嵌入式系统设计实验报告

实验：	实验 2 控制 LED 灯
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 23 日

目　　录

一、实验目的

1、掌握 GPIO 的原理与应用，学会使用 GPIO 控制 LED 灯的亮灭。

2、掌握 I/O 的配置和按键驱动的方法，掌握轮询和中断两种工作模式的原理。

二、实验方案设计

1. 基础实验

实验要求:

（1）通过配置 GPIO 点亮实验箱上的 LED 灯。

（2）使用拨码开关点亮实验箱上的 LED 灯。

（3）使用轮询方式，用按键开关点亮 LED 灯。

（4）使用中断方式，用按键开关点亮 LED 灯。

2. 进阶实验

实验要求:

(1) 使用拨码开关控制 LED 灯的亮灭：

当拨码开关 1 拨为 1 时，若将拨码开关 2 也拨为 1 则 LED 灯顺序亮起，将拨码开关 2 拨为 0、拨码开关 3 拨为 1 则灯顺

序熄灭，若拨码开关 2、3 同时拨为 1 灯亮起；若拨码开关 1 拨为 0，则无论拨码开关 2、3 为何值 LED 灯均不亮起。对应的逻辑表如下表所示:

拨码1	拨码2	拨码3	灯状态
0	X	X	都不亮
1	1	0	1-8顺序亮起
1	0	1	1-8顺序熄灭
1	1	1	1-8亮
1	0	0	1-8灭

(2) 使用按键开关控制 LED 灯的亮灭：将任一 LED 灯作为指示灯。

按下按键 1，所有 LED 灯亮起；

而后按下按键 4，除指示灯外所有灯间隔闪烁；

再次长按按键 4，灯亮时长即为按键时长；

最后按下按键 1，包括指示灯在内的所有灯熄灭，此时按下按键 2，LED 灯不亮起。

其对应的逻辑表如下表所示:

按键1第一次	按键1第二次	按键4第一次	按键4第二次	灯状态
1	0	0	0	都亮起
1	0	1	0	除指示灯外闪烁
1	0	1	1	亮灯
1	1	1	1	熄灭

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和进阶实验的代码；在 main 函数中，我使用了 switch-case 语句，来表示当前需要运行的是哪一个实验。定义一个 int 类型的变量 type， 数字表示它需要运行的是何种功能。其对应关系如下表:

type	功能
0	基础实验(1)
1	基础实验(2)
2	基础实验(3)
3	进阶实验(1)
4	进阶实验(2)

由于基础实验 4 是中断实验，并不需要写在函数中，所以没有和 type 类型对应。

```c
int main(void)
{
  systeminit();
  int type = 4;
  while(1){
    switch (type) {
      case 0: test1_1(); break;
      case 1: test1_2(); break;
      case 2: test1_3(); break;
      case 3: test2_1(); break;
      case 4: test2_2(); break;
      default: break;
    }
  }
}
```

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:

```c
int main(void)
{
  systeminit();
  int type = 0;
  while(1){
    switch (type) {
      case 0: test1_1(); break;
      case 1: test1_2(); break;
      case 2: test1_3(); break;
      case 3: test2_1(); break;
      case 4: test2_2(); break;
      default: break;
    }
  }
}
```

main中调用初始化函数

1. 基础实验

(1)　通过配置 GPIO 点亮实验箱上的 LED 灯。

将 type 设置为 0，调用 test1_1 函数，完成基础实验(1)。

```
17
18 void test1_1() {
19     gd_eval_led_on(LED1);
20     gd_eval_led_on(LED2);
21     gd_eval_led_on(LED3);
22 }
```

点亮三个LED灯

(2)　使用拨码开关点亮实验箱上的 LED 灯。

将 type 设置为 1，调用 test1_2 函数，完成基础实验(2)。

```
24 void test1_2() {
25   if(gpio_input_bit_get(DIP1_GPIO_PORT, DIP1_PIN) == SET) gd_eval_led_on(LED1);
26   else if(gpio_input_bit_get(DIP1_GPIO_PORT, DIP1_PIN) == RESET) gd_eval_led_off(LED1);
27 }
```

该实验的逻辑很简单，使用已封装好的函数 gpio_input_bit_get()获取拨码开关的状态，如果是高电平，说明拨码开关打开，点亮 LED 灯；反之，说明拨码开关关闭，熄灭 LED 灯。

(3)　使用轮询方式，用按键开关点亮 LED 灯。

将 type 设置为 2，调用 test1_3 函数，完成基础实验(3)。

```
29 void test1_3() {
30   while(1) {
31     if(gd_eval_key_state_get(USER1_KEY) == RESET) {
32       delay_1ms(100); // 防抖
33       if(gd_eval_key_state_get(USER1_KEY) == RESET) {
34         gd_eval_led_toggle(LED1);
35       }
36     }
37     if(gd_eval_key_state_get(USER4_KEY) == RESET) {
38       delay_1ms(100); // 防抖
39       if(gd_eval_key_state_get(USER4_KEY) == RESET) {
40         gd_eval_led_toggle(LED4);
41       }
42     }
43   }
44 }
45
```

我们这里将按键 1 和按键 4 设置成 GPIO 模式，将按键 2 和

按键 3 设置成中断模式，所以只用按键 1 和按键 4 来完成本实验。

实验的逻辑也很简单:

调用封装好的函数 gd_eval_key_state_get()获取按键是否有被按下，如果被按下，就调用 gd_eval_led_toggle()函数将 LED 灯的状态反转。

(4) 使用中断方式，用按键开关点亮 LED 灯。

使用按键 2 和按键 3 的中断模式，当按下按键 2 时就点亮 LED 灯 3；当按下按键 3 时就熄灭 LED 灯 3。

```
100  // 使用按键2中断控制LED3亮
101 void EXTI3_IRQHandler() {
102   if(exti_interrupt_flag_get(USER2_KEY_EXTI_LINE) != RESET) {
103     gd_eval_led_on(LED3);
104       exti_interrupt_flag_clear(USER2_KEY_EXTI_LINE);
105   }
106 }
107
```

注意这里要清空标志

```
108 void EXTI4_IRQHandler() {
109   if(exti_interrupt_flag_get(USER3_KEY_EXTI_LINE) != RESET) {
110     gd_eval_led_off(LED3);
111       exti_interrupt_flag_clear(USER3_KEY_EXTI_LINE);
112   }
113 }
```

