Tuya IOT 介绍

作者：小 Young

日期：2020 年 12 月 24 日

目录

一. 介绍

中文：该项目基于 Tuya App SDK 进行开发，它使您能够快速开发连接和控制许多设备智能场景的品牌应用程序。

想了解更多信息，请访问[涂鸦开发者网站]

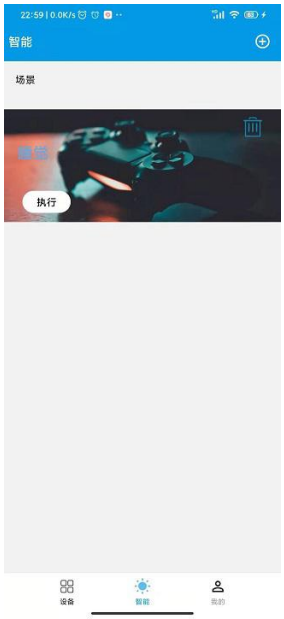
(https://developer.tuya.com/en/docs/iot/app-development/sdk-development/app-sdk-instruction?id=K9kjstc7t376p

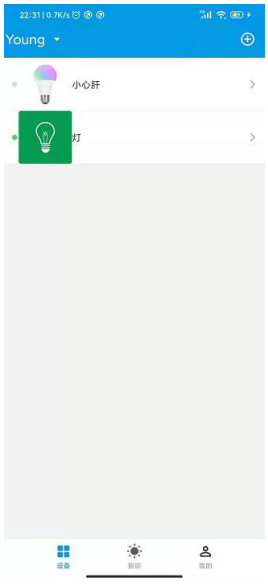
English: This project is developed using Tuya App SDK, which enables you to quickly develop branded apps connecting and controlling smart scenarios of many devices. For more information, please check Tuya Developer Website

