init_PA8_Output();
void Init_PA9_Output();
void Init_PA10_Output();
void Init_PA11_Output();
void Init_PA12_Output();
void Init_PA13_Output();
void Init_PA14_Output();
void Init_PA15_Output();
//
```

```
// Set Output
void Set_PB0();
void Set_PB1();
void Set_PB2();
void Set_PB3();
void Set_PB4();
void Set_PB5();
void Set_PB6();
void Set_PB7();
void Set_PB8();
void Set_PB9();
void Set_PB10();
void Set_PB11();
void Set_PB12();
void Set_PB13();
void Set_PB14();
void Set_PB15();
void Set_PA0();
void Set_PA1();
void Set_PA2();
void Set_PA3();
void Set_PA4();
void Set_PA5();
void Set_PA6();
void Set_PA7();
void Set_PA8();
void Set_PA9();
void Set_PA10();
void Set_PA11();
void Set_PA12();
void Set_PA13();
void Set_PA14();
void Set_PA15();

void Clear_PA0();
void Clear_PA1();
void Clear_PA2();
void Clear_PA3();
void Clear_PA4();
void Clear_PA5();
void Clear_PA6();
void Clear_PA7();
void Clear_PA8();
void Clear_PA9();
void Clear_PA10();
void Clear_PA11();
void Clear_PA12();
void Clear_PA13();
void Clear_PA14();
void Clear_PA15();
void Clear_PB4();
void Clear_PB0();
void Clear_PB1();
void Clear_PB2();
void Clear_PB3();
void Clear_PB4();
void Clear_PB5();
void Clear_PB6();
void Clear_PB7();
void Clear_PB8();
void Clear_PB9();
void Clear_PB10();
void Clear_PB11();
void Clear_PB12();
void Clear_PB13();
void Clear_PB14();
void Clear_PB15();

void Toggle_PB0();
void Toggle_PB1();
void Toggle_PB2();
void Toggle_PB3();
void Toggle_PB4();
```