调用 exti_interrupt_flag_get()函数获取中断模式的按键的状态，检测到一个脉冲，就改变灯的状态。

2. 进阶实验

(1) 使用拨码开关控制 LED 灯的亮灭

将 type 设置为 3，调用 test2_1 函数，完成进阶实验(1)。

```
void test2_1() {
  if(gpio_input_bit_get(DIP1_GPIO_PORT, DIP1_PIN) == RESET) { // 如果DIP1是0  直接全部熄灭
    led_off_immediately();
  } else { // DIP是1
    if(gpio_input_bit_get(DIP2_GPIO_PORT, DIP2_PIN) == SET && gpio_input_bit_get(DIP3_GPIO_PORT, DIP3_PIN) == RESET) led_on_by_order();
    if(gpio_input_bit_get(DIP2_GPIO_PORT, DIP2_PIN) == RESET && gpio_input_bit_get(DIP3_GPIO_PORT, DIP3_PIN) == SET) led_off_by_order();
    if(gpio_input_bit_get(DIP2_GPIO_PORT, DIP2_PIN) == SET && gpio_input_bit_get(DIP3_GPIO_PORT, DIP3_PIN) == SET) led_on_immediately();
  }
}
```

实现的逻辑是这样的:

a. 判断拨码开关 1 是否置低，如果其置低，那么所有灯都不会亮，调用 led_off_immediately()函数，该函数的作用是将所有 LED 灯都熄灭，也即实现了让所有灯都不亮的效果：

```
199 void led_off_immediately() {
200     gd_eval_led_off(LED1);
201     gd_eval_led_off(LED2);
202     gd_eval_led_off(LED3);
203     gd_eval_led_off(LED4);
204     gd_eval_led_off(LED5);
205     gd_eval_led_off(LED6);
206     gd_eval_led_off(LED7);
207     gd_eval_led_off(LED8);
208 }
```

b. 如果拨码开关 1 置高了，这个时候就可以对其他拨码开关进行判断。如果拨码开关 2 置高、3 置低，就让 LED 灯 1-8 顺序亮起，调用 led_on_by_order()函数实现这个功能；如果拨码开关 2 置低、3 置高，就让 LED 灯 1-8 顺序熄灭，调用 led_off_by_order()函数实现这个功能。

这两个函数的具体实现如下:

```
150 void led_on_by_order() {
151     gd_eval_led_on(LED1);
152     delay_1ms(100);
153     gd_eval_led_on(LED2);
154     delay_1ms(100);
155     gd_eval_led_on(LED3);
156     delay_1ms(100);
157     gd_eval_led_on(LED4);
158     delay_1ms(100);
159     gd_eval_led_on(LED5);
160     delay_1ms(100);
161     gd_eval_led_on(LED6);
162     delay_1ms(100);
163     gd_eval_led_on(LED7);
164     delay_1ms(100);
165     gd_eval_led_on(LED8);
166     delay_1ms(100);
167 }
```

```
169 void led_off_by_order() {
170     gd_eval_led_off(LED1);
171     delay_1ms(100);
172     gd_eval_led_off(LED2);
173     delay_1ms(100);
174     gd_eval_led_off(LED3);
175     delay_1ms(100);
176     gd_eval_led_off(LED4);
177     delay_1ms(100);
178     gd_eval_led_off(LED5);
179     delay_1ms(100);
180     gd_eval_led_off(LED6);
181     delay_1ms(100);
182     gd_eval_led_off(LED7);
183     delay_1ms(100);
184     gd_eval_led_off(LED8);
185     delay_1ms(100);
186 }
```

如果拨码开关 2 置高、3 置高，就让所有 LED 灯瞬间点亮，调用 led_on_immediately()实现该功能。

(2)　使用按键开关控制 LED 灯的亮灭

将 type 设置为 4，调用 test2_2 函数，完成进阶实验(2)。

```
void test2_2() {
  /*使用按键开关控制LED灯的亮灭*/
  uint8_t lightKey1Cnt = 0;
  uint8_t lightKey4Cnt = 0;


  while(1) {
    if(gd_eval_key_state_get(USER1_KEY) == RESET) {
      delay_1ms(50);
      if(gd_eval_key_state_get(USER1_KEY) == RESET) {
        if(gd_eval_key_state_get(USER1_KEY) == RESET && lightKey4Cnt == 0 && lightKey1Cnt == 0) {
          lightKey1Cnt = 1;
        }
        if(gd_eval_key_state_get(USER1_KEY) == RESET && lightKey4Cnt == 2) {
          lightKey1Cnt = 2;
        }
        delay_1ms(50);
      }
    }

    if(gd_eval_key_state_get(USER4_KEY) == RESET) {
      delay_1ms(50);
      if(gd_eval_key_state_get(USER4_KEY) == RESET) {
        while(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey1Cnt == 1 && lightKey4Cnt == 1) {
          led_on_immediately();
        }
        if(lightKey4Cnt == 1) lightKey4Cnt = 2;
        if(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey4Cnt == 0) lightKey4Cnt = 1; // first
      }
    }

    if(lightKey1Cnt == 1 && lightKey4Cnt == 0) {
      led_on_immediately();
    } else if(lightKey1Cnt == 1 && lightKey4Cnt == 1) {
      flashLED2_8();
    } else if(lightKey1Cnt == 1 && lightKey4Cnt == 2) {
      led_off_immediately();
      gd_eval_led_on(LED1);
    } else if(lightKey1Cnt == 2 && lightKey4Cnt == 2) {
      led_off_immediately();
      lightKey1Cnt = lightKey4Cnt = 0;
    }
  }
}
```

该实验相较于其他几个实验的逻辑稍微复杂一些。

首先，我们需要在函数里定义两个 uint8_t 类型的变量，lightKey1Cnt 和 lightKey4Cnt，它们分别表示当前按过的按键 1 的次数和按键 4 的次数。

然后，我们需要写一个 while 循环，如果不写这个循环的话，那么每次进来的时候，上面的 lightKey1Cnt 和 lightKey4Cnt 两个变量的值都会是 0，无法完成实验要求。

进入 while 循环，首先判断按键 1 有没有被按下，为了实现防抖，在这里调用 delay_1ms()函数，检测到按键 1 被按下后延迟 50 毫秒再检测一次，如果仍然检测到按键 1 被按下，

说明这次的按键按下不是由抖动引起的，就可以进行下一步的判断了。

```
while(1) {
  if(gd_eval_key_state_get(USER1_KEY) == RESET) {
    delay_1ms(50);
    if(gd_eval_key_state_get(USER1_KEY) == RESET) {
      if(gd_eval_key_state_get(USER1_KEY) == RESET && lightKey4Cnt == 0 && lightKey1Cnt == 0) {
        lightKey1Cnt = 1;
      }
      if(gd_eval_key_state_get(USER1_KEY) == RESET && lightKey4Cnt == 2) {
        lightKey1Cnt = 2;
      }
      delay_1ms(50);
    }
  }
```

