嵌入式系统设计实验报告

实验：	实验 3 I2C 与 OLED 屏显示
专业：	计算机科学与技术
班级：	1 班
姓名：	姚怀聿
学号：	22920202204632

2022 年 11 月 28 日

目　　录

一、实验目的

1、 掌握 I2C 总线通讯的基本原理和使用 I2C 读写 EEPROM 的方法。

2、 掌握使用 OLED 显示屏显示数据的方法。

二、实验方案设计

1. 基础实验

实验要求:

定义 Tab=“日期+学号后三位+姓名英文首字母”，使用 12C 将 Tab 写入 EEPROM。将 EEPROM 中的内容读出并存放在 ReadE2P 中。将 Tab 的内容与 ReadE2P 的内容同时显示在 LCD 屏上,若二者内容相同则显示"=", LED1 灯亮，若不相等显示"≠", LED2 灯亮。

OLED 显示示意图如图 3.6 所示。

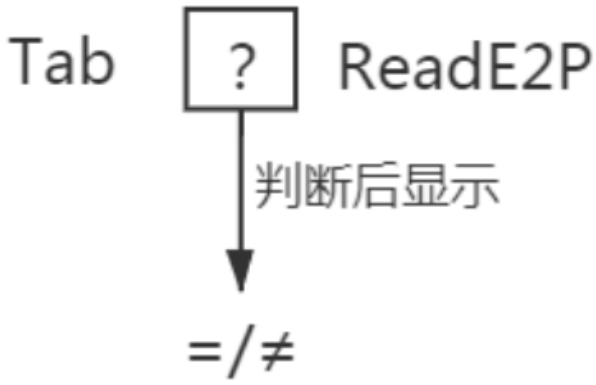


图 3.6 OLED 显示示意

2. 进阶实验

实验要求:

自己制作 bmp 格式的图片，在 OLED 中显示图片，在 EEPROM 中存数字，比如 00、01、02、03，每三秒显示对应数字的图片，显示三秒。

3. 扩展实验

分五次将五个三个字母的组合写入 EEPROM。随机读出一个三字母组合并将其显示在 OLED 屏的左方，再随机读出一个三字母组合显示在 OLED 屏右方。若二者相等，则 OLED 屏下方显示 Bingo，词语消失，同时 EEPROM 中删去该词。若二者不等，则 OLED 屏下方显示 False，同时右边的词语消失，程序再次从 EEPROM 中读出一个词语进行配对。所有词语配对成功后清空 OLED 和 EEPROM，而后屏上显示“Congratulations！”

整体实验方案设计如下:

为了将所有程序能整合在一个.c 文件里，我主要编写了一个 test.c 函数，里面编写了基础实验、进阶实验和扩展实验的代码；在 main 函数中，调用 test1()、test2()、test3()分别实现不同的功能，其对应关系如下表:

函数	功能
test1()	基础实验
test2()	进阶实验
test3()	扩展实验

三、实验过程

首先，对于所有的实验，我们都需要进行一些初始化的设置，这个函数在 main.c 中调用:

```
1   #include "test.h"
2
3   int main(void) {
4       systemInit();
5
6       while(1) {
7           //test1();
8           //test2();
9           test3();
10      }
11  }
12
```

```
11  /* 系统初始化 */
12  void systemInit(void) {
13      systick_config();
14      gd_XII_systeminit();
15      OLED_Gpio_Init();
16      OLED_Init();
17      i2c_gpio_config();
18      i2c_config();
19      i2c_eeprom_init();
20      usart0Init(EVAL_COM0, 115200U);
21  }
```

