HeiSha SDK 集成指导 1.14

【修订历史】

时间	编辑	版本	备注
2021-09-15	Wmy	1.0	原始版本
2021-12-02	Wmy	1.14	兼容 1.1.4 及以下版本 SDK

概述

简介

HeiSha Android SDK 对黑砂设备 DNEST 底层协议进行了封装，提供了上层 API 给开发者调用，以实现获取设备运行信息、控制设备完成相关功能、设置设备运行参数等功能，缩短开发者对接时间及阅读协议文档的时间。此对接指南文档记录了对接过程中的常见问题疑惑及 API 调用说明。

SDK 体系架构

Manager

包含 SDK 协议的各类管理器。

Product

定义 HeiSha 产品的基类及其衍生类。

Component

定义了产品的各种组件模块。

SDK 导入

重要提示

建议先测试我们集成 Demo，并仔细阅读我们 Demo 中的集成代码，会对您的对接过程有很大的帮助。集成 Demo 代码已上传到 GitHub 远程库，地址：

https://github.com/Ezio-Wen/HeiSha_SDK_Demo.git

也可以下载 Demo 的集成安装包进行测试使用，下载连接：

http://118.190.91.165:8080/download/HeiSha/Apps/

Android studio 导入 HeiSha SDK 步骤如下

1)获取 SDK，打开 http://118.190.91.165:8080/download/HeiSha/SDK/ ，可先在线阅读此网页目录下的 HeiSha SDK Version Description.pdf ，该文档记录了 HeiSha SDK 的版本记录及各版本更新内容。然后点击下载该网页目录下对应版本的 SDK jar 包。

2）拷贝 HeiSha SDK jar 包到 Android studio 的项目的 lib 目录下，然后打开项目的 app 目录下的 build.gradle 文件，添加以下依赖，然后同步项目。

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.5'
    implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
}
```

3）打开项目的 AndroidManifest.xml，添加网络访问权限。

```
<uses-permission android:name="android.permission.INTERNET" />
```

API 调用

提示

JavaDoc 形式的 API 文档已上传至服务器，可随时查阅，地址：

http://118.190.91.165:8080/download/HeiSha/API/HeiSha_SDK/

连接设备及获取设备组件

1） 调用注册 app 方法，传入连接的黑砂设备的序列号、连接的 MQTT 服务器 URL 及 SDK 管理器回调接口的实例。

```
HSSDKManager.getInstance().registAPP(deviceSerial, serverURI, new SDKManagerCallback() {
    @Override
    public void onRegister() { Log.d(TAG, msg: "onRegister: 注册成功"); }

    @Override
    public void onServerConnected(boolean b, String s) {...}

    @Override
    public void onServerDisconnected(Throwable throwable) {...}

    @Override
    public void onProductConnected(final String deviceName) {...}

    @Override
    public void onProductDisconnected() {...}

    @Override
    public void onComponentChanged(BaseComponent baseComponent, ConnStatus connStatus) {...}
});
```

注册成功后，SDK 会自动连接至服务器，并等待设备登录上线，以上几个回调函数中，

onRegister()在注册成功后回调；

onServerConnected() 在 SDK 成功连接至服务器时回调；

onServerDisconnected()在 SDK 与服务器连接断开是回调；

onProductConnected()将在设备登录上线时回调，一般我们在这里初始化设备实例及组件实例，并且注册各组件的事件回调监听；

onProductDisconnected()在 SDK 与设备连接断开是回调；

onComponentChanged()会在设备主要主件模块连接状态发生改变是回调。

设备上线后，实例化设备和设备的组件，

```
private void initDevice(String productName) {
    mDNEST = (DNEST) HSSDKManager.getInstance().getProduct(productName);
    mCanopy = mDNEST.getCanopy();
    mPositionBar = mDNEST.getPositionBar();
    mCharger = mDNEST.getCharger();
    mEdgeComputing = mDNEST.getEdgeComputing();
    mControlCenter = mDNEST.getControlCenter();
    mAirConditioner = mDNEST.getAirConditioner();
    mRemoteControl = mDNEST.getRemoteControl();
}
```

注册各组件的事件回调监听

```
private void initDeviceCallback() {
    mCanopy.setStateCallback(new CanopyStateCallback() {...});
    mPositionBar.setStateCallback(new PositionBarStateCallback() {...});
    mCharger.setStateCallback(new ChargerStateCallback() {...});
    mEdgeComputing.setStateCallback(new EdgeStateCallback() {...});
    mControlCenter.setStateCallback(new ControlCenterStateCallback() {...});

    mAirConditioner.setStateCallback(new AirConditionerStateCallback() {...});

    mRemoteControl.setStateCallback(new RemoteControlStateCallback() {...});
}
```

除 ControlCenter 模块外，其他组件都只有两个监听事件，分别是

onUpdate()，在组件模块发生属性上报时被回调；onOperateResult()，

在对组件的功能进行操作后收到操作结果是被回调。ControlCenter 模块的

onGetConfigVersionInfo()，在获取设备可配置参数的版本时回调；

onGetConfig()，在获取到设备具体的可配置参数时被回调；

onSetConfig()；在进行设置参数后设备将反馈设置结果，以及出错原因。

OnThingPost()；在设备有各种事件上报时被回调，事件等级有

THING_LEVEL_INFORMATION、THING_LEVEL_WARNING 以及

THING_LEVEL_ERROR 三种。

操作各组件模块提供的功能

1) 开、关和复位 Canopy：