```
void Toggle_PB5();
void Toggle_PB6();
void Toggle_PB7();
void Toggle_PB8();
void Toggle_PB9();
void Toggle_PB10();
void Toggle_PB11();
void Toggle_PB12();
void Toggle_PB13();
void Toggle_PB14();
void Toggle_PB15();
void Toggle_PA0();
void Toggle_PA1();
void Toggle_PA2();
void Toggle_PA3();
void Toggle_PA4();
void Toggle_PA5();
void Toggle_PA6();
void Toggle_PA7();
void Toggle_PA8();
void Toggle_PA9();
void Toggle_PA10();
void Toggle_PA11();
void Toggle_PA12();
void Toggle_PA13();
void Toggle_PA14();
void Toggle_PA15();
#endif /* NUCLEO_H_ */
```

nucleo.c

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

#define KILO    1000
#define MEGA    1000000

void nucleo_PA0_button_init(){
       RCC -> IOPENR |= (1U <<0 );
       GPIOA -> MODER &= ~ (3U << 0);
       GPIOA -> PUPDR &= ~ (3U << 0);
       GPIOA -> PUPDR |=  (2U << 0);
}

int nucleo_PA0_button_read(void) {

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

}

void nucleo_PA0_button_INT(){
       EXTI -> RTSR1 |= (1U <<0 );
       EXTI -> EXTICR[0] |= (0U <<0 );

       EXTI -> IMR1 |= (1U <<0 );
       NVIC_SetPriority(EXTI0_1_IRQn , 0);
       NVIC_EnableIRQ(EXTI0_1_IRQn);
}

void nucleo_PA0_button_statclear(){
       EXTI -> RPR1 &= ~ (1U << 0);
}

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(void) {

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

}

void nucleo_ext_led_init(void){

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

}
void nucleo_ext_led_set(void){
	GPIOB -> ODR |= (1U <<4 );
}
void nucleo_ext_led_clear(void){
	GPIOB -> BRR |= (1U <<4 );
}
void nucleo_ext_led_toggle(void){
	GPIOB -> ODR ^= (1U <<4 );
}
void timer1_init(void) {

	RCC -> APBENR2 |= (1U << 11 );
	TIM1 -> CR1 = 0;
	TIM1 -> CR1 |= (1 << 7 );
	TIM1 -> CNT = 0;
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16000;
	TIM1 -> DIER |= (1 << 0 );
	TIM1 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn , 1);
	NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
}
void timer1_s(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16000;
}
void timer2_s(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 16000;
}

void timer1_s2(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 8000;
}
void timer2_s2(void){
	TIM2 -> PSC = 999;
```

```
	TIM2 -> ARR = 8000;
}

void timer1_s3(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 1600;
}
void timer2_s3(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 1600;
}

void timer1_s4(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 160;
}
void timer2_s4(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 160;
}

void timer1_s5(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16;
}
void timer2_s5(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 16;
}

void timer2_init(void) {
	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 systic_init(void){
	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(void){
	TIM1 -> SR &= ~ (1U << 0);
}
void timer2_statclear(void){
	TIM2 -> SR &= ~ (1U << 0);
}

void systick_delay_ms() {
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / KILO));
}
void systick_delay_s(){
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / MEGA));
}

// Project functions

// GPIO Functions
// Input Init
void Init_PA0_Input(){
```

```
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*0);

}
void Init_PA1_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*1);

}
void Init_PA2_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*2);

}
void Init_PA3_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*3);

}
void Init_PA4_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*4);

}
void Init_PA5_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*5);

}
void Init_PA6_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*6);

}
void Init_PA7_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*7);

}
void Init_PA8_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*8);

}
void Init_PA9_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*9);

}
void Init_PA10_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*10);

}
void Init_PA11_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*11);

}
void Init_PA12_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*12);

}
void Init_PA13_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*13);

}
void Init_PA14_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*14);
```

```
}
void Init_PA15_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*15);

}
void Init_PB0_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*0);

}
void Init_PB1_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*1);

}
void Init_PB2_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*2);

