



# Tuya\_RTOS 平台接入测试说明

版本号	时间	作者	修改内容
1.0.0	2021-07-29	王阳	初订版本

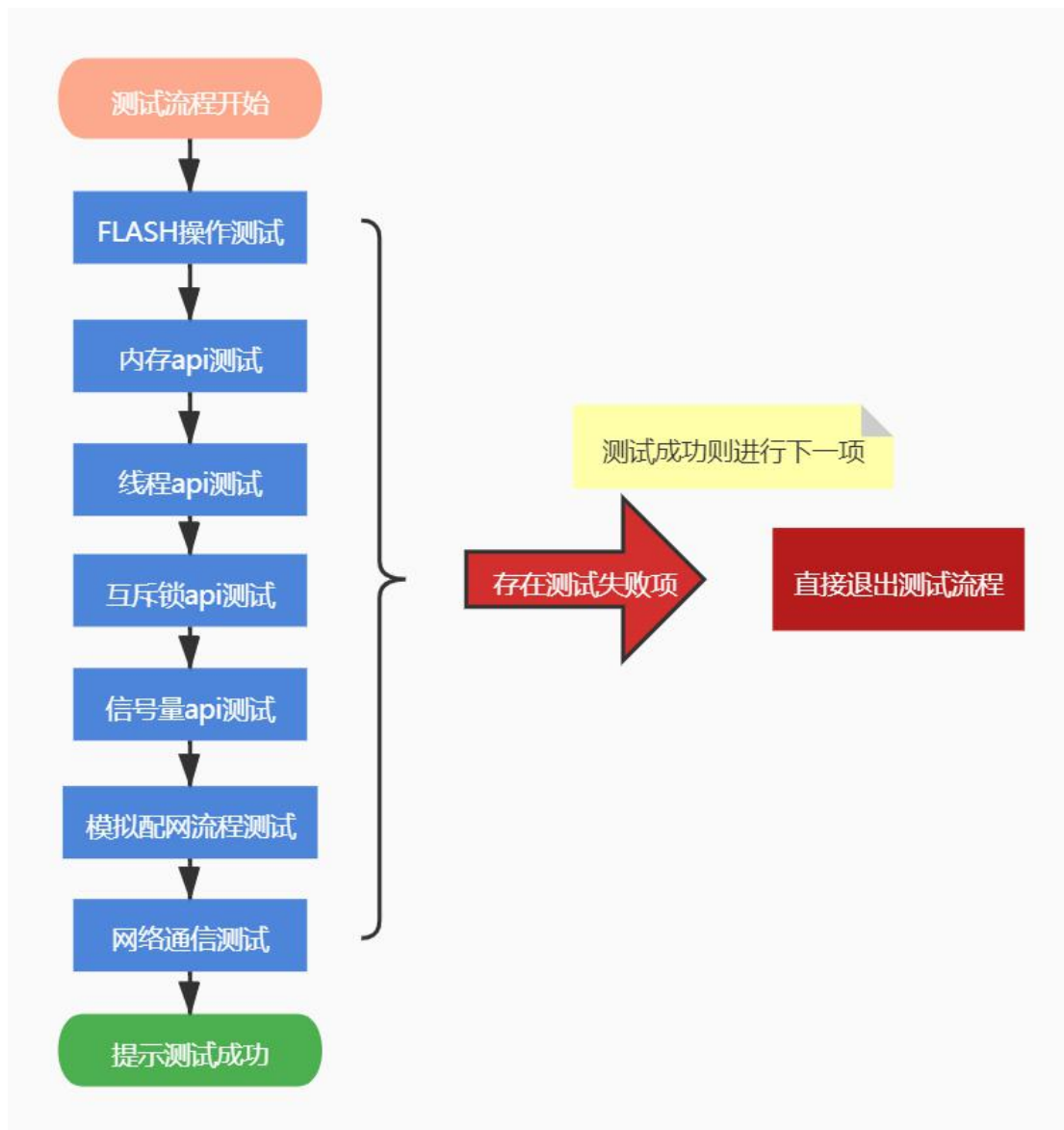


## 目录

Tuya_RTOS 平台接入测试说明.....	1
一、 简介.....	3
二、 文档说明.....	3
三、 测试模块说明.....	4
（一）FLASH 操作测试.....	4
1. 测试成功.....	5
2. 测试失败（未展示全部场景）.....	5
（二）内存 api 测试.....	5
1. 测试成功.....	6
2. 测试失败（未展示全部场景）.....	6
（三）线程 api 测试.....	6
1. 测试成功.....	6
2. 测试失败（未全部展示测试失败场景）.....	6
（四）互斥锁 api 测试.....	6
1. 测试成功.....	7
2. 测试失败（未展示全部场景）.....	7
（五）信号量 api 测试.....	7
1. 测试成功.....	7
2. 测试失败（未展示全部场景）.....	8
（六）配网流程模拟测试.....	8
（1） 测试成功.....	10
（2） 测试失败（未展示全部场景）.....	10
（七）网络通信测试.....	10
1. UDP 通信测试.....	10
2. TCP 通信测试.....	12
（八）枚举类型的长度测试.....	13
三、 注意事项.....	14


## 一、简介

为了提高客户导入速度，在涂鸦音视频 SDK 输出前，我们提供此 Demo 供客户适配。该 Demo 中使用到的函数都是涂鸦提供的适配层 api，主要作用是提前检查客户适配的函数是否正确（包括返回值、实现内容等）。Demo 的主体框架流程如下：




