物联网 WiFi 设备 SoftAP 方式配网

修订记录

修订日期	修订版本	修改描述	作者
20191209	V1.0	创建	spikelin
20191231	V1.1	优化小程序 UDP 配网流程	spikelin

目录

SoftAP 配网简介及交互流程

WiFi“配网”指的是，由外部向 WiFi 设备提供 SSID 和密码（PSW），让 WiFi 设备可以连接指定的热点或路由器并加入后者所建立的 WiFi 网络。对于具备丰富人机界面包括屏幕/键盘的设备比如电脑或者手机，可以直接输入 SSID/PSW 来进行连接，而对于不具备丰富人机交互界面的物联网 WiFi 设备比如智能灯、扫地机器人等，则可以借助手机等智能设备，以某种配网方式将 SSID/PSW 告诉该设备。

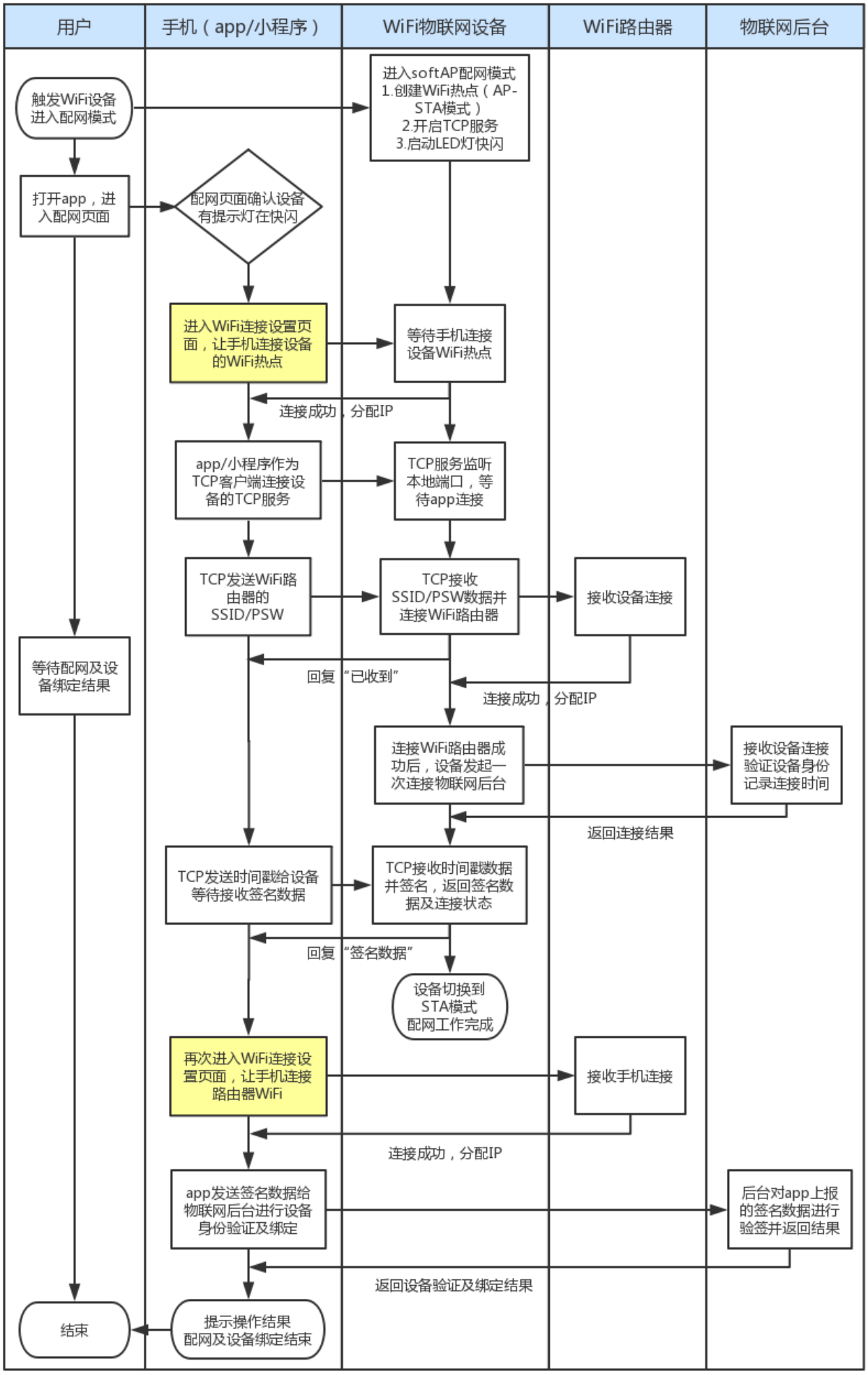
配网有多种方式，包括 WPS、smartconfig、softAP 等等，其中 WPS 存在安全性问题，而 smartconfig 虽然比较便捷，但是一般都是各个厂商采用私有协议，兼容性和互操作性较差，而 softAP 方式配网的优点是适配性兼容性都比较好，对手机或者设备只需要标准的 WiFi 连接和 TCP/UDP 操作。缺点是手机端需要做两次 WiFi 连接设置的切换，步骤稍微复杂。