防抖

```
if(gd_eval_key_state_get(USER1_KEY) == RESET && lightKey4Cnt == 0 && lightKey1Cnt == 0)
```

该 if 语句表示当前按键 1 和按键 4 一次都没有按过，并且此时检测到按键 1 被按下了，就将 lightKey1Cnt 置为 1，表示当前已经按了按键 1 一次了。于是下一次进入 while 循环就会进入下面的 if 判断语句:

```
if(lightKey1Cnt == 1 && lightKey4Cnt == 0) {
  led_on_immediately();
```

如果按键 1 按了一次，按键 4 还一次都没有按的话，就让所有的灯都亮起。

```
if(gd_eval_key_state_get(USER4_KEY) == RESET) {
  delay_1ms(50);
  if(gd_eval_key_state_get(USER4_KEY) == RESET) {
    while(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey1Cnt == 1 && lightKey4Cnt == 1) {
      led_on_immediately();
    }
    if(lightKey4Cnt == 1) lightKey4Cnt = 2;
    if(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey4Cnt == 0) lightKey4Cnt = 1; // first
  }
}
```

按键 4 的防抖机制与按键 1 的实现相同，这里不再赘述；

```
if(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey4Cnt == 0) lightKey4Cnt = 1; // first
```

该 if 语句表示如果按键 4 还一次都没有被按过并且此时检测到按键 4 被按下的话，就将 lightKey4Cnt 置为 1，表示当前

已经按了按键 4 一次了。于是下一次进入 while 循环就会进入下面的 if 语句:

```
} else if(lightKey1Cnt == 1 && lightKey4Cnt == 1) {
  flashLED2_8();
```

flashLED2_8 实现 LED2 到 LED8 的闪烁功能，此后如果不再按按键 1 和按键 4，LED2 到 LED8 就会间隔闪烁。

```
if(gd_eval_key_state_get(USER4_KEY) == RESET) {
  while(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey1Cnt == 1 && lightKey4Cnt == 1) {
    led_on_immediately();
  }
```

该 if 语句表示如果当前按键 1 和按键 4 都已经被按过一次了，并且此时又检测到按键 4 被按下，注意，这里又有一个 while 循环，它表示，当我一直按着按键 4 的时候，所有的灯就会一直亮起。

```
if(lightKey4Cnt == 1) lightKey4Cnt = 2;
```

然后，当退出循环时，此时已经按过按键 4 两次了，将 lightKey4Cnt 置为 2。

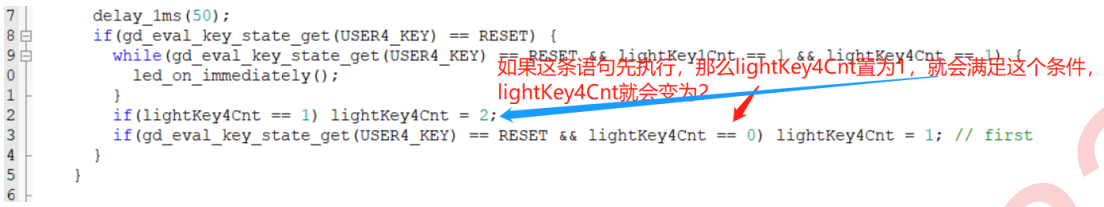
然后会进入到下面的 if 语句:

```
} else if(lightKey1Cnt == 1 && lightKey4Cnt == 2) {
  led_off_immediately();
  gd_eval_led_on(LED1);
```

将除了 LED1 的灯都熄灭，这就实现了要求的功能。

这里需要注意一点，按键 4 检测第二次被按下的语句要写在

检测第一次被按下的语句之前，否则程序会陷入逻辑错误:




```
    delay_1ms(50);
    if(gd_eval_key_state_get(USER4_KEY) == RESET) {
      while(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey1Cnt == 1 && lightKey4Cnt == 1) {
        led_on_immediately();
      }
      if(lightKey4Cnt == 1) lightKey4Cnt = 2;
      if(gd_eval_key_state_get(USER4_KEY) == RESET && lightKey4Cnt == 0) lightKey4Cnt = 1; // first
    }
  }
```

最后，当按键 4 已经被按了两次并且此时又检测到按键 1 被按下了，就将 lightKey1Cnt 置为 2:

```
if(gd_eval_key_state_get(USER1_KEY) == RESET && lightKey4Cnt == 2) {
  lightKey1Cnt = 2;
}
```

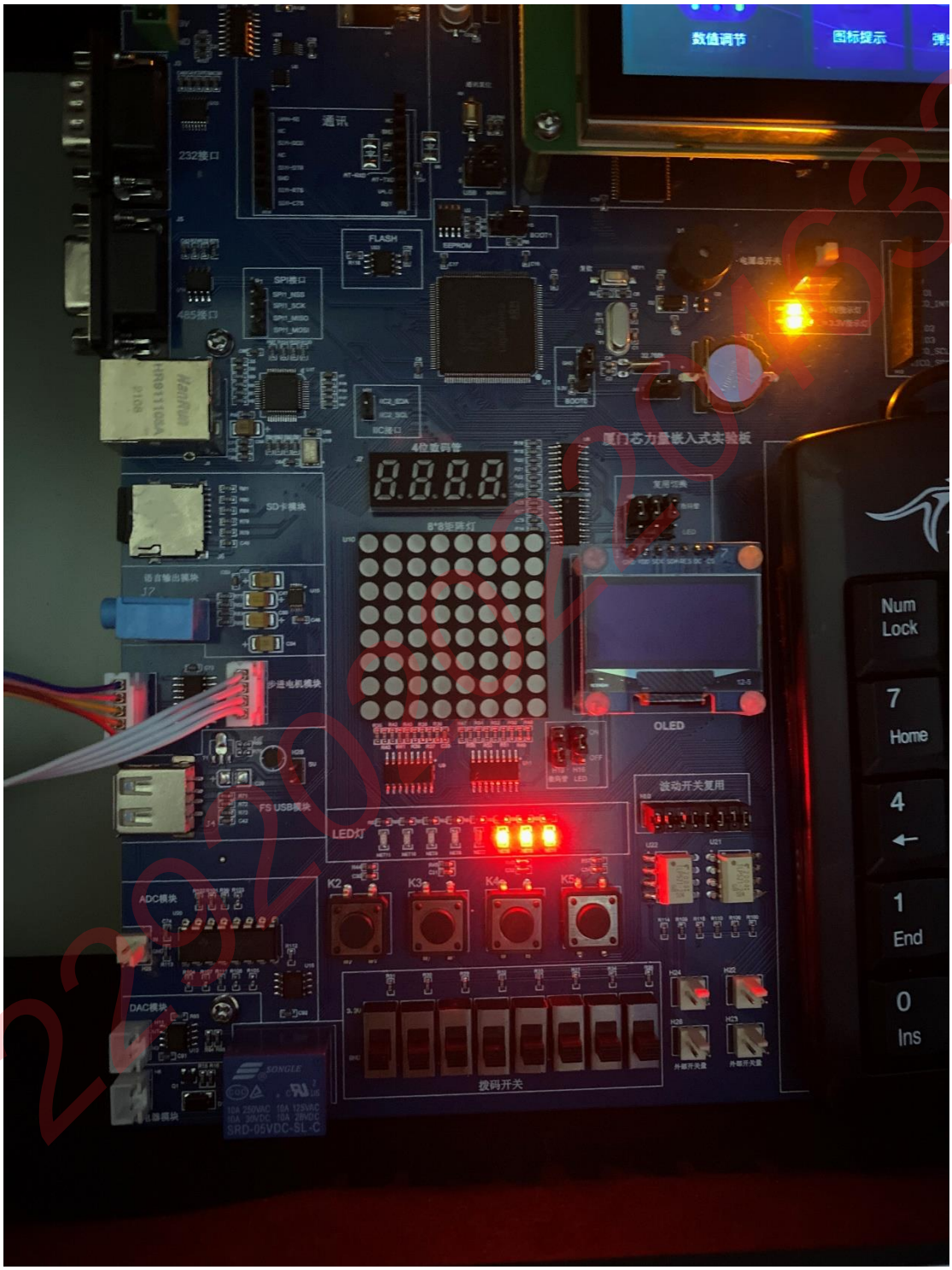
并会进入下面的 if 语句:

```
} else if(lightKey1Cnt == 2 && lightKey4Cnt == 2) {
  led_off_immediately();
  lightKey1Cnt = lightKey4Cnt = 0;
}
```

