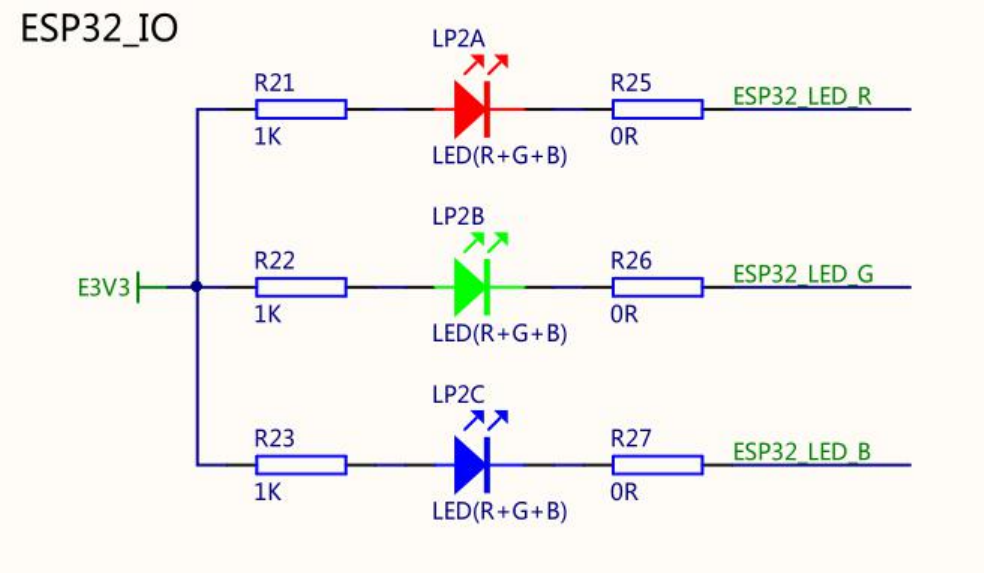
第一章 ESP32 定时器控 LED 灯

1. 学习目的及目标

- 掌握 LED 灯电路设计：控制方式
- 掌握 ESP32 定时器的库函数
- 编写 LED 闪烁灯程序

2. 硬件设计及原理

本实验板连接了一个 RGB 彩灯，RGB 彩灯实际上由三盏分别为红色、绿色、蓝色的 LED 灯组成，通过控制 RGB 颜色强度的组合，可以混合出各种色彩，此章只学习如何开关，调色放在 PWM 章学习。



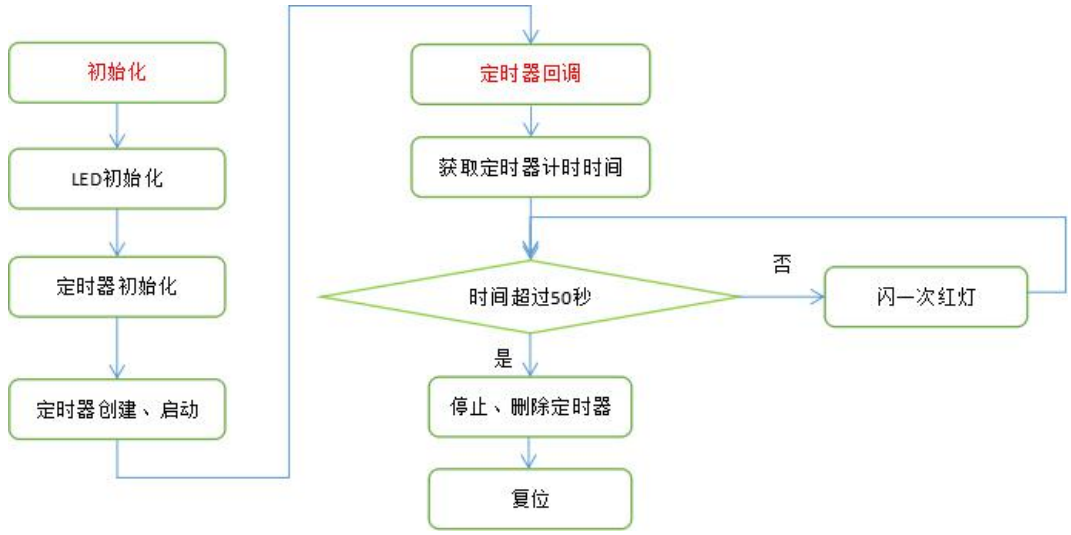
这些 LED 灯的阴极都是通过 0 欧姆电阻连接到 ESP32 的 GPIO 引脚，只要我们控制 GPIO 引脚的电平输出状态，即可控制 LED 灯的亮灭。图中去掉 0 欧姆电阻，可以切断和单片机的连接，释放这个 GPIO。3 个 LED 灯占用 ESP32 的引脚如下：

LED 标号	LED 颜色	接至 ESP32 的引脚
LP2A	红色	IO2
LP2B	绿色	IO18
LP2C	蓝色	IO19

若您使用的实验板 LED 灯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 的软定时器接口介绍

➢ 创建定时器函数：esp_timer_create();