## 二、文档说明

压缩包里面主要包括了说明文档，测试验证的 Demo SourceCode,请参考说明文档了解


 tuya\_cloud\_error\_code.h

Demo 的测试流程。其中，

 tuya\_os\_adapter.rar

为适配层代码及头文件，也是我们

我们 libtuya\_iot.a 对接使用到平台相关函数接口，需要客户自己根据平台来实现具体过程方法。

 tuya\_test\_demo.c

这个是检查适配层的 demo 代码，客户需要将适配层代码、头文件、demo 代码加入到平台应用工程里面用于验证适配接口正确与否。

### 三、测试模块说明

#### (一) FLASH 操作测试

这一测试项主要是测试当前 HAL 层中 flash 操作相关 api 实现是否能够正常使用,注意主要是检测 api 功能,不确定 flash 分区划分是否正确。该测试项主要从 erase、read、write 三个方向来检测,测试程序中相关宏定义如下:

```
#define FLASH_TEST_BUFF "test"    //由使用者指定测试的数据

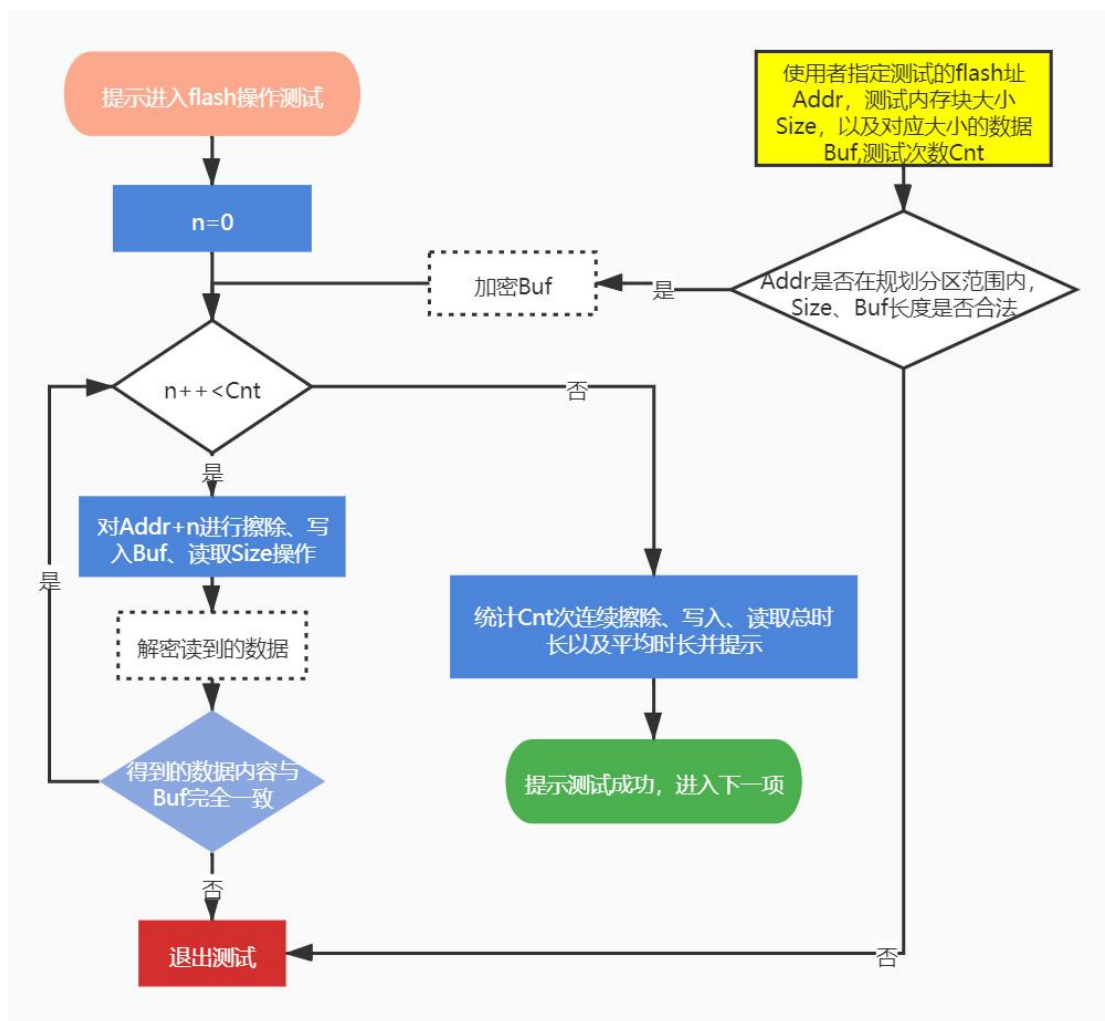
#define FLASH_TEST_CNT 100        //由使用者指定测试的次数

#define UF_TEST_ADDR 0x3E4000    //由使用者指定 flash 操作地址,注意请选择 uf 分区中的地址

#define FLASH_UF_SIZE 0x8000     //uf 分区大小(适配层中有宏定义,可参考)

#define FLASH_UF_ADDR 0x3E3000   //uf 分区起始地址(适配层中有宏定义,可参考)
```

