GEBZE TECHNICAL UNIVERSITY
ELECTRONIC ENGINEERING

ELEC 334 - PROJECT 3

DIGITAL VOİCE RECORDER

PREPARED BY	MEHMET ADANIR	171024076

1.Introduction

In the final project of the microprocessors course, we had to make a project that included the topics we learned during the whole semester. we are create a digital voice recorder that can record our voice, playback a selected voice recording, and delete single or all recording data. During this project, I used various modules such as Timer, PWM, ADC, and External Interrupts

2. Block Diagram

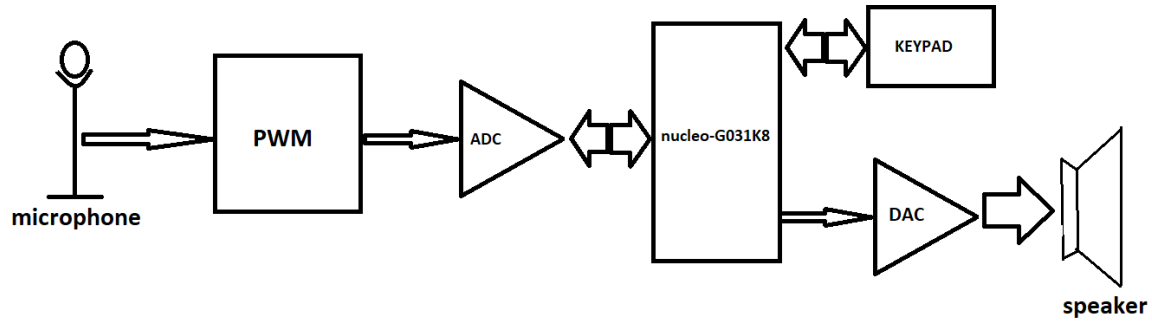


Figure 1. Block diagram of project

3.Flowchart

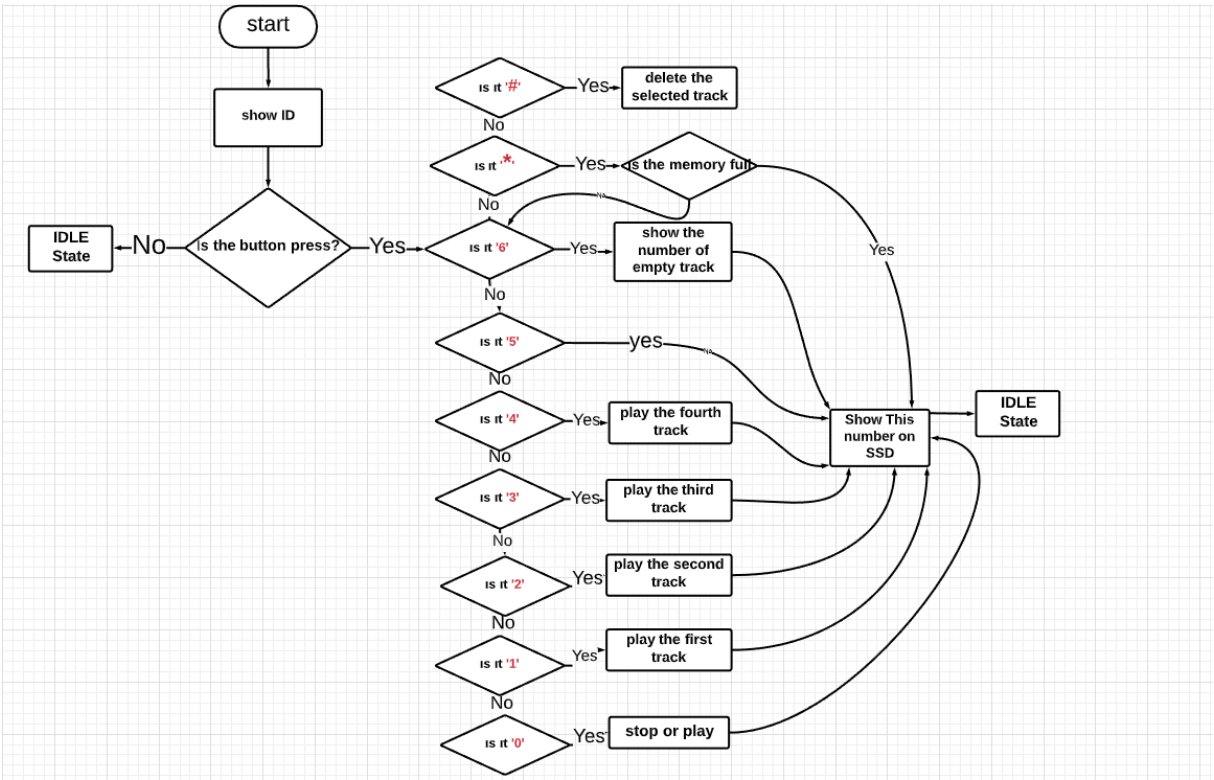


Figure 2. Flowchart of project