其他：主要美化 UI 布局，以及增加了删除场景、定时触发等功能。 此外，还引入了小爱同学语音控制功能。

二. 基本功能

2.1 UI 美化

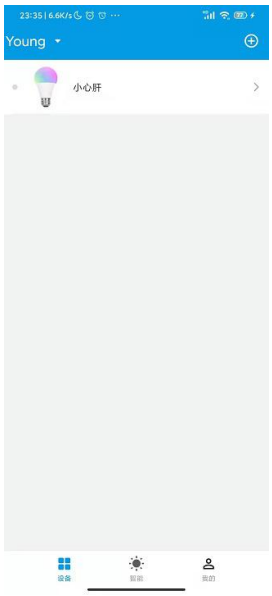




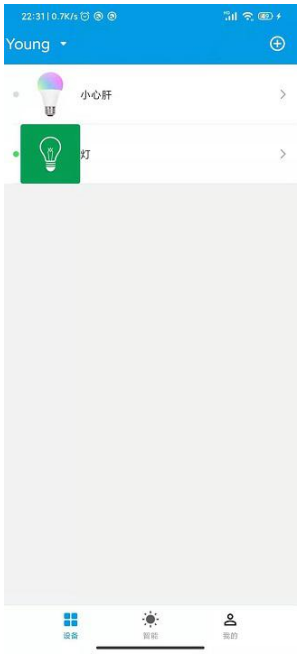
2.2 配网功能

①在设备页面（确保设备处于配网状态），点击右上角的**加号**进入选择设备类型界面。

②在设备类型界面选择 **WIFI 设备**进入到添加设备界面。

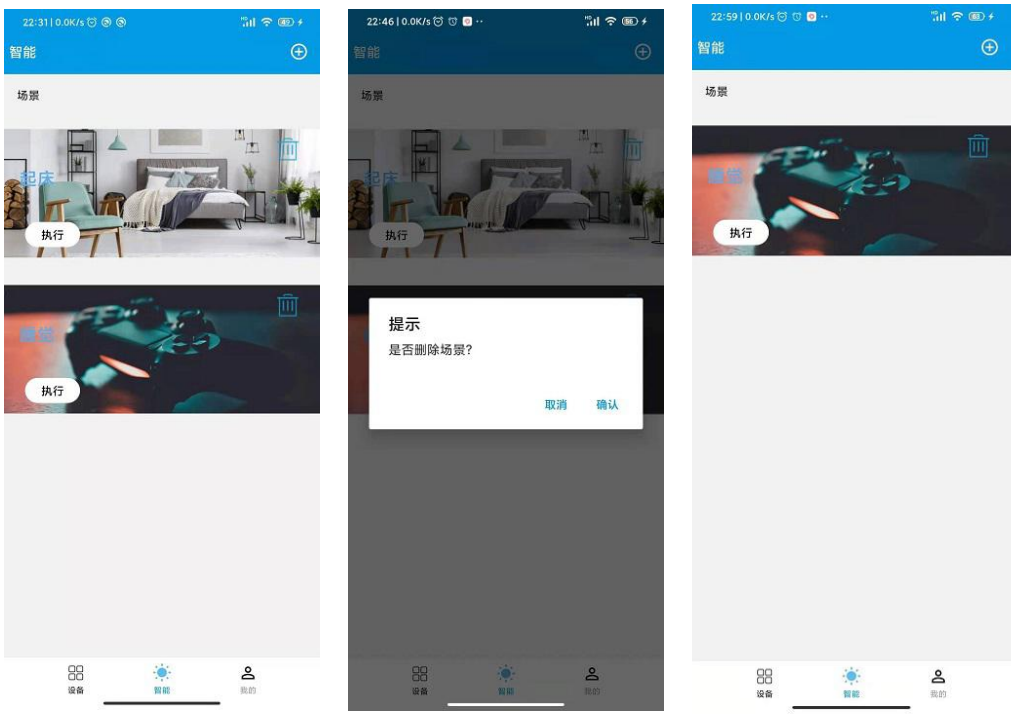


③选择“指示灯快速闪烁中”按钮进入到配网界面。
④在配网界面设置你手机连接的 WIFI 密码，点击下一步即可配网。
⑤配网成功后，回到设备界面下拉刷新即可看到刚刚配网的设备。



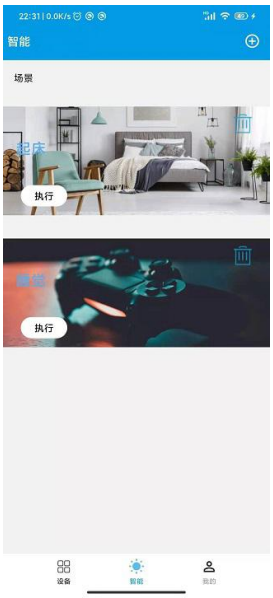
2.3 删除场景功能

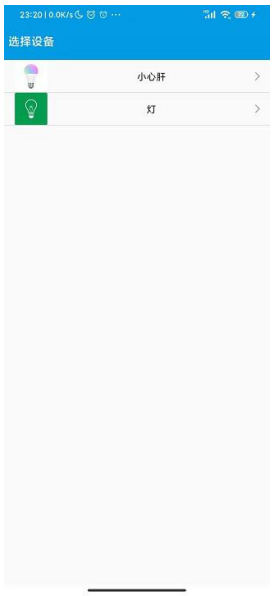
①**智能**页面点击背景图片右上角的**删除图标**（垃圾箱），会弹出一个对话框，用于用户确认是否删除场景。
②点击确认后即可删除场景。
③将页面往下拉刷新一下即可发现之前的场景不见了。



2.4 定时触发任务

①在智能页面点击最右上角的加号会弹出菜单。
②点击添加场景进入到添加场景页面。
③点击添加任务，其次选择设备。





④选择功能，其次选择操作值。
⑤点击下一步即可回到添加场景页面，这个目的是为了获取设备 ID,及 dp 点。
⑥ 回到添加场景界面后，选择时间会弹出时间选择对话框，选择好时间后点击设置。
⑦最后点击设定时间按钮即可设置完成。 APP 通过定时接口设置好定时器信息后，硬件模块会自动根据定时要求进行预订的操作。

