



嵌入式系统设计实验报告

实验:	综合实验 智慧农业
专业:	计算机科学与技术
班级:	1 班
姓名:	姚怀聿
学号:	22920202204632

2022 年 3 月 5 日

目 录

一、	实验目的	3
二、	实验方案设计	3
1.	基础实验	3
2.	进阶实验	3
三、	实验过程	5
3.1	实验步骤.....	5
3.2	初始化配置	7
3.3	基础实验	8
3.4	进阶实验	11
3.5	综合性实验要求	14
四、	实验结果	17
五、	心得体会	19

一、实验目的

1. 模拟智慧农业场景，利用温湿度、光照值和实验箱上各模块模拟实际场景。
2. 熟悉实验箱中 E53_IA1 扩展版的功能与基本使用方法，掌握基于 E53_IA1 的智慧农业场景下的编程与应用。

二、实验方案设计

1. 基础实验

实验要求:

直观显示温湿度、光照值。

2. 进阶实验

实验要求:

当光照值小于（或大于）设定阈值时，自动进行灯光控制操作；当湿度小于（或大于）设定阈值时，自动进行某一执行部件的动作。

配置 E53_IA1 模块使其能够：

(1) 实时测量温度、湿度和光照强度；

(2) 当温度、湿度超过（或小于）设定阈值时，开启电机模拟通风、加湿；当温度、湿度恢复至正常区间后关闭电机。

2、在 1 的基础上，将传感器采集到的当前温湿度、光照强度值上传到云端，实验者可以通过云端实时监控当前的环境信息。

3. 综合性实验要求

在基础实验的基础上，设计完成不同的结果展示方法。可参考的结果展示方法如下：

- 使用触摸屏显示实时温度或湿度检测的图表，图表能对超出阈值的情况作出表示。
- 在温度/湿度超出阈值后，响起 E53_IA1 上的蜂鸣器作为警报，警报声为三声长鸣；此外，通过 PWM 控制电机转动模拟通风系统开启，在电机转动后响起 MCU 上的蜂鸣器作为开启提示声，提示声为三声短鸣；若温度/湿度回归正常范围，电机关闭，同时 MCU 上的蜂鸣器长响一声作为关闭提示。
- 发生温度/湿度超出阈值、电机受控制关闭或开启的情况，矩阵 LED 分别显示“高 (T/M)”“开”“关”；同时 LED 灯根据不同情况进行长度不一的闪烁。

三、实验过程

3.1 实验步骤

步骤一：智慧农业模块 E53_IA1 的光照传感器 BH1750，温湿度传感器 SHT30 与 MCU 之间通过 I2C 通信，在 I2C.c 文件中将 PB6 与 PB7 配置为 I2C 的时钟线与数据线，编写 I2C 初始化函数。

步骤二：智慧农业模块 E53_IA1 的电机模块与 MCU 的 PB8 相连，模组内设置好了电机驱动电路，在 E53_SC1.c 文件中将 PB8 设置为 GPIO 上拉输出模式，实验者也可以将实验八 PWM 实验的程序移植至本实验程序使用电机。

步骤三：调用传感器数据处理函数，将温湿度、光照强度数值读出通过串口输出，实验者自定温湿度阈值，当温湿度超过（或小于）设定阈值时，开启电机模拟通风、加湿；当温度、湿度恢复至正常区间后关闭电机。