测试流程如下,图中虚线框加解密部分为非必要部分,若需要可自行添加:



测试结果展示：

### 1. 测试成功

```
Test: [WARNING] flash api erase/write/read 100 times cost 4054ms.
Test: [WARNING] average time = 40.540000.
*****
Test: [WARNING] flash api test success.
*****
```

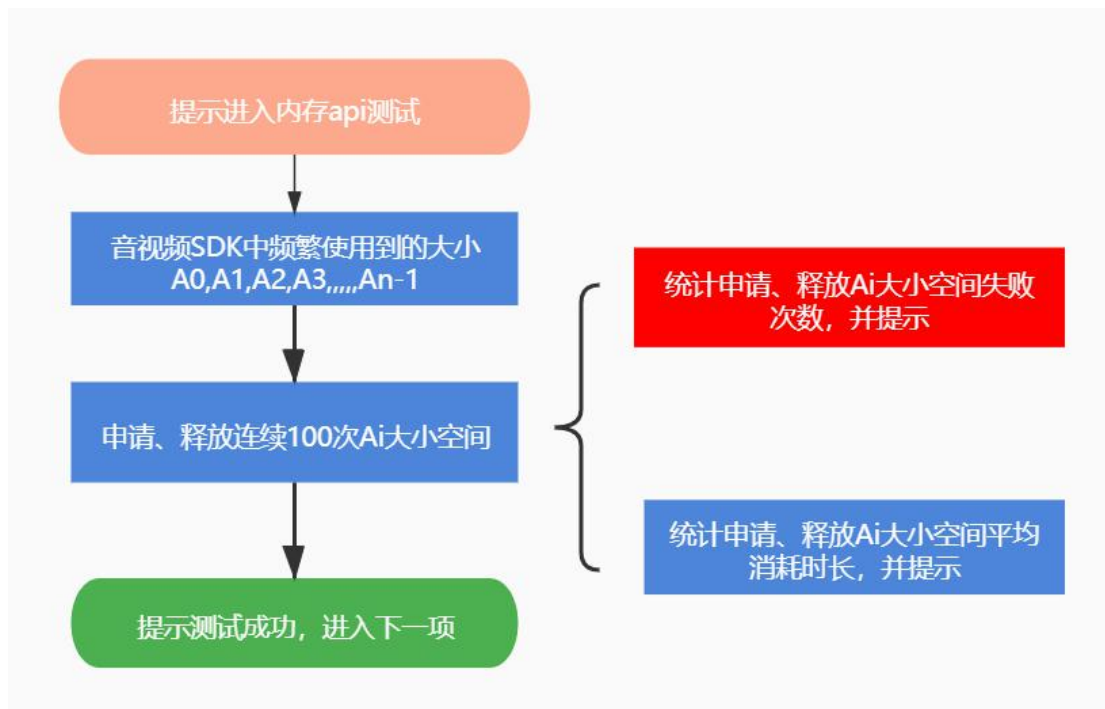
### 2. 测试失败（未展示全部场景）

```
Test: [ERROR] flash test read buff err, exit...
Test: [ERROR] flash api test err, exit...
*****
Test: [ERROR] flash api test err, exit...
*****
```

