Tuya Zigbee SDK 开发者指南

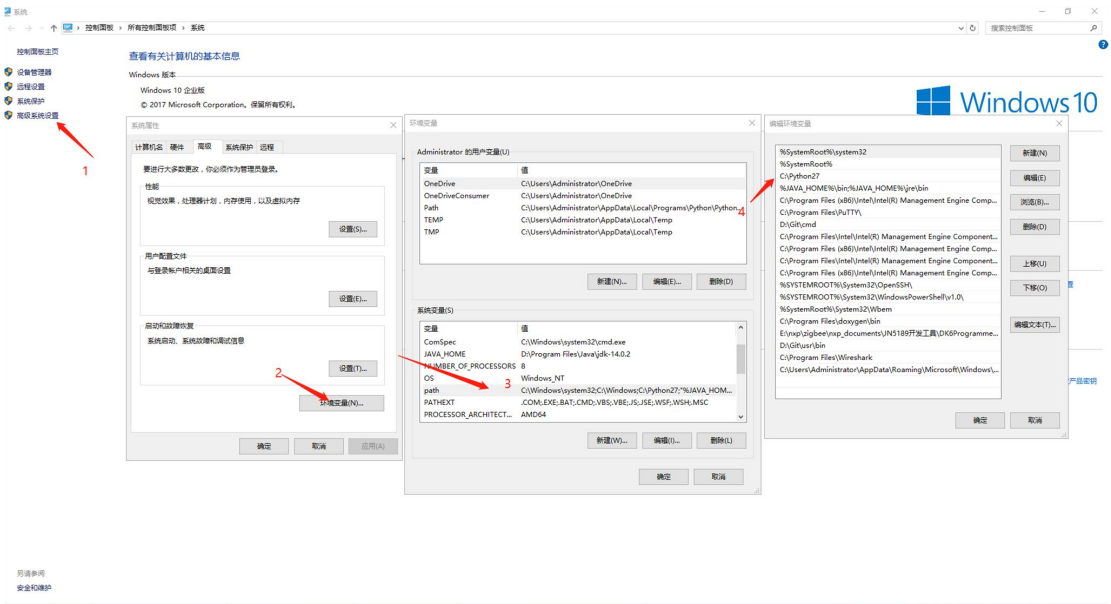
版本号	修订说明	修订人	修订日期
V1.0.0	初版	马晶	2021-9-16
V1.0.1	新增 dp 增加说明 增加 dp 下行回调接口说明	马晶	2021-10-18
V1.0.2	1. 自恢复函数使用说明，2. dp 透传方式使用说明	马晶	2021-12-15
V1.0.3	增加 OTA 大小说明	马晶	2022-1-24

目录

1. 环境搭建

a. 芯科平台安装 IAR8.40 及以上版本。

b. 安装 python2.7.x 版本，并设置好环境变量，切记不要以高版本替代否则生产文件会出错。



c. 芯科平台安装 Command 工具用于烧录。

2. IOT 平台操作

2.1 申请 IOT 账号权限

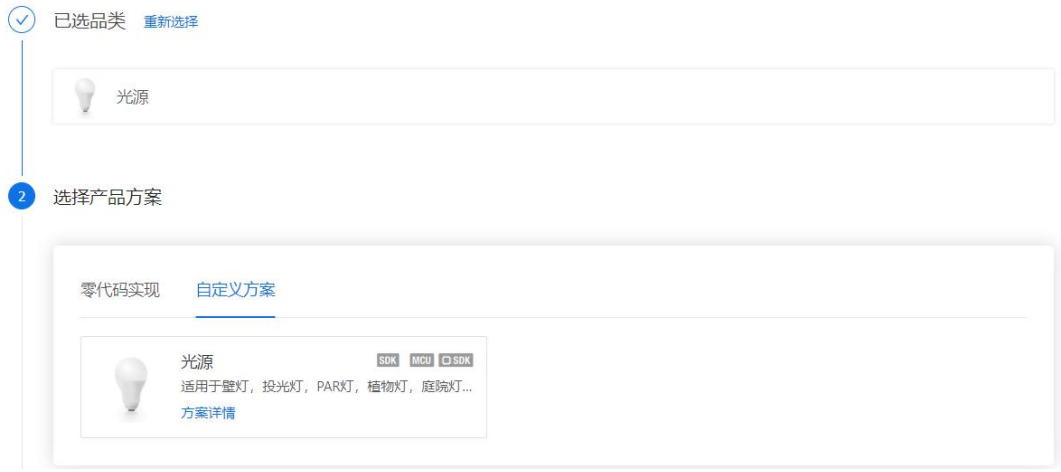
用户可个人申请 IOT 平台账号，网址：https://www.tuya.com/cn。

2.2 创建产品

a. IOT 平台中选择创建产品

b. 选择要创建的产品类型

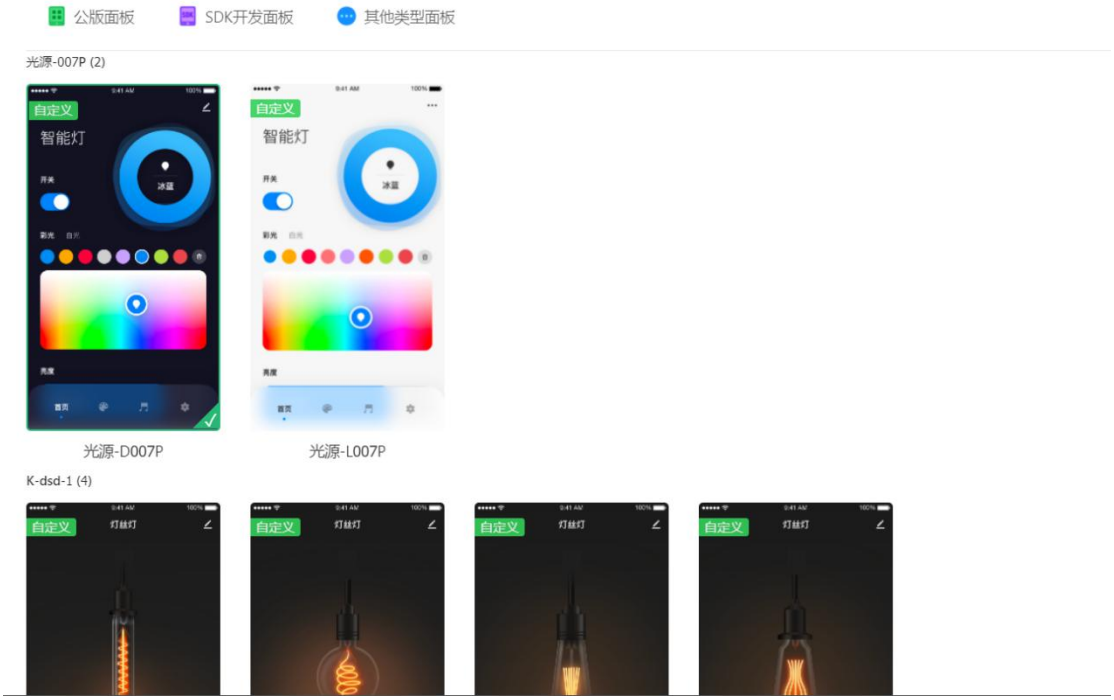
c. 选择自定义方案，并设定产品名称及通讯协议等来完善产品信息。信息完善后点击创建产品即可进入下一步产品开发界面。



