ProtoBuf协议Android蓝牙SDK操作说明

接入说明

1. 请将SDK中的**xxx.aar**文件拷贝到自己的peoject工程中的lib下
2. 在自己project的app中的build.gradle dependencies下面添加`implementation(name: 'blesdk-debug', ext: 'aar')`然后在build.gradle android中添加如下代码:

```
repositories {
        flatDir {
            dirs 'libs'
        }
    }
```

3. 在Application中启动**BleService.class**和添加`SuperBleSDK.addBleListener(this, new IDataReceiveHandler())`,具体的说明参考示例代码的**BleApplicaiton**.
4. 注册一个广播接收器`MyReceiver.class`继承`BluetoothCallbackReceiver.class`这样全局都可以接收到该接收器的数据,可统一处理分发给需要数据的页面,拷贝Demo中`BaseActionUtils.class`和`BluetoothCallbackReceiver.class`到自己的工程中,可以写在自己Application的`oncrete()`,具体代码如下:

```
IntentFilter intentFilter = BaseActionUtils.getIntentFilter();
MyReceiver receiver = new MyReceiver();
LocalBroadcastManager.getInstance(this).registerReceiver(receiver, intentFilter);
```

5. 复制Demo中`BleApplication.class`中的代码`SuperBleSDK.addBleListener(this, new IDataReceiveHandler())`中的包括实现和发送的广播到代码到自己的Application中.
6. 初始化SDKType:调用代码`SuperBleSDK.switchSDKType(this,Constants.Bluetooth.Zeroner_protobuf_Sdk)`.注意第二个参数选择**Bluetooth.Zeroner_protobuf_Sdk**,然后调用`MyApplication.getInstance().getmService().setSDKType(this.getApplicationContext(), Constants.Bluetooth.Zeroner_protobuf_Sdk)`.**注意此方法必须在`BleService.class`启动才能**执行,如果启动项目就开始扫描建议加一个延时.如:

```
Handler handler = new Handler(Looper.getMainLooper());
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {

MyApplication.getInstance().getmService().setSDKType(this.getApplicationContext(),
Constants.Bluetooth.Zeroner_protobuf_Sdk);
            }
        },3000);
```

扫描连接

1. 建议使用`BluetoothUtil.class`来进行操作.扫描调用`BluetoothUtil.startScan()`方法.通过`LocalBroadcastManager.getInstance(context).registerReceiver(你的接收器对象,BaseActionUtils.getIntentFilter())`来回调.有三个回调方法如下解释:

- `onPreConnect()`发起连接回调
- `onScanResult(WristBand band)`扫描结果回调,在此回调方法中返回扫描的手环
- `connectStatue(boolean isConnect)`连接状态回调,断开和连接成功时候此方法
- `onBluetoothInit()`连接成功,可以和蓝牙进行数据交互,收发指令调用此方法,**一般情况在此方法处理蓝牙连接成功的标识.以便可以进行发送接收指令**

2. 调用`BluetoothUtil.stopScan();`来停止扫描,调用`BluetoothUtil.connect(band)`来进行连接.注意**band对象是`onScanResult(WristBand band)`中的band**

收发指令

1. 发送指令需要在`onBluetoothInit()`之后进行发送数据;
2. 有返回值的命令需要在最后加`BackgroundThreadManager.getInstance().addWriteData(context,cmd)`如:

```
/*设置时间*/
byte[] bytes = SuperBleSDK.getSDKSendBluetoothCmdImpl(context).setTime();
BackgroundThreadManager.getInstance().addWriteData(context,bytes);
```

APP下发类型指令

> 一般下发指令手环返回model解析一样的,通知成功,发送成功或者失败.

