PPWrite Android SDK

参考手册

V4.0.8版

@2019 RobotPen-PPWrite Android SDK 参考手册

**版本记录**

|  |  |  |  |
| --- | --- | --- | --- |
| V1.0 | 更新画布功能 | 2016-3-20 |  |
| V1.1 | 更新白板功能，包括录制、截屏等功能 | 2016-6-28 |  |
| V1.15 | 增加P2P功能，包括音频、笔迹、图片同步等 | 2016-7-23 |  |
| V2.3.2 | 修改笔服务创建形式，修改画布类等 | 2016-11-2 |  |
| V2.4.7 | 修改了笔服务的封装方式，将常用方法进行开放 | 2016-12-5 |  |
| V2.5.2 | 修改了笔服务的提供和封装方式 | 2017-1-26 |  |
| V2.5.9 | 由jcenter提供笔服务 | 2017-2-15 |  |
| V3.0.0 | SDK重新整合，文档结构调整，修改部分集成流程，增加了部分接口，修改相关方法名。 | 2017-11-15 |  |
| V4.0.0 | SDK重新整合，增加优化旧接口，增加新接口 | 2019-03-04 |  |
| V4.0.1 | SDK加入红笔，橡皮擦功能 | 2019-03-11 |  |
| V4.0.2 | SDK加入usb主动连接，并加入新的usb断开类型 | 2019-03-18 |  |
| V4.0.3 | SDK加入usb性能优化 | 2019-03-26 |  |
| V4.0.4 | SDK加入手写板原始点旋转（90度，180度，270度，360度） | 2019-03-29 |  |
| V4.0.5 | SDK 加入实时上报电量信息 | 2019-04-02 |  |
| V4.0.6 | SDK修改1插电会不上报连接信息。2多次传入升级指令过滤 | 2019-04-04 |  |
| V4.0.7 | SDK的ble连接加入重试 | 2019-04-22 |  |
| V4.0.8 | SDK的USB切换报点频率 | 2019-05-27 |  |

目录

1. 介绍 3

1.1简介 3

1.2准备环境 4

1. 快速使用-获取点坐标 4

2.1 创建工程 5

2.2 获取点坐标 7

1. 快速使用-白板 10

3.1普通白板 10

3.2录制白板 12

3.3直播间白板 13

1. 快速使用-笔记管理 14

4.1 笔记本的创建、获取、删除 15

4.2 笔记编辑（删除、合并、拆分） 16

1. 设备操作 16

5.1概述 16

5.2具体方法 17

1. 详细方法说明详见JAVA\_DOC. 19

## 介绍

#### 1.1简介

写写是依托电磁本、智能笔两个必备的硬件设备及其附件产品的配合提供微课录制、直播活动的专业解决方案。目前电磁本产品分为P1、P7、T7、T7Plus，T8A，X8-A5，T9A，T9W\_A,K7\_C5,W9\_XF等几十款产品。写写产品的主要实现原理为，当书写时，智能笔前端发射器以每秒钟100-200个发出点坐标的数据，电磁板会把这些点数据上报，接收端接收到点数据后可根据业务进行相应的处理，例如whiteBoard会将点数据连接并优化处理从而得到最初绘制时的笔迹。

SDK共有5个包（2个jar包，3个aar包）和1个笔服务（’cn.robotpen.pen:penservice:1.6.22.61’（1.6.22.61为版本号））。

表1-1产品简介及特点说明

|  |  |  |
| --- | --- | --- |
| 产品型号 | 产品尺寸 | 产品特点 |
| P1微课本 | A5（宽\*高17407\* 10751） | USB方式连接 |
| P7智写本 | A5（宽\*高14335 \* 8191） | BLE方式连接 |
| T7 智写板 | A5（宽\*高14335 \* 8191） | BLE USB方式连接、支持离线 |
| T7Plus智写板 | A4（宽\*高22015 \* 15359） | BLE USB方式连接、支持离线 |
| T8A | A4（宽\*高22600 \* 16650） | BLE USB方式连接、支持离线 |
| X8-A5 | A4（宽\*高22100 \* 14650） | BLE USB方式连接、支持离线 |
| T9A | A4（宽\*高22600 \* 16650） | BLE USB方式连接、支持离线 |

