吉风物联安卓开发指南

## 简介

在物联网系统中，app客户端也属于设备的一类。在通信中扮演的角色与智能硬件一至。引用sdk后，使app客户端成为万物互联的节点之一。

通过基本的设置，可远程配置、控制设备，实时接收设备状态等数据。

基本工作流程：

1.App启动，调用so库。

2.App tcp登录

3.根据缓存，或远程http获得设备信息为通信层添加设备：

**DeviceHelper**.*AttachDevice()*

4.添加监听函数，来接收来自设备的数据：

**DeviceHelper**.*setSectionListener*(

**new DeviceHelper**.**onSectionListener**() {});

5.向设备发送读来得到设备端口状态

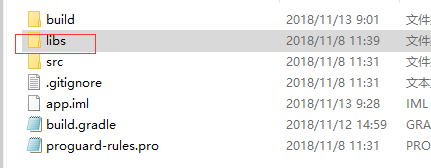
6.向设备发送写 更改设备状态

7.初始添加设备时，可先缓存设备信息，或者上传到自己服务器。

更多详细内容，请参考api

## 导入开发包，工具类

2.1参考demo，把libs开发包放在与src同等级目录。



2.2 修改 build.gradle

在andriod 节点添加

sourceSets {  
 main {  
 jniLibs.srcDirs = ['libs']  
 }  
}

dependencies 节点添加

compile fileTree(include: ['\*.jar'], dir: 'libs')

2.3 参考demo 把 pluto复制在与com同目录下



## 基本配置

3.1 修改 pluto 文件夹里的Pluto.java 的static 启动参数

**NativeInterface**.*reqSetServerUrl*("www.glalaxy.com");*//你的域名，底层会解析这个域名，得到ip后，连接这个ip的服务器***NativeInterface**.*reqSetServerIP*("119.23.8.181");*//域名解析失败后，会默认连接这个ip***NativeInterface**.*reqSetServerTcpPort*(16728);*//服务器tcp 端口***NativeInterface**.*reqSetServerUDPPort*(16729);*//服务器udp端口***NativeInterface**.*reqSetLocalIP*("192.168.1.2");*//本地ip 可不改***NativeInterface**.*reqSetLocalUDPPort*(17729);*//本地端口 可不改*

注：static {} 方法为开启应用时启动，所以无法使用外部变量或方法。

## 四、常用api(示例)

#### 4.1 tcp登录

4.1.1在通信层要与服务器通信之前，首先要登录：

**byte**[] localmac = **Clib**.*hexToBytes*("81" + "86" + "0" + "13888888888");*//手机***byte**[] localSN =**StringHelper**.*strToEncryptKey*("123456", 16);*//密码 与服务器密码一至*

localSN 是字符串密码转byte[] ，使用US-ASCII 编码，长度是16，过长则截取，不足则补0**Pluto**.*reqLogin*(localmac, localSN);*//硬件socket登录*

4.1.2登录状态监听：

**Pluto**.*setOnLoginStateListener*(**new Pluto**.**onLoginStateListener**() {*//登录状态 监听  
 @Override* **public void** recieve(**int** state) {  
 */\* dev\_startLogin,//开始登录 =0  
 dev\_stopLogin,//停止=1  
 dev\_online,//登录成功，在线=2  
 dev\_offline,//离线=3  
 dev\_loginFailed,//登录失败=4  
 dev\_logoutFailed,//退出登录失败=5  
 \*/*  
 }  
});

注：

1. 本地测试，手机号与密码，可乱填。
2. 当长时间在后台运行，再次返回到app 在onResume 里需要重新登录

#### 4.2 smartconfig 给设置入网

Ssid 连接的ssid

Pwd wifi的连接密码

4.2.1广播wifi

**DeviceHelper**.*wifiSmartConfigStart*(Ssid, pwd, **false**, 60000);*//本地广播发送入网配置，超时60S后自动停止*

4.2.2入网的设备监听：

**DeviceHelper**.*setSectionListener*(

**new DeviceHelper**.**onSectionListener**() {*//入网监听  
 @Override* **public void** DeviceJoinIndicates(**byte**[] addr)  
 {  
 injoin(addr);*//有新设备入网* }

});

#### 4.3 扫描发现局域网内的设备

4.3.1先向网内的设备广播 三次beacon

**byte**[] addr = {(**byte**) 0xFF, (**byte**) 0xFF, (**byte**) 0xFF, (**byte**) 0xFF, (**byte**) 0xFF, (**byte**) 0xFF, (**byte**) 0xFF, (**byte**) 0xFF};  
**DeviceHelper**.*reqSendBeacon*((**byte**) 0, addr, **Common**.*getSeq*());*//广播beaocn***DeviceHelper**.*reqSendBeacon*((**byte**) 0, addr, **Common**.*getSeq*());  
**DeviceHelper**.*reqSendBeacon*((**byte**) 0, addr, **Common**.*getSeq*());

4.3.2设备beacon回复 监听

**DeviceHelper**.*setSectionListener*(

**new DeviceHelper**.**onSectionListener**() {*//入网监听*

*@Override***public void** ReceiveBeacon(**byte**[] addr, **int** state) {*//新设备通知 相同设备有可能收到多次*

**synchronized** (list\_dev) {  
 **boolean** isAdd = **true**;*//是否可添加* **for** (**dev\_hardware** item : list\_dev) {  
 **if** (**Arrays**.*equals*(**StringHelper**.*hexStringToBytes*(item.getMac()), addr)) {  
 isAdd = **false**;*//已经存在* **break**;  
 }  
 }  
 **if** (isAdd)*//可添加* {  
 **DeviceHelper**.*AttachDevice*(addr, **null**, **null**, **null**);*//获取设备信息* }  
 }  
}