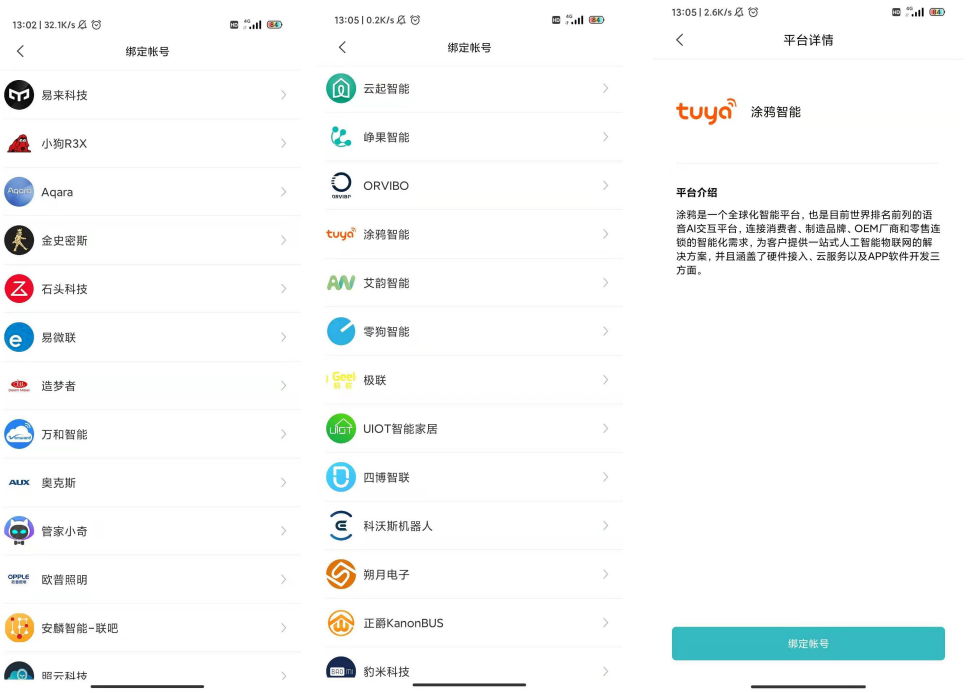
2.5 小爱同学语音控制

我收到的灯包装上写着支持语音控制，所以我寻思着，既然支持语音控制，涂鸦这么大的公司应该接入了小米小爱同学吧，一试不知道试了吓一跳，哇！真的可以哎~敲级开森

①首先打开米家 app(小米手机自带)，点击底部导航栏”我的“，然后将页面往下滑，找到其他平台设备，点击进去。

②点击底部的”点击添加“，进入到绑定账号页面， 进入到绑定账号页面后往下滑动，找到涂鸦智能。

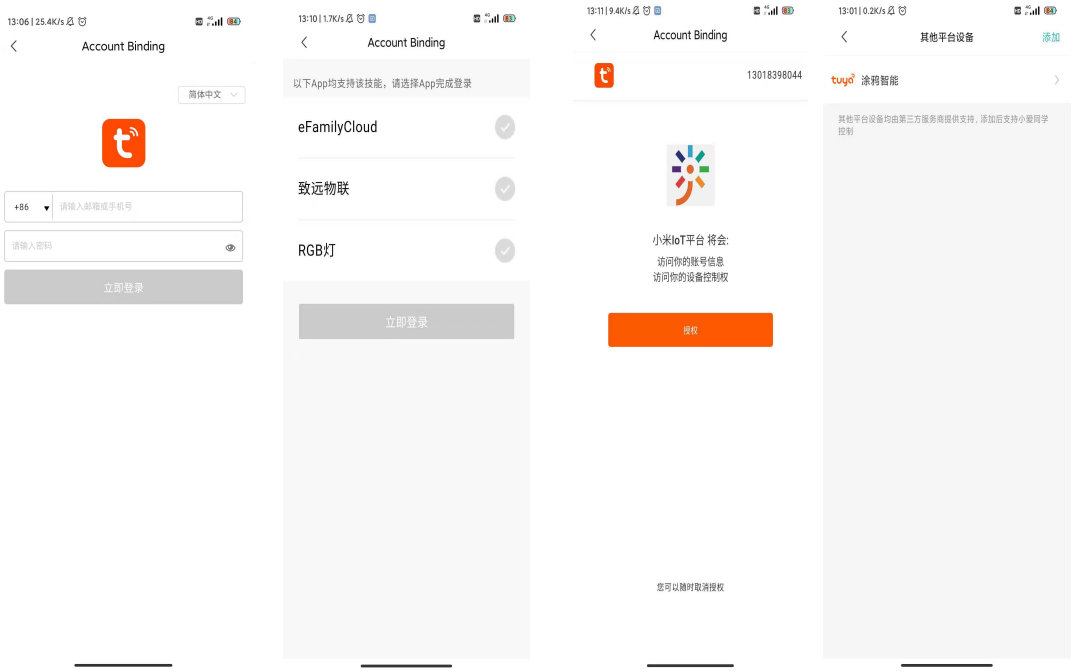
③点击后进入到平台详情页面，点击底部的“账号绑定”进入到账号绑定页面。



④在账号绑定页面输入你登陆涂鸦云平台的账号及密码， 然后点击“立即登录”，登录成功后勾选你的 APP，然后点击“立即登录”。

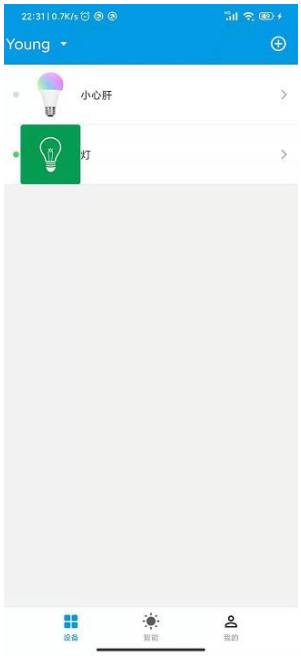
⑤登录成功后会进入到授权界面，点击“授权”即可完成设备绑定。

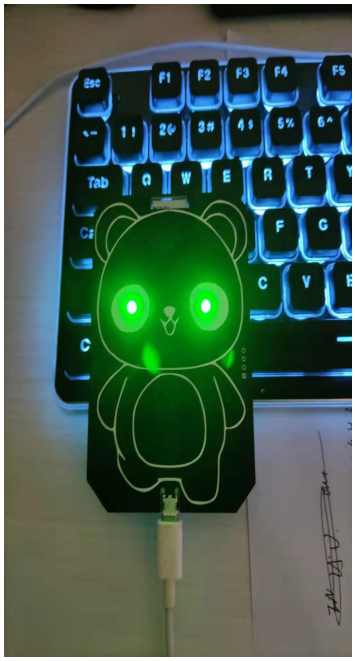
注意：授权完成后会默认进入到显示设备列表的页面，如果是空的，点击页面底部的同步设备即可，如果你想删除，只需要点击底部的接触绑定即可。