- **设置时间**

```
//设置时间 当前系统的实时时间
public byte[] setTime()
//设置时间带参数 ,单位(s)
byte[] setTime(long time)
```

- **设置心率报警**

```
/**
 * 设置心率报警
 *
 * @param
 * @param enable   使能位
 * @param hrH      心率最大值 // 50~200
 * @param hrL      心率最小值 // 40~190
 * @param second   持续时间 (s)
 * @param interval 报警间隔时间(min)
 */
byte[] setHeartAlarm(boolean enable, int hrH, int hrL, int second, int interval)
```

- **设置用户信息**

```
/**
 * 设置用户信息
 *
 * @param height 高 默认170
 * @param weight 体重 kg 默认60
 * @param gender 性别 默认male
 * @param age    年龄 默认20
 * @param walk   50-200之间 默认100
 * @param run    50-200之间 默认100
 */
byte[] setUserConf(int height, int weight, boolean gender, int age, int walk, int run)
```

- **设置运动目标**

```
/**
 * 设置全局配置
 *
 * @param calorie  卡路里 default = 400 kC
 * @param step     default = 10000 unit seconds
 * @param distance default = 10000 unit meters
 */
byte[] setGoalConf(int calorie, int step, int distance)
```

- **设置GPS经纬度海拔**

```
/**
* GPS
*
* @param  altitude  海拔
* @param  latitude  纬度
* @param  longitude 经度
*/
byte[] setGnssConf(int altitude, String latitude, String longitude)
```

- **设置血压**

```
/**
 * 血压
 *
 * @param src_sbp = 2;手环测量舒张压
 * @param src_dbp;手环测量收缩压
 * @param dst_sbp;血压计测量舒张压
 * @param dst_dbp;血压计测量收缩压
 * @param dif_sbp = 6;//保留 传0
 * @param dif_dbp = 7;//保留 传0
 */
 byte[] setBpCaliConf(int src_sbp,int src_dbp,int dst_sbp,int dst_dbp,int dif_sbp,int dif_dbp)
```

- **设置消息免打扰**

```
/**
 * 消息免打扰
 *
 * @param
 * @param policy    true 开启 false 关闭
 * @param startHour 开始小时
 * @param endHour   结束小时
 * @param startMin  结束分钟
 * @param endMin    结束分钟
 */
byte[] setMsgNotificationTime(boolean policy, int startHour, int endHour, int startMin, int
endMin)
```

- **设置消息通知**

```
/**
 * 消息通知
 *
 * @param
 * @param id
 * @param type   ADDED = 0; // incoming call and sms
 *               REMOVED = 1; // incoming call accepted or rejected
 *               UPDATED = 2; // incoming call turn to missed-call
 * @param accept 接收
 * @param reject 拒接
 * @param mute   无声
 * @param title  发送类别
 * @param detail 发送内容
 *               <p>
 *               B.对于来电
 *               当有来电时，Peer应向设备发送一个MsgNotify
 *               {id = xxx，type = TYPE_CALL，status = ADDED，option = {accept = true，reject =
true，mute = true}，title = “Incoming Call” ，detail = “（0755）10010” }
 *               <p>
 *               如果被拒绝或接受，Peer将向设备发送MsgNotify
 *               {id = xxx，type = TYPE_CALL，status = REMOVED}
 *               <p>
 *               否则会错过，Peer将发送一个MsgNotify 给设备
 *               {id = xxx，type = TYPE_CALL，status = UPDATED}
 *               <p>
 *               C.对于短信
 *               当SMS出现时，Peer将向设备发送MsgNotify
 *               {id = xxy，type = TYPE_SMS，status = ADDED}
 */
public byte[] setMsgNotificationNotifyByCall(int id, int type, boolean accept, boolean reject,
boolean mute, String title, String detail)
```

- **设置电话通知**

```
/**
 * 同消息通知
 */
public byte[] setMsgNotificationNotifyByCall(int id, int type, boolean accept, boolean reject,
boolean mute, String title, String detail)
```

