涂鸦 MCU 对接指导手册

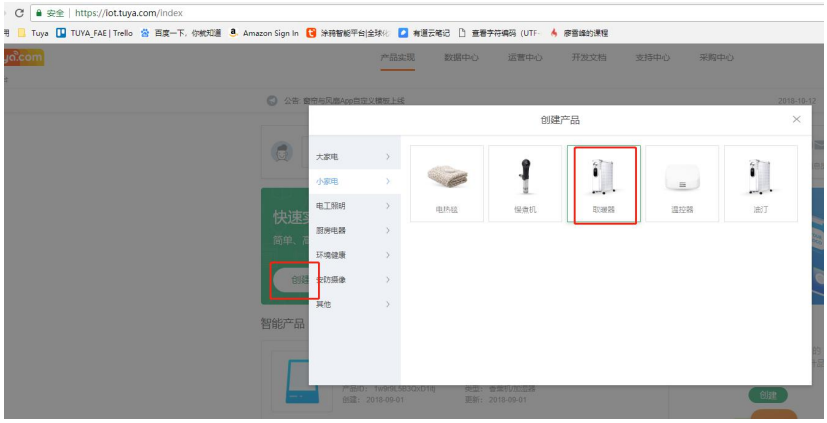
目录

一、平台创建产品，下载 MCU 开发包

这里我们以取暖器为例，以个人开发者角度，简要介绍产品创建

1) .登录 https://iot.tuya.com，创建自己的开发者账号

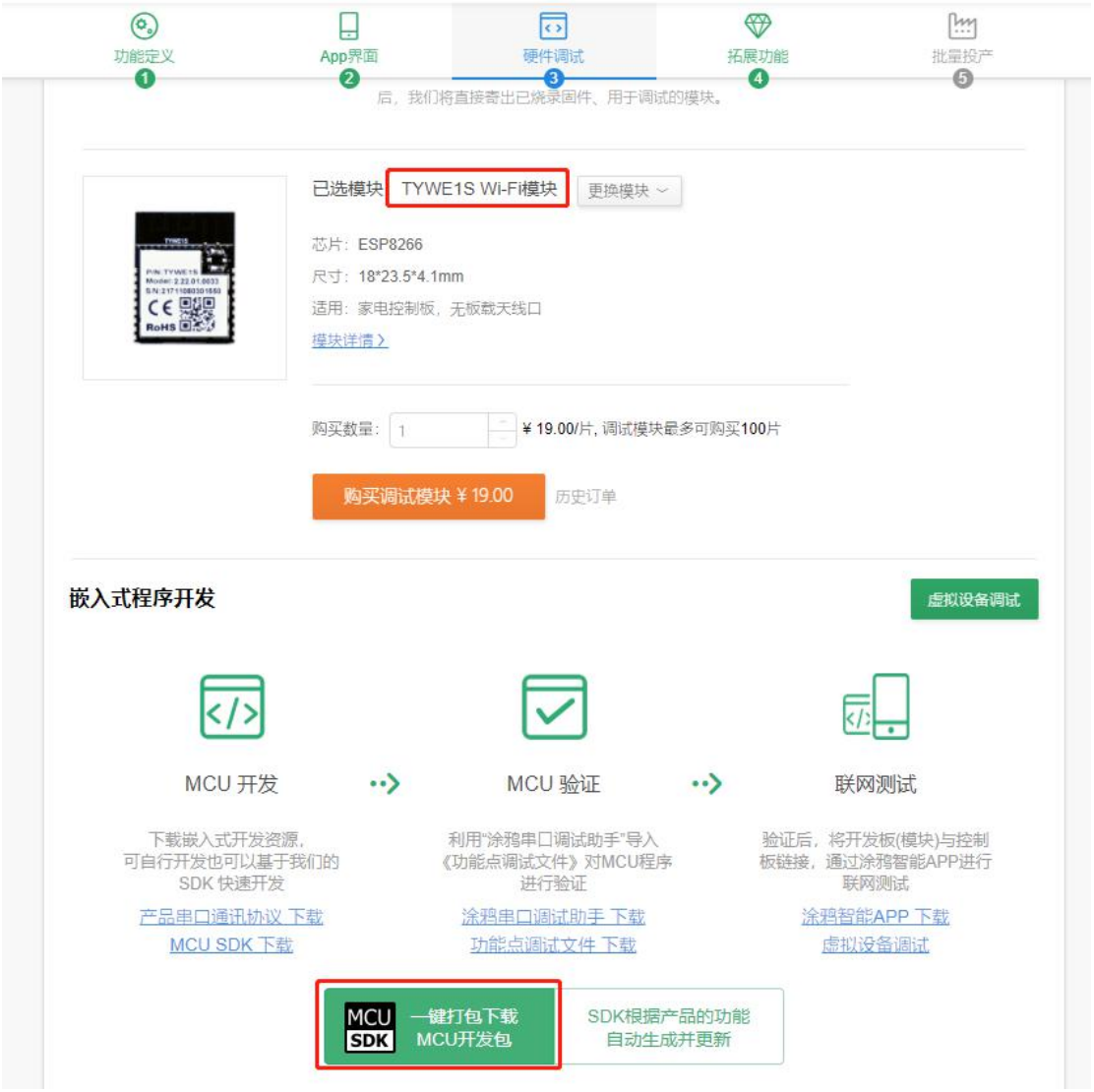
2) .点击创建产品，根据实际产品选择产品品类，这里我们选择取暖器。



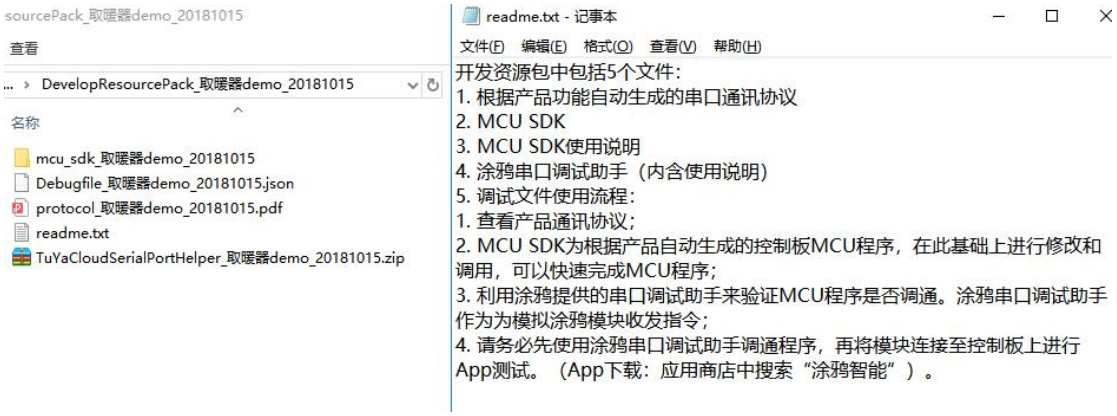
3) .根据产品实际需求，选择 DP 功能点。若还有自定义功能，可在添加完毕后根据实际功能增加自定义功能点。自定义产品功能说明：https://docs.tuya.com/cn/product/function.html

