厦门大学

XIAMEN UNIVERSITY

嵌入式系统开发

实验三 定时器与串口通讯

姓 名	
学 号	
学 院	信息学院
专 业	计算机科学与技术专业
年 级	2022
日 期	2024/11/29

目录

1 实验目的 1

2 实验方案设计 1

2.1 基础实验 1

2.2 提高实验 3

3 实验过程与结果 3

3.1 基础实验 3

3.1.1 写王字 3

3.1.2 通过 ADC 读取电压值，使用数码管显示。 5

3.2 提高实验 7

3.2.1 使用按键控制 DAC 输出，改变输出数值 7

3.3 拓展实验 3.3.1 贪吃蛇 9

4 实验分析 12

5 实验总结 13

1 实验目的

（1） 掌握串口通讯的工作原理与配置方法，以及定时器的原理。

（2） 学会使用串口调试工具，以及触摸屏的使用和软件配置方法。

2 实验方案设计

2.1 基础实验

2.1.1

（1）实验要求

学生选择一个想要选择的汉字，将该字按照笔画顺序，按照逐个灯亮起的模式在实验箱的矩阵 LED 上“写”出该字。同笔画的 LED 灯亮起间隔为 3s，上一笔画的最后 LED 灯和下一笔画的 LED 灯亮起的间隔为 6s。

（2）实验步骤

步骤一：在 Soft_Drive 中新建 74HC595 的驱动文件 74HC595_LED.c,配置 PB1，PB13，PB15 为输出模式，根据 74HC595 的驱动逻辑写对应的驱动程序。

步骤二：利用 8*8LED 点阵的取模软件如图 4.8 所示，取出对应位的逻辑，转换成 16 进制数组，对应每个笔画需要定义一个数组。

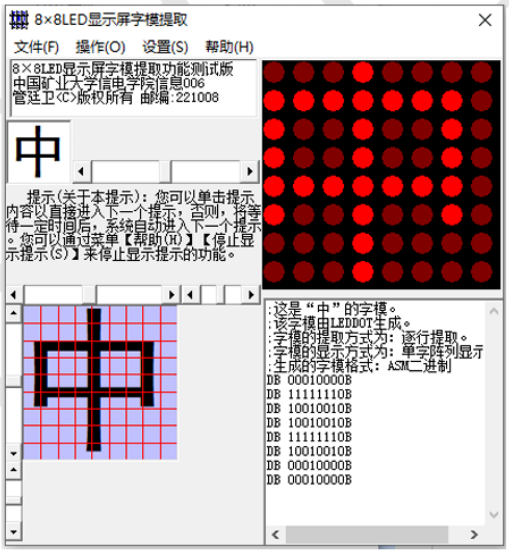


图 4.8 8*8LED 矩阵灯取模软件

步骤三：新建 timer.c 文件配置定时器初始化函数以及定时器处理函数，例程中配置了定时器 3 每 3S 发生一次中断，在定时器中断中调用串口的发送指令，实现每 3S 发送指令到触摸屏显示对应的文字，利用字符是否整除 5 的特性处理每句显示间隔为 6S 的要求。

在定时器中断处理函数中设置好对应的显示字符的程序。

步骤四：上传实验结果：如图 4.9 在 main.c 函数中定义当前使用的设备号和自定矩阵等显示的字符（选择的汉字：A：中/B：开/C：火/D：文），调用 json_student_imformation 函数自动上传实验结果。

```
char machinenum[]= "001";//在这填写设备号;
char charused[]="A";//在这填写自定文字;
```