第 17、18 行的两个参数给的代码框架中并没有写，参考老师发的课件需要在函数中补充:

```c
void i2c_gpio_config(void) {
    rcu_periph_clock_enable(RCU_GPIOB);
    rcu_periph_clock_enable(RCU_I2C1);
    gpio_af_set(GPIOB, GPIO_AF_4, GPIO_PIN_10);
    gpio_af_set(GPIOB, GPIO_AF_4, GPIO_PIN_11);
    gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_10);
    gpio_output_options_set(GPIOB, GPIO_OTYPE_OD, GPIO_OSPEED_50MHZ, GPIO_PIN_10);
    gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_11);
    gpio_output_options_set(GPIOB, GPIO_OTYPE_OD, GPIO_OSPEED_50MHZ, GPIO_PIN_11);
}
```

```c
void i2c_config(void) {
    rcu_periph_clock_enable(RCU_I2C1);
    i2c_clock_config(I2C1,I2C1_SPEED,I2C_DTCY_2);
    i2c_mode_addr_config(I2C1,I2C_I2CMODE_ENABLE,I2C_ADDFORMAT_7BITS,I2C1_SLAVE_ADDRESS7);
    i2c_enable(I2C1);
    i2c_ack_config(I2C1,I2C_ACK_ENABLE);
}
```

1. 基础实验

在 test.c 文件中编写 test1()函数

首先在全局定义字符数组变量 tab、BUFFER_SIZE 是要往 EEPROAM 中写的大小、ReadE2P 是从 EEPROM 中读出后数据存放的位置，还需要在 test.h 中宏定义 EEPROM 的起始位置:

```c
#ifndef _TEST_H
#define _TEST_H

#include "gd32f4xx.h"

#define EEPROM_FIRST_PAGE           0x00            //EEPROM起始地址

void systemInit();
void test1();
void test2();
void test3();
uint8_t i2c_24c02_test();
#endif
```

```c
uint8_t tab[] = "2022.11.26+yao";                   //定义日期和姓名缩写
const uint8_t BUFFER_SIZE = sizeof(tab) / sizeof(uint8_t);
uint8_t ReadE2P[BUFFER_SIZE];
uint8_t res1 = '=';
```

res1 用于显示相等时的等号。

首先将 tab 中的数据写入 EEPROM，然后从 EEPROM 中读出并将读出的数据放入 ReadE2P 中。之后将写入和读出的数据分别显示在 OLED 的第一行和第三行。最后用一个 for 循环判断每一个字符是否相等，若不相等就返回 0，否则返回 1。

```
uint8_t i2c_24c02_test(void) {
  uint8_t i;
  eeprom_buffer_write(tab, EEPROM_FIRST_PAGE, BUFFER_SIZE); // 写进EEPROAM
  eeprom_buffer_read(ReadE2P, EEPROM_FIRST_PAGE, BUFFER_SIZE); // 从EEPROAM中读出
  OLED_ShowString(0, 0, tab);
  OLED_ShowString(0, 4, ReadE2P);
  for(i = 0; i < BUFFER_SIZE; i++)
    if(tab[i] != ReadE2P[i])
      return 0;
  return 1;
}
```

在 test1()函数中调用 i2c_24c02_test()函数判断写入和读出的数据是否相等，若相等则点亮 LED1 灯，并在 OLED 第二行显示"="；否则，点亮 LED2 灯，并在 OLED 第二行显示"≠"。

之后进入一个死循环，让该显示的东西一直显示。

```
/*基础实验*/
void test1(void) {
  OLED_Clear();
  while(1) {
    if (i2c_24c02_test()) {
      gd_eval_led_on(LED1);
      OLED_ShowChar(0, 2, res1);     // 显示=
      Uart0Printf("= \r\n");
    }
    else {
      gd_eval_led_on(LED2);
      OLED_ShowCHinese(0, 2, 0);     // 显示≠
      Uart0Printf("≠ \r\n");
    }
    while(1){ }
  }
}
```

这里需要先用取模软件获得"≠"的字模，放在 oledfont.h 文件中:

```
char Hzk[][32]={

{0x00,0x20,0x20,0x20,0x20,0x20,0x20,0xE0,0x20,0x38,0x26,0x20,0x20,0x20,0x00,0x00},
{0x00,0x02,0x02,0x62,0x1A,0x06,0x03,0x02,0x02,0x02,0x02,0x02,0x02,0x02,0x00,0x00},/*"≠",0*/
```