4) .选择一个喜欢的 APP 面板，可扫码体验虚拟面板（企业账号会有更多面板选择，账号升级可联系商务同学）

5) .一键下载 MCU 开发包资料。



6) .开发资源包，包含以下资料



二、协议解析

协议主要分为两部分：基础协议、功能协议。基础协议和产品无关，是模组共有协议，包括模组初始化指令及部分扩展功能指令。功能协议是平台根据每个产品 DP 点定义不同，自动生成的功能点数据收发指令。

2.1 协议框架

MCU 和 WiFi 模组通过串口交互，通用固件：波特率 9600/数据位 8 位/无奇偶校验/停止位 1 位/无数据流控。数据帧格式如下：

字段	长度（byte）	说明
帧头	2	固定为 0x55aa
版本	1	升级扩展用
命令字	1	具体帧类型
数据长度	2	大端
数据	xxxx	
校验和	1	从帧头开始按字节求和得出的结果对 256 求余

命令字索引表如下：

命令字	说明
0x00	心跳检测
0x01	查询产品信息
0x02	查询设定模块工作模式
0x03	报告设备联网状态
0x04	重置 WiFi-切换配网模式
0x05	重置 WiFi-选择配网模式
0x06	DP 命令下发
0x07	DP 状态上报
0x08	状态查询
0x0a	OTA 升级启动（可选）
0x0b	OTA 升级包传输（可选）
0x1c	获取本地时间（可选）
0x0e	WiFi 功能性测试（产测指令）

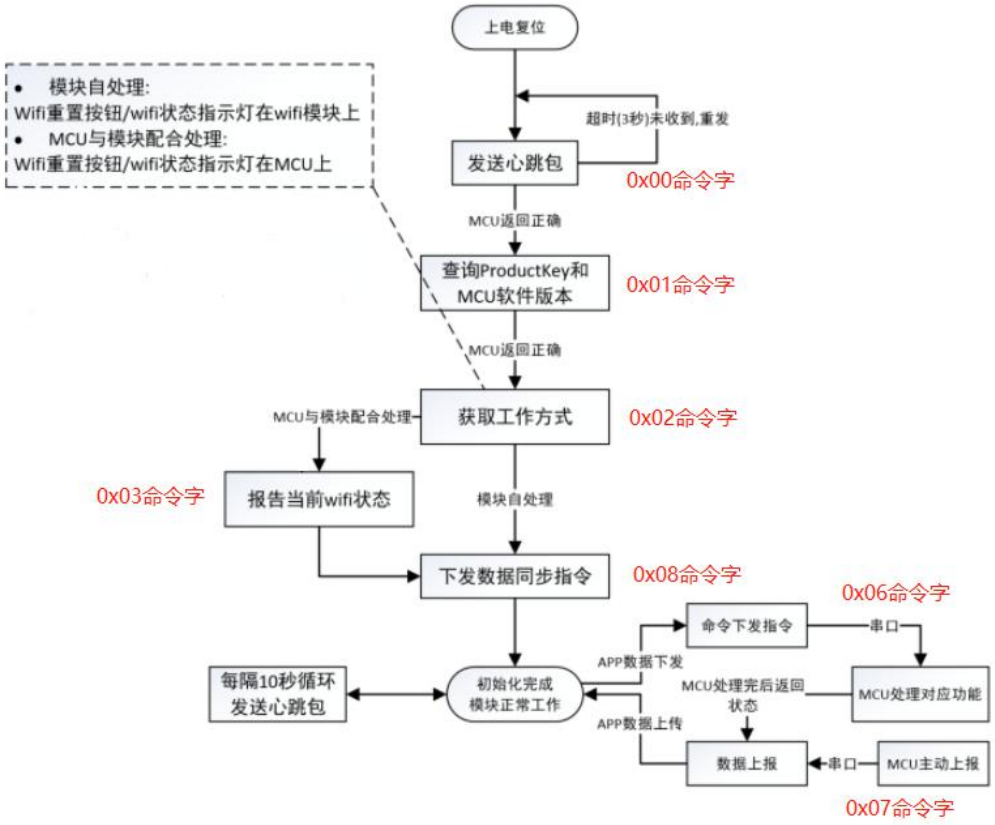
2.1.1 基础协议

基础协议每个产品都一样，是模组工作所必须的协议。主要包含:心跳检测、产品信息查询、模组工作方式、WiFi 工作状态等功能。

其中，命令字 0x00-0x08 为模组基本指令。命令字 0x0a-0x0e 为模组基础功能，包括 MCU OTA、获取本地时间、产测功能等。

要使模组正常工作，我们要实现的协议主要分为两个部分：模组初始化、配网。

模组初始化协议主要涉及命令字，主要流程如下：



模组上电会不断发送心跳包，等到 MCU 端回复心跳，会启动如上的初始化流程。

1).心跳检测。心跳包回复要**注意**，MCU 上电第一次回复数据为 0x00，之后的数据回复为 0x01。这样第一次回复 0x00，模组会自动进行初始化数据同步。之后的心跳包用来判定设备是否离线和断网自动重连。

		帧头 版本	命令字	数据长度	数据	校验和
心跳检测	模块发送	0x55aa 0x00	0x00	0x0000		0xff
	MCU上报	0x55aa 0x03	0x00	0x0001	**0x00(第1次) 0x01(其它)**	校验和

例：

模组发送：55 aa 00 00 00 00 ff

MCU 回复：55 aa 03 00 00 01 00 03 （第一次）/55 aa 03 00 00 01 01 04（其他）

2).查询产品信息。收到心跳包回复后，模组会发送查询产品指令信息。MCU 上报 PID、版本、模式等信息。这里要注意{}:""这些字符也是要写的，具体格式参考后图。

查询产品信息	模块发送	0x55aa 0x00	0x01	0x0000		0x00
	MCU上报	0x55aa 0x03	0x01	0x0015 (0x0010- 0x0018)	模式： 0：默认配网 1：低功耗 2：特殊配网 格式：{"p":"svizlf0dzs4rz85c","v":"1.0.0","m":0}	校验和

例：

模组发送：55 aa 00 01 00 00 00

MCU 回复：下图以 PID:RN2FVAgXG6WfAktU 为例，用户根据自己产品 PID 转换成 ASCII 码替换到相应位置即可。

例：{"p":"RN2FVAgXG6WfAktU","v":"1.0.0","m":0}

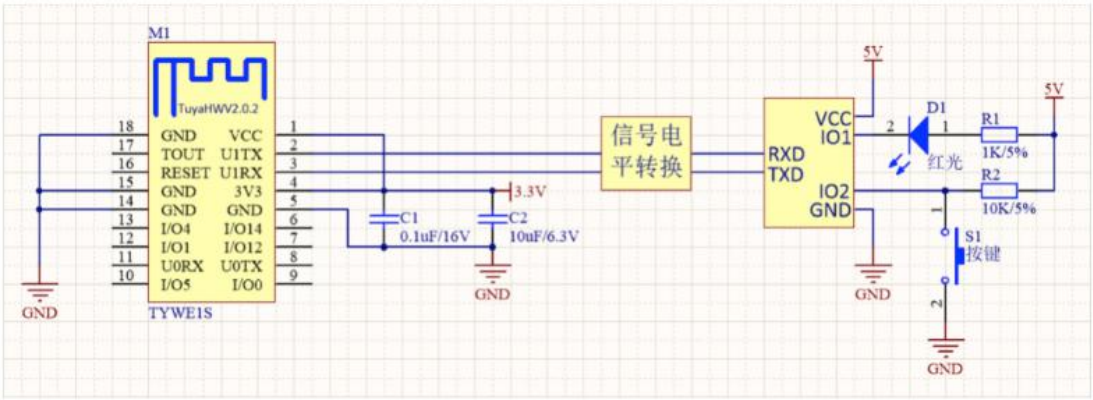
p 表示产品 ID 为 RN2FVAgXG5WfAktU， v 表示 mcu 版本为 1.0.0， m 表示配网方式为 0（0：默认配网 1：低功耗 2：特殊配网）