4.State Transition Diagram

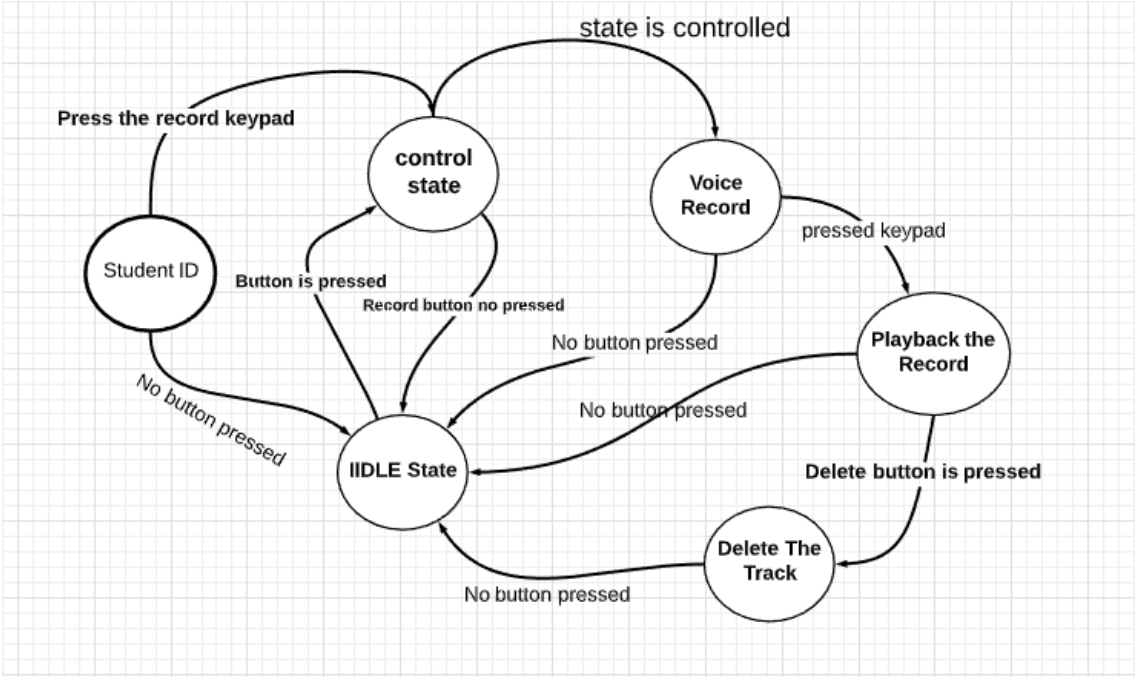


Figure 3. State transition diagram of project

5.Circuit Diagram

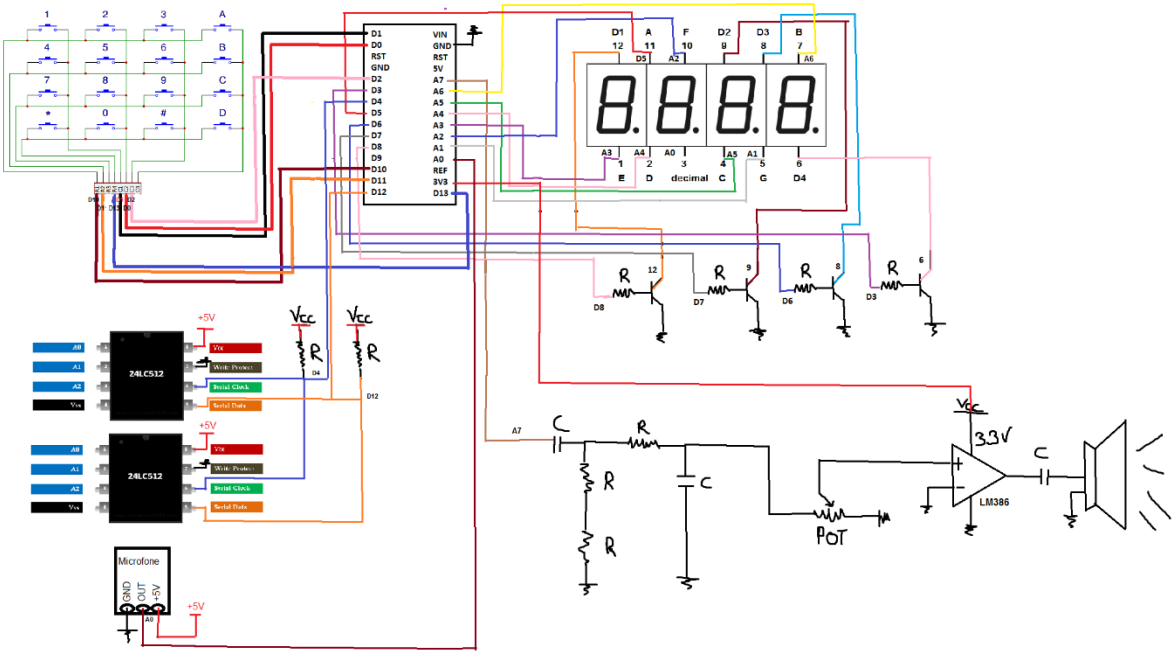


Figure 4. Circuit diagram of project

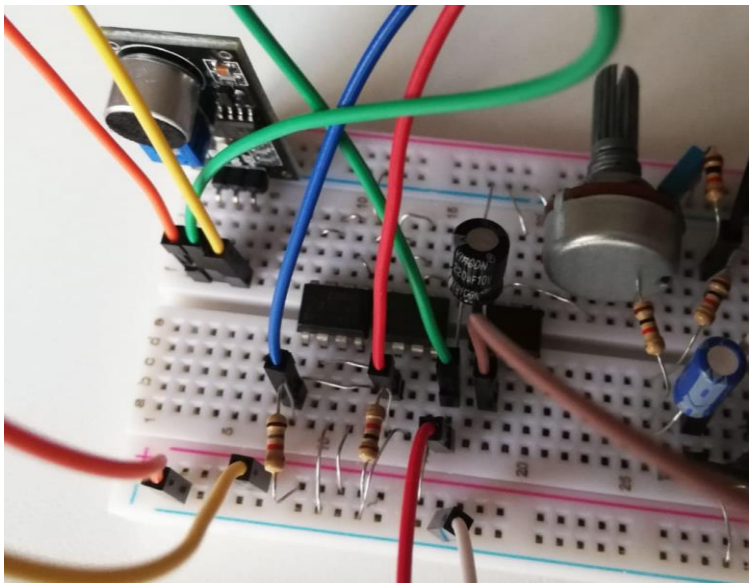
6.Tasks List

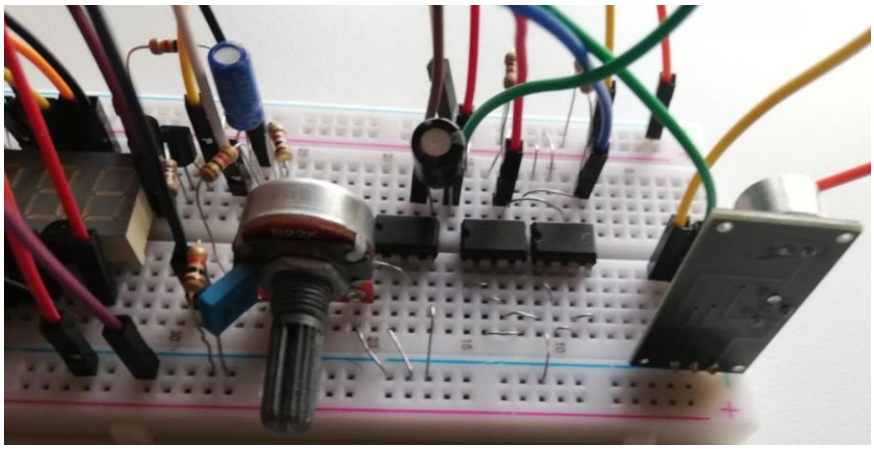
1.How the microphone will record the sound and how to keep in mind.✓
2.How to export the recorded sounds with the speaker.✓
3.How do I keep the data on my EEPROM.✗
4.Appointments will be made to keypad numbers for locations to record.✓
5.I'll record numbers in 4 different numbers.✓
6.With the record button, the recording will start and stop.✗
7.Record the sound given to the microphone from the outside and play from the speaker.✗