图 4.9 上传信息定义

步骤五：下载到实验板中验证实验。

2.2.2

基础实验：通过 ADC 读取电压值，使用数码管显示。

2.2 提高实验

（1）实验要求

使用按键控制 DAC 输出，改变输出数值。

3 实验过程与结果

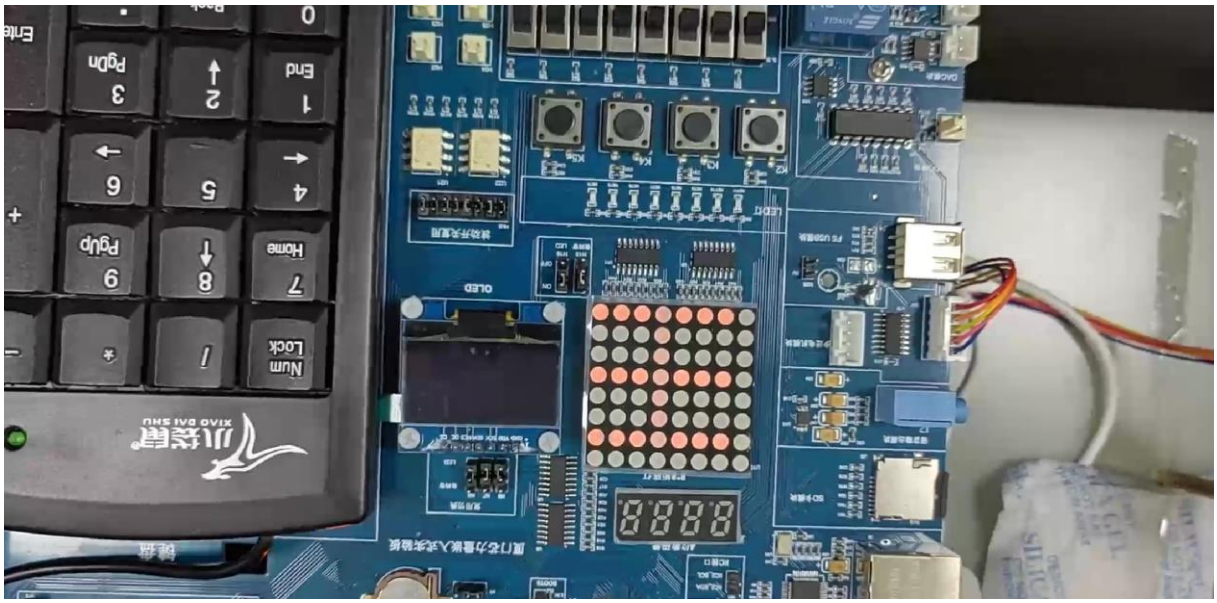
3.1 基础实验

3.1.1 写王字

初始化系统和外设。system_init(void);

使用 HC595_Send_Byte 函数将数据发送到 8x8 LED 矩阵。

按顺序显示王字的各个部分。



```
void test3(){
    if(tim3_count==0)
        for(i=0;i<8;i++)
    {
        HC595_Send_Byte(row_clear[i]);
```

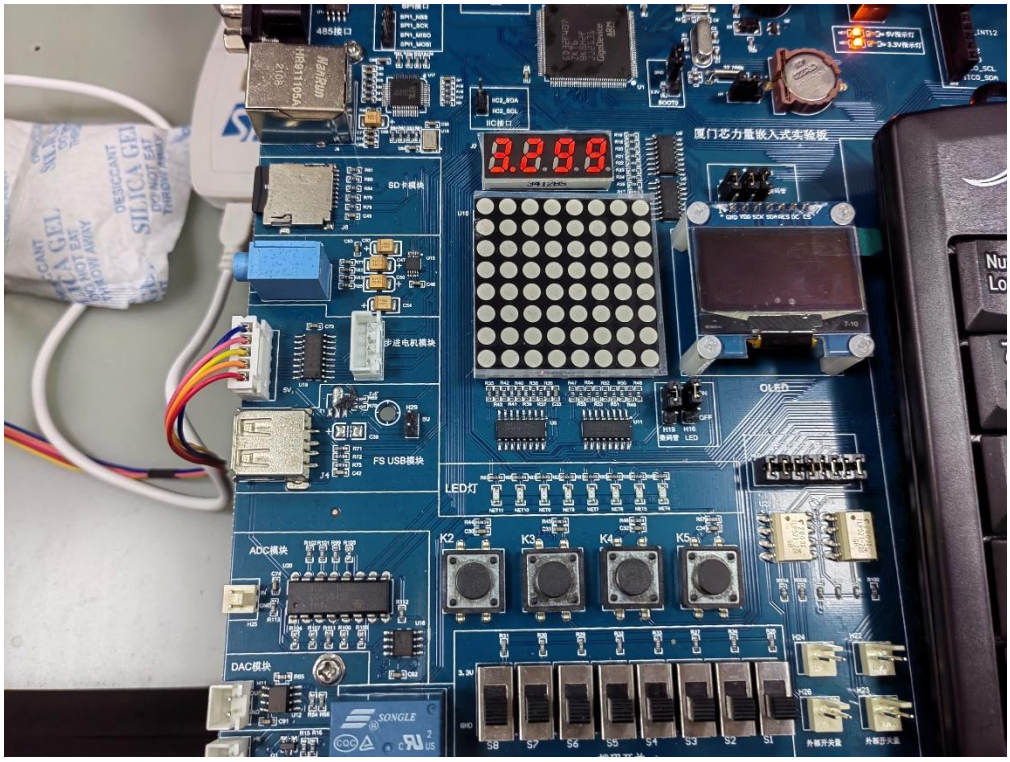
```
        HC595_Send_Byte(cal_clear[i]);
        HC595_CS();
    }

    if(tim3_count==1)
        for(i=0;i<8;i++)
    {
        HC595_Send_Byte(row1[i]);
        HC595_Send_Byte(core1[i]);
        HC595_CS();
    }
    if(tim3_count==2)
        for(i=0;i<8;i++)
    {
        HC595_Send_Byte(row2[i]);
        HC595_Send_Byte(core2[i]);
        HC595_CS();
    }
    if(tim3_count==3)
        for(i=0;i<8;i++)
    {
        HC595_Send_Byte(row3[i]);
        HC595_Send_Byte(core3[i]);
        HC595_CS();
    }
    if(tim3_count==4)
        for(i=0;i<8;i++)
    {
        HC595_Send_Byte(row4[i]);
        HC595_Send_Byte(core4[i]);
        HC595_CS();
    }
    if(tim3_count==5)
        for(i=0;i<8;i++)
    {
        HC595_Send_Byte(row_clear[i]);
        HC595_Send_Byte(cal_clear[i]);
        HC595_CS();
    }
}
```