函数原型	esp_err_t esp_timer_create (const esp_timer_create_args_t* create_args, esp_timer_handle_t* out_handle)
函数功能	创建定时器函数

参数	[in]create_args:定时器结构体
	typedef struct {
	esp_timer_cb_t callback; //定时器时间到回调
	void* arg; //要传入回调的参数
	esp_timer_dispatch_t dispatch_method; //从任务或 ISR 调用回调
	const char* name; //定时器名称，esp_timer_dump 函数使用
	} esp_timer_create_args_t;
	[in]out_handle:定时器句柄

返回值	ESP_OK：成功 ESP_ERR_INVALID_ARG ： 参数错误 ESP_ERR_INVALID_STATE：定时器已经运行

➢ 启动单次定时器函数：esp_timer_start_once();基本同下

➢ 启动周期定时器函数：esp_timer_start_periodic();

函数原型	esp_err_t esp_timer_start_periodic (esp_timer_handle_t timer, uint64_t period)
函数功能	启动周期定时器
参数	[in]timer:定时器句柄 [in]period:定时周期，单位微秒，1000 表示 1ms
返回值	ESP_OK：成功 ESP_ERR_INVALID_ARG ： 参数错误 ESP_ERR_INVALID_STATE：定时器已经运行

➢ 停止定时器函数：esp_timer_stop();

函数原型	esp_err_t esp_timer_stop (esp_timer_handle_t timer);
函数功能	停止定时器
参数	[in]timer:定时器句柄
返回值	ESP_OK：成功 ESP_ERR_INVALID_STATE：定时器已经停止

➢ 删除定时器函数：esp_timer_delete();

函数原型	esp_err_t esp_timer_delete (esp_timer_handle_t timer);
函数功能	删除定时器
参数	[in]gpio_num:引脚编号，0~34（存在部分） [in]pull:IO 模式，可以设置： 0：输出低 1：输出高
返回值	ESP_OK：成功 ESP_ERR_INVALID_ARG ： 参数错误

➢ 获取定时器时间函数：esp_timer_get_time();

函数原型	int64_t esp_timer_get_time()
函数功能	设置 IO 输出值
参数	none
返回值	自调用 esp 计时器 init 以来的微秒数（通常在应用程序启动的早期发生）

更多更详细接口请参考官方指南。

3.3. 代码编写

➢ 定时器配置

```
1   void app_main() {
2      //选择 IO
3      gpio_pad_select_gpio(LED_R_IO);
4      //设置 IO 为输出
5      gpio_set_direction(LED_R_IO, GPIO_MODE_OUTPUT);
6
7     //定时器结构体初始化
8     esp_timer_create_args_t fw_timer =
9     {
10          .callback = &fw_timer_cb,      //回调函数
11          .arg = NULL,                   //参数
```

```
        .name = "fw_timer"              //定时器名称
    };

    //定时器创建、启动
    esp_err_t err = esp_timer_create(&fw_timer, &fw_timer_handle);
    err = esp_timer_start_periodic(fw_timer_handle, 1000 * 1000);//1 秒回调
    if(err == ESP_OK)
    {
        printf("fw timer cteate and start ok!\r\n");
    }
}
```

- 定时器回调函数

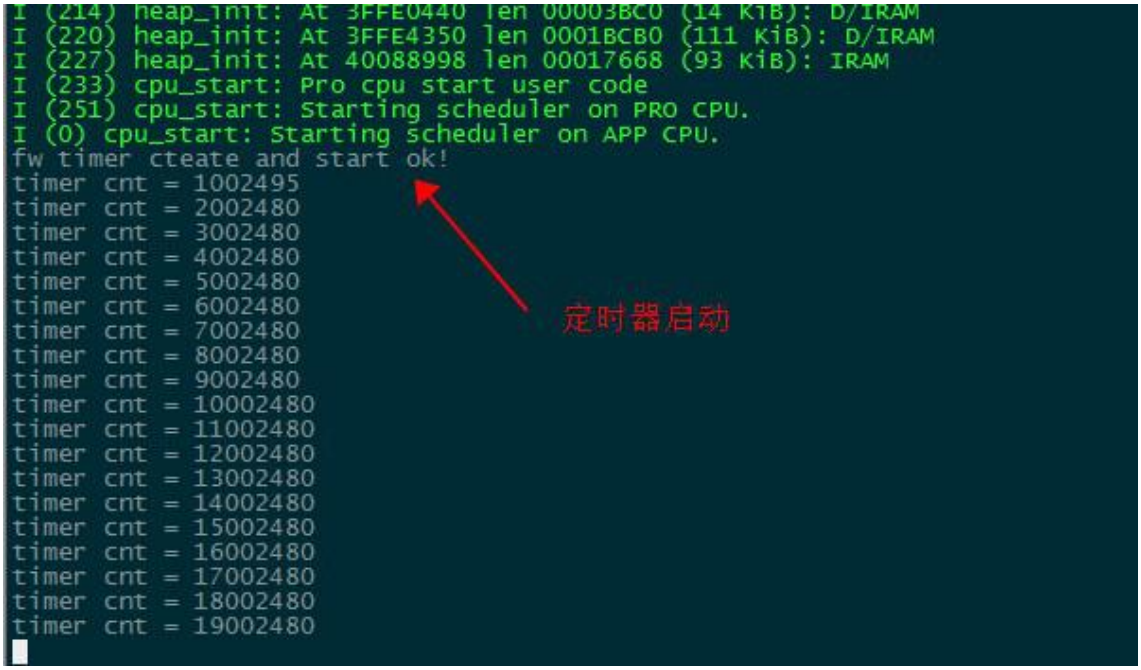
```
void fw_timer_cb(void *arg)
{
    //获取时间戳
    int64_t tick = esp_timer_get_time();
    printf("timer cnt = %lld \r\n", tick);

    if (tick > 50000000) //50 秒结束
    {
        //定时器暂停、删除
        esp_timer_stop(fw_timer_handle);
        esp_timer_delete(fw_timer_handle);
        printf("timer stop and delete!!! \r\n");
        //重启
        esp_restart();
    }

    gpio_set_level(LED_R_IO, 0);
    vTaskDelay(100 / portTICK_PERIOD_MS);
    gpio_set_level(LED_R_IO, 1);
    vTaskDelay(100 / portTICK_PERIOD_MS);

}
```