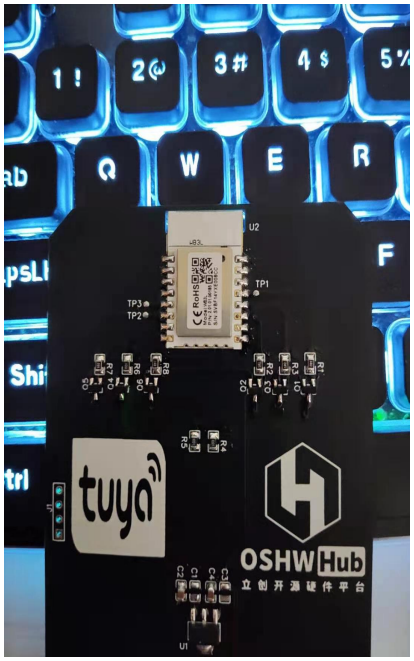


2.6. 涂鸦 WIFI 模块的使用

目的：验证 SDK 是否支持自家所有的模组。 这里用了一块在涂鸦平台上买的 WIFI 模组。首先还是给模块上电后配网，步骤同 2.2 配网功能，配网成功后回到“设备”页面，发现在设备列表里已经添加设备了。 结论：说明 SDK 是支持自己其他模组的。







操作和灯的操作都是一样的没什么特别的。

三.源码实现

3.1 最好奇的部分

拿到源码后肯定好奇的一部分就是手机怎么实现登录后跳转到灯的控制界面的，这实现的背后肯定有他的逻辑。

①启动运行 TuyaSmartApp，TuyaSmartApp 继承自 MultiDexApplication，
在 TuyaSmartApp 中主要对 SDK 进行初始化,然后对登录进行监听。

②App 首先会进入 SplashActivity，在 SplashActivity 中判断是否已经登录，如果登录则进入 HomeActivity，如果之前未登录则进入登陆页面 LoginActivity ，LoginActivity 继承 BaseActivity

③进入 LoginActivity 后首先对布局进行初始化。

④点击登录按钮后通过 LoginPresenter 类的 login 方法传递登录信息，在 login 方法中调用 TuyaHomeSdk.getUserInstance().loginWithPhonePassword 方法，同时注册 mLoginCallback 回调，在回调中向子线程传递成功信息，在线程处理方法中调用 ActivityUtils.gotoHomeActivity 方法进入到主页面 HomeActivity。

⑤进入 HomeActivity 后初始化底部导航栏等，其中 initTab() 方法主要初始化 容器，容器里放的就是“我的设备“”页面，“智能”页面，“个人中心”页面。

⑥“我的设备”页面相当于容器中的一个被管理器管理的碎片，也就是设备列表碎片（DeviceListFragment），在 DeviceListFragment 中首先对页面布局进行初始化。initSwipeRefreshLayout（）方法主要用于刷新设备列表，首次刷新可能啥都没有，需要添加设备后才能刷新到设备，刷新的过程主要向服务器获取设备列表，也就是说你有硬件设备配网成功后你的账号下才有设备，获取到设备列表后向 listView 中添加设备，initAdapter()方法主要对设备列表进行监听，主要监听是点击还是长按列表项，点击或长按了那一项，长按弹出对话框，提示是否移除设备，短按时首先会判断设备是否在线，不在线是会弹出对话框提示“设备已离线”。如果设备在线的话，还会判断是否是“标准产品”（还有其他类型的），这里只说明“标准产品”，在标准产品里通过 TuyaHomeSdk.getDataInstance().getStandardProductConfig(devBean.getProductId()).category 方法获取产品的品类值，这里是灯具（dj），然后获取灯的类型，条件满足后就跳转 LampActivity 界面，在 LampActivity 中就可以看到一些操作方法了。

3.2 删除场景功能实现

在 SceneFragment 中初始化 initAdapter 时会监听 OnExecuteListener 接口的 onExecute 方法和 onDeleteScene 方法。

```
public interface OnExecuteListener {
    void onExecute(SceneBean bean);
    void onDeleteScene(SceneBean bean);
}
```

```
private void initAdapter() {
    mSceneAdapter = new SmartAdapter(getActivity(), SMART_TYPE_SCENE); //场景
    mAutoAdapter = new SmartAdapter(getActivity(), SMART_TYPE_AUTOMATION);//智能化

    mRcv_scene_list.setAdapter(mSceneAdapter);
    mRcv_auto_list.setAdapter(mAutoAdapter);

    mRcv_scene_list.setLayoutManager(new LinearLayoutManager(getActivity()));

    mRcv_auto_list.setLayoutManager(new LinearLayoutManager(getActivity()));

    mSceneAdapter.setOnExecuteListener(new SmartAdapter.OnExecuteListener() {
        @Override
        public void onExecute(SceneBean bean) {
            mPresenter.execute(bean);
        }

        @Override
        public void onDeleteScene(SceneBean bean) {
            mPresenter.deleteScene(bean);
        }

    });
```