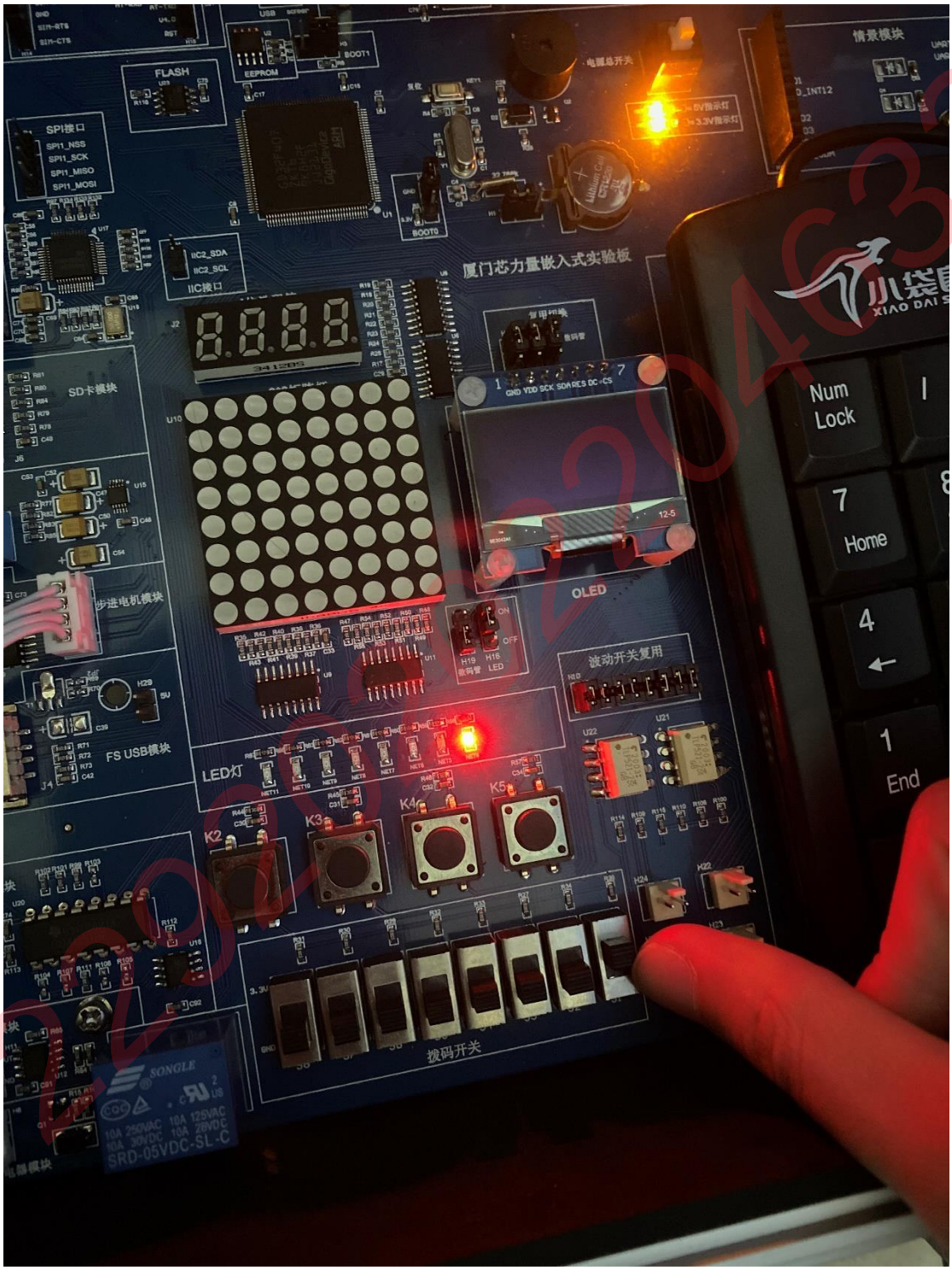
将所有灯（包括 LED1）熄灭，并将两个变量都赋值为 0，进入初始状态。

四、实验结果

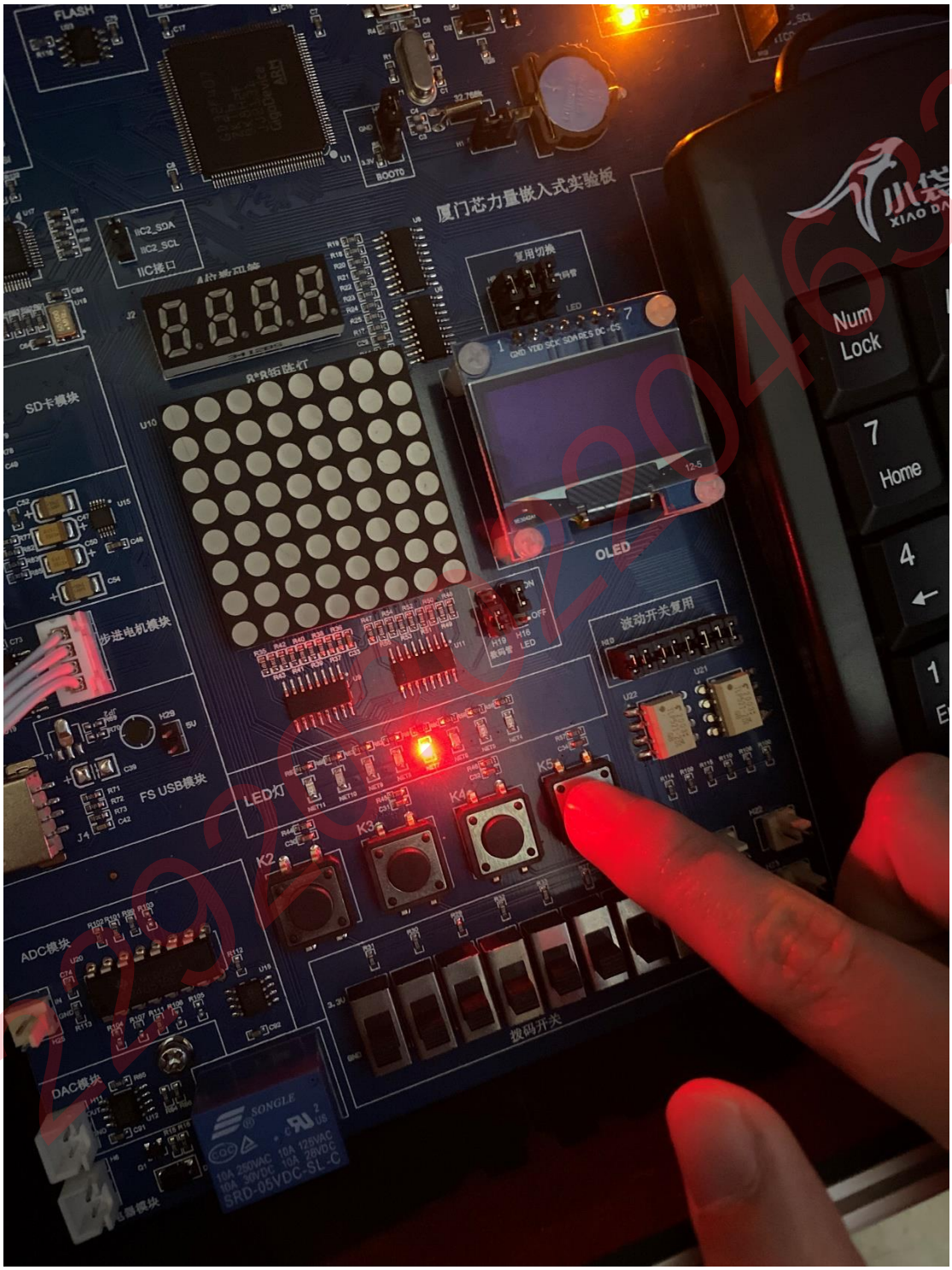
1. 基础实验 1

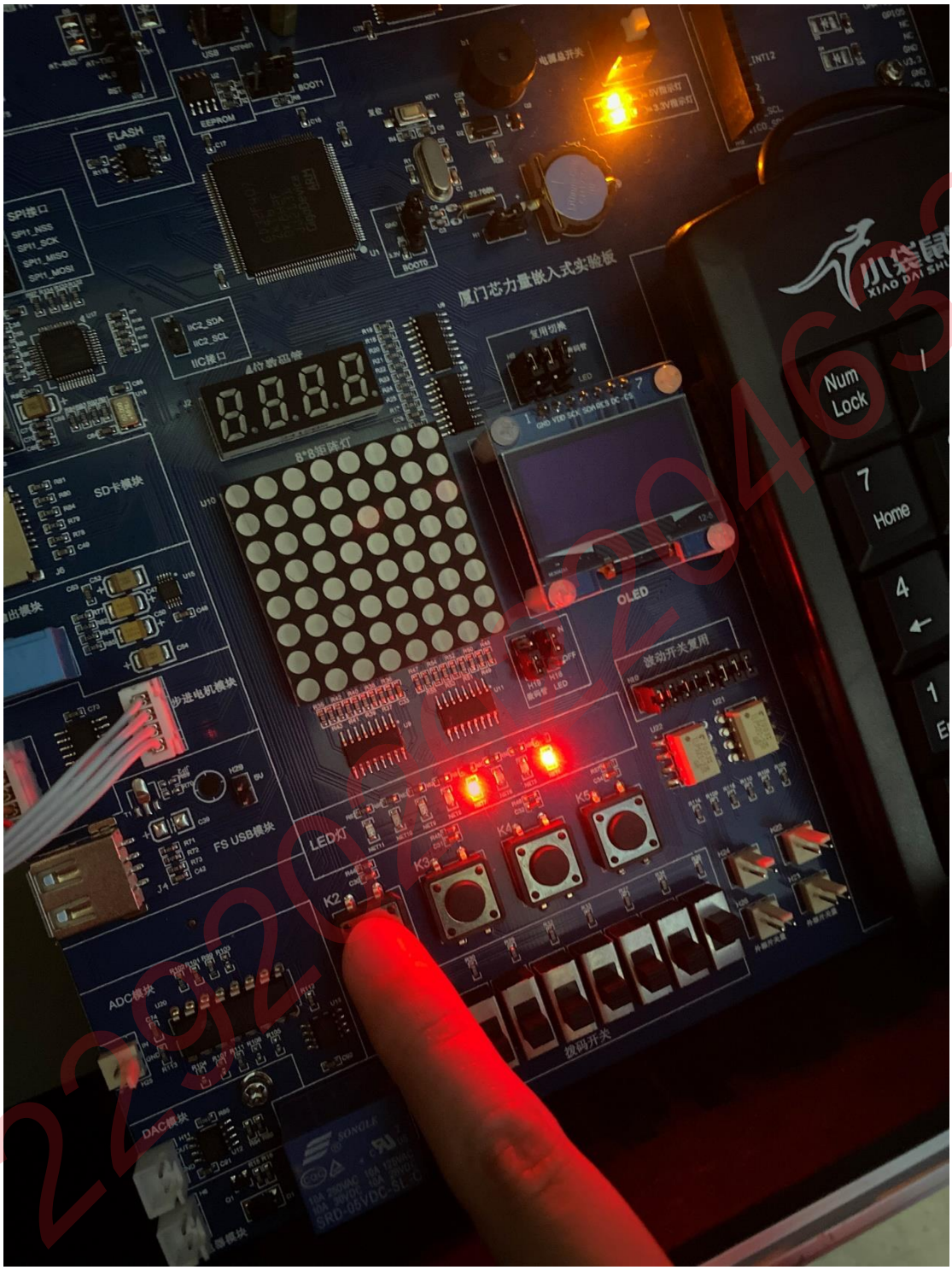

数值调节
图标提示
通讯