2.3 IOT 产品配置

a. 找到已创建的产品，选择继续或从创建产品界面完成产品品类等选择及信息配置后开发进入开发页面。

b. 根据需求选择功能 DP 点，及设备面板、硬件开发、产品配置等。



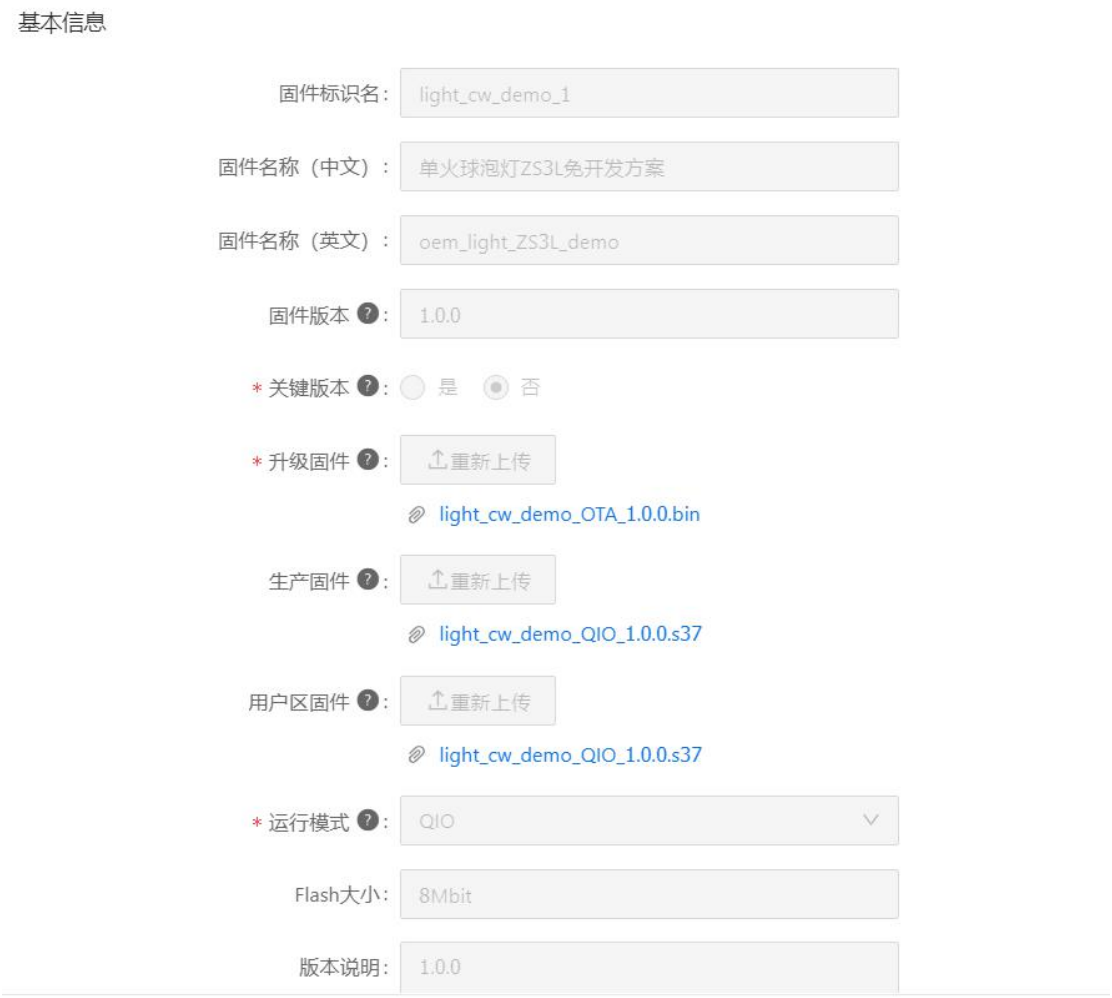
c. 硬件开发中，已选云端对接方式根据需求，可选择根据涂鸦标准模组 MCU SDK，或者 TuyaOS 两种方式。在选择完云端对接方式之后，可以选择我们所使用的 Zigbee 模组。这里我们选择使用涂鸦标准模组 SDK 开发，使用芯科的 ZS3L 模组（泰凌微的模组可选择 ZT 开头的模组）;MCU SDK 为 MCU 串口对接的方案。

d. 在页面底部有下载资料入口，开发者可以下载相应芯片的 SDK，开发者先仔阅读 SDK 目录下的 readme.txt 文档，demo 可以直接编译生成相应的固件，生产的固件可以上次平台进行烧录授权。若开发者只进行开发调试可以略过后续步骤。

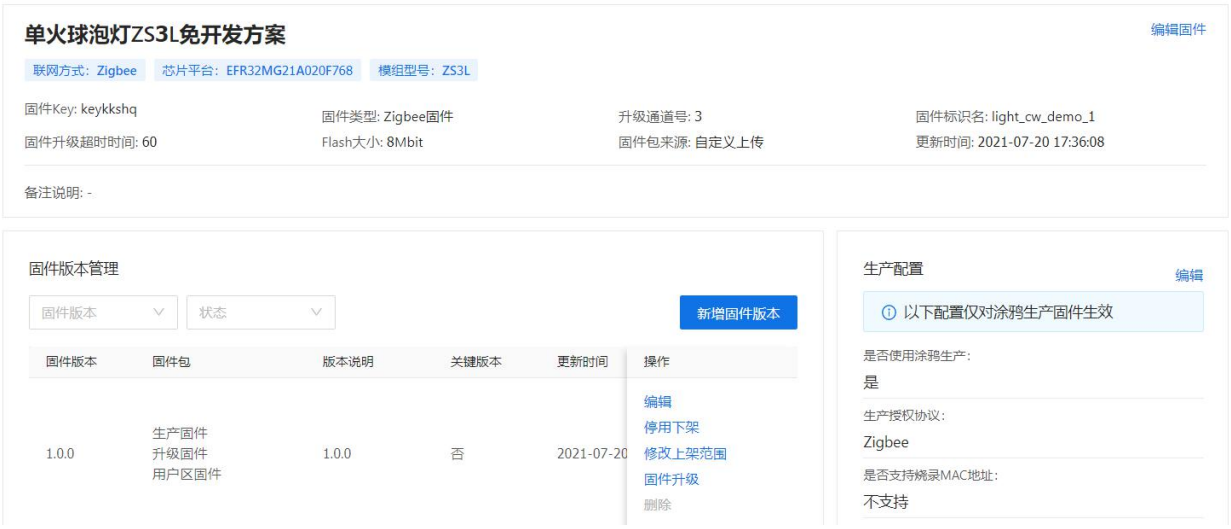
e. 在选择完模组及云端对接方式之后，在原本的界面会出现一个固件选择框，通过点击新增自定义固件，可以添加固件。

点击添加自定义固件后，填写固件的基本信息以及生产配置等，完成后，点击上传固件

e. 进入上传固件界面，我们需要填写我们固件的版本号，以及选择上传我们的生产固件以及用户区固件，还有升级固件(此处为芯科平台，编译之后生成的文件为.s37 后缀的文件，升级固件选择为文件名称中带有 OTA 的文件后缀为.bin 的文件；泰凌微平台上传 xx_QIO.bin 与 xx_OTA.bin 即可；**此处示例固件已经上架发布，故选项全为灰色无法编辑**)。



f. 当固件配置完成且上传成功后，需要进行发布上架。



Note：开发中的设备，如交付物中的 IO 配置不是调试状态或者发布状态，此界面无模组可选择，发布状态后 IO 配置便不可修改。

3. Demo 创建与固件编译生成

3.1 芯科平台 demo 创建

a. 下载 SDK 并仔细阅读开发者使用涂鸦 SDK 先仔细阅读 Zigbee SDK 说明与 Zigbee SDK demo 说明。也可参考 sdk 目录下 doc/开发手册/tuya_zigbee_sdk.chm 文件，该文件可提供开发者查询 SDK 的各接口的使用，功能非常全面。

b. 复制 app 目录下的 demo 工程，修改用户定义的工程文件夹名称。

3.2 Json 文件修改

a. json 文件主要分为 firmwareInfo、ioconfig、uartconfig。在 firmwareinfo 中 name 为后续编译生成固件的名称；Version 为固件的版本号；model_id 与 manufacture_name 查询 doc/