}
void Init_PB3_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*3);

}
void Init_PB4_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*4);

}
void Init_PB5_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*5);

}
void Init_PB6_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*6);

}
void Init_PB7_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*7);

}
void Init_PB8_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*8);

}
void Init_PB9_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*9);

}
void Init_PB10_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*10);

}
void Init_PB11_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*11);

}
void Init_PB12_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*12);

}
void Init_PB13_Input(){
    RCC->IOPENR |= (1U << 1);
```

```
    GPIOA->MODER &= ~(3U << 2*13);

}
void Init_PB14_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*14);

}
void Init_PB15_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*15);

}
//Input Functions

//Output Init
void Init_PB0_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*0);
    GPIOB->MODER |= (1U << 2*0);
}
void Init_PB1_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*1);
    GPIOB->MODER |= (1U << 2*1);
}
void Init_PB2_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*2);
    GPIOB->MODER |= (1U << 2*2);
}
void Init_PB3_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*3);
    GPIOB->MODER |= (1U << 2*3);
}
void Init_PB4_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*4);
    GPIOB->MODER |= (1U << 2*4);}
void Init_PB5_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*5);
    GPIOB->MODER |= (1U << 2*5);}
void Init_PB6_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*6);
    GPIOB->MODER |= (1U << 2*6);}
void Init_PB7_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*7);
    GPIOB->MODER |= (1U << 2*7);}
void Init_PB8_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*8);
    GPIOB->MODER |= (1U << 2*8);}
void Init_PB9_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*9);
    GPIOB->MODER |= (1U << 2*9);}
void Init_PB10_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*10);
    GPIOB->MODER |= (1U << 2*10);}
void Init_PB11_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*11);
    GPIOB->MODER |= (1U << 2*11);}
void Init_PB12_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*12);
    GPIOB->MODER |= (1U << 2*12);}
void Init_PB13_Output(){
```

```
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*13);
    GPIOB->MODER |= (1U << 2*13);}
void Init_PB14_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*14);
    GPIOB->MODER |= (1U << 2*14);
}
void Init_PB15_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*15);
    GPIOB->MODER |= (1U << 2*15);
}
void Init_PA0_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*0);
    GPIOB->MODER |= (1U << 2*0);
}
void Init_PA1_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*1);
    GPIOB->MODER |= (1U << 2*1);
}
void Init_PA2_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*2);
    GPIOB->MODER |= (1U << 2*2);
}
void Init_PA3_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*3);
    GPIOB->MODER |= (1U << 2*3);
}
void Init_PA4_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*4);
    GPIOB->MODER |= (1U << 2*4);
}
void Init_PA5_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*5);
    GPIOB->MODER |= (1U << 2*5);
}
void Init_PA6_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*6);
    GPIOB->MODER |= (1U << 2*6);
}
void Init_PA7_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*7);
    GPIOB->MODER |= (1U << 2*7);
}
void Init_PA8_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*8);
    GPIOB->MODER |= (1U << 2*8);
}
void Init_PA9_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*9);
    GPIOB->MODER |= (1U << 2*9);
}
void Init_PA10_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*10);
    GPIOA->MODER |= (1U << 2*10);
}
void Init_PA11_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*11);
    GPIOB->MODER |= (1U << 2*11);
}
```

```
void Init_PA12_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*12);
    GPIOB->MODER |= (1U << 2*12);
}
void Init_PA13_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*13);
    GPIOB->MODER |= (1U << 2*13);
}
void Init_PA14_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*14);
    GPIOB->MODER |= (1U << 2*14);
}
void Init_PA15_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*15);
    GPIOB->MODER |= (1U << 2*15);
}
//output Functions
void Set_PB0(){
    GPIOB->ODR |= (1U<<0) ;
}
void Set_PB1(){
    GPIOB->ODR |= (1U<<1) ;
}