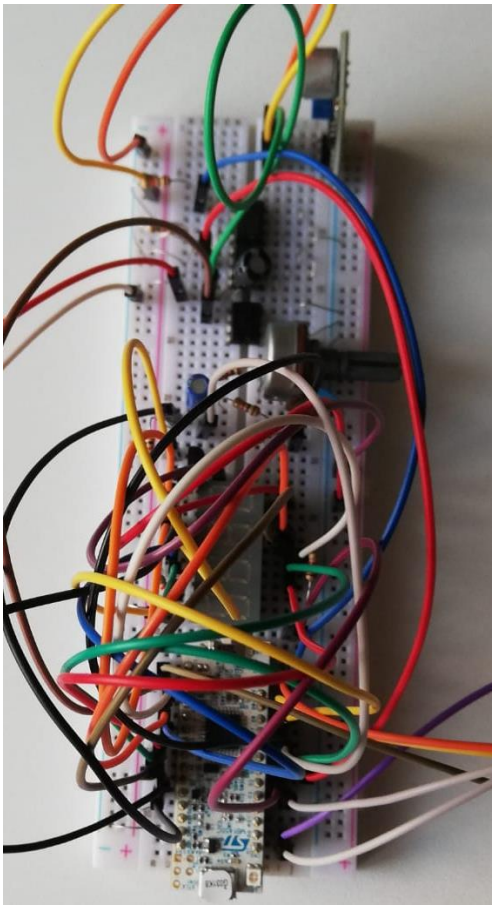
8.Pressing the '#' key will delete the recorded voice.✗
9.SSD's status will be shown.✗
10.No recording when SSD is full.✗
11.While playing any record, show the number of that record on SSD.✗
12.The free space location will be shown.✗
13.SSD will also show the current status.✓
14.School number will be shown at first. ✓

7.Project Setup

→I draw the flow chart and take up the parts.
→ I set up all pin in the stm32G0+ and pick up the part in board.
→I writed code funciton of IDLE state.
→I writed code funciton of ADC.
→I writed code funciton of PWM duty cycle.
→I writed code added on keypad number of track.
→I writed code delete number of track.