55	aa	03	01	00	2a	7b	22	70	22	3a	22	52	4e	32	46
帧头						{	"	P	"	:	"	R	N	2	F
56	41	67	58	47	36	57	66	41	6b	74	55	22	2c	22	76
V	A	g	X	G	6	W	f	A	k	t	U	"	,	"	v
22	3a	22	31	2e	30	2e	30	22	2c	22	6d	22	3a	30	7d
"	:	"	1	.	0	.	0	"	,	"	m	"	:	0	}
0c															
校验位															

3).设定模组工作方式。收到产品信息后，模组会发送查询 MCU 设定模组工作方式 0x02 命令字。

模块工作模式主要针对如何指示 WIFI 的工作状态以及如何重置 WIFI，主要分两种情况：

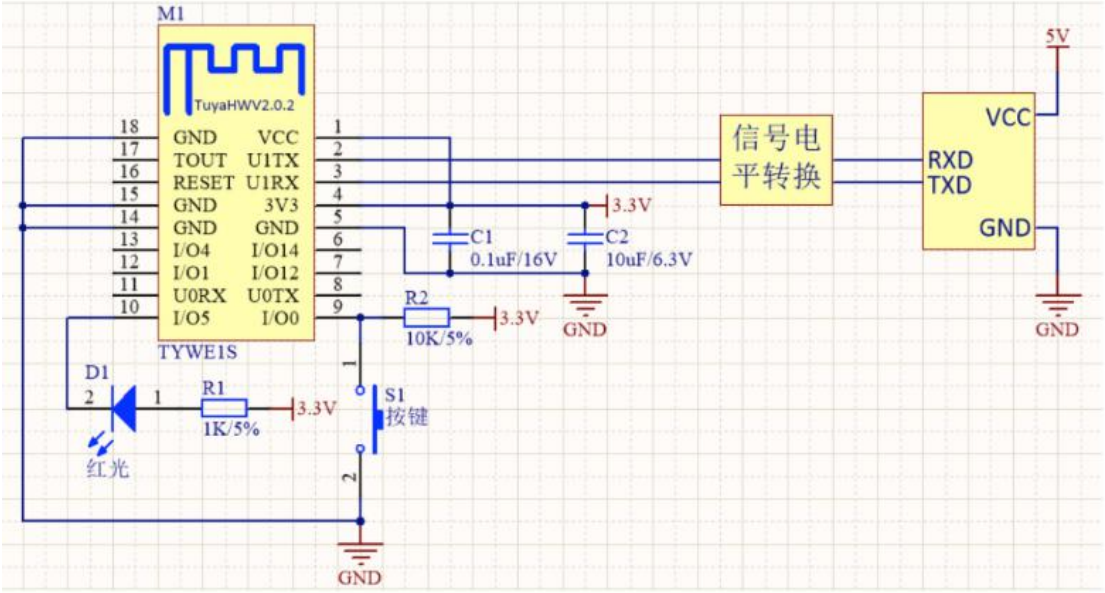
a. MCU 与模块配合处理模式。即模块通过串口通知 MCU WIFI 当前的工作状态，由 MCU 控制配网指示灯显示。



MCU与模块配合处理模式

b. 模块自处理模式。WIFI 模块的配网状态通过 WIFI 的 GPIO 引脚驱动 LED 状态显示；WIFI 重置通过检测 GPIO 输入需求处理。

模块自处理 WIFI 重置方法为：WIFI 检测 GPIO 入口低电平持续 5s 以上触发 WIFI 重置。指示灯与按钮所使用的 GPIO 管脚由以下命令配置。



模块自处理模式

MCU 端若选择 MCU 与模组配合处理模式的上报数据为 0，若选择模块自处理模式上报数据为接指示灯 IO 口和按键 IO 口。如果产品采用模块自处理方式，则以下 4-6 协议无须关心。

查询 MCU 设定模块工作方式	模块发送	0x55aa 0x00	0x02	0x0000		0x01
	MCU上报(MCU与模块配合处理)	0x55aa 0x03	0x02	0x0000		校验和
	MCU上报(模块自处理)	0x55aa 0x03	0x02	0x0002	首字节为WiFi状态指示GPIO序号；次字节为WiFI重置键GPIO序号	校验和

例：

模组发送：55 aa 00 02 00 00 01

MCU 回复：

55 aa 03 02 00 00 04（MCU 与模组配合处理模式）/

55 aa 03 02 00 02 05 00 0b（模组自处理模式，指示灯接 IO5，按键接 IO0）

4).报告 wifi 状态（模块自处理方式无须关心）。模组主动发送，当模块检测到 MCU 重启或模块的 WIFI 状态发生变化，会主动发送状态到 MCU。MCU 可以根据 03 命令字中的 wifi 状态控制指示灯的闪烁。03 版本协议中，共有 6 中状态：

设备联网状态	描述	状态值	LED 显示
状态 1	smart 配置状态	0x00	快闪 250ms

状态 2	AP 配置状态	0x01	慢闪 1500ms
状态 3	WIFI 已配置但未连上路由器	0x02	熄灭
状态 4	WIFI 已配置且连上路由器	0x03	常亮
状态 5	已连上路由器且连接到云端	0x04	常亮
状态 6	WIFI 设备处于低功耗模式	0x05	熄灭

报告 WiFi 工作状态	模块发送	0x55aa 0x00	0x03	0x0001	指示WiFI状态： 0x00：Smartconfig 配网模式（灯快闪）； 0x01：AP配网模式（灯慢闪）； 0x02：WiFi配置成功但未连上路由（灯熄灭）； 0x04：已连上路由器且连接到云端（灯长亮）；	校验和
	MCU上报	0x55aa 0x03	0x03	0x0000		校验和

