SCHOOL OF ELCTRICAL ENGINEERING
FACULTY OF ENGINEERING
UNIVERSITI TEKNOLOGI MALAYSIA

20222023 - 1
SEEL 5123 – 01 / MKEL 1123 – 01
PROJECT REPORT
GROUP 2

Temperature Detection and Alarm System

1 ZHANG ZHEN

2 TIAN LINXUE

3 LIM ZHI

1. Purpose

In this course design, an intelligent temperature detection and alarm system based on STM32F103 (Blue Pill) is implemented.to realize the collection and display of temperature, set the alarm temperature, and alarm the environment exceeding the set temperature. The two interfaces which are I2C and ADC that proposed in stage 2 proposal is applied in this project.

2. Equipment

STM32F103C8T6 (Blue Pill) microcontroller is selected as the main control chip, a DS18B20 temperature sensor is used to collect temperature data, and the temperature is displayed on the OLED screen, as well as the buzzer for notification and button are connected to set the alarm temperature. After the threshold temperature is exceeded, the alarm function is realized. The specific circuit can be seen in the figure 1 below. The push buttons are connected in active low connected and is pulled up through hardware. All the pins of Blue Pill that used is showed in Table 1.

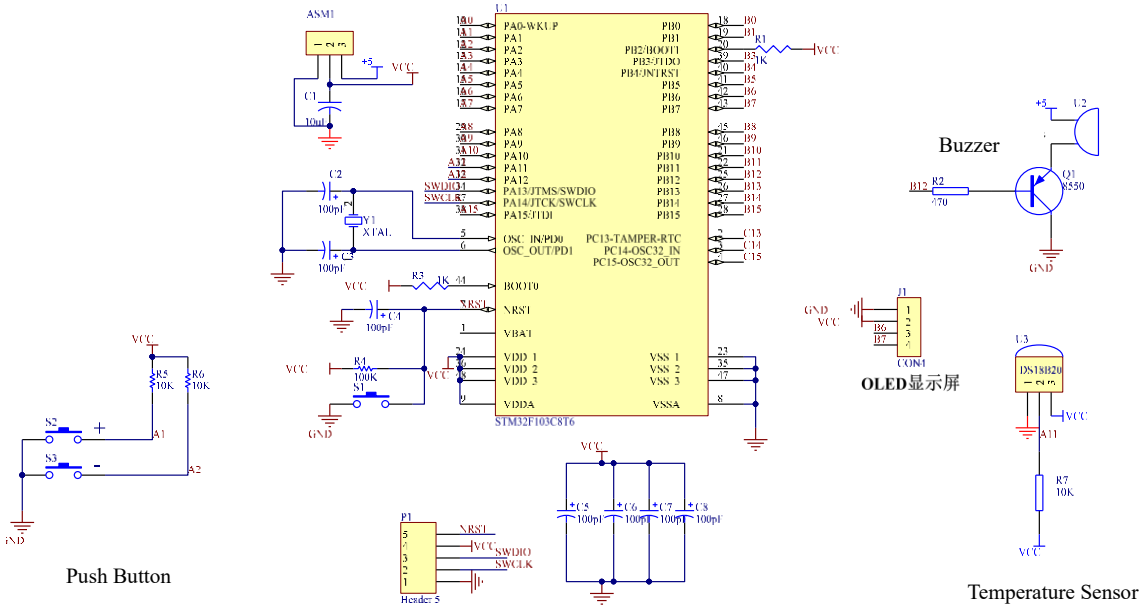


Figure 1. Schematic diagram of temperature detection and alarm system

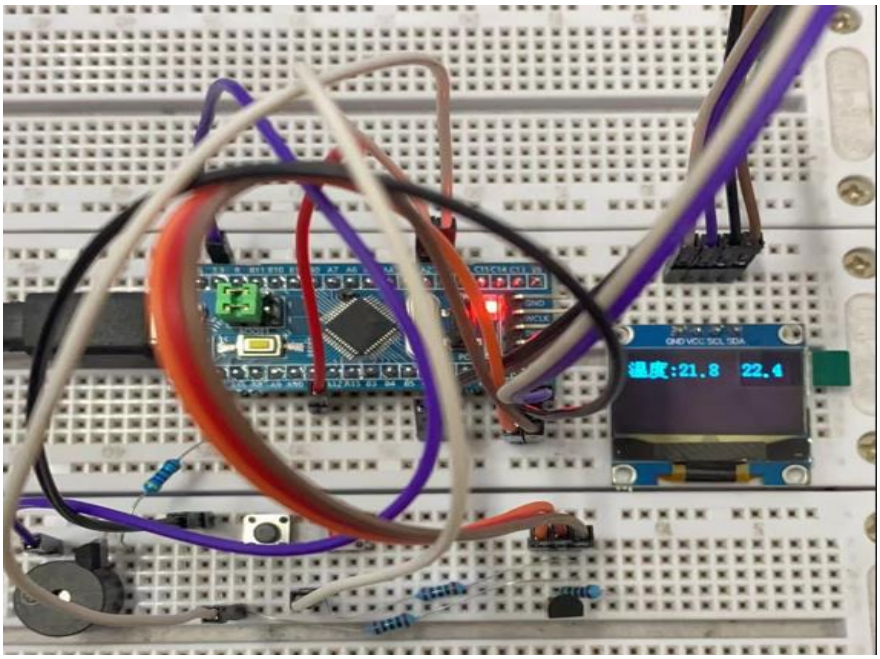


Figure 2. Physical connection of temperature detection and alarm system circuit

Components	Ports
Push Buttons	A1, A2
Buzzer	B12
OLED	B6, B7
DS18B20	A11

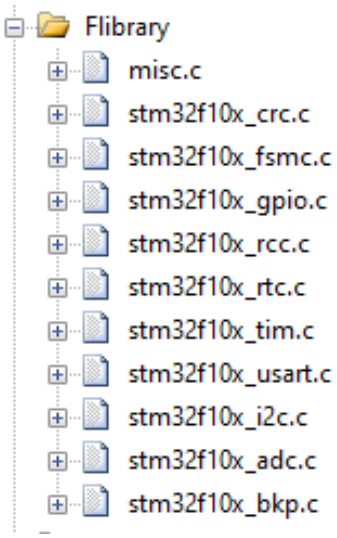
Table 1. Connection of components to the Blue Pill

3. Library

All library files and device call rulebooks come from the official standard library and the system auto generated. All the required source file can obtain form the STMicroelectronics official website https://www.st.com/content/st_com/zh.html.