步骤四：调用 4G 上传函数，将光照信息值、温湿度值上传至云端服务器。

步骤五：下载程序，实验者可通过遮光观察当前光照强度的变化，对传感器哈气改变温湿度观察电机动作。

步骤六：登录云端监测当前光照强度值和温湿度值。

浏览器输入信息系统链接，进入信息系统登录页面，如图 11.1：



图 11.1 登录页面

如图 11.2：输入账号、密码以及验证码，点击登录，进入学生系统首页。

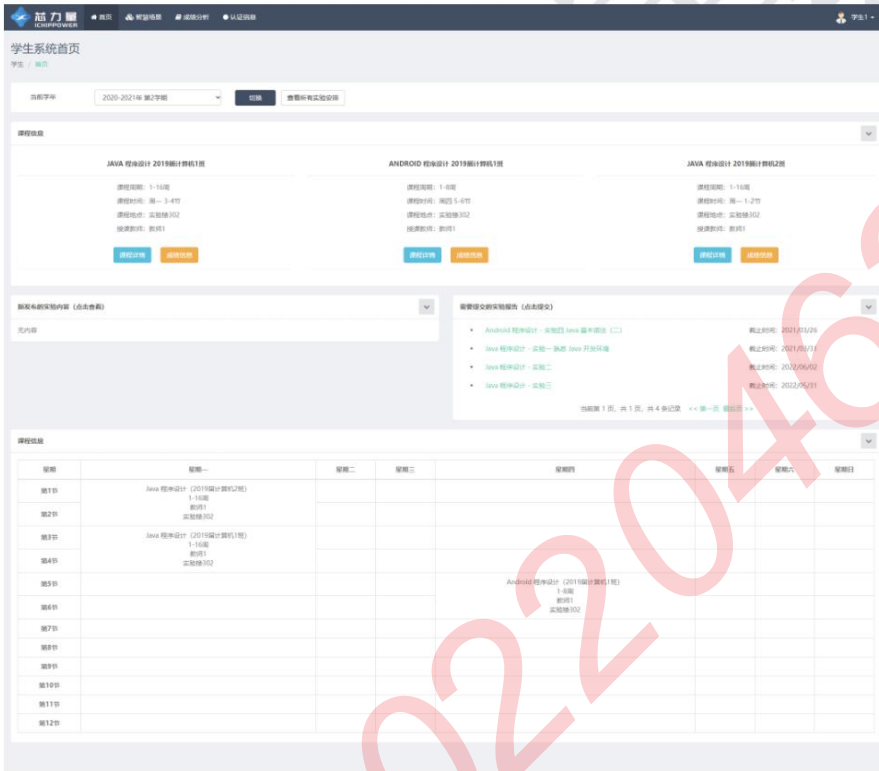
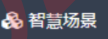