});

4.3.3往通信层添加设备，并且获取得设备信息

**DeviceHelper**.*AttachDevice*(addr, **null**, **null**, **null**);*//获取设备信息 初次密码传null*

4.3.4 设备信息获取完成回调 监听

**DeviceHelper**.*setSectionListener*(

**new DeviceHelper**.**onSectionListener**() {

*@Override***public void** CompleteDevice(**byte**[] addr, **JSONObject** devInfo, **byte**[] ports, **ArrayList**<**JSONObject**> descList) {*//获取到设备信息* **synchronized** (list\_dev) {  
  
 **boolean** isAdd = **true**;*//是否可添加* **for** (**dev\_hardware** item : list\_dev) {  
 **if** (**Arrays**.*equals*(**StringHelper**.*hexStringToBytes*(item.getMac()), addr)) {  
 isAdd = **false**;*//已经存在* **break**;  
 }  
 }  
 **if** (isAdd)*//可添加* {  
 **dev\_hardware** devHardware = **DevManage**.*getDevInfo*(addr);  
 }  
 }  
}

*})*

注:在和设备通信之前先要给底层添加设备列表,app启动之后，每个设备只需要添加一次即可

**DeviceHelper**.*AttachDevice*(addr, adkey,ckey, gkey);*//添加设备*

adkey管理员密码

ckey普通密码

gkey 体验密码//可不传

#### 4.4 获取单个设备详情

4.4.1使用 获取设备信息

**DeviceHelper**.*reqReadDeviceInfo*(keyid, addr,seq);*//发送获取设备指令*

4.4.2收到设备信息回调

**DeviceHelper**.*setSectionListener*(

**new DeviceHelper**.**onSectionListener**() {

*@Override***public void** ReceiveDeviceInfo(**byte**[] addr, **JSONObject** devInfo, **int** state) {  
  
 **System**.***out***.printf("devvice info addr:"+ **Clib**.*bytes2Hex*(addr,':') + devInfo.toString()+"\r\n");*//获取到设备信息*  
}

}

*});*

4.4.3修改设备信息

**DeviceHelper**.*setLocalLockFlag*(addr, islocal); *//远程锁*  
**DeviceHelper**.*setRemoteLockFlag*(addr, isremote);//本地锁  
**DeviceHelper**.*setGuestLockFlag*(addr, isGeust);//体验锁  
**DeviceHelper**.*setShareLockFlag*(addr, isshare);//分享锁  
**DeviceHelper**.*setDeviceName*(addr, name);*//修改名称*

**DeviceHelper**.*updateDeviceInfo*(addr, keyid);*//更新设备*

#### 4.5 获取状态，发送数据

4.5.1读取某个设备端口当前数据值（如灯是开还是关的状态）

**byte**[] data = **new byte**[1];  
data[0] = 0;

**Aps**.*reqSend*(keyid, mac,**Common**.*getSeq*(), port,aid,**Common**.***cmd\_read***,**Common**.***aID\_Common\_Option***, data, 1);

返回状态监听

**Aps**.*setSectionListener*(**Common**.***aID\_Gen\_Type\_KeyValue***, **Common**.***aID\_Gen\_Type\_Unkown***, **new Aps**.**onSectionListener**() {  
 *@Override* **public void** RecieveCB(**int** keyID, **byte**[] src, **int** seq, **int** port, **int** aID, **int** cmd, **int** option, **byte**[] pdata, **int** len) {*//收到设备状态*  
 myHandler.sendEmptyMessage(1);*//刷新listview* }  
  
});

4.5.2发送数据(并灯，或者关灯)

**byte**[] data = **new byte**[1];  
data[0] = 0;//或data[0]=1;

**Aps**.*reqSend*(keyid, mac,**Common**.*getSeq*(), port,aID, **Common**.***cmd\_write***, **Common**.***aID\_Common\_Option***, data, data.length);

#### 4.6情景编程

4.6.1读取所有情景文件

**Scene**.*reqReadAllName*(keyid, devAddr, **Common**.*getSeq*());

读取情景名称回调

**Scene**.*setSectionListener*(**new Scene**.**onSectionListener**() {  
  
 *@Override* **public void** ReadNameCB(**int** keyID, **byte**[] addr, **String** name, **int** state) {  
 **if** (**Scene**.*isSceneState*((**byte**) state) == **true**) {  
 **if** ((state != **Common**.***op\_faile***)&&(state != **Common**.***op\_invalid***)&&(state != **Common**.***op\_vmInvalid***)&&(state != **Common**.***op\_vmNoTask***)) {  
 **synchronized** (list\_one) {

**boolean** ishave = **false**;  
 **for** (**v\_scene** vS : list\_one) {  
 **if** (vS.getId().equals(id)) {  
 ishave = **true**;  
 **break**;  
 }  
 }  
 **if** (!ishave) {  
  
 **Bundle** bundle = **new** Bundle();  
 bundle.putSerializable("mode", vScene);  
 bundle.putString("id", id);  
 **Message** msg = **new** Message();  
 msg.what = 1;  
 msg.setData(bundle);  
 myHandler.sendMessage(msg);//刷新listview  
 }  
  
 }

}  
 }  
 }  
  
});

4.6.2运行情景

**Scene**.*reqRun*(keyid, addr, **Common**.*getSeq*(),"filename.vm", "");