**PPWrite Android SDK**

参考手册

**V2.5.9版**

@2016 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 |  |

目录

1、 介绍 4

1.1简介 4

1.2准备环境 4

2、 快速使用-点坐标 6

2.1demo图文详解 6

3、 快速使用-白板 12

3.1demo图文详解 12

普通白板 12

录制白板 12

4、 详细方法说明详见JAVA\_DOC. 12

## 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四款产品。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。

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） | 使用USB线进行连接 |
| P7智写本 | A5（宽\*高14335 \* 8191） | BLE方式连接 |
| T7 智写板 | A5（宽\*高14335 \* 8191） | BLE方式连接、支持离线 |
| T7Plus智写板 | A4（宽\*高22015 \* 15359） | BLE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：

笔迹坐标点

微课功能

直播功能

设备交互（蓝牙版的连接、升级、离线笔记的同步等）

#### 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-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：

* + 启动罗博智慧笔服务；
  + 将activity与笔服务进行绑定；
  + 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）；
  + 通过回调获取点数据；
  + 设备连接成功后可同时同步笔记数据(仅限Elite、ElitePlus)、固件升级；
  + 使用完毕，将activity与笔服务进行解绑。

注：与设备的交互操作必须在服务启动之后.

#### 2.1demo图文详解

点坐标获取的demo请详见RobotPenDemo\_point.

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 'com.codingmaster:slog:1.0.0'

compile 'cn.robotpen.pen:penservice:1.2.6'



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。



1. 开始代码示例。
   * 启动罗博智慧笔服务；

创建MyApplication并在创建过程中，创建RobotPenService服务，同时通过startRobotPenService将其启动。

注意：1、正常情况下服务启动后，将在设备的通知栏中进行显示；在实际使用过程中，如果不想将启动的服务显示在通知栏中则可以通过设置robotPenService.startRobotPenService(this.getBaseContext(), **false**);将其隐藏。

2、RobotPenService服务只需要启动一次，在实际使用过程如果只有一个Activity有使用需求则可以将启动操作放置到Activity中执行。

* + 将activity与笔服务进行绑定；

服务启动后，需要通过bindRobotPenService将服务与当前活动的Activity进行绑定。

提示：demo中提供一个基类BaseConnectPenServiceActivity<T extends IRemoteRobotServiceCallback.Stub>统一处理关于服务的相关操作。

* + 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）；

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。

提示：在demo中如果是在activity启动之前接入设备则将无法进行设备的识别，如果需要识别则需要修改为不继承BaseConnectPenServiceActivity在onResume中注册回调来处理。

* + 通过回调获取点数据和其他状态数据；
    - 在demo中的BleConnectActivity中将RemoteCallback进行了注册，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。
    - 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，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Elite、ElitePlus)、固件升级；
    - 当T7或者T7 Plus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。

* + 使用完毕，将activity与笔服务进行解绑。

在Activity使用完毕RobotPenServi后必须通过unBindRobotPenService方法进行解绑。可以详见BaseConnectPenServiceActivity的onDestroy方法。

## 快速使用-白板

#### 3.1demo图文详解

白板按照大功能分为普通白板、录制白板、直播版本三大类。这里先对普通白板和录制白板进行介绍

#### 3.1.1普通白板

白板首先必须实现WhiteBoardView.CanvasManageInterface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);//允许在屏幕上直接绘制
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，只要在设备连接成功后，通过RemoteCallback的onPenPositionChanged方法中实现whiteBoardView.drawLine(p);即可完成白板的绘制，这一步绘制必须手动执行。**

|  |  |
| --- | --- |
| **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。 |
| getIsRubber | 设置当前是否为橡皮擦，当此值大于0时即为橡皮擦。 |
| 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。 |

**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。**

#### 3.1.2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.RecordManageListener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。

|  |  |
| --- | --- |
| **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 |  |

**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。**

## 详细方法说明详见JAVA\_DOC.