图 11.2 学生系统首页

智慧场景实验，点击页面顶部导航栏  按钮，可以进入智慧场景实验数据获取页面，如下图：

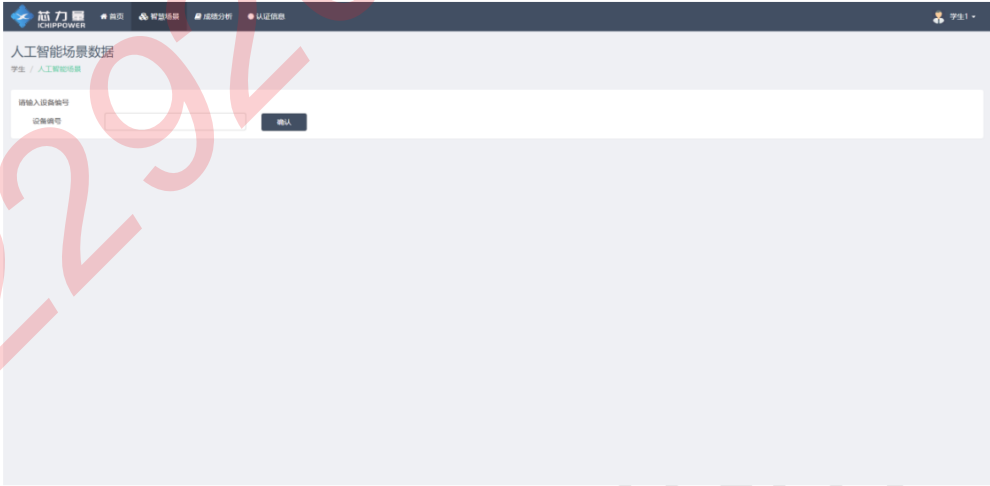


图 11.3 智慧场景实验数据获取页面

输入实验设备编号后，点击确认，会弹出模态框提示正在获取数据，如下图：

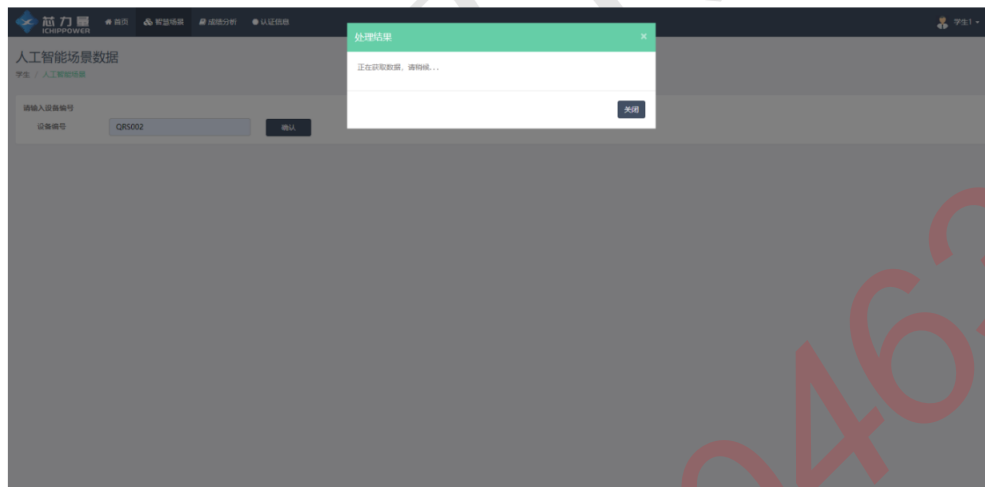


图 11.4 正在获取实验数据

获取到实验数据后，页面会显示相关数据结果，实验者通过云端监测当前光照强度值和温湿度值，如下图：

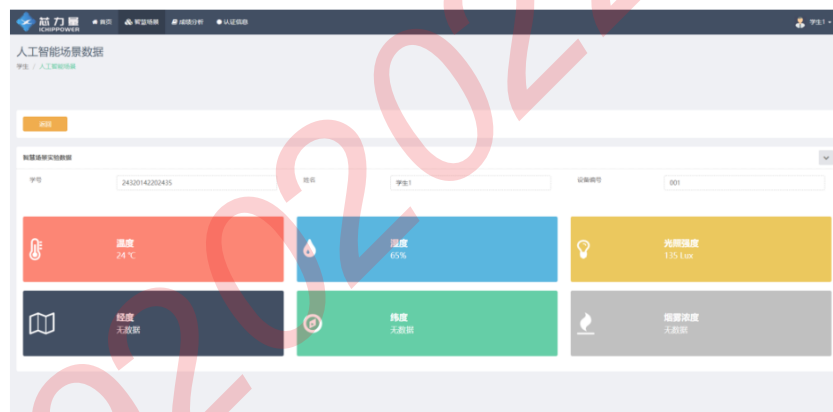


图 11.5 智慧场景实验数据

3.2 初始化配置

进行本次实验，我们需要用到的模块有 OLED 模块、E53_IA1 智慧农业模块、Usart 串口通信模块、PWM 步进电机模块。同时，为了显示功能，我们还需要用到 LED、拨码开关等。所以首先将初始化函数编写如下：

```

91 //初始化 系统时钟、XLL\拨码开关、OLED、智慧农业模块等按需编写
92 void system_init(void) {
93     systick_config(); // 初始化系统时钟
94     gd_XII_systeminit(); // 初始化8个LED灯
95     dipinit(); // 初始化拨码开关
96     OLED_Gpio_Init();
97     OLED_Init();
98     i2c_gpio_config();
99     i2c_config();
100     usart0Init(EVAL_COM0 ,115200U);
101     Init_E53_IA1();
102     HC595_GPIO_Config();
103     motor_gpio_config(); // 步进电机初始化
104 }

```

并在 main 函数中调用该初始化函数:

```

18 int main(void) {
19     system_init();
20
21     while(1) {
22         test();
23     }
24 }

```

3.3 基础实验

基础实验主要是将湿度、温度、光照强度直观地显示出来，这里我们使用 OLED 模块来显示这三个数值。

首先，我们来分析一下 E53_IA1.c 文件中的部分代码:

函数 E53_IA1_Read_Data()可以读取传感器传来的湿度、温度、光照强度的数据，并将其存放在结构体 E53_IA1_Data_TypeDef 中:


```
E53_IA1.h  E53_IA1.c  main.c  test.c  gd32f4xx_rcu.c  gd32f4xx_XII-IOT.h

1  #include "gd32f4xx.h"
2  #include "gd32f4xx_XII-IOT.h"
3  #include "systick.h"
4  #include "test.h"
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include "user.h"
8  #include "E53_IA1.h"
9  #include "I2C.h"
10 #include "oled.h"
11 #include "74HC595_LED.h"
12
13 E53_IA1_Data_TypeDef E53_IA1_Data;
14

15 /* E53_IA1传感器数据类型定义 -----
16 typedef struct
17 {
18     float Lux;           //光照强度
19     float Humidity;      //湿度
20     float Temperature;   //温度
21 } E53_IA1_Data_TypeDef;
22
23 extern E53_IA1_Data_TypeDef E53_IA1_Data;
```

