嵌入式系统基础　　实验报告

评 语	成绩	
教 师： 年　　月　　日		

学院班级：　计算机科学与技术学院 52 班

学生学号：　19030500200

学生姓名：　张璟

实验日期：　2021 年 10 月 19 日

1 实验目的

1. 熟悉 STM32F103 系列开发板的硬件环境
2. 掌握使用 Keil uVision5 进行 STM32 程序开发
3. 熟练硬件开发板和 Keil 软件环境联调的方法

2 实验环境

本次实验基于野火指南者（STM32F103VET6）开发板。使用 VS Code 和 KEIL uVision5（ARMCC）作为开发、编译环境。

2.1 配置编译环境

首先需要配置编译和下载环境。由于开发时需要使用官方提供的固件库进行编程，因此需要配置好编译链相关参数。

首先，在 KEIL 的编译设置模块添加预定义参数：

```
STM32F10X_HD
USE_STDPERIPH_DRIVER
```

接着将相关库函数加入项目中，设置 include 路径：

```
.\CMSIS;
.\FWLIB\inc;
.\USER
```

结束后项目文件不会再产生错误提示，但是如果进行编译却会报错，**这里需要注意将编译器改为第五版的，即在 Target 选项中选择 ARM 的编译器为**`default version 5`。

2.2 配置下载环境

为了提供对应的下载环境，也需要对应设置相关参数。本节内容参考野火指南者开发板使用说明。

选择使用 Debugger 模块中的`CMSIS DAP Debugger`，Target 模块中启用`Use MicroLIB`。此外，在 Debugger 的设置中要将`Connect:`设置为`under Reset`，模式设置为`HW Reset`。这意味着将需要我们手动 Reset 硬件。

其余保持默认即可。

3 相关辅助函数的定义

这一章所完成的代码主要为后续实验提供辅助作用。

3.1 LED 灯的初始化

要使用小灯来完成实验，第一步是查阅相关手册，确定小灯使用的 IO 引脚。由图一可知，该开发板使用的是 GPIOB 的引脚 0、引脚 1、引脚 5。分别对应 RGB 的三种颜色，低电平点亮，高电平熄灭。

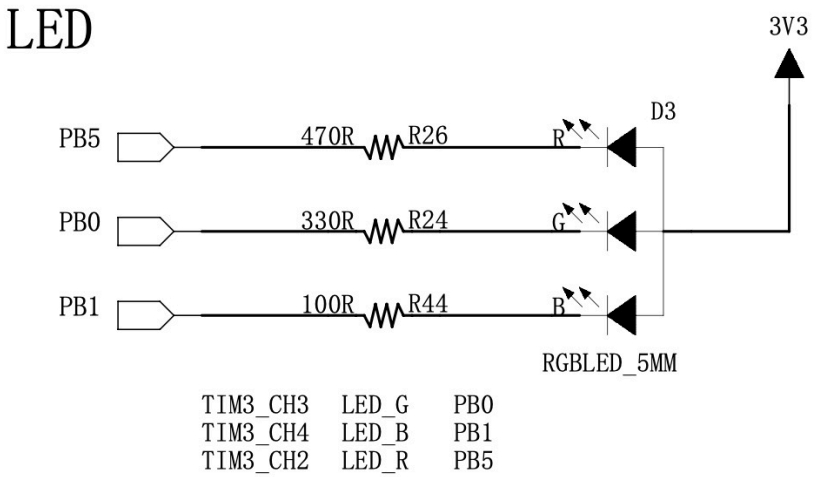

图 1: LED 灯的引脚

因此我们首先分别定义对应的宏：

```
// Red:
#define LED_RED_GPIO_BASE GPIOB
#define LED_RED_GPIO_CLK RCC_APB2Periph_GPIOB
#define LED_RED_GPIO_PIN GPIO_Pin_5

// Blue:
#define LED_BLUE_GPIO_BASE GPIOB
#define LED_BLUE_GPIO_CLK RCC_APB2Periph_GPIOB
#define LED_BLUE_GPIO_PIN GPIO_Pin_1

// Green:
#define LED_GREEN_GPIO_BASE GPIOB
#define LED_GREEN_GPIO_CLK RCC_APB2Periph_GPIOB
#define LED_GREEN_GPIO_PIN GPIO_Pin_0
```

