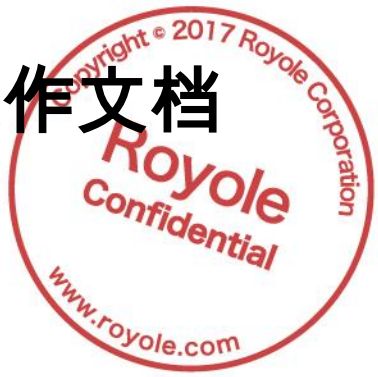
柔宇手写板 SDK 操作文档

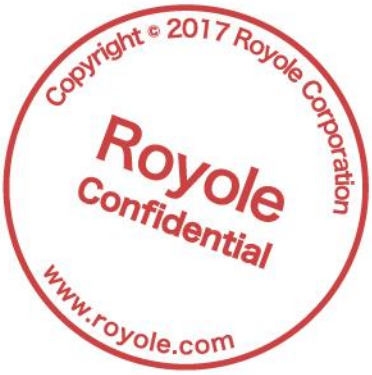


Android 版本

项目名称：RYDrawingSDK

版 本 号：1.0

更新日期：2018/05/09



操作文档更新说明

版本编号	更新说明	更新日期	更新人员	审核人员
1.0	实时模式版本	2018/05/09	牛佳明	赵聪

1、 简介

1、概述

RYDrawingSDK 支持标准蓝牙 4.0 或者以上的设备，本 SDK 定义了柔宇手写板作为蓝牙设备的接入协议，通过本 SDK 提供的接口可以进行扫描柔宇手写板设备，连接设备，读取设备数据，向设备发送数据等操作，从而实现和手写板设备的数据交互。

2、 配置

2.1、运行环境

Android 4.3 或者以上版本的设备

2.2、开发环境配置

本 SDK 以 jar 文件形式提供，将 jar 文件加入项目依赖中即可使用。

开发时请在 AndroidManifest.xml 文件中添加蓝牙权限：


```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

3、SDK 使用快速入门

本 SDK 封装了一个管理类，名为：RyDrawingManager。此类的实现采用了单例模式。RyDrawingManager 中封装了所有与连接、信息获取，事件通知等有关的接口。使用的初始化示例如下：

```
private void initManager() {
    mRyDrawingManager = RyDrawingManager.getInstance();
    //建议传Application的context，以防忘记回收导致泄露。回收资源调用RydrawingManager的destory()方法
    mRyDrawingManager.init(RyApplication.mAppContext);
    mRyDrawingManager.setLeScanListener(this);                    //扫描状态通知
    mRyDrawingManager.setPushEventListener(this);                 //设备推送信息通知
    mRyDrawingManager.setDrawingDataListener(this);               //画笔数据通知
    mRyDrawingManager.setDrawingBusinessListener(this);           //命令回包通知
    mRyDrawingManager.setRyDrawingServiceConnectionListener(this); //服务连接状态通知
}
```

以上代码获得了 RyDrawingManager 的实例，并调用 init 方法进行了初始化（必须）。其中 init 方法的参数需要传递一个 Context 对象。建议传入 Application 的 Context，以防泄露。

以上代码中还调用了 5 个注册 Listener 的接口，用来监听状态、请求，通知的回调事件。此处代码只是起到示例作用，这些接口何时调用请根据具体业务情况决定。

3.2 扫描

```
private void startScan() {
    if (mRyDrawingManager.isSupportBle()) {
        if (mRyDrawingManager.isBluetoothEnable()) {
            //开始扫描带有手写板服务的设备，20000ms后自动停止扫描
            mRyDrawingManager.startScanRyDrawingDevice(20000);
        } else {
            Toast.makeText(this, "Please open you bluetooth!", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, "Sorry, you device do not support ble!", Toast.LENGTH_SHORT)
                .show();
    }
}
```


以上代码中进行了启动扫描的操作，扫描找到的设备会被传递到之前设置的回调接口中。

```
@Override
public void onDeviceFound(BluetoothDevice device, int rssi, byte[] scanRecord) {
    Log.d(TAG, "onDeviceFound: address = " + device.getAddress() + ", name = " + device
            .getName());
    if (mDevDataList.contains(device)) {
        return;
    }
    mDevDataList.add(device);
    mDeviceAdapter.notifyDataSetChanged();
}
```

以上回调方法接收到扫描到的设备并加入到列表中进行展示。

（注：本 SDK 中已保证所有的回调接口都在主线程中调用。）

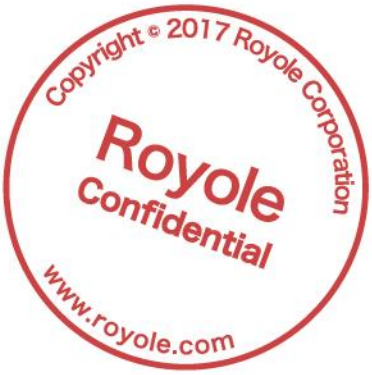
3.3 连接