在 E53_IA1.h 文件中 extern E53_IA1_Data_TypeDef E53_IA1_Data 使得在 E53_IA1.c 中能够改变结构体中的数值。

```
375
376 tmp = SHT3x_CheckCrc(data, 2, data[2]);
377 if (!tmp) /* value is ture */
378 {
379     dat = ((uint16_t)data[0] << 8) | data[1];
380     E53_IA1_Data.Temperature = SHT3x_CalcTemperatureC(dat);
381 }
382
383 /* check humidity */
384 data[0] = SHT3X_Data_Buffer[3];
385 data[1] = SHT3X_Data_Buffer[4];
386 data[2] = SHT3X_Data_Buffer[5];
387
388 tmp = SHT3x_CheckCrc(data, 2, data[2]);
389 if (!tmp) /* value is ture */
390 {
391     dat = ((uint16_t)data[0] << 8) | data[1];
392     E53_IA1_Data.Humidity = SHT3x_CalcCRH(dat);
393 }
394 }
395
```

调用 E53_IA1_Read_Data()得到数值后，我们就可以实验

OLED 屏幕将数值直观地显示出来，同时调用 `Uart0Printf()` 函数将数据发送给串口。

在 `test()` 中，将温度、湿度和光照检测及显示通过在一个死循环中重复调用 `E51_IA1_Read_Data()`，将得到的温度、湿度、光照强度数值转换为字符串的形式，并在 OLED 屏幕上显示出来，具体代码如下：

```
103 void test(void) {
104     OLED_ShowString(0, 0, (uint8_t *) "Humd: ");
105     OLED_ShowString(0, 2, (uint8_t *) "Temp: ");
106     OLED_ShowString(0, 4, (uint8_t *) "Lux: ");
107     while(1) {
108         E53_IA1_Read_Data();
109
110         Uart0Printf("Humidity:%3.1f, Temperature:%3.1f°C, Lux: %3.1f \r\n",
111                     E53_IA1_Data.Humidity,
112                     E53_IA1_Data.Temperature,
113                     E53_IA1_Data.Lux);
114
115         sprintf((char *) humd, "%3.1f%%", E53_IA1_Data.Humidity);
116         sprintf((char *) temp, "%3.1f°C", E53_IA1_Data.Temperature);
117         sprintf((char *) lux, "%3.1f", E53_IA1_Data.Lux);
118
119         OLED_ShowString(42, 0, blank);
120         OLED_ShowString(42, 2, blank);
121         OLED_ShowString(42, 4, blank);
122         OLED_ShowString(42, 0, temp);
123         OLED_ShowString(42, 2, humd);
124         OLED_ShowString(42, 4, lux);
```

如果对数据的精度要求不高的话，实现到这一步我们就已经完成了基础实验部分。但是为了实验的严谨性，我对其进行了进一步实现：将三个数值都保留到一位小数并为相应的数值加上单位。实现到这里我发现了一个问题，只要数值稍微有一点点更新，OLED 屏幕就会更新，这样展现出来的效果非常不好，于是，我们新定义三个 `float` 类型的变量：

```
41 float humd_last, temp_last, lux_last;
```

它们分别记录 humidity、temperature、lux 在上一次循环中的值，如果这个数值与下一次读取得到的数值对比，改变得非常小，我们就近似认为它没有改变，并且不重新写到 OLED 屏幕上。这样的话，OLED 就不会一直刷新，展示效果得到很好的提升。这一部分的代码如下：

```
40 float eps = 0.1;

123
124 if(abs(humd_last - E53_IAl_Data.Humidity) > eps || abs(temp_last - E53_IAl_Data.Temperature) > eps
125 || abs(lux_last - E53_IAl_Data.Lux) > eps) {
126     OLED_ShowString(42, 0, blank);
127     OLED_ShowString(42, 2, blank);
128     OLED_ShowString(42, 4, blank);
129     OLED_ShowString(42, 0, humd);
130     OLED_ShowString(42, 2, temp);
131     OLED_ShowString(42, 4, lux);
132 }

192 // 保存上一次的数值
193 humd_last = E53_IAl_Data.Humidity;
194 temp_last = E53_IAl_Data.Temperature;
195 lux_last = E53_IAl_Data.Lux;
```

3.4 进阶实验

进阶实验的要求是通过设定光照阈值和湿度阈值来控制模块进行一些执行部件的操作，具体如下：

(1) 首先考虑对湿度的控制：

```
33 float humd_ub = 80.0;
34 float humd_lb = 20.0;
```

humd_ub(upper_bound)表示湿度上限；humd_lb(lower_bound)表示湿度下限。当湿度超过湿度上限时，打开电机，同时在 OLED 的对应位置显示'H'；当湿度低于湿度下限时，关闭电机，同时在 OLED 的对应位置显示'L'；当湿度处于两阈值之间时，关闭电机，同时在 OLED 的对应