- **设置天气**

```
/**
 * @param timeMills   单位s  建议使用当天的凌晨时间作为ID
 * @param weatherDesc // required for ADD
 *                    WeatherDesc {
 *                    Sunny = 0;
 *                    Cloudy = 1;
 *                    Overcast = 2;
 *                    LightRain = 3;
 *                    ModerateRain = 4;
 *                    HeavyRain = 5;
 *                    Rainstorm = 6;
 *                    Snow = 7;
 *                    Haze = 8;
 *                    SandStorm = 9;
 *                    CloudyTurnToFine = 10;
 *                    Thunderstorm = 11;
 *                    }
 * @param weatherType EACH_HOUR = 0;
 *                    EACH_DAY = 1;
 * @param degreeMax   最高温度
 * @param degreeMin   最低温度
 * @param pm2p5       pm2.5
 */
public byte[] setWeather(int timeMills, int weatherDesc, int weatherType, int degreeMax, int
degreeMin, int pm2p5)

/**
 *WeatherEvent 同上面的参数.传24小时天气.这样设置每小时都会变化.
 *
 */
byte[] setWeather(List<WeatherEvent> weatherEvents)
//清除天气指令
byte[] clearWeather()
```

- **添加闹钟**

```
/**
 * 添加闹钟
 * 闹钟个数最好不要超过5个
 * @param id     AlarmId
 * @param repeat repeat times
 * @param week   0000 0001   mon 0x01
 *               0000 0010   tue 0x20
 *               0000 0100    ...
 *               0000 1000
 *               0001 0000
 *               0010 0000
 *               0100 0000
 *               <p>
 *               sample set mon,tue,wed --  0000 0111
 * @param hour   set hour
 * @param min    set min
 * @param text   set text 最长不要超过12字符
 */
byte[] addAlarm(int id, boolean repeat, int week, int hour, int min, String text)
//根据ID移除闹钟
byte[] removeAlarm(int id)
//清除闹钟
byte[] clearAlarm()
```

- **设置久坐提醒**

```
/**
 * 久坐提醒
 *
 * @param repeat
 * @param week      同setAlarm()
 * @param startHour 开始时间 // 0~23
 * @param endHour   结束时间   // 0~23
 * @param duration  // 0~1440
 * @param threshold
 */
byte[] setSedentariness(boolean repeat, int hash, int week, int startHour, int endHour, int
duration, int threshold)
//清除久坐提醒
byte[] clearSedentariness()
```

- **添加日程**

```
/**
 *
 * @param id 唯一ID
 * @param timeMillis 单位s
 * @param text 日程信息
 * @return
 */
byte[] addCalendar(int id, int timeMillis, String text)

//移除日程
public byte[] removeCalendar(int hash, int second)
```

- **设置语言**

```
/**
 * 设置语言
 *
 * @param context
 * @param language English = 0;
 *                 Chinese = 1;
 *                 Italian = 2;
 *                 Japanese = 3;
 *                 France = 4;
 *                 German = 5;
 *                 Portuguese = 6;
 *                 Spanish = 7;
 *                 Russian = 8;
 *                 Korean = 9;
 *                 Arabic = 10;
 *                 Vietnam = 11;
 *                 Polish = 12;
 */
void setLanguage(Context context, int language)
```

- **设置距离,温度,时间,日期格式**

```
/**
 * 设置距离单位
 *
 * @param distance_unit  false[default]: metric unit, true 1: imperial units
 */
byte[] setDistenceUnit(boolean distance_unit)


/**
 * 设置温度单位
 *
 * @param temperature_unit  false[default]: Celsius, true: Fahrenheit
 */


byte[] setTemperatureUnit(boolean temperature_unit)


/**
 * 设置小时格式
 *
 * @param hour_format  false[default]: 24hour, true1: 12hour
 */
byte[] setHourFormat(boolean hour_format)


/**
 * 设置日期格式
 *
 * @param date_format  0[default]: month/day, 1: day/month
 */
byte[] setDateFormat(boolean date_format)
```

- **设置自动心率**

```
/**
 * 设置自动心率
 *
 * @param autoHeartrate true on  false off
 */
byte[] setAutoHeartrate(boolean autoHeartrate)
```

- **设置自动运动**

```
/**
 * 设置自动运动
 *
 * @param auto_sport true on  false off
 */
byte[] setAutoSport(boolean auto_sport)
```