## （二）内存 api 测试

由于涂鸦音视频 SDK 中存在大量频繁的内存操作，之前对接 rtt 系统时，出现过频繁申请、释放空间导致内存碎片化严重问题，故将内存 api 独立为一个测试项。该测试项主要目的是告知客户当前设备系统平台对频繁申请、释放内存的承受能力，以便于及时发现问题并优化。该测试项涉及到的宏定义如下：

```
#define MEMORY_OPERATE_CNT 1000 //由使用者指定操作次数
```



测试结果展示如下：

### 1. 测试成功

```
Test: [DEBUG] size_array[6] = 4096.
Test: [DEBUG] begin time 4896.
Test: [DEBUG] end time 4905.
Test: [WARNING] operate malloc and free size = 4096, cost 9ms, average operating time is 0.009000ms, malloc fail cnt is 0.
Test: [DEBUG] size_array[7] = 10240.
Test: [DEBUG] begin time 4916.
Test: [DEBUG] end time 4924.
Test: [WARNING] operate malloc and free size = 10240, cost 8ms, average operating time is 0.008000ms, malloc fail cnt is 0.
Test: [DEBUG] size_array[8] = 16384.
Test: [DEBUG] begin time 4935.
Test: [DEBUG] end time 4944.
Test: [WARNING] operate malloc and free size = 16384, cost 9ms, average operating time is 0.009000ms, malloc fail cnt is 0.
*****
Test: [WARNING] memory api test success.
*****
```

### 2. 测试失败（未展示全部场景）

无，使用者可根据 malloc fail cnt 的值做判断。

## （三）线程 api 测试

这一项主要是为了测试 SDK 中会使用到的线程相关 api，主要有创建和释放，创建线程时 SDK 会将优先级和线程名传递给设备端，在调试阶段时，若碰到优先级引起的调度问题，可由设备端自行根据线程名改变线程优先级。测试流程如下：创建 test 线程，并在线程处理函数中循环处理 5 次工作后，退出，释放线程。循环以上步骤 5 次，如果中间出现工作结果出错，或是线程创建出错，则提示出错内容后，直接退出测试程序。测试程序中使用到的宏如下：

```
#define THREAD_OPERATE_CNT 5 //由使用者指定操作次数
```

测试结果展示如下：

### 1. 测试成功

```
Test: [DEBUG] thread working...
Test: [DEBUG] p_handle->exit = 2.
*****
Test: [WARNING] thread api test success.
*****
```

### 2. 测试失败（未全部展示测试失败场景）

无，若未出现成功提示，请检查 create 和 release 接口

## （四）互斥锁 api 测试

HAL 层关于 mutex 的 api 主要有创建、加锁、解锁、销毁。这一项测试流程如下：创建一个 mutex，两个线程 thread1，thread2，一个 Buf，线程一向 Buf 中逐个字符填充”hello”，线程二向 Buf 中逐个字符填充”world”，在线程填充处加锁，每填充一个字符后都会给 1 秒延时而后解锁，最后得到 Buf，释放线程，在 Buf 中判断是否每个”world”，”hello”都完整，若完整则测试通过，反之测试失败，退出测试流程。该测试项涉及到的宏定义如下：

```
#define THREAD_OPERATE_CNT 5 //由使用者指定操作次数
```



