

一、概述

该SDK 适用于Android 8.0 (API 26) 以上的机型

该SDK支持的设备是无锡富华科技(Fofia)出产的RFID手持机

该SDK 以源代码形式提供(基于Android Studio 开发环境)，工程有2个模块：

- sdk 模块 (FuDeviceSDK.jar)

SDK 本体, 名称空间为 "rid.ptdevice", 编译完成后通过 makejar 任务以 jar 包的形式产出, 位于 "FuDeviceSDK\sdk\release\FUDeviceSDK.jar" 目录下。

客户开发应用程序时,可以将SDK 的jar包 放入自己的工程(也可以考虑将模块源代码直接放入工程项目中)

- app 模块 (app.ptdevice.bletest)

使用SDK开发的演示软件, 供开发人员测试及参考, 演示 APK 位于 "FuDeviceSDK\app\build\outputs\apk" 目录下。

另外, FuDeviceDemo 是另一个基于本SDK开发的独立演示工程, 可作为集成参考。

二、SDK开发说明

2.1 主要接口/类

- BleStateListener

蓝牙设备状态监听接口

- TagListener

标签读取监听接口

- FUDevice

代表富华RFID手持设备(BLE5.0), FUDevice 从BleMonitor 继承而来

2.2 基本操作流程

2.2.1 开启手机端蓝牙权限

这部分不属于SDK, 可以参考FuDeviceDemo 源代码

2.2.2 启动蓝牙扫描并监听

```
1  import android.bluetooth.BluetoothAdapter;
2  import android.bluetooth.BluetoothDevice;
3  import android.bluetooth.BluetoothManager;
4  ...
5
6  public class MainActivity extends AppCompatActivity {
7      ...
8
9      private BluetoothAdapter mBtAdapter;
10
11      @Override
12      protected void onCreate(Bundle savedInstanceState) {
13          ...
14
15          // Register for broadcasts when a device is discovered
```

```

16         IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
17         this.registerReceiver(_mBroadcastReceiver, filter);
18
19         // Register for broadcasts when discovery has finished
20         filter = new
IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
21         this.registerReceiver(_mBroadcastReceiver, filter);
22
23         BluetoothManager manager =
(BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
24         mBtAdapter = manager.getAdapter();
25         mBtAdapter.startDiscovery();
26     }
27
28     private final BroadcastReceiver _mBroadcastReceiver = new
BroadcastReceiver() {
29         @Override
30         public void onReceive(Context context, Intent intent) {
31             String action = intent.getAction();
32             ...
33         }
34     }
35     ...
36 }

```

2.2.3 发现富华RFID设备

注意，富华的设备名一般以"FU"开头,长度为10（部分型号以"PT"或"FS"开头），所以可以用这个条件进行筛选，以过滤不相干的其他设备

```

1     private final BroadcastReceiver _mBroadcastReceiver = new
BroadcastReceiver() {
2         @Override
3         public void onReceive(Context context, Intent intent) {
4             String action = intent.getAction();
5
6             if (BluetoothDevice.ACTION_FOUND.equals(action)) {
7                 //发现一个设备
8                 BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
9                 String name=device.getName();
10                if( name!=null && name.startsWith("FU") &&
(name.length()==10) ){
11                    ...
12                }
13            } else if (
BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action) ) {
14                ...
15            }
16        }
17    };

```

2.2.4 注册富华RFID设备监听器

有2种监听类型, 一种是状态, 一种是标签, 用户可依据需要进行注册

```
1      BleStateListener stateListener = new BleStateListener(){
2          @Override
3          public void onChange(int state, BluetoothGatt gatt) {
4              BluetoothDevice device=gatt.getDevice();
5              ...
6          }
7      };
8
9      BleMonitor.bindStateListener(stateListener);
10
11
12      TagListener tagListener = new TagListener(){
13          @Override
14          public void onMessageReceived( Tag tag,FUDevice device) {
15              ...
16          }
17      };
18
19
20      FUDevice.bindTagListener(tagListener);
```

2.2.5 启动设备监听器

```
1  try {
2      FUDevice device= (FUDevice) FUDevice.getInstance(mDeviceAddress);
3
4      if( device.start()) {
5          ...
6      }else{
7          ...
8      }
9  } catch (Exception e) {
10     e.printStackTrace();
11 }
```

2.2.6 结束设备监听器

```
1  FUDevice device= FUDevice.getInstance(mDeviceAddress);
2  device.stop();
```

三、类库描述

3.1 BleStateListener

3.1.1 原型

```
1  public interface BleStateListener {
2      void onChange(int state, BluetoothGatt gatt);
3  }
```

3.1.2 说明

发现设备状态有改变时触发

- state 设备状态

```
1 public static final int STATE_DISCONNECTED =  
BluetoothProfile.STATE_DISCONNECTED;  
2 public static final int STATE_CONNECTING =  
BluetoothProfile.STATE_CONNECTING;  
3 public static final int STATE_CONNECTED =  
BluetoothProfile.STATE_CONNECTED;  
4 public static final int STATE_DISCONNECTING =  
BluetoothProfile.STATE_DISCONNECTING;
```

- gatt 蓝牙设备, Android内置类BluetoothGatt

3.2 TagListener

3.2.1 原型

```
1 public interface TagListener {  
2     void onMessageReceived(Tag tag,FUDevice device);  
3 }
```