2. 进阶实验

使用一个 while 循环显示提前准备好的 BMP 图片即可，每显示一张图片延迟 3 秒:

```
/*进阶实验*/
void test2() {
  uint8_t arr[] = {0x00, 0x01, 0x02, 0x03};
  const uint8_t ARR_SIZE = sizeof(arr) / sizeof(uint8_t);
  eeprom_buffer_write(arr, EEPROM_FIRST_PAGE, ARR_SIZE); // 往EEPRAOM中存数字
  while(1) {
    OLED_DrawBMP(0, 0, 127, 7, BMP1);
    delay_1ms(3000);
    OLED_Clear();
    OLED_DrawBMP(0, 0, 127, 7, BMP2);
    delay_1ms(3000);
    OLED_Clear();
    OLED_DrawBMP(0, 0, 127, 7, img1);
    delay_1ms(3000);
    OLED_Clear();
    OLED_DrawBMP(0, 0, 127, 7, img2);
    delay_1ms(3000);
    OLED_Clear();
  }
}
```

3. 扩展实验

首先，在 test3()函数中定义需要用到的字符串:

```
uint8_t chars[5][3] = {"abc", "bcd", "cde", "def", "efg"};
```

在这里定义了 5 个需要用到的字符串，分五次写入 EEPROM:

```
  for(i = 0; i < 5; i++) { // 分五次写入
    eeprom_buffer_write(chars[i], EEPROM_FIRST_PAGE + i * 3, 3);
  }
```

注意每次需要至少间隔 3 个去写，否则会覆盖掉之前写入的。

定义一个变量 cnt 用于记录当前还有几个字符串没有被匹配到，其初始值为 5，表示一开始都没有被匹配到。

```
95 |   uint8_t cnt = 5;
```

当 cnt 不为 0 的时候，进入 while 循环。定义两个 uint8_t 类型的变量 left 和 right 分别表示左右两边要显示的字符串在原五个字符串数组中的下标，使用 rand()函数得到一个随机正整数后对 5 取模，取模后的值落在 0~4 之间，于是便实现了“随机取一个字符串放在左边，再随机取一个字符串放在右边”：

```
108  while(cnt) {
109    left = rand() % 5;
110    eeprom_buffer_read(left_string, EEPROM_FIRST_PAGE + left * 3, 3);
111    if(left_string[0] == 0) continue;
112    left_string[3] = 0;
113    OLED_ShowString(0, 0, left_string); // 在左边写入
114    for(i = 0; i < 5; i++) {
115      right = rand() % 5;
116      eeprom_buffer_read(right_string, EEPROM_FIRST_PAGE + right * 3, 3); // 随便选取一个
117      if(right_string[0] == 0) continue;
118      right_string[3] = 0;
119      OLED_ShowString(104, 0, right_string);
120      if(is_match(left_string, right_string)) { // 如果匹配  直接退出for循环
121        cnt--;
122        eeprom_buffer_write(zero, EEPROM_FIRST_PAGE + left * 3, 3);
123        OLED_ShowString(0, 6, bingo_string);
124        delay_1ms(1000);
125        break;
126      } else { // 如果不匹配
127        OLED_ShowString(0, 6, false_string);
128        delay_1ms(1000);
129      }
130    }
131  }
```

left_string 和 right_string 分别存储要放在左边显示和放在右边显示的字符串。

如果两个字符串相等，则在下方显示"Bingo!"，并将这个字符串从 EEPROM 中删除。删除这里的实现，我将需要删除的那个位置的数据改为‘\0’；如果两个字符串不匹配，就重新从 EEPROM 中取出一个还没有匹配过的字符串放在右边继续比较。代码实现如下:

```
114     for(i = 0; i < 5; i++) {
115       right = rand() % 5;
116       eeprom_buffer_read(right_string, EEPROM_FIRST_PAGE + right * 3, 3); // 随便选取一个
117       if(right_string[0] == 0) continue;
118       right_string[3] = 0;
119       OLED_ShowString(104, 0, right_string);
120       if(is_match(left_string, right_string)) { // 如果匹配  直接退出for循环
121         cnt--;
122         eeprom_buffer_write(zero, EEPROM_FIRST_PAGE + left * 3, 3);
123         OLED_ShowString(0, 6, bingo_string);
124         delay_1ms(1000);
125         break;
126       } else { // 如果不匹配
127         OLED_ShowString(0, 6, false_string);
128         delay_1ms(1000);
129       }
130     }
```