3.1.2 通过 ADC 读取电压值，使用数码管显示。

初始化系统和外设。

读取 ADC 数据并转换为电压值。

使用数码管显示电压值。




```
float adc_read_data(void)
{
    adc_software_trigger_enable(ADC0, ADC_INSERTED_CHANNEL);
    delay_1ms(50);
    adc_value = ADC_IDATA0(ADC0);
    return adc_value * 3.3 / 4096;
}

void shumaguan_play(char *arr)
{
    uint8_t i;
    for(i = 0; i < 5; i++)
    {
        if(i == 0)
        {
            if(arr[i] == '0')
                HC595_SelecTube(3-i,16);
            else if(arr[i] == '1')
                HC595_SelecTube(3-i,17);
```

```
            else if(arr[i] == '2')
                HC595_SelecTube(3-i,18);
            else if(arr[i]=='3')
                HC595_SelecTube(3-i,19);
            else if(arr[i]=='4')
                HC595_SelecTube(3-i,20);
            else if(arr[i]=='5')
                HC595_SelecTube(3-i,21);
            else if(arr[i]=='6')
                HC595_SelecTube(3-i,22);
            else if(arr[i]=='7')
                HC595_SelecTube(3-i,23);
            else if(arr[i]=='8')
                HC595_SelecTube(3-i,24);
            else if(arr[i]=='9')
                HC595_SelecTube(3-i,25);
        }
        else if(i != 1)
        {
            if(arr[i] == '0')
                HC595_SelecTube(4-i,0);
            else if(arr[i] == '1')
                HC595_SelecTube(4-i,1);
            else if(arr[i] == '2')
                HC595_SelecTube(4-i,2);
            else if(arr[i]=='3')
                HC595_SelecTube(4-i,3);
            else if(arr[i]=='4')
                HC595_SelecTube(4-i,4);
            else if(arr[i]=='5')
                HC595_SelecTube(4-i,5);
            else if(arr[i]=='6')
                HC595_SelecTube(4-i,6);
            else if(arr[i]=='7')
                HC595_SelecTube(4-i,7);
            else if(arr[i]=='8')
                HC595_SelecTube(4-i,8);
            else if(arr[i]=='9')
                HC595_SelecTube(4-i,9);
        }
    }
}

void test1(void) {
```

```
    adc_output_value = adc_read_data();
    sprintf(arr, "%.3f", adc_output_value);
    while(1)
    {
        shumaguan_play(arr);
    }
}
```