- **设置翻腕亮屏**

```
/**
 * @param lcdGsswitch  0 disable lcd switching by gesture, 1[default]：enable lcd switching by
gesture
 * @param startHour   0[default]~23:  the start hour of switching lcd by gesture
 * @param endHour     0[default]~23:  the end hour of switching lcd by gesture
 */
byte[] setLcdGsTime(boolean lcdGsswitch, int startHour, int endHour)
```

- **设置电机震动**

```
/**
 * 电机震动类型
 *
 * @param mode
 * @param round
 * @param type  ALARM_CLOCK = 0;
 *              INCOMING_CALL = 1;
 *              SMS = 2;
 *              SEDENTARINESS = 3;
 *              CHARGING = 4;
 *              CALENDAR = 5;
 *              DISTANCE_ALARM=6;
 *              HEARTRATE_ALARM=7;
 *              OTHERS = 8;
 */
byte[] setMototConf(int mode, int round, int type)

/**
 * 设置电机震动
 *
 * @param mode  最大15
 * @param round
 */
byte[] setMotorVibrate(int mode, int round)
```

- **设置智拍**

```
/**
 * 设置智拍模式
 *
 * @param mode RT_MODE_BACK_NORMAL = 0; //退出
 *             RT_MODE_ENTER_CAMERA = 1; 进入智拍
 */
byte[] setSmartShotData(int mode)
```

手环上报类型指令

返回值是以json字符串形式返回的,可以根据谷歌Gson或者阿里fastJson解析model

- **获取设备信息**

```
byte[] getHardwareFeatures()

//retrurn model:
public class ProtoBufHardwareInfo {

    private String version;//版本号
    private String model;//名称
    private String mac;
    private String deviceTime;
    private String factory;
    private String fota;
    private int fotaType;
}
```

- **获取电量**

```
//获取电量
byte[] getBattery() 当为电量时.isBattery = true

//返回model
public class ProtoBufRealTimeData {
    /**
     是否显示电量  电量多少 是否在充电
     */
    private boolean isBattery;
    private int  level;//电量
    private boolean charging;//是否充电

    /**
     * 是否显示健康数据  卡路里 距离 步数
     */
    private boolean isHearth;
    private float calorie;
    private int distance;
    private int steps;

    /**
     * 是都显示时间  秒
     */
    private boolean isTime;
    private int seconds;


    /**
     * 是否显示按键时间 按键模式 0 normal 1 camera
     */
    private boolean isKey;
    private int keyMode;
}
```

- **获取健康数据**

```
//获取健康数据
byte[] getRealHealthData()

//返回值同获取电量
```

- **获取设备支持的同步数据**

```
/**
 * 获取设备支持的同步数据
 */
byte[] getDataInfo()

//返回model:
public class ProtoBufSupportInfo {

    private boolean  support_health;
    private boolean  support_gnss;
    private boolean  support_ecg;
    private boolean  support_ppg;
    private boolean  support_rri;
}
```

- **同步历史数据**

1.先获取index table
2.根据index table中的startSeq和endSeq进行历史数据同步.
3.同步类型可通过getDataInfo()来获取支持的类型来同步数据,否则会出新发送指令手环不回复的情况导致APP的假死.

```
/**
 *
 * get index table
 *
 * @param type  可参考文档查看类型
 *        HEALTH_DATA = 0;
 *        GNSS_DATA = 1;
 *        ECG_DATA = 2;
 */
byte[] itHisData(int type)

/**
 * 获取历史数据
 *
 * @param
 * @param type  可参考文档查看类型,通过
          HEALTH_DATA = 0;
 *        GNSS_DATA = 1;
 *        ECG_DATA = 2;
 * @param startSeq 开始seq
 * @param endSeq   结束seq
 */
byte[] startHisData(int type, int startSeq, int endSeq)


//index table类用于记录同步信息
public class ProtoBuf_index_80 extends DataSupport {

    private long uid;
    private String data_from;
    private int start_idx;start seq
    private int end_idx;end seq
    private int year;
    private int month;
    private int day;
    private int hour;
    private int min;
    private int second;
    private int time;//时间戳
    private int indexType;//类型
}
```