其中，`XXX_CLK`是初始化 RCC 时钟使用的，`XXX_PIN`则是上述对应的三个引脚。

之后我们就可以来定义我们的初始化函数了：

```
void init_led(void) {
    /*定义一个GPIO_InitTypeDef类型的结构体*/
    GPIO_InitTypeDef GPIO_InitStructure;

    /*开启LED相关的GPIO外设时钟*/
    RCC_APB2PeriphClockCmd(LED_RED_GPIO_CLK |
```

```
                                LED_BLUE_GPIO_CLK |
                                LED_GREEN_GPIO_CLK, ENABLE);
    /*选择要控制的GPIO引脚*/
    GPIO_InitStructure.GPIO_Pin = LED_RED_GPIO_PIN;

    /*设置引脚模式为通用推挽输出*/
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

    /*设置引脚速率为50MHz */
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    /*调用库函数，初始化GPIO*/
    GPIO_Init(LED_RED_GPIO_BASE, &GPIO_InitStructure);

    /*选择要控制的GPIO引脚*/
    GPIO_InitStructure.GPIO_Pin = LED_BLUE_GPIO_PIN;

    /*调用库函数，初始化GPIO*/
    GPIO_Init(LED_BLUE_GPIO_BASE, &GPIO_InitStructure);

    /*选择要控制的GPIO引脚*/
    GPIO_InitStructure.GPIO_Pin = LED_GREEN_GPIO_PIN;

    /*调用库函数，初始化GPIOF*/
    GPIO_Init(LED_GREEN_GPIO_BASE, &GPIO_InitStructure);

    //* 打开所有led灯 */
    GPIO_ResetBits(LED_RED_GPIO_BASE, LED_RED_GPIO_PIN);
    GPIO_ResetBits(LED_BLUE_GPIO_BASE, LED_BLUE_GPIO_PIN);
    GPIO_ResetBits(LED_GREEN_GPIO_BASE, LED_GREEN_GPIO_PIN);
}
```

根据后述的功能要求，在这我们默认打开全部的 LED 灯。

3.2 电容按键的初始化

完成了 LED 灯的初始化之后，我们需要完成对按键的初始化。同样的，查阅手册，找到按键的 IO 引脚如图二所示。

可以看到 KEY1 对应的是 GPIOA 的 0 号引脚，KEY2 对应的是 GPIOC 的 13 号引脚。

我们同样进行宏定义：

```
#define KEY1_GPIO_CLK RCC_APB2Periph_GPIOA
#define KEY1_GPIO_PORT GPIOA
#define KEY1_GPIO_PIN GPIO_Pin_0

#define KEY2_GPIO_CLK RCC_APB2Periph_GPIOC
#define KEY2_GPIO_PORT GPIOC
```