void Set_PB2(){
    GPIOB->ODR |= (1U<<2) ;
}
void Set_PB3(){
    GPIOB->ODR |= (1U<<3) ;
}
void Set_PB4(){
    GPIOB->ODR |= (1U<<4) ;
}
void Set_PB5(){
    GPIOB->ODR |= (1U<<5) ;
}
void Set_PB6(){
    GPIOB->ODR |= (1U<<6) ;
}
void Set_PB7(){
    GPIOB->ODR |= (1U<<7) ;
}
void Set_PB8(){
    GPIOB->ODR |= (1U<<8) ;
}
void Set_PB9(){
    GPIOB->ODR |= (1U<<9) ;
}
void Set_PB10(){
    GPIOB->ODR |= (1U<<10) ;
}
void Set_PB11(){
    GPIOB->ODR |= (1U<<11) ;
}
void Set_PB12(){
    GPIOB->ODR |= (1U<<12) ;
}
void Set_PB13(){
    GPIOB->ODR |= (1U<<13) ;
}
void Set_PB14(){
    GPIOB->ODR |= (1U<<14) ;
}
void Set_PB15(){
    GPIOB->ODR |= (1U<<15) ;
}
void Set_PA0(){
    GPIOA->ODR |= (1U<<0) ;
}
void Set_PA1(){
```

```
    GPIOA->ODR |= (1U<<1) ;
}
void Set_PA2(){
    GPIOA->ODR |= (1U<<2) ;
}
void Set_PA3(){
    GPIOA->ODR |= (1U<<3) ;
}
void Set_PA4(){
    GPIOA->ODR |= (1U<<4) ;
}
void Set_PA5(){
    GPIOA->ODR |= (1U<<5) ;
}
void Set_PA6(){
    GPIOA->ODR |= (1U<<6) ;
}
void Set_PA7(){
    GPIOA->ODR |= (1U<<7) ;
}
void Set_PA8(){
    GPIOA->ODR |= (1U<<8) ;
}
void Set_PA9(){
    GPIOA->ODR |= (1U<<9) ;
}
void Set_PA10(){
    GPIOA->ODR |= (1U<<10) ;
}
void Set_PA11(){
    GPIOA->ODR |= (1U<<11) ;
}
void Set_PA12(){
    GPIOA->ODR |= (1U<<12) ;
}
void Set_PA13(){
    GPIOA->ODR |= (1U<<13) ;
}
void Set_PA14(){
    GPIOA->ODR |= (1U<<14) ;
}
void Set_PA15(){
    GPIOA->ODR |= (1U<<15) ;
}


void Clear_PB0(){
    GPIOB->ODR &= ~(1U<<0) ;
}
void Clear_PB1(){
    GPIOB->ODR &= ~(1U<<1) ;
}
void Clear_PB2(){
    GPIOB->ODR &= ~(1U<<2) ;
}
void Clear_PB3(){
    GPIOB->ODR &= ~(1U<<3) ;
}
void Clear_PB4(){
    GPIOB->ODR &= ~(1U<<4) ;
}
void Clear_PB5(){
    GPIOB->ODR &= ~(1U<<5) ;
}
void Clear_PB6(){
    GPIOB->ODR &= ~(1U<<6) ;
}
void Clear_PB7(){
    GPIOB->ODR &= ~(1U<<7) ;
}
void Clear_PB8(){
    GPIOB->ODR &= ~(1U<<8) ;
}
```

```
void Clear_PB9(){
    GPIOB->ODR &= ~(1U<<9) ;
}
void Clear_PB10(){
    GPIOB->ODR &= ~(1U<<10) ;
}
void Clear_PB11(){
    GPIOB->ODR &= ~(1U<<11) ;
}
void Clear_PB12(){
    GPIOB->ODR &= ~(1U<<12) ;
}
void Clear_PB13(){
    GPIOB->ODR &= ~(1U<<13) ;
}
void Clear_PB14(){
    GPIOB->ODR &= ~(1U<<14) ;
}
void Clear_PB15(){
    GPIOB->ODR &= ~(1U<<15) ;
}
void Clear_PA0(){
    GPIOA->ODR &= ~(1U<<0) ;
}
void Clear_PA1(){
    GPIOA->ODR &= ~(1U<<1) ;
}
void Clear_PA2(){
    GPIOA->ODR &= ~(1U<<2) ;
}
void Clear_PA3(){
    GPIOA->ODR &= ~(1U<<3) ;
}
void Clear_PA4(){
    GPIOA->ODR &= ~(1U<<4) ;
}
void Clear_PA5(){
    GPIOA->ODR &= ~(1U<<5) ;
}
void Clear_PA6(){
    GPIOA->ODR &= ~(1U<<6) ;
}
void Clear_PA7(){
    GPIOA->ODR &= ~(1U<<7) ;
}
void Clear_PA8(){
    GPIOA->ODR &= ~(1U<<8) ;
}
void Clear_PA9(){
    GPIOA->ODR &= ~(1U<<9) ;
}
void Clear_PA10(){
    GPIOA->ODR &= ~(1U<<10) ;
}
void Clear_PA11(){
    GPIOA->ODR &= ~(1U<<11) ;
}
void Clear_PA12(){
    GPIOA->ODR &= ~(1U<<12) ;
}
void Clear_PA13(){
    GPIOA->ODR &= ~(1U<<13) ;
}
void Clear_PA14(){
    GPIOA->ODR &= ~(1U<<14) ;
}
void Clear_PA15(){
    GPIOA->ODR &= ~(1U<<15) ;
}



void Toggle_PB0(){
```

```
	GPIOB->ODR ^= (1U<<0) ;
}
void Toggle_PB1(){

	GPIOB->ODR ^= (1U<<1) ;
}
void Toggle_PB2(){

	GPIOB->ODR ^= (1U<<2) ;
}
void Toggle_PB3(){

	GPIOB->ODR ^= (1U<<3) ;
}
void Toggle_PB4(){

	GPIOB->ODR ^= (1U<<4) ;
}
void Toggle_PB5(){

	GPIOB->ODR ^= (1U<<5) ;
}
void Toggle_PB6(){

	GPIOB->ODR ^= (1U<<6) ;
}
void Toggle_PB7(){

	GPIOB->ODR ^= (1U<<7) ;
}
void Toggle_PB8(){

	GPIOB->ODR ^= (1U<<8) ;
}
void Toggle_PB9(){

	GPIOB->ODR ^= (1U<<9) ;
}
void Toggle_PB10(){

	GPIOB->ODR ^= (1U<<10) ;
}
void Toggle_PB11(){

	GPIOB->ODR ^= (1U<<11) ;
}
void Toggle_PB12(){

	GPIOB->ODR ^= (1U<<12) ;
}
void Toggle_PB13(){

	GPIOB->ODR ^= (1U<<13) ;
}
void Toggle_PB14(){

	GPIOB->ODR ^= (1U<<14) ;
}
void Toggle_PB15(){

	GPIOB->ODR ^= (1U<<15) ;
}
void Toggle_PA0(){

	GPIOA->ODR ^= (1U<<0) ;
}
void Toggle_PA1(){

	GPIOA->ODR ^= (1U<<1) ;
}
void Toggle_PA2(){
```

```
        GPIOA->ODR ^= (1U<<2) ;
}
void Toggle_PA3(){

        GPIOA->ODR ^= (1U<<3) ;
}
void Toggle_PA4(){

        GPIOA->ODR ^= (1U<<4) ;
}
void Toggle_PA5(){

        GPIOA->ODR ^= (1U<<5) ;
}
void Toggle_PA6(){

        GPIOA->ODR ^= (1U<<6) ;
}
void Toggle_PA7(){

        GPIOA->ODR ^= (1U<<7) ;
}
void Toggle_PA8(){

        GPIOA->ODR ^= (1U<<8) ;
}
void Toggle_PA9(){

        GPIOA->ODR ^= (1U<<9) ;
}
void Toggle_PA10(){

        GPIOA->ODR ^= (1U<<10) ;
}
void Toggle_PA11(){

        GPIOA->ODR ^= (1U<<11) ;
}
void Toggle_PA12(){

        GPIOA->ODR ^= (1U<<12) ;
}
void Toggle_PA13(){

        GPIOA->ODR ^= (1U<<13) ;
}
void Toggle_PA14(){

        GPIOA->ODR ^= (1U<<14) ;
}
void Toggle_PA15(){

        GPIOA->ODR ^= (1U<<15) ;
}
```