开发手册/model_id 文档查询相关品类。

```
{
    "fimwareInfo": {
        "name": "sample_switch2",
        "description": "this is a project demo",
        "version": "1.0.0",
        "bv_version": "1.0",
        "ic": "EFR32MG21A020F768",
        "ota_image_type":"0x1602",
        "manufacture_id":"0x1002",
        "model_id":"TS0001",
        "pid": "nPGIPl5D",
        "manufacture_name": "_TZ3000_",
        "module_name":"ZS3L"
    },
    "uartConfig": {
        "uart_enable": "true",
        "uart_num": 2,
        "uart": [
          {
            "uart0_rx_port": "PORT_A",
            "uart0_rx_pin": "PIN_1",
            "uart0_rx_loc": "LOC_0",
            "uart0_tx_port": "PORT_A",
            "uart0_tx_pin": "PIN_0",
            "uart0_tx_loc": "LOC_0",
            "uart0_bandrate": 115200
```

b. Json 文件中的 IoConfig 中若 enable 使能，则在 include 目录下的 config.h 中自动生成相应的 IO 宏定义，不要直接修改 config.h 文件，该文件由脚本自动生成，切记。

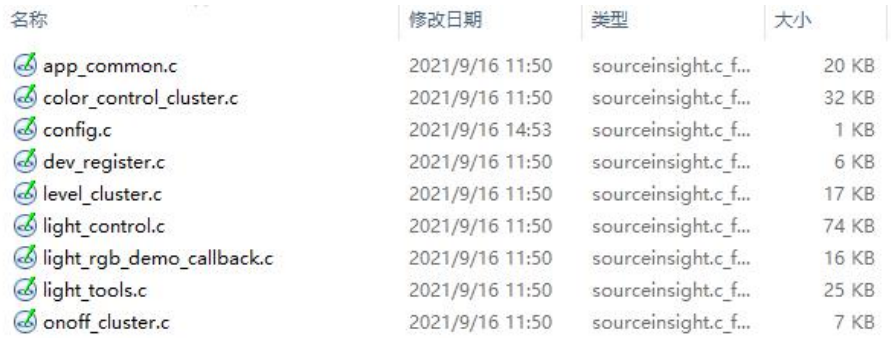
```
  "ioConfig": {
    "led_enable": "true",
    "led_num": 1,
    "key_enable": "true",
    "key_num": 1,
    "led": [
          {
            "led0_port": "PORT_B",
            "led0_pin": "PIN_4",
            "led0_mode": "GPIO_MODE_OUTPUT_PP",
            "led0_out": "GPIO_DOUT_LOW",
            "led0_driver": "GPIO_DOUT_HIGH"
          },
    ],
    "key": [
          {
            "key0_port": "PORT_B",
            "key0_pin": "PIN_5",
            "key0_mode": "GPIO_MODE_INPUT_PULL",
            "key0_out": "GPIO_DOUT_LOW",
            "key0_driver": "GPIO_LEVEL_LOW"
          },
          {
```

```
/* io config! */
/* led config! */
#define LED0_PORT                       PORT_B
#define LED0_PIN                        PIN_4
#define LED0_MODE                       GPIO_MODE_OUTPUT_PP
#define LED0_OUT                        GPIO_DOUT_LOW
#define LED0_DRIVER                     GPIO_DOUT_HIGH


/* io config! */
/* key config! */
#define KEY0_PORT                       PORT_B
#define KEY0_PIN                        PIN_5
#define KEY0_MODE                       GPIO_MODE_INPUT_PULL
#define KEY0_OUT                        GPIO_DOUT_LOW
#define KEY0_DRIVER                     GPIO_LEVEL_LOW
```

3.3 Xxx.callback 文件二次开发

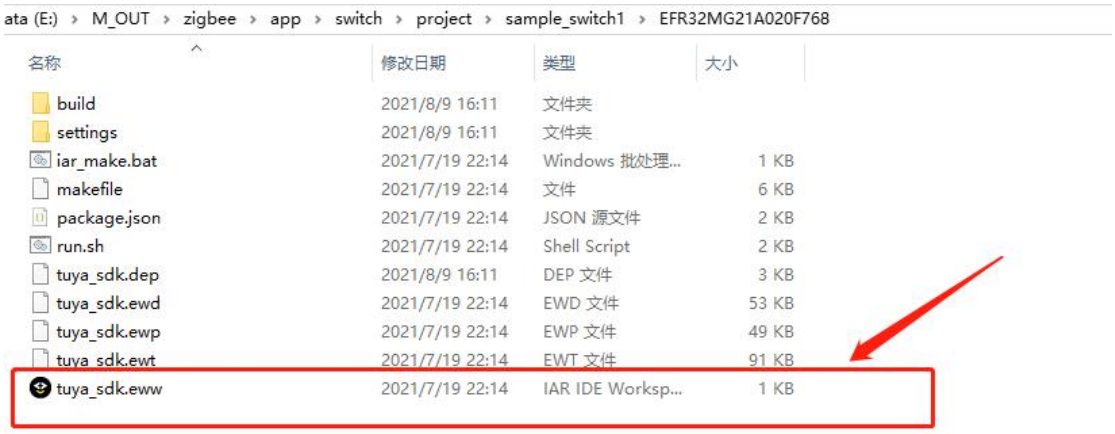
用户在 xxx.callbac.c 文件进行应用二次开发。cluster 相关文件放在对应 cluster.c 中，设备注册放在 dev_register.c 中，公共函数放在 app_common.c 中，工具函数放在 light_tools.c 中。



名称	修改日期	类型	大小
app_common.c	2021/9/16 11:50	sourceinsight.c_f...	20 KB
color_control_cluster.c	2021/9/16 11:50	sourceinsight.c_f...	32 KB
config.c	2021/9/16 14:53	sourceinsight.c_f...	1 KB
dev_register.c	2021/9/16 11:50	sourceinsight.c_f...	6 KB
level_cluster.c	2021/9/16 11:50	sourceinsight.c_f...	17 KB
light_control.c	2021/9/16 11:50	sourceinsight.c_f...	74 KB
light_rgb_demo_callback.c	2021/9/16 11:50	sourceinsight.c_f...	16 KB
light_tools.c	2021/9/16 11:50	sourceinsight.c_f...	25 KB
onoff_cluster.c	2021/9/16 11:50	sourceinsight.c_f...	7 KB

3.4 编译生成固件

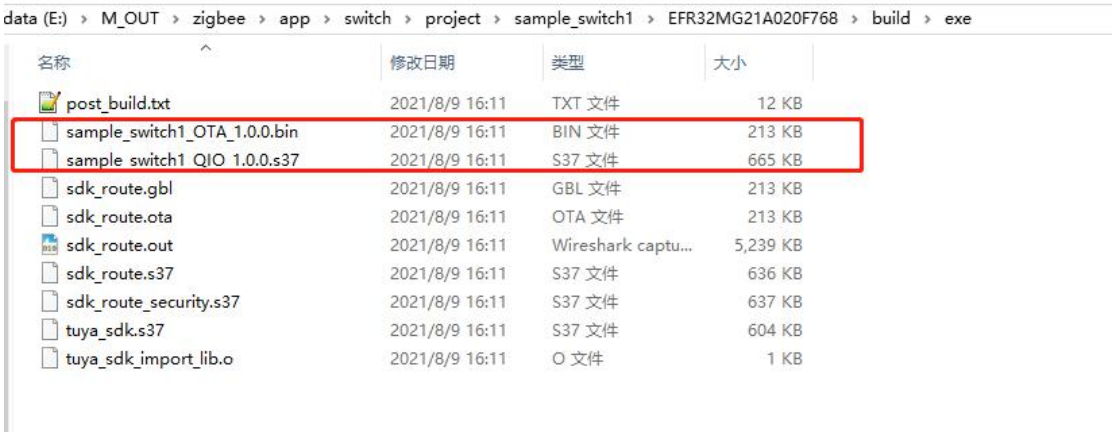
a. 直接点击文件夹中对应芯片的 IAR 工程文件，不要从 IAR 软件中载入，切记！！！



ata (E:) › M_OUT › zigbee › app › switch › project › sample_switch1 › EFR32MG21A020F768