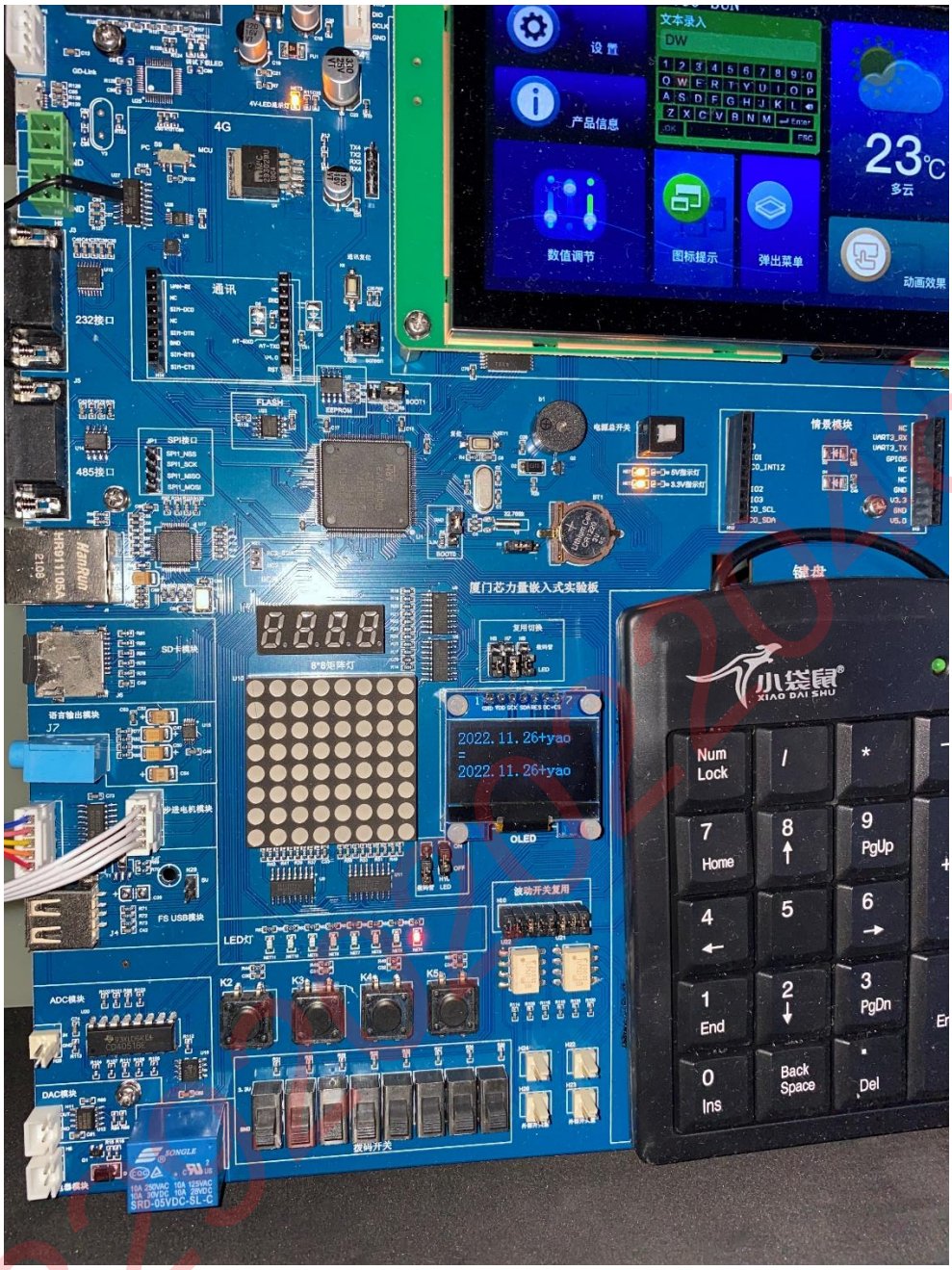
这里第 120 行的 is_match()是我实现的一个很简单的字符串比较的函数:

```
80  uint8_t is_match(uint8_t a[], uint8_t b[]) { // 判断两字符串是否相等
81    uint8_t i;
82    for(i = 0; i < 3; i++) {
83      if(a[i] != b[i]) {
84        return 0;
85      }
86    }
87    return 1;
88  }
```