位置显示'Y'；代码如下：

```
126      /*Humidity 操作*/
127      if(E53_IAl_Data.Humidity > humd_ub) {
128          E53_IAl_Motor_StatusSet(ON); // 湿度超过高阈值 打开电机
129          OLED_ShowString(80, 0, (uint8_t*)"H");
130      }
131      else if(E53_IAl_Data.Lux > lux_ub) {
132          E53_IAl_Motor_StatusSet(OFF); // 湿度低于低阈值 关闭电机
133          OLED_ShowString(80, 0, (uint8_t*)"L");
134      }
135      else { // 处于两阈值之间
136          E53_IAl_Motor_StatusSet(OFF);
137          OLED_ShowString(80, 0, (uint8_t*)"Y"); // 表示正常
138      }
139      /*Humidity 操作结束*/
```

(2) 然后考虑对温度的控制：

```
35 float temp_ub = 30.0;
36 float temp_lb = 15.0; temp_ub(upper_bound)表示温度上限；
```

lb(lower_bound)表示温度下限。当温度超过温度上限时，在 OLED 的对应位置显示'H'；当温度低于温度下限时，在 OLED 的对应位置显示'L'；当湿度处于两阈值之间时，在 OLED 的对应位置显示'Y'；代码如下：

```
150      /*Temperature 操作*/
151      if(E53_IAl_Data.Temperature > temp_ub) {
152          OLED_ShowString(112, 2, (uint8_t*)"H");
153      }
154      else if(E53_IAl_Data.Temperature < temp_lb) {
155          OLED_ShowString(112, 2, (uint8_t*)"L");
156      }
157      else { // 处于两阈值之间
158          OLED_ShowString(112, 2, (uint8_t*)"Y"); // 表示正常
159      }
160      /*Temperature 操作结束*/
```

(3) 最后考虑对光照强度的控制：

lux_ub(upper_bound) 表示光照强度上限；
humd_lb(lower_bound)表示光照强度下限。当光照强度超过

光照强度上限时,打开补光灯和所有的 LED 灯(LED1~LED8),
同时在 OLED 的相应位置显示'H';当光照强度低于光照强度下限时,补光灯和所有 LED 灯,同时在 OLED 的相应位置显示'L';当光照强度处于两阈值之间时,关闭电机,同时在 OLED 的对应位置显示'Y';代码如下:

```
162      /*Lux 操作*/
163      if(E53_IA1_Data.Lux < lux_lb) { // 如果光照小于最低阈值 将补光灯开启 并将所有LED灯打开
164          E53_IA1_Light_StatusSet(ON);
165          all_led_on;
166          OLED_ShowString(112, 4, (uint8_t*)"L");
167      }
168      else if(E53_IA1_Data.Lux > lux_ub) {
169          E53_IA1_Light_StatusSet(OFF);
170          all_led_off;
171          OLED_ShowString(112, 4, (uint8_t*)"H");
172      }
173      else { // 处于两阈值之间
174          E53_IA1_Light_StatusSet(OFF);
175          all_led_off;
176          OLED_ShowString(112, 4, (uint8_t*)"Y"); // 表示正常
177      }
178      /*Lux 操作结束*/
```

关于补光灯的初始化操作已经在函数 Init_E53_IA1()中做过了:

```
333 void Init_E53_IA1(void)
334 {
335     /* configure GPIO */
336     i2c_gpio_config();
337     /* configure I2C */
338     i2c_config();
339     /* init BHL750 */
340     Init_BHL750();
341     /* init SHT30 */
342     Init_SHT30();
343     /* init Motor */
344     Init_Motor();
345     /* init Light*/
346     Init_Light();
347 }
```


3.5 综合性实验要求

在基础实验的基础上，设计完成不同的结果展示方法。可参考的结果展示方法如下：

- 使用触摸屏显示实时温度或湿度检测的图表，图表能对超出阈值的情况作出表示。
- 在温度/湿度超出阈值后，响起 E53_IA1 上的蜂鸣器作为警报，警报声为三声长鸣；此外，通过 PWM 控制电机转动模拟通风系统开启，在电机转动后响起 MCU 上的蜂鸣器作为开启提示声，提示声为三声短鸣；若温度/湿度回归正常范围，电机关闭，同时 MCU 上的蜂鸣器长响一声作为关闭提示。
- 发生温度/湿度超出阈值、电机受控制关闭或开启的情况，矩阵 LED 分别显示“高 (T/M)” “开” “关”；同时 LED 灯根据不同情况进行长度不一的闪烁。