名称	修改日期	类型	大小
build	2021/8/9 16:11	文件夹	
settings	2021/8/9 16:11	文件夹	
iar_make.bat	2021/7/19 22:14	Windows 批处理...	1 KB
makefile	2021/7/19 22:14	文件	6 KB
package.json	2021/7/19 22:14	JSON 源文件	2 KB
run.sh	2021/7/19 22:14	Shell Script	2 KB
tuya_sdk.dep	2021/8/9 16:11	DEP 文件	3 KB
tuya_sdk.ewd	2021/7/19 22:14	EWD 文件	53 KB
tuya_sdk.ewp	2021/7/19 22:14	EWP 文件	49 KB
tuya_sdk.ewt	2021/7/19 22:14	EWT 文件	91 KB
tuya_sdk.eww	2021/7/19 22:14	IAR IDE Worksp...	1 KB

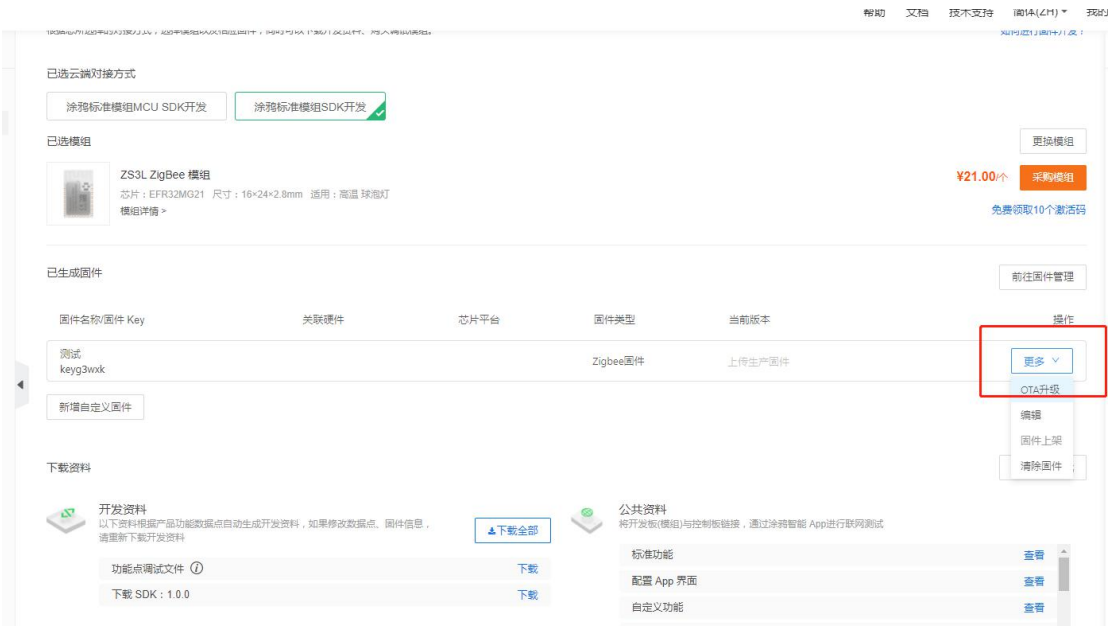
b. 编译完成后再 build 文件夹下会生成对应的.37 生产固件与.bin 的 OTA 固件。


data (E:) › M_OUT › zigbee › app › switch › project › sample_switch1 › EFR32MG21A020F768 › build › exe
名称
修改日期
类型
大小
post_build.txt 2021/8/9 16:11 TXT 文件 12 KB
sample_switch1_OTA_1.0.0.bin 2021/8/9 16:11 BIN 文件 213 KB
sample_switch1_QIO_1.0.0.s37 2021/8/9 16:11 S37 文件 665 KB
sdk_route.gbl 2021/8/9 16:11 GBL 文件 213 KB
sdk_route.ota 2021/8/9 16:11 OTA 文件 213 KB
sdk_route.out 2021/8/9 16:11 Wireshark captu... 5,239 KB
sdk_route.s37 2021/8/9 16:11 S37 文件 636 KB
sdk_route_security.s37 2021/8/9 16:11 S37 文件 637 KB
tuya_sdk.s37 2021/8/9 16:11 S37 文件 604 KB
tuya_sdk_import_lib.o 2021/8/9 16:11 O 文件 1 KB