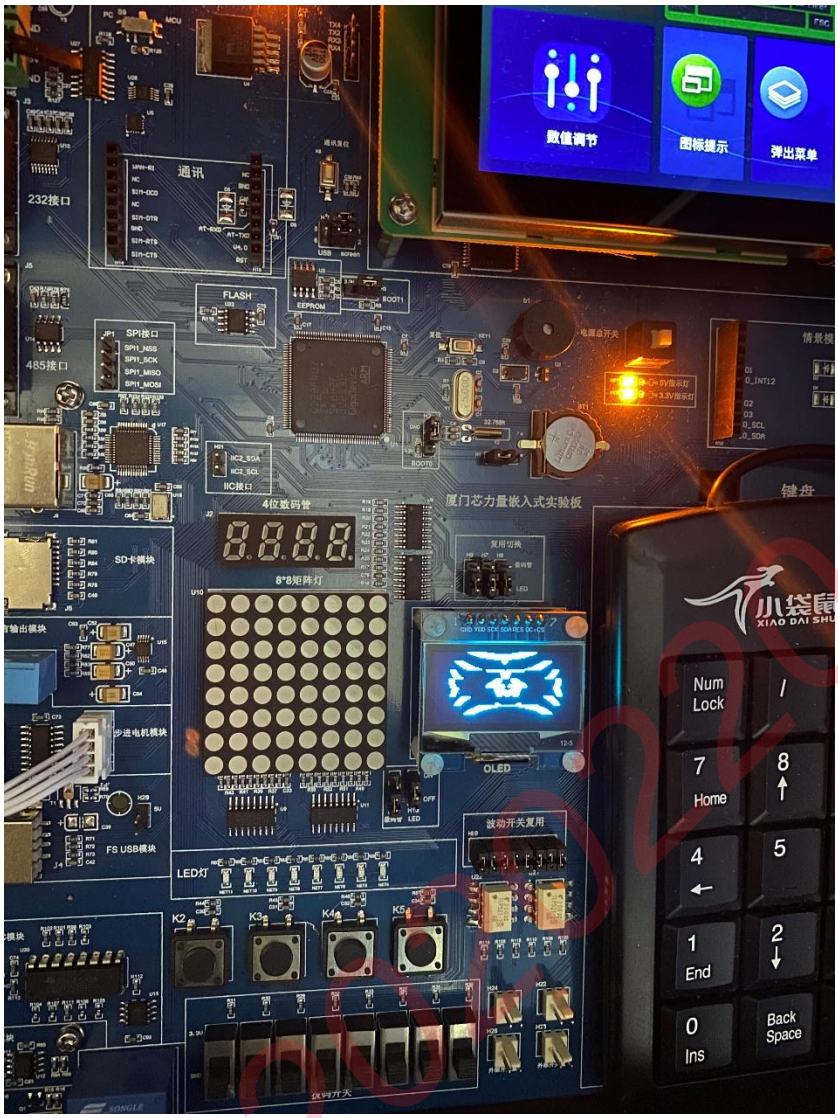
因为字符串个数都是 3，所以只需要判断三次，如果都相等返回 1，否则返回 0。

四、实验结果

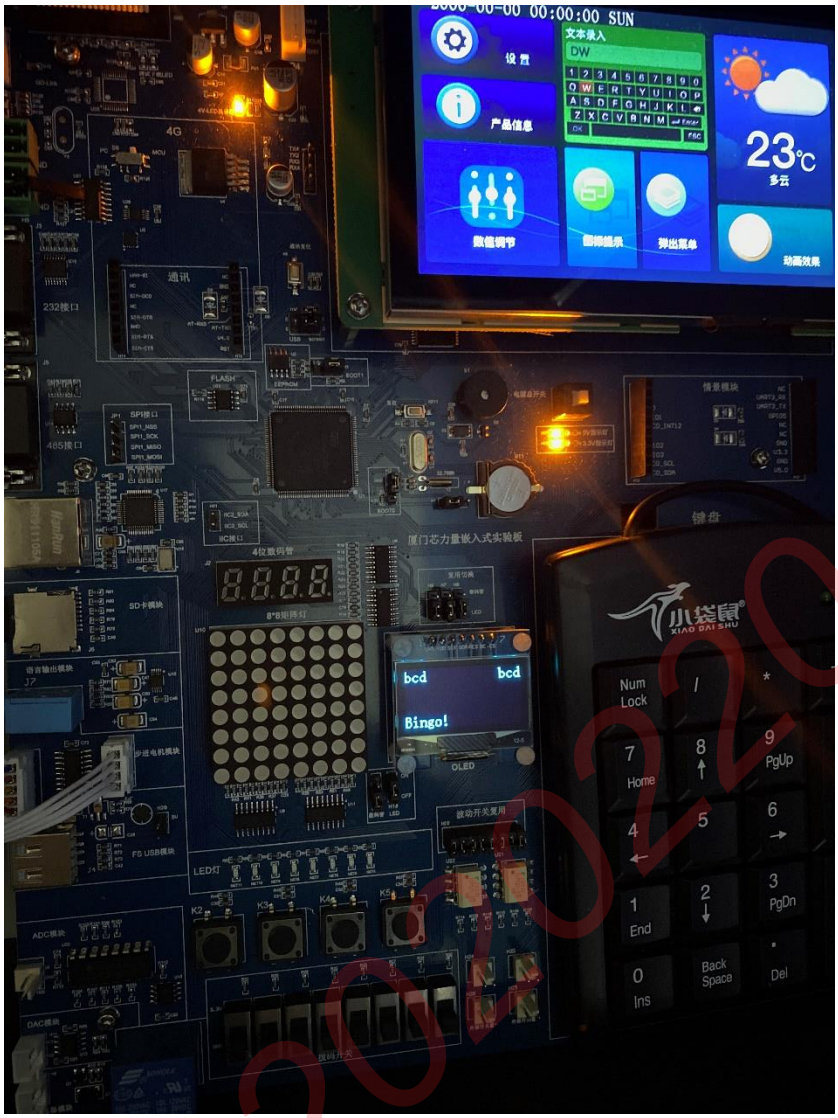
1. 基础实验


设置
产品信息
数值调节
图标提示
弹出菜单
动画效果
23℃
多云
文本录入
DW
4G
通讯
232接口
485接口
SPI接口
FLASH
EEPROM
情景模块
厦门芯力量嵌入式实验板
8*8矩阵灯
OLED
2022.11.26+yao
2022.11.26+yao
LED灯
ADC模块
DAC模块
SD卡模块
FS USB模块
拨码开关
键盘
小袋鼠
XIAO DAI SHU
Num Lock
Home
End
Ins
Back Space
PgUp
PgDn
Del

2. 进阶实验



3. 扩展实验



五、实验分析

本次实验的扩展实验一开始没有弄懂实验书上要表达的是什么意思，后来只能按照自己理解的意思去做扩展实验，结果效果还不错，大体上能实现实验书上说的那个效果。对于基础实验，一开始我写了很长时间，但是就是没办法从 EEPROM 中读出数据显示到 OLED 屏幕上。后来才查看课件才发现原来是没有初始化。对于这种硬件编程来说，初始化

是很重要的，它往往决定了你的实验能否成功。

六、心得体会

本次实验主要是使用 I2C 以及 OLED。以前写 C 语言程序都喜欢看到程序的结果在终端上输出出来，现在无法看到程序结果在终端上输出了，反而显示在 OLED 屏幕上，多少有点不方便。我目前还不会调试程序，如果程序没有运行成功，我经常需要花很长的时间去找 BUG，这让我感到很无力，希望以后能学会调试硬件程序吧。