例：

模组发送：55 aa 00 03 00 01 01 校验和（01:AP 配置模式）

MCU 回复：55 aa 03 03 00 00 05

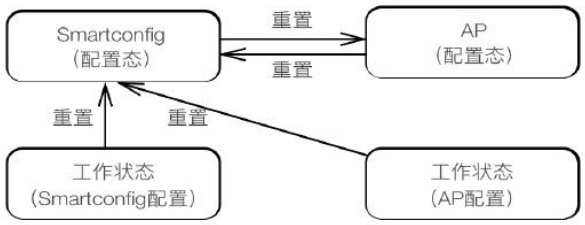
5) **.重置 wifi**（模块自处理方式无须关心）。配网指令，可以通过重置 WiFi 使设备处于待配网状态。配网模式分两种：

a.smart 模式（快闪模式），配网方式简单快捷

b.AP 模式（慢闪模式），配网稳定可靠

建议两种模式都要做，触发机制可以自定义，以指示灯快闪、慢闪给用户做区分。

0x04 命令字，MCU 每发送一次，模组切换一次配网模式。第一次默认 smart 模式，之后在 smart 与 AP 之间来回切换。



重置 WiFi	MCU发送	0x55aa 0x03	0x04	0x0000		校验和
	模块上报	0x55aa 0x00	0x04	0x0000		0x03

例：

MCU 发送：55 aa 03 04 00 00 06

模组回复：55 aa 00 04 00 00 03

6).重置 wifi-选择模式（模块自处理方式无须关心）。重置 wifi 并选择配网模式，根据 MCU 发送的参数不同，可指定进入 smart 或 AP 模式。此指令与 0x04 命令字作用相同，均可配网，但可指定进入某种配网模式

重置 WiFi 选择模式（MCU上报二选一）	MCU上报(Smartconfig模式)	0x55aa 0x03	0x05	0x0001	0x00	校验和
	MCU上报(AP模式)	0x55aa 0x03	0x05	0x0001	0x01	校验和
	模块发送	0x55aa 0x00	0x05	0x0000		0x04

例：

MCU 发送：55 aa 03 05 00 01 00 08 （smart 模式）/55 aa 03 05 00 01 01 09 (AP 模式)

模组回复：55 aa 00 05 00 00 04

7).MCU 工作状态产查询。08 命令字，用于模块查询 MCU 所有 datapoint 状态，作为 APP 显示初值。MCU 收到后，分条上报所有 DP 点数据。“状态查询”发送情况主要有两种：

a. 模块首次上电，通过心跳与 MCU 建立连接后，查询发送；

b. 模块检测到 MCU 重启或发生了离线再上线的过程，查询发送。

查询 mcu 工作状态	模块发送	0x55aa 0x00	0x08	0x0000		校验和
	MCU上报	0x55aa 0x03	0x07	N	上报所有DP点数据，作为显示初值	校验和

例：

模组发送：55 aa 00 08 00 00 07

MCU 回复：

55 aa 03 07 N **** 校验和（DP1）

55 aa 03 07 N **** 校验和（DP2）

.....（DPN）

8).产测指令。主要用于产品量产时的 WiFi 模组射频性能测试。产测指令，建议等待上电初始化完成，5S 以后调用。模组收产测指令到后，会自动扫描名称为 tuya_mdev_test WiFi 信号，返回扫描结果和信号强度百分比（0-100 步进 20）。

WiFi功能 产测（注：扫描 tuya_mde v_test 的 指定SSID）	MCU上报	0x55aa 0x03	0x0e	0x0000		校验和
	模块发送	0x55aa 0x00	0x0e	0x0002	数据长度为2字节： Data[0]:0x00失败，0x01成功；当Data[0]为0x01，即成功时， Data[1]表示信号强度（0-100，0信号最差，100信号最强）当Data[0]为0x00，即失败时， Data[1]为0x00表示未扫描到指定的ssid， Data[1]为0x01 表示模块未烧录授权key	校验和

例：

MCU 发送：55 aa 03 0e 00 00 校验和

模组回复：55 aa 00 0e 00 02 01 28 38 （产测成功，信号强度 40）

2.1.2 功能协议

DP 点数据下发和上报指令。模组下发数据命令字 0x06，MCU 上报数据命令字 0x07。MCU 在收到数据下发指令后，根据收到的功能指令数据不同，进行相应的逻辑控制。并在 DP 状态改变时及时上报，更改 APP 显示状态。DP 点数据上报，MCU 遵循一个宗旨：状态改变就上报。（若同一 DP 点数据与上次上报相同，模组端会进行数据过滤）

例:

ID	功能名称		帧头版本	命令字	数据长度	dpID	数据类型	功能长度	功能指令	校验
1	开关	模块发送	0x55aa 0x00	0x06	0x00 0x05	0x01	0x01	0x00 0x01	off:0x00 on:0x01	校验和
		MCU上报	0x55aa 0x03	0x07	0x00 0x05	0x01	0x01	0x00 0x01		校验和
2	目标温度	模块发送	0x55aa 0x00	0x06	0x00 0x08	0x02	0x02	0x00 0x04	0x1e-0x50	校验和
		MCU上报	0x55aa 0x03	0x07	0x00 0x08	0x02	0x02	0x00 0x04		校验和
11	倒计时剩余时间	模块发送	0x55aa 0x00	0x06	0x00 0x05	0x0b	0x04	0x00 0x01	1hour:0x00 2hour:0x01 3hour:0x02	校验和
		MCU上报	0x55aa 0x03	0x07	0x00 0x05	0x0b	0x04	0x00 0x01		校验和
13	故障告警	MCU上报	0x55aa 0x03	0x07	0x00 0x06	0x0d	0x05	0x00 0x02	bit0:1 bit1:2 bit2:4 bit3:8 bit4:16 bit5:32 bit6:64 bit7:128 bit8:512	校验和
17	周程序	模块发送	0x55aa 0x00	0x06	N	0x11	0x00	N	0x00-0xff	校验和
		MCU上报	0x55aa 0x03	0x07	N	0x11	0x00	N		校验和
102	数据串	模块发送	0x55aa 0x00	0x06	N	0x66	0x03	N	0x00-0xff	校验和
		MCU上报	0x55aa 0x03	0x07	N	0x66	0x03	N		校验和

注意：

a. value 型数据，有 4 个字节，使用不满 4 字节的，前边补 0 即可。

例：上图目标温度数据

MCU 发送：55 aa 03 07 00 08 02 02 00 04 00 00 00 1e 校验和（目标温度 30 度）

b. 故障型数据，支持多故障同时上报。每一个 bit 位可代表一个警告，置 1 表示故障发生，置 0 表示无故障。

例:上图故障警告

MCU 发送：55 aa 03 07 00 06 0d 05 00 02 00 09 校验和（故障 bit0 和故障 bit3 发生故障）

c. 字符型数据，字符串的含义与显示需与面板配套，自定义的可与面板沟通。

d. Raw 类型数据，透传型数据，通常用于比较复杂的功能实现。不建议用户自行使用。

三、移植 SDK 实战

3.1 移植须知

mcu_sdk 包是根据涂鸦开发平台上定义的产品功能，自动生成的 MCU 代码。通讯及协议解析架构已写好，可直接添加到原有 MCU 工程中，快速完成 MCU 程序开发。

SDK 包对 MCU 硬件资源需求：Flash 4Kbyte、RAM 与 DP 点数据长度有关几十字节左右（OTA 功能需大于 260byte）。函数嵌套级数 9 级。若资源不足的用户，可自行对接协议，SDK 包中的函数依然可以作为参考。

执行文件	头文件	说明
mcu_api.c	mcu_api.h	内含 WiFi 相关函数，客户可按需调用
protocol.c	protocol.h	协议文件，用户根据项目需求，需修改这个 2 个文件，内含数据处理函数
system.c	system.h	串口通讯协议的具体实现,可以不做了解
	wifi.h	WiFi 相关宏定义