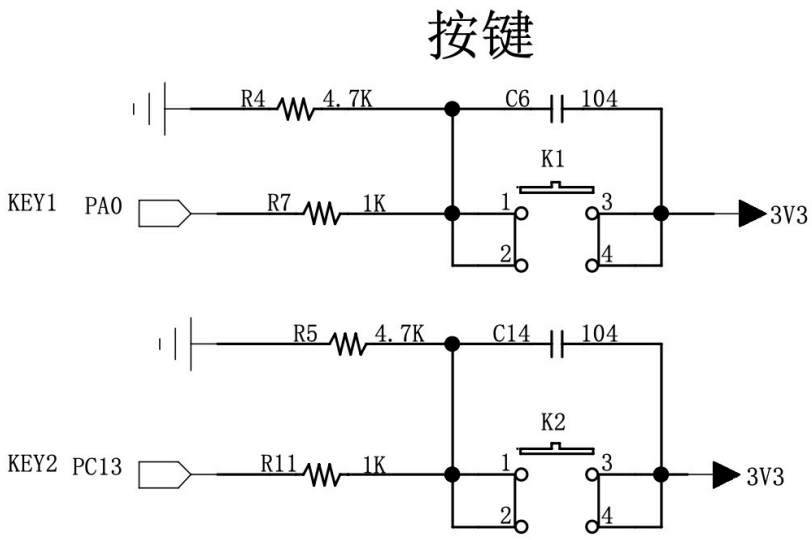

图 2: 电容按键的引脚

```
#define KEY2_GPIO_PIN GPIO_Pin_13
```

为了方便起见，这里我们再定义两个宏用来表示按键是否被按下：

```
#define KEY_ON 1
#define KEY_OFF 0
```

最后我们可以参照上述方法初始化按键。

```
void init_key(void) {
    GPIO_InitTypeDef GPIO_InitStructure;

    /*开启按键端口的时钟*/
    RCC_APB2PeriphClockCmd(KEY1_GPIO_CLK | KEY2_GPIO_CLK, ENABLE);

    //选择按键的引脚
    GPIO_InitStructure.GPIO_Pin = KEY1_GPIO_PIN;
    // 设置按键的引脚为浮空输入
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    //使用结构体初始化按键
    GPIO_Init(KEY1_GPIO_PORT, &GPIO_InitStructure);

    //选择按键的引脚
    GPIO_InitStructure.GPIO_Pin = KEY2_GPIO_PIN;
    //设置按键的引脚为浮空输入
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    //使用结构体初始化按键
    GPIO_Init(KEY2_GPIO_PORT, &GPIO_InitStructure);
}
```

4 任务一：点亮 LED 灯

根据题目要求，**LED 灯初始状态为亮，按键按下时对应 LED 灯灭，按键抬起时对应 LED 灯亮**。

由于实验环境的开发板只有一个灯泡，因此只能将对应的功能设置为初始状态为三种颜色的 LED 灯同时亮，此时为**红 + 蓝 + 绿**，即**白色**。*Key1* 按下时蓝色灯灭；*key2* 按下时绿色灯灭。蓝色灯灭时为**红 + 绿 = 黄色**，绿灯灭时为**红 + 蓝 = 紫色**。

4.1 定义接口

我们首先来定义一些 LED 的接口：

```
/* define the condition to trigger the led */
#define LED_RED(x)                                                  \
    if (x)                                                          \
        GPIO_SetBits(LED_RED_GPIO_BASE, LED_RED_GPIO_PIN);          \
    else                                                            \
        GPIO_ResetBits(LED_RED_GPIO_BASE, LED_RED_GPIO_PIN)

#define LED_BLUE(x)                                                   \
    if (x)                                                            \
        GPIO_SetBits(LED_BLUE_GPIO_BASE, LED_BLUE_GPIO_PIN);          \
    else                                                              \
        GPIO_ResetBits(LED_BLUE_GPIO_BASE, LED_BLUE_GPIO_PIN)

#define LED_GREEN(x)                                                    \
    if (x)                                                              \
        GPIO_SetBits(LED_GREEN_GPIO_BASE, LED_GREEN_GPIO_PIN);          \
    else                                                                \
        GPIO_ResetBits(LED_GREEN_GPIO_BASE, LED_GREEN_GPIO_PIN)
```

这些接口实际上就是利用标准库对 LED 灯对应的 IO 引脚进行操作，传入的 x 是一个触发条件。接下来我们设置对应的触发条件，由于灯泡是低电平触发，所以 `ON` 设置为 0:

```
/* trigger the led */
#define ON 0
#define OFF 1
```