SoftAP 方式配网的基本原理是让设备通过 softAP 方式创建一个 WiFi 热点，手机连接到该设备热点，再通过数据通道比如 TCP/UDP 通讯将目标 WiFi 路由器的 SSID/PSW 告诉设备，设备获取到之后就可以连接目标 WiFi 路由器从而连接互联网。同时，为了对设备进行绑定，手机 app 可以利用该 TCP/UDP 数据通道让设备提供签名数据并将该数据发送至物联网后台进行验证和绑定。

下页是一个 softAP 方式配网及设备绑定的示例流程图：

注意：下图是基于 WiFi 设备支持 AP-STA 模式的流程。在 WiFi 网络中，AP(Access Point)是指可以创建网络并提供接入能力的设备，比如一个 WiFi 热点。STA(station)是指可接入到 AP 的节点，比如一个手机。而 AP-STA 模式是指设备可以同时作为 AP 供其他 STA 接入，同时又能作为 STA 去连接其他 AP 的模式。利用该模式，设备在获取手机发送过来的 WiFi 路由器 SSID/PSW 并去连接 WiFi 路由器和互联网的过程中，可以保持与手机的数据连接，并能将连接路由器和物联网后台的结果及时告诉手机。大部分 WiFi 设备都支持这种模式，对于不支持该模式的设备，则在接收到手机发送过来的 SSID/PSW 信息之后，要先关闭本身创建的 WiFi 热点（即与手机断开了连接通道）后才能去连接 WiFi 路由器。

softAP方式配网及设备绑定（模组支持AP-STA共存模式）



ESP8266 腾讯云定制模组配合腾讯连连小程序 softAP 配网协议示例

1. ESP8266 模组接到 PC 端 USB 串口，使用串口工具进入 AT 指令交互
2. 输入 AT 指令，使 WiFi 模组进入 softAP 配网模式（第一个参数是创建的 WiFi 热点的 SSID，第二个参数是密码）

   AT+TCSAP="ESP8266-SoftAP","12345678"

   看到模组说明有指示灯在快闪，则说明进入配网模式成功

3. 手机连接该 WiFi 热点"ESP8266-SoftAP"，获取到 IP

4. 小程序作为 UDP 客户端连接 WiFi 模组上面的 UDP 服务（IP 为网关地址，在 ESP8266 上是 192.168.4.1，端口为 8266）

5. 小程序给模组 UDP 服务发送目标 WiFi 路由器的 SSID/PSW，JSON 格式为：

   {"cmdType":1,"ssid":"Home-WiFi","password":"abcd1234"}

   发送完了之后等待模组 UDP 服务回复确认消息：

   {"cmdType":2,"deviceReply":"dataRecived"}

6. 如果 2 秒之内没有收到模组回复，则重复步骤 5，UDP 客户端重复发送目标 WiFi 路由器的 SSID/PSW。

   如果重复发送 5 次都没有收到回复，则认为配网失败，模组有异常，需要对模组进行重启复位。

7. 如果步骤 5 收到模组回复了，则说明模组已经收到 WiFi 路由器的 SSID/PSW，正在进行连接。这个时候小程序需要等待 3 秒钟，然后发送时间戳信息给模组，JSON 格式为：

   {"cmdType":0,"timestamp":1234567890}

   发送完了之后等待模组 UDP 服务回复签名消息：

{"cmdType":2,"productId":"0C88P7AQQ9","deviceName":"wifi_kit","connId":"eaAZy","signature":"e595bf7703cc383d76567f8cc13591012c495c50","timestamp":1234567890,"wifiState":"connected","mqttState":"connected"}