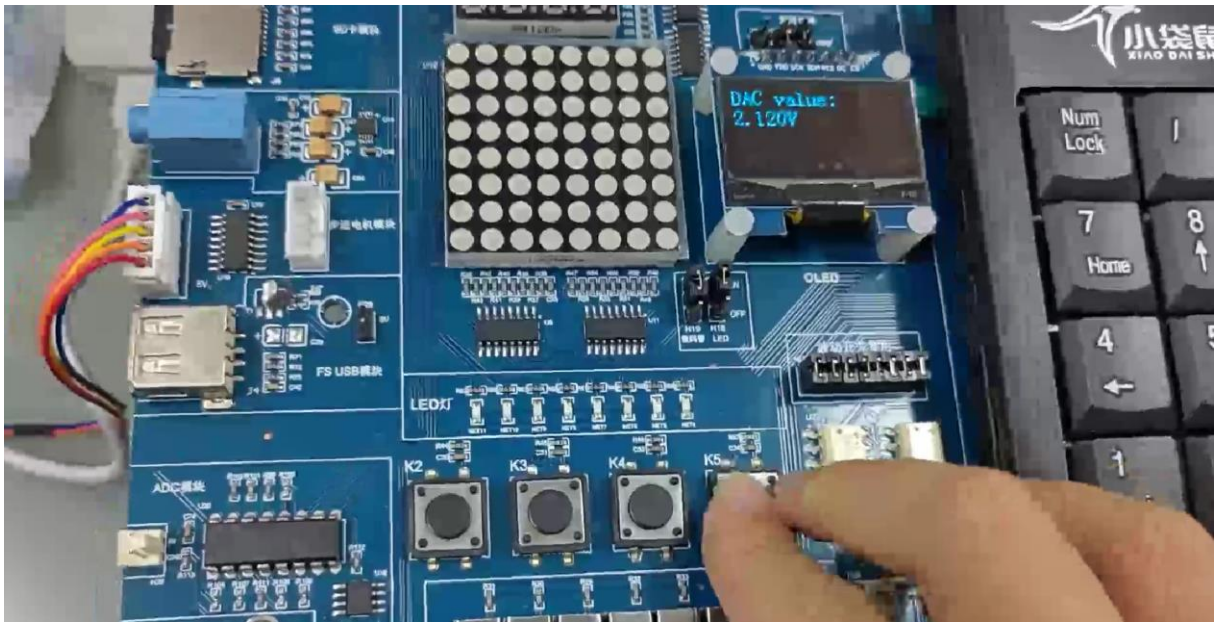
3.2 提高实验

3.2.1 使用按键控制 DAC 输出，改变输出数值

初始化系统和外设。

使用按键控制 DAC 输出值。

在 OLED 显示屏上显示 DAC 输出值。




```
void test2(void) {
    while(1)
    {
        char str[4];

        if(dac_value1 > 3.300){
            dac_value1 = 0.000;
            OLED_Clear();
            OLED_ShowString(0, 4, "out of range");
```