3.2.2 说明

发现RFID设备读到标签时触发

- tag 读到的标签, 解析失败时可能为 null, 请做判空处理
- device 读到标签的设备

3.3 FUDevice

FUDevice从BleMonitor继承而来,代表富华RFID设备

3.3.1 getInstance

通过蓝牙设备编号得到FUDevice 设备类

```
1 public static synchronized FUDevice getInstance(BluetoothDevice device);  
2 public static synchronized FUDevice getInstance(String deviceAddress);
```

3.3.2 bindTagListener

静态方法, 注册标签处理类

```
1 public static void bindTagListener(TagListener tl);
```

3.3.3 BleMonitor.clearAll

BleMonitor 的静态方法，清除所有连接的设备

```
1 | public static synchronized void clearAll();
```

3.3.4 BleMonitor.remove

BleMonitor 的静态方法，清除某一个连接的设备

```
1 | public static synchronized void remove(String deviceAddress);
```

3.3.5 BleMonitor.bindStateListener

BleMonitor的静态方法, 注册设备状态处理类

```
1 | public static void bindStateListener(BleStateListener tl);
```

3.3.6 [BleMonitor]getName

从**BleMonitor** 继承而来,获取设备名称

```
1 | public String getName();
```

3.3.7 [BleMonitor]getAddress

从**BleMonitor** 继承而来,获取设备地址

```
1 | public String getAddress();
```

3.3.8 [BleMonitor]start

从**BleMonitor** 继承而来,启动设备监听

```
1 | public boolean start();
```

3.3.9 [BleMonitor]stop

从**BleMonitor** 继承而来,停止设备监听

```
1 | public void stop();
```

3.3.10 [FUDevice]parseRecord

解析一条标签记录, 记录格式 "TYPE=DATA", 例如 "FDXB=00112233445566"

```
1 public static Tag parseRecord(String str);
2 public static Tag parseRecord(String strType, String strData);
```

说明: strType 支持 FDXB、FDXB-S、HDX、HDX-S、FDXM、FDXM-S、ID64、EPC、BARCODE、BARCODE-S, 未识别的类型将生成 NomalTag 对象; 另兼容协议文档中的拼写 "BARCOARD"

3.3.11 [BleMonitor]destroy

从BleMonitor 继承而来,销毁设备监听器并关闭底层连接 (protected, SDK 内部使用)

```
1 protected void destroy();
```

3.3.12 [FUDevice]queryAlertTagsCount

查询预警号码数量, 同步等待设备响应

```
1 public int queryAlertTagsCount() throws DeviceException;
```

- 返回当前预警号码数量
- 超时或设备返回错误时抛出 DeviceException

3.3.13 [FUDevice]getAlertTag

获取指定索引的预警号码, 同步等待设备响应

```
1 public String getAlertTag(int index) throws DeviceException;
```

- index 索引, 从 0 开始
- 返回标签编号 HEX 字符串

3.3.14 [FUDevice]addAlertTag

添加预警号码, 同步等待设备响应

```
1 public int addAlertTag(String hex) throws DeviceException;
```

- hex 标签编号 HEX 字符串, 例如 "32333435" (对应 ASCII "2345")
- 返回添加后的预警号码总数量

3.3.15 [FUDevice]clearAlertTags

清空预警号码, 同步等待设备响应

```
1 public void clearAlertTags() throws DeviceException;
```

说明: 以上预警标签接口为同步交互模式, 会阻塞当前线程等待设备响应, 请勿在主线程直接调用。

3.3.16 [FUDevice]showDialog

显示远程对话框（指令 4120），异步交互：设备收到指令后立即应答对话框 id，不等待用户按键；用户按键后通过消息通道（0003CDD1）发送 "：DIALOG={对话框id},KEY={Y/N}" 通知，该消息作为 DIALOG 类型标签由 TagListener 接收（tagType="DIALOG"，id 为对话框 id，属性 KEY 为 "Y" 确认 / "N" 取消）。

```
1 public String showDialog(String title, String cancelButton, String
   confirmButton,
2                               String content1, String content2, String content3)
   throws DeviceException;
```

- title 标题（SDK 自动转成 UTF-16BE HEX 发送，居中显示在屏幕最上方）
- cancelButton 取消按钮提示（为空表示无特殊提示，分隔符保留）
- confirmButton 确认按钮提示（为空表示无特殊提示，分隔符保留）
- content1/content2/content3 内容第一/二/三行（每行最多 20 个英文字符长度）
- 返回设备分配的对话框 id
- 设备返回错误时抛出 DeviceException

说明：以上远程显示接口为异步交互模式，发送后立即返回对话框 id，不会阻塞等待用户按键；按键结果通过 TagListener 以 DIALOG 标签回调获取。

3.4 Tag

代表所有标签类的基类,实际的标签类型有多种,如

FDXB,FDXM,HDX,ID64,EPC,NomalTag,Barcode,UnknownTag，都是从Tag继承而来

3.4.1 getId

返回标签编号

```
1 public String getId();
```

3.4.2 getTagType

返回标签类型

```
1 public String getTagType();
```

3.4.3 getAttr

返回标签某些特殊属性，譬如某些设备能读取温度, 那么就有温度属性"T"(绝对温标)

```
1 public Object getAttr(String attrId)
```

3.4.4 hasAttr

判断标签是否包含某个特殊属性

```
1 public boolean hasAttr(String attrId);
```

3.4.5 parseAttr

解析并写入一条属性, 例如 "T=300.15", 解析成功后可通过 getAttr("T") 取得

```
1 public void parseAttr(String attrMsg);
```

3.4.6 putAttr

写入一个属性

```
1 public void putAttr(String attrId, Object value);
```