SDK根据客户的实际使用需要，分为4大功能：

设备交互（蓝牙版的连接、升级、离线笔记的同步等）

上报点坐标

白板功能

录制功能

#### 1.2准备环境

* JDK1.7及以上
* Android SDK API Level 大于或等于 16
* Android Studio 2.0 或以上版本
* 支持OTG、BLE、语音和视频功能的真机
  + 如果是使用P1款设备则必须支持OTG；
  + 如果是使用蓝牙款设备则支持BLE即可；

注：建议使用AS作为开发的IDE.

## 快速使用-获取点坐标

本章节适用于已拥有自己的画布或者绘制区域，只需要根据智能设备上报的点数据的客户进行参考。

关键步骤：

* + SDK使用“获取点数据的jar”目录中的jar包，结合笔服务’cn.robotpen.pen:penservice:1.6.22.61’（1.6.22.61为版本号，将不断更新）使用。
  + 接收数据的Activity 继承RobotPenActivity。
  + 连接设备（USB设备无需进行连接活动，服务启动后将自动监听）；
  + 实现onPenPositionChanged方法，获取点数据；
  + 设备连接成功后可同步笔记数据、固件升级；

#### 2.1 创建工程

1. 点坐标获取的demo请详见RobotPenDemo\_point.(运行我们的github的demo，使用classpath 'com.android.tools.build:gradle:2.3.3'）

)

1. 通过AS创建一个project，Minimum SDK必须大于等于16；





1. 项目创建后结构如下图所示：



1. 引入必要的jar包、配置必要的权限；

将jar文件夹中的这两个jar包复制到项目的libs文件夹中。

打开项目中的build.gradle文件，添加jcenter上发布的jar包，特别是下图中红框中的包。

compile 'com.android.support:support-v4:25.1.1'

compile 'com.android.support:appcompat-v7:25.1.1'

compile ‘com.android.support:support-annotations:25.1.1'

compile 'org.greenrobot:greendao:3.2.0'

compile 'com.google.code.gson:gson:2.8.0'

compile "com.github.bumptech.glide:glide:3.7.0"

compile "jp.wasabeef:glide-transformations:2.0.1"

compile 'com.amitshekhar.android:glide-bitmap-pool:0.0.1'

compile ‘cn.robotpen.pen:penservice:1.6.22.61’

注 compile ‘cn.robotpen.pen:penservice:1.6.22.61’是引入的笔服务，笔服务的主要功能是提供于设备交互的接口，传递数据。