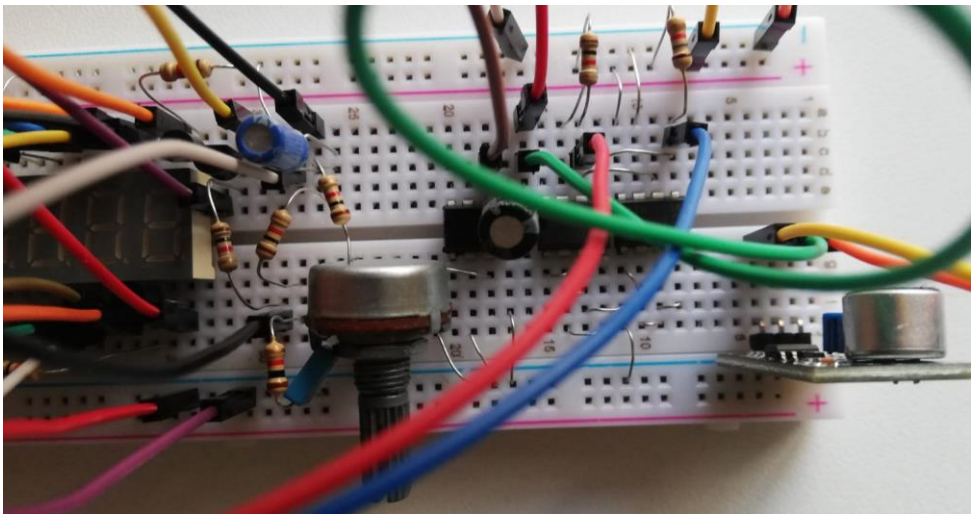


Figure 5. Circuit diagram of project

8.Parts List

NAME	NUMBER	PRİCE
NUCLEO-G031K8	1	110tl
POTENTIOMETER	1	0.25tl
TRANSİSTOR	4	3tl
RESİSTANCE	9	2tl
4XSEVEN SEGMENT	1	10tl
JUMPER	40	10tl
KEYPAD	1	8tl
MİCROPHONE	1	15tl
CAPASİTANCE(1µF,47 µF,220µF)	3	6tl
SPEAKER	1	5tl
TOTAL	62	196.25

9.Results

In this project, we were first asked to set up a digital sound recorder circuit and then write the code for it and work it.
First of all, I wrote my flowchat, I first show the last two number my school number and the last two number of it ssd. Then when I didn't press any button on the keypad in 10 second, it would go in the IDLE state.
Then I assigned the operations to be performed on the keypad from 1 to 5 numbers. And I wrote the codes for microphone and speaker to record voice.
In general, I could not do everything requested from me in the project, I could not do what was asked of us because I could not fully solve it with ADC, PWM, duty cycle and trigger. first out of the pwm output, I can hear the analog signal when mounting a microphone to the hunks, but I only heard the buzzing noise. I couldn't hear another sound
As a result, I did not understand much about this project as I lacked knowledge in many places.