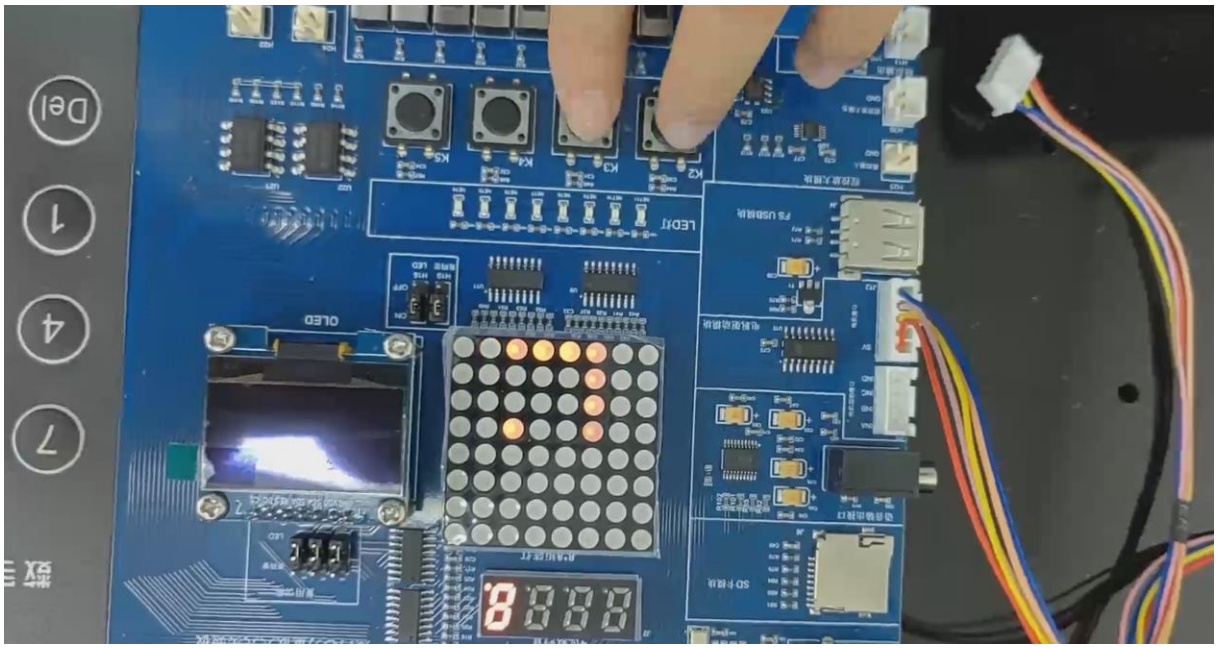
```
            delay_1ms(500);
            OLED_Clear();
        }

        uint16_t dac_register_value = (uint16_t)(dac_value1 / 3.3 * 4096);
        DAC_OUT_VAL1 = dac_register_value;
        DAC_OUT_VAL = DAC_OUT_VAL1 * 16;

        dac_data_set(DAC0, DAC_ALIGN_12B_L, DAC_OUT_VAL);
        sprintf(str, "%.3f", dac_value1);
        OLED_ShowString(0, 0, "DAC value:");
        OLED_ShowString(0, 2, str);
        OLED_ShowString(40, 2, "V");
    }
}

void EXTI2_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER1_KEY_EXTI_LINE)) {
        dac_value1 += 1.000;
    }
    exti_interrupt_flag_clear(USER1_KEY_EXTI_LINE);
}

void EXTI3_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER2_KEY_EXTI_LINE)) {
        dac_value1 += 0.100;
    }
    exti_interrupt_flag_clear(USER2_KEY_EXTI_LINE);
}

void EXTI4_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER3_KEY_EXTI_LINE)) {
        dac_value1 += 0.010;
    }
    exti_interrupt_flag_clear(USER3_KEY_EXTI_LINE);
}

void EXTI5_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER4_KEY_EXTI_LINE)) {
        dac_value1 += 0.001;
    }
    exti_interrupt_flag_clear(USER4_KEY_EXTI_LINE);
}
```

3.3 拓展实验

3.3.1 贪吃蛇

初始化系统和外设。
使用按键控制贪吃蛇的方向。
实现贪吃蛇的移动和食物生成。

(见 qq 视频)



```
uint8_t snake_row[] = {0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
uint8_t dct = 1; // direction
uint8_t queue[64] = {0, 1, 2, 3}, head = 3, tail = 0, new_head, food = 0;
uint8_t run_game = 1, eat = 1;

void snake_run(){
    for(int i = 0; i < 5000; i++){
        if(RESET == gd_eval_key_state_get(USER4_KEY))
            dct = 3;
        snake_draw();
    }
}

void snake_play(){
    for(;;){
        if(run_game == 0){
```

```
            for(int j = 0; j < 8; j++){
                snake_row[j] = 0;
            }
            int len = (head + 64 - tail) % 64;
            int t = 0;
            while(len > 8){
                snake_row[t++] = 0xff;
                len -= 8;
            }
            for(int j = 0; j <= len; j++){
                snake_row[t] |= (1 << j);
            }
            for(;;){
                snake_run();
            }
        }

        if(eat != 0){
            new_food();
            eat = 0;
        }
        snake_run();
        snake_step();
    }
}

void snake_step(){
    snake_row[queue[tail] >> 3] ^= (1 << (queue[tail] & 7));
    new_head = queue[head];
    tail++;
    head++;
    if (tail == 64) tail = 0;
    if (head == 64) head = 0;
    if (dct == 0)
        new_head = (new_head & 56) | (((new_head & 7) + 7) % 8);
    else if (dct == 1)
        new_head = (new_head & 56) | (((new_head & 7) + 1) % 8);
    else if (dct == 2)
        new_head = ((((new_head >> 3) + 7) % 8) << 3) | (new_head & 7);
    else if (dct == 3)
        new_head = ((((new_head >> 3) + 1) % 8) << 3) | (new_head & 7);
    if (snake_row[new_head >> 3] & (1 << (new_head & 7))) {
        if((new_head >> 3) == (food >> 3) && (new_head & 7) == (food & 7)){
            tail--;
```

```
            if(tail == 0){
                tail = 63;
            }
            snake_row[queue[tail] >> 3] ^= (1 << (queue[tail] & 7));
            eat = 1;
        }
        else
            run_game = 0;
    }
    snake_row[new_head >> 3] |= (1 << (new_head & 7));
    queue[head] = new_head;
}

void snake_draw(){
    uint8_t i;
    for(i = 0; i < 8; i++){
        HC595_Send_Byte(snake_row[i]);
        HC595_Send_Byte(core1[i]);
        HC595_CS();
    }
}

void new_food(){
    int r, c;
    do{
        r = rand() % 8;
        c = rand() % 8;
    }while((snake_row[r] & (1 << c)) != 0);
    food = (r << 3) | c;
    snake_row[r] |= (1 << c);
}

void EXTI2_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER1_KEY_EXTI_LINE)) {
        dct = 0;
    }
    exti_interrupt_flag_clear(USER1_KEY_EXTI_LINE);
}

void EXTI3_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER2_KEY_EXTI_LINE)) {
        dct = 1;
    }
    exti_interrupt_flag_clear(USER2_KEY_EXTI_LINE);
```

```
}

void EXTI4_IRQHandler(void) {
    if (RESET != exti_interrupt_flag_get(USER3_KEY_EXTI_LINE)) {
        dct = 2;
    }
    exti_interrupt_flag_clear(USER3_KEY_EXTI_LINE);
}
```

