

第一章 SmartConfig 一键配置

1. 学习目的及目标

- 掌握 SmartConfig 原理和工作过程
- 掌握乐鑫 ESP32 的 SmartConfig 的程序设计

2. WIFI 设备配网方法

- WiFi 设备处于 AP 模式，配置工具连上这个 AP，对这个 AP 发送联网信息，WiFi 设备收到后，切换到 STA 模式，利用收到的信息联网。此方法优势是成功率基本在 100%，缺点是配置过程复杂，做出的产品不易操作。
- SmartConfig 模式，采用 UDP 广播模式（UDP 接收 IP 地址是 255.255.255.255）。WiFi 设备先 scan 环境下 AP，得到 AP 的相关信息，如工作的 channel，然后配置 WiFi 芯片工作在刚才 scan 到的 channel 上去接收 UDP 包，如果没有接收到，继续配置工作在另外的 channel 上，如此循环，直到收到 UDP 包为止。随意此种办法的致命缺点是成功率只有 70%，而且有些路由器不支持；优点显而易见，一键完成。

3. SmartConfig 特点

SmartConfig 又名快连，当前设备在没有和其他设备建立任何实际性通信链接的状态下，一键配置该设备接入 WIFI。

3.1. SmartConfig 实际应用场景

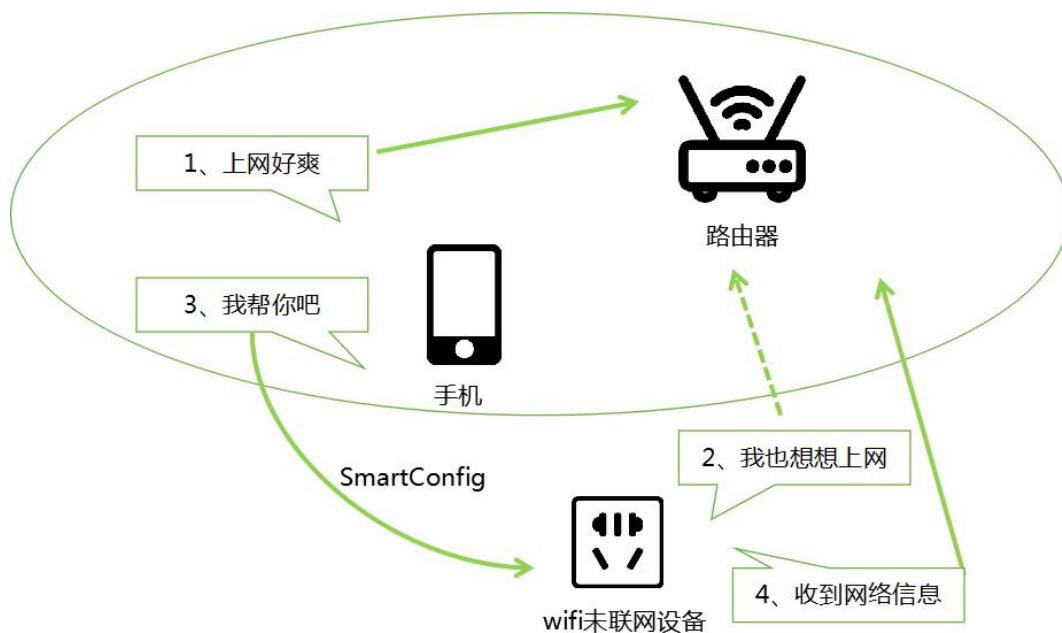
现状：当手机端接入路由器的 WIFI，未联网设备没有任何实质性通信链接（信息孤立）。