点击背景页面上的删除图标后，会触发 onDeleteScene 方法，从而触发 deleteScene 方法，在 deleteScene 方法中会调用 TuyaHomeSdk.newSceneInstance(bean.getId()).deleteScene 执行删除场景操作。

```
public void onBindViewHolder(@NonNull SceneViewHolder viewHolder, int i) {
    final SceneBean sceneBean = datas.get(i);

    Picasso.with(mContext).load(Uri.parse(sceneBean.getBackground())).into(viewHolder.scene_bg);

    viewHolder.tv_title.setText(sceneBean.getName());

    if(mType == SMART_TYPE_SCENE){
        /**操作开关*/
        viewHolder.tv_excute.setVisibility(View.VISIBLE);
        viewHolder.tv_excute.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(null != mExecuteListener) mExecuteListener.onExecute(sceneBean);
            }
        });
        viewHolder.swtich.setVisibility(View.GONE);

        /**删除场景*/
        viewHolder.mImageView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                if(null != mExecuteListener) mExecuteListener.onDeleteScene(sceneBean);
            }
        });

    } else {
```

```
/**删除场景*/
public  void  deleteScene(SceneBean bean)
{
    DialogUtil.simpleConfirmDialog(mActivity, "Delete scene", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            if (which == DialogInterface.BUTTON_POSITIVE) {
                TuyaHomeSdk.newSceneInstance(bean.getId()).deleteScene(new IResultCallback() {
                    @Override
                    public void onSuccess() { ToastUtil.shortToast(mActivity,  tips: "删除成功！"); }
                    @Override
                    public void onError(String errorCode, String errorMessage) {
                    }
                });
        }
        }
    });
}
```

3.3 定时功能实现

来自涂鸦文档：

一个分组可以有多个定时器。每个定时属于或不属于一个分组，分组目前仅用于展示。例如一个开关可能有多个 dp 点，可以根据每个 dp 点设置一个定时分组，每个分组可以添加多个定时器，用于控制这个 dp 点各个时段的开启或关闭。

dp 点(功能点)：对产品功能的抽象表示，是具体智能设备功能的抽象，用于描述产品能及其参数

功能点 ID：功能点的编码。设备与云端的功能数据通过功能点 ID 进行传输。

功能点名称：自定义的功能名称。

标识名：功能点 Code 值，用于 App 显示功能名称的多语言管理。支持字母、数字和下划线，以字母开头。

搞清楚概念后开始实现。
目的：添加多个 dp 点定时任务。
老接口 API:

```
void addTimerWithTask(String taskName, String loops, String devId, String dpId, String time, final IResultStatusCallback callback);
```

参数说明

参数	说明
taskName	定时分组名称
loops	一周7天定时执行次数，"0000000"，每一位 0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六 。"0000000" 表示只执行一次，"1111111" 表示每天执行。
devId	设备id或群组id
dpId	dp点名称
time	下发执行的时间，例如"14:00",目前只支持24小时制
callback	回调，返回成功或失败的结果，不能为 null

但是该 API 仅支持添加单个 dp 点，所以实现也比较简单，dp 点为 1 指的是开关功能（其他产品可能不一样），到达 7:00 后任务就会触发。

```
TuyaHomeSdk.getTimerManagerInstance().addTimerWithTask( taskName: "task02", OperatorValuePresenter.device_id, devId: "1111111",
        dpId: "1",  time: "07:00", new IResultStatusCallback() {
            @Override
            public void onSuccess() {
                Toast.makeText(mAc,  text: "添加定时任务成功", Toast.LENGTH_LONG).show();
            }

            @Override
            public void onError(String errorCode, String errorMsg) {
                Toast.makeText(mAc,  text: "添加定时任务失败 " + errorMsg, Toast.LENGTH_LONG).show();
            }
        });
```

老接口还有一个方法：

增加一个定时器

```
void addTimerWithTask(String taskName, String devId, String loops, Map<String, Object> dps, String time, final IResultStatusCallback
```

参数说明

参数	说明
taskName	定时分组名称
devId	设备 id 或群组 id
loops	一周7天定时执行次数，"0000000"，每一位 0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六 。"0000000" 表示只执行一次，"1111111" 表示每天执行。
dps	由 dpId 和 dpValue 构成的键值对，例如{"1":true}
time	下发执行的时间，例如"14:00",目前只支持24小时制
callback	回调，返回成功或失败的结果，不能为 null

