我们终于把 51 单片机和安卓应用开发的基础学完了，想必那些心急一点的同学已经提前跳到这里了，但是笔者建议还是要打好基础的！那么从本节开始，我们将正式进入实战阶段。为了让大家循序渐进搭建起自己的智能家居系统，笔者想先用几个难度逐渐增加的例子考验一下大家。话不多说，让我们来看第一个考验——自制智能蓝牙防丢器。

1 什么是智能蓝牙防丢器

所谓智能蓝牙（Smart Bluetooth）防丢器，是采用蓝牙技术专门为智能手机设计的防丢器。其工作原理主要是通过距离变化来判断物品是否还控制在你的安全范围。主要适用于手机、钱包、钥匙、行李等贵重物品的防丢，也可用于防止儿童或宠物的走失[①]。

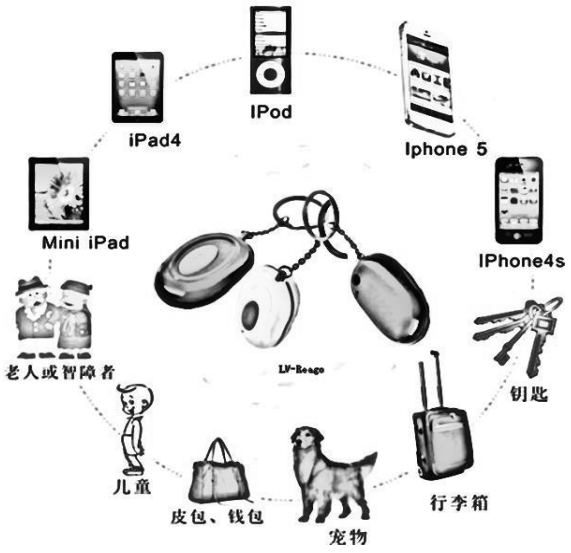


图 1-1 蓝牙防丢器应用领域

2 蓝牙防丢器的主要构造

目前比较成熟的产品一般是采用蓝牙 4.0 技术，具有低功耗、双向防丢、自动报警等优点。虽然市场上该类产品种类繁多、层出不穷，但其核心构成一般包括：蓝牙 4.0 芯片、蓝牙芯片辅助电路、蓝牙天线、蜂鸣器、开关、电源等。

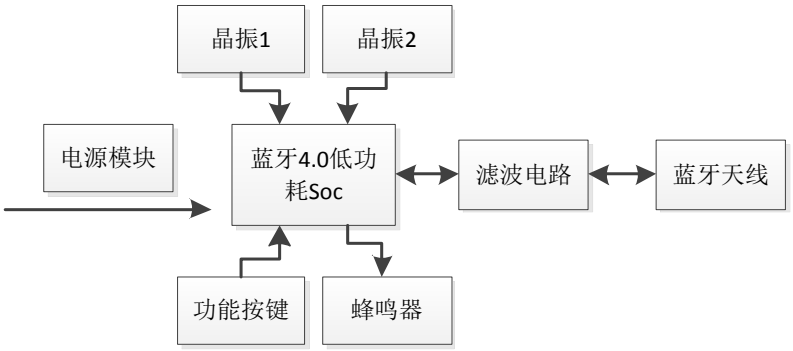


图 2-1 蓝牙防丢器构成

① 参考百度百科——“蓝牙 4.0 防丢器”。词条中介绍的是用蓝牙 4.0 技术实现的方案，笔者觉得这里直接拿 4.0 介绍，难度可能会有点大，所以就挑稍微简单一点的 HC-05/06 蓝牙芯片讲解了。

3 蓝牙模块的选择

由于这是第一个考验，笔者可不想一下子把大家给吓倒了。所以这里我们先用一个相对简单但常用的蓝牙模块 HC-05/HC-06 进行 DIY [①]。如下图该模块把天线、滤波电路、Soc、晶振都集成在了一起，当我们用的时候只要关注 1、2、12、13、24、26 这几个引脚就能实现比较完整的蓝牙通信功能，这样就为我们制作蓝牙防丢器节省了很多挑选元件、设计电路、焊接制板的功夫，是不是超赞呀？