Comments

Since we are using ADC output as our signal source, due to ADC having 12-bit resolution in conversion of analog data, our limitation of Auto reload register is 4095. This means we cannot use the PWM in max frequency.

Video Link

https://youtu.be/sl3tnxIP_dI

Problem 2

Code

main.c

```c
#include"lab6_problem2.h"
#include<stdio.h>

typedef struct Sensor{

	uint8_t ax, ay, gx, gy;
} Sensor;

Sensor mpuSensor_data;

int main(void) {

	init_system();
	init_I2C();
	uint8_t data[10];

	read_I2C(MPU6050_ADDRESS,  MPU6050_WHO_AM_I, data, 1);
	printf("WHOAMI: %x\r\n",  data[0]);

	read_I2C(MPU6050_ADDRESS,  MPU6050_PWR_MGMT_I, data, 1);
	printf("PWR MG: %x\r\n",  data[0]);

	write_I2C(MPU6050_ADDRESS,  MPU6050_PWR_MGMT_I, 0x00);
	delay_ms(1000);

	read_I2C(MPU6050_ADDRESS,  MPU6050_PWR_MGMT_I, data, 1);
	printf("PWR MG: %x\r\n",  data[0]);

	// MPU data
	data[0] = MPU6050_PWR_MGMT_I; // regAddress
	data[1] = 0; // value for regAddress

	// EEPROM data
	data[0] = 0x1; // regAddress high
	data[1] = 0x00; // regAddress low
	data[2] = 1; // value for regAddress


	while(1) {
	delay_ms(10);

	writeCommon(MPU6050_ADDRESS, data, 2);
	writeCommon(EEPROM_ADDRESS, data, 3);

	read_I2C(MPU6050_ADDRESS,  MPU6050_GYRO_XOUT_L, mpuSensor_data.ax, 2);
	uint32_t b = (uint32_t)data[0] | ((uint32_t)(data[1]) << 8);
	double a = (double)b / 131.0;
	printf("%x\r\n",a);
    }
    return 0;
}
```