```
mListview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        BluetoothDevice dev = mDevDataList.get(position);
        mRyDrawingManager.connectDevice(dev);     //连接设备
    }
});
```

以上代码中对找到的设备进行了连接操作。

“connectDevice()”方法对扫描到的指定设备进行了连接操作。

还有一个方法是可以直接传递一个指定设备名称就可以自动进行扫描和连接，此方法是：connectDeviceByName（String devName, int timeOut）。这个方法将扫描和连接过程统一进行了封装，当扫描到指定名称设备时便会自动进行连接。（注：只会连接第一个命中的设备，所以目标名称最好唯一）



3.4 回调

```
@Override
public void onDrawingServiceStateChange(int oldState, int newState) {
    Log.d(TAG, "onDrawingServiceStateChange: odlState = " + oldState + ", newState = " + "" +
            "" + "" + "" + newState);
    if (newState == Constant.ServiceConnectionState.STATE_CONNECTED) {
        Toast.makeText(MainActivity.this, "drawing service connected!", Toast.LENGTH_SHORT)
                .show();
        if (mRyDrawingManager.isLeScanning()) {
            mRyDrawingManager.stopLeScan();
        }
        mRyDrawingManager.prepareDevice();//连接成功后要调用此方法初始化设备
    } else if (newState == Constant.ServiceConnectionState.STATE_DISCONNECTED) {
        Toast.makeText(MainActivity.this, "drawing service disconnected!", Toast
                .LENGTH_SHORT).show();
    }
    mCurrentConnectedState = newState;
    mJumpBtn.setEnabled(mCurrentConnectedState == Constant.ServiceConnectionState
            .STATE_CONNECTED);
    mDisconnectBtn.setEnabled(mCurrentConnectedState == Constant.ServiceConnectionState
            .STATE_CONNECTED);
    mScanBtn.setEnabled(mCurrentConnectedState != Constant.ServiceConnectionState
            .STATE_CONNECTED);
}
```

连接成功后会回调此方法返回最新的连接状态。

3.5 业务接口信息

详细信息请见下节中 SDK 方法详细说明。

4、SDK 方法使用详解

4.1、基础功能

getInstance()	
方法描述	获得 RyDrawingManager 单例对象
参数	无
返回值	RyDrawingManager
示例	无

init()	
方法描述	初始化 RyDrawingManager
参数	Context
返回值	无
示例	无

注意：RyDrawingManager 的所有方法必须在调用 init 方法后才能正常使用。此处建议传入 Application 的 context 对象做为参数。

destory ()	
方法描述	销毁 RyDrawingManager，释放资源和引用
参数	无
返回值	无
示例	无

4.2、蓝牙连接

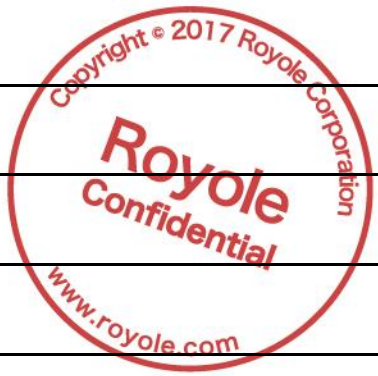
4.2.1、扫描设备

startScanRyDrawingDevice	
方法描述	移动端扫描手写板设备。（注：默认扫描停止时间为 10s，下同）
参数	无
返回值	void
示例	

startScanRyDrawingDevice	
方法描述	移动端扫描手写板设备，可自行设置扫描时常。
参数	@param period 扫描时长，单位为毫秒