这样，在调用的时候我们只需要使用 `LED_RED(ON);` 就可以实现打开红色 LED 灯的功能了。

4.2 功能逻辑

要使按下按键后颜色变化，松开后还原，我们需要考虑探测按键的状态。固件库提供了函数 `GPIO_ReadInputDataBit` 来实现引脚输入状态的探测，我们针对两个按键的 IO 引脚进行检测，根据返回的结果就可以判断按键是否按下。进一步，我们还能通过信号的时间来确定单击、双击、长按等复杂操作。这里我们就单纯实现对按下这一个操作进行检测并处理的功能。

```
void press_and_show_color(void) {
    // K1 is pressed
    if (GPIO_ReadInputDataBit(KEY1_GPIO_PORT, KEY1_GPIO_PIN) == KEY_ON)
        LED_BLUE(OFF);
    else
        LED_BLUE(ON);

    // K2 is pressed
    if (GPIO_ReadInputDataBit(KEY2_GPIO_PORT, KEY2_GPIO_PIN) == KEY_ON)
        LED_GREEN(OFF);
    else
        LED_GREEN(ON);
}
```

实现的方法非常简单，只需要首先判断该按键的引脚是否被按下，若按下，则关闭对应的 LED 灯，否则就打开。

5 任务二：闪烁的 LED 灯

第二个任务是根据按键状态来控制 LED 灯的闪烁方式。这里由于只有一个 LED 灯，因此还是只能通过颜色的变化来处理。当 *key1* 按下的时候，**红、绿、蓝三种颜色交替慢速闪烁**，当 *key2* 按下的时候，**红、蓝、绿、黄、紫、青、白七种颜色交替快速闪烁**。

5.1 延时函数

首先要实现的是延时函数，这里的延时函数采用最简单的轮询方式：

```
void Delay(__IO u32 n) {
    while (n--)
        ;
}
```

为了调用方便，这里再定义一快一慢两个延时函数宏。

```
#define SLOW_DELAY Delay(0x1FFFFF)
#define QUICK_DELAY Delay(0x0DFFFF)
```

5.2 LED 的颜色

接下来要处理多种 LED 的颜色。我们知道 LED 有红、绿、蓝三种颜色，而这三种颜色进行恰当的组合就可以得到我们想要的效果。我们根据此前的基础进行包装：

```
#define RED         \
    LED_RED(ON);    \
    LED_BLUE(OFF); \
    LED_GREEN(OFF)
```

```
#define BLUE      \
    LED_RED(OFF); \
    LED_BLUE(ON); \
    LED_GREEN(OFF)

#define GREEN      \
    LED_RED(OFF);  \
    LED_BLUE(OFF); \
    LED_GREEN(ON)

#define YELLOW     \
    LED_RED(ON);   \
    LED_BLUE(OFF); \
    LED_GREEN(ON)

#define PURPLE     \
    LED_RED(ON);   \
    LED_BLUE(ON);  \
    LED_GREEN(OFF)

#define CYAN       \
    LED_RED(OFF);  \
    LED_BLUE(ON);  \
    LED_GREEN(ON)

#define WHITE      \
    LED_RED(ON);   \
    LED_BLUE(ON);  \
    LED_GREEN(ON)

#define BLACK      \
    LED_RED(OFF);  \
    LED_BLUE(OFF); \
    LED_GREEN(OFF)
```

5.3 功能逻辑

完成上面的一系列铺垫，我们的功能逻辑就呼之欲出了。写一个循环判断按键是否按下即可。

```
void press_and_show_combination_of_color(void) {
    // K1 is pressed
    while (GPIO_ReadInputDataBit(KEY1_GPIO_PORT,
                KEY1_GPIO_PIN) == KEY_ON) {
        RED;
        SLOW_DELAY;
```

```
            GREEN;
            SLOW_DELAY;

            BLUE;
            SLOW_DELAY;
        }

        // K2 is pressed
        while (GPIO_ReadInputDataBit(KEY2_GPIO_PORT,
                    KEY2_GPIO_PIN) == KEY_ON) {
            RED;
            QUICK_DELAY;

            GREEN;
            QUICK_DELAY;

            BLUE;
            QUICK_DELAY;

            YELLOW;
            QUICK_DELAY;

            PURPLE;
            QUICK_DELAY;

            CYAN;
            QUICK_DELAY;

            WHITE;
            QUICK_DELAY;

            BLACK;
            QUICK_DELAY;
        }
    }
```