232接口
485接口
FLASH
SPI接口
厦门芯力量嵌入式实验板
4位数码管
8*8矩阵灯
OLED
步进电机模块
FS USB模块
LED灯
波动开关复用
K2
K3
K4
K5
ADC模块
DAC模块
拨码开关
Num Lock
7 Home
4
1 End
0 Ins

2. 基础实验 2

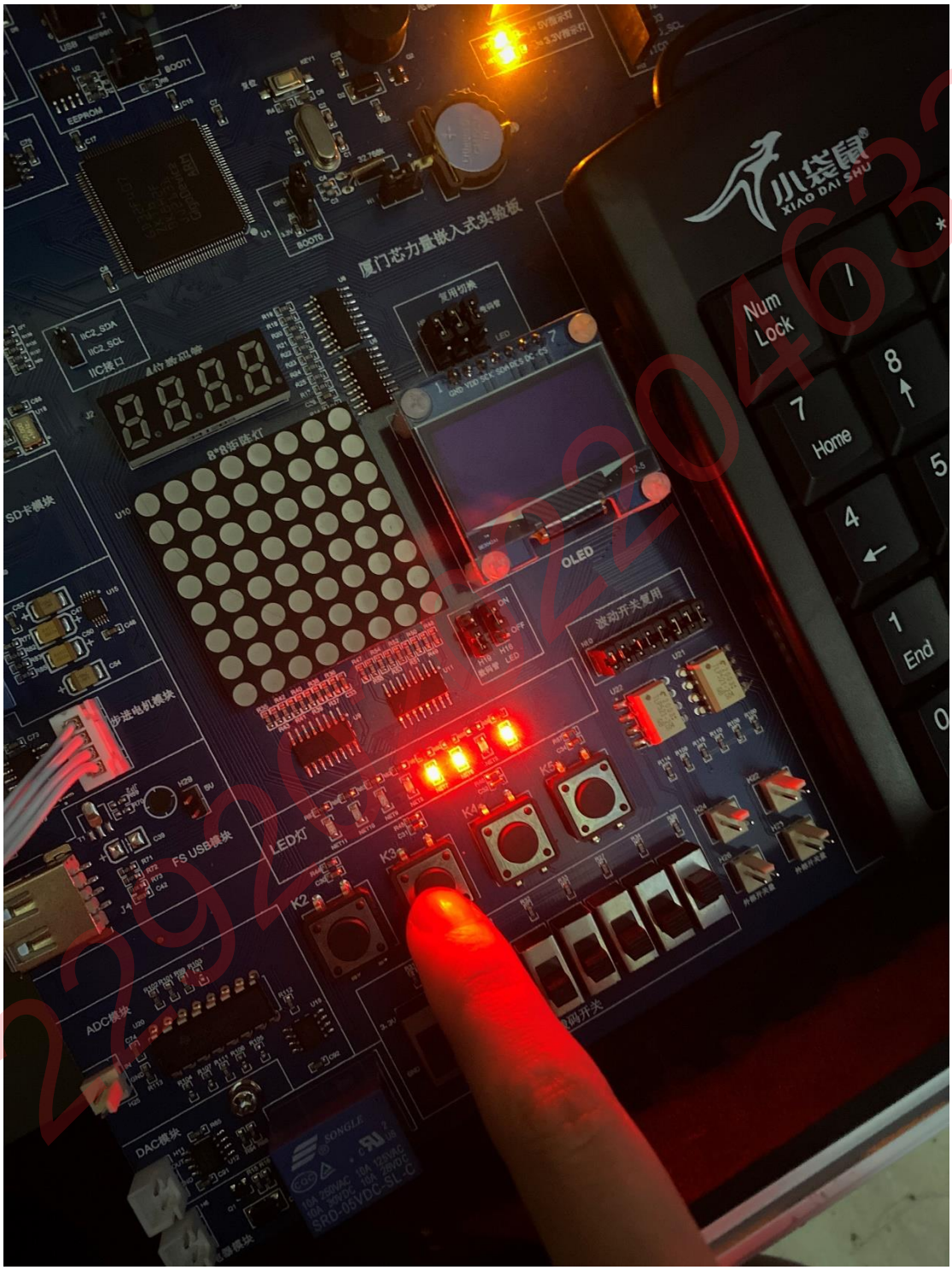

厦门芯力量嵌入式实验板
FLASH
EEPROM
SPI接口
SPI1_NSS
SPI1_SCK
SPI1_MISO
SPI1_MOSI
IIC2_SDA
IIC2_SCL
IIC接口
SD卡模块
步进电机模块
FS USB模块
LED灯
K2
K3
K4
K5
OLED
波动开关复用
拨码开关
SRD-05VDC-SL-C
Num Lock
7 Home
4
1 End