返回值	void
示例	

startScanAllLeDevice	
方法描述	移动端扫描所有 ble 设备
参数	无
返回值	void
示例	

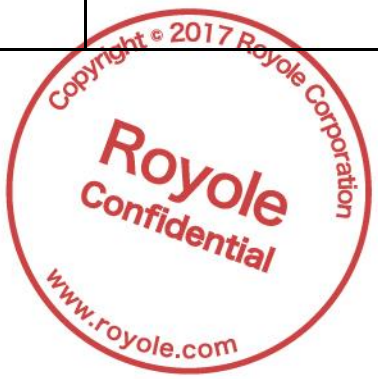
startScanAllLeDevice	
方法描述	移动端扫描所有 ble 设备
参数	@param period 扫描时长，单位为毫秒
返回值	void
示例	



isLeScanning	
方法描述	判断是否正在进行扫描
参数	无
返回值	Boolean true ： 正在扫描 ； false：没有进行扫描
示例	

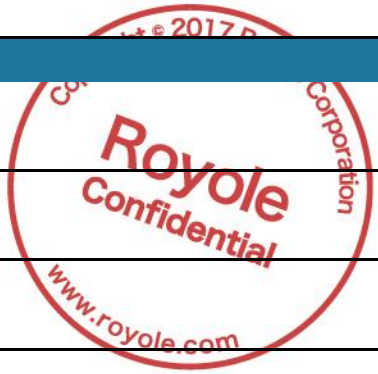
4.2.2、停止扫描设备

stopLeScan	
方法描述	移动端退出扫描手写板设备
参数	无
返回值	void
示例	



4.2.3、连接设备

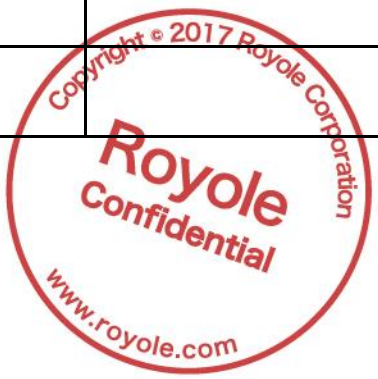
connectDevice	
方法描述	移动端通过蓝牙连接 Bluetooth 设备
参数	@param BluetoothDevice 蓝牙客户端对象
返回值	void
示例	



connectDeviceByName	
方法描述	移动端通过写字板设备名称自动扫描与连接写字板（注：调用此方法会自动开启扫描，找到设备后会自动连接与目标设备名相同的第一个设备）
参数	@param deviceName 要连接写字板身边的名字 @param timeOut 扫描超时时间，单位 ms
返回值	void
示例	

4.2.4、断开当前连接

disconnectDevice	
方法描述	断开移动端与手写板设备的蓝牙连接
参数	无
返回值	void
示例	



4.2.5、获取连接状态

getRyConnectionState	
方法描述	获取移动端与手写板的连接状态
参数	无
返回值	void
示例	Constant.ServiceConnectionState //移动端与手写板连接状态 Constant.ServiceConnectionState.STATE_CONNECTING, //移动端和手写板正在连接 Constant.ServiceConnectionState.STATE_CONNECTED, //移动端和手写板已经成功连接 Constant.ServiceConnectionState.STATE_DISCONNECTED //移动端和手写板正在断开连接

4.2.6、是否支持 ble

isBluetoothEnable	
方法描述	获取移动端与手写板的连接状态
参数	无
返回值	Boolean ***true：支持*** ***false：不支持***
示例	Constant.ServiceConnectionState //移动端与手写板连接状态 Constant.ServiceConnectionState.STATE_CONNECTING, //移动端和手写板正在连接 Constant.ServiceConnectionState.STATE_CONNECTED, //移动端和手写板已经成功连接 Constant.ServiceConnectionState.STATE_DISCONNECTED //移动端和手写板正在断开连接

4.3、配置信息

4.3.1、获取电量

getBatteryInfo	
方法描述	获取手写板电量信息
参数	无
返回值	void
示例	

4.3.2、获取手写板信息

getDeviceDescInfo	
方法描述	获取手写板基础信息
参数	无
返回值	void
示例	

4.3.3、获取手写板信息

getDeviceWidth	
方法描述	获取设备宽度
参数	无
返回值	void
示例	

4.3.4、获取手写板信息

getDeviceHeight	
方法描述	获取设备高度
参数	无
返回值	void
示例	

4.4、设置监听器

4.4.1、扫描状态监听器

setLeScanListener	
方法描述	设置 ble 扫描状态监听器
参数	IScanListener （注：回调接口中具体方法说明见下节内容，下同。）
返回值	void
示例	

4.4.2、手写板服务连接状态监听器

setRyDrawingServiceConnectionListener	
方法描述	设置手写板服务连接状态监听器
参数	IDrawingServiceConnectionListener
返回值	void
示例	

4.4.3、书写事件监听器

setDrawingDataListener	
方法描述	设置手写板书写事件监听器
参数	IDrawingDataListener
返回值	void
示例	

4.4.4、手写板回复监听器

setDrawingBusinessListener	
方法描述	设置手写板返回数据的监听器
参数	IDrawingBusinessListener
返回值	void
示例	

4.4.5、手写板推送事件监听器

setPushEventListener	
