

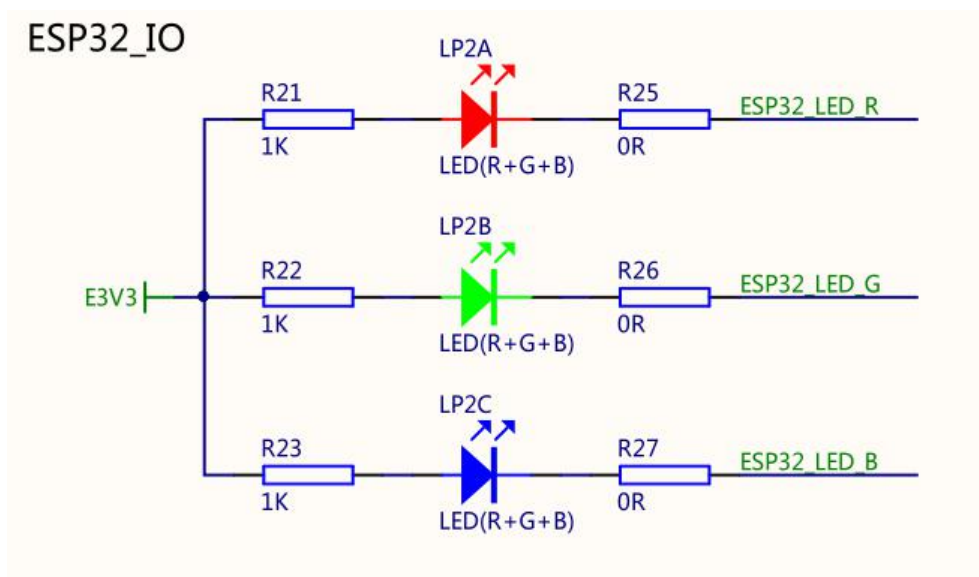
第一章 ESP32 定时器控 LED 灯

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。



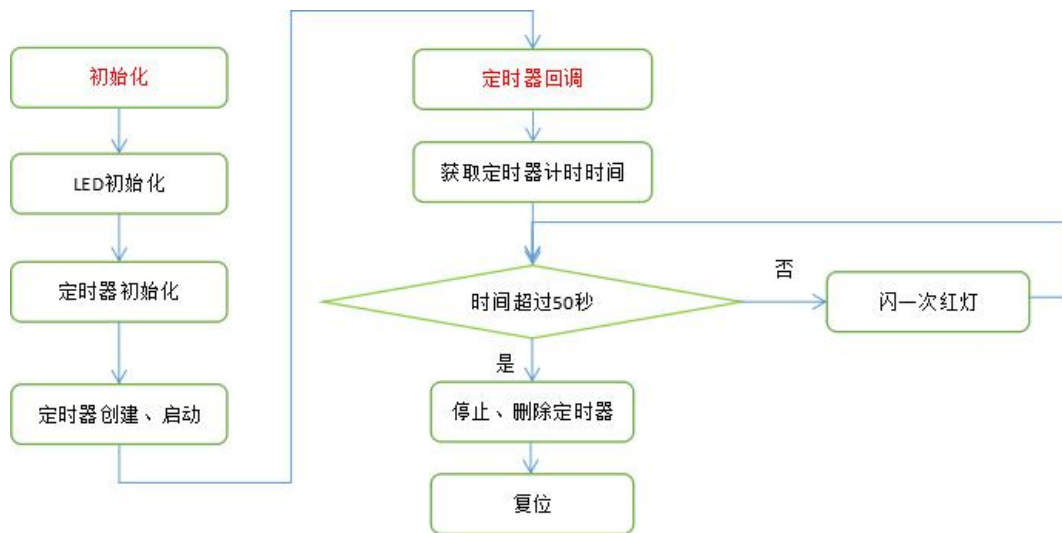
这些 LED 灯的阴极都是通过 0 欧姆电阻连接到 ESP32 的 GPIO 引脚, 只要我们控制 GPIO 引脚的电平输出状态, 即可控制 LED 灯的亮灭。图中去掉 0 欧姆电阻, 可以切断和单片机的连接, 释放这个 GPIO。3 个 LED 灯占用 ESP32 的引脚如下:

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	<pre> esp_err_t esp_timer_create (const esp_timer_create_args_t* create_args, esp_timer_handle_t* out_handle) </pre>
函数功能	创建定时器函数
参数	<p>[in]create_args:定时器结构体</p> <pre> 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; </pre> <p>[in]out_handle:定时器句柄</p>
返回值	<p>ESP_OK : 成功</p> <p>ESP_ERR_INVALID_ARG : 参数错误</p> <p>ESP_ERR_INVALID_STATE : 定时器已经运行</p>

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

➤ 停止定时器函数: `esp_timer_stop()`;

函数原型	<code>esp_err_t esp_timer_stop</code> (<code>esp_timer_handle_t timer</code>));
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	<code>esp_err_t esp_timer_delete</code> (<code>esp_timer_handle_t timer</code>));
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	<code>int64_t esp_timer_get_time()</code>
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
```

```
12         .name = "fw_timer"           //定时器名称
13     };
14
15     //定时器创建、启动
16     esp_err_t err = esp_timer_create(&fw_timer, &fw_timer_handle);
17     err = esp_timer_start_periodic(fw_timer_handle, 1000 * 1000); //1 秒回调
18     if(err == ESP_OK)
19     {
20         printf("fw timer cteate and start ok!\r\n");
21     }
22 }
```

➤ 定时器回调函数

```
1 void fw_timer_cb(void *arg)
2 {
3     //获取时间戳
4     int64_t tick = esp_timer_get_time();
5     printf("timer cnt = %lld \r\n", tick);
6
7     if (tick > 50000000) //50 秒结束
8     {
9         //定时器暂停、删除
10        esp_timer_stop(fw_timer_handle);
11        esp_timer_delete(fw_timer_handle);
12        printf("timer stop and delete!!! \r\n");
13
14        //重启
15        esp_restart();
16    }
17
18    gpio_set_level(LED_R_IO, 0);
19    vTaskDelay(100 / portTICK_PERIOD_MS);
20    gpio_set_level(LED_R_IO, 1);
21    vTaskDelay(100 / portTICK_PERIOD_MS);
22 }
```

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

```

I (214) heap_init: At 3FFE0440 len 00003BC0 (14 KiB): D/IRAM
I (220) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (227) heap_init: At 40088998 len 00017668 (93 KiB): IRAM
I (233) cpu_start: Pro cpu start user code
I (251) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
fw timer create and start ok!
timer cnt = 1002480
timer cnt = 2002480
timer cnt = 3002480
timer cnt = 4002480
timer cnt = 5002480
timer cnt = 6002480
timer cnt = 7002480
timer cnt = 8002480
timer cnt = 9002480
timer cnt = 10002480
timer cnt = 11002480
timer cnt = 12002480
timer cnt = 13002480
timer cnt = 14002480
timer cnt = 15002480
timer cnt = 16002480
timer cnt = 17002480
timer cnt = 18002480
timer cnt = 19002480
timer cnt = 20002480
timer cnt = 21002480
timer cnt = 22002480
timer cnt = 23002480
timer cnt = 24002480
timer cnt = 25002480
timer cnt = 26002480
timer cnt = 27002480
timer cnt = 28002480
timer cnt = 29002480
timer cnt = 30002480
timer cnt = 31002480
timer cnt = 32002480
timer cnt = 33002480
timer cnt = 34002480
timer cnt = 35002480
timer cnt = 36002480
timer cnt = 37002480
timer cnt = 38002480
timer cnt = 39002480
timer cnt = 40002480
timer cnt = 41002480
timer cnt = 42002480
timer cnt = 43002480
timer cnt = 44002480
timer cnt = 45002480
timer cnt = 46002480
timer cnt = 47002480
timer cnt = 48002480
timer cnt = 49002480
timer cnt = 50002480
timer stop and delete!!!
ets Jun 8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2

```

定时器启动

停止删除定时器

重启

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>