打开项目中的AndroidManifast文件，添加必须的权限，下图红框的权限为必须添加的权限。

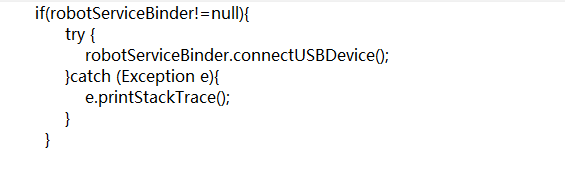


#### 2.2 获取点坐标

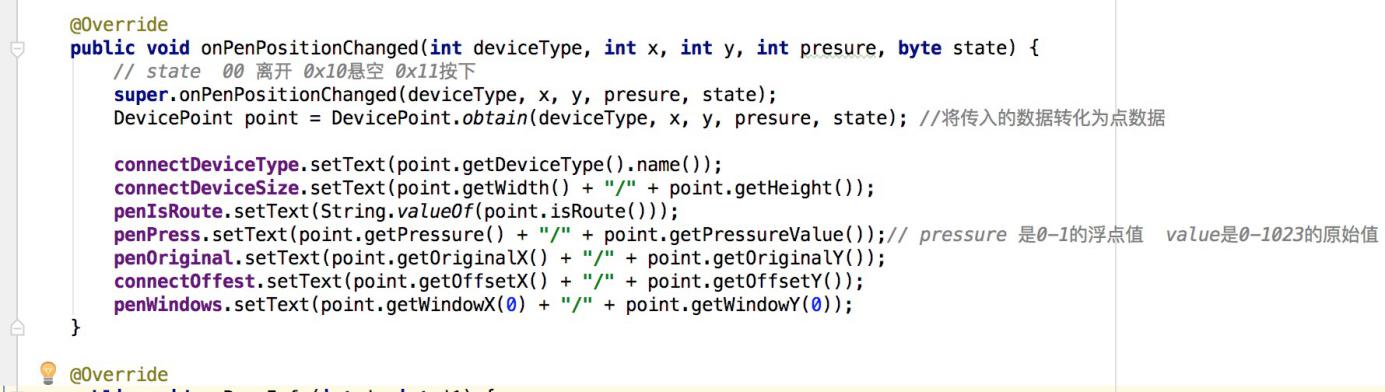
* + 继承 RobotPenActivity，实现未实现的方法；
  + 连接设备（USB设备在USB连接下，可以主动启动服务连接，也可以开关USB进行连接，启动服务后将自动监听）；

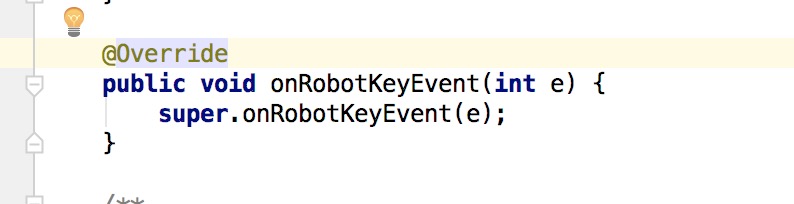
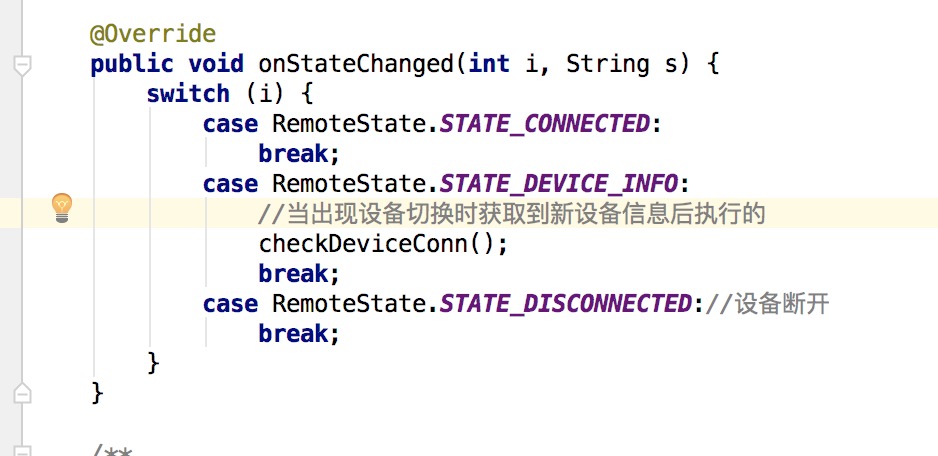
这是蓝牙连接方式，只要RobotPenService启动后在任何情况下，当接入USB设备后都将通过监听弹出授权窗口。

注释：getPenServiceBinder() 是获取笔服务，获取笔服务后，通过getPenServiceBinder().getConnectedDevice()判断是否存在已有的连接，如果未连接，getPenServiceBinder().connectDevice(addr)实现连接手写板蓝牙，addr为蓝牙地址。