Lab6_problem2.h

```c
#ifndef LAB6_PROBLEM2_H_
#define LAB6_PROBLEM2_H_

#include "stm32g0xx.h"
#include <stdio.h>


#define MPU6050_ADDRESS      0x68
```

```
#define EEPROM_ADDRESS              0xD0
#define MPU6050_WHO_AM_I     0x75
#define MPU6050_PWR_MGMT_I   0x6B

#define MPU6050_ACCEL_XOUT_H 0x3B
#define MPU6050_ACCEL_XOUT_L 0x3C
#define MPU6050_ACCEL_YOUT_H 0x3D
#define MPU6050_ACCEL_YOUT_L 0x3E

#define MPU6050_GYRO_XOUT_H 0x43
#define MPU6050_GYRO_XOUT_L 0x44

void init_system();
void init_USART(uint32_t);
void init_I2C();
void read_I2C(uint8_t devAddr, uint8_t regAddr, uint8_t *data, uint32_t num);
void write_I2C(uint8_t devAddr, uint16_t regAddr, uint8_t data);
void writeCommon(uint8_t devAddr, uint8_t* data, uint32_t num);
void printChar(uint8_t);
void print (char*);
void USART2_IRQHandler(void);
void SysTick_Handler(void);
void delay_ms(uint32_t s);
void delay(unsigned int s);


#endif /* LAB6_PROBLEM2_H_ */
```

Lab6_problem2.c

```
#include "lab6_problem2.h"

static volatile uint32_t tick = 0;

void init_system() {
	SysTick_Config(SystemCoreClock / 1000);
	init_USART(9600);
}

void init_I2C() {
	RCC->IOPENR |= (1U << 1);    // Enable GPIOB
	// setup PB8 as AF6
	GPIOB->MODER &= ~(3U << 2*8);
	GPIOB->MODER |=  (2U << 2*8);
	GPIOB->OTYPER |= (1U << 8);
	// choose AF from mux
	GPIOB->AFR[1] &= ~(0xFU << 4*0);
	GPIOB->AFR[1] |=  (6  << 4*0);

	// setup PB9 as AF6
	GPIOB->MODER &= ~(3U << 2*9);
	GPIOB->MODER |=  (2 << 2*9);
	GPIOB->OTYPER |= (1U << 9);
	// choose AF6 from mux
	GPIOB->AFR[1] &= ~(15U << 4*1);
	GPIOB->AFR[1] |=  (6  << 4*1);

	RCC->APBENR1 |= (1U << 21);  // Enable I2C1

	I2C1->CR1 = 0;//clear
	I2C1->CR1 = (1U << 7);       //Error detection interrupts enabled

	I2C1->TIMINGR = (3 << 28);   //Timing prescaler
	I2C1->TIMINGR = (0x13 << 0);//SCL low period
	I2C1->TIMINGR = (0xF << 8); //SCL high period
	I2C1->TIMINGR = (0x2 << 16);//SDADEL - data hold time
	I2C1->TIMINGR = (0x4 << 20);//SCLDEL - data setup time

	I2C1->CR1 = (1U << 0); // PE Enable

	NVIC_SetPriority(I2C1_IRQn, 1);
```

```
	NVIC_EnableIRQ(I2C1_IRQn);
}
void I2C1_IRQHandler(void) {
	// only enters when error occurs
}
void init_USART(uint32_t baud) {
	RCC->IOPENR |= (1U << 0);
	RCC->APBENR1 |= (1U << 17);//: USART2 clock enable

	// setup PA2 as AF1
	GPIOA->MODER &= ~(3U << 2*2);
	GPIOA->MODER |=  (2U << 2*2);
	// choose AF1 from mux
	GPIOA->AFR[0] &= ~(0xFU << 4*2);
	GPIOA->AFR[0] |=  (1U  << 4*2);

	// setup PA3 as AF1
	GPIOA->MODER &= ~(3U << 2*3);
	GPIOA->MODER |=  (2U << 2*3);
	// choose AF1 from mux
	GPIOA->AFR[0] &= ~(15U << 4*3);
	GPIOA->AFR[0] |=  (1U  << 4*3);

	// setup UART2
	USART2->CR1 = 0; //clear
	USART2->CR1 |= (1U << 3); // TE - transmit enable
	USART2->CR1 |= (1U << 2); // RE - receive enable
	USART2->CR1 |= (1U << 5); //RXFIFO not empty interrupt enable
	USART2->BRR = (uint16_t)(SystemCoreClock / baud);
	USART2->CR1 |= (1U << 0); // UA -usart enable

	NVIC_SetPriority(USART2_IRQn, 1);
	NVIC_EnableIRQ(USART2_IRQn);
}
void USART2_IRQHandler(void) {
	uint8_t data = (uint8_t) USART2->RDR;
	printChar(data);
}
void read_I2C(uint8_t devAddr, uint8_t regAddr, uint8_t *data, uint32_t num) {
	// WRITE OPERATION
	I2C1->CR2 = 0;//clear
	I2C1->CR2 |= ((uint32_t)devAddr << 1);
	I2C1->CR2 |= (1U << 16);     // Number of bytes
	I2C1->CR2 |= (1U << 13);     // Generate Start
	while(!(I2C1->ISR & (1 << 1))); // TXIS

	I2C1->TXDR =  (uint32_t)regAddr;
	while(!(I2C1->ISR & (1 << 6))); // transmission complete

	// READ OPERATION
	I2C1->CR2 = 0;
	I2C1->CR2 |= ((uint32_t)devAddr << 1); // slave addres
	I2C1->CR2 |= (1U << 10);     // READ mode
	I2C1->CR2 |= (num << 16);    // Number of bytes
	I2C1->CR2 |= (1U << 15);     // NACK
	I2C1->CR2 |= (1U << 25);     // AUTOEND
	I2C1->CR2 |= (1U << 13);     // Generate Start

	for(size_t i = 0; i< num; i++) {
		while(!(I2C1->ISR & (1 << 2))); // wait until RXNE = 1
		data[i] = (uint8_t)I2C1 ->RXDR;
	}
}
void write_I2C(uint8_t devAddr, uint16_t regAddr, uint8_t data) {

	// WRITE OPERATION
	I2C1->CR2 = 0;
	I2C1->CR2 |= ((uint32_t)devAddr << 1);
	I2C1->CR2 |= (3U << 16);     // Number of bytes
	I2C1->CR2 |= (1U << 25);     // AUTOEND
	I2C1->CR2 |= (1U << 13);     // Generate Start

	while(!(I2C1->ISR & (1 << 1))); // TXIS
```

```
        I2C1->TXDR = (uint32_t)regAddr;

        while(!(I2C1->ISR & (1 << 1))); // TXIS
        I2C1->TXDR = ( uint32_t)regAddr;

        while(!(I2C1->ISR & (1 << 1))); // TXIS
        I2C1->TXDR = (uint32_t)data;
}
void writeCommon(uint8_t devAddr, uint8_t* data, uint32_t num) {
        // WRITE OPERATION
        I2C1->CR2 = 0;
        I2C1->CR2 |= ((uint32_t)devAddr << 1);
        I2C1->CR2 |= (num << 16);    //Number of bytes
        I2C1->CR2 |= (1U << 25);     //AUTOEND
        I2C1->CR2 |= (1U << 13);     //Generate Start

        for(size_t i=0; i<num; ++i) {
                while(!(I2C1->ISR & (1 << 1))); // TXIS
                I2C1->TXDR = data[i];
        }
}
void printChar(uint8_t c) {
        USART2->TDR = (uint16_t) c;
        while(!(USART2->ISR & (1 << 6)));
}

int write(int fd, char *buf, int len) {
        (void)fd;
        for(int i=0; i<len; ++i) {
                printChar(buf[i]);
        }
        return len;
}
void print (char*buf) {
        int len = 0;
        while(buf[len++] != '\0') {
                write(0, buf, len);
        }
}
void SysTick_Handler(void) {
        if(tick > 0) {
                --tick;
        }
}
void delay_ms(uint32_t s) {
        tick = s;
        while(tick);
}
void delay(unsigned int s) {
        for(; s>0; s--);
}
```

Comments

In this problem we wanted to implement the code written in the lesson. Code builded correctly but we couldn't try it due to lack of material.