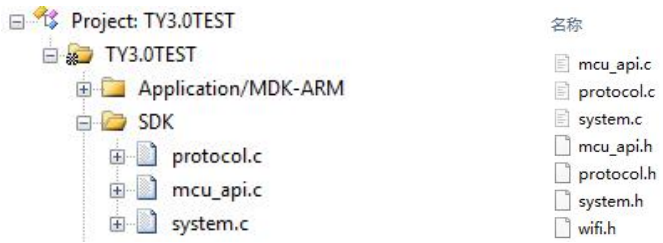
3.2 涂鸦 MCU SDK 对接流程六步走

步骤 1： 编写 MCU 基础程序，移植 SDK 文件
步骤 2： 确认 protocol.h 宏定义
步骤 3： 移植 protocol.c 文件及函数调用
步骤 4： DP 上报下发函数完善调用
步骤 5： 配网功能及闪灯函数完善
步骤 6： 产测功能完善

3.3 移植步骤详解

步骤 1.编写 MCU 基础程序，移植 SDK 文件

首先在原项目工程中，新建一个文件夹，加入 mcu_sdk 文件夹中的.c.h 文件，添加相应头文件引用路径。完成 MCU 相关外设初始化，包括：串口、外部中断（按键）、定时器（指示灯闪烁）等



步骤 2. 确认 protocol.h 宏定义

1.确认产品信息

PRODUCT_KEY 为产品 PID 宏定义,请确保代码与网站所列两者一致，该 PID 为每个产品的唯一标识。如果不是重新下载最新 SDK 开发包。

MCU_VER 为软件版本，默认 1.0.0 。若 MCU 需要 OTA 功能，OTA 升级后需更新版本号。

CONFIG_MODE 为配网方式，通常选择“默认配网”方式。

protocol.h

```
**
**-----------------------------------------------------------------------------
*******************************************************************************/
#ifndef __PROTOCOL_H_
#define __PROTOCOL_H_



/*******************************************************************************
                            用户相关信息配置
*******************************************************************************/
/*******************************************************************************
                            1:修改产品信息
*******************************************************************************/
#define PRODUCT_KEY "svizlf0dzs4rz85c"    //开发平台创建产品后生成的16位字符产品唯一标识

#define MCU_VER "1.0.0"                                 //用户的软件版本,用于MCU固件升级,MCU升级版本需修改

//配网方式选择,默认为CONFIG_MODE_DEFAULT,只能三选一
#define CONFIG_MODE     CONFIG_MODE_DEFAULT             //默认配网方式
//#define CONFIG_MODE     CONFIG_MODE_LOWPOWER              //低功耗配网方式
//#define CONFIG_MODE     CONFIG_MODE_SPECIAL               //特殊配网方式

/*******************************************************************************
```

2.确认 MCU 是否需要支固件升级

如需要支持 MCU 固件 OTA 升级,请开启该宏（默认关闭）。

```
/*******************************************************************************
                        2:MCU是否需要支固件升级
如需要支持MCU固件升级,请开启该宏
MCU可调用mcu_api.c文件内的mcu_firm_update_query()函数获取当前MCU固件更新情况
                        ********WARNING!!!**********
当前接收缓冲区为关闭固件更新功能的大小,固件升级包为256字节
如需要开启该功能,串口接收缓冲区会变大
*******************************************************************************/
//#define          SUPPORT_MCU_FIRM_UPDATE                  //开启MCU固件升级功能(默认关闭)
```

3.定义收发缓存

修改缓冲区大小，根据 DP 点定义，串口接收和发送缓存大小要大于数据最长的 DP 数据长度。默认大小 24。需要做 MCU OTA 升级的用户缓存大小建议大于 260 字节。接收队列大小，若 RAM 紧张可以适当缩小。

```
                    3:定义收发缓存:
                如当前使用MCU的RAM不够,可修改为24
*****************************************************************************/
#ifndef SUPPORT_MCU_FIRM_UPDATE
#define WIFI_UART_QUEUE_LMT          16          //数据接收队列大小,如MCU的RAM不够,可缩小
#define WIFI_UART_RECV_BUF_LMT       128          //根据用户DP数据大小量定,必须大于24
#else
#define WIFI_UART_QUEUE_LMT          128         //数据接收队列大小,如MCU的RAM不够,可缩小
#define WIFI_UART_RECV_BUF_LMT       300         //固件升级缓冲区,需大缓存,必须大于260
#endif

#define WIFIR_UART_SEND_BUF_LMT      128          //根据用户DP数据大小量定,必须大于24
```

4.定义模块工作方式（必要）

1).如果配网触发及指示由 MCU 控制（按键和 LED 接在 MCU 端），选择“模块和 MCU 配合处理”工作模式（常用），请保持 define 被注释状态

```
/*****************************************************************************
                    4:定义模块工作方式
模块自处理:
        wifi指示灯和wifi复位按钮接在wifi模块上(开启WIFI_CONTROL_SELF_MODE宏)
        并正确定义WF_STATE_KEY和WF_RESET_KEY
MCU自处理:
        wifi指示灯和wifi复位按钮接在MCU上(关闭WIFI_CONTROL_SELF_MODE宏)
        MCU在需要处理复位wifi的地方调用mcu_api.c文件内的mcu_reset_wifi()函数,并可调用mcu_get_reset_wifi_flag()函数返回复位wifi结果
        或调用设置wifi模式mcu_api.c文件内的mcu_set_wifi_mode(WIFI_CONFIG_E mode)函数,并可调用mcu_get_wifi_work_state()函数返回设置wifi结果
*****************************************************************************/
//#define         WIFI_CONTROL_SELF_MODE                  //wifi自处理按键及LED指示灯;如为MCU外界按键/LED指示灯请关闭该宏
```

2).如果 wifi 指示灯和按键是接在 wifi 模块上的（模块自处理工作模式），那么请开启

#ifdef WIFI_CONTROL_SELF_MODE

然后根据实际的硬件连接，将指示灯和按键所连接的 GPIO 脚位填入下面两行

```
#define         WIFI_CONTROL_SELF_MODE                  //wifi自处理按键及LED指示灯;如为MCU外界按键/LED指示灯请关闭该宏
#ifdef          WIFI_CONTROL_SELF_MODE                  //模块自处理
  #define     WF_STATE_KEY            14              //wifi模块状态指示按键, 请根据实际GPIO管脚设置
  #define     WF_RESERT_KEY           0               //wifi模块重置按键, 请根据实际GPIO管脚设置
#endif
```

5.确认 MCU 是否需要支持校时功能

如需要支持校时功能,请开启该宏

```
/*****************************************************************************
                    5:MCU是否需要支持校时功能
如需要请开启该宏,并在Protocol.c文件内mcu_write_rtctime实现代码
mcu_write_rtctime内部有#err提示,完成函数后请删除该#err
mcu在wifi模块正确联网后可调用mcu_get_system_time()函数发起校时功能
*****************************************************************************/
//#define          SUPPORT_MCU_RTC_CHECK                  //开启校时功能
```

并在 Protocol.c 文件内 mcu_write_rtctime 实现代码， mcu 在 wifi 模块正确联网后可调用 mcu_get_system_time()函数发起校时功能