这个是USB连接方式，连接前，要保证手写板和android设备保持连接

* + 通过回调获取点数据和其他状态数据；
    - 在onPenPositionChanged方法中获取设备上报的点坐标信息，转化后进行相关的处理。
  + 通过回调获取点数据和其他状态数据;
    - OnStateChange主要是在设备接入、断开等动作时进行触发。设备状态不进行更改时是不会触发此方法。
    - 当硬件设备的有按键需求时（仅限支持按键的设备），可以通过RemoteCallback的onRobotKeyEvent方法进行单击和双击动作的监听。



* + 设备连接成功后可同时同步笔记数据(仅限支持同步笔记功能的设备)、固件升级；
    - 当设备产生离线数据，当设备连接成功可以通过robotService.startSyncOffLineNote();进行离线数据的获取，并通过RemoteCallback的onOffLineNoteHeadReceived、onSyncProgress、onOffLineNoteSyncFinished（离线数据小于1M）、onLargeOffLineNoteSyncFinished(离线数据大于1M),打开离线getPenServiceBinder().requestLargeFile(true);
    - 三个方法进行同步开始、同步进度、同步结束进行同步整个过程的交互。
    - 在检测设备固件的过程中，需要使用写写平台的两个后台接口：

|  |  |
| --- | --- |
| 接口 | 简述 |
| "http://dl.robotpen.cn/fw/"+设备类型名称+"\_svrupdate.txt" | 获取当前类型设备的最新固件版本 |
| "http://dl.robotpen.cn/fw/"+设备类型名+ "\_" + 获取到的最新的版本号 + ".bin"; | 获取当前类型设备的最新固件升级文件 |

经过判断和获取两个步骤后则可以通过robotService.startUpdateFirmware(NewVersion,newFirmwareVer);方法将最新的固件版本和固件文件写入到固件中。同时通过RemoteCallback的onUpdateFirmwareFinished、onUpdateFirmwareProgress两个方法来监听整个升级过程。

