WiFi 连接

STA 模式

淘宝店铺：

PC 端：

http://n-xytrt8gqu585po94mwj5atokcyd4.taobao.com/index.htm

手机端：

https://shop.m.taobao.com/shop/shop_index.htm?sellerId=755668508&shopId=104493595&inShopPageId=423890608&pathInfo=shop/index2

资料下载地址：

链接：https://pan.baidu.com/s/1kCjD8yktZECSGmHomx_veg?pwd=q8er
提取码：q8er

源码下载地址：

https://gitee.com/vi-iot/esp32-board.git

一、ESP32 的 WiFi 功能介绍

前面章节内容，基本上都是描述了 ESP32 强大的 MCU 能力，这些 MCU 能力使得 ESP32 可以替换许多类型的单片机工作，而自己承担这部分功能；当然 ESP32 的 IOT 能力才是它的主业，从硬件配置来看，ESP32 支持 2.4GHz 频段 WiFi+BT（LE）4.2，而 esp-idf 对 WiFi 的驱动支持可谓是十分完善，我们不需要花费太多的精力研究底层实现，更多的将精力放在自己的应用设计上。

对于 ESP32 的 WIFI，有三种工作模式：

- STA 模式，这种模式是 ESP32 最常用的模式，ESP32 可以连接到任何已经存在的 WiFi 网络，从而允许 ESP32 与网络上的其他设备进行通信，类似于一台普通的 WiFi 客户端设备。
- AP 模式，这种模式下 ESP32 创建自己的 WiFi 网络，成为一个小型 WiFi 路由器，接受其它 WiFi 终端设备连接，这种模式多用于设备配网。
- STA+AP 模式，在这种模式下，ESP32 同时工作在 STA 和 AP 两种模式下，既可以连接到已有的 WiFi 网络，也可以提供 WiFi 热点，这种高级功能使 ESP32 能够连接到现有的 WiFi 网络，同时创建自己的网络，充当桥接器或中继器，通俗的讲就是连接到一个热点后，ESP32 自己又创建了一个网络，允许其它设备接入，这些设备以 ESP32 为中继节点，可以访问互联网。

二、STA 模式代码

STA 模式代码位于 esp32-board/wifi，因为 esp-idf 的 WiFi 驱动已经十分的完善，所以我们不需要很多代码就能完成 WiFi 连接，看到 esp32-board/wifi/main/simple_wifi_sta.c 文件，

```
#define DEFAULT_WIFI_SSID           "wifitest"
#define DEFAULT_WIFI_PASSWORD       "12345678"
static const char *TAG = "wifi";
/** 事件回调函数
 * @param arg   用户传递的参数
 * @param event_base    事件类别
 * @param event_id      事件 ID
 * @param event_data    事件携带的数据
 * @return 无
*/
static void event_handler(void* arg, esp_event_base_t event_base,int32_t
event_id, void* event_data)
{
    if(event_base == WIFI_EVENT)
    {
        switch (event_id)
        {
        case WIFI_EVENT_STA_START:      //WIFI 以 STA 模式启动后触发此事件
            esp_wifi_connect();         //启动 WIFI 连接
            break;
        case WIFI_EVENT_STA_CONNECTED:  //WIFI 连上路由器后，触发此事件
            ESP_LOGI(TAG, "connected to AP");
            break;
```

```
        case WIFI_EVENT_STA_DISCONNECTED:   //WIFI 从路由器断开连接后触发此
事件
            esp_wifi_connect();             //继续重连
            ESP_LOGI(TAG,"connect to the AP fail,retry now");
            break;
        default:
            break;
        }
    }
    if(event_base == IP_EVENT)                  //IP 相关事件
    {
        switch(event_id)
        {
            case IP_EVENT_STA_GOT_IP:           //只有获取到路由器分配的 IP,
才认为是连上了路由器
                ESP_LOGI(TAG,"get ip address");
                break;
        }
    }
}
//WIFI STA 初始化
esp_err_t wifi_sta_init(void)
{
    ESP_ERROR_CHECK(esp_netif_init());  //用于初始化 tcpip 协议栈
    ESP_ERROR_CHECK(esp_event_loop_create_default());       //创建一个默
认系统事件调度循环，之后可以注册回调函数来处理系统的一些事件
    esp_netif_create_default_wifi_sta();    //使用默认配置创建 STA 对象

    //初始化 WIFI
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    //注册事件
    ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT,ESP_EVENT_ANY_
ID,&event_handler,NULL));
    ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT,IP_EVENT_STA_GOT
_IP,&event_handler,NULL));
    //WIFI 配置
    wifi_config_t wifi_config =
    {
        .sta =
        {
            .ssid = DEFAULT_WIFI_SSID,              //WIFI 的 SSID
            .password = DEFAULT_WIFI_PASSWORD,      //WIFI 密码
```

```
            .threshold.authmode = WIFI_AUTH_WPA2_PSK,    //加密方式

            .pmf_cfg =
            {
                .capable = true,
                .required = false
            },
        },
    };

    //启动 WIFI
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA) );          //设置工
作模式为 STA
    ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA,
&wifi_config) );    //设置 wifi 配置
    ESP_ERROR_CHECK(esp_wifi_start() );                          //启动
WIFI

    ESP_LOGI(TAG, "wifi_init_sta finished.");
    return ESP_OK;
}
```