6.确认是否开启 WiFi 产测功能（开启）

为保证最终量产效率及品质，建议开启该宏。具体产测功能实现，见文档产测部分。

```
                    6:MCU是否需要支持wifi功能测试
如需要请开启该宏,并且mcu在需要wifi功能测试处调用mcu_api.c文件内mcu_start_wifitest
并在protocol.c文件wifi_test_result函数内查看测试结果,
wifi_test_result内部有#err提示,完成函数后请删除该#err
*****************************************************************************/
#define          WIFI_TEST_ENABLE                       //开启WIFI产测功能
```

步骤 3. 移植 protocol. c 文件及函数调用

1.在需要使用到 wifi 相关文件的文件中 include "wifi.h"文件。如 main.c

2.在 MCU 外设初始化后调用 mcu_api.c 文件中的 wifi_protocol_init()函数

3.将 MCU 串口单字节发送函数填入 protocol.c 文件中 uart_transmit_output 函数内,并删除 #error。例如：

```c
/*******************************************************************************
函数名称 : uart_transmit_output
功能描述 : 发数据处理
输入参数 : value:串口收到字节数据
返回参数 : 无
使用说明 : 请将MCU串口发送函数填入该函数内,并将接收到的数据作为参数传入串口发送函数
*******************************************************************************/
void uart_transmit_output(unsigned char value)
{
//  #error "请将MCU串口发送函数填入该函数,并删除该行"
  UART3_SendByte(value);
/*
  //示例:
  extern void Uart_PutChar(unsigned char value);
  Uart_PutChar(value);                                    //串口发送函数
*/
}
```

4. 在串口接收中断服务函数里面调用 mcu_api.c 文件内的 uart_receive_input 函数,并将接收到的字符作为参数传入。例如：

```c
void USART3_IRQHandler(void)
{
  /* USER CODE BEGIN USART3_IRQn 0 */
  unsigned char Res=0;

    if((USART3->SR&UART_FLAG_RXNE) != 0)
    {
      Res=USART3->DR;
      uart_receive_input(Res);
    }

}
```

5.单片机进入 while 循环后调用 mcu_api.c 文件内的 wifi_uart_service()函数

main.c 中示例代码结构如下：

```
include "wifi.h"
...
void main(void)
{
    wifi_protocol_init();
    ...
    while(1)
    {
        wifi_uart_service();
        ...
    }
}
```

注意：

MCU 必须在 while 中直接调用 mcu_api.c 内的 wifi_uart_service()函数程序正常初始化完成后,建议不进行关串口中断,如必须关中断,关中断时间必须短,关中断会引起串口数据包丢失，请勿在中断内调用上报函数。

步骤 4. DP 点数据上报和下发函数处理

1.所有 DP 数据上报

在模块重启或者重新配网后，WiFi 模块主动下发状态查询指令，此时需要 MCU 上报设备所有 DP 状态给 WiFi 模块进行同步

1).打开 protocol.c 找到函数 all_data_update(void)

2).把所有需要上报的 DP 点初值填入相应上报函数，为面板显示提供初值。

注意：用户请勿随意调用 all_data_update()函数，该函数会在特定时间主动调用

2. 单个 DP 数据上报

在某 DP 点状态发生变化时，mcu 需要主动上报，APP 会更新显示。上报格式为 mcu_dp_xxxx_updata(DPID_X,n),DPID_X 为状态改变的 DP 点。

例如：

```
mcu_dp_bool_update(DPID_SWITCH,1);                    //BOOL 型数据上报
mcu_dp_value_update(DPID_TEMPER_SET,25);              //VALUE 型数据上报
mcu_dp_string_update(DPID_DAY,"1234",4);              //STRING 型数据上报
mcu_dp_raw_update(DPID_SWITCH,1);                     //RAW 型数据上报
mcu_dp_bitmap_update(DPID_TEMPER_SET,25);             //故障型数据上报
mcu_dp_enum_update(DPID_DAY,"1234",4);                //枚举型数据上报
```

3.DP 数据下发处理函数

在 protocol.c 文件中，每个可下发的 DP 点，都有一个单独下发数据处理函数。格式为 dp_download_DPX_handle()DPX 为可下发 DP 点。函数解析功能点之后，用户需映射到相关执行函数，完成具体动作函数。

以接收到开关 DP 数据为例：

```c
/*****************************************************************************
函数名称 : dp_download_switch_handle
功能描述 : 针对DPID_SWITCH的处理函数
输入参数 : value:数据源数据
         : length:数据长度
返回参数 : 成功返回:SUCCESS/失败返回:ERROR
使用说明 : 可下发可上报类型,需要在处理完数据后上报处理结果至app
*****************************************************************************/
static unsigned char dp_download_switch_handle(const unsigned char value[], unsigned short length)
{
  //示例:当前DP类型为BOOL
  unsigned char ret;
  //0:关/1:开
  unsigned char  switch1;

  switch1 = mcu_get_dp_download_bool(value,length);
  if(switch1 == 0)
  {
    MCU_ON_switch1();//开关关
  }
  else
  {
    MCU_OFF_switch1();//开关开
  }

  //处理完DP数据后应有反馈
  ret = mcu_dp_bool_update(DPID_SWITCH,switch1);
  if(ret == SUCCESS)
    return SUCCESS;
  else
    return ERROR;
}
```

MCU_ON_switch1()和 MCU_OFF_switch1()为 MCU 控制开关函数，完成具体动作。当设备状态在非 APP 控制下发生变化，MCU 中需要调用 mcu_dp_bool_update(DPID_SWITCH_1,switch_1);上传功能点(开关)状态实时状态，形成反馈，一般接收处理函数已经自动调用该函数；

步骤 5. 配网功能及闪灯函数完善

当移植协议成功后，要想设备配网，还需要将配网指令及指示完善。模组自处理工作模式的，无需关注此小节。

配合处理模式的 MCU，配网触发方式和指示方式，可以根据实际情况自行决定。通常为按键触发和 LED 快闪/慢闪指示。

配网有两种模式，建议产品都要做上：

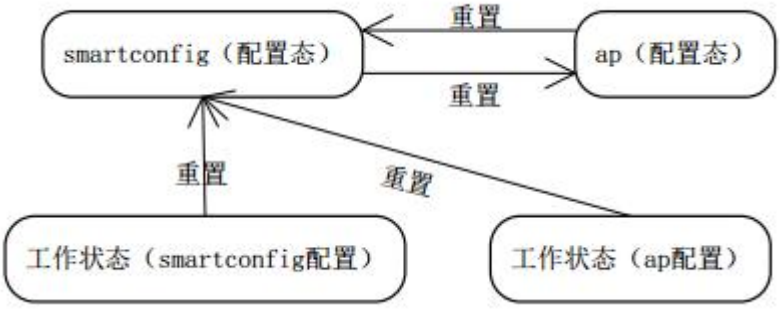
smart 模式(快闪)：操作简便，通常以灯快闪做指示

AP 模式（慢闪）：配网可靠，通常以灯慢闪做指示

1.配网指令

配网指令有两个函数可以实现：mcu_reset_wifi()和 mcu_set_wifi_mode()。

mcu_reset_wifi()调用后复位 wifi 模组，复位后以前的配网信息全部清除。mcu_reset_wifi()调用一次切换一次配网模式，在 AP 和 smart 之间切换。



mcu_set_wifi_mode()参数：SMART_CONFIG、AP_CONFIG。调用后清除配网信息，明确进入 SMART 模式或者 AP 模式。

2.配网指示

通常在 while(1)调用 mcu_get_wifi_work_state()函数返回 wifi 状态，根据 WiFi 状态，写入相应闪灯的模式。

设备联网状态	描述	状态值	LED 显示
状态 1	smart 配置状态	0x00	快闪 250ms
状态 2	AP 配置状态	0x01	慢闪 1500ms
状态 3	WIFI 已配置但未连上路由器	0x02	熄灭
状态 4	WIFI 已配置且连上路由器	0x03	常亮
状态 5	已连上路由器且连接到云端	0x04	常亮
状态 6	WIFI 设备处于低功耗模式	0x05	熄灭