这里我选择第二种方式来对结果进行展示：

这里用到了芯力量实验箱上的板载蜂鸣器，其电路图如下：

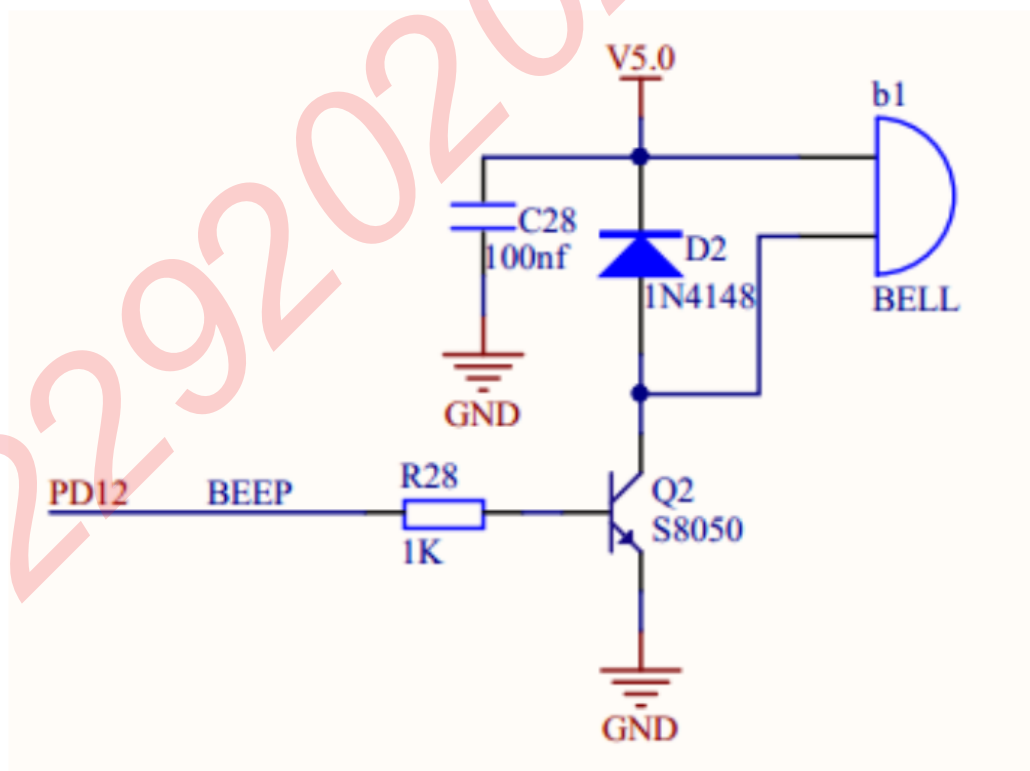


图 2.16 蜂鸣器电路

可以看到其通过引脚 PD12 控制板载蜂鸣器。

首先对板载蜂鸣器进行初始化:

```
102 | gd_gpio_init(); // 板载蜂鸣器、继电器初始化
```

```
346 //板载蜂鸣器、继电器初始化
347 void gd_gpio_init(void) {
348     rcu_periph_clock_enable(RCU_GPIOA);
349
350     gpio_mode_set(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, GPIO_PIN_3);
351     gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_3);
352
353     gpio_mode_set(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, GPIO_PIN_12);
354     gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_12);
355 }
```

这里两个反了

//蜂鸣器口

//继电器口

要求超过阈值时进行三次蜂鸣器长鸣:

```
49 void BEEP(uint32_t t) {
50     gpio_bit_set(GPIOA, GPIO_PIN_12);
51     delay_lms(t);
52     gpio_bit_reset(GPIOA, GPIO_PIN_12);
53     delay_lms(t);
54 }
55
56 void BEEP_LONG_3() { // 3声长鸣
57     uint8_t i;
58     for(i = 0; i < 3; i++) BEEP(1000);
59 }
```

, 通过改变 delay 的时间

间长短来改变蜂鸣器发出声音的时间长短。

三声长鸣后打开步进电机模拟通风口通风:

```
189 BEEP_LONG_3(); //long beep for 3 times
190 PWM_ON();
```

打开电机后蜂鸣器短鸣三声作为开启提示音:

```
190 PWM_ON();
191 BEEP_SHORT_3(); //short beep for 3 times
```