方法描述	设置手写板推送事件的监听器
参数	IPushEventListener
返回值	void
示例	

5、回调接口详细描述

5.1、IScanListener

5.1.1、扫描开始回调

onLeScanStart	
方法描述	开启 ble 扫描时被调用
参数	无
返回值	无
示例	

5.1.2、扫描结束回调

onLeScanEnd	
方法描述	停止 ble 扫描时被调用
参数	无
返回值	无
示例	

5.1.3、蓝牙状态改变回调

onBluetoothConnectionStateChange	
方法描述	蓝牙连接状态变化时被调用
参数	@param oldState @param newState 参数值请参考 BluetoothAdapter 中关于蓝牙状态变化的常量定义
返回值	无
示例	

5.1.4、扫描到设备回调

onDeviceFound	
方法描述	扫描到设备时被调用
参数	@param BluetoothDevice 蓝牙设备封装类 @param rssi 信号强度 @param scanRecord 远程设备提供的附加内容
返回值	无
示例	

5.2、IDrawingServiceConnectionListener

5.2.1、没有找到写字板服务

onDrawingServiceNotFoundError	
方法描述	扫描到设备时被调用
参数	连接后没有找到写字板服务时被调用，SDK 会自动断开连接。
返回值	无
示例	

5.2.2、通过蓝牙名称连接设备发生超时回调

onConnectDeviceByNameTimeout	
方法描述	通过蓝牙名称连接设备，发生超时错误时被调用
参数	@param devName 发生连接超时的蓝牙名称
返回值	无
示例	

5.2.3、写字板服务连接状态变化

onDrawingServiceStateChange	
方法描述	写字板服务连接状态变化时被调用
参数	@param oldState @param newState Constants. ServiceConnectionState. STATE_CONNECTING 连接中 Constants. ServiceConnectionState. STATE_CONNECTED 已连接 Constants. ServiceConnectionState. STATE_DISCONNECTED 已断开
返回值	无

示例	

5.2.4、写字板服务连接出错

onDrawingServiceConnectError	
方法描述	服务连接出错
参数	@param errorCode//错误码
返回值	无
示例	

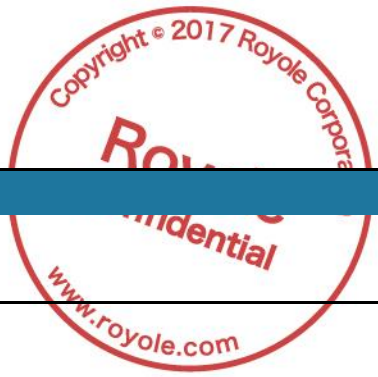
5.3、IDrawingDataListener

onDrawingDataChange	
方法描述	写字板在实时模式下发生写事件时被调用
参数	@param x x 坐标 @param y y 坐标 @param p 压力值 @param time 写字板端时间值
返回值	无
示例	

注：如何知道 down, up , move, hover 等状态？

“IDrawingDataListener”中有一个参数压力值 p，可以根据 p 的值来判断 down 和 up。P 大于 0 就是有接触，小于等于 0 就是 up 或 hover 状态。（这里没有严格的 up 和 hover 的区分，跟笔与板子的距离有关，可以认为这个接口有值传过来但是压力为 0 就是抬笔后的 hover 状态）

5.4、IDrawingBusinessListener

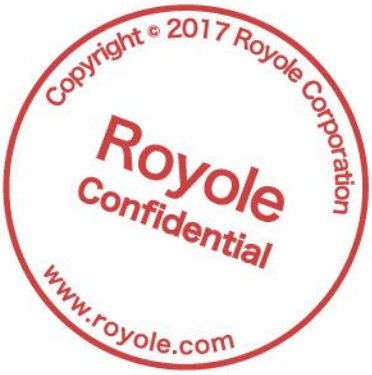


5.4.1、获取电量信息回调

onGetBatteryInfoResponse	
方法描述	获取电量信息返回时被调用
参数	@param resultCode @param percent 当前电量百分百，1-100 @param charging 是否在充电中，0 没有充电，1 在充电
返回值	无
示例	

5.4.2、获取版本号信息

onGetDeviceVersionResponse	
方法描述	获取设备信息返回时被调用
参数	@param resultCode @param deviceName 设备名 @ param version 版本号
返回值	无
示例	



5.5、IPushEventListener

5.5.1、电量低事件

onLowPowerEvent	
方法描述	在手写板端电量低时被调用
参数	@param percent　当前电量百分百，1-100
返回值	无
示例	

5.5.2、写字板 A 按钮被按下事件

onButtonAClick	
方法描述	写字板 A 按钮被按下时被调用
参数	无
返回值	无
示例	

5.5.3、写字板 B 按钮被按下事件

onButtonBClick	
方法描述	写字板 B 按钮被按下时被调用
参数	无
返回值	无
示例	