**倍泰血压计蓝牙SDK-v2.0.5说明文档**

目录

[1.前言 1](#_Toc12276)

[2. 引入sdk到项目中 1](#_Toc4894)

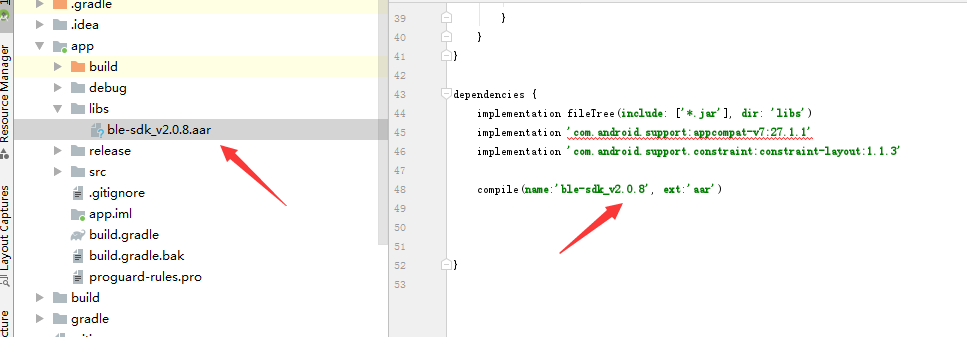
[3. 血压计的使用 3](#_Toc19678)

# 1.前言

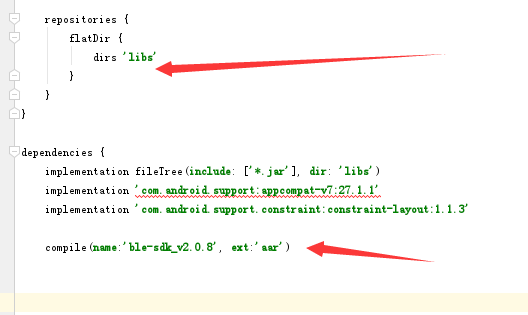
本SDK主要包含倍泰蓝牙通信产生的数据的解析，和与系统蓝牙相关的内容，血压计、于此sdk，具体的android手机蓝牙通信相关的内容可参照demo例子。

# 引入sdk到项目中

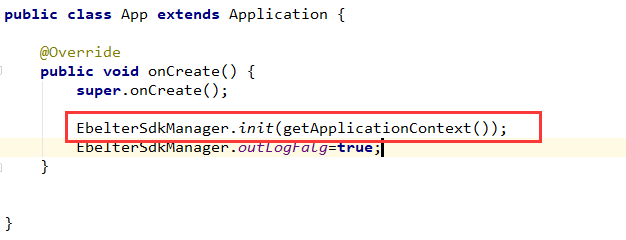
## 第一步：把aar文件下的所有文件复制到项目中的lib库下



## 第二步: 在项目module下的build.gradle文件下添加如下配置



## 第三步：在Application的onCreate方法中初始化一下sdk就能正常使用血压计了



# 血压计的使用

## 第一步：创建一个血压计引擎

BpmManager mBpmManager = **new** BpmManager(**this**);

## 第二步：设置各种监听器

1.设置手机蓝牙开关状态监听器**mBpmManager**.addBluetoothStationListener(**mIBlueStationListener**);

**private final** IBlueStationListener **mIBlueStationListener** = **new** IBlueStationListener() {  
 @Override  
 **public void** STATE\_OFF() {  
 //**手机蓝牙状态:已关闭**  
 }  
  
 @Override  
 **public void** STATE\_TURNING\_OFF() {  
 //**手机蓝牙状态:正在关闭...**  
 }  
  
 @Override  
 **public void** STATE\_ON() {  
 //**手机蓝牙状态:已开启**  
 }  
  
 @Override  
 **public void** STATE\_TURNING\_ON() {  
 //**手机蓝牙状态:正在打开...**  
 }  
};  
  