4 实验分析

实验 1.1 通过逐步发送数据到 8x8 LED 矩阵，可以显示出王字的各个部分。每个部分对应一个 row 和 core 数组。通过定时器计数 tim3_count 控制显示的顺序和时间。

在这个实验中，主要使用了 74HC595 移位寄存器来控制 LED 矩阵的显示。通过 HC595_Send_Byte 函数将数据发送到移位寄存器，并通过 HC595_CS 函数锁存数据，从而控制 LED 矩阵的点亮状态。每个 row 和 core 数组分别表示 LED 矩阵的行和列，通过组合这些数组可以显示出不同的图案。

实验 1.2 通过 ADC 读取电压值并转换为实际电压值，然后使用数码管显示出来。使用 HC595_SelecTube 函数选择数码管的显示内容。

在这个实验中，主要使用了 ADC 模块来读取电压值，并通过数码管显示出来。通过 adc_read_data 函数读取 ADC 数据，并转换为实际电压值。然后使用 shumaguan_play 函数将电压值显示在数码管上。

实验 2 通过按键控制 DAC 输出值，并在 OLED 显示屏上显示当前 DAC 输出值。使用 dac_data_set 函数设置 DAC 输出值。

在这个实验中，主要使用了 DAC 模块来输出电压值，并通过按键控制输出值的变

化。通过 EXTI 中断处理函数来检测按键按下事件，并相应地调整 DAC 输出值。然后使用 OLED_ShowString 函数将当前 DAC 输出值显示在 OLED 显示屏上。

实验 3 通过按键控制贪吃蛇的方向，并实现贪吃蛇的移动和食物生成。使用 HC595_Send_Byte 函数更新贪吃蛇的位置。

在这个实验中，主要使用了 74HC595 移位寄存器来控制 LED 矩阵的显示，并通过按键控制贪吃蛇的移动方向。通过 EXTI 中断处理函数来检测按键按下事件，并相应地调整贪吃蛇的移动方向。然后使用 snake_step 函数实现贪吃蛇的移动逻辑，并使用 new_food 函数生成新的食物位置。。

5 实验总结

实验 1.1 展示了如何使用 8x8 LED 矩阵显示特定图案，掌握了 HC595_Send_Byte 函数的使用。

实验 1.2 展示了如何使用 ADC 读取电压值并在数码管上显示，掌握了 HC595_SelecTube 函数的使用。

实验 2 展示了如何使用按键控制 DAC 输出值，并在 OLED 显示屏上显示，掌握了 dac_data_set 函数的使用。

实验 3 展示了如何实现一个简单的贪吃蛇游戏，掌握了贪吃蛇的基本算法，主要是位运算和 HC595_Send_Byte 函数的使用。我知道了 0b111 之类的用法并不可靠，直接算出来用十进制好。