8. 如果 2 秒之内没有收到模组回复，则重复步骤 7，UDP 客户端重复发送时间戳信息。

   如果重复发送 5 次都没有收到签名回复，则认为配网失败。

9. 在以上 5-8 步骤中，如果小程序收到模组 UDP 服务发送过来的错误日志，且 deviceReply 字段的值为"Current_Error"，则表示当前配网绑定过程中出错，需要退出配网操作。如果 deviceReply 字段是"Previous_Error"，则为上一次配网的出错日志，只需要上报，不影响当此操作。

错误日志 JSON 格式例子：

{"cmdType":2,"deviceReply":"Current_Error","log":"ESP WIFI connect error! (10, 2)"}

10. 如果步骤 7 收到模组回复的签名串，并且 wifiState 和 mqttState 字段的值都是"connected"，则表示模组已经通过 WiFi 连接到云端后台，配网成功。手机可以连接到目标 WiFi 路由器，与云端后台通讯。

UDP 通讯数据交互注意事项

UDP 相比 TCP 是不可靠的通讯，存在丢包的可能，特别在比较嘈杂的无线 WiFi 环境中，丢包率会比较大。为了保证小程序和设备之间的数据交互是可靠的，需要在应用层设计一些应答以及超时重发的机制。其中小程序端如上面步骤所述，采用了超时等待并重复查询的方式，设备端则结合了状态保存以支持重复回复，以及多次发包的方式来提高可靠性。

模组发送给 app/小程序的配网错误日志格式说明:

当模组在配网过程中出现异常或错误，会通过 TCP/UDP 数据通道向 app/小程序发送错误日志，以便于分析和统计相关配网错误。日志采用 JSON 格式，有三个字段，其中 cmdType 字段固定为 2，deviceReply 字段表示配网失败的错误日志类型，上一次失败的日志用"Previous_Error"表示，当前失败的日志用"Current_Error"表示，log 字段为具体的错误信息，具体内容的字符串后面会跟一个括号，括号里面前面的数值是跟前面字符串对应的 error code，具体可以见下面表格，后面会跟一个 sub error code，与具体的 API 调用或异常相关。
如下面的例子：

```
{
        "cmdType":      2,
        "deviceReply":  "Previous_Error",
        "log":  "WIFI boarding stop! (3, 102)"
}

{
        "cmdType":      2,
        "deviceReply":  "Current_Error",
        "log":  "MQTT connect error! (1, -30592)"
}
```

模组配网及设备绑定错误日志 error code

错误提示文本	错误码
"Everything OK!"	SUCCESS_TYPE = 0
"MQTT connect error!"	ERR_MQTT_CONNECT = 1
"APP command error!"	ERR_APP_CMD = 2
"WIFI boarding stop!"	ERR_BD_STOP = 3
"RTOS task error!"	ERR_OS_TASK = 4
"RTOS queue error!"	ERR_OS_QUEUE = 5
"WIFI STA init error!"	ERR_WIFI_STA_INIT = 6
"WIFI AP init error!"	ERR_WIFI_AP_INIT = 7
"WIFI start error!"	ERR_WIFI_START = 8
"WIFI config error!"	ERR_WIFI_CONFIG = 9
"WIFI connect error!"	ERR_WIFI_CONNECT = 10
"WIFI disconnect error!"	ERR_WIFI_DISCONNECT = 11
"WIFI AP STA error!"	ERR_WIFI_AP_STA = 12
"ESP smartconfig start error!"	ERR_SC_START = 13
"ESP smartconfig data error!"	ERR_SC_DATA = 14
"SRV socket open error!"	ERR_SOCKET_OPEN = 15
"SRV socket bind error!"	ERR_SOCKET_BIND = 16
"SRV socket listen error!"	ERR_SOCKET_LISTEN = 17
"SRV socket fcntl error!"	ERR_SOCKET_FCNTL = 18
"SRV socket accept error!"	ERR_SOCKET_ACCEPT = 19
"SRV socket recv error!"	ERR_SOCKET_RECV = 20
"SRV socket select error!"	ERR_SOCKET_SELECT = 21
"SRV socket send error!"	ERR_SOCKET_SEND = 22