图 3-1 蓝牙模块 1

其实还有更赞的呢！由于考虑到很多读者在硬件方面还都是新手，初次拿到邮票边缘式引脚的模块会焊接不好，于是笔者又找到了一款封装更好的蓝牙模块（其实就是把上面的模块加一个托，然后将 VCC\GND\TXD\RXD 四个关键的引脚引出）。当我们只是想把蓝牙模块作为标签时，只要在 VCC 和 GND 之间给它加上相应的电压就行了；当想用它进行无线数据传输时，这时 TXD 和 RXD 两个引脚就起作用了。

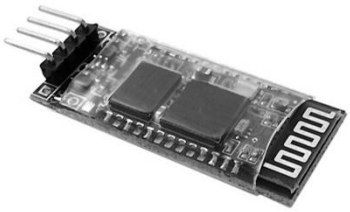


图 3-2 蓝牙模块 2

4 开始制作一个简易的蓝牙防丢器

上面说了这么多了，那么我们的蓝牙防丢器的设计方案到底是什么样的呢？简单起见，咱么仅仅实现通过距离变化来判断物品是否还控制在你的安全范围内的具有核心功能的防丢器，对于节能功能、双向报警功能甚至是自拍功能咱们就先不考虑了 [②]。哈哈，可能说到这里大家还是对咱们要做的防丢器一点想法都没有，其实通过上面的铺垫笔者有信心大家可以在一分钟之内知道怎么完成它！

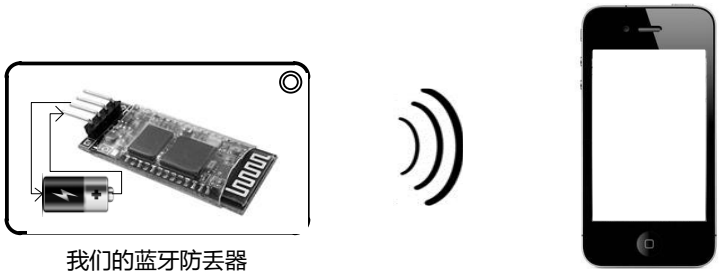


图 4-1 简易蓝牙防丢器

① HC-05 或 HC-06 蓝牙串口模块不同于上面讲的蓝牙 4.0 模块，常见的蓝牙 4.0 模块有 CC2540 和 CC2541,这些大家都可以在淘宝上比较方便地买到。由于考虑到 CC2540 和 CC2541 开发稍微有一定难度，所以挑了 HC 系列的讲解。

② 笔者觉得自拍功能是个比较有趣而且简单的设计，其实就是通过蓝牙的通信功能向智能手机相应的应用发送一个拍照的信号，然后该应用调用相关函数进行拍照，当大家学完下一篇之后可以尝试对本节制作的蓝牙防丢器进行功能扩展。

相信很多看完上面图片同学会恍然大悟——不是嘛，只要将蓝牙模块接上相应的电源模块就能够做成一个简单的可以发出蓝牙信号的防丢器了！对的，由于我们没有加入复杂的通信功能，所以我们仅仅把蓝牙模块通上电做成一个蓝牙标签就可以了。但是大家不要高兴太早，虽然是第一个测试，笔者也不会这么轻易放大家过关的（没发现上面图片中手机屏幕里的应用还是一片空白吗？哈哈）。

5 如何找到并学习要用到的 API

上面制作蓝牙防丢器的硬件部分让大家觉得没什么挑战性，那么接下来的东西可就有一定难度了！记得笔者当时学安卓蓝牙开发的时候费了好大力气的。这里笔者强烈建议大家着手了解安卓某个功能的应用时最好去安卓开发者社区，但是随着 Google 被屏 Android Developer 也不能被访问了。虽然笔者在 MIT 网站上又找到了一个类似的网页，但是访问速度不是那么流畅，为了方便，笔者挑出了和本节相关的知识帮助大家理解和运用。