测试结果展示如下：

1. 测试成功

```
*****
Test: [WARNING] 4、enter the mutex api test...
*****
Test: [DEBUG] mutex thread create ok.
Test: [DEBUG] mutex lock in thread1.
Test: [DEBUG] mutex unlock in thread1.
Test: [WARNING] mutex test thread1_func exit.
Test: [DEBUG] mutex lock in thread2.
Test: [DEBUG] mutex unlock in thread2.
Test: [WARNING] mutex test thread2_func exit.
Test: [DEBUG] mutex test, buf : helloworld==
*****
Test: [WARNING] mutex test success.
*****
```

2. 测试失败（未展示全部场景）

```
*****
Test: [WARNING] 4、enter the mutex api test...
*****
Test: [DEBUG] mutex thread create ok.
Test: [DEBUG] mutex lock in thread1.
Test: [DEBUG] mutex lock in thread2.
Test: [DEBUG] mutex unlock in thread1.
Test: [WARNING] mutex test thread1_func exit.
Test: [DEBUG] mutex unlock in thread2.
Test: [WARNING] mutex test thread2_func exit.
Test: [DEBUG] mutex test, buf : hweolrlld==
Test: [ERROR] mutex test err, exit.
*****
Test: [ERROR] mutex api test err, exit...
*****
```

（五）信号量 api 测试

HAL 层关于 semaphore 的 api 主要有创建、post、wait、销毁。这一项测试流程如下：创建一个 semaphore、两个线程 thread3 thread4，一个线程 post semaphore，每 post 一次 semaphore，num 的值加 1；另一个线程 wait semaphore，每 wait 成功一次 semaphore，num 的值减 1，最后根据 num 的值判断 semaphore api 是否正确实现。测试结果展示如下：

1. 测试成功

```
Test: [DEBUG] sem thread3 work end, exit : 1, num : 0.
Test: [WARNING] sem test thread3_func exit.
Test: [DEBUG] thread4_func wait sem success.
Test: [DEBUG] sem test -1.
Test: [WARNING] sem test thread4_func exit.
[E/NTP]: ERROR no such host
*****
Test: [WARNING] semaphore test success.
*****
```

## 2. 测试失败（未展示全部场景）

```

*****
Test: [WARNING] 5、enter the semaphore api test...
*****
semaphore addr : 0x904160.
Test: [DEBUG] thread3_func post sem success.
Test: [DEBUG] sem test thread3 : 1.
Test: [DEBUG] thread4_func wait sem success.
Test: [DEBUG] sem test 0.
Test: [DEBUG] thread4_func wait sem success.
Test: [DEBUG] sem test -1.
Test: [WARNING] sem test thread4_func exit.
Test: [ERROR] thread3 test sem failed.
Test: [DEBUG] sem thread3 work end, exit : -1, num : -1, flag : -1.
Test: [WARNING] sem test thread3_func exit.
Test: [ERROR] semaphore test err.
*****
Test: [ERROR] semaphore api test err, exit...
*****

```

### （六）配网流程模拟测试

配网模拟测试流程如下：

1. 提示"已经进入配网流程"
2. 获取 serialNo: tuya\_hal\_get\_serialNo
3. 设置国家码: tuya\_hal\_wifi\_set\_country\_code
4. 提示"模拟操作 flash 中..."
5. 提示"模拟操作 flash 完成"
6. 提示"首先测试进入 ap 配网模拟测试"
7. 设置网卡工作模式为 SoftAp 模式: tuya\_hal\_wifi\_set\_work\_mode(WWM\_SOFTAP)
8. 获取网卡 station 模式下的 mac 地址: tuya\_hal\_wifi\_get\_mac(WF\_STATION, &mac)
9. 将 Ap 的配置信息传给设备，设备收到后自己开启 Ap: tuya\_hal\_wifi\_ap\_start(cfg)
10. tuya\_hal\_net\_socket\_create  
tuya\_hal\_net\_set\_reuse  
tuya\_hal\_net\_bind  
tuya\_hal\_net\_recvfrom ==>通过网络调试助手 udp 通信发送 ssid 和 password（格式见第三章）
11. 提示"已经获取到 wifi 配置信息"
12. 通知设备端关闭 Ap 热点: tuya\_hal\_wifi\_ap\_stop
13. 通知设备端切换为 station 模式: tuya\_hal\_wifi\_set\_work\_mode(WWM\_STATION)





