**1.简介**

Android中为蓝牙的使用提供了一整套API，可以进行如下操作：

* 搜索其他蓝牙设备
* 查询适配设备
* 建立RFCOMM通道
* 通过发现服务连接其他蓝牙设备
* 交换数据
* 管理多项连接

**2.集成使用**

**第一步：声明所需要的权限**

<uses-permission android:name="android.permission.BLUETOOTH"/> 使用蓝牙所需要的权限

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/> 使用扫描和设置蓝牙的权限（申明这一个权限必须申明上面一个权限）

在Android5.0之前，是默认申请GPS硬件功能的。而在Android 5.0 之后，需要在manifest 中申明GPS硬件模块功能的使用。

<!-- Needed only if your app targets Android 5.0 (API level 21) or higher. -->

<uses-feature android:name="android.hardware.location.gps" />

在 Android 6.0 及以上，还需要打开位置权限。如果应用没有位置权限，蓝牙扫描功能不能使用（其它蓝牙操作例如连接蓝牙设备和写入数据不受影响）。

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>

**第二步：初始化实例**  
在页面首先初始化一个BlueManager。

private BlueManager bluemanage;

bluemanage = BlueManager.getInstance(getApplicationContext());

**第三步：设置实例监听**

然后为这个蓝牙管理器设置监听(OnSearchDeviceListener，OnConnectListener，OnSendMessageListener，OnReceiveMessageListener)

/\*\*

\* 初始化蓝牙管理，设置监听

\*/

public void initBlueManager() {

bluemanage = BlueManager.getInstance(getApplicationContext());

bluemanage.setOnSearchDeviceListener(onSearchDeviceListener);

bluemanage.setOnConnectListener(onConnectListener);

bluemanage.setOnSendMessageListener(onSendMessageListener);

bluemanage.setOnReceiveMessageListener(onReceiveMessageListener);

bluemanage.requestEnableBt();

}

**第四步：开启蓝牙搜索蓝牙设备**

通过调用 bluemanage.requestEnableBt()开启蓝牙，调用searchDevices 获取蓝牙设备。在做蓝牙操作前，要确保各个监听器已经设置好。

搜索监听如下：

onSearchDeviceListener =new OnSearchDeviceListener() {

@Override

public void onStartDiscovery() {

Log.d(TAG, "onStartDiscovery()");

}

@Override

public void onNewDeviceFound(BluetoothDevice device) {

Log.d(TAG, "new device: " + device.getName() + " " + device.getAddress());

}

@Override

public void onSearchCompleted(List<BluetoothDevice> bondedList, List<BluetoothDevice> newList) {

Log.d(TAG, "SearchCompleted: bondedList" + bondedList.toString());

Log.d(TAG, "SearchCompleted: newList" + newList.toString());

}

@Override

public void onError(Exception e) {

e.printStackTrace();

}

}

通过 BlueManager里的searchDevices方法，里边其实就是获取了一个BluetoothAdapter然后，通过调用mBluetoothAda pter.startDiscovery()方法来搜索经典蓝牙设备。这里如果调用 mBluetoothAdapter.startLeScan(mLeScanCallback); 搜索的就是BLE蓝牙。然后在这之前需要动态注册一个BroadcastReceiver来监听 蓝牙的搜索情况，在通过onReceive中去判 断设备的类型，是不是新设备，是不是已经连接过。将设备加入集合当中。

搜索代码如下

/\*\*

\* discovery the devices.

\*/

public void searchDevices() {

try {

if (mCurrStatus == STATUS.FREE) {

mCurrStatus = STATUS.DISCOVERING;

checkNotNull(mOnSearchDeviceListener);

if (mBondedList == null) mBondedList = new ArrayList<>();

if (mNewList == null) mNewList = new ArrayList<>();

if (mBluetoothAdapter == null) {

mOnSearchDeviceListener.onError(new NullPointerException(DEVICE\_HAS\_NOT\_BLUETOOTH\_MODULE));

return;

}

if (mReceiver == null) mReceiver = new Receiver();

// ACTION\_FOUND

IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION\_FOUND);

mContext.registerReceiver(mReceiver, filter);

// ACTION\_DISCOVERY\_FINISHED

filter = new IntentFilter(BluetoothAdapter.ACTION\_DISCOVERY\_FINISHED);

mContext.registerReceiver(mReceiver, filter);

mNeed2unRegister = true;

mBondedList.clear();

mNewList.clear();

if (mBluetoothAdapter.isDiscovering()) //先判断是否在扫描

mBluetoothAdapter.cancelDiscovery();//取消扫描

mBluetoothAdapter.startDiscovery(); //开始扫描蓝牙

mOnSearchDeviceListener.onStartDiscovery();

}

} catch (Exception e) {

e.printStackTrace();

}

}

**第五步：连接蓝牙设备**

搜索的大致流程就是这样子。接下来说下配对连接流程。当调用connectDevice(mac)方法时，因为连接蓝牙是一个 很耗时的操作，所以需要开启一个线程去连接蓝牙。