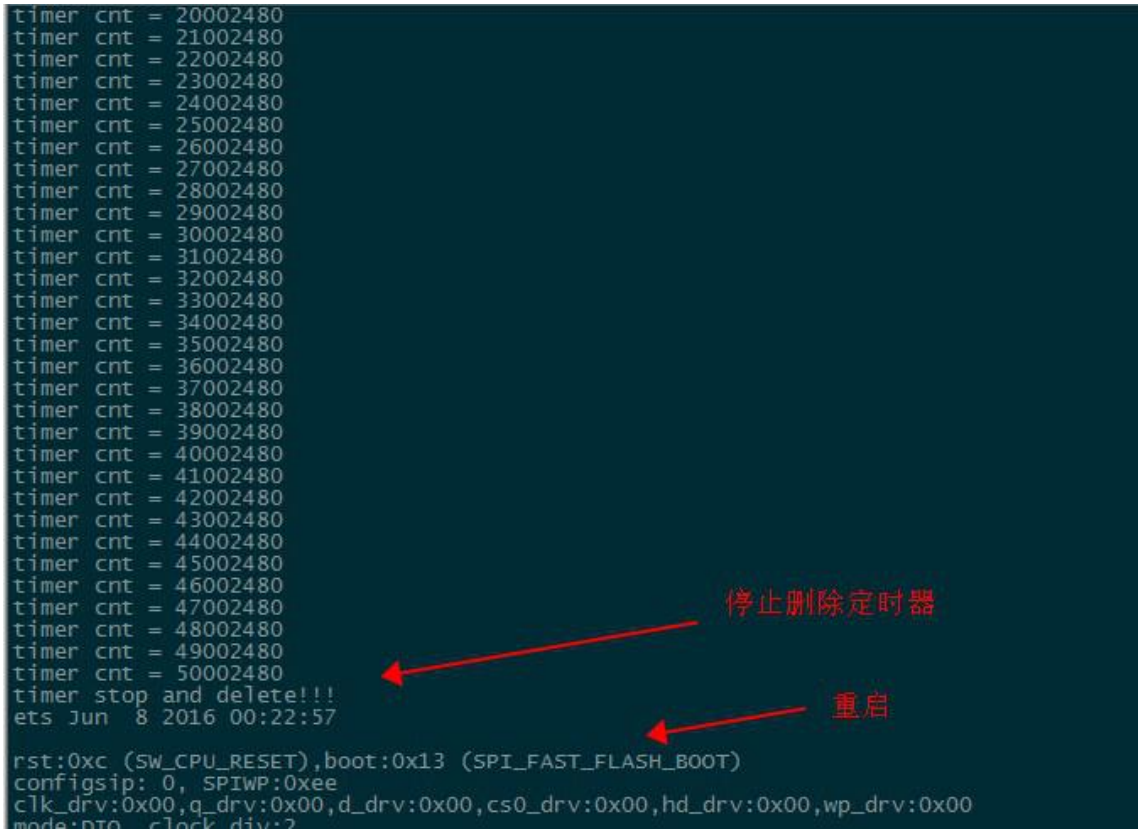
3.4. 硬件连接

红旭开发板默认已经连接好 LED，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。

3.5. 效果展示

- 红灯 1000ms 闪一次





4. 定时器总结

主要学习 ESP32 软定时器的使用，方法很简单。创建》开始》回调，可获取时间，也可以停止》暂停。

源码地址：https://github.com/xiaolongba/wireless-tech