14. 通知设备去连接网络: `tuya_hal_wifi_station_connect(ssid,passwd)`
15. 获取当前的网卡工作模式: `tuya_hal_wifi_get_work_mode`
16. 获取 status: `tuya_hal_wifi_station_get_status` ==>预想要得到的值 WSS\_GOT\_IP 状态
17. 提示"连接网络成功, Ap 配网模式模拟成功"
18. 断开网络连接: `tuya_hal_wifi_station_disconnect`
19. 设置当前模式为 station 模式: `tuya_hal_wifi_set_work_mode(WWM_STATION)`
20. `tuya_hal_wifi_release_ap`
21. 获得当前环境中存在的所有热点: `tuya_hal_wifi_all_ap_scan`
22. 获得 mac 地址: `tuya_hal_wifi_get_mac(WF_STATION, &mac)`
23. 设置为 sniffer 模式: `tuya_hal_wifi_set_work_mode(WWM_SNIFFER)`
24. `tuya_hal_wifi_sniffer_set(true, cb)`
25. 提示"正常 SDK 中需要设备端将 sniffer 模式下抓到的网络数据包通过 cb 不断地传递给 SDK 解析, Demo 中直接使用固定网络配置信息"
26. =====>直接填充 ssid 和 password
27. 获取当前信道: `tuya_hal_wifi_get_cur_channel`
28. 通知设备端切换为 station 模式: `tuya_hal_wifi_set_work_mode(WWM_STATION)`
29. 通知设备去连接网络: `tuya_hal_wifi_station_connect(ssid,passwd)`
30. 获取当前的网卡工作模式: `tuya_hal_wifi_get_work_mode`
31. 获取 status: `tuya_hal_wifi_station_get_status` ==>预想要得到的值 WSS\_GOT\_IP 状态
32. 提示"连接网络成功, 配网模式模拟成功"

该测试项涉及到的宏定义如下:

```
//connect network test
```

```
#define UDP_RECV_MAX 1024
```

```
#define UDP_SERVER_PORT 7789
```

```
#define DEMO_DEF_AP_SSID_PF "SmartLife"
```



测试结果展示如下：

### (1) 测试成功

```
Test: [DEBUG] Ap mode wifi config : ssid:16F-S-4-19;pw:tuya1602.=
Test: [DEBUG] Ap mode get ssid : 16F-S-4-19.=
Test: [DEBUG] Ap mode get passwd : tuyal602.=
Test: [DEBUG] get ssid and password success.
Test: [DEBUG] wifi_ssid : 16F-S-4-19, wifi_password : tuyal602.

Test: [WARNING] got ip success...
*****
Test: [WARNING] Ap mode test success.
*****

Test: [WARNING] got ip success...
*****
Test: [WARNING] Smart mode test success.
*****
```

### (2) 测试失败（未展示全部场景）

```
Test: [ERROR] network test get wifi config err.
Test: [DEBUG] test fail, exit.
*****
Test: [ERROR] connect network test err, exit...
*****
```

## (七) 网络通信测试

此测试选项分为 tcp/udp 两种连接方式，该测试项在配网模拟测试项之后，故而在上一项测试通过时，设备应具有在局域网下通信的能力。此测试项需要准备网络测试工具。测试流程如下：先测试 tcp 通信，tcp 通信成功后再测试 udp 通信。测试程序中有指定的 tcp、udp 连接端口，测试通信的方式为发送、接收一段固定数据，数据校验成功则测试成功，否则测试失败。