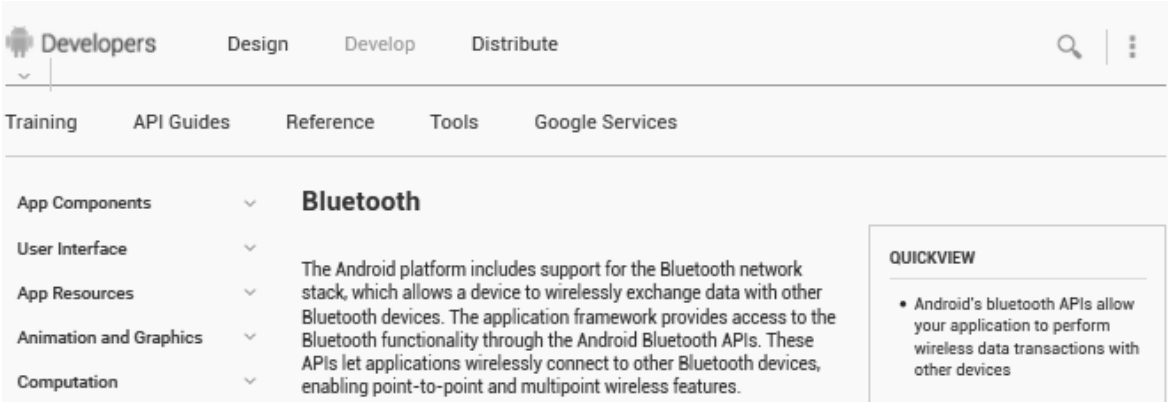


图 5-1 Android Developer 页面

6 安卓蓝牙编程小知识

安卓平台支持蓝牙协议栈，允许一台设备和其他设备进行无线数据交换，当大家想使用蓝牙时可以通过调用相应的 API 实现功能。这些 API 提供的主要功能有：

- 扫描搜索其他蓝牙设备(Scan for other Bluetooth devices)
- 查询本地蓝牙适配器配对的蓝牙设备(Query the local Bluetooth adapter for paired Bluetooth devices)
- 建立 RFCOMM 通道(Establish RFCOMM channels)
- 连接其他设备(Connect to other devices through service discovery)
- 与其他设备进行数据交换(Transfer data to and from other devices)
- 管理多组连接(Manage multiple connections)

在准备在应用程序中使用蓝牙时，首先需要在 manifest 文件中包含 BLUETOOTH 和 BLUETOOTH_ADMIN 权限：

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

接着涉及蓝牙的操作主要有：1、开启蓝牙 2、关闭蓝牙 3、能被搜到 4、获取配对设备 5、传输数据。由于本节只涉及到搜索周边设备，所以配对并传输数据暂时不介绍。

7 安卓蓝牙编程主要操作

要想通过编程操作蓝牙设备，首先我们得了解一下有可供我们使用的相关类及其成员函数有哪些，如下图：蓝牙设备包括本地设备和远程设备，本地设备对应的类为 BluetoothAdapter，远程设备对应的类为 BluetoothDevice，两者的成员函数基本相同，笔者将在接下来详细讲解。