Video_Link: https://youtu.be/WNh2TmZ98jI

10.References

https://www.youtube.com/playlist?list=PLiWDuW_1eKN5IpCAaeE9ncaPtxK61YFf-

11.Detailed Code

```c
/*
  main.c

 Author: Mehmet Adanır
 ID: 171024076
  */
#include "stm32g0xx.h"
#include "time.h"
#include "stdio.h"
#include "Project3.h"


#define LEDDELAY 1600000U

int main(void) {

      BSP_System_init();
      init_adc() ;
      init_timer1();
      init_I2C();

      Keypad_enable();


    return 0;
}
```

```c
/*
  project3.h

 Author: Mehmet Adanır
 ID: 171024076
  */
#ifndef PORJECT3_H_
#define PROJECT3_H_

#include "stm32g0xx.h"

/* Common API functions for nucleo */

void delay_ms(uint32_t);
void delay(volatile unsigned int);


void showID();
void _print(int , char * , int );
void BSP_UART_init(uint32_t);
void printChar(uint8_t);
void BSP_System_init();

void init_adc() ;
unsigned int ADC_start(void);
void init_timer1() ;
void EXTI4_15_IRQHandler ();
// LED related functions
void Keypad_enable();
void BSP_led_init();
void BSP_led_set();
void BSP_led_clear();
void BSP_led_toggle();

void setSSD(int x , int y);
void SwitchSSD(int x);

// Button related functions
void BSP_button_init();
int BSP_button_read();

void IDLE();
void ssd_clear();
void setRowsKeypad();
void clearRowsKeypad();
#endif
```

```c
/*
  project.c

 Author: Mehmet Adanır
 ID: 171024076
  */
#include "stm32g0xx.h"
#include "project3.h"
#include "math.h"
#include "time.h"

#define LC512_ADDRESS 0X68
#define LC512_ADDRESS 0X65

#define LC512_WHO_AM_I 0X75
#define LC512_PWR_MGMT_1 0X6B


static volatile  uint32_t tick = 0;
int t=0;
static volatile int num;

void BSP_led_init(void) {

       /* Enable GPIOA clock */  /* Enable GPIOB clock */
           RCC->IOPENR |= (3U << 0);

                  /* setup PA(0,1,4,5,6,8,9,11,12) for seven segment
A,B,C,D,E,F,G,DH for bits in MODER */
                        GPIOA->MODER &= ~(0x3CF3F0F);
                        GPIOA->MODER |= (0x1451505);
                     /* setup PB(0,1,2,8) for seven segment D4,D3,D2,D1 for in
MODER */
                        GPIOB->MODER &= ~(0x3003F);
                        GPIOB->MODER |= (0x10015);

}

void delay(volatile unsigned int s) {
    for(; s>0; s--);
}

void delay_ms(uint32_t s) {
    tick = s;
    while(tick);
}

void SysTick_Handler(void) {

      if(tick > 0){

            --tick;
      }

}
```

```
void init_timer1() {

      RCC->APBENR1 |= (1U << 1); // enable TIM3 module clock

      TIM3->CCR3 = 0; // Zero out the control register just in case
      TIM3->CR1 |= (1 << 7); // ARPE
      TIM3->CNT = 0; // Zero out counter
    TIM3->CCMR2 |= (1U << 16); // PWM Mode1
    TIM3->CCER |= (0U << 0); // capture / compare output

    /// 1  second interrupt
      TIM3->PSC = 999; //1Mhz
      TIM3->ARR = 1600;

      TIM3->DIER |= (1 << 0);  // update interrupt enable
      TIM3->CR1  |= (1 << 0);  // TIM1 Enable

    NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn, 1);
    NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);

}

//Timer Handler
void TIM3_IRQHandler(){


    // clear interrupt status
    if (TIM3->DIER & 0x01) {
        if (TIM3->SR & 0x01) {
            TIM3->SR &= ~(1U << 0);
        }
    }

    GPIOD->ODR ^= (1 << 13);

}
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 = 99;                                     //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 TIM1_BRK_UP_TRG_COM_IRQHandler(void) {
    num = (int)ADC_start();
    double buffer = tick;
    for(;tick-buffer>1800;);
    if(num>2200){
    read_write_data();
    TIM1->SR &= ~(1U << 0);
    }
}
void init_adc() {
RCC->APBENR2 |= (1U << 20); //enable rcc for adc
RCC->IOPENR = (1U << 1); //enable GPIOB
GPIOA->MODER|=(1U<<7);
//PB1 pin for adc in analog mode (by default)
ADC1->CR=0; //reset adc cr
ADC1->CFGR1 = 0;//reset adc cfgr1
ADC1 ->CR |= (1U << 28); // Enable adc voltage regulator
delay(500); //delay >20 us

//enable calibration, wait until completion
ADC1->CR |= (1U << 31); //calibration enable
while(((ADC1->CR>>31)==1));//Wait until calibration.
//enable end of cal. or sequence interrupts
// ADC1->IER |= (1U << 3); //end of conversion sequence interrupt
ADC1->IER |= (1U << 11); //end of calibration interrupt
// select resolution [conf. bit sample (6,8,10,12)]
ADC1 ->CFGR1 |= (2U << 3);// ; 8bit
//conf. single/continuous;
ADC1->CFGR1 &= ~(1U << 13);//cont=0;
ADC1->CFGR1 &= ~(1U << 16);//discen =8; single
//select sampling time from SMPR
ADC1->SMPR = (0 << 0);//SMP1