调用函数 mcu_get_wifi_work_state()获取连接状态，函数架构如下：

```
void main(void)

{           ...

    while(1)
    {
        ...
        switch(mcu_get_wifi_work_state())
        {
            case SMART_CONFIG_STATE:
                //Smart 配置状态 LED 快闪 ，led 闪烁请用户完成
            break;
            case AP_STATE:
                //AP 配置状态 LED 慢闪，led 闪烁请用户完成
            break;
            case WIFI_NOT_CONNECTED:
                //WIFI 配置完成，正在连接路由器，LED 常暗
            break;
            case WIFI_CONNECTED:
                //路由器连接成功 LED 常亮
            break;
            case WIFI_CONN_CLOUD:
                //wifi 已连上云端 LED 常亮
            default:break;
        }
        ...

    }
}
```

步骤 6. 产测功能完善

1.MCU 需要支持 wifi 功能测试，打开 protocol.h 开启下面的 define
#define WIFI_TEST_ENABLE //开启 WIFI 产测功能
2.MCU 在需要 wifi 功能测试处调用 mcu_api.c 文件内 mcu_start_wifitest()
3.在 protocol.c 文件 wifi_test_result 函数内查看测试结果

产测功能指令需在模组上电初始化完成后才能调用，建议 5s 后调用，触发条件可用户自行设定。开启产测后，模组会自动搜索名称为 tuya_mdev_test 无线信号，并返回信号强度。因此无线热点名称需改为 tuya_mdev_test。

可选功能：MCU 在线升级

若要支持 MCU 在线升级，打开 protocol.h，开启宏定义，设置 MCU 版本号
#define SUPPORT_MCU_FIRM_UPDATE //开启 MCU 固件升级功能(默认关闭)
#define MCU_VER "1.0.0"
//用户的软件版本,用于 MCU 固件升级,MCU 升级版本需修改

如果数据包较大请根据实际需求调整缓冲区大小：
WIFI_UART_RECV_BUF_LMT 300 //固件升级缓冲区,需大缓存,必须大于 260

相应的升级函数为 protocol.c 中：

protocol.c

```
/*****************************************************************************
函数名称 : mcu_firm_update_handle
功能描述 : MCU进入固件升级模式
输入参数 : value:固件缓冲区
           position:当前数据包在于固件位置
           length:当前固件包长度(固件包长度为0时,表示固件包发送完成)
返回参数 : 无
使用说明 : MCU需要自行实现该功能
*****************************************************************************/
unsigned char mcu_firm_update_handle(const unsigned char value[],unsigned long position,unsigned short length)
{
  #error "请自行完成MCU固件升级代码,完成后请删除该行"
  if(length == 0)
  {
    //固件数据发送完成

  }
  else
  {
    //固件数据处理
  }

  return SUCCESS;
}
#endif
```

MCU 可调用 mcu_api.c 文件内的 mcu_firm_update_query()函数获取当前 MCU 固件更新情况。
注：升级过程由手机发起，调试时可使用涂鸦串口调试助手点击升级启动，协助调试

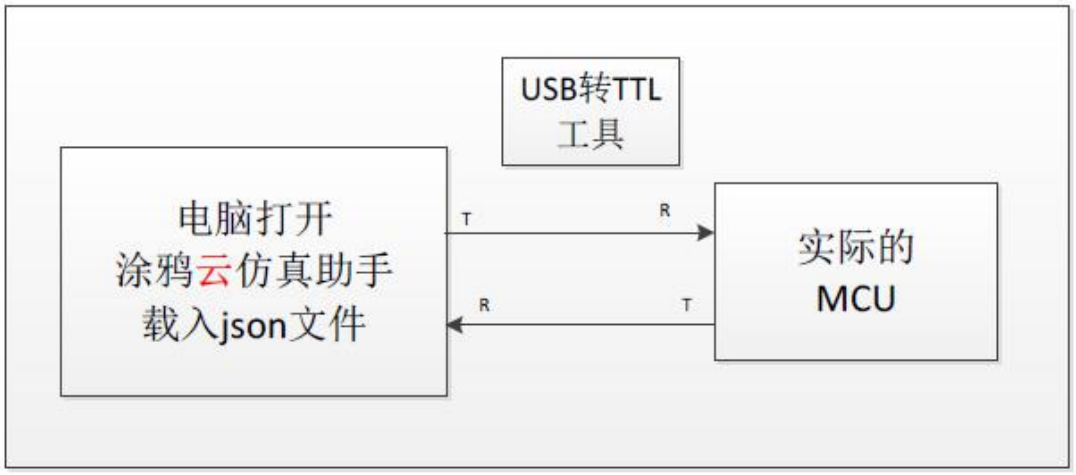
四、两个串口模拟工具的使用

为提高对接效率，了解协议格式，验证数据正确性，涂鸦提供了两款模拟助手，一款模拟模组端，一款模拟 MCU 端，配合使用可加快开发效率。两款助手使用说明：https://docs.tuya.com/cn/mcu/debug_assistant.html