目的：如果未联网设备也想接入这个路由器的 WIFI。

分析：肯定需要有人告诉未联网设备，路由器的 WIFI 的账号(ssid)和密码(password)。

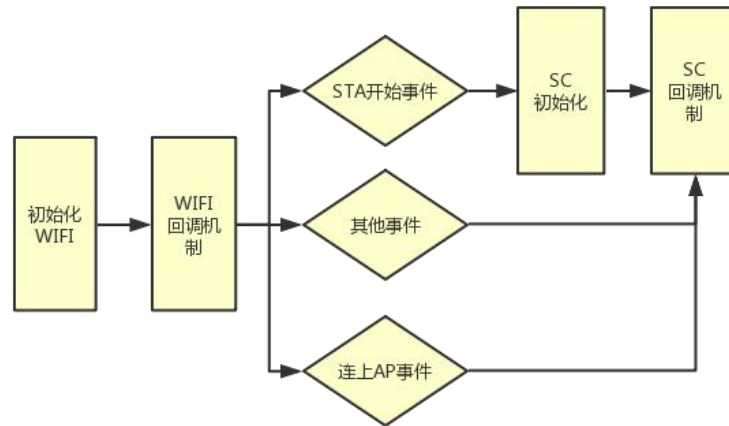
手段：目前我们只有手机端的资源可以利用，所以只能手机端告诉设备未联网设备。

未联网设备在没有任何链接的情况下，手机端是如何告知未联网设备信息，这个方法就是 SmartConfig。流程如下图所示：

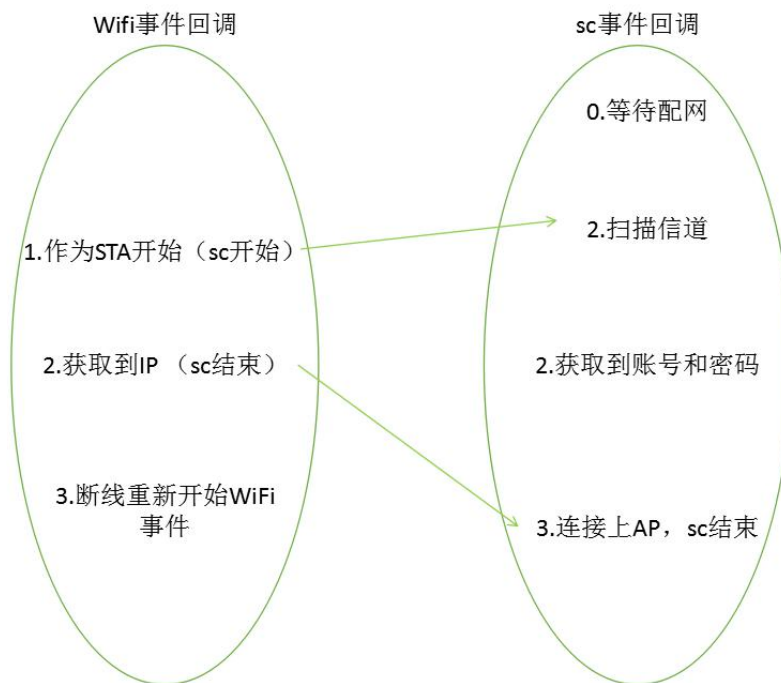


4. 软件设计

4.1. ESP32 的 SmartConfig 主逻辑



4.2. ESP32 的 SmartConfig 详细过程逻辑



4.3. ESP32 的 SmartConfig 接口介绍

➤ SmartConfig 配置类型设置函数: `esp_smartconfig_set_type()`;

回调原型	<pre> esp_err_t esp_smartconfig_set_type (smartconfig_type_t type) </pre>
函数功能	SmartConfig 配置类型设置函数
参数	<pre> [in] type, typedef enum { SC_TYPE_ESPTOUCH = 0, /*ESPTouch */ </pre>

	<pre>SC_TYPE_AIRKISS, /*AirKiss */ SC_TYPE_ESPTOUCH_AIRKISS, /*ESPTouch and AirKiss */ } smartconfig_type_t;</pre>
返回值	ESP_OK : 成功 other : 失败

➤ SmartConfig 开始一键配置函数 `esp_smartconfig_start()`;

回调原型	<pre>esp_err_t esp_smartconfig_start (sc_callback_t cb, ...)</pre>
函数功能	SmartConfig 开始一键配置函数
参数	[in] cb: smartconfig 的回调函数 [in] ...: log 1: 串口输出 log; 0: 串口直接输出结果
返回值	ESP_OK : 成功 other : 失败