// ADC1->SMPR |= (10 << 4);//SMP2

//select tim trgo
ADC1->CFGR1 |= (3U << 6); //TGRO (extsel); 0xb011=3U for TIM3_TRGO
ADC1->CFGR1 |= (1U << 10); //Choose detect at rising edge (exten); 01

//enable channels (for the Anx pins)
ADC1->CFGR1 |= (9U << 26);//analog input channel 9; PB1
ADC1->CHSELR |= (1U << 9);//analog input channel 9; PB1
//Clear the ADRDY bit in ADC_ISR register by programming this bit to 1.
ADC1->ISR |= (1 << 0);
//enable adc and wait until it is ready
ADC1->CR |= (1 << 0);
while( (ADC1->ISR & (1 << 0)));
//Start conversion
ADC1->CR |= (1U << 2);
NVIC_SetPriority(ADC1_IRQn, 2); //Set priority to 2
NVIC_EnableIRQ(ADC1_IRQn); //Enable NVIC for TIM1
   }
```

```c
void I2C1_IRQHandler(void) {
	// only enters when error
}
void Init_I2C(void) {

	GPIOB->MODER &= ~(3U << 2*8);
	GPIOB->MODER |= (2 << 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] &= ~(0XFU<< 4*1);
	GPIOB->AFR[1] |= (6 << 4*1);

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

    I2C1->CR1 = 0;
    I2C1->CR1 |= (1U << 7); // ERRI

    I2C1->TIMINGR |= (3 << 28); // PRESC
    I2C1->TIMINGR |= (0x13 << 0); // SCLL
    I2C1->TIMINGR |= (0xF << 8); // SCLH
    I2C1->TIMINGR |= (0x2 << 16); // SDADEL
    I2C1->TIMINGR |= (0x4 << 20); // SCLDEL

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

    NVIC_SetPriority(I2C1_IRQn, 1);
    NVIC_EnableIRQ(I2C1_IRQn);
}
void Read(uint8_t devAddr, uint8_t regAddr, uint8_t *data, uint32_t num){
//WRITE OPERATION (Send address and register to read)
I2C1->CR2 = 0;
I2C1->CR2 |= ((uint32_t) devAddr << 1); // slave address
I2C1->CR2 |= (1U << 16); // Number of byte
I2C1->CR2 |= (1U << 13); // Generate Start
while(!(I2C1->ISR & (1 << 1))); // TXTS
I2C1 ->TXDR = (uint32_t) regAddr;
while(!(I2C1->ISR & (1 << 6))); // TC
// READ OPERATION (read data)
I2C1->CR2 = 0;
I2C1->CR2 |= ((uint32_t) devAddr << 1);
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

}
   }
```

```
void Write(uint8_t devAddr , uint16_t num , uint8_t* data){

	//WRITE OPERATION (Send address and register to read)
	I2C1->CR2 = 0;
	I2C1->CR2 |= ((uint32_t)devAddr << 1);//slave address
	I2C1->CR2 |= (3U << 16); // Number of byte
	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 read_write_data(){

uint8_t data[10]; // stack , not zero , garbage data

Read(LC512_ADDRESS , LC512_WHO_AM_I , data , 1);

Read(LC512_ADDRESS , LC512_PWR_MGMT_1 , data , 1);

Write(LC512_ADDRESS , LC512_PWR_MGMT_1, 0x00);
delay_ms(1000);

Read(LC512_ADDRESS , LC512_PWR_MGMT_1 , data , 1);

}

/*
void enableEEPROM(uint16_t regAddr,uint8_t data){
	data[0]=I2C1->CR2 | ((uint32_t)devAddr << 1);//regADDRESS high
    data[1]=I2C1->CR2 | ((uint32_t)devAddr << 0);//regAddress low
	data[2]=(uint32_t)regAddr;//value of regADDRESS
    data[3]=(uint32_t)regAddr;//VALUE for regAddress

	//write to address 0x100
	data[0]=1;
	data[0]=0x00;
	data[1]=0;
	write_general(EEPROM_ADDRESS,data,3);
}*/
unsigned int ADC_start(void){

	ADC1->CR |= (1U << 2);                          /* Start ADC */
	while(!(ADC1->ISR & (1U << 2)));/* Is there any data? */
	return ADC1->DR;                                /* Data from pin */
   }
```

```
void EXTI4_15_IRQHandler(void) {   //INTERRUPT function
      ssd_clear();
         //Small delay introduced to prevent bouincing
          delay(200);
          EXTI->RPR1 |= (1U << 5);    //Set hardware raised flag to zero by
software                        //Sets only the interrupted pin to zero

      if((EXTI->RPR1 >>6) & 1  ){/* Interrupt from PB6 */
      clearRowsKeypad();
      GPIOB->ODR ^= (1U << 9);  // PB9
      if((GPIOB->IDR >> 6) & 1 ){//'1'
            setSSD(1,3);
            read_write_data();

            //play track 1 and show 1 on SSD
            }

      GPIOB->ODR ^= (1U << 9);

      GPIOB->ODR ^= (1U << 5);  // PB5
      if((GPIOB->IDR >> 6) & 1 ){//'4'
            setSSD(4,3);
            //play track 4 and show 4 on SSD
      }

       GPIOB->ODR ^= (1U << 5);
       GPIOB->ODR ^= (1U << 4);  // PB4
       if((GPIOB->IDR >> 6) & 1 ){//'7'

             //NOTHING
      }
       GPIOB->ODR ^= (1U << 4);

       GPIOB->ODR ^= (1U << 3);  // PB3
       if((GPIOB->IDR >> 6) & 1 ){//*
             TIM1_BRK_UP_TRG_COM_IRQHandler();

             //start recording tracks
      }
       GPIOB->ODR ^= (1U << 3);

        EXTI->RPR1 |= (1U << 6);//Clear interrupt flag
        setRowsKeypad();
}
      if((EXTI->RPR1 >>7) & 1 ){/* Interrupt from PB7 */

      clearRowsKeypad();