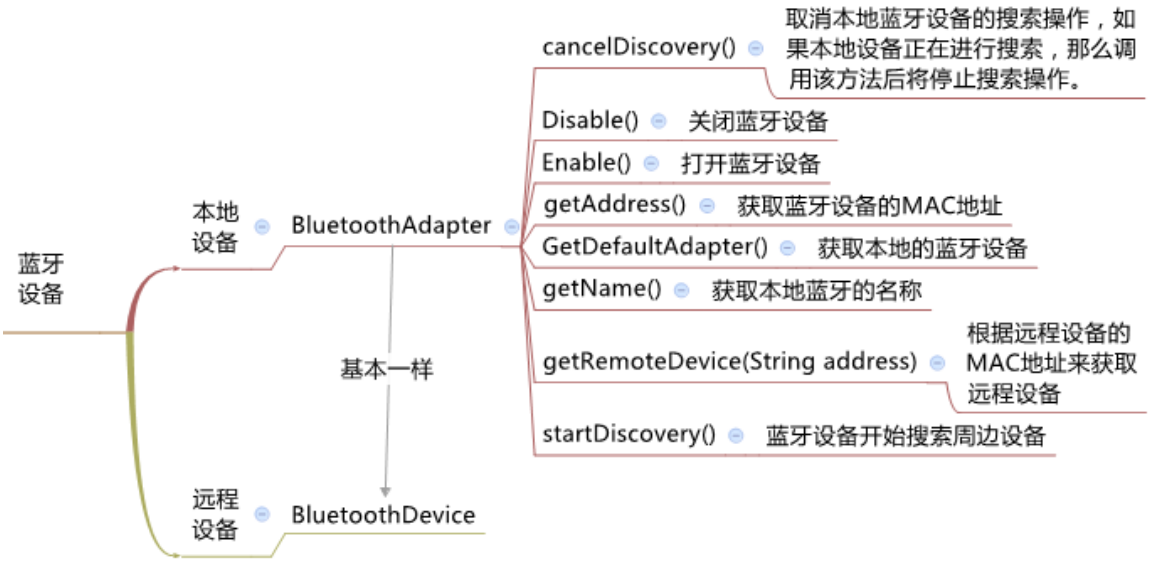


图 7-1 蓝牙设备相关函数

◆ 打开本地蓝牙设备

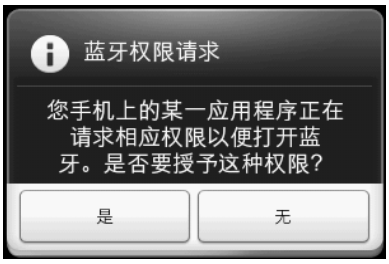
我们要做带有蓝牙的应用，首先要保证用户的本地蓝牙设备是打开的，否则什么都不管直接使用搜索、连接、通信等函数肯定会得到和预期不一样的效果。但是有时候用户为了节省电池电量会把蓝牙关闭，那我们该怎么办呢？

其实，遇到这种情况大家也不用担心！请看下面的解决方案：其中第 1 行调用 BluetoothAdapter 的 getDefaultAdapter()方法获得本地蓝牙设备，接着调用 isEnabled()判断本地蓝牙设备是否被打开，如果被打开了就执行接下来的操作，否则可以发送 Intent 消息，请求打开本地蓝牙设备，接着待用户打开蓝牙后再进入接下来的操作。

```
1   mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
2   if (!mBluetoothAdapter.isEnabled()) {
3       Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
4       startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
5   } else {
6       nextOperation();
7   }
```

这时候有些读者可能要吐槽笔者了“你上面发送、请求、接着、然后说的挺轻巧，我怎么知道我发送完 Intent 消息后系统到底干了什么？用户又如何打开蓝牙的？应用程序又在哪里等待用户完成打开蓝牙事件，然后在哪里执行 nextOperation()函数的？”哈哈，笔者知道大家动手心切啦！下面将给大家详细介绍这一过程。

想解答大家的这些问题还得看上面代码：其中第 4 行 startActIvityForResult 会启动一个系统 Preference Activity 并将 ACTION_REQUEST_ENABLE 静态常量作为其动作字符串，得到的 Preference Activity 如右图：



该 Activity 提醒用户是否授予权限打开本地蓝牙设备,当用户点击“是”或者“否”的时候，该 Activity 将会关闭。我们可以使用 onActivityResult 处理程序中返回的结果代码参数来确定是否操作成功。

正如上面介绍，当用户点击“是”授予蓝牙权限请求后，确认请求的 Activity 会关闭，在下面的函数中将会收到此操作的消息，这样我们就能很潇洒的在第 4 行安全的进入接下来的操作了。

```
1   protected void onActivityResult(int requestCode,int resultCode,Intent data){
2       if(requestCode==ENABLE_BLUETOOTH){
3           if(resultCode==RESULT_OK){
4               nextOperation();
5           }
6       }
7   }
```

◆ 搜索周边蓝牙设备

上面解决了蓝牙设备打开问题，接下来我们就要尝试调用相关函数搜索周边的蓝牙设备，这样我们就能知道我们制作的蓝牙防丢器是否在我们周边了（是不是很兴奋呢？）[①]。

这里我们要用到 BluetoothAdapter 的成员函数 startDiscovery，该方法可以执行一个异步方式获得周边蓝牙设备,因为是一个异步的方法所以我们不需要考虑线程被阻塞问题,整个过程大约需要 12 秒时间。这时我们可以注册一个 BroadcastReceiver 对象来接收查找到的蓝牙设备信息，我们通过 Filter 来过滤 ACTION_FOUND 这个 Intent 动作以获取每个远程设备的详细信息，过滤 ACTION_DISCOVERY_FINISHED 这个 Intent 动作以获取整个蓝牙搜索过程结束的信息。

```
1   // Register for broadcasts when a device is discovered
2   IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
3   this.registerReceiver(mReceiver, filter);
4   // Register for broadcasts when discovery has finished
5   filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
6   this.registerReceiver(mReceiver, filter);
```