这个方法即可实现添加多个 dp 点的功能：

```
HashMap<String, Object> dpCodeMap =new HashMap<>();
dpCodeMap.put("1",true);
dpCodeMap.put("2","white");
dpCodeMap.put("3",254);

TuyaHomeSdk.getTimerManagerInstance().addTimerWithTask( taskName: "task02",  devId: "1111111",OperatorValuePresenter.device_id,
        dpCodeMap,  time: "07:00", new IResultStatusCallback() {
            @Override
            public void onSuccess() {
                Toast.makeText(mAc,  text: "添加定时任务成功", Toast.LENGTH_LONG).show();
            }

            @Override
            public void onError(String errorCode, String errorMsg) {
                Toast.makeText(mAc,  text: "添加定时任务失败 " + errorMsg, Toast.LENGTH_LONG).show();
            }
        });
```

新接口 API:

```
void addTimer(TuyaTimerBuilder builder, final IResultCallback callback);
```

参数说明

TuyaTimerBuilder 及 IResultCallback 说明

参数	说明
taskName	定时分组名称
devId	设备 id 或群组 id
loops	一周7天定时执行次数，"0000000", 每一位 0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六 。"0000000" 表示只执行一次，"1111111" 表示每天执行。
actions	dps任务 json格式: {/"dps/":{},/"time/":""}，其中 raw 类型的 dp 点需要格式转换： `new String(Base64.encodeBase64(HexUtil.hexStringToBytes(dps)))`
status	初始化定时器开关状态 0：关闭，1：开启
appPush	是否支持定时执行结果推送
aliasName	设置定时备注名
TimerDeviceTypeEnum	枚举，定时类型：设备定时或群组定时
callback	回调，返回成功或失败的结果，不能为 null

在写入 actions 参数是仿照老接口是不行的，actions 是 String 型参数，所以需要转换成 String 型。

```
public TuyaTimerBuilder.Builder actions(String actions) {
    this.actions = actions;
    return this;
}
```

```
HashMap<String, Object> dpCodeMap =new HashMap<>();
dpCodeMap.put("1",true);
dpCodeMap.put("2","white");
dpCodeMap.put("3",254);

TuyaTimerBuilder builder = new TuyaTimerBuilder.Builder()
        .taskName("task01")
        .devId(OperatorValuePresenter.device_id)
        .deviceType(TimerDeviceTypeEnum.DEVICE)
        .actions(dpCodeMap.toString())
        .loops("0111111")
        .aliasName("Test")
        .status(1)
        .appPush(true)
        .build();

TuyaHomeSdk.getTimerInstance().addTimer(builder, new IResultCallback() {
    @Override
    public void onSuccess() {
        Toast.makeText(mAc,  text: "添加定时任务成功", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onError(String errorCode, String errorMsg) {
        Toast.makeText(mAc, text: "添加定时任务失败", Toast.LENGTH_LONG).show();
    }
});
```

文档里面 actions 是 JSON 格式的数据，于是做了后面的改动，结果还是失败。

TuyaTimerBuilder 及 IResultCallback 说明

参数	说明
taskName	定时分组名称
devId	设备 id 或群组 id
loops	一周7天定时执行次数，"0000000", 每一位 0:关闭,1:开启, 从左至右依次表示: 周日 周一 周二 周三 周四 周五 周六 。"0000000" 表示只执行一次，"1111111" 表示每天执行。
actions	dps任务 json格式: {/"dps/":{},/"time/":""}, 其中 raw 类型的 dp 点需要格式转换： `new String(Base64.encodeBase64(HexUtil.hexStringToBytes(dps)))`
status	初始化定时器开关状态 0：关闭，1：开启
appPush	是否支持定时执行结果推送
aliasName	设置定时备注名
TimerDeviceTypeEnum	枚举，定时类型：设备定时或群组定时
callback	回调，返回成功或失败的结果，不能为 null

```
HashMap<String, Object> dpCodeMap =new HashMap<>();
dpCodeMap.put("1",true);
dpCodeMap.put("2","white");
dpCodeMap.put("3",254);

HashMap<String, Object> actions =new HashMap<>();
actions.put("time","07:00");
actions.put("dps",dpCodeMap);

TuyaTimerBuilder builder = new TuyaTimerBuilder.Builder()
        .taskName("task01")
        .devId(OperatorValuePresenter.device_id)
        .deviceType(TimerDeviceTypeEnum.DEVICE)
        .actions(actions.toString())
        .loops("0111111")
        .aliasName("Test")
        .status(1)
        .appPush(true)
        .build();

TuyaHomeSdk.getTimerInstance().addTimer(builder, new IResultCallback() {
    @Override
    public void onSuccess() {
        Toast.makeText(mAc,  text: "添加定时任务成功", Toast.LENGTH_LONG).show();
    }
```

文档中通过

TuyaHomeSdk.getDataInstance().getStandardConverter().convertCodeToIdMap(Map<String, Object> dpCodes, String devId)方法转换后还在不好使，转换出的 dps 为{}，结果还是失败。。。