6 main 函数的控制逻辑

写完了两个功能逻辑，我们接下来需要做的就是管理好两种模式，并且处理好每一次执行功能逻辑前后的初始化工作。

我们设计一个区分模式的宏：

```
#define MODE 0
```

默认 MODE 的值为 0，可以手动设置为 1。利用这个宏实现一个分支：

```
int main(void) {
    // 端口初始化
    init_led();
    init_key();

    while (1) {
#if MODE
        press_and_show_color();
#else
        WHITE;
        press_and_show_combination_of_color();
#endif
    }
}
```

对于任务一，由于我们两个按键之间不应该互相影响，因此打开和关闭 LED 灯的功能都应该放在功能函数中实现，main 函数中不进行刷新。

而对于任务二，按键之间是不能同时触发的，且如果采用与任务一样的 if 语句，会导致按键松开后的颜色保持为闪烁颜色的最后一个。为了解决这个问题，任务二的处理逻辑采用的是一个 while 循环，然后我们在 main 中进行刷新。这样就可以实现闪烁结束后自动回到白色的效果。

7 补充：如果有三个灯泡怎样实现相关的功能

本实验基于的开发板只有一个灯泡，如果有三个灯泡，希望实现相关功能，实际上也是非常好处理的。本质上说，上面提到的 LED 灯三个颜色对应的引脚，就可以看作是三个灯泡的引脚，对应我们再设置三个按键引脚。

1. 在任务一中，与上述代码类似，先点亮全部灯泡，按顺序检测 key1、2、3 是否被按下，若检测到已被按下，则 `GPIO_ResetBits(ledx_port, ledx_pin)(x = 1, 2, 3)`。因此，可以看到，这里的逻辑与我们在任务一实现的逻辑是完全一样的。
2. 在任务二中，三个灯泡同步闪烁的实现方式就是全部一起点亮或全部一起熄灭，与我们上述的 `WHITE` 和 `BLACK` 是一样的，闪烁速度的快慢由`delay`函数的轮询次数决定。而跑马灯的效果则是一个灯泡点亮、熄灭之后，接下去第二个灯泡点亮、熄灭。这个效果其实与我们在任务二中实现的效果其实是一样的。任务二中，我们实现了 LED 灯红、蓝、绿三种颜色交替闪烁的效果，其原理就是三个 LED 引脚交替点亮、熄灭。

8 总结

本次实验是第一次实验，对我而言是对嵌入式开发了解、入门的一步。通过对嵌入式开发板的了解，我加深了对嵌入式开发的理解。对开发板上外设的调用实际上就是对通用 IO 寄存器的调用，更底层来说便是对具体引脚的地址进行改写，引脚的高低电平决定了硬件的功能。固

件库对相关功能进行了封装，使得我们只需要简单地查询手册了解功能引脚和总线引脚，就可以轻松地完成对硬件的调用。

在真实硬件环境下实现编程，最难的还是环境的搭建。从编译器的预定义、头文件引入到后面的编译器版本太新（version 6 使用 ARMCLANG 编译器）导致固件函数中的特殊语法无法通过编译，每一次解决问题都为我积攒了大量的经验。