上面讲的可能有点专业，下面笔者用一种简单非专业的方式向大家介绍一下：每次我们想编程搜索周边的蓝牙设备时，只要简单调用 BluetoothAdapter 的成员函数 startDiscovery 就可以了。所谓的异步方法大家可以这样理解——start 方法就好像一个总司令告诉情报搜查员去搜查情报，然后自己继续做自己的事，情报员去收集各种情报；这里的 filter 就好像总司令告诉情报搜查员，我只要 ACTION_FOUND 和 ACTION_DISCOVERY_FINISHED 信息；那么这里的 mReceiver 就是总司令指定的情报搜查员了。

① 其实蓝牙室内导航或者是 wifi 室内导航也和这个原理类似，之所以选这个作为测试一，不仅仅在于其制作简单，更在于其中蕴含的技术具有一定的通用性，大家只要稍加转换思想，就能以此为基础 DIY 出更炫酷的东西。

接下来就让咱们神奇的情报搜查员登场！如下，相信大家一看就明白了，咱们的情报搜查员会一丝不苟地将总司令下达的任务执行。这样当他收到 FOUND 动作信息时就用第 6~8 行的方法将每个发现的蓝牙设备的名字和地址存储进一个 Vector 中，这样等自己完成任务时，就能告诉总司令任务完成，所有周边蓝牙情报都在 xxx 向量中（嘻嘻，多么让领导喜欢的员工呀！）。

```
1   private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2     @Override
3     public void onReceive(Context context, Intent intent) {
4         String action = intent.getAction();
5         if (BluetoothDevice.ACTION_FOUND.equals(action)) {
6             BluetoothDevice device =
7                         intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
8             mDevicesVector.add(device.getName() + "\n" + device.getAddress());
9         } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
10            //...
11        }
12    }
13  };
```

◆ 获取蓝牙设备信号强度

到目前想必很多人已经能够一气呵成自己的简单蓝牙防丢器了，因为大家已经掌握了硬件部分的设计方法、拥有软件蓝牙开发打开和搜索周边蓝牙的编程技巧。但是笔者也能肯定如果心急的同学只用前面介绍的一点小知识的话，肯定设计的小东西很不尽人意（自己都不好意思给自己及格）。是不是出现了蓝牙防丢器放到很远很远时应用程序才会报告东西已丢？如果是这样请耐心地看完下面的介绍（也许能给你更多的启发(⊙o⊙)哦）。

想必大家也想到了问题所在——距离没控制好！如果我们能够知道蓝牙防丢器距离我们手机的距离那就好了，这样我们就能设定一个范围值，当它超出这个范围时手机就发出丢失警告，这样就不会出现防丢器要放很远才能被手机察觉已丢失的问题了。要解决这个问题笔者就要向大家介绍一下无线传感器网络中的 RSSI 了！

RSSI 全名 Received Signal Strength Indication，翻译过来是接收的信号强度指示。在我们的经验中一般离得越近信号越强，所以通过这个 RSSI 能够大致估计接收点与发射点之间的距离。而在无线传感器网络中经常会设置多个固定的发射源（也是所谓的标签），然后根据一定的算法来确定移动目标的空间位置信息。例如下图左分别在 A、B、C 三点放置三个发射功率相同的信号源（标签），这样移动点 D 通过收集 A、B、C 三点的 RSSI 并估计距离 r_A、r_B、r_C，由于这个距离并不是完全精准，所以三个圆并不一定交于一点，大多数情况是下面交于一个公共区域的情况，而移动点 D 的当前位置很有可能在该区域。