定时器

现有的定时器不支持标准Code 定时，需要进行标准Code转义成Id才能进行设置定时器，参考原有定时器

方法调用

```
/**
 * @param dpCodes  标准DpCode指令
 * @param devId    设备ID
 */
Map<String, Object> convertCodeToIdMap(Map<String, Object> dpCodes, String devId);
```

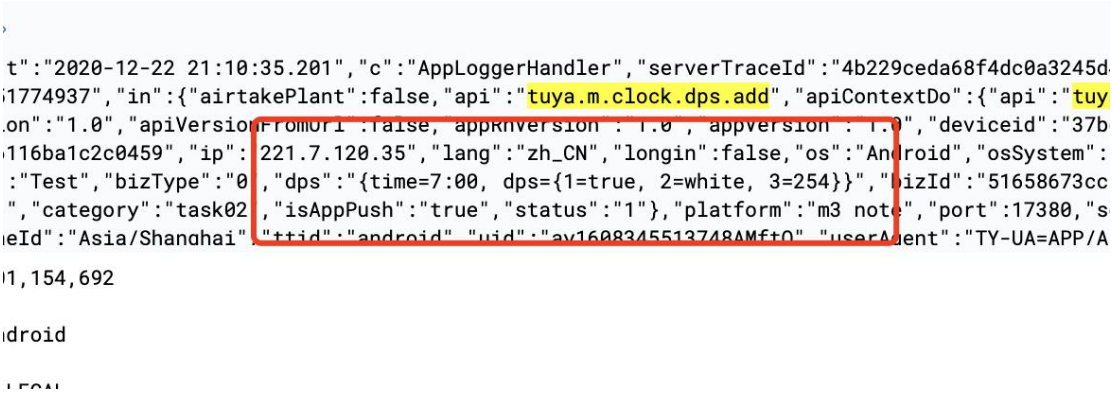
示例代码

```
Map<String, Object> dps=TuyaHomeSdk.getDataInstance().getStandardConverter().convertCodeToI
TuyaTimerBuilder builder = new TuyaTimerBuilder.Builder()
        .taskName(mTaskName)
        .devId("efw9990wedsew")
        .deviceType(TimerDeviceTypeEnum.DEVICE)
        .actions(dps)
        .loops("1100011")
        .aliasName("Test")
        .status(1)
        .appPush(true)
        .build();
TuyaHomeSdk.getTimerInstance().addTimer(builder, new IResultCallback() {
    @Override
    public void onSuccess() {
            }

    @Override
    public void onError(String errorCode, String errorMsg) {

    }
});
```

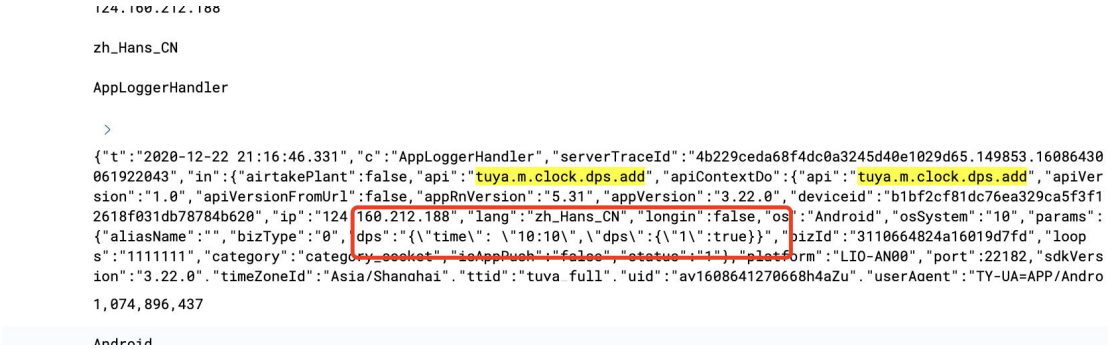
但是还是失败了，最后我想自己一个人埋头苦干还是不行，于是找”小助手”姐姐帮忙，她也非常的耐心，最后我把问题发在群里并@了龙马大佬，大佬一点也没有架子，尽管是非常简单的问题，大佬也是非常耐性的回答。