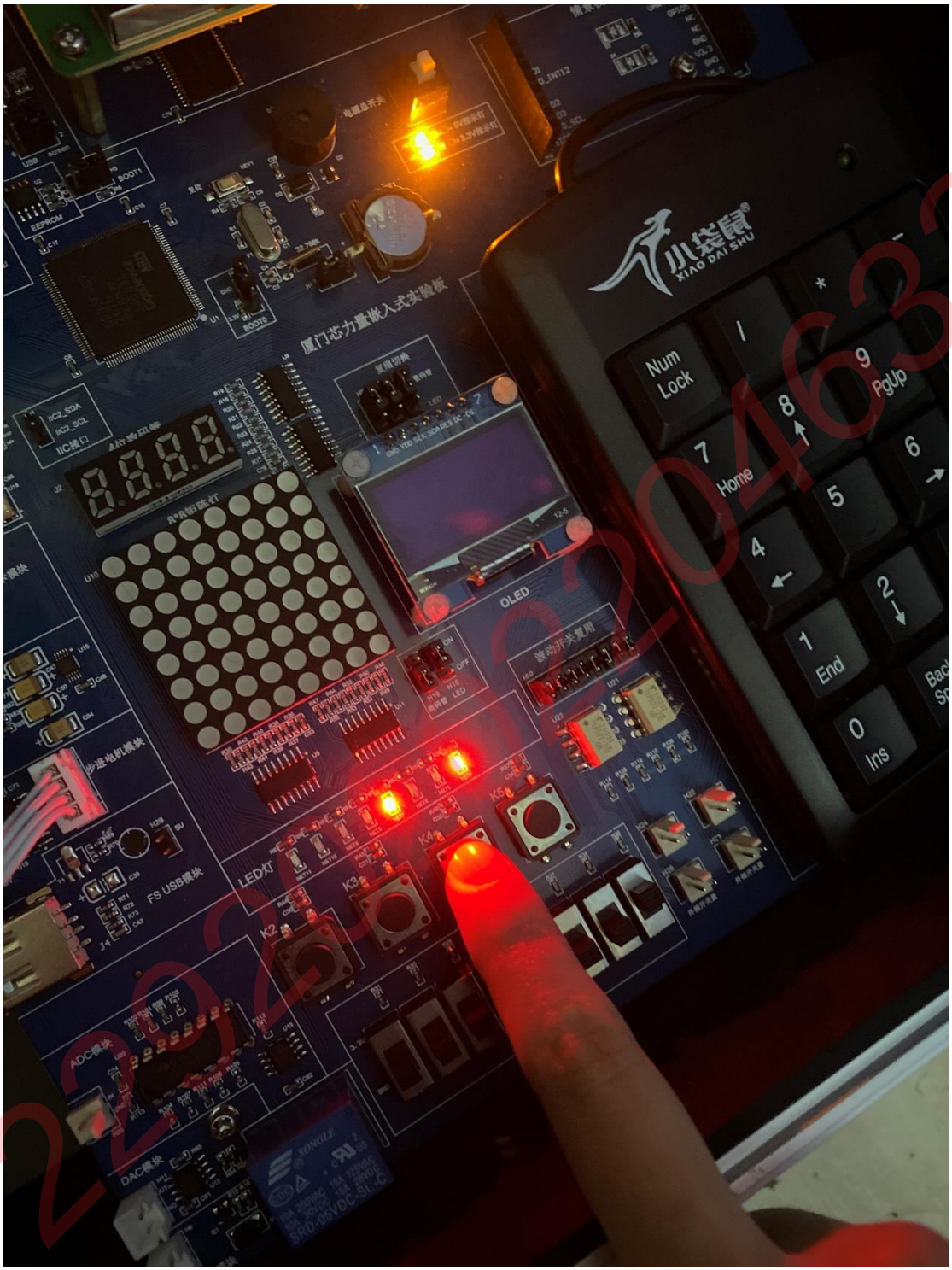
3. 基础实验 3


厦门芯力量嵌入式实验板
FLASH
IIC2_SDA
IIC2_SCL
IIC接口
SD卡模块
步进电机模块
FS USB模块
ADC模块
DAC模块
LED灯
OLED
拨码开关
K2
K3
K4
K5
Num Lock
7 Home
4
1

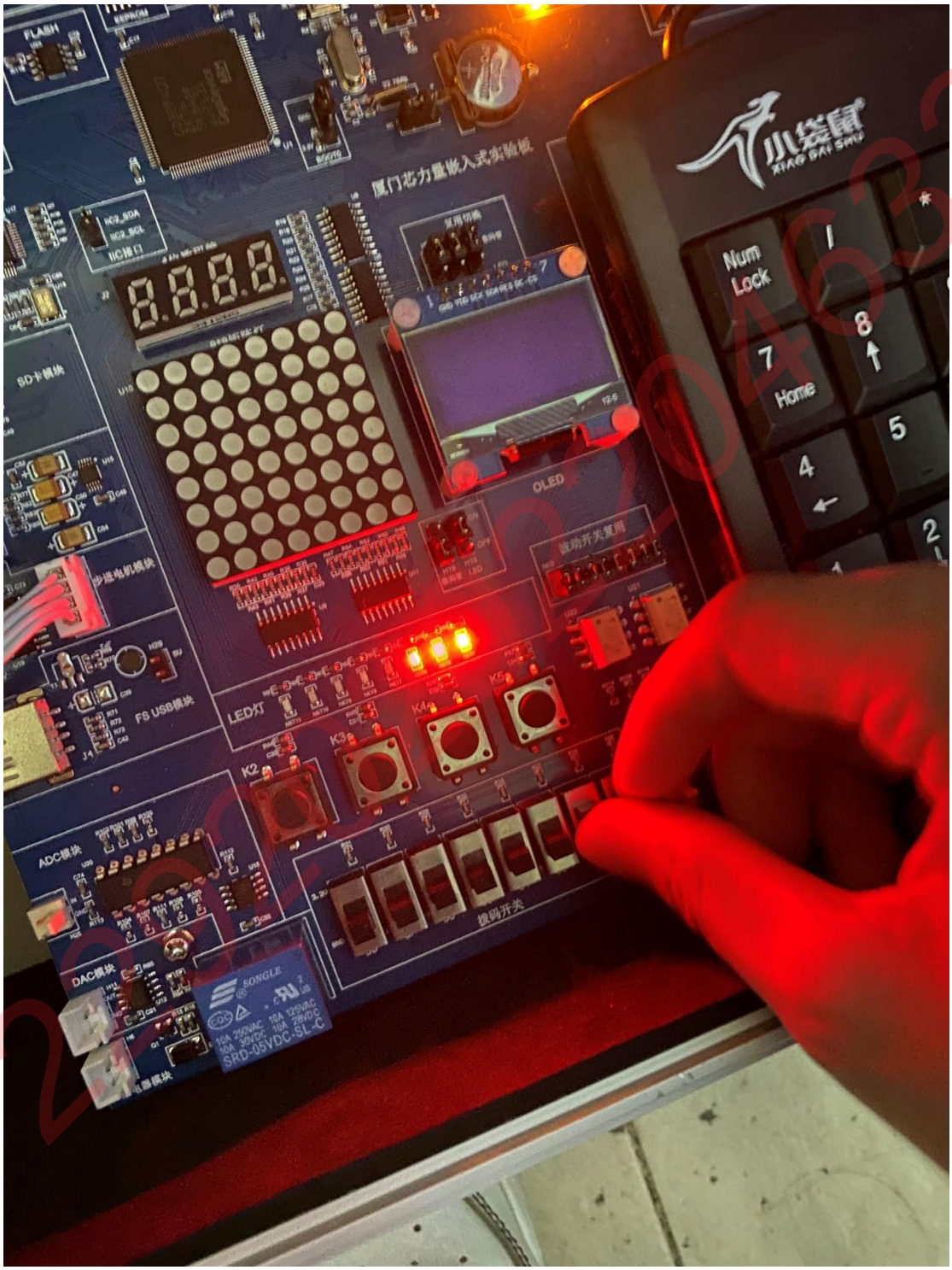

厦门芯力量嵌入式实验板
FLASH
EEPROM
SPI接口
IIC接口
SD卡模块
步进电机模块
FS USB模块
ADC模块
DAC模块
LED灯
K2
K3
K4
K5
OLED
拨动开关复用
编码开关
Num Lock
7 Home
4