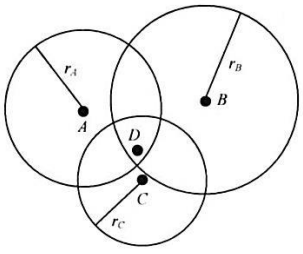


图 7-2 蓝牙标签定位

我们这里只要简单地用到通过 RSSI 估计手机和防丢器之间的距离就行了,上面的定位技术相信大家也能举一反三轻松搞定（\(^o^)/~其实没那么简单，哈哈)。那么在安卓编程时如何获得 RSSI 呢？其实并不难，我们可以利用和获取周边蓝牙设备的名称和地址类似的方法获取 RSSI:

```
1   BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
2   mDevicesVector.add(device.getName() + "\n" + device.getAddress());
3   short rssi = intent.getExtras().getShort(BluetoothDevice.EXTRA_RSSI);
4   mRSSIVector.add(rssi);
```

8 着手开发我们的 APP

1） 使用 Eclipse 创建一个安卓项目,命名为 first_test,并将 Activity Name 命名为 UI_Main,Layout Name 命名为 ui_main（如图 8-1 所示)。

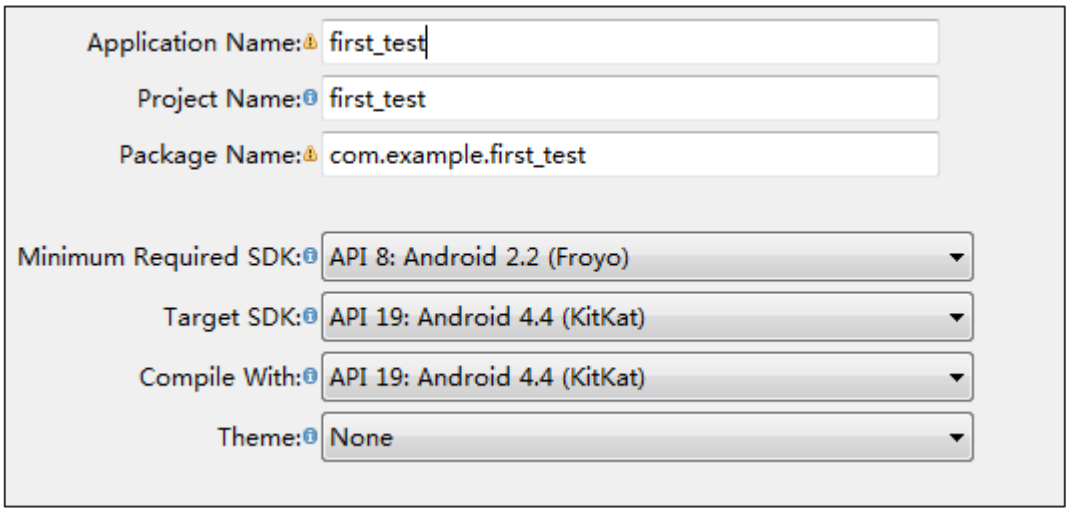


图 8-1 新建安卓项目

2） 展开 res 文件夹下的 layout 文件夹，双击 ui_main.xml 文件，选择 xml 编辑模式，并将其代码改为：

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="fill_parent"
4       android:layout_height="fill_parent"
5       android:orientation="vertical" >
6
7       <SurfaceView
8           android:id="@+id/surfaceView"
9           android:layout_width="match_parent"
10          android:layout_height="wrap_content"
11          android:layout_weight="0.75" />
```