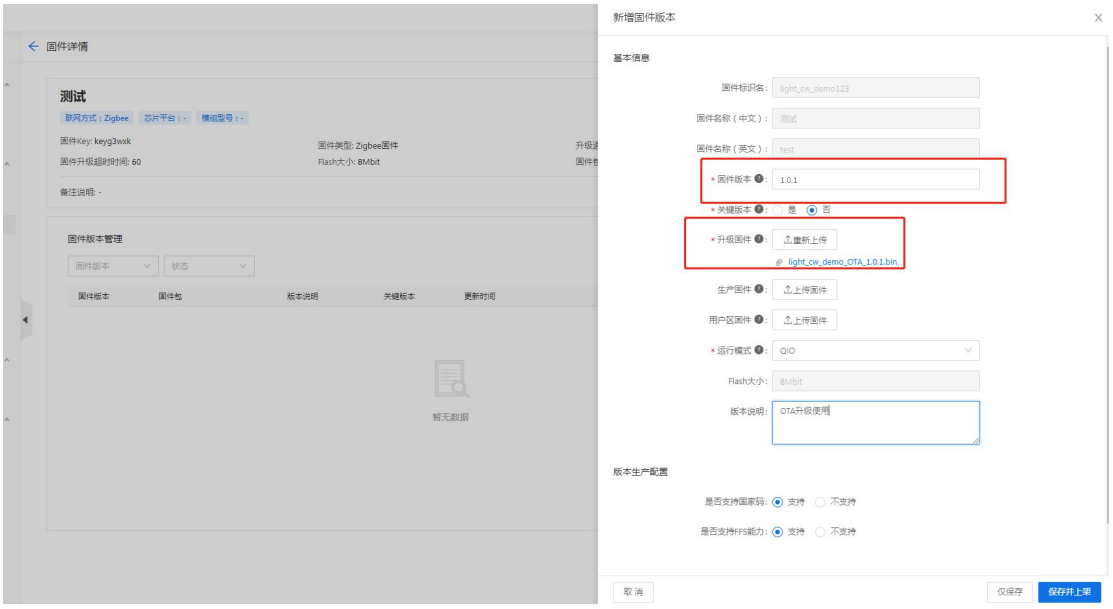
4. 设备 OTA 测试

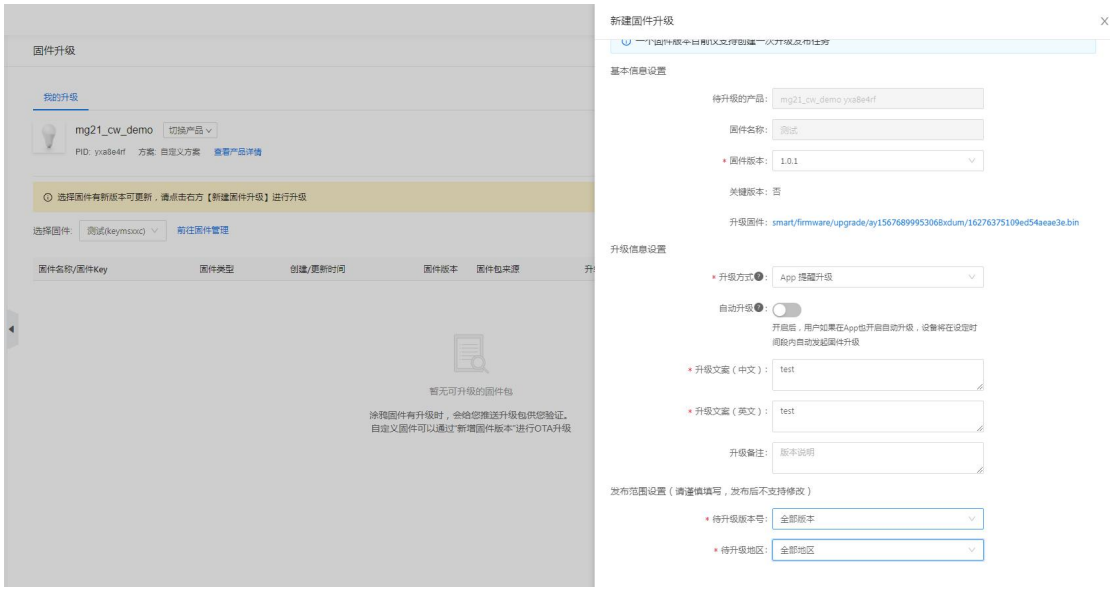
4.1 添加 OTA 升级固件

a. 选择固件更多操作中的 OTA 升级选项。



b. 上传高版本固件。





c.设备配网成功后，在设备属性中获取虚拟 ID，填入 IOT 平台中，app 刷新即可提醒用户升级。

Note：一个网关只能同时升级一个子设备。

Note：MG21 平台芯片 OTA 文件大小为 305K，超出会升级失败。

4.2 验证版本

设备 OTA 完成后，可在此界面点击验证，确认是否均升级完成。

Note：OTA 测试目前无法降级。

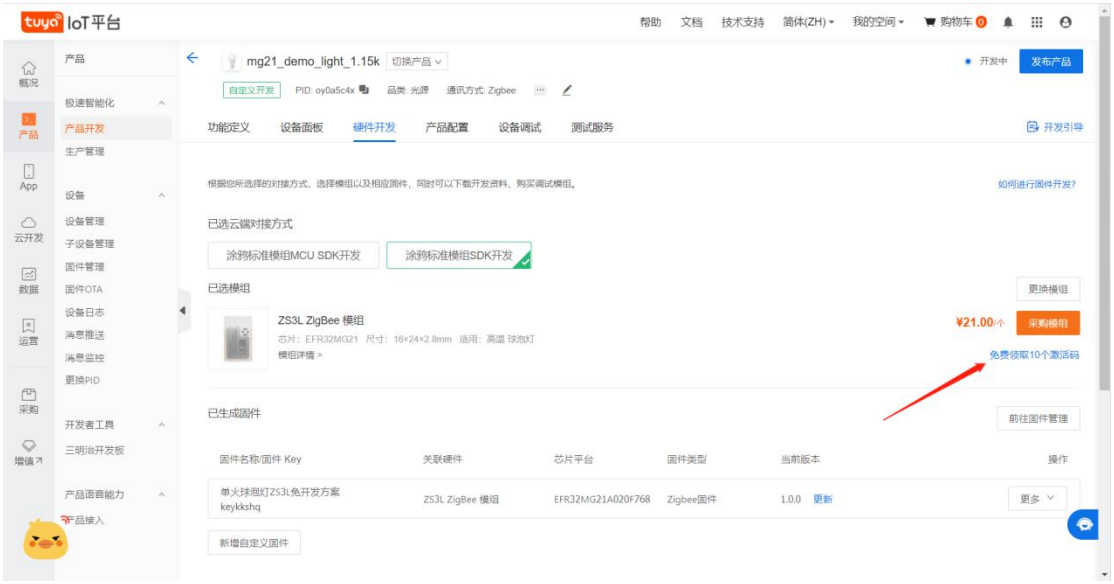
5. 烧录与产测

5.1 烧录授权

a. 外部下单

烧录授权需要授权码，在选择完模块后可以免费申请 10 个授权码，需要在 IOT 平台下单，得到授权码。

步骤：登录 IOT 平台 -> 进入产品开发 ->继续开发 -> 已选模组右侧：免费申请 10 个激活码 -> 确认领取



确认下单后，即可得到授权码。

b. 烧录前准备

软件安装：**云模组烧录授权平台**

账号权限：https://pms.tuya-inc.com:7799/#/ 自行注册

烧录接线：接线除了依据不同 IC 方案接烧录脚，还需接**产测串口**。

c. 线上烧录

在云模组烧录授权平台中选择输入生产凭证 -> 生产凭证，将内部下单得到的生产凭证输入，并选择工位烧录授权

选择正确的 COM 口，点击运行，即可开始烧录

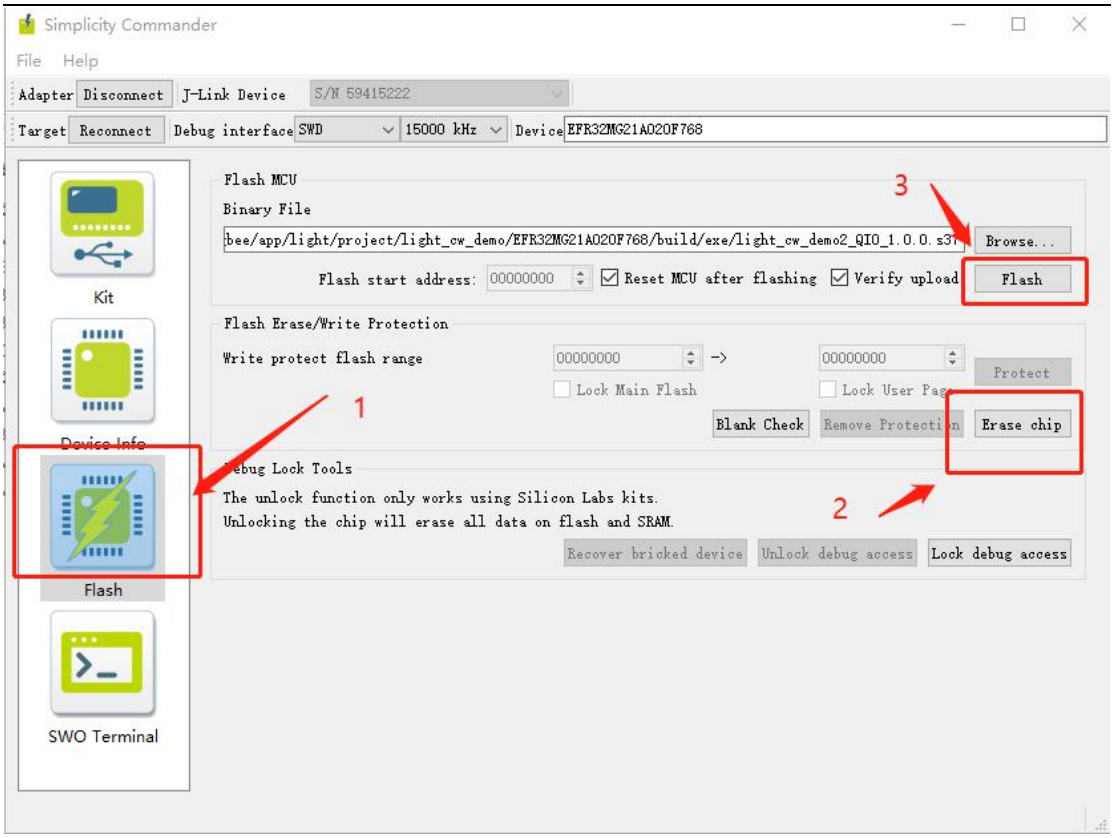
Note：Telink 烧录前需注意 Activate，激活 IC 才可复位

d. 二次检测

烧录完毕后，可选择工位二次检测进行授权二次检测

5.2 无授权烧录

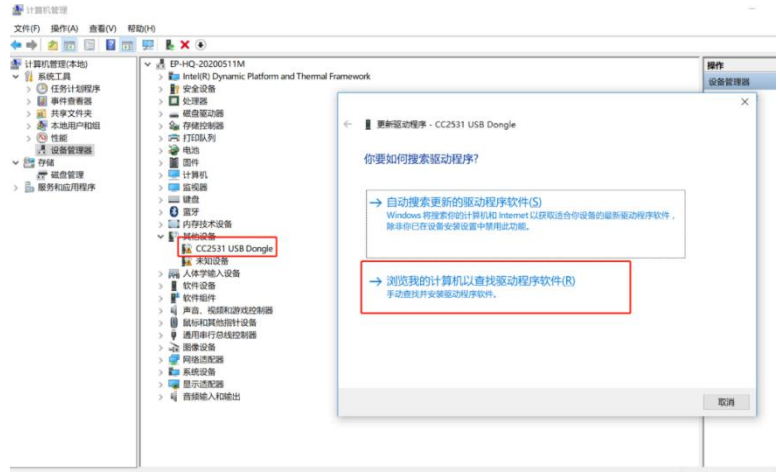
a. 芯科 Command 烧录，通过 J-link 连接开发板，使用芯科平台的 commander 进行固件烧录。注：先擦除再进行烧录。