➤ SmartConfig 停止一键配置函数 `esp_smartconfig_stop()`;

回调原型	<pre>esp_err_t esp_smartconfig_stop(void)</pre>
函数功能	WIFI 状态机回调函数
参数	无
返回值	ESP_OK : 成功 other : 失败

更多更详细接口请参考[官方指南](#)。

4.4. ESP32 的 SmartConfig 总结

初始化 wifi 配置后, 程序会根据 WIFI 的实时状态, 在回调函数中给出状态返回, 所以只需要在回调中进行相关操作, STA 开始事件触发 SC 开始进行一键配置, 在 SC 的回调中处理 SC 配置过程的事件。

4.5. SmartConfig 任务编写

配置工具设置->注册 smartconfig 回调函数->等待回调事件

```
1 void smartconfig_example_task(void *parm)  
2 {  
3     EventBits_t uxBits;  
4     //使用 ESP-TOUCH 配置工具  
5     ESP_ERROR_CHECK(esp_smartconfig_set_type(SC_TYPE_ESPTOUCH));  
6     //开始 sc , 注册回调  
7     ESP_ERROR_CHECK(esp_smartconfig_start(sc_callback));  
8     while (1)  
9     {
```

```
10      //死等事件组: CONNECTED_BIT | ESPTOUCH_DONE_BIT
11      uxBits = xEventGroupWaitBits(wifi_event_group, CONNECTED_BIT | ESPTOUCH_DONE_BIT, true, false,
12      portMAX_DELAY);
13
14      //sc 结束事件
15      if (uxBits & ESPTOUCH_DONE_BIT)
16      {
17          ESP_LOGI(TAG, "smartconfig over");
18          esp_smartconfig_stop();
19          //此处 smartconfig 结束, 并且连上设置的 AP, 以下可以展开自定义工作了
20
21          vTaskDelete(NULL);
22      }
23      //连上 ap 事件
24      if (uxBits & CONNECTED_BIT)
25      {
26          ESP_LOGI(TAG, "WiFi Connected to ap");
27      }
28  }
29 }
```

4.6. SmartConfig 的回调函数解析

```
1 static void sc_callback(smartconfig_status_t status, void *pdata)
2 {
3     switch (status)
4     {
5         case SC_STATUS_WAIT:                //等待配网
6             ESP_LOGI(TAG, "SC_STATUS_WAIT");
7             break;
8         case SC_STATUS_FIND_CHANNEL:         //扫描信道
9             ESP_LOGI(TAG, "SC_STATUS_FINDING_CHANNEL");
10            break;
11         case SC_STATUS_GETTING_SSID_PSWD:    //获取到 ssid 和密码
12             ESP_LOGI(TAG, "SC_STATUS_GETTING_SSID_PSWD");
13             break;
14         case SC_STATUS_LINK:                 //连接获取的 ssid 和密码
15             ESP_LOGI(TAG, "SC_STATUS_LINK");
16             wifi_config_t *wifi_config = pdata;
17             //打印账号密码
18             ESP_LOGI(TAG, "SSID:%s", wifi_config->sta.ssid);
19             ESP_LOGI(TAG, "PASSWORD:%s", wifi_config->sta.password);
20             //断开 smartconfig
21             ESP_ERROR_CHECK(esp_wifi_disconnect());
```

