链滴

Android 低功耗 Ble 蓝牙 4.0 多连接 开源框架

作者：pencilso
原文链接：https://ld246.com/article/1501311180977
来源网站：链滴

ManyBlue

最近在开发Ble的项目，自己也在用这个 有发现bug会第一时间修复提交更新

如果有好的建议 可以邮件联系我 admin@javac.io

文档以Github上的为准，博客不会经常更新文章。

Github仓库地址 https://github.com/pencilso/ManyBlue

添加依赖 compile 'io.javac:ManyBlue:1.0.5'

依赖内部维护了一个Service 所以需要优先检测Service

ManyBlue.runing(Context context)

- 启动服务　ManyBlue.blueStartService(Context context);
- 关闭服务　ManyBlue.blueStopService(this);
- 蓝牙开启状态　ManyBlue.blueEnableState();
- 打开手机蓝牙　ManyBlue.blueEnable(true);
- 关闭手机蓝牙　ManyBlue.blueEnable(false);

注册|取消 事件

```
建议新建一个BaseActivity  然后继承自你现有的BaseActivity
然后重写onStart onStop 进行取消 和注册事件 跟处理监听事件
 @Override
protected void onStart() {
    super.onStart();
    EventManager.getLibraryEvent().register(this);//注册
}

@Override
protected void onStop() {
    super.onStop();
    EventManager.getLibraryEvent().unregister(this);//取消
}


//订阅消息
@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(NotifyMessage notifyMessage) {
    LogUtils.log(notifyMessage);
    if (this instanceof BaseNotifyListener)
        ManyBlue.dealtListener((BaseNotifyListener) this, notifyMessage);//处理监听
}
```

手机蓝牙的监听

实现接口 BaseNotifyListener.MobileBlueListener

```
@Override
public void onMobileBlueState(boolean enabled) {
    appToast("蓝牙开启状态:" + enabled);
}

@Override
public void onMobileBlueEnabled(boolean success) {
    appToast("开启手机蓝牙:" + success);
}

@Override
public void onMobileBlueDisable(boolean success) {
    appToast("关闭手机蓝牙:" + success);
}
```

蓝牙服务的监听

实现接口 BaseNotifyListener.ServiceListener

```
`@Override
public void onServiceStart() {
    appToast("蓝牙服务已开启");
}

@Override
public void onServiceStop() {
    appToast("蓝牙服务已关闭");
}`
```

扫描设备

- 扫描蓝牙　ManyBlue.blueStartScaner();
- 停止扫描　ManyBlue.blueStopScaner();
- 连接设备　ManyBlue.blueConnectDevice(String address, Object tag);//tag是自定义的标记来标记多设备

实现接口 BaseNotifyListener.DeviceListener

```
`/**
 * 扫描到蓝牙设备
 *
 * @param device
 */
@Override
public void onDeviceScanner(BluetoothDevice device) {
    adapter.addDevice(device);
}
```

```
/**
 * 蓝牙设备连接或者断开
 *
 * @param state true为连接 false为断开
 */
@Override
public void onDeviceConnectState(boolean state, Object tag) {
    if (!state) {
        appToast("连接失败");
        dismissDialog();
    } else setDialog("连接成功 正在发现服务");
}

@Override
public void onDeviceServiceDiscover(List services, Object tag) {
    setDialog("正在注册服务");
    //services 这是该设备中所有的服务 在这里找到需要的服务 然后再进行注册
//                services.get(0).getUuid().toString();//这是获取UUID的方法
    //找到需要的UUID服务  然后进行连接  比如说我需要的服务UUID是00003f00-0000-1000-8000-0805f9b34fb UUID的话  一般设备厂家会提供文档 都有写的
    UUIDMessage uuidMessage = new UUIDMessage();//创建UUID的配置类
    uuidMessage.setCharac_uuid_service("00003f00-0000-1000-8000-00805f9b34fb");//需要注的服务UUID
    uuidMessage.setCharac_uuid_write("00003f02-0000-1000-8000-00805f9b34fb");//写出数据通道UUID
    uuidMessage.setCharac_uuid_read("00003f01-0000-1000-8000-00805f9b34fb");//读取通道UUID
    uuidMessage.setDescriptor_uuid_notify("00002902-0000-1000-8000-00805f9b34fb");//这读取通道当中的notify通知
    /**
     * 这里简单说一下  如果设备返回数据的方式不是Notify的话  那就意味着向设备写出数据之后   再己去获取数据
     * Notify的话 是如果蓝牙设备有数据传递过来  能接受到通知
     * 使用场景中如果没有notify的话  notify uuid留空即可
     */
    ManyBlue.blueRegisterDevice(uuidMessage, tag);//注册设备
}

@Override
public void onDeviceRegister(boolean state) {
    dismissDialog();
    appToast(state ? "设备注册成功" : "设备注册失败");
}

`
```

监听所有回调

实现接口 BaseNotifyListener.NotifyListener

发送|接收 蓝牙数据

- 获取已连接设备　ManyBlue.getConnDeviceAll();
- 实现接口　BaseNotifyListener.DeviceDataListener
- 发送字符转十六进制 ManyBlue.blueWriteDataStr2Hex(str, tag); //例如 0x0a0a01

回调事件

```
/**
 * 向蓝牙发送数据后的回调
 *
 * @param state 发送成功true  发送失败false
 */
@Override
public void onDeviceWriteState(boolean state, Object tag) {
    dismissDialog();//关闭Dialog
    appToast("指令发送状态:" + state);
    /**
     * 如果是非Notify的接收方式的话 这里需要手动去调用读取通道
     */
    //ManyBlue.blueReadData(tag); //主动获取数据
}

/**
 * 主动读取的通道数据
 *
 * @param characteristicValues 读取到的数据
 */
@Override
public void onDeviceReadMessage(CharacteristicValues characteristicValues) {
    LogUtils.log("onDeviceReadMessage   strValue:" + characteristicValues.getStrValue() + " he
2Str:" + characteristicValues.getHex2Str() + " byArr:" + characteristicValues.getByArr());
}

/**
 * Notify监听收到的数据
 *
 * @param characteristicValues 读取到的数据
 */
@Override
public void onDeviceNotifyMessage(CharacteristicValues characteristicValues) {
    LogUtils.log("onDeviceNotifyMessage    strValue:" + characteristicValues.getStrValue() + "
ex2Str:" + characteristicValues.getHex2Str() + " byArr:" + characteristicValues.getByArr());
}
```

已连接设备

- 获取所有已连接设备　ManyBlue.getConnDeviceAll()
- 获取指定标识设备　ManyBlue.getConnDevice(tag);
- 断开指定设备　ManyBlue.blueDisconnectedDevice(tag);
- 断开所有设备　ManyBlue.blueDisconnectedDeviceAll();