### 1. UDP 通信测试

udp 通信测试用到的宏定义如下：

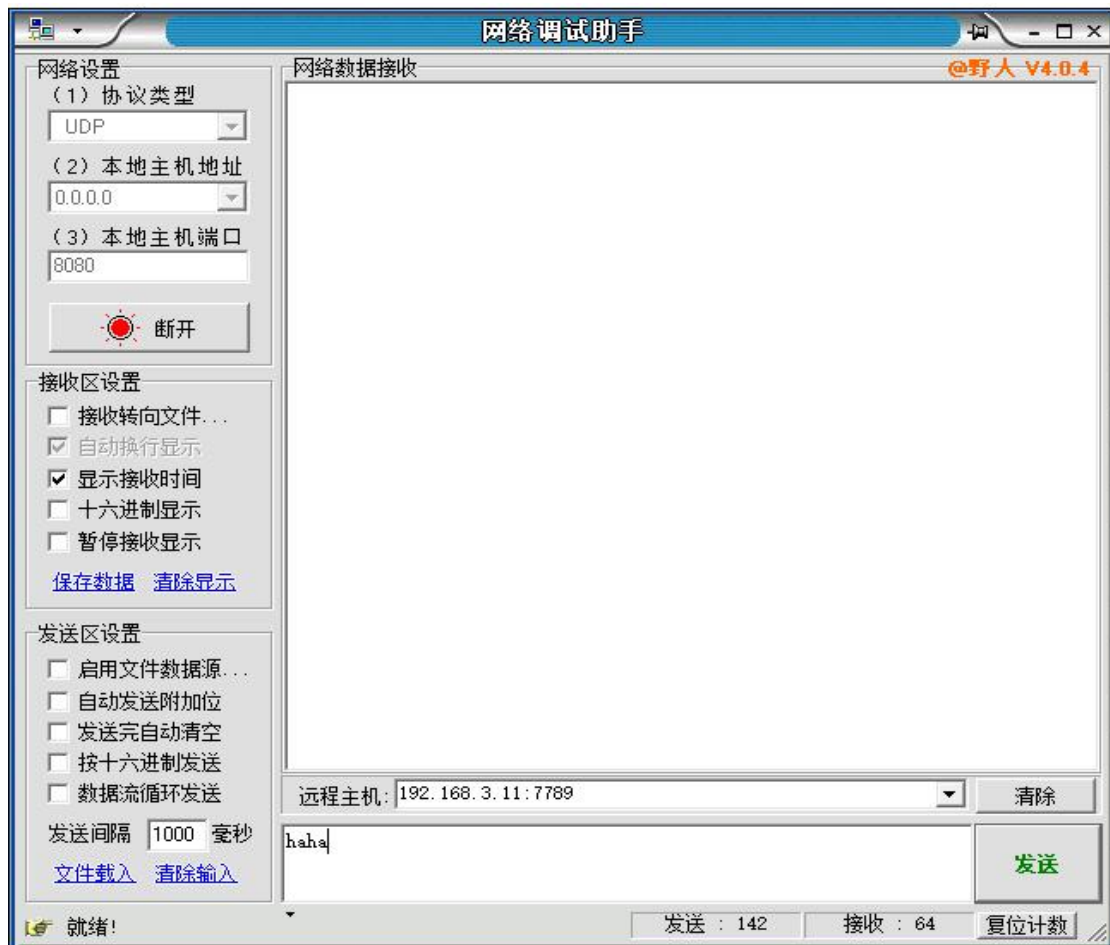
```
#define UDP_RECV_MAX 1024
```

```
#define UDP_SERVER_PORT 7789
```

```
#define UDP_TIMEOUT_MAX 100
```

Udp 交互流程（端口：7789）

- a. 首先需要使用者使用网络调试工具随意发送一串数据给设备端



- b. 设备端收到后，会有如下提示

```
Test: [DEBUG] Udp socket has readfd.
Test: [DEBUG] Udp Server rcv begin test from client ip -1062730996, port 8080.
Test: [WARNING] Udp test: please send "begin test" to the udp server.
```

- c. 使用者通过网络调试工具发送“begin test”给设备端

- d. 设备端收到正确的“begin test”后，会发送“tuya api test”，并有如下提示：

```
Test: [DEBUG] Udp socket has readfd.
Test: [DEBUG] Udp Server rcv begin test from client ip -1062730996, port 8080.
Test: [WARNING] Udp test: please send the string to server that you receive.
```

- e. 使用者通过网络调试工具发送“tuya api test”给设备端

- f. 设备端收到正确数据后，提示 udp 测试成功

```
*****
Test: [WARNING] udp test: test success.
*****
```

## g. 测试失败提示

```
*****
Test: [ERROR] udp transmission test err, exit...
*****
```

## 2. TCP 通信测试

TCP 通信测试用到的宏定义如下：

```
#define TCP_SERVER_PORT 7788

#define TCP_RECV_MAX    1024

#define TCP_SEND_MAX    1024

#define TCP_TIMEOUT_MAX 30
```

