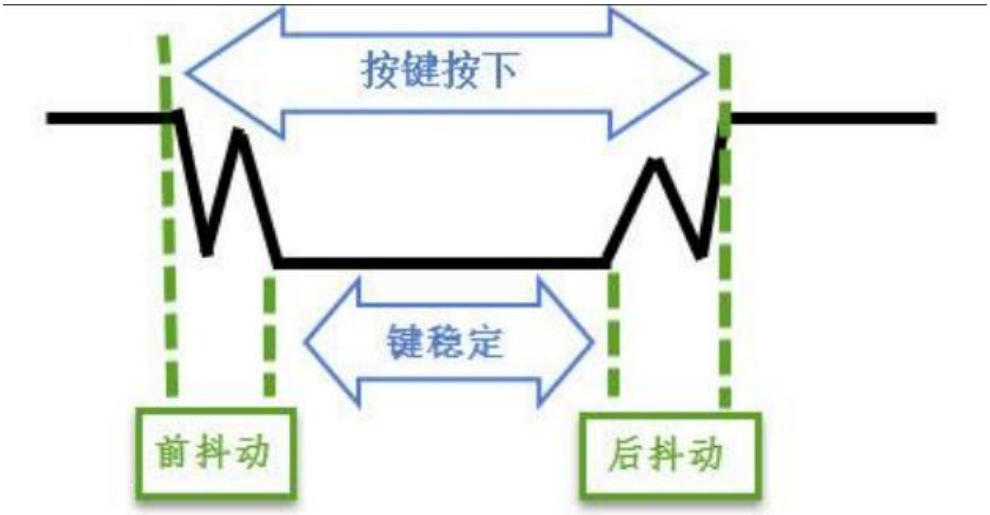
第一章 GPIO 输入按键操作

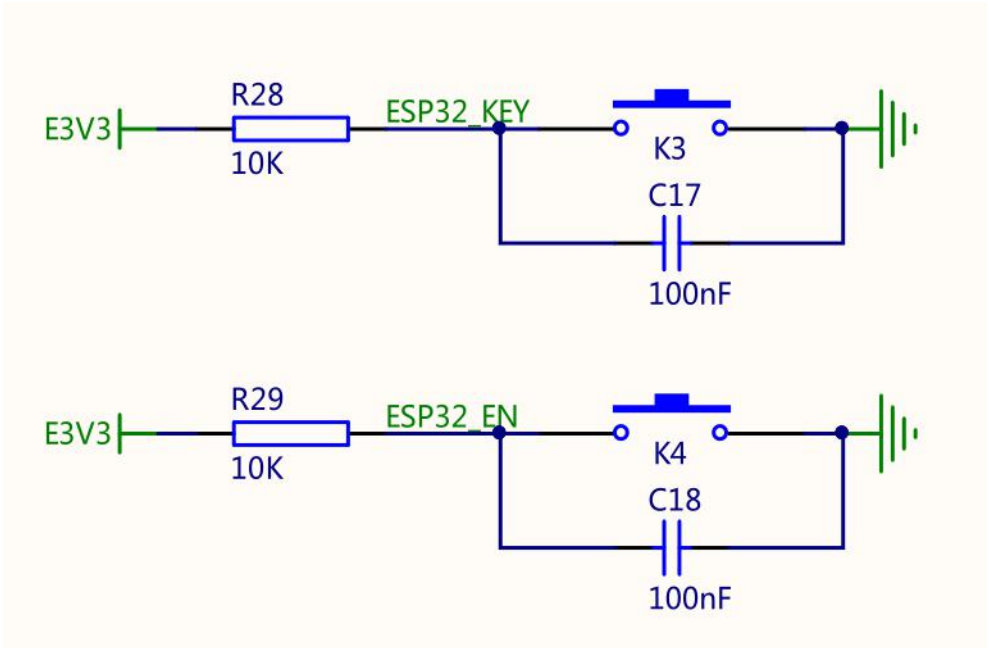
1. 学习目的及目标

- 学习轻触按键检测电路及硬件原理
- 学习 ESP32 GPIO 作为输入时候的配置
- 掌握库函数读取 GPIO 状态
- 掌握轻触按键检测程序

2. 硬件设计及原理

按键机械触点断开、闭合时，由于触点的弹性作用，按键开关不会马上稳定接通或一下子断开，使用按键时会产生下图的带波纹信号，需要用软件消抖处理滤波，不方便输入检测。本实验板连接的按键带硬件消抖功能，如下图，它利用电容充放电的延时，消除了波纹，从而简化软件的处理，软件只需要直接检测引脚的电平即可，如果效果不佳，可以再使用软件去抖。