A 完整代码

```
/*<--  filename: led.c  -->*/
#include "led.h"

/**
 * @brief delay function, implement by empty loop
 * @param n the times of empty loop
 */
void Delay(__IO u32 n)
{
    while (n--)
        ;
}

/**
 * @brief init led
 * @param None
 * @retval None
 */
void init_led(void)
{
    /*定义一个GPIO_InitTypeDef类型的结构体*/
    GPIO_InitTypeDef GPIO_InitStructure;

    /*开启LED相关的GPIO外设时钟*/
    RCC_APB2PeriphClockCmd(LED_RED_GPIO_CLK | LED_BLUE_GPIO_CLK |
       LED_GREEN_GPIO_CLK, ENABLE);
    /*选择要控制的GPIO引脚*/
    GPIO_InitStructure.GPIO_Pin = LED_RED_GPIO_PIN;

    /*设置引脚模式为通用推挽输出*/
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

    /*设置引脚速率为50MHz */
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    /*调用库函数，初始化GPIO*/
    GPIO_Init(LED_RED_GPIO_BASE, &GPIO_InitStructure);
```

```
    /*选择要控制的GPIO引脚*/
    GPIO_InitStructure.GPIO_Pin = LED_BLUE_GPIO_PIN;

    /*调用库函数，初始化GPIO*/
    GPIO_Init(LED_BLUE_GPIO_BASE, &GPIO_InitStructure);

    /*选择要控制的GPIO引脚*/
    GPIO_InitStructure.GPIO_Pin = LED_GREEN_GPIO_PIN;

    /*调用库函数，初始化GPIOF*/
    GPIO_Init(LED_GREEN_GPIO_BASE, &GPIO_InitStructure);

    /* 打开所有led灯  */
    GPIO_ResetBits(LED_RED_GPIO_BASE, LED_RED_GPIO_PIN);
    GPIO_ResetBits(LED_BLUE_GPIO_BASE, LED_BLUE_GPIO_PIN);
    GPIO_ResetBits(LED_GREEN_GPIO_BASE, LED_GREEN_GPIO_PIN);
}

/**
 * @brief init the key
 */
void init_key(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /*开启按键端口的时钟*/
    RCC_APB2PeriphClockCmd(KEY1_GPIO_CLK | KEY2_GPIO_CLK, ENABLE);

    //选择按键的引脚
    GPIO_InitStructure.GPIO_Pin = KEY1_GPIO_PIN;
    // 设置按键的引脚为浮空输入
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    //使用结构体初始化按键
    GPIO_Init(KEY1_GPIO_PORT, &GPIO_InitStructure);

    //选择按键的引脚
    GPIO_InitStructure.GPIO_Pin = KEY2_GPIO_PIN;
    //设置按键的引脚为浮空输入
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    //使用结构体初始化按键
    GPIO_Init(KEY2_GPIO_PORT, &GPIO_InitStructure);
}

/**
 * @brief control the color of light.
 *      If K1 ia pressed, show green;
```

```
 *       if K2 is pressed, show blue;
 *       else show red.
 */
void press_and_show_color(void)
{
    // K1 is pressed
    if (GPIO_ReadInputDataBit(KEY1_GPIO_PORT, KEY1_GPIO_PIN) == KEY_ON)
        LED_BLUE(OFF);
    else
        LED_BLUE(ON);

    // K2 is pressed
    if (GPIO_ReadInputDataBit(KEY2_GPIO_PORT, KEY2_GPIO_PIN) == KEY_ON)
        LED_GREEN(OFF);
    else
        LED_GREEN(ON);
}

/**
 * @brief control the speed of color.
 *       if no button is pressed, show the combination of three color(white);
 *       if K1 is pressed, show three kinds of light one after another;
 *       if K2 is pressed, show seven colors one after another.
 */
void press_and_show_combination_of_color(void)
{
    // K1 is pressed
    while (GPIO_ReadInputDataBit(KEY1_GPIO_PORT, KEY1_GPIO_PIN) == KEY_ON) {
        RED;
        SLOW_DELAY;

        GREEN;
        SLOW_DELAY;

        BLUE;
        SLOW_DELAY;
    }

    // K2 is pressed
    while (GPIO_ReadInputDataBit(KEY2_GPIO_PORT, KEY2_GPIO_PIN) == KEY_ON) {
        RED;
        QUICK_DELAY;

        GREEN;
        QUICK_DELAY;

        BLUE;
```

```
        QUICK_DELAY;

        YELLOW;
        QUICK_DELAY;

        PURPLE;
        QUICK_DELAY;

        CYAN;
        QUICK_DELAY;

        WHITE;
        QUICK_DELAY;

        BLACK;
        QUICK_DELAY;
    }
}