4.1 涂鸦云串口调试助手

涂鸦云串口调试助手，模拟 WiFi 模组的数据收发。连接 MCU 可验证 MCU 数据发送是否满足涂鸦的通信协议，确定是否移植成功。

注：涂鸦云串口调试助手只能验证收发协议格式正确性，没有联网功能。

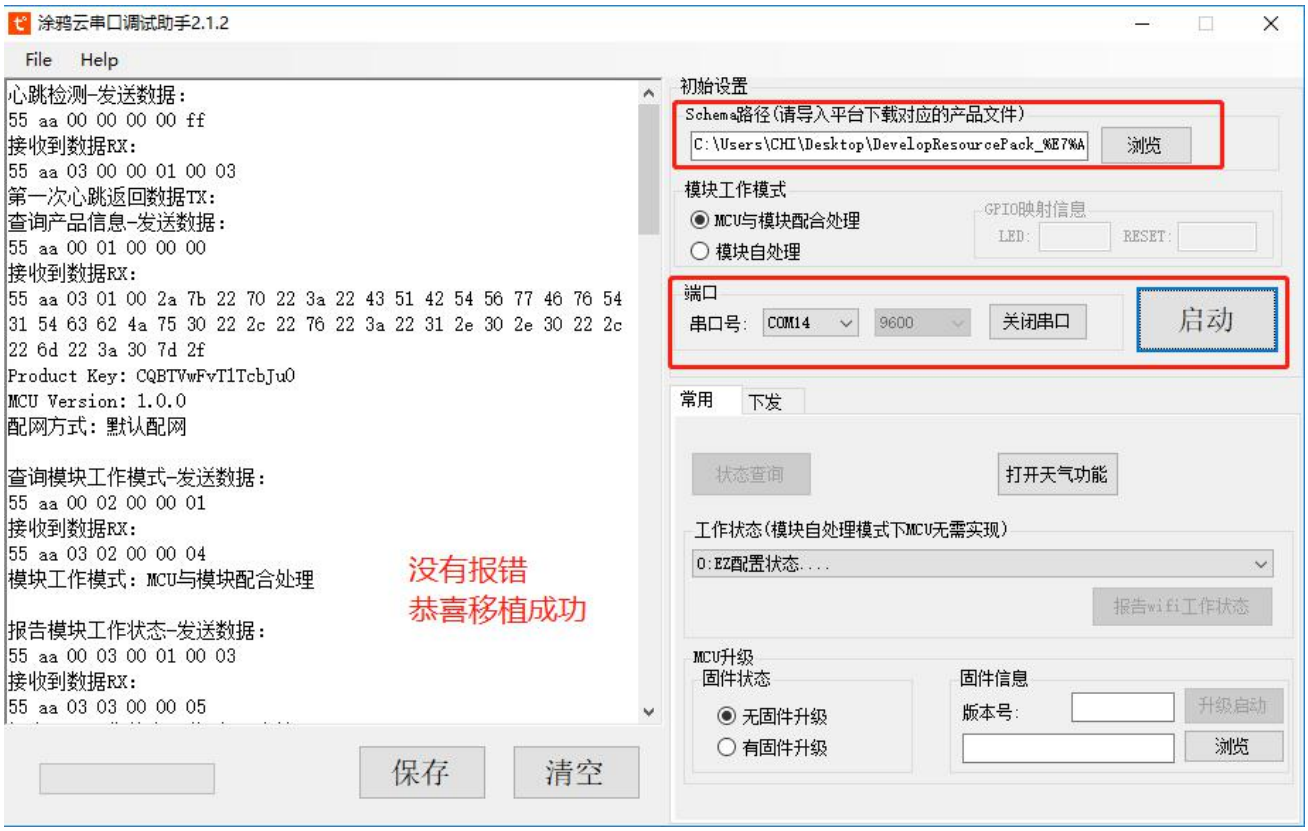


使用方法：

1.连接单片机的串口和电脑的串口

2.打开涂鸦云串口调试助手.exe

3.点击浏览，导入资料包中的 json 文件，点击启动



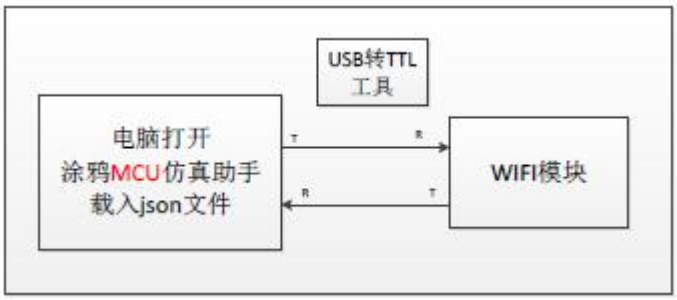
4.2 涂鸦 MCU 仿真调试助手

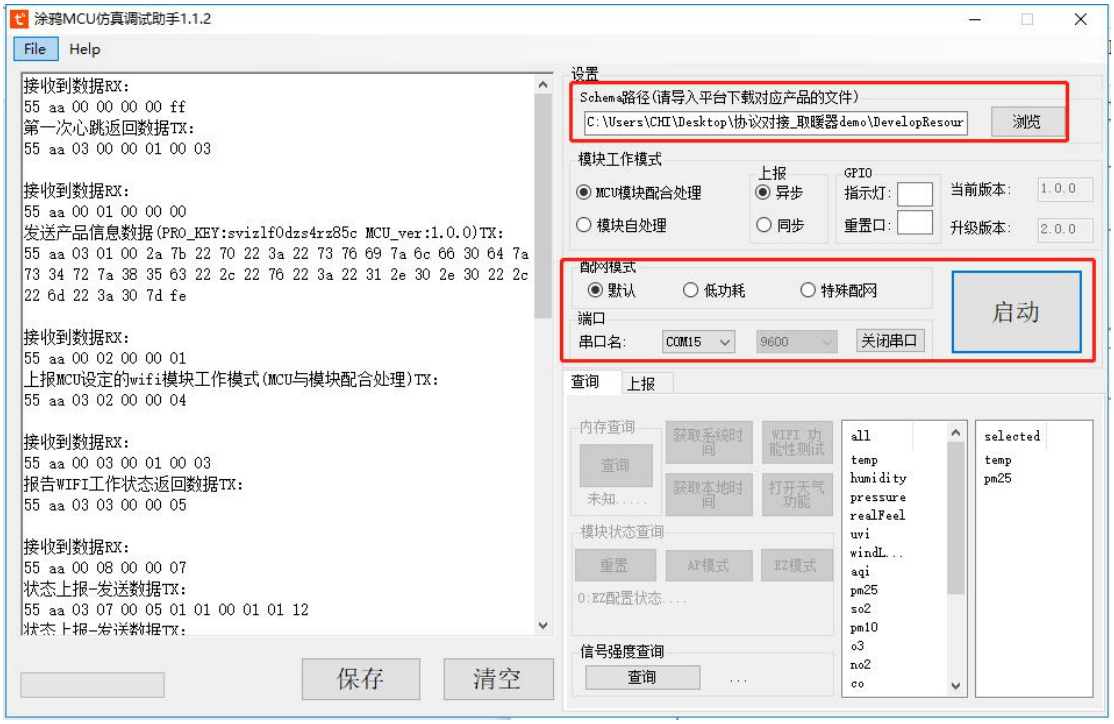
涂鸦 MCU 仿真调试助手，模拟 MCU 的数据收发。将 wifi 模组最小系统搭建好，连接涂鸦 MCU 仿真调试助手可有以下用途：

1、 与联网模块配对使用，验证 wifi 模块是否正常工作。

2、 在 MCU 未完成开发前，调试 APP 面板显示

3、 当 MCU 开发者不知如何发送、回复数据给模块时，仿真助手的数据可作为参考。





五、SDK 函数架构解析


SDK程序结构图
V1.0.0 栎子
wifi_protocol_init() 初始化函数
queue_in/out=wifi_queue_buf; 初始化队列指针
wifi_uart_service （） 串口数据处理函数
wifi_uart_rx_buf[rx_in ++] = Queue_Read_Byte(); 取队列数据函数
Queue_Read_Byte （） 读队列单字节函数
data_handle （） 数据处理函数
switch（cmd_type）判断命令字
heat_beat_check （） 心跳检测
product_info_update （） 产品信息
get_mcu_wifi_mode （） 查询模块工作模式
data_point_handle （） 下发DP点数据处理
all_data_update （） 全DP点状态上报函数
uart_receive_input （） 串口接收函数
*queue_in ++ = value; 存数据到队列buffer