```
12
13      <Button
14          android:id="@+id/button_add"
15          android:layout_width="match_parent"
16          android:layout_height="wrap_content"
17          android:text="添加蓝牙防丢器" />
18
19      <Button
20          android:id="@+id/button_start"
21          android:layout_width="match_parent"
22          android:layout_height="wrap_content"
23          android:text="开始防丢" />
24
25  </LinearLayout>
```

操作说明：这里修改的 ui_main.xml 为应用的主界面（如图 8-2 所示）。该页面采用 LinearLayout 布局模式，从上到下依次为用于动态显示蓝牙信号强弱的 SurfaceView 控件、用于添加防丢设备的按钮和用于开始防丢控制的按钮。

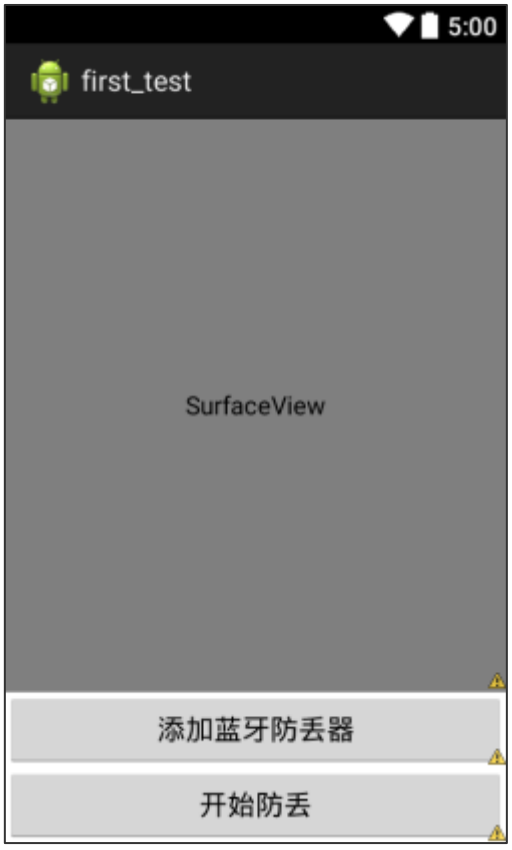


图 8-2 ui_main.xml 效果

3） 右击 layout，依次选择 New|Android XML File 新建安卓 XML 文件。

4） 将新建的 Android XML File 命名为 ui_list，点击 Finish 按钮。接着同第二步将新建的 ui_list.xml 修改为：

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
7      <ListView
8          android:id="@+id/listView1"
9          android:layout_width="match_parent"
10         android:layout_height="wrap_content"
11         android:layout_weight="1" >
12     </ListView>
13
14     <Button
15         android:id="@+id/button_search"
16         android:layout_width="match_parent"
17         android:layout_height="wrap_content"
18         android:text="search" />
19
20     <Button
21         android:id="@+id/button_ok"
22         android:layout_width="match_parent"
23         android:layout_height="wrap_content"
24         android:text="OK" />
25
26 </LinearLayout>
```

操作说明：这里新建的 ui_list.xml 为搜索并添加蓝牙防丢器界面。该界面同样采用 LinearLayout 布局模式，包含一个用于开始搜索的按钮、一个用于列出搜索结果的 list 和一个用于确认返回的 OK 按钮。

5） 右击 src 文件夹下的包名，依次选择 New|Class 命令（如图 8-3 所示）

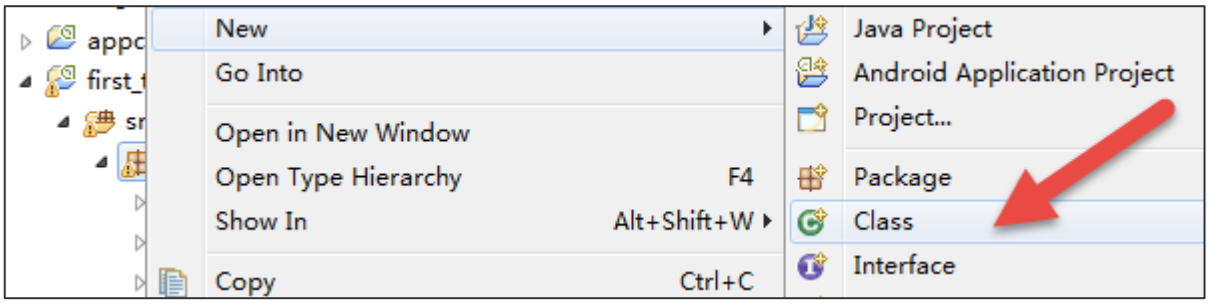


图 8-3 新建类

6） 新建类文件命名为 My_BTS，点击 Finish 按钮。

7） 将 My_BTS.java 文件修改为：

```
1  package com.example.first_test;
2
3  import java.util.Vector;
4
5  public class My_BTS {
6      public String mName;
7      public String mAddr;
8      public Vector<Short> mRSSIVector;
9
10     public My_BTS() {
11         mName = new String();
12         mAddr = new String();
13         mRSSIVector = new Vector<Short>();
14     }
15
16     public My_BTS(String name, String addr) {
17         mName = name;
18         mAddr = addr;
19         mