GEBZE TECHNICAL UNIVERSITY

ELECTRONICS ENGINEERING

ELEC 335

Microprocessors Lab

LAB #7

Prepared by
1) 1801022037 Ömer Emre POLAT

1. Introduction

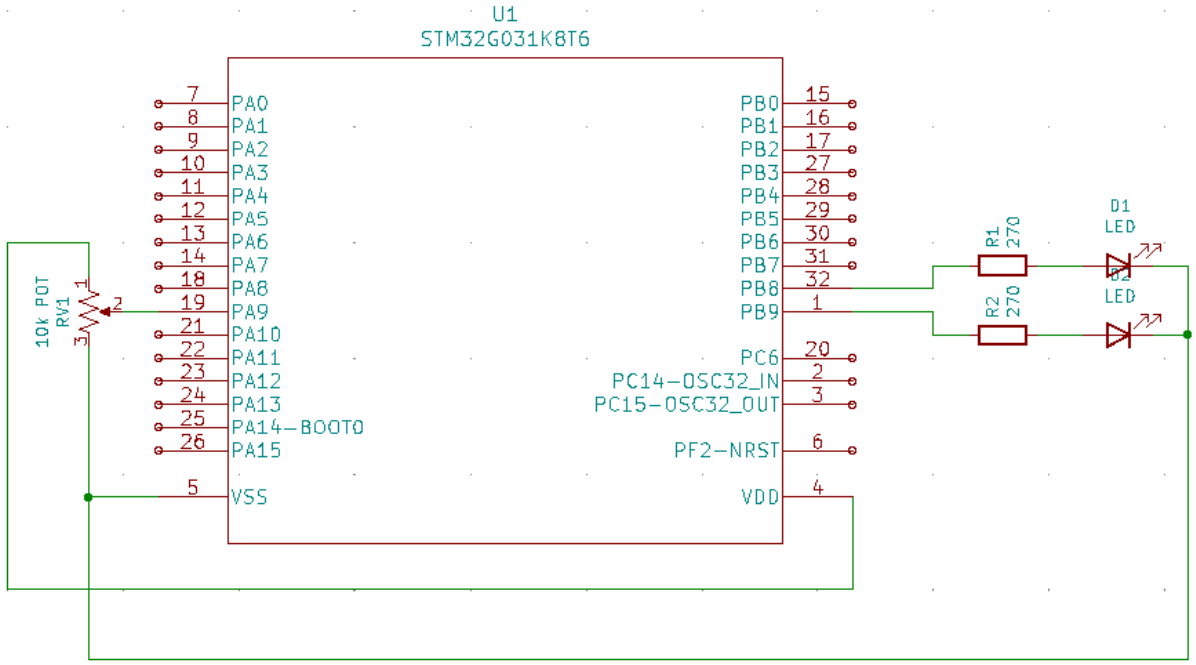
In this lab for the first problem we will implement an ADC continuous input and 2 PWM outputs to create a PWM led dimmer controlled by a potentiometer. For the second problem we will implement a knock counter using a microphone and an ADC input from the microphone.

2. Problem 1 Design

2.1. Flowchart and Block Diagram

First, we will design the board connections and the flowchart for our code for ease of design.

The board connections will be drawn on KiCAD using the model libraries of the components



All the resistors seen in the figure are 270 ohm resistors.

The resistor values are calculated using the datasheet of the STM32 board and the HS420361K-32.

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA

5.2 Absolute maximum ratings

Stresses above the absolute maximum ratings listed in *Table 18*, *Table 19* and *Table 20* may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

All voltages are defined with respect to V_{SS}.

Table 18. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
V_{DD}	External supply voltage	- 0.3	4.0	V
V_{BAT}	External supply voltage on VBAT pin	- 0.3	4.0	
V_{REF+}	External voltage on VREF+ pin	- 0.3	Min(V_{DD} + 0.4, 4.0)	
V_{IN}(1)	Input voltage on FT_xx	- 0.3	V_{DD} + 4.0(2)	
	Input voltage on any other pin	- 0.3	4.0	

Table 19. Current characteristics

Symbol	Ratings	Max	Unit
$I_{VDD/VDDA}$	Current into VDD/VDDA power pin (source)(1)	100	mA
$I_{VSS/VSSA}$	Current out of VSS/VSSA ground pin (sink)(1)	100	
$I_{IO(PIN)}$	Output current sunk by any I/O and control pin except FT_f	15	
	Output current sunk by any FT_f pin	20	
	Output current sourced by any I/O and control pin	15	
$\Sigma I_{IO(PIN)}$	Total output current sunk by sum of all I/Os and control pins	80	
	Total output current sourced by sum of all I/Os and control pins	80	
$I_{INJ(PIN)}$(2)	Injected current on a FT_xx pin	-5 / NA(3)	
$\Sigma\lvert I_{INJ(PIN)}\rvert$	Total injected current (sum of all I/Os and control pins)(4)	25	

$$I_{LED} = \frac{V}{R_{total}} = \frac{3.33V}{270} = 12.3mA < 15mA \;\&\; 30mA$$

Now that the absolute maximum DC forward current values are adjusted we can layout the pins.

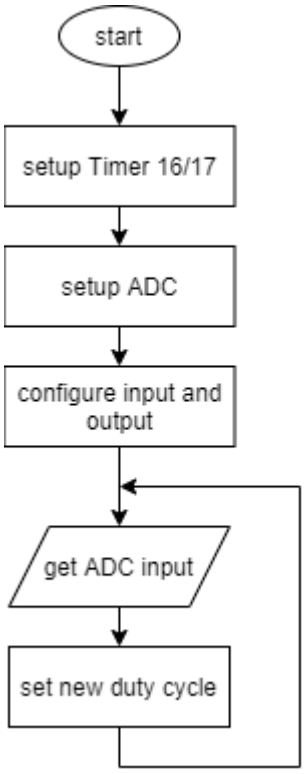
The pin connections are as follows:

PB8 -> LED1

PB9 -> LED2 PA9 -> Potantiometer input (POT)

Now that the pins are chosen we can start designing the flowchart and the code.

Code will first set the timer 16 and timer 17 for the PWM output modes that go to the LEDs. After that the ADC setup and calibration will be done and the analog input will be written to the adc1 data register. Last part of the setup phase is to setup the IO pins for the Led and Potentiometer. In the while loop the analog input will be read from the adc1 data register and processed to a new duty cycle value and the duty cycle will be set for timer 16 and timer 17.



Now that the flowchart is drawn the full code can be written.

2.2. Setting up the TIM16 and TIM17 for PWM Outputs

TIM16 and TIM17 is set knowing that they are connected to PB8 and PB9.

```
/* input and output pins setup */

    RCC->IOPENR |= (3U << 0); /* enable A and B clock */

      /* pwm timer 16 AF2 setup */

      GPIOB->MODER &= ~(3U << 2*8); /* PB8 as AF mode */
      GPIOB->MODER |= (2U << 2*8);

      GPIOB->AFR[1] &= ~(0xFU << 4*0); /* Choose AF2 from mux */
      GPIOB->AFR[1] |= (0x2U << 4*0);

      RCC->APBENR2 |= (1U << 17); /* Enable TIM16 clock */

      TIM16->CR1 = 0U; /* resetting control register */
      TIM16->CR1 |= (1U << 7); /* ARPE buffering */
```

```
      TIM16->CNT = 0U; /* reset the timer counter */

      TIM16->PSC = ((SystemCoreClock/RefreshRate) - 1); /* prescaler set to 1600-
1 */
      TIM16->ARR = 1U; /* set the autoreload register for 1 milliseconds */

      TIM16->CR1 |= (1U << 0); /* Enable TIM16 */

      TIM16->CCMR1 |= (6U << 4);

      TIM16->CCMR1 |= (1U << 3);

      TIM16->CCER |= (1U << 0);

      TIM16->CCR1 |= (1U << 0);

      TIM16->BDTR |= (1U << 15); /* main output enable */

   /* pwm timer 17 AF2 setup */

      GPIOB->MODER &= ~(3U << 2*9); /* PB9 as AF mode */
      GPIOB->MODER |= (2U << 2*9);

      GPIOB->AFR[1] &= ~(0xFU << 4*1); /* Choose AF2 from mux */
      GPIOB->AFR[1] |= (0x2U << 4*1);

      RCC->APBENR2 |= (1U << 18); /* Enable TIM17 clock */

      TIM17->CR1 = 0U; /* resetting control register */
      TIM17->CR1 |= (1U << 7); /* ARPE buffering */
      TIM17->CNT = 0U; /* reset the timer counter */

      TIM17->PSC = ((SystemCoreClock/RefreshRate) - 1); /* prescaler set to 1600-
1 */
      TIM17->ARR = 1U; /* set the autoreload register for 1 milliseconds */

      TIM17->CR1 |= (1U << 0); /* Enable TIM17 */

      TIM17->CCMR1 |= (6U << 4);

      TIM17->CCMR1 |= (1U << 3);

      TIM17->CCER |= (1U << 0);

      TIM17->CCR1 |= (1U << 0);

      TIM17->BDTR |= (1U << 15);
```

Setting the prescaler is done like this because the auto reload register is set to 1 and the duty cycle is set to %50. The duty cycle will be edited later using the data register on ADC input.

2.3. Setting the ADC input

Setting the ADC is done by following the instructions on the RM0444 reference manual.

```
/* ADC setup */

	RCC->APBENR2 |= (1U << 20); /* ADC clock enable */

	RCC->APBRSTR2 |= (1U << 20);
	RCC->APBRSTR2 &= ~(1U << 20);

	ADC1->CFGR1 = 0x0U;

	ADC1->CR |= (1U << 28); /* open voltage reg */
	for(int i = 0; i<16000; i++);

	ADC1->CR |= (1U << 31); /* ADC calibration */
	while(!(ADC1->CR >> 31) & (0x1U)); /* wait for end of calibration */

	ADC1->CFGR1 |= (2U << 3); /* 8 bit resolution */

	ADC1->CFGR1 |= (1U << 13); /* continuous conversion enable */
	ADC1->CFGR1 &= ~(1U << 16); /* discontinuous mode disable */

	ADC1->SMPR &= ~(1U << 0);

	ADC1->CFGR1 &= ~(7U << 6); /* trigger selection trg0 */
	ADC1->CFGR1 &= ~(3U << 10); /* trigger as continuos mode */

	ADC1->CFGR1 |= (9U << 26); /* PA9 channel selection */
	ADC1->CHSELR |= (1U << 9);

	GPIOA->MODER |= (3U << 9*2);

	ADC1->ISR |= (1 << 0); /* reset ready status */

	ADC1->CR |= (1 << 0); /* adc enable */
	while( !((ADC1->ISR) & (0x1U)));

	ADC1->CR |= (1U << 2);
```

ADC is set as continuous mode so that the sampling is done every 1.5 ADC clocks the setting of the values are done in while loop continuously.

2.4. Combined Code

After the setup functions are written we can combine them to make the combined code.

main.c

```
/* PROBLEM1.C              */
/* Author: Ömer Emre Polat */
/* Student No: 1801022037  */

#include "bsp.h"
```

```c
#include "stdlib.h"

int main(void) {

      BSP_system_init();

    while(1)
    {

    }

    return 0;
}
```

bsp.c

```c
/* BSP.C                    */
/* Author: Ömer Emre Polat */
/* Student No: 1801022037  */

#include "bsp.h"
#include <math.h>

/*///////////////BSP FUNCTIONS///////////////*/

void BSP_IWDG_init(void)
{
    IWDG->KR = 0x5555;
    IWDG->PR = 1; // prescaler
    while(IWDG->SR & 0x1); // wait while status update
    IWDG->KR = 0xCCCC;
}

void BSP_IWDG_refresh(void)
{
    IWDG->KR = 0xAAAA;
}

void BSP_system_init()
{
    __disable_irq();

    RCC->APBENR2 = 0x0U;

    /* input and output pins setup */

    RCC->IOPENR |= (3U << 0); /* enable A and B clock */

      /* pwm timer 16 AF2 setup */

      GPIOB->MODER &= ~(3U << 2*8); /* PB8 as AF mode */
      GPIOB->MODER |= (2U << 2*8);

      GPIOB->AFR[1] &= ~(0xFU << 4*0); /* Choose AF2 from mux */
      GPIOB->AFR[1] |= (0x2U << 4*0);

      RCC->APBENR2 |= (1U << 17); /* Enable TIM16 clock */
```

```
      TIM16->CR1 = 0U; /* resetting control register */
      TIM16->CR1 |= (1U << 7); /* ARPE buffering */
      TIM16->CNT = 0U; /* reset the timer counter */

      TIM16->PSC = ((SystemCoreClock/RefreshRate) - 1); /* prescaler set to 1600-
1 */
      TIM16->ARR = 1U; /* set the autoreload register for 1 milliseconds */

      TIM16->CR1 |= (1U << 0); /* Enable TIM16 */

      TIM16->CCMR1 |= (6U << 4);

      TIM16->CCMR1 |= (1U << 3);

      TIM16->CCER |= (1U << 0);

      TIM16->CCR1 |= (1U << 0);

      TIM16->BDTR |= (1U << 15); /* main output enable */

   /* pwm timer 17 AF2 setup */

      GPIOB->MODER &= ~(3U << 2*9); /* PB9 as AF mode */
      GPIOB->MODER |= (2U << 2*9);

      GPIOB->AFR[1] &= ~(0xFU << 4*1); /* Choose AF2 from mux */
      GPIOB->AFR[1] |= (0x2U << 4*1);

      RCC->APBENR2 |= (1U << 18); /* Enable TIM17 clock */

      TIM17->CR1 = 0U; /* resetting control register */
      TIM17->CR1 |= (1U << 7); /* ARPE buffering */
      TIM17->CNT = 0U; /* reset the timer counter */

      TIM17->PSC = ((SystemCoreClock/RefreshRate) - 1); /* prescaler set to 1600-
1 */
      TIM17->ARR = 1U; /* set the autoreload register for 1 milliseconds */

      TIM17->CR1 |= (1U << 0); /* Enable TIM17 */

      TIM17->CCMR1 |= (6U << 4);

      TIM17->CCMR1 |= (1U << 3);

      TIM17->CCER |= (1U << 0);

      TIM17->CCR1 |= (1U << 0);

      TIM17->BDTR |= (1U << 15);

   /* ADC setup */

      RCC->APBENR2 |= (1U << 20); /* ADC clock enable */

      RCC->APBRSTR2 |= (1U << 20);
      RCC->APBRSTR2 &= ~(1U << 20);

      ADC1->CFGR1 = 0x0U;
```

```
	ADC1->CR |= (1U << 28); /* open voltage reg */
	for(int i = 0; i<16000; i++);

	ADC1->CR |= (1U << 31); /* ADC calibration */
	while(!(ADC1->CR >> 31) & (0x1U)); /* wait for end of calibration */

	ADC1->CFGR1 |= (2U << 3); /* 8 bit resolution */

	ADC1->CFGR1 |= (1U << 13); /* continuous conversion enable */
	ADC1->CFGR1 &= ~(1U << 16); /* discontinuous mode disable */

	ADC1->SMPR &= ~(1U << 0);

	ADC1->CFGR1 &= ~(7U << 6); /* trigger selection trg0 */
	ADC1->CFGR1 &= ~(3U << 10); /* trigger as continuos mode */

	ADC1->CFGR1 |= (9U << 26); /* PA9 channel selection */
	ADC1->CHSELR |= (1U << 9);

	GPIOA->MODER |= (3U << 9*2);

	ADC1->ISR |= (1 << 0); /* reset ready status */

	ADC1->CR |= (1 << 0); /* adc enable */
	while( !((ADC1->ISR) & (0x1U)));

	ADC1->CR |= (1U << 2);

	//BSP_IWDG_init(); // Watchdog init

    __enable_irq();
}
```

bsp.h

```
/* BSP.H                    */
/* Author: Ömer Emre Polat */
/* Student No: 1801022037  */

#ifndef BSP_H_
#define BSP_H_

#include "stm32g031xx.h"

#define RefreshRate 10000
#define ButtonBounceTime 400

void BSP_system_init(void);

void BSP_IWDG_init(void);
void BSP_IWDG_refresh(void);

#endif
```

The setup is all implemented into the bsp_system_init function. The bsp_init_function is called before the main loop to setup the timers and ADC input.

2.2 Prices and Parts List

Parts that are used in the problem 1 can be written into the table down below.

	Part Name	Amount	Price
1	Breadboard	1	7.50TL
2	Jumper Cable (Male-Male)	10	40 piece is around 3.16TL
3	270 ohm resistors	2	~
4	LED	2	0.15TL
5	STM32G031K8T6 Board	1	102.50TL
6	10K Potentiometer	1	1.10
			TOTAL
			112.04TL

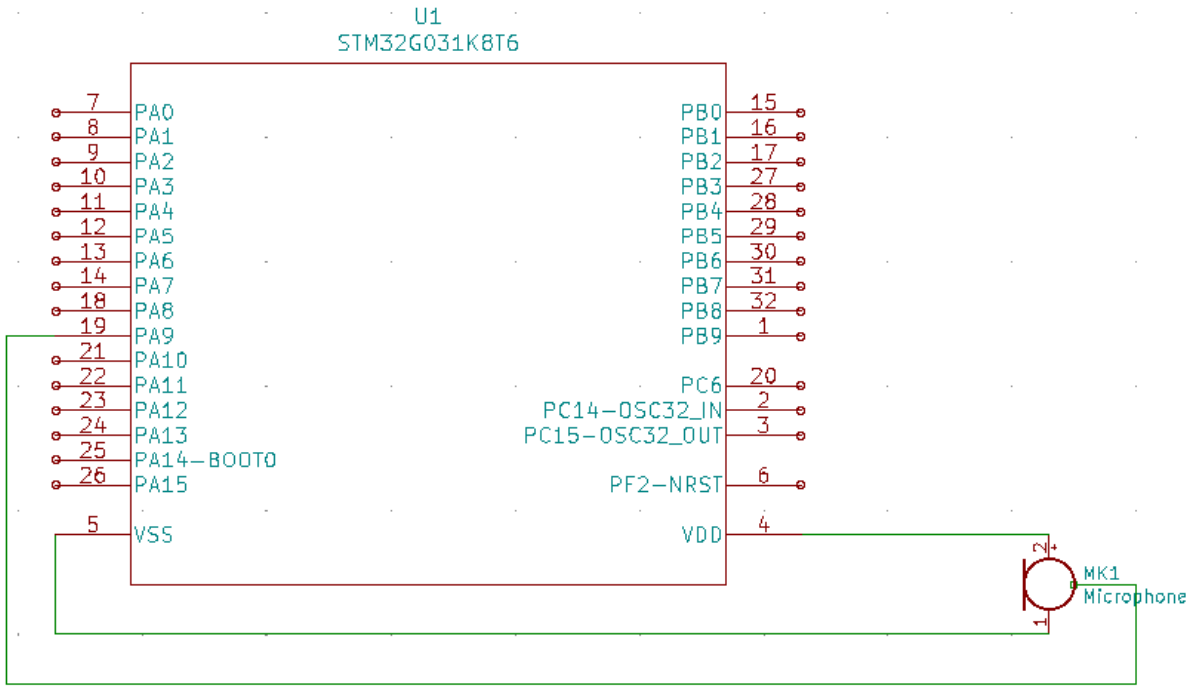
Total price list comes to around 112.04TL excluding the cargo costs. Build takes around 5 minutes using the block diagram given at the beginning of the report.

3. Problem 1 Design

3.1. Flowchart and Block Diagram

First, we will design the board connections and the flowchart for our code for ease of design.

The board connections will be drawn on KiCAD using the model libraries of the components



There are no resistors because we used a microphone unit with an amplifier.

Now that the block diagram is drawn we can layout the pins.

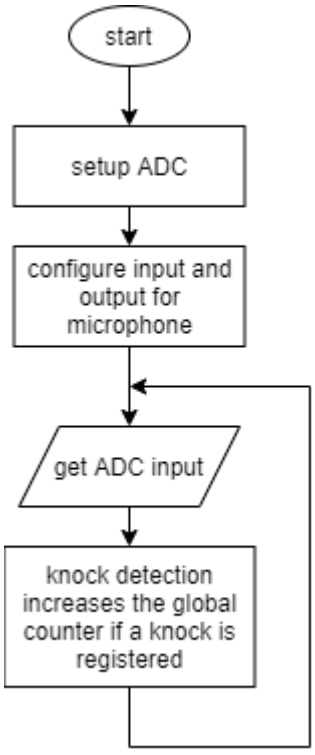
The pin connections are as follows:

3.33V -> + (Microphone)

GND -> - (Microphone) PA9 -> Microphone analog input (Microphone)

Now that the pins are chosen we can start designing the flowchart and the code.

Code will first set the peripherals and pin input output modes. Then the code will store a number and the keypad input in a global variable. These variables will be modified inside the keypad interrupt and using the input numbers the TIM17 prescaler value will be set to the correct value for the chosen frequency. At last a chosen button input will turn off the PW output.



Now that the flowchart is drawn the full code can be written.

Prices and Parts List

Parts that are used in the problem 1 can be written into the table down below.

	Part Name	Amount	Price
1	Breadboard	1	7.50TL
2	Jumper Cable (Male-Male)	10	40 piece is around 3.16TL
3	STM32G031K8T6 Board	1	102.50TL
4	MAX4466 Microphone unit	1	15.46
			TOTAL
			128.62TL

Total price list comes to around 128.62TL excluding the cargo costs. Build takes around 2 minutes using the block diagram given at the beginning of the report.

2.3 Video and Documenting the Problems

Video documentation was not done, unfortunately code was not working as intended.

4. Results and General Comments

The results were not taken because the code for the problem 1 was not working as intended and the problem 2 code couldn't be written for the same reason. ADC analog input was not consistent and the calibration may be wrong.

As general comments there were a lot of debugging to get the ADC to set and work even without calibration. The PWM output signals worked as intended but because the analog input couldn't be taken consistently the calculations for the PWM duty cycle couldn't be made.

5. References

[1]. RM0444 Reference manual
https://www.st.com/resource/en/reference_manual/dm00371828-stm32g0x1-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[2]. STM32G031K8 Datasheet

https://www.st.com/en/microcontrollers-microprocessors/stm32g031k8.html