此文件只有两个函数，一个是 WiFi 连接事件回调函数，一个是初始化函数，先解析初始化内容。

```
    ESP_ERROR_CHECK(esp_netif_init());  //用于初始化 tcpip 协议栈
    ESP_ERROR_CHECK(esp_event_loop_create_default());       //创建一个默
认系统事件调度循环，之后可以注册回调函数来处理系统的一些事件
    esp_netif_create_default_wifi_sta();    //使用默认配置创建 STA 对象
    //初始化 WIFI
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));
```

上述这部分是标准的 WiFi 底层初始化代码，包含网卡初始化、创建事件循环系统、STA 默认配置、WIFI 默认配置，这部分代码不用去动它，这些代码涉及到底层的初始化，感兴趣的同学可以看看 WIFI_INIT_CONFIG_DEFAULT()这个默认配置到底配置了什么。

然后调用 esp_event_handler_register 函数，注册回调函数处理各种 WIFI 事件（WIFI_EVENT）的以及 IP 事件（IP_EVENT）。ESP_EVENT_ANY_ID 参数表示任何的 WIFI 事件都执行回调函数，IP_EVENT_STA_GOT_IP 表示获取到 IP 地址才会执行回调函数。

wifi_config_t wifi_config 定义了 WiFi 连接配置，这个配置里面我们填入了要连接的 SSID 和密码，还有加密方式，目前大部分路由器都支持 WPA2-PSK 方式，pmf_cfg 这个成员表示对 WiFi 保护管理帧的配置，.capable 成员不推荐使用，设置为 true 即可，表示 ESP32 一直使用保护管理帧(pmf)方式连接，.required 表示是否只与支持保护管理帧(pmf)的设备进行连接。如果大家不知道这里是什么意思，按我这里设定就行，esp-idf 官方例程也是如此。

esp_wifi_set_mode(WIFI_MODE_STA)表示 WiFi 工作在 STA 模式

esp_wifi_set_config(WIFI_IF_STA, &wifi_config)表示将设置 WiFi 连接配置

esp_wifi_start()表示启动 wifi

在启动 WiFi 后，WiFi 驱动程序就会用配置的 SSID 和密码去尝试连接路由器，如果连接成功会触发 WiFi 事件 WIFI_EVENT_STA_CONNECTED，我们在回调函数中可以将这个事件打印出来，但这个事件不代表可以连接网络，仅仅表示已经连接了 AP 热点，只有我们获取到了 IP 事件 IP_EVENT_STA_GOT_IP，这个事件是路由器给 ESP32 分配了地址，到这才认为是已经连接到了网络，当然这个也是要求你的路由器能连接互联网。

主函数实现如下：

```
void app_main(void)
{
    //NVS 初始化（WIFI 底层驱动有用到 NVS，所以这里要初始化）
    nvs_flash_init();
    //wifi STA 工作模式初始化
    wifi_sta_init();
    while(1)
    {
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

由于 WiFi 底层驱动使用了 nvs 来一份默认可连接的 SSID 和密码，因此我们需要先初始化 nvs。