4. 基础实验 4

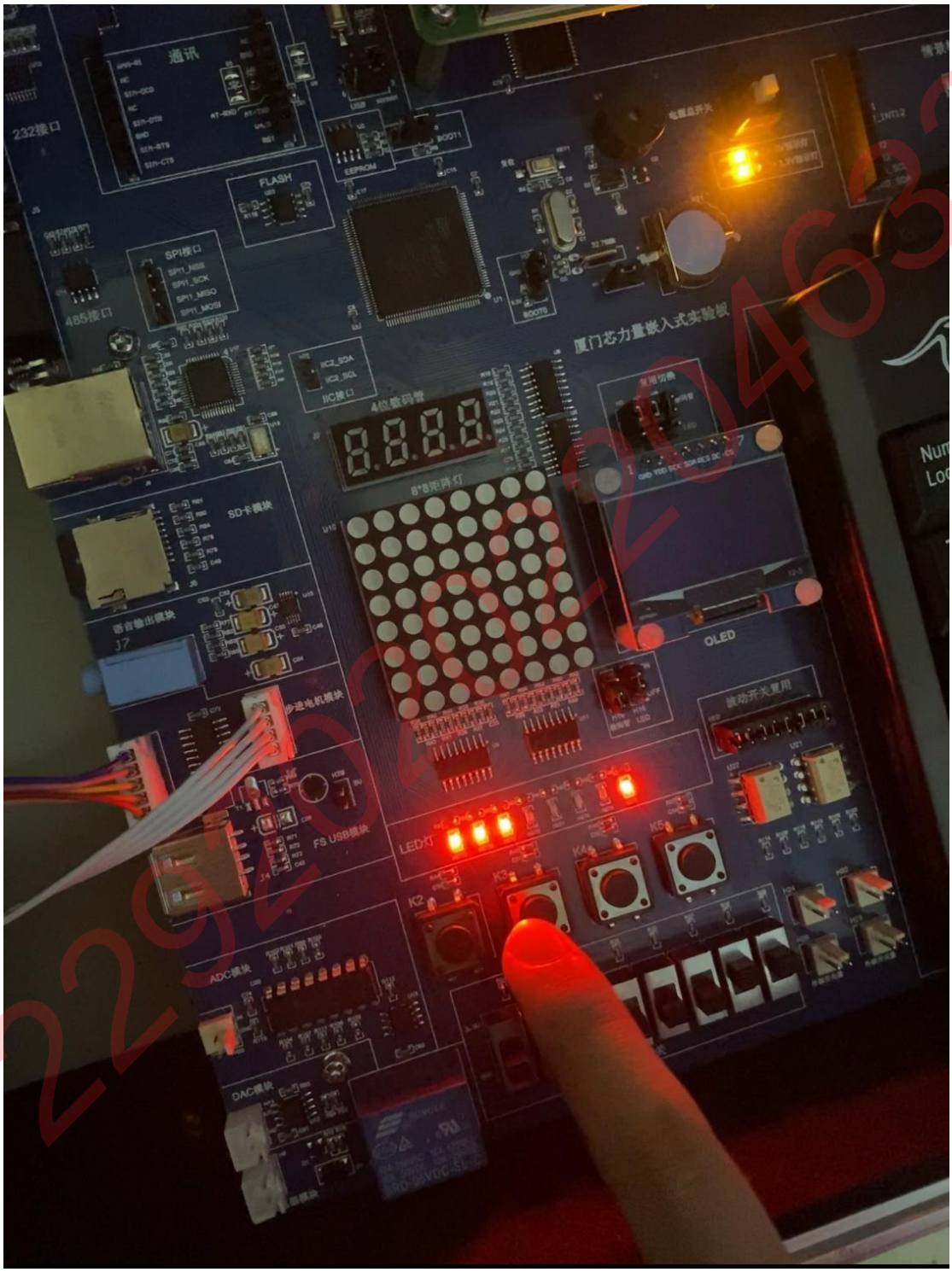

厦门芯力量嵌入式实验板
OLED
LED灯
FS USB模块
ADC模块
DAC模块
步进电机模块
SD卡模块
小袋鼠
XIAO DAI SHU
Num Lock
Home
End


厦门芯力量嵌入式实验板
OLED
LED灯
FS USB模块
ADC模块
DAC模块
步进电机模块
小袋鼠
XIAO DAI SHU
Num Lock
Home
PgUp
End
Ins

5. 进阶实验 1


厦门芯力量嵌入式实验板
OLED
LED灯
ADC模块
DAC模块
FS USB模块
步进电机模块
SD卡模块
拨码开关
Num Lock
Home

6. 进阶实验 2


通讯
232接口
485接口
FLASH
SPI接口
SPI1_NSS
SPI1_SCK
SPI1_MISO
SPI1_MOSI
IIC接口
IIC2_SDA
IIC2_SCL
4位数码管
8*8点阵灯
SD卡模块
语音输出模块
步进电机模块
FS USB模块
ADC模块
DAC模块
OLED
LED灯
K2
K3
K4
K5
厦门芯力量嵌入式实验板

五、实验分析

本次实验并不是很难，很可能是所有实验中最简单的一次实验。但是有一些逻辑问题需要弄清楚，尤其是完成进阶实验 2 的那一块，一开始我以为需要用中断函数来实现这个实验最后熄灭的功能，但是始终失败，后来才发现，这个功能的实现只要逻辑清晰，也是能够完成地非常好的。

六、心得体会

LED 灯实验是所有单片机实验中最简单的一关，但是如果想把它的效果做得炫酷也是非常有意思的。本次实验是嵌入式系统设计这门课真正的第一次实验，我觉得编写嵌入式这种工程代码和我以前写 C 语言程序的感觉完全不同，以前不会用到这种多文件的模式，当刚接触这种多文件的模式会有些不太适应，尤其是各种库函数 include 让我感到困惑，这还是我的 C 语言基础打得不牢固的原因吧，希望以后水平能一点点得到提高。