注释：继承RobotPenActivity后可通过getPenServiceBinder()获取笔服务robotService

以上具体细节详见RobotPenDemo\_poin 的Demo源码。

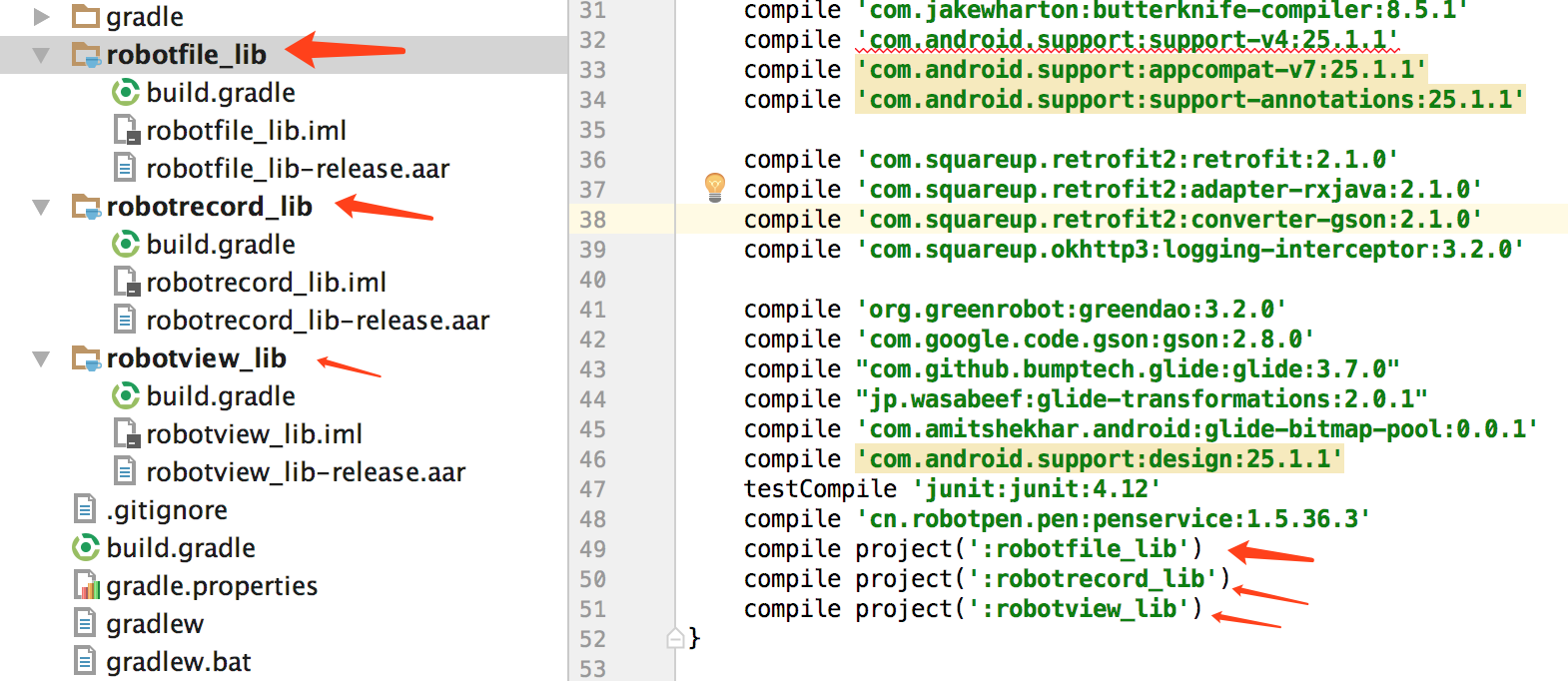
## 快速使用-白板

白板按照大功能分为普通白板、录制白板两大类。

普通白板对应的类是cn.robotpen.views.widget.WhiteBoardView

录制白板对应的类是cn.robotpen.record.widget.RecordBoardView

其中包含诸多操控白板的接口方法，下面将介绍主要的功能。详细方法说明详见JAVA\_DOC

SDK使用”获取点数据的jar”目录中的jar包和“微课版的jar”目录中的aar包（不使用录制可不引入robotrecord\_lib），结合’cn.robotpen.pen:penservice:1.6.22.61’（1.6.22.61为版本号,将不断更新）使用。

注意：使用白板，“获取点数据”的集成任务同样需要集成，“获取点数据”是使用白板的基础。

#### 3.1普通白板

白板首先必须继承RobotPenActivity实现WhiteBoardView.WhiteBoardInterface接口才能够完成在画布上的简单绘制，实现接口后则会出现笔颜色等多个方法，进行相关设置后则即可完成相关功能。

想要实现通过屏幕绘制必须实现以下几处，详见demo中的WhiteBoardActivity。

1. whiteBoardView.setIsTouchWrite(true);//设置笔迹输入方式true 代表手写 false代表笔写
2. whiteBoardView.initDrawArea();//初始化画布的绘制区域，特别是在设备接入、切换或者app进行横竖屏切换时都需要进行这一步，否则将绘制混乱。
3. NoteKey 为白板设置一个唯一的key值，当此key值改变时，画布将进行重新加载。

**将设备和白板结合使用非常简单，通过onPenPositionChanged方法中实现whiteBoardView.drawDevicePoint(type,x,y,presure,state);即可完成白板的绘制，这一步绘制必须手动执行。**