/\*\*

\* 连接bluetooth

\*

\* @param mac

\*/

public void connectDevice(String mac) {

try {

if (mCurrStatus != STATUS.CONNECTED) {

if (mac == null || TextUtils.isEmpty(mac))

throw new IllegalArgumentException("mac address is null or empty!");

if (!BluetoothAdapter.checkBluetoothAddress(mac))

throw new IllegalArgumentException("mac address is not correct! make sure it's upper case!");

if (mReadable = false) {

mReadable = true;

}

if (mWritable = false) {

mWritable = true;

}

if (onConnectListener != null) {

onConnectListener.onConnectStart();

ConnectDeviceRunnable connectDeviceRunnable = new ConnectDeviceRunnable(mac);

checkNotNull(mExecutorService);

mExecutorService.submit(connectDeviceRunnable);

}

} else {

Log.i("blue", "the blue is connected !");

}

} catch (IllegalArgumentException e) {

e.printStackTrace();

}

}

在连接的线程run方法中，通过调用mBluetoothAdapter.getRemoteDevice 获取远程蓝牙信息，通过 createInsecureRfcommSocketToServiceRecord获得一个与远程蓝牙的socket连接。通过这个socket连接获取输入 流和输出流进行数据的读写。

if (onConnectListener == null) {

Log.i("blue", "the connectListener is null !");

return;

}

BluetoothDevice remoteDevice = mBluetoothAdapter.getRemoteDevice(mac);

mBluetoothAdapter.cancelDiscovery();

mCurrStatus = STATUS.FREE;

Log.d(TAG, "prepare to connect: " + remoteDevice.getAddress() + " " + remoteDevice.getName());

mSocket = remoteDevice.createInsecureRfcommSocketToServiceRecord(UUID.fromString(Constants.STR\_UUID));

onConnectListener.onConnectting();

mSocket.connect();

mInputStream = mSocket.getInputStream();

mOutputStream = mSocket.getOutputStream();

mCurrStatus = STATUS.CONNECTED;

onConnectListener.onConectSuccess();

**第六步：向蓝牙设备发送消息**

当设备连接成功之后，就可以给蓝牙设备发送消息了。 通过调用bluemanage.sendMessage(MessageBean mesaage， needResponse)方法，在bluemange里会开起一个WriteRunnable写线程和一个ReadRunnable去获取输入流和输出流 的实时数据，读线程只会在第一次发消息时初始化一次。以后都是用这个线程去读从蓝牙返回的数据。写数据的线程 在每次调用的时候都会从新初始化。(待优化)

在WriteRunnable中的润写数据

writer.write(item.text);

writer.newLine();

writer.flush();

在WriteRunnable 的run方法中通过mOutputStream流将数据传送给蓝牙设备,当蓝牙接受到消息之后会和串口进行 通信，具体的通信协议是根据各个厂商自己协商的。当串口接受数据执行操作，获取数据然后在返回数据给蓝牙，蓝牙也就有返回数据，在ReadRunnable中从mInputStream里不断的读取数据。

**第七步：从蓝牙设备读取消息**

读数据不完整，是因为我们开启线程之后会一直读，有时候蓝牙并没有返回数据，或者没有返回完整数据，这个时候

我们需要在这做一些特殊处理。

int count = 0;

while (count == 0) {

count = stream.available();//输入流中的数据个数。

}

通过以上代码可以确保读的数据不会是0。通过下边的代码可以确保读到完整数据之后才会走我的回调，保证了数据

的完整性。这里的what只是我用来区分当前读到的数据是进度信息，还是真正想要的信息。

if (onReceiveMessageListener == null) {

Log.i("blue", "the receiverMessageListener is null !");

return;

}

mReadable = true;

InputStream stream = mInputStream;

while (mCurrStatus != STATUS.CONNECTED && mReadable) ;

checkNotNull(stream);

byte[] buffer = new byte[DEFAULT\_BUFFER\_SIZE];

StringBuilder builder = new StringBuilder();

while (mReadable) {

int count = 0;

while (count == 0) {

count = stream.available();//输入流中的数据个数。

}

if (count == 10 && what) {

int num = stream.read(buffer);

String progress = TypeConversion.bytesToHexStrings(buffer);

Log.i("progress", progress);

onReceiveMessageListener.onProgressUpdate(progress, 0);

} else if (count >= 10) {

what = false;

int num = stream.read(buffer);

String detect = TypeConversion.bytesToHexStrings(buffer);

builder.append(detect);

Log.i("detect", detect);

if (detect.endsWith("04 ")) {

number++;

}

if (number == 5) {

onReceiveMessageListener.onDetectDataFinish();

onReceiveMessageListener.onNewLine(builder.toString().trim());

builder.delete(0, builder.length());

} else {

onReceiveMessageListener.onDetectDataUpdate(detect);

}

}

}

当读到满足条件的完整数据，就会调用ReceiveMessageListener 中的各个方法。到这里从蓝牙读取数据的流程，

# 下边是BlueManager提供的一些方法：

requestEnableBt() 开启蓝牙

searchDevices() 搜索蓝牙设备

connectDevice() 连接蓝牙设备

closeDevice() 断开蓝牙连接

sendMessage() 发送消息

close() 关闭销毁蓝牙