由于同步历史数据不是一条数据,数据量很大,SDK仅仅提供命令,示例代码中有关于同步数据demo,如下:

```
//同步历史数据
ProtoBufSync.getInstance().syncData();

//返回model
public class ProtoBuf_80_data {

    private long uid;
    /**
     * data
     */
    private int year;
    private int month;
```

```
    private int day;
    private int hour;
    private int minute;
    private int second;

    /**
     * 时间戳(单位s)
     */
    private int time;
    /**
     * 排序用的
     */
    private int seq;
    /**
     * 设备名
     */
    private String data_from;
    /**
     * 睡眠数据
     */
    private String sleepData;
    private boolean charge;
    private boolean shutdown;
    /**
     * 健康
     */
    private int type;
    private int state;
    private float calorie;
    private int step;
    private float distance;
    /**
     * 心率
     */
    private int min_bpm;
    private int max_bpm;
    private int avg_bpm;
    /**
     * 疲劳度
     */
    private float SDNN;
    private float RMSSD;
    private float PNN50;
    private float MEAN;
    private float fatigue;


    /**
     * 血压
     *
     */
    private int sbp;
    private int dbp;
    private int bpTime;//bp数据

    /**
     * 呼吸训练
     */
    private float mdt_SDNN;
    private  float mdt_RMSSD;
```

```
    private  float mdt_PNN50;
    private  float mdt_MEAN;
    private  int mdt_status;
    private  float mdt_RESULT;
    private  float mdt_RELAX;
}
```

其他指令

- **AGPS升级**

```
仅仅支持GPS功能的设备支持AGPS升级,其他设备不支持.
/**
 * 文件下载init
 *
 * @param fuType
 * @param fileSize
 * @param fileCyc
 * @param fileName
 * @param fileOffset
 */
byte[] setFileDescUpdate(boolean isDesc)
/**
 * 文件下载data
 *
 * @param fuType
 * @param
 * @param
 * @param buf
 */
byte[] setFileDataUpdate(int fuType, int fileCyc, int crc32AtOffset, ByteString buf)
byte[] setFileDataExit(int fuType)

SDK提供AGPS升级的指令.具体的功能可以看示例代码中的demo,如下:
// ProtoBufUpdate.Type.TYPE_GPS FONT
ProtoBufUpdate.getInstance().startUpdate(ProtoBufUpdate.Type.TYPE_GPS);
```

- **固件升级**
  - 参考示例代码 `ProtoBufFirmwareUpdateActivity.class` 类,该类是固件升级的整个实现过程,可以在示例代码APP中查看固件升级的效果.添加自己的工程则需要示例代码中**no.nordicsemi.android**下面的所有文件.

接收数据

- 如**接入说明步骤4**中,可注册一个全局广播接收器继承 `BluetoothCallbackReceiver` 可全局收发广播.分到到具体的类可以使用**EventBus**来发送,实现类如下介绍:

```
/*
 *
 * @param ble_sdk_type 指SDK的类型一般来说指定的是Constants.Bluetooth.Zeroner_protobuf_Sdk
 * @param dataType 指令类型,可参考协议文档
 * @param data SDK解析的数据,通过JSON来解析,SDK使用的是fastJson.关于boolean类型的数据使用Gson有可能出现永远为false的情况,这种情况建议使用fastJson
 */
void onDataArrived(Context context, int ble_sdk_type, int dataType, String data)
```

- 全局广播一般用使用在同步历史数据解析实时数据等.针对某个单独的页面也可以使用局部广播.拷贝Demo中`BleReceiverHelper.class`到工程中的代码如下:

```
BleReceiverHelper.registerBleReceiver(context,new MyDataReceive());

private class MyDataReceive extends BluetoothCallbackReceiver{
    @Override
    public void onDataArrived(Context context, int ble_sdk_type, int dataType, String data) {
        super.onDataArrived(context, ble_sdk_type, dataType, data);
    }
}
```