      GPIOB->ODR ^= (1U << 9);  // PB9
      if((GPIOB->IDR >> 7) & 1 ){//'2'

            //play track 2 and show 2 on SSD
      }

      GPIOB->ODR ^= (1U << 9);

      GPIOB->ODR ^= (1U << 5);  // PB5
      if((GPIOB->IDR >> 7) & 1 ){//'5'

            //show the number of tracks on SSD
      }
```

```
void ssd_clear(void) {

	/* Set all output connected to SSD (clear SSD)*/
	GPIOA->BRR |= (0x1A72);

}

void showNumber() {

  for (unsigned int retTime = time(0) + 2000; time(0) < retTime; retTime--){
// Loop until it arrives.

	showID();    //My school ID show and loop

	 if(retTime == 0)//wait 10 sec and no press button go to clear SSD
	   break;
  }

	 IDLE();

}
void IDLE(){
	 ssd_clear();//off SSD
	 while(1){
	    //wait here until the press button
	 }

}

void Keypad_enable(){

/*   Setup Output pins (rows) */

	  GPIOB->MODER &= ~(3U << 2*9);  /// PB9 is output
	  GPIOB->MODER |= (1U << 2*9);


	  GPIOB->MODER &= ~(3U << 2*5);  /// PB5 is output
	  GPIOB->MODER |= (1U << 2*5);


	  GPIOB->MODER &= ~(3U << 2*4);  /// PB4 is output
	  GPIOB->MODER |= (1U << 2*4);


	  GPIOB->MODER &= ~(3U << 2*3);  /// PB3 is output
	  GPIOB->MODER |= (1U << 2*3);


	  /*   Setup Input pins (Columns)   */

	  GPIOB->MODER &= ~(3U << 2*6);  /// PB6 is input
	  GPIOB->PUPDR |= (2U << 2*6);   /// Pull-Down mode


	  GPIOB->MODER &= ~(3U << 2*7);  /// PB7 is input
	  GPIOB->PUPDR |= (2U << 2*7);    /// Pull-Down mode
```

```
   /* Setup interrupts for inputs */
        EXTI->EXTICR[1] |= (1U << 8*2);   // PB6
        EXTI->EXTICR[1] |= (1U << 8*3);   // PB7
        EXTI->EXTICR[3] |= (0U << 8*3);   // PA15
        EXTI->EXTICR[2] |= (0U << 8*2);   // PA10


        /* RISING Edge*/
        EXTI->RTSR1 |= (1U << 6);      // 6th pin
        EXTI->RTSR1 |= (1U << 7);      // 7th pin
        EXTI->RTSR1 |= (1U << 15);      // 15th pin
        EXTI->RTSR1 |= (1U << 10);      // 10th pin

        /* MASK*/
        EXTI->IMR1 |= (1U << 6);
        EXTI->IMR1 |= (1U << 7);
        EXTI->IMR1 |= (1U << 15);
        EXTI->IMR1 |= (1U << 10);

        /*NVIC */

        NVIC_SetPriority(EXTI4_15_IRQn , 0);
        NVIC_EnableIRQ(EXTI4_15_IRQn);


           /* Setup all rows*/
       GPIOB->ODR |= (1U << 9);   /// PB9
       GPIOB->ODR |= (1U << 5);   /// PB5
       GPIOB->ODR |= (1U << 4);   /// PB4
       GPIOB->ODR |= (1U << 3);   /// PB3

       ssd_clear();//turn off SSD
       while(1){
            if(t==0){ // start value t=0 must be in
            showNumber();    // show School number  wait here
             }

    }
}

void showID(){ //My school ID show
    setSSD(1 , 3);//1
    delay(1600);//delay ms
    setSSD(7 , 2);//7
    delay(1600);//delay ms
    setSSD(7 , 1);//2
    delay(1600);//delay ms
    setSSD(6 , 0);//4
    delay(1600);//delay ms
}
```

```c
void SwitchSSD(int x) {


	switch (x)
		{

		case 0://'D'

			 /* turn on led connected to A,B,C,D,E,F in ODR*/
			GPIOA->ODR |= (0x1A70);
			/* turn off led connected to G in ODR*/
			GPIOA->BRR |= (0x2);
			break;

		case 1:
			 /* turn on led connected to B,C in ODR*/
			GPIOA->ODR |= (0x840);
			/* turn off led connected to A,D,E,F,G in ODR*/
			GPIOA->BRR |= (0x1232);
			break;
		case 2:
			 /* turn on led connected to A,B,D,E,G in ODR*/
			GPIOA->ODR |= (0x1262);
			/* turn off led connected to C,F in ODR*/
			GPIOA->BRR |= (0x810);
			break;
		case 3:
			 /* turn on led connected to A,B,C,D,G in ODR*/
			GPIOA->ODR |= (0x1A42);
			/* turn off led connected to E,F in ODR*/
			GPIOA->BRR |= (0x30);
			break;
		case 4:
			 /* turn on led connected to B,C,G,F in ODR*/
			GPIOA->ODR |= (0x852);
			/* turn off led connected to A,D,E in ODR*/
			GPIOA->BRR |= (0x1220);
			break;

		case 5:
			 /* turn on led connected to A,C,D,F,G in ODR*/
			GPIOA->ODR |= (0x1A12);
			/* turn off led connected to B,E in ODR*/
			GPIOA->BRR |= (0x60);
			break;

		case 6:
			 /* turn on led connected to A,B,C,D,E,F,G in ODR*/
			GPIOA->ODR |= (0x1A32);
			/* turn off led connected to B in ODR*/
			GPIOA->BRR |= (0x40);
			break;
		case 7:
			 /* turn on led connected to A,B,C in ODR*/
			GPIOA->ODR |= (0xA40);
			/* turn off led connected to D,E,F,G in ODR*/
			GPIOA->BRR |= (0x1032);
			break;
```

```
case 8://'B'
                    /* turn on led connected to all in ODR*/
                   GPIOA->ODR |= (0x1A72);
                   break;

             case 9:
                    /* turn on led connected to A,B,C,D,F,G in ODR*/
                   GPIOA->ODR |= (0x1A52);