当温度、湿度恢复正常时, 长鸣一声:

```

194 while(E53_IAl_Data.Temperature > temp_ub
195 || E53_IAl_Data.Temperature < temp_lb
196 || E53_IAl_Data.Humidity > humd_ub
197 || E53_IAl_Data.Humidity < humd_lb) E53_IAl_Read_Data();
198
199 if(flag == false && E53_IAl_Data.Temperature <= temp_ub
200 && E53_IAl_Data.Temperature >= temp_lb
201 && E53_IAl_Data.Humidity <= humd_ub
202 && E53_IAl_Data.Humidity >= humd_lb) { // 进过上面
203     BEEP(2000);
204     cnt = 0; // 清空
205 }

```

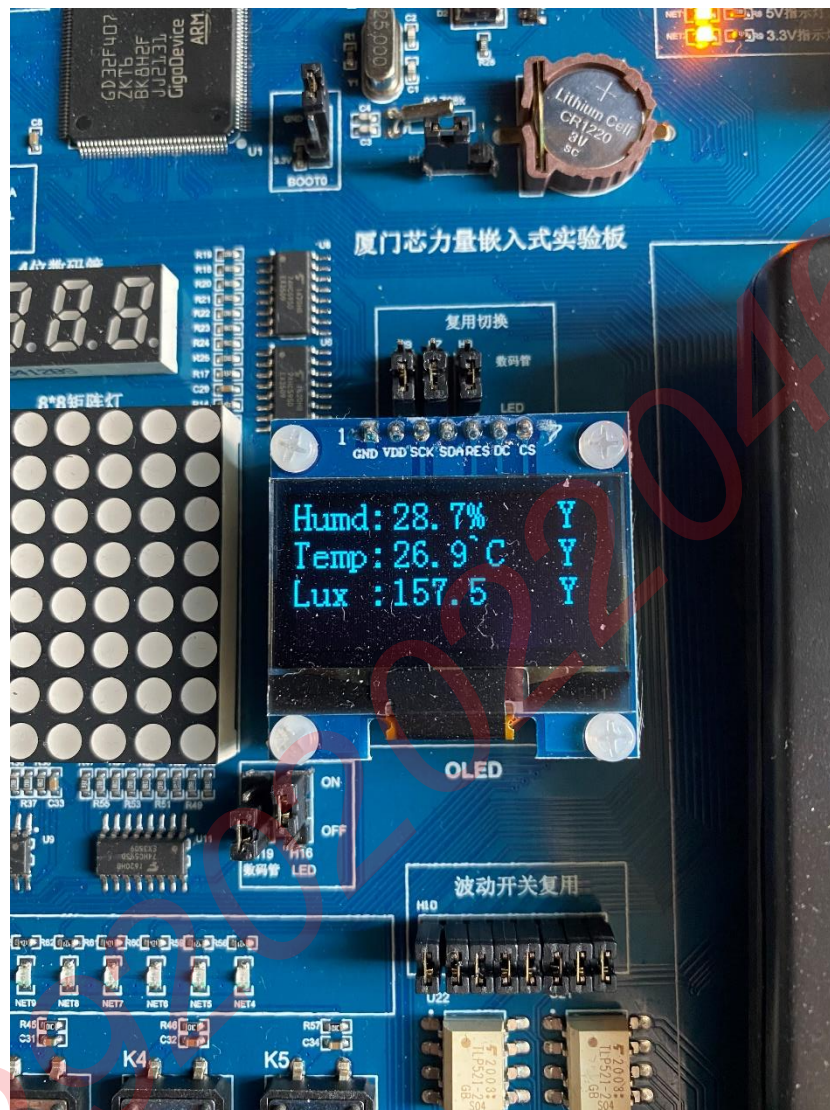
完整代码如下:

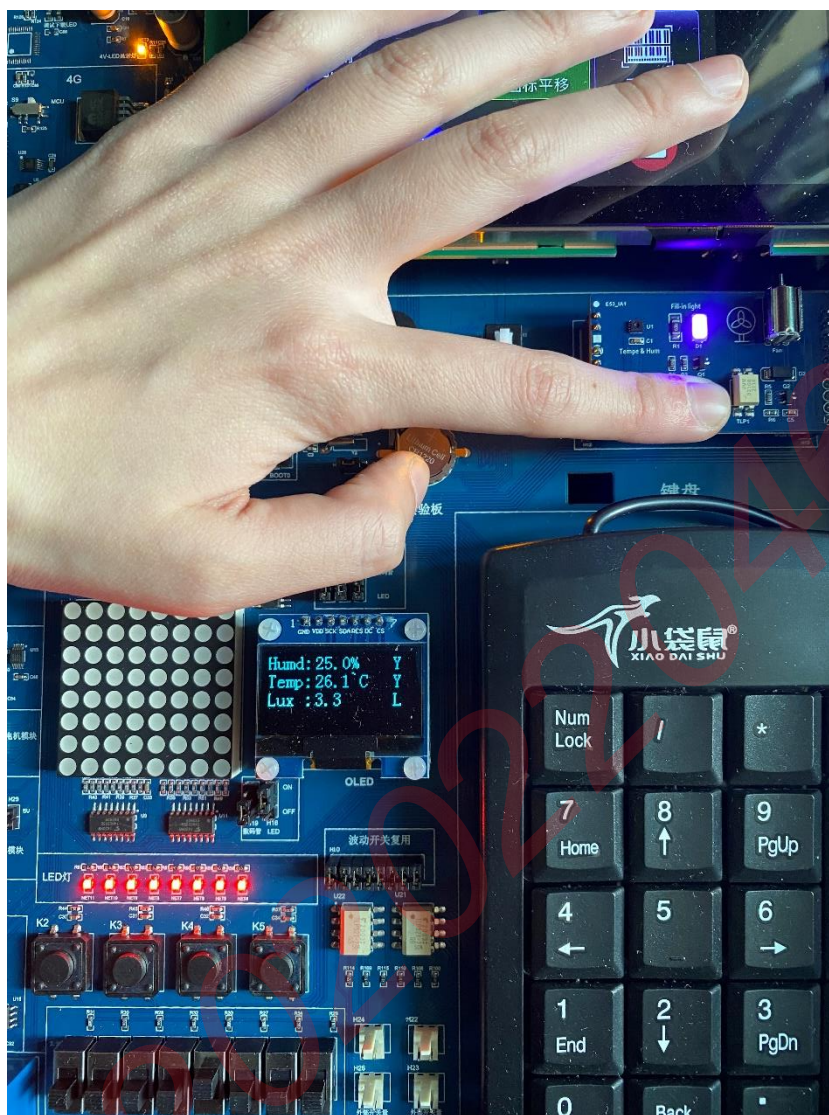
```

182 bool flag = true;
183 if((E53_IAl_Data.Temperature > temp_ub
184 || E53_IAl_Data.Temperature < temp_lb
185 || E53_IAl_Data.Humidity > humd_ub
186 || E53_IAl_Data.Humidity < humd_lb) && cnt == 0) { // 只有第一次进入这里才执行
187     flag = false; // 如果进过这里 flag置false;
188     cnt++;
189     BEEP_LONG_3(); //long beep for 3 times
190     PWM_ON();
191     BEEP_SHORT_3(); //short beep for 3 times
192 }
193
194 while(E53_IAl_Data.Temperature > temp_ub
195 || E53_IAl_Data.Temperature < temp_lb
196 || E53_IAl_Data.Humidity > humd_ub
197 || E53_IAl_Data.Humidity < humd_lb) E53_IAl_Read_Data();
198
199 if(flag == false && E53_IAl_Data.Temperature <= temp_ub
200 && E53_IAl_Data.Temperature >= temp_lb
201 && E53_IAl_Data.Humidity <= humd_ub
202 && E53_IAl_Data.Humidity >= humd_lb) { // 进过上面
203     BEEP(2000);
204     cnt = 0; // 清空
205 }

```


四、实验结果





用手挡住环境光传感器，补光灯开启，同时所有 LED 灯打开。



用手触摸湿度传感器，可以观察到湿度数值上升，达到高阈值后蜂鸣器发出声响。

五、心得体会

本次实验是一次综合性的大实验，运用到了多个模块。主要学习了 E53_IA1 模块的运用。该模块可以检测环境的湿度、温度以及光照强度，有补光灯、电机等多个外围设备。该模块的使用为智慧农业的应用场景提供了便捷的方式。

同时，在做综合实验模块时，我翻阅实验箱手册，知道了板载蜂鸣器的 IO 接口是 PD12。也发现了实验箱代码中的注释将继电器和蜂鸣器的接口弄反了。如果没有查看电路图，也就不知道蜂鸣器代码应该怎么写。

通过本次实验，我对嵌入式系统有了更深入的认识。在实验箱上插入一个模块就可以实现一些更专业的功能，如这次的智慧农业模块，我们可以检测环境的温度、湿度、光照强度等。如果实验箱能够接入物联网，那么我们可以将这些数据上传到云端，并对数据进行分析，由此来控制现实农场的环境。