计算机科学与技术学院

嵌入式系统实验报告

（三）

姓　　名：　Banban

专　　业：　计算机科学与技术

班　　级：

学　　号：

指导教师：

2023 年 3 月 23 日

一、任务要求

结合电路原理图和示例工程项目理解独立按键和矩阵键盘操控原理。创建mdk 工程，编写程序，选择合适的模块完成硬件连接，实现如下任务：

1、 任务 1：读取 4 个独立按键，对应控制 4 个 led 灯，键按下则灯状态翻转

2、 任务 2：读取 4 个独立按键，控制 8 个流水灯产生 4 种不同显示花样。

3、 任务 3（选做）：读取 4x4 矩阵键盘，从上到下，从左到右，依次定义为 0-9，A-F，按下相应按键，则在 8 段数码管上显示对应数符。

二、实验报告要求

1、 任务 1-任务 2 中自编程序的源代码（加上注释）

2、 能说明软件仿真结果的截图、反映硬件电路连接和硬件验证结果的图片或视频

3、 任务 3 的源码、注释、调试验证结果（选做）

三、实验过程

一. 任务一：读取 4 个独立按键，对应控制 4 个 led 灯

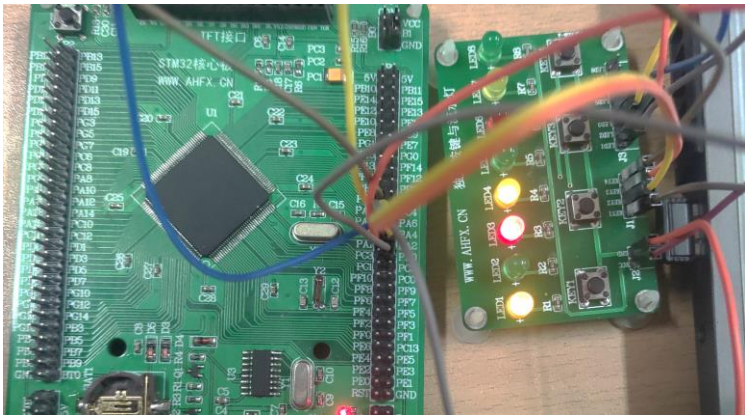
1. 代码

```
// key.c
// -------------------------------------------------------------
#include "key.h"

#define KEY_ON  0
#define KEY_OFF 1

void Delay(__IO u32 nCount) {
    for(; nCount != 0; nCount--);
}

// 配置按键用到的 I/O 口
```

```c
void Key_GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;

    /* 开启按键端口 PE 的时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3;
    // GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOE, &GPIO_InitStructure);
}

// 检测是否有按键按下
// KEY_OFF(没按下按键)、KEY_ON（按下按键）
uint8_t Key_Scan(GPIO_TypeDef* GPIOx,u16 GPIO_Pin) {
    /* 检测是否有按键按下 */
    if(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON ) {
        Delay(10000); /* 延时消抖 */
        if(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON ) {
            /* 等待按键释放 */
            while(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON);
            return  KEY_ON;
        }
        else return KEY_OFF;
    }
    else return KEY_OFF;
}



// main.c
// ----------------------------------------------------------
#include "key.h"

void Key_GPIO_Config(void){
```

```
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOE, &GPIO_InitStructure);
}

int main(void){
    GPIO_InitTypeDef GPIO_InitStructure;

    /* 配置 4 个 LED 灯的引脚为输出模式 */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    /* 配置 4 个独立按键的引脚为输入模式 */
    Key_GPIO_Config();

    while(1){
        /* 读取 4 个独立按键状态并根据按键编号控制相应的 LED */
        if(Key_Scan(GPIOE, GPIO_Pin_0) == KEY_ON)
            GPIO_SetBits(GPIOC, GPIO_Pin_0);
        if(Key_Scan(GPIOE, GPIO_Pin_1) == KEY_ON)
            GPIO_SetBits(GPIOC, GPIO_Pin_1);
        if(Key_Scan(GPIOE, GPIO_Pin_2) == KEY_ON)
            GPIO_SetBits(GPIOC, GPIO_Pin_2);
        if(Key_Scan(GPIOE, GPIO_Pin_3) == KEY_ON)
            GPIO_SetBits(GPIOC, GPIO_Pin_3);
    }
}
```