```
case R.id.btn_canopy_open:
    mContainerActivity.mCanopy.startOpening();
    break;
case R.id.btn_canopy_close:
    mContainerActivity.mCanopy.startClosing();
    break;
case R.id.btn_canopy_reset:
    mContainerActivity.mCanopy.resetState();
    break;
```

2) 释放、收紧及复位 Charge Bar：

```
case R.id.btn_position_bar_release:
    mContainerActivity.mPositionBar.startReleasing();
    break;
case R.id.btn_position_bar_tighten:
    mContainerActivity.mPositionBar.startTightening();
    break;
case R.id.btn_position_bar_reset:
    mContainerActivity.mPositionBar.resetState();
    break;
```

3) 开始充电、停止充电、开、关无人机：

```
case R.id.btn_charging_start:
    mContainerActivity.mCharger.startCharging();
    break;
case R.id.btn_charging_stop:
    mContainerActivity.mCharger.stopCharging();
    break;
case R.id.btn_drone_turn_on:
    mContainerActivity.mCharger.getDroneSwitch().turnDroneON();
    break;
case R.id.btn_drone_turn_off:
    mContainerActivity.mCharger.getDroneSwitch().turnDroneOFF();
    break;
```

4) 开、关 Android、NVIDIA 等边缘计算模块的电源：

```
case R.id.btn_android_turn_on:
    mContainerActivity.mEdgeComputing.androidTurnOn();
    break;
case R.id.btn_android_turn_off:
    mContainerActivity.mEdgeComputing.androidTurnOff();
    break;
case R.id.btn_nvidia_turn_on:
    mContainerActivity.mEdgeComputing.NVIDIATurnOn();
    break;
case R.id.btn_nvidia_turn_off:
    mContainerActivity.mEdgeComputing.NVIDIATurnOff();
    break;
```

5) 开关无人机遥控器以及插拔遥控器的 USB 数据线：

```
case R.id.btn_rc_turn_on:
    mContainerActivity.mRemoteControl.RCTurnOn();
    break;

case R.id.btn_rc_turn_off:
    mContainerActivity.mRemoteControl.RCTurnOff();
    break;

case R.id.btn_rc_usb_plug_in:
    mContainerActivity.mRemoteControl.RCUSBPlugIn();
    break;

case R.id.btn_rc_usb_pull_out:
    mContainerActivity.mRemoteControl.RCUSBPullOut();
    break;
```

6) 操作完成后，可在回调函数 ControlCenter.onOperateResult()中得到操作的结果。

7) 获取可配置参数的当前值，只需传入具体参数索引即可，获取完成后，可在回调函数 ControlCenter.onGetConfig()中得到获取的参数值：

```
mContainerActivity.mControlCenter.getConfigParameter(ConfigParameter.SERVICE_PARAM_POST_RATE_CANOPY);
```

8) 设置可配置参数值，传入具体参数索引和要设置的参数值，设置完成后，可在回调函数 ControlCenter.onSetConfig()中得到设置结果：

```
private void setParam(ConfigParameter parameter, int value) {
    if (mContainerActivity.isServerConnected && mContainerActivity.isDeviceConnected) {
        mContainerActivity.mControlCenter.setConfigParameter(parameter, value);
    }
}
```

9)

对接 FQA

1） Q：怎么调用 API 收紧及松开充电限位推杆？

A:用获取到的 PositionBar 组件实例调用收紧及松开方法即可。

```
mPositionBar.startTightening();
mPositionBar.startReleasing();
```

2） Q：调用 PositionBar.getBarLimitSwitchFaultStateSet() 获取到的归中杆限位传感器的错误状态怎么解析？

A：调用 getBarLimitSwitchFaultStateSet() 方法会返回一个 byte 类型的值，这个值的 8bit 从低到高分别代码传感器 S1 到 S8 故障状态，0 表示正常，1 表示故障。

3） Q：调用 mPositionBar.getMotorFaultStateSet() 获取到的归中杆驱动电机的错误状态怎么解析？

A：调用 getMotorFaultStateSet() 方法会返回一个 byte 类型的值，这个值的低 4bit 从低到高分别表示电机 Motor1 到 Motor4 故障状态，0 表示正常，1 表示故障。