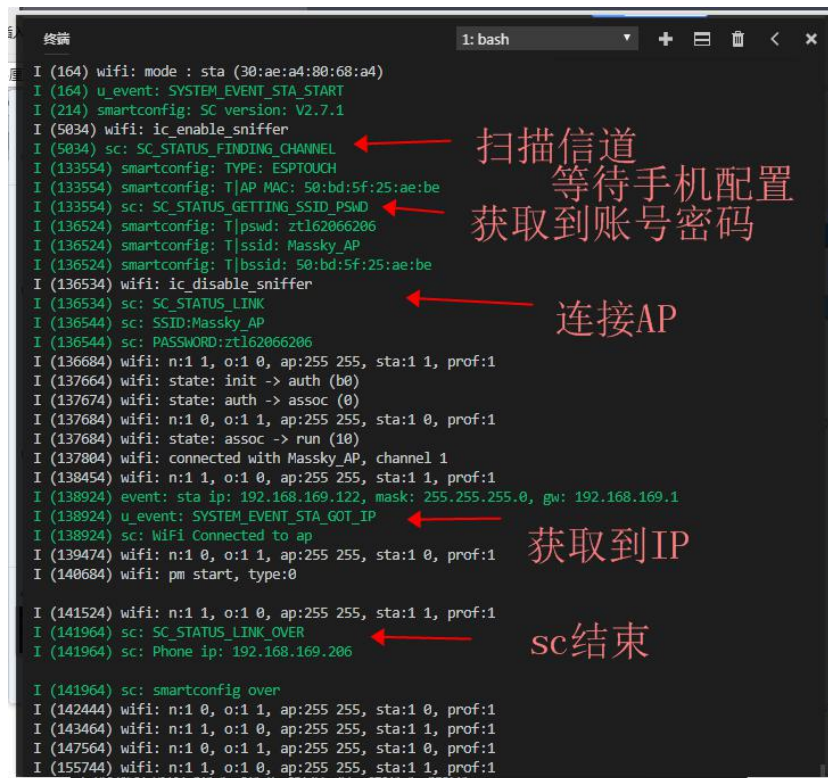
```
22      //设置获取的 ap 和密码到寄存器
23      ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, wifi_config));
24      //连接 AP
25      ESP_ERROR_CHECK(esp_wifi_connect());
26      break;
27 case SC_STATUS_LINK_OVER:          //连接上配置后, 结束
28     ESP_LOGI(TAG, "SC_STATUS_LINK_OVER");
29     //打印 ip
30     if (pdata != NULL)
31     {
32         uint8_t phone_ip[4] = {0};
33         memcpy(phone_ip, (uint8_t *)pdata, 4);
34         ESP_LOGI(TAG, "Phoneip:%d.%d.%d.%d\n", phone_ip[0], phone_ip[1], phone_ip[2], phone_ip[3]);
35     }
36     //发送 sc 结束事件
37     xEventGroupSetBits(wifi_event_group, ESPTOUCH_DONE_BIT);
38     break;
39 default:
40     break;
41 }
42 }
```

4.7. WiFi 连接的回调函数解析

```
1 static esp_err_t event_handler(void *ctx, system_event_t *event)
2 {
3     switch (event->event_id)
4     {
5     case SYSTEM_EVENT_STA_START://STA 开始工作
6         ESP_LOGI(TAG1, "SYSTEM_EVENT_STA_START");
7         //创建 smartconfig 任务
8         xTaskCreate(smartconfig_example_task, "smartconfig_example_task", 4096, NULL, 3, NULL);
9         break;
10    case SYSTEM_EVENT_STA_GOT_IP://获取 IP
11        ESP_LOGI(TAG1, "SYSTEM_EVENT_STA_GOT_IP");
12        //sta 链接成功, set 事件组
13        xEventGroupSetBits(wifi_event_group, CONNECTED_BIT);
14        break;
15    case SYSTEM_EVENT_STA_DISCONNECTED://断线
16        ESP_LOGI(TAG1, "SYSTEM_EVENT_STA_DISCONNECTED");
17        //断线重连
18        esp_wifi_connect();
19        xEventGroupClearBits(wifi_event_group, CONNECTED_BIT);
20        break;
21    default:
22        break;
23    }
```

```
24     return ESP_OK;  
25 }
```

5. 效果展示



The terminal window displays the following log messages:

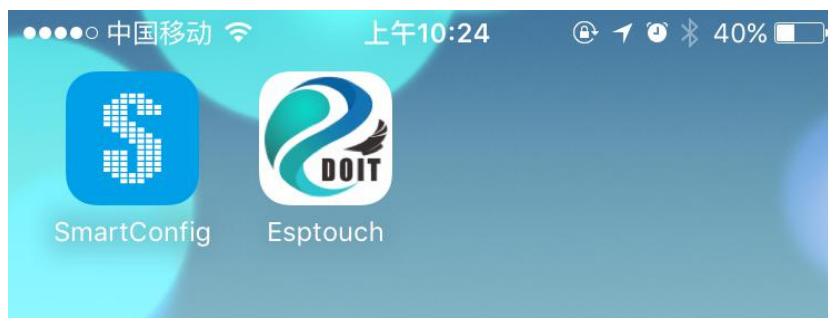
```
I (164) wifi: mode : sta (30:ae:a4:80:68:a4)  
I (164) u_event: SYSTEM_EVENT_STA_START  
I (214) smartconfig: SC version: V2.7.1  
I (5034) wifi: ic_enable_sniffer  
I (5034) sc: SC_STATUS_FINDING_CHANNEL  
I (133554) smartconfig: TYPE: ESPTOUCH  
I (133554) smartconfig: T|AP MAC: 50:bd:5f:25:ae:be  
I (133554) sc: SC_STATUS_GETTING_SSID_PSWD  
I (136524) smartconfig: T|pswd: zt162066206  
I (136524) smartconfig: T|ssid: Massky_AP  
I (136524) smartconfig: T|bssid: 50:bd:5f:25:ae:be  
I (136534) wifi: ic_disable_sniffer  
I (136534) sc: SC_STATUS_LINK  
I (136544) sc: SSID:Massky AP  
I (136544) sc: PASSWORD:zt162066206  
I (136684) wifi: n:1 1, o:1 0, ap:255 255, sta:1 1, prof:1  
I (137664) wifi: state: init -> auth (b0)  
I (137674) wifi: state: auth -> assoc (0)  
I (137684) wifi: n:1 0, o:1 1, ap:255 255, sta:1 0, prof:1  
I (137684) wifi: state: assoc -> run (10)  
I (137804) wifi: connected with Massky_AP, channel 1  
I (138454) wifi: n:1 1, o:1 0, ap:255 255, sta:1 1, prof:1  
I (138924) event: sta ip: 192.168.169.122, mask: 255.255.255.0, gw: 192.168.169.1  
I (138924) u_event: SYSTEM_EVENT_STA_GOT_IP  
I (138924) sc: WiFi Connected to ap  
I (139474) wifi: n:1 0, o:1 1, ap:255 255, sta:1 0, prof:1  
I (140684) wifi: pm start, type:0  
  
I (141524) wifi: n:1 1, o:1 0, ap:255 255, sta:1 1, prof:1  
I (141964) sc: SC_STATUS_LINK_OVER  
I (141964) sc: Phone ip: 192.168.169.206  
  
I (141964) sc: smartconfig over  
I (142444) wifi: n:1 0, o:1 1, ap:255 255, sta:1 0, prof:1  
I (143464) wifi: n:1 1, o:1 0, ap:255 255, sta:1 1, prof:1  
I (147564) wifi: n:1 0, o:1 1, ap:255 255, sta:1 0, prof:1  
I (155744) wifi: n:1 1, o:1 0, ap:255 255, sta:1 1, prof:1
```

Annotations in the image:

- 扫描信道 (Scan channel) points to `SC_STATUS_FINDING_CHANNEL`.
- 等待手机配置 (Waiting for phone configuration) points to `SC_STATUS_GETTING_SSID_PSWD`.
- 获取到账号密码 (Obtained account password) points to `T|pswd: zt162066206`.
- 连接AP (Connect to AP) points to `SC_STATUS_LINK`.
- 获取到IP (Obtained IP) points to `SYSTEM_EVENT_STA_GOT_IP`.
- sc结束 (SC ended) points to `SC_STATUS_LINK_OVER`.

6. SmartConfig 总结

- SC 工作模式也是初始化+回调的状态机结构，逻辑清晰简单，方便学习和使用。
- Wifi 回调和 SC 回调相互协调工作，保证产品顺利配网成功。
- 配网工具有很多，我这里（IOS）测试使用两个，安卓使用后者。并且官方有[源码](#)。



- 此处没有将配网信息保存，产品中要保存，下次就不用再配了，如果配错了，可以加按键操作清楚配置信息，官方已经把这个功能做好了，自己怼怼。
- 源码地址: <https://github.com/xiaolongba/wireless-tech>