2. 设置设备蓝牙连接断开的监器**mBpmManager**.setmIConnectStationCallback(**mIConnectStationCallback**);

**private final** IConnectStationCallback **mIConnectStationCallback**=**new** IConnectStationCallback() {  
 @Override  
 **public void** onConnected(ProductStyle productStyle,BluetoothDevice device) {  
 //**已连接**  
 }  
  
 @Override  
 **public void** onConnecting(ProductStyle productStyle,BluetoothDevice device) {  
 //**正在连接**  
 }  
  
 @Override  
 **public void** onDisConnected(ProductStyle productStyle) {  
 //**已断开**  
 }  
};  
  
  
**3. 设置设备测**量的监听器**mBpmManager**.setMesuresCalBack(**mBPMMeasureCallback**);

**private final** BPMMeasureCallback **mBPMMeasureCallback** = **new** BPMMeasureCallback() {  
 @Override  
 **public void** dynamicSystolicPressure(**int** value) {  
 addTv4Str(TimeUtils.*getCurrentTime1*() + **":收到动态压力值 value = "** + value);  
 }  
  
 @Override  
 **public void** measureError(String connectName, **int** errorType) {  
 addTv4Str(TimeUtils.*getCurrentTime1*() + **":收到测量错误信息 errorType = "** + errorType);  
 **if** (TextUtils.*equals*(connectName, BPMConstant.***BPM\_NAME2***)) {  
 String result = **""**;  
 **if** (errorType == 1) {  
 result = **"检测不到足够的心跳或算不出血压，请重新测量！"**;  
 } **else if** (errorType == 2) {  
 result = **"传感器震荡异常！"**;  
 } **else if** (errorType == 3) {  
 result = **"袖带过松或漏气(10秒内加压不到30mmHg)，请检查臂带并重新邦好。"**;  
 } **else if** (errorType == 5) {  
 result = **"测量结果异常！"**;  
 } **else if** (errorType == 7) {  
 result = **"测量出现错误，请重试！"**;  
 } **else if** (errorType == 0x0b) {  
 result = **"电池电量过低，请及时更换电池！"**;  
 } **else** {  
 result = **"测量出现错误，请重试！"**;  
 }  
 addTv4Str(result);  
 } **else** {  
 String result = **""**;  
 **if** (errorType == 1) {  
 result = **"检测不到结果"**;  
 } **else if** (errorType == 2) {  
 result = **"干扰太大"**;  
 } **else if** (errorType == 3) {  
 result = **"充气异常，打不上气或者压力超过上限"**;  
 } **else if** (errorType == 5) {  
 result = **"测量结果异常"**;  
 } **else if** (errorType == 7) {  
 result = **"其他错误"**;  
 } **else if** (errorType == 0x0b) {  
 result = **"电池电量过低，提示用户更换电池"**;  
 } **else** {  
 result = **"测量出现错误，请重试！"**;  
 }  
 addTv4Str(result);  
 }  
 }  
  
 @Override  
 **public void** measureResult(BpmMesureResult mesureResult) {  
  
 *//注意：测量出来的收缩压和舒张压永远是mmHg的值* addTv4Str(TimeUtils.*getCurrentTime1*() + **":收到测量结果 "** + mesureResult.toString());  
 }  
  
 @Override  
 **public void** historyInProgress() {  
 addTv4Str(**"正在接收历史记录"**);  
 }  
  
 @Override  
 **public void** historyResult(BpmHistoryResult bpmHistoryResult) {  
 addTv4Str(TimeUtils.*getCurrentTime1*() + **":收到历史结果 "** + bpmHistoryResult.toString());  
 }  
  
 @Override  
 **public void** historyDome() {  
 addTv4Str( **"历史记录上传完毕"**);  
 }  
  
 @Override  
 **public void** noHistory() {  
 addTv4Str(**"没有历史数据"**);  
 }  
  
 @Override  
 **public void** reviceStartMeasure() {  
 addTv4Str(**"血压计收到开始测量"**);  
 }  
  