2. 图片效果



二. 任务二：读取 4 个独立按键，控制 8 个流水灯产生 4 种不同显示花样

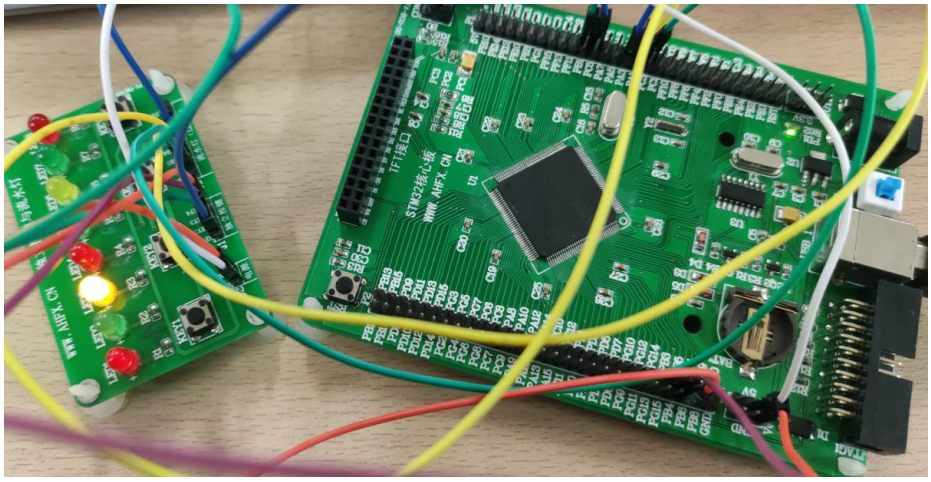
1. 代码

```
// main.c
// ----------------------------------------------------------
#include "stm32f10x.h"
#include "delay.h"
#include "key.h"

GPIO_InitTypeDef GPIO_InitStructure;
void LED_GPIO_Config(void){
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);  // 使能 PORTC 时钟
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;  // 输出模式
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);  // 初始化 GPIOC
}

void Key_GPIO_Config(void){
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE); // 使能 PORTE 时钟
```

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 |
GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;  // 输入模式，带上拉电阻
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOE, &GPIO_InitStructure);  // 初始化 GPIOE
}

// 定义 8 个变量表示 8 个流水灯灯状态，并编写函数控制流水灯产生不同的显示效果
u8 led_state[8] = {0, 0, 0, 0, 0, 0, 0, 0};

void LED_Display(void){
    u8 bits[8] = {0, 0, 0, 0, 0, 0, 0, 0};
    int i;
    for(i = 0; i < 8; i++){
        bits[i] = (led_state[i] << i);
    }
    // 以下是不同的流水灯显示模式
    LED_Display(bits); delay_ms(100);
    LED_Display(bits + 1); delay_ms(100);
    LED_Display(bits + 3); delay_ms(100);
    LED_Display(bits + 4); delay_ms(100);
}
// 读取 4 个独立按键状态，根据按键状态改变 8 个流水灯的状态。
int main(void) {
    LED_GPIO_Config();
    Key_GPIO_Config();
    while (1) {
        //读取四个独立按键状态
        //根据按键状态改变八个流水灯的状态
        if(Key_Scan(GPIOE, GPIO_Pin_5) == KEY_ON)
            led_state[0] = ~led_state[0]
        if(Key_Scan(GPIOE, GPIO_Pin_4) == KEY_ON)
            led_state[1] = ~led_state[1]
        if(Key_Scan(GPIOE, GPIO_Pin_3) == KEY_ON)
            led_state[2] = ~led_state[2]
```

```
        if(Key_Scan(GPIOE, GPIO_Pin_2) == KEY_ON)
            led_state[3] = ~led_state[3]
        LED_Display();
    }
}
```

2. 图片效果



三. 任务三（选做）：读取 4x4 矩阵键盘，按下相应按键数码管上显示对应数符

1. 代码

```
// KeyboardValue.c
// ------------------------------------------------------------
#include "stm32f10x.h"
#include "KeyBoard.h"
#include "Delay.h"

#define COL_ALL GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3
#define ROW_ALL GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7
void KeyBoard_Init(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    GPIO_InitStructure.GPIO_Pin = COL_ALL; // pa0--pa3 对应从左到右四根列线
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;// 置为推挽输出模式
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = ROW_ALL; // pa4--pa7 对应从上到下四根行线
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;// 置为下拉输入模式
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_SetBits(GPIOA, COL_ALL);// 列线全部置 1
}
/*
   4*4 的矩阵键盘
   从上到下，从左到右
   键号依次为 0,1,2,3,
             4,5,6,7,
             8,9,A,B,
             C,D,E,F
*/
s16 Read_KeyValue(void){
    s16 KeyValue = -1; // 无按键
    if((GPIO_ReadInputData(GPIOA)&0xff)!=0x0f) {
    // 列线全部置 1 后读入的行线值不等于全 0，则表明有键按下
        delay_us(2); // 软件消抖
        // 延时等待后如果读入的行值仍然不全等于 0，则确认有键按下
        if((GPIO_ReadInputData(GPIOA)&0xff) != 0x0f) {