**public void onPenPositionChanged(int deviceType, int x, int y, int presure, byte state) {**

**super.onPenPositionChanged(deviceType, x, y, presure, state);**

**if(isRubber==0){**

**DeviceType type = DeviceType.toDeviceType(deviceType);**

**whiteBoardView.drawDevicePoint(type,x,y,presure,state);**

**}**

**}**

|  |  |
| --- | --- |
| **回调方法** | **简介** |
| getDeviceType | 根据不同类型的设备为白板设置场景，不同类型白板的画布区域尺寸是不一致的。 |
| getPenWeight | 对画笔的粗细进行设置。 |
| getPenColor | 对画笔颜色进行设置。 |
| getIsPressure | 设置绘制笔迹是否体现绘制的压力值。 |
| getIsHorizontal | 设置画布是否为横向 |
| getNoteKey | 设置画布的key值，用于数据库存储和区分 |
| onEvent | 白板的各种事件则是通过此方法回调的。 |
| getCurrUserId | 设置当前用户*ID* |
| getNewNoteName | 创建新笔记，设置笔记名字 |
| onMessage | 返回当前正在处理的消息，设置为*true*拦截，白板不显示 |
| onPageInfoUpdated | 画布初始化后，上报白板笔记页码 |
| onPenServiceError | 笔服务错误返回信息 |
| onSupportPenPressureCheck | 上报是否支持笔压感校验 |
| onCheckPressureing | 正在笔压感校验 |
| onCheckPressurePen | 开始笔压感校验 |
| onCheckPressureFinish | 笔压感校验完成 |

**除了横竖屏调整，其他方法均无需进行白板的刷新，修改后即可生效。**

#### 3.2录制白板

上面说明了白板的绘制实现，如果要使用白板的录制功能，则必须实现RecordBoardView.RecordBoardInterface接口，实现接口后则会出现与录制相关的多个方法，进行相关设置后则即可完成相关录制功能，详见demo中的 RecordBoardActivity。

屏幕录制的主要方法：

1 recordBoardView.startRecord() 开始录制屏幕。

2 recordBoardView.setIsPause(true) 暂停录制。

3 recordBoardView.setIsPause(false) 取消暂停。

4 recordBoardView.endRecord()。结束录制屏幕

|  |  |
| --- | --- |
| **方法** | **简介** |
| getRecordLevel | 设置录制清晰度：22（超清）、12（高清）、2（标清） |
| onRecordButClick | 提供了录制控件，可以监控录制按钮的点击 |
| onRecordError | 录制过程中出现存储空间不足等异常信息时将通过这里进行上报。 |
| onRecordState | 将录制过程中的开始、暂停、结束等全部状态上报。 |
| onRecordTimeChange | 返回false显示默认录制时间，返回true隐藏。 |
| getRecordVideoName | 返回录制的视频文件的名称。 |

**录制白板已经提供了时间控件，可以通过控件知道录制的具体时分秒。**

## 快速使用-笔记管理

笔记管理的类分为两个：

一个是cn.robotpen.views.module.NoteManageModule

一个是cn.robotpen.views.module.TrailsManageModule。

其中包含了诸多管理笔记的接口。下面只介绍主要的方法，详细方法说明详见JAVA\_DOC

#### 4.1 笔记本的创建、获取、删除

首先实例化一个NoteManageModule类，例如：

mNoteManageModule = new NoteManageModule(this, MyApplication.getInstance().getDaoSession());

This 是当前上下文环境Context

MyApplication.getInstance().getDaoSession()是存储笔记的数据库。

例如：

NoteManageModule里面实现了一些管理笔记的接口。主要使用流程如下：

NoteEntity createNote(name, isHorizontal, userId);

功能：创建笔记。

参数： 笔记名称、是否是横屏笔记、用户id

返回：笔记类实体

NoteEntity createNote(String name, boolean isHorizontal, int category, boolean isT9A)

功能：创建针对T9A设备的笔迹。

参数：笔记名称、是否是横屏笔记、笔记所属种类，是否是T9A笔记

返回：笔记类实体

List<NoteEntity> getNotes(int updateTime, int count)

功能：按更新时间获取最新的count条数据

参数：更新时间、获取笔记数目

返回：笔记列表

List<NoteEntity> getAllNotes()

功能：获取全部笔记

返回：笔记列表

boolean deleteNote(String noteKey)

功能：删除笔记

参数：要删除笔记的key

返回：删除结果

#### 4.2 笔记编辑（删除、合并、拆分）

实例化一个NoteManageModule类。

TrailsManageModule mTrailsManageModule = new TrailsManageModule(Context,daoSession);

Context 是当前上下文环境，daoSession是存储笔迹的数据库

void delBlock(String block)

功能：删除区块

参数：被删除页

List<TrailsEntity> divideBlock(String block, long endTime)

功能：拆分笔记

参数：被拆分页、拆分时间点

返回：拆分后产生的笔迹列表

List<TrailsEntity> mergeBlock(String dstBlock, String srcBlock, long srcEndTime)

功能：合并两页笔迹

参数：目标页、被合并页、被合并页的结束时间。

返回：合并后的笔迹列表

## 设备操作

#### 5.1概述

对于手机于手写板的通信的相关操作主要是利用的笔服务：

cn.robotpen.pen.service.RobotRemotePenService.