```

```
/*<-- filename: led.h -->*/
#ifndef _LED_H_
#define _LED_H_

#include "stm32f10x.h"

// Delay function
void Delay(__IO u32 n);
#define SLOW_DELAY Delay(0x1FFFFF)
#define QUICK_DELAY Delay(0x0DFFFF)

/* define the IO port that used for led */
// Red:
#define LED_RED_GPIO_BASE GPIOB
#define LED_RED_GPIO_CLK RCC_APB2Periph_GPIOB
#define LED_RED_GPIO_PIN GPIO_Pin_5

// Blue:
#define LED_BLUE_GPIO_BASE GPIOB
#define LED_BLUE_GPIO_CLK RCC_APB2Periph_GPIOB
#define LED_BLUE_GPIO_PIN GPIO_Pin_1

// Green:
#define LED_GREEN_GPIO_BASE GPIOB
#define LED_GREEN_GPIO_CLK RCC_APB2Periph_GPIOB
#define LED_GREEN_GPIO_PIN GPIO_Pin_0

/* trigger the led */
```

```
#define ON 0
#define OFF 1

/* define the condition to trigger the led */
#define LED_RED(x)                                          \
    if (x)                                                  \
        GPIO_SetBits(LED_RED_GPIO_BASE, LED_RED_GPIO_PIN); \
    else                                                    \
        GPIO_ResetBits(LED_RED_GPIO_BASE, LED_RED_GPIO_PIN)

#define LED_BLUE(x)                                           \
    if (x)                                                    \
        GPIO_SetBits(LED_BLUE_GPIO_BASE, LED_BLUE_GPIO_PIN); \
    else                                                      \
        GPIO_ResetBits(LED_BLUE_GPIO_BASE, LED_BLUE_GPIO_PIN)

#define LED_GREEN(x)                                            \
    if (x)                                                      \
        GPIO_SetBits(LED_GREEN_GPIO_BASE, LED_GREEN_GPIO_PIN); \
    else                                                        \
        GPIO_ResetBits(LED_GREEN_GPIO_BASE, LED_GREEN_GPIO_PIN)

/* init function */
void init_led(void);

/* Detect the button */
#define KEY1_GPIO_CLK RCC_APB2Periph_GPIOA
#define KEY1_GPIO_PORT GPIOA
#define KEY1_GPIO_PIN GPIO_Pin_0

#define KEY2_GPIO_CLK RCC_APB2Periph_GPIOC
#define KEY2_GPIO_PORT GPIOC
#define KEY2_GPIO_PIN GPIO_Pin_13

#define KEY_ON 1
#define KEY_OFF 0

/* init the button */
void init_key(void);

/* combination of color */
#define RED         \
    LED_RED(ON);    \
    LED_BLUE(OFF); \
    LED_GREEN(OFF)

#define BLUE      \
```

```
    LED_RED(OFF); \
    LED_BLUE(ON); \
    LED_GREEN(OFF)

#define GREEN      \
    LED_RED(OFF);  \
    LED_BLUE(OFF); \
    LED_GREEN(ON)

#define YELLOW     \
    LED_RED(ON);   \
    LED_BLUE(OFF); \
    LED_GREEN(ON)

#define PURPLE     \
    LED_RED(ON);   \
    LED_BLUE(ON); \
    LED_GREEN(OFF)

#define CYAN       \
    LED_RED(OFF); \
    LED_BLUE(ON); \
    LED_GREEN(ON)

#define WHITE      \
    LED_RED(ON);   \
    LED_BLUE(ON); \
    LED_GREEN(ON)

#define BLACK      \
    LED_RED(OFF);  \
    LED_BLUE(OFF); \
    LED_GREEN(OFF)

/* lab1: show the color of led */
void press_and_show_color(void);
void press_and_show_combination_of_color(void);

#endif /* _LED_H_ */
```

```
#include "led.h"
#include "stm32f10x.h"

// 控制模式
#define MODE 0
```

```c
/**
 * @brief   主函数
 * @param   无
 * @retval  无
 */
int main(void)
{
    // 端口初始化
    init_led();
    init_key();

    while (1) {
#if MODE
        press_and_show_color();
#else
        WHITE;
        press_and_show_combination_of_color();
#endif
    }
}
```

B 效果演示

效果演示请查看：

1. 任务一：点亮小灯
2. 任务二：小灯闪烁视频演示

点击网址跳转到 GitHub 项目页面，点击 **View raw** 下载视频即可。视频文件非常小，下载速度不会太长。