Tcp 交互流程（端口：7788）：

a. 首先需要使用者使用网络调试工具搭建客户端，并连接到设备端的 server



b. 连接设备端 server 成功后，设备端会有如下提示：

```
Test: [DEBUG] tcp :tcp socket begin select.
Test: [ERROR] tcp :Tcp socket has readfd.
Test: [DEBUG] tcp :Tcp client connected, client_fd 1.
Test: [DEBUG] welcome to client 1 room.
Test: [WARNING] please send "begin test" to the tcp server.
```





c. 使用者通过网络调试助手发送“begin test”给设备端 server

d. 设备端收到“begin test”后，会有如下提示，若未收到正确的数据则会接收到的数据发送给网络调试助手搭建的客户端，直到收到正确的数据。

```
Test: [WARNING] please send "begin test" to the tcp server.
Test: [ERROR] tcp : client socket 1 has readfd.
Test: [DEBUG] tcp : client 1 recv data : ssid:16F-S-4-19;pw:tuya1602.
Test: [DEBUG] tcp : recv_size : 27.
Test: [ERROR] tcp : client socket recv the test word : ssid:16F-S-4-19;pw:tuya1602.
Test: [ERROR] tcp : client socket 1 has readfd.
Test: [DEBUG] tcp : client 1 recv data : begin test.
Test: [DEBUG] tcp : recv_size : 10.
Test: [WARNING] tcp : please send "Tuya api test" to the tcp server.
```

e. 根据提示，使用者需要通过网络调试助手发送“Tuya api test”给设备端 server

f. 设备端 server 成功接收后，会有如下提示：

```
Test: [DEBUG] tcp : recv_size : 10.
Test: [WARNING] tcp : please send "Tuya api test" to the tcp server.
Test: [DEBUG] tcp : client 1 recv tuyaa api test success.
Test: [ERROR] tcp : client socket 1 has readfd.
Test: [DEBUG] tcp : client 1 recv data : Tuya api test.
Test: [DEBUG] tcp : recv_size : 13.
Test: [WARNING] tcp : client 357125 tcp test over.
Test: [DEBUG] exit flag : 1.
Test: [DEBUG] tcp test over.
*****
Test: [ERROR] tcp transmission test success
*****
```

g. 测试失败提示参考 udp 通信测试

#### （八）枚举类型的长度测试

定义一个枚举类型，打印出大小，如果计算的 enum 类型小于 4 字节，会有相应提示建议设备端设置为 4 字节（不强制）。



### 三、注意事项

1、涂鸦测试程序中存在三种打印信息提示（如下图所示）：DEBUG, WARNING, ERROR，其中如果遇到 ERROR 信息则表示为当前程序中有错误发生，请及时纠正后再进行测试。

```
app_init finished
*****
Test: [DEBUG] Welcome to Tuya api test.
*****

*****
Test: [WARNING] Ap mode test success.
*****

*****
Test: [ERROR] mutex api test err, exit...
*****
```

2、涂鸦提供的测试程序需要使用者集成到自己的代码工程中后，进行编译烧录测试。

3、进行 FLASH 操作测试时，即使测试项测试通过也不能确定当前分区划分一定正确。

4、在配网模拟测试项的 AP mode 测试时，需要使用者连接到设备发出的热点，并使用网络调试工具通过 udp 以“ssid:xxxx;pw:xxxx”格式发送网络配置信息到设备，设备解析到后将会进行连接网络操作。

5、如果本身设备不支持 Ap mode 配网，可以通过宏定义，提前写好网络配置信息后，调用 `tuya_smart_mode_test` 函数进行配网模拟测试。

6、UDP/TCP 测试项测试时，测试程序中分别搭建了一个 UDP/TCP server，使用者需要通过网络调试工具和设备进行通信，并根据打印提示完成测试。

7、枚举测试时，当提示枚举为 1 字节时，建议修改为 4 字节对齐。

8、代码中存在很多测试用到的宏定义，请使用者根据注释自行修改。

Eg: `#define TEST_THREAD_PRIORITY 20 //测试程序中线程优先级`

9、tcp 通信测试时，测试程序中只接受一个客户端连接，当未接收到指定字符串时，服务端程序会将接收到的非“正确”字符串发还客户端直到收到“正确的字符串”。

10、Udp, tcp 通信测试、以及 Ap mode 接收 wifi 信息中都设有超时机制，若在指定时间内未接收到数据，则认为测试失败。