经过几番折腾后，大佬发了两张截图：

这张是失败的（如下）



这张是成功的（如下）



于是我就把成功的那串字符（{“time”:”00”,”dps”:{“1”:true}}）复制进 actions 参数，结果成功了！于是我就问大佬怎么实现扎转化的，趁到大佬没回复我的时候，我就想这个是标准的 JSON 格式数据，可不可以用 JSON 来实现的，果不其然，成功能了！后来大佬回复说可用 fastJson 实现，嗯！后面值得去学习一下。

```
public void addTimer(String setTime) {
    JSONObject dps =new JSONObject();
    try {
        dps.put( name: "1",  value: true);
        dps.put( name: "2",  value: "white");
        dps.put( name: "3",  value: 254);
    } catch (JSONException e) {
        e.printStackTrace();
    }

    JSONObject jsonObject =new JSONObject();
    try {
        jsonObject.put( name: "time", setTime);
        jsonObject.put( name: "dps", dps);
    } catch (JSONException e) {
        e.printStackTrace();
    }

    TuyaTimerBuilder builder = new TuyaTimerBuilder.Builder()
            .taskName("task01")
            .devId(OperatorValuePresenter.device_id)
            .deviceType(TimerDeviceTypeEnum.DEVICE)
            .actions(jsonObject.toString())
            .loops("0111111")
            .aliasName("Test")
            .status(1)
            .appPush(true)
            .build();

    TuyaHomeSdk.getTimerInstance().addTimer(builder, new IResultCallback() {
        @Override
        public void onSuccess() {
            Toast.makeText(mAc...  text: "添加定时任务成功"...  Toast.LENGTH_LONG).show();
```

四．个人心得

我现就职于重庆一家民营企业，在公司主要负责仪表相关产品的软件开发工作，也就是主要让代码和硬件交流的工作。 虽说已经毕业一年半了，但是始终觉得自己在技术上没有多大的进步，我对自己非常不满意，于是着手规划一些自己的技术能力提升计划，想起自己毕设是做智能家居相关的作品，于是就想打造属于自己的智能家居系统，最终部署到自己的家中。

想法固然美好，实现起来却异常艰难，从搭建 MQtt 服务器，写 APP，搞硬件，写代码，这些都去做了一遍，但是结果却不敬人意，虽说是实现了远程控制功能， 但是我是一个不太满足的人（可以说比较贪心吧。。。），功能太单一了，有同学说调侃说这是一种简约风，哈哈。那就继续构造吧，但是系统构建的复杂性，对于个人来说，工作量异常的大，利用业余时间去完成，估计得花不少时间，起初写 APP 用 E4A 写，E4A 是完全中文编程，学习起来也不是太难，代码从零开始写，写了 3000 多行，但是自己还是不太满意，想实现的功能要受限于别人写好的类库，于是去年元旦后开始自学 android ,由于自己不会 java ,刚开始看郭神的数据有点吃力，后来春节在家开始自学 java，把基础都过了一遍，后面看起数来也没那么吃力了。

基于以上痛点自己不得不重新寻找一个比较规范可靠的物联网平台，基于网联平台来打造自己的智能家居系统。恰巧去年有学校老师找到我，要我给他做一套智能家居系统，要有 APP。起初选择的是中移物联的 onenet 平台，但是也有它的一些缺点，很多都要自己去实现，包括硬件设备去联网，自己还得写物联网模组的代码，也没有较标准化的设备管理协议，虽说是减轻了我的的工作量，但是并没有让我感到满意。直到有一天晚上我躺在床上刷微信时，看到涂鸦公众号（之前参加立创训练营关注的）推送基于 APP SDK 开发活动时，我想都没想都报了名，经过这一段时间的对涂鸦的摸索及了解，发现这就是一片新天地呐，哈哈，爱了爱了。标准化的设备管理，丰富的 API,零代码物联网模组…哇塞！太完美了，涂鸦牛 X。

接触的这段时间里我时常进入涂鸦智能官网，觉得涂鸦实在是太厉害了！之前看涂鸦的直播，大佬说把小米的智能家居产品生态链比作苹果 IOS，那涂鸦就是安卓，这个比喻也确实恰当，从这几年涂鸦的发展成果来看，也确实做到了。

参加此次训练营让我收获颇丰，也让我找到自己想找到的物联网平台，离实现我的目标也不远了。当然从开始运行 demo 到现在还是遇到不少坎坷的，刚开始运行 demo 还是出问题，询问群里小伙伴，结果发现发现是自己的 app key 没复制完。。。我也太坑了叭~，运行 demo 成功后发现添加场景太多，但是也没有删除的功能，于是就查文档，最终成功实现了删除，也遇到太大的麻烦，就是多花了点时间去阅读源码。后面我就想：如果把灯插在床头设置定时点亮，我就不用起床开灯了吧（冬天起床开灯简直要冷成二哈）！要得！开干！查看文档，用老接口实现比较顺利，但是用新接口可是花了我不少时间的，老接口既然都可以了干嘛还取用新接口呀？很简单！好奇吖！后面通过咨询群类小伙伴、小助手姐姐、龙马大佬终于还是解决了。小猪手姐姐特别耐心，还有姐姐的声音真的超甜，尽管很晚问她问题，她都会回我，给小助手姐姐比心，biubiu！龙马大佬也是特别的耐心，尽管是很基础的问题依然还是很耐心的回答我，为大佬大佬 call！

最后，感谢涂鸦智能！感谢龙马大佬！感谢小助手姐姐！感谢群里的各位小伙伴！感谢你们！