 @Override  
 **public void** reviceStopMeasure() {  
 addTv4Str(**"血压计收到停止测量"**);  
 }  
  
 @Override  
 **public void** reviceOffDisplay() {  
 addTv4Str(**"血压计收到关屏"**);  
 }  
  
 @Override  
 **public void** reviceShutdown() {  
 addTv4Str(**"血压计收到关机"**);  
 }

@Override  
**public void** onlyScreen() {  
 addTv4Str(**"13B4-LT专用息屏成功"**);  
}  
  
@Override  
**public void** onlyRecovery() {  
 addTv4Str(**"13B4-LT专用恢复成功"**);  
}

@Override  
 **public void** devicePower(**int** devicePower) {  
 addTv4Str(**"设备电量值 = "** + devicePower);  
 }  
  
 @Override  
 **public void** unitKpa() {  
 addTv4Str(**"单位 Kpa"**);  
 }  
  
 @Override  
 **public void** unitMmhg() {  
 addTv4Str(**"单位 mmHg"**);  
 }  
  
 @Override  
 **public void** pleaseSendUpdateTime() {  
 addTv4Str(**"血压计请求更新时间"**);  
 }  
  
 @Override  
 **public void** isUpdateDeviceTime(**boolean** isSuccess) {  
 addTv4Str(**"血压计更新时间是否成功 = "** + isSuccess);  
 }  
  
 @Override  
 **public void** zhiLin(String s) {  
 addTv4Str(**"原始指令内容 = "**+s);  
 }  
  
};

### 血压计测量结果BpmMesureResult信息说明

**public int mesureUnit**;*//单位类型 1代表mmHg 2代表KPA***public int mesureStation**;*//0正常数据 1正在测量的数据***public int systolicPressure**;*//收缩压***public int diastolicPressure**;*//舒张压***public int pulse**;*//脉搏*

**public boolean** isArrhythmia // 是否心律不齐**public** String **testTime**;*//测量时间 时间格式 "yyyy-MM-dd HH:mm:ss"*

## 第三步：让血压计引擎运行起来 就可以正常使用了

mBpmManager.startWork();

# 4.其他功能说明及细节处理

## 让引擎暂时挂起 处于不运行状态 此时会断开蓝牙设备

**mBpmManager**.stopWork();

## 2、默认为null，引擎会随机扫描进行连接，如果设置了mustConnectAddress，引擎只会连接这个mustConnectAddress对应的设备

**mTemManager**.setMustConnectBlueAddress(mustConnectAddress);//mustConnectAddress

## 3、切换单位 0 mmHg 1 KPA (注意:正在测量的时候无法切换单位 只有在测量完成时或者测量错误之后才可以切换单位)

**mBpmManager**.sendSwitUnitCmd(0);*//把单位切换成mmHg*

**mBpmManager**.sendSwitUnitCmd(1);*//把单位切换成KPA*

## 获取历史记录

**mBpmManager**.get\_history();

## 获取设备SN号

mBpmManager.getDeviceSn();

## 获取电量

**mBpmManager**.sendToGetDevicePower()

## 开始测量

**mBpmManager**.sendToStartMeasure()

## 停止测量

**mBpmManager**.sendToStopMeasure()

## 关机

**mBpmManager**.shutdown()

## 关屏

mBpmManager.offDisplay();

## 11、13B4-LT专用熄屏

**mBpmManager**.onlyScreen();

## 12、13B4-LT专用恢复

**mBpmManager**.onlyRecovery();

## 13、蓝牙是否连接

**mBpmManager**.isConnected()

# 5、引擎退出时要主动释放资源

**if** (**mBpmManager** != **null**) {*//界面退出的时候清理一下 内存回收比较快* **mBpmManager**.removeBluetoothStationListener(**mIBlueStationListener**);  
 **mBpmManager**.exit();  
 **mBpmManager** = **null**;  
}