Simplicity Commander
File Help
Adapter Disconnect J-Link Device S/N 59415222
Target Reconnect Debug interface SWD 15000 kHz Device EFR32MG21A020F768
Kit
Device Info
Flash
SWO Terminal
Flash MCU
Binary File
Browse...
Flash start address: 00000000 Reset MCU after flashing Verify upload
Flash
Flash Erase/Write Protection
Write protect flash range 00000000 -> 00000000
Protect
Lock Main Flash
Blank Check
Remove Protection
Erase chip
Debug Lock Tools
The unlock function only works using Silicon Labs kits.
Unlocking the chip will erase all data on flash and SRAM.
Recover bricked device
Unlock debug access
Lock debug access
1
2
3

6. Zigbee 空中数据抓包及分析

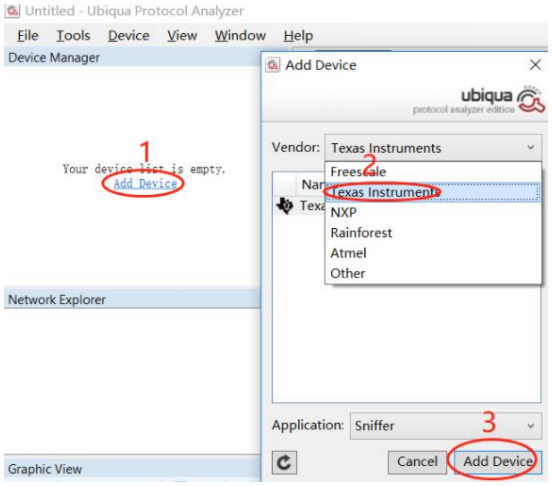
6.1 使用软件和 Dongle

抓包上位机软件使用 Ubiqua，插入 USB Dongle 抓包器(驱动正常安装)。

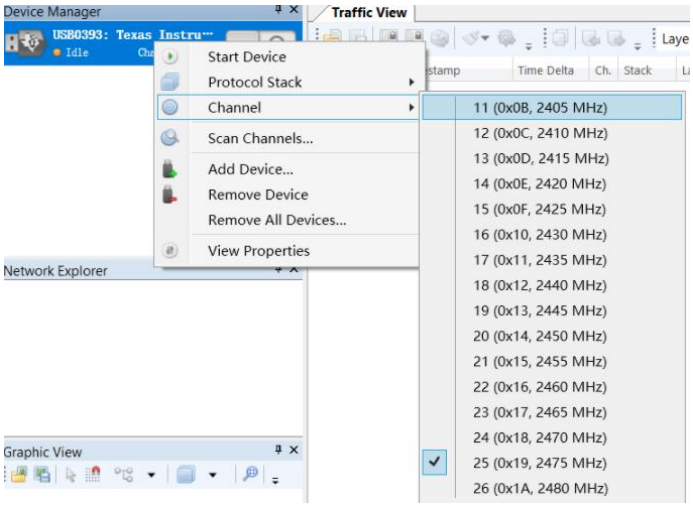


6.2 软件使用

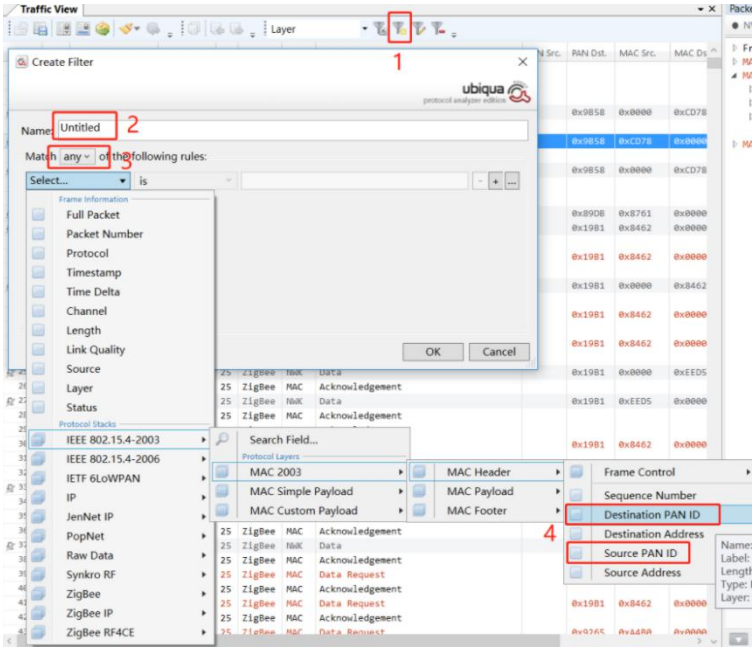
点击 Add_Device，再选择 TI，将会出现抓包器，点击 Add Device 完成设备添加

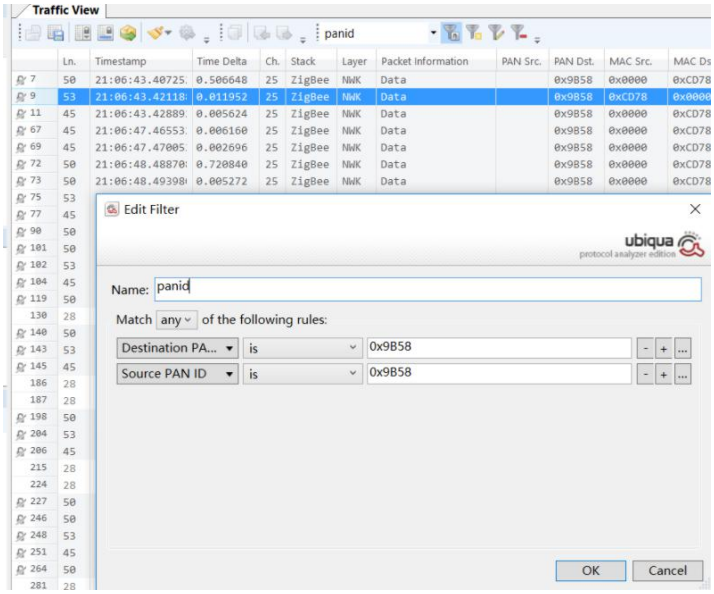


右击设备可以选择信道，选择想要抓取的信道，可以使用 SSH 进入网关内部使用 debugtool 查看 channel 和 panId 以及 NetWorkKey

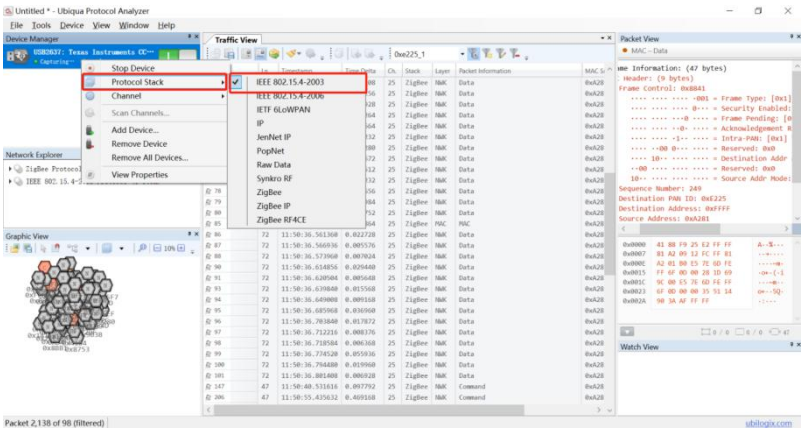


一个信道内仍然有大量 zigbee 在进行通信，可能一部分不是我们所需要的，我们还需要设置过滤器进一步将需要的数据提取出来，这就要依靠 panId 了。在位置 1 可以创建过滤器；在位置 2 可以重命名；位置 3 支持成立逻辑，源 panId 和目的 panId 其中一个成立就是过滤数据，因此这里选择 any(any 为或条件 all 为与条件)。