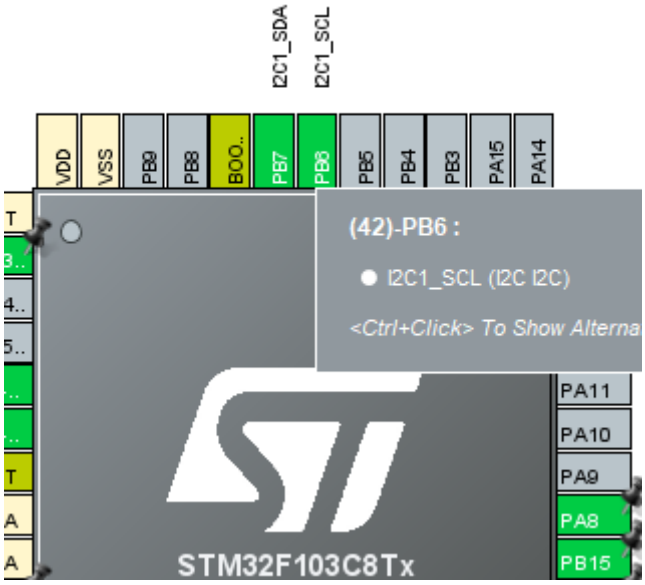
Some Required Library

- Initial library file



- OLED_I2C library

In this project, OLED is communicated through the I2C interface. The I2C is initialize in the Pinout & Configuration as shown in Figure below. OLED_I2C library is downloaded from the official website which already provided the functions that needed to display data. For details of all interface calls and functions, please refer to the supporting documents.




```
void IIC_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB, ENABLE );

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP ;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_SetBits(GPIOB,GPIO_Pin_6|GPIO_Pin_7);
}

void IIC_Start(void)
{
    SDA_OUT();
    IIC_SDA=1;
    IIC_SCL=1;
    delay_us(4);
    IIC_SDA=0;
    delay_us(4);
    IIC_SCL=0;
}

void IIC_Stop(void)
{
    SDA_OUT();
    IIC_SCL=0;
    IIC_SDA=0;
    delay_us(4);
    IIC_SCL=1;
    IIC_SDA=1;
    delay_us(4);
}
```

```
void OLED_Init(void)
{
    delay_ms(100); //The delay here is very important
     OLED_WR_Byte(0xAE,OLED_CMD);//display off
    OLED_WR_Byte(0x00,OLED_CMD);//Set the low column address
    OLED_WR_Byte(0x10,OLED_CMD);//Set the high column address
    OLED_WR_Byte(0x40,OLED_CMD);//Set the starting row address
    OLED_WR_Byte(0xB0,OLED_CMD);//page address setting
    OLED_WR_Byte(0x81,OLED_CMD);//Set the contrast control register
    OLED_WR_Byte(0xFF,OLED_CMD);//Set SEG output current brightness (128)
    OLED_WR_Byte(0xA1,OLED_CMD);//setting part remapping 0xa0 left and right reversed 0xa1 normal
    OLED_WR_Byte(0xA6,OLED_CMD);//Set normal display normal/reverse
    OLED_WR_Byte(0xA8,OLED_CMD);//Set multi-channel ratio (1 ~ 64)
    OLED_WR_Byte(0x3F,OLED_CMD);//1/32 duty
    OLED_WR_Byte(0xC8,OLED_CMD);//Set row scanning direction 0xc0 reverse up and down 0xc8 normal
    OLED_WR_Byte(0xD3,OLED_CMD);//Set display offset
    OLED_WR_Byte(0x00,OLED_CMD);//Set the low column address
    OLED_WR_Byte(0xD5,OLED_CMD);//Set display clock ratio/oscillator frequency
    OLED_WR_Byte(0x80,OLED_CMD);//Set the frequency division ratio, set the clock to 100 frames per sec
    OLED_WR_Byte(0xD8,OLED_CMD);//set area color mode off
    OLED_WR_Byte(0x05,OLED_CMD);//
    OLED_WR_Byte(0xD9,OLED_CMD);//Set pre-charge period
    OLED_WR_Byte(0xF1,OLED_CMD);//Set 15 clocks for pre-charging and 1 clock for discharging
    OLED_WR_Byte(0xDA,OLED_CMD);//Set com pin hardware configuration
    OLED_WR_Byte(0x12,OLED_CMD);//
    OLED_WR_Byte(0xDB,OLED_CMD);//Set Vcomh
    OLED_WR_Byte(0x30,OLED_CMD);//
    OLED_WR_Byte(0x8D,OLED_CMD);//Set charge pump enable
    OLED_WR_Byte(0x14,OLED_CMD);//set(0x10) disabled
    OLED_WR_Byte(0xAF,OLED_CMD);//Open the oled panel
}
```

- DS18B20 library

Temperature sensor (DS18B20) library is included in this project to initialize, read and manipulate the result. The Analog Digital Convertor is used in this library to read the temperature from temperature sensor.