在继承了RobotPenActivity的类中，调用getPenServiceBinder()可获取笔服务（笔服务已启动的情况下）。通过笔服务可获取笔服下面提供的接口方法与手写板通信。例如：getPenServiceBinder().connectDevice(Mac);此接口是实现蓝牙连接的功能。

#### 5.2具体方法

具体方法如下：

/\*\*

\* 注册回调接口

\*/

void registCallback(IRemoteRobotServiceCallback callback);

/\*\*

\* 取消注册回调接口

\*/

void unRegistCallback(IRemoteRobotServiceCallback callback);

/\*\*

\* 链接设备

\*/

boolean connectDevice(String macAddr);

/\*\*

\* 修改设备名称

\*/

boolean editDeviceName(String name);

/\*\*

\* 断开链接

\*/

void disconnectDevice();

/\*\*

\* 返回当前的模式

\* 0 笔模式，1同步模式，2 OTA模式

\*/

byte getCurrentMode();

/\*\*

\* 进入同步模式

\*/

boolean enterSyncMode();

/\*\*

\* 退出同步模式

\*/

boolean exitSyncMode();

/\*\*

\* 获取当前链接设备

\*/

RobotDevice getConnectedDevice();

/\*\*

\* 开始同步离线笔记

\*/

boolean startSyncOffLineNote();

/\*\*

\* 停止同步离线笔记

\*/

boolean stopSyncOffLineNote();

/\*\*

\* 升级 ble 固件

\*/

boolean startUpdateFirmware(String version,in byte[] data);

/\*\*

\* 升级 mcu 和 ble 固件

\*/

boolean startUpgradeDevice(String bleVersion,in byte[] bleFirmdata,String mcuVersion,in byte[] mcuFirmdata);

/\*\*

\* 退出OTA模式（发生在升级过程中用户强制退出升级页面）

\*/

boolean exitOTA();

/\*\*

\* 退出模组更新模式（发生在升级过程中用户强制退出升级页面）

\*/

void getCurrentModuleVersion();

/\*\*

\* 升级 模组 固件

\*/

boolean startUpdateModule(String version,in byte[] data);

/\*\*

\* 退出模组更新模式（发生在升级过程中用户强制退出升级页面）

\*/

boolean exitModuleUpdate();

/\*\*

\* 设置当前页

\*/

void setPageInfo(int cur,int total);

/\*\*

\* 请求当前的页码信息

\*/

void requestPageInfo();

/\*\*

\* 请求校验压感

\*/

void checkPenPressure();

/\*\*

\* 请求剩余电量

\*/

int getRemainBattery();

/\*\*

\* 请求剩余电量

\*/

Battery getRemainBatteryEM();

/\*\*

\* USB请求报点频率

\*/

requestFrequency(int num);0代表切换每秒80个点上报，1代表切换每秒120个点上报，2代表切换每秒200个点上报.（onRobotKeyEvent返回值110表示每秒80个点上报成功，111表示每秒120点上报成功，112表示每秒200点上报成功）

## 原始点处理方法

/\*\*

\* 获取旋转点

\*/

DevicePoint mDevicePoint=new DevicePoint();  
mDevicePoint.setDeviceType(deviceType);  
mDevicePoint.setOriginalX(x);  
mDevicePoint.setOriginalY(y);  
mDevicePoint.setDegree(180);  
float rotateX= mDevicePoint.getRotateX();  
float rotateY= mDevicePoint.getRotateY();

## 详细方法说明详见JAVA\_DOC.