            // 仅第 1 列列线置 1,检查按键是否位于第 1 列
            GPIO_SetBits(GPIOA, GPIO_Pin_0);
            // 第 2、3、4 列列线置 0
            GPIO_ResetBits(GPIOA, GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3);
            // 获取 PA 口低 8 位 PA[7..4]PA[3..0]
            switch(GPIO_ReadInputData(GPIOA)&0xff) {
                //PA[7..4]PA[3..0]=0001_0001 表明第 1 行第 1 列之间的按键闭合
                case 0x11: KeyValue = 0; break;
                //PA[7..4]PA[3..0]=0010_0001 表明第 2 行第 1 列之间的按键闭合
```

```
    case 0x21: KeyValue = 4; break;
    //PA[7..4]PA[3..0]=0100_0001 表明第 3 行第 1 列之间的按键闭合
    case 0x41: KeyValue = 8; break;
    //PA[7..4]PA[3..0]=1000_0001 表明第 4 行第 1 列之间的按键闭合
    case 0x81: KeyValue = 0x0C; break;
}

//仅第 2 列列线置 1，检查按键是否位于第 2 列
GPIO_SetBits(GPIOA, GPIO_Pin_1);
//第 1、3、4 列列线置 0
GPIO_ResetBits (GPIOA, GPIO_Pin_0 | GPIO_Pin_2 | GPIO_Pin_3);
//获取 PA 口低 8 位 PA[7..4]PA[3..0]
switch(GPIO_ReadInputData(GPIOA)&0xff) {
    //PA[7..4]PA[3..0]=0001_0010 表明第 1 行第 2 列之间的按键闭合
    case 0x12: KeyValue = 1; break;
    //PA[7..4]PA[3..0]=0010_0010 表明第 2 行第 2 列之间的按键闭合
    case 0x22: KeyValue = 5; break;
    //PA[7..4]PA[3..0]=0100_0010 表明第 3 行第 2 列之间的按键闭合
    case 0x42: KeyValue = 9; break;
    //PA[7..4]PA[3..0]=1000_0010 表明第 4 行第 2 列之间的按键闭合
    case 0x82: KeyValue = 0x0D; break;
}
//仅第 3 列列线置 1，检查按键是否位于第 3 列
GPIO_SetBits(GPIOA, GPIO_Pin_2);
GPIO_ResetBits(GPIOA, GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_3);
switch(GPIO_ReadInputData(GPIOA)&0xff) {
    case 0x14: KeyValue = 2; break;
    case 0x24: KeyValue = 6; break;
    case 0x44: KeyValue = 0x0A; break;
    case 0x84: KeyValue = 0x0E; break;
}
//仅第 4 列列线置 1，检查按键是否位于第 4 列
GPIO_SetBits(GPIOA, GPIO_Pin_3);
GPIO_ResetBits(GPIOA, GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2);
switch(GPIO_ReadInputData(GPIOA)&0xff){
```

```
                case 0x18: KeyValue = 3; break;
                case 0x28: KeyValue = 7; break;
                case 0x48: KeyValue = 0x0B; break;
                case 0x88: KeyValue = 0x0F; break;
            }
            GPIO_SetBits(GPIOA, COL_ALL);  // 全部列线置 1
            while((GPIO_ReadInputData(GPIOA)&0xff)!=0x0f);  // 等待按键释放
            return KeyValue;
        }
    }
    return -1; // 没有键按下
}


// main.c
// -------------------------------------------------------------
int main(void) {
    KeyBoard_Init();
    // 共阴，数字 0-9、A-F 的 7 段显示代码
    const uint8_t SEG_CODES[16] = {
        0x3F, 0x06, 0x5B, 0x4F,
        0x66, 0x6D, 0x7D, 0x07,
        0x7F, 0x6F, 0x77, 0x7C,
        0x39, 0x5E, 0x79, 0x71
    };

    // 初始化数码管，配置 GPIOE0-E7 为推挽输出
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE);  // 启用 GPIOB 时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);  // 使能 PE 口时钟
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1| GPIO_Pin_2 |
GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOE, &GPIO_InitStructure);
    GPIO_SetBits(GPIOE, GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 |
GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7);
```

```
    uint8_t key_val = 0;  // 当前按键值
    while(1) {
        key_val = Read_KeyValue();
        if(key_val >= 0) {  // 如果按下一个键
            GPIO_Write(GPIOE, SEG_CODES[key_val]);   // 亮
        } else {  // 如果没有按键被按下
            GPIO_Write(GPIOE, 0x00);  // 灭
        }
    }
    return 0;
}
```

四、总结与分析

在老师的帮助下，本次实验得以顺利完成。本任务一中我通过使用 GPIOC 来控制 4 个 LED 灯的输出，使用 GPIOE 读取 4 个独立按键的状态。根据按键的编号（从 E2 到 E5），控制相应的 LED 输出高电平。另外在主循环中使用适当的延迟以避免按键抖动。

在任务二中我根据 4 个独立按键改变 8 个流水灯的状态，使用 GPIOC 来控制 4 个 LED 灯的输出，产生 4 种不同的显示效果，在代码中通过 led_state[8]控制了 8 个 LED 灯，并使用延迟函数以避免闪烁。实验达到预期效果。