                   /* turn off led connected to E in ODR*/
                   GPIOA->BRR |= (0x20);
                   break;
             case 10://'A'
                    /* turn on led connected to A,B,C,F,E,G in ODR*/
                   GPIOA->ODR |= (0xA72);
                   /* turn off led connected to D in ODR*/
                   GPIOA->BRR |= (0x1000);
            break;
             case 11://'V'
                   /* turn on led connected to B,F,G in ODR*/
                   GPIOA->ODR |= (0x52);
                   /* turn off led connected to A,D,E,C in ODR*/
                   GPIOA->BRR |= (0x1A20);
            break;

             case 12://'R'
                   /* turn on led connected to A,D,E,B,F,G in ODR*/
                   GPIOA->ODR |= (0x1272);
                   /* turn off led connected to C in ODR*/
                   GPIOA->BRR |= (0x800);
            break;
             case 13://'C'
                   /* turn on led connected to A,D,E,F in ODR*/
                   GPIOA->ODR |= (0x1230);
                   /* turn off led connected to B,C,G in ODR*/
                   GPIOA->BRR |= (0x842);
            break;
             case 14://'P'
                   /* turn on led connected to A,B,G,E,F in ODR*/
                   GPIOA->ODR |= (0x272);
                   /* turn off led connected to D,C in ODR*/
                   GPIOA->BRR |= (0x1800);
            break;
             case 15://'L'
                   /* turn on led connected to D,E,F in ODR*/
                   GPIOA->ODR |= (0x1030);
                   /* turn off led connected to A,B,G,C in ODR*/
                   GPIOA->BRR |= (0xA42);
            break;
             }
   }
```

```
void setSSD(int x , int y) { // x is the number led(0 , 1)  Y is digit (SSD1 ,
SSD2)

      if(y == 3){

                    /* turn on SSD 1(LEFT).*/
                     /* turn on ODR*/
                    GPIOB->ODR |= (0x100);

                    /* turn off SSD 2.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x4);

                    /* turn off SSD 3.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x1);

                    /* turn off SSD 4.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x2);

                     SwitchSSD(x);
  }
      if(y == 2){

                    /* turn off SSD 1(LEFT).*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x100);

                    /* turn on SSD 2.*/
                     /* turn on ODR*/
                    GPIOB->ODR |= (0x4);

                    /* turn off SSD 3.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x1);

                    /* turn off SSD 4.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x2);

                     SwitchSSD(x);
  }    if(y == 1){
                    /* turn off SSD 1(LEFT).*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x100);

                    /* turn off SSD 2.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x4);

                    /* turn on SSD 3.*/
                     /* turn on ODR*/
                    GPIOB->ODR |= (0x1);

                    /* turn off SSD 4.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x2);

                     SwitchSSD(x);
     }
```

```
      if(y == 0){

                    /* turn off SSD 1(LEFT).*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x100);

                    /* turn off SSD 2.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x4);

                    /* turn off SSD 3.*/
                     /* turn off ODR*/
                    GPIOB->BRR |= (0x1);

                    /* turn on SSD 4.*/
                     /* turn on ODR*/
                    GPIOB->ODR |= (0x2);

                     SwitchSSD(x);
  }

}
void BSP_System_init() {

      __disable_irq();
      SystemCoreClockUpdate();
      BSP_led_init();
      SysTick_Config(SystemCoreClock / 1000);
    __enable_irq();
}


void clearRowsKeypad(void){
           /* Clearing the rows here */
            GPIOB->ODR &= ~(1U << 9);   /// PB9
              GPIOB->ODR &= ~(1U << 5);   /// PB5
              GPIOB->ODR &= ~(1U << 4);   /// PB4
              GPIOB->ODR &= ~(1U << 3);   /// PB3
}

void setRowsKeypad(void){
         /* Setting the rows here   */
              GPIOB->ODR |= (1U << 9);   /// PB9
            GPIOB->ODR |= (1U << 5);   /// PB5
              GPIOB->ODR |= (1U << 4);   /// PB4
              GPIOB->ODR |= (1U << 3);   /// PB3
}
```