Initialize DS18B20

```
u8 DS18B20_Init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(DS18B20_PORT_RCC,ENABLE);

    GPIO_PinRemapConfig(GPIO_Remap_SWJ_Disable,ENABLE);

    GPIO_InitStructure.GPIO_Pin=DS18B20_PIN;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
    GPIO_Init(DS18B20_PORT,&GPIO_InitStructure);

    DS18B20_Reset();
    return DS18B20_Check();
}
```

Read DS18B20 sensor and calculate temperature based on the reading

```
float DS18B20_GetTemperture(void)
{
    u16 temp;
    u8 a,b;
    float value;

    DS18B20_Start();                    // ds1820 start convert
    DS18B20_Reset();
    DS18B20_Check();
    DS18B20_Write_Byte(0xcc);// skip rom
    DS18B20_Write_Byte(0xbe);// convert
    a=DS18B20_Read_Byte(); // LSB
    b=DS18B20_Read_Byte(); // MSB
    temp=b;
    temp=(temp<<8)+a;

    if((temp&0xf800)==0xf800)
    {
        temp=(~temp)+1;
        value=temp*(-0.0625);
    }
    else
    {
        value=temp*0.0625;
    }
    return value;
}
```

4. Overall Code

- Include all the system file needed.

```
#include "stm32f10x.h"
#include "main.h"
#include "delay.h"
#include "OLED_I2C.h"
#include "ds18b20.h"
```

- The **KEY_GPIO_Config** function

Initialization of Button and buzzer

```
void KEY_GPIO_Config(void)//Button and buzzer initialization IO port
{
    GPIO_InitTypeDef GPIO_InitStructure;//Configuration button IO
    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOB,ENABLE);
    /*Turn on the LED-related GPIO peripheral clock*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1|GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;//
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD ;   //
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_WriteBit(GPIOB,GPIO_Pin_12,Bit_SET);//1
}
```

- The **uint8_t Key_Scan2** function key scan

Adjust the threshold temperature by pressing the button, one for increase and another one for decrease.

```
uint8_t Key_Scan2(GPIO_TypeDef* GPIOx,uint16_t GPIO_Pin)//key scan
{
    if(GPIO_ReadInputDataBit(GPIOx, GPIO_Pin)==0)
    {
        delay_us(2);
        return 0;
    }
    else return 1;
}
```

```
if(Key_Scan2(GPIOA,GPIO_Pin_1)==0)
{
    wd=wd+1;
}
if(Key_Scan2(GPIOA,GPIO_Pin_2)==0)
{
    wd=wd-1;
}
```

- Setting and alarm function

Set the initial threshold temperature to 30℃. If the temperature detected is greater than threshold temperature, the alarm will ring to notice the user.

Temperature = DS18B20 GetTemperture()*10

```
short temperature;
unit_16 wd=300;
```

```
if(wd<temperature) GPIO_WriteBit(GPIOB,GPIO_Pin_12,Bit_RESET);//0 alarm
else GPIO_WriteBit(GPIOB,GPIO_Pin_12,Bit_SET);//1ªstop alarm
```

- OLED screen initialization and display

```
IIC_Init();                                          //Initialize IIC

OLED_Init();                                         //Initialize the OLED
OLED_Clear();                                        //clear screen function
    OLED_ShowCHinese(0,0,0);
    OLED_ShowCHinese(16,0,1);
    OLED_ShowChar(32,0,':',16);
```

```
 OLED_ShowNum(40,0,temperature/100,1,16);//display temperature
 OLED_ShowNum(48,0,temperature%100/10,1,16);
 OLED_ShowChar(56,1,'.',0);
 OLED_ShowNum(64,0,temperature%10,1,16);
 OLED_ShowNum(90,0,wd/100,1,16);//Display alarm value
 OLED_ShowNum(98,0,wd%100/10,1,16);
 OLED_ShowChar(106,1,'.',0);
 OLED_ShowNum(114,0,wd%10,1,16);
```

5) Result

In conclusion, all the function is performed as we expected. The alarm will sound up when the detected temperature is higher than the threshold voltage. We are able to can change the threshold temperature through push buttons or program. The detected temperature and threshold temperature will display on the OLED screen through I2C interface. The demonstration video shows the result of whole project.

6) References

- https://www.st.com/content/st_com/zh.html
- https://controllerstech.com/oled-display-using-i2c-stm32/
- https://deepbluembedded.com/stm32-lm35-temperature-sensor-example-lm35-with-stm32-adc/