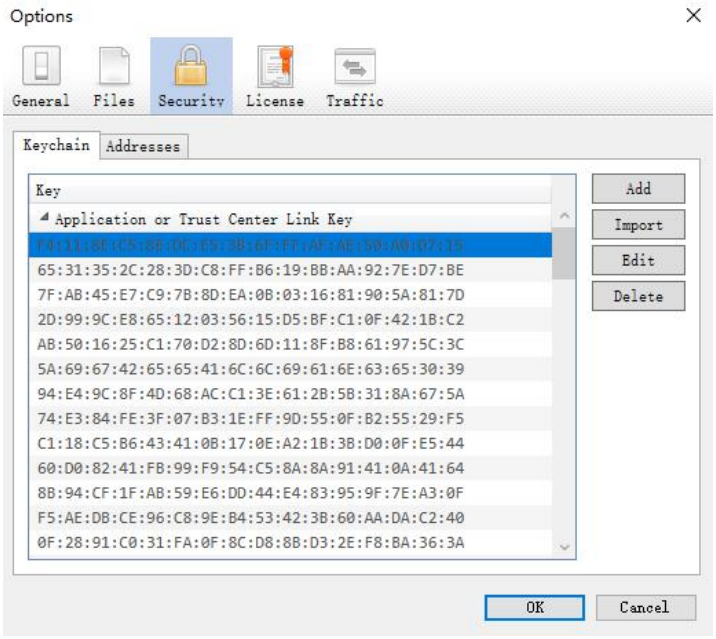
这里需要注意，右键自己设备，查看 Protocol Stack ，查看协议栈，是否与过滤器中选择的协议栈相同，若不同的话，需要更改为相同协议栈。默认选择 Zigbee。

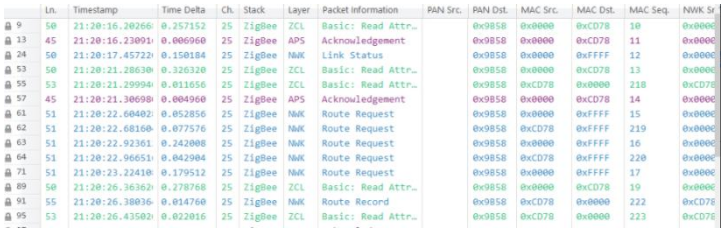


如果 zigbee 通信设备过多，所以如果想只看自己指定的设备，需要多重组合几个条件。这里给出一个各参数解释较清楚的链接，了解各参数作用后，可根据自己的实际需求来组合过滤条件。

6.3 添加全局密钥

application or trust center link key 5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39 添加完全局 link key 后，需要在网关所在信道抓取一个节点入网的整个过程，从中获取 networkkey，即可发现数据是非加密状态。





	Ln.	Timestamp	Time Delta	Ch.	Stack	Layer	Packet Information	PAN Src.	PAN Dst.	MAC Src.	MAC Dst.	MAC Seq.	NWK Sr
9	50	21:20:16.20266	0.257152	25	ZigBee	ZCL	Basic: Read Attr…		0x9858	0x0000	0xCD78	10	0x0000
13	45	21:20:16.23091	0.006960	25	ZigBee	APS	Acknowledgement		0x9858	0x0000	0xCD78	11	0x0000
24	50	21:20:17.45722	0.150184	25	ZigBee	NWK	Link Status		0x9858	0x0000	0xFFFF	12	0x0000
53	50	21:20:21.28630	0.326320	25	ZigBee	ZCL	Basic: Read Attr…		0x9858	0x0000	0xCD78	13	0x0000
55	53	21:20:21.29994	0.011656	25	ZigBee	ZCL	Basic: Read Attr…		0x9858	0xCD78	0x0000	218	0xCD78
57	45	21:20:21.30698	0.004960	25	ZigBee	APS	Acknowledgement		0x9858	0x0000	0xCD78	14	0x0000
61	51	21:20:22.60402	0.052856	25	ZigBee	NWK	Route Request		0x9858	0x0000	0xFFFF	15	0x0000
62	51	21:20:22.68160	0.077576	25	ZigBee	NWK	Route Request		0x9858	0xCD78	0xFFFF	219	0x0000
63	51	21:20:22.92361	0.242008	25	ZigBee	NWK	Route Request		0x9858	0x0000	0xFFFF	16	0x0000
64	51	21:20:22.96651	0.042904	25	ZigBee	NWK	Route Request		0x9858	0xCD78	0xFFFF	220	0x0000
71	51	21:20:23.22410	0.179512	25	ZigBee	NWK	Route Request		0x9858	0x0000	0xFFFF	17	0x0000
89	50	21:20:26.36362	0.278768	25	ZigBee	ZCL	Basic: Read Attr…		0x9858	0x0000	0xCD78	19	0x0000
91	55	21:20:26.38036	0.014760	25	ZigBee	NWK	Route Record		0x9858	0xCD78	0x0000	222	0xCD78
95	53	21:20:26.43502	0.022016	25	ZigBee	ZCL	Basic: Read Attr…		0x9858	0xCD78	0x0000	223	0xCD78

关于 zigbee 层次分析请参考：https://zhuanlan.zhihu.com/p/296282083

7. 现有 DEMO 新增 DP 流程

7.1 IOT 平台创建相应品类来参考 DP 点

OT 平台创建产品参考 2.2 创建产品，这里以 zigbee 调光产品为例。

根据要在 DEMO 中添加的功能找到相应的 DP ID。

7.2 参考 DP 对应关系

dpid	dp名称	操作方式	Cluster	Attr Id/cmd	Type	具体格式	网关支持版本（有线网关）	备注
1、7	开关	下行cmd	0x0006	0x00~02	-		1.3.8	
		上行report	0x0006	0x0000	bool		1.3.8	
2、8	亮度	下行cmd	0x0008	0xf0	-	2字节 0~1000	1.3.8	
		上行report	0x0008	0xf000	uint16	0~1000	1.3.8	
3、9	最小亮度	读写	0x0008	0xfc03	uint16	0~1000	1.3.8	
5、11	最大亮度	读写	0x0008	0xfc04	uint16	0~1000	1.3.8	
4、10	灯源类型	读写	0x0008	0xfc02	uint8	dp透传	1.3.8	
6、12	倒计时	下行cmd	0x0006	0xf0	uint32	4字节	1.3.8	
		上行report	0x0006	0xf000	uint32		1.3.8	
13	模式	下行cmd	0x0300	0xf0	-	dp透传	1.3.8	
		上行report	0x0300	0xf000	uint8	dp透传	1.3.8	
14	上电状态	读写	0x0006	0x8002	enum8	dp透传	1.3.8	

在相应品类的文档中找到 dpid 对应的 cluster id 和对应的 cluster id 下的 attribute id 或 cmd id。下行的方式有两种：CMD 下发和写属性，开发者需要关注相应 dp 点的操作方式；上行的方式统一使用属性上报。

下行的 CMD 的回调接口如下所示：

```
/**
 * @description: device receive message callback
 * @param {*dev_msg} received message information
 * @return: ZCL_CMD_RET_T
 */
ZCL_CMD_RET_T dev_msg_recv_callback(dev_msg_t *dev_msg)···

/**
```

下行写属性的回调接口如下所示：

```
/**
 * @description: write attribute callback function
 * @param {endpoint} the endpoint of the write attribute
 * @param {cluster} the cluster id of the write attribute
 * @param {attr_id} the attribute id  of the write attribute
 * @return: none
 */
void dev_msg_write_attr_callback(uint8_t endpoint, CLUSTER_ID_T cluster, uint16_t attr_id)
```