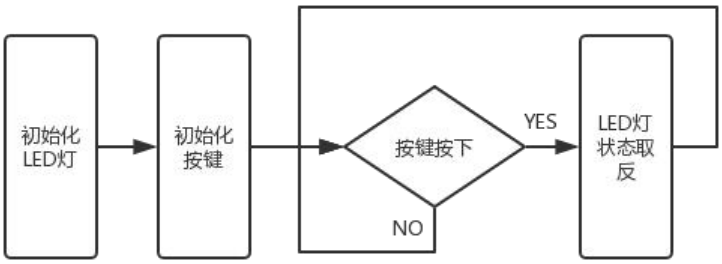
从按键的原理图可知，这些按键在没有被按下的时候，GPIO 引脚的输入状态为高电平(按键所在的电路不通，引脚通过电阻接 3.3V)，当按键按下时，GPIO 引脚的输入状态为低电平(按键所在的电路导通，引脚接到 GND)。只要我们检测引脚的输入电平，即可判断按键是否被按下。1 个用户按键占用 ESP32 的引脚如下：

按键标号	接至 ESP32 的引脚
K3	IO34

若您使用的实验板按键的连接方式或引脚不一样，只需根据我们的工程修改引脚即可，程序的控制原理相同。

3. 软件设计

3.1. 代码逻辑



3.2. ESP32 的 GPIO 接口介绍

- 设置 IO 输出值函数：int gpio_get_level(gpio_num_t gpio_num);

函数原型	int gpio_get_level (gpio_num_t gpio_num)
函数功能	读取 IO 输入值
参数	[in]gpio_num:引脚编号，0~34（存在部分）
返回值	按键的输入值 0：输入低 1：输入高

更多更详细接口请参考官方指南。

3.3. 按键按下后松手在灯切换源码编写

获取按键状态，等于 0 表示按键按下，然后死等松手，切换灯状态。

- 包含头文件、IO 宏定义、led 灯状态变量定义

```
1  #include <stdio.h>
2  #include <stdio.h>
3  #include "freertos/FreeRTOS.h"
4  #include "freertos/task.h"
5  #include "driver/gpio.h"
6  #include "sdkconfig.h"
7
```

```
#define LED_R_IO        2
#define LED_G_IO        18
#define KEY_IO 34
unsigned char led_r_status = 0;
unsigned char led_g_status = 0;

```

- 按键识别函数

```
void key_read(void)
{
    if(gpio_get_level(KEY_IO)==0)//按键按下
    {
        //等待松手，最傻的办法
        while(gpio_get_level(KEY_IO)==0);
        if (led_r_status==1)
        {
            led_r_status = 0;
            gpio_set_level(LED_R_IO, 1);//不亮
        }
        else
        {
            led_r_status = 1;
            gpio_set_level(LED_R_IO, 0);//亮
        }
    }
}
```

- 主函数：配置 IO 和调用按键识别

```
void app_main()
{
    //选择 IO
    gpio_pad_select_gpio(LED_R_IO);
    gpio_pad_select_gpio(LED_G_IO);
    gpio_pad_select_gpio(KEY_IO);
    //设置灯 IO 为输出
    gpio_set_direction(LED_R_IO, GPIO_MODE_OUTPUT);
    gpio_set_level(LED_R_IO, 1);//不亮
    gpio_set_direction(LED_G_IO, GPIO_MODE_OUTPUT);
    gpio_set_level(LED_G_IO, 1);//不亮
    //设置按键 IO 输入
    gpio_set_direction(KEY_IO, GPIO_MODE_INPUT);

    while(1) {
```

```
        key_read();//按键识别
    }
}
```

3.4. 按键按下红灯切换，松手绿灯切换源码编写

新建一个缓存 key_status[2]，保存按键的实时状态，当按键状态发送下降沿就是按键按下动作，当发生上升沿就是按键松手动作。

```
void key_read1(void)
{
    //按键识别
    if(gpio_get_level(KEY_IO)==0){
        key_status[0] = 0;
    }
    else{
       key_status[0] = 1;
    }
    if(key_status[0]!=key_status[1]) {
        key_status[1] = key_status[0];
        if(key_status[1]==0){//按键按下
            if (led_r_status==1){
                led_r_status = 0;
                gpio_set_level(LED_R_IO, 1);//不亮
            }else{
                led_r_status = 1;
                gpio_set_level(LED_R_IO, 0);//亮
            }
        }else{//按键松手
            if (led_g_status==1){
                led_g_status = 0;
                gpio_set_level(LED_G_IO, 1);//不亮
            }else{
                led_g_status = 1;
                gpio_set_level(LED_G_IO, 0);//亮
            }
        }
    }
}
```

3.5. 硬件连接

红旭开发板默认已经连接好按键，下载程序即可，使用其他开发板需要修改程序或者修改硬件连接皆可。

3.6. 效果展示

简单，如期。

4. 按键总结

- 按键去抖最好使用硬件解决掉。
- 实际项目中也可能会使用专用的按键芯片，例如 CH452、AW9523 等。
- 按键除了这种扫描方式（简单），还有中断识别（常用实用）的方式，我们再后期讲解。
- 源码地址：https://github.com/xiaolongba/wireless-tech