7.3 打开的 demo 工程的 device_register.h 文件

```
    ON_OFF_PRIVATE_ATTR_LIST
};
const attr_t g_level_attr_list[] = {
    LEVEL_CONTROL_ATTR_LIST
    LEVEL_PRIVATE_ATTR_LIST
};

const attr_t g_color_attr_list[] = {
    COLOR_CONTROL_ATTR_LIST
    COLOR_PRIVATE_ATTR_LIST
};
const attr_t g_private_attr_list[] = {
    PRIVATE_ATTR_LIST
};
const cluster_t g_server_cluster_id[] = {
    DEF_CLUSTER_GROUPS_CLUSTER_ID(g_group_attr_list)
    DEF_CLUSTER_SCENES_CLUSTER_ID(g_scene_attr_list)
    DEF_CLUSTER_ON_OFF_CLUSTER_ID(g_onoff_attr_list)
    DEF_CLUSTER_LEVEL_CONTROL_CLUSTER_ID(g_level_attr_list)
    DEF_CLUSTER_COLOR_CONTROL_CLUSTER_ID(g_color_attr_list)
    DEF_CLUSTER_PRIVATE_CLUSTER_ID(g_private_attr_list)
};
```

在相应的 cluster ID 下添加相应的 attribute id。宏定义写入格式参考 attribute 结构体进行添加，参考例子如下：

{ 0x0000(attr_id), ATTR_BOOLEAN_ATTRIBUTE_TYPE(attr_type), 1(字节数),

(ATTR_MASK_TOKEN_FAST)(attr_mask），(uint8_t*)0x00 (default value)},

Attr_mask 常用类型：ATTR_MASK_WRITABLE (0x01)，当 DP 对应关系中涉及到读写属性时，需要添加可写标志；ATTR_MASK_TOKEN_FAST (0x80)，当属性选择存储功能时，可添加 TOKEN 标志；既需要写属性又需要存储时可以通过或（|）操作进行打标。

7.4 DP 透传使用说明

若开发者自定义 dp（100 之后），想通过透传的方式接入设备，需要在开发中添加以下步骤：

1. 设备注册函数属性中添加 0xEF00 私有属性

```
const attr_t g_private_attr_list[] = {
    PRIVATE_ATTR_LIST
};

const attr_t g_power_attr_list[] = {
    POWER_ATTR_LIST
};

const attr_t g_light_attr_list[] = DEV_CONGIG;

const cluster_t g_server_cluster_id[] =
{
    DEF_CLUSTER_GROUPS_CLUSTER_ID(g_group_attr_list)
    DEF_CLUSTER_SCENES_CLUSTER_ID(g_scene_attr_list)
    DEF_CLUSTER_ON_OFF_CLUSTER_ID(g_light_attr_list)
    DEF_CLUSTER_PRIVATE_CLUSTER_ID(g_private_attr_list)
    DEF_CLUSTER_POWER_CLUSTER_ID(g_power_attr_list)
};
```

2. Package.json 文件中的 manufacture_name 修改能力值：“_TZ3210_”。

```
{
        "fimwareInfo": {
                "name": "oem_si32_zg_watering_bn_1",
                "description": "this is a project demo",
                "version": "1.0.5",
                "ic": "EFR32MG13P732F512",
                "ota_image_type":"0x1602",
                "manufacture_id":"0x1002",
                "model_id
                "pid":
                "manufacture_name": "_TZ3210_",
                "module_name":"TYZS3"
        },
        "uartConfig": {
```

3. 面板下发内容在下图中的接口中回调，注：dp 透传业务需要面板进行配合，透传上报格式如下图所示，

```c
/**
 * @description: device receive message callback
 * @param {*dev_msg} received message information
 * @return: ZCL_CMD_RET_T
 */
ZCL_CMD_RET_T dev_msg_recv_callback(dev_msg_t *dev_msg)
{
    ZCL_CMD_RET_T result = ZCL_CMD_RET_SUCCESS;
    switch (dev_msg->cluster)
    {
        case CLUSTER_PRIVATE_TUYA_CLUSTER_ID:   //私有数据处理
        {
            uint8_t len = dev_msg->data.bare_data.len;
            uint8_t *data = dev_msg->data.bare_data.data;

            switch(dev_msg->data.bare_data.commandId)
            {
                case TY_MODULE_DATA_REQUEST_CMD:
                {
                    dev_private_msg_handle(&dev_msg->data.bare_data);
                    dev_report_private_msg(dev_msg->endpoint,len,data,TY_MODULE_DATA_RESPONE_CMD,QOS_1);
                    change_poll_interval(QUICKLY_POLL_PER);/*接收到app下发的数据后10s内加快数据获取频率*/
                    dev_timer_start_with_callback(POLL_FREQUENECY_CHANGE_USE, COMMUNICATE_POLL_QUICKLY_TIME, dev_evt_callback);
                    break;
                }
                default:
                {
                    break;
                }
            }
            break;
        }
        //标准数据处理
        case CLUSTER_ON_OFF_CLUSTER_ID:
        {
```

1

2

```c
/*********************************************************************************
 * Function:
 * Description:    serve  to client special onoff
 * Input:
 * Outpot:
 * Return:
 * History:
 *********************************************************************************/
void dev_report_private_msg(uint8_t ep, uint8_t len,uint8_t *data ,uint8_t command_id,SEND_QOS_T qos)
{
    // NET_EVT_T t_nwk_status = nwk_state_get();

    // if((NET_POWER_ON_ONLINE == t_nwk_status) || (NET_JOIN_OK == t_nwk_status) || (NET_REJOIN_OK == t_nwk_status))
    {
        dev_send_data_t send_data;
        memset(&send_data, 0, sizeof(dev_send_data_t));
        send_data.zcl_id = 0;//++zcl_seq;
        send_data.qos = qos;
        send_data.commmand_type = ZCL_COMMAND_CLUSTER_SPEC_CMD;
        send_data.direction = ZCL_DATA_DIRECTION_SERVER_TO_CLIENT;
        send_data.command_id = command_id;
        send_data.addr.mode = SEND_MODE_GW;
        send_data.addr.type.gw.cluster_id = CLUSTER_PRIVATE_TUYA_CLUSTER_ID;
        send_data.addr.type.gw.src_ep = ep;
        send_data.delay_time = 0;
        send_data.random_time = 0;
        send_data.data.private.len = len;
        memcpy((send_data.data.private.data),data,len);
        dev_zigbee_send_data(&send_data, NULL, 5000);
    }
}
```

3 0XEF00

```c

/**
 * @description: dp透传主动上报电量百分比
 * the stack will call this function to notify the user;
 * @param
 * @return: none
 */

void report_private_week_delay(uint8_t endpoint,DEV_WEEK_DELAY_ST type)
{
    const uint8_t t_msg_len = 7;/*数据长度*/
    uint8_t *data = NULL;

    if(0xFFFF == g_private_report_str.sequence_num)
    {
        g_private_report_str.sequence_num = 0;
    }
    else
    {
        g_private_report_str.sequence_num ++;
    }

    data = malloc(t_msg_len*sizeof(uint8_t));
    if(data)
    {
        memset(data,0x00,t_msg_len);
        data[0] = (uint8_t)(g_private_report_str.sequence_num>>8);
        data[1] = (uint8_t)(g_private_report_str.sequence_num);
        data[2] = WEEK_DELAY_DP;
        data[3] = TYPE_ENUM;
        data[4] = 0;
        data[5] = 0x01;
        data[6] = type;

        dev_report_private_msg(endpoint,t_msg_len,data,TY_MODULE_DATA_REPORT_CMD,QOS_1);
        free(data);
    }

}
```

主动上报的 command id 变一下，date 数据自己定的

4. 透传协议相关 CMD 如下图所示：

		报。mcu将收到cmd=0x07
TY_MODULE_DATA_REQUEST	0x04	Gw→Zigbee 网关向模组发送DP数据
TY_MODULE_DATA_RESPONE	0x05	Zigbee→Gw 模组应答网关的DP数据
TY_MODULE_DATA_REPORT	0x06	Zigbee→Gw 模组向网关主动上报DP数据

8. 特殊 API 说明

8.1 dev_zigbee_recovery_network_set

在非低功耗设备中，若想增加自恢复的功能可在 power_on_init 中添加 dev_zigbee_recovery_network_set(TRUE)接口函数。即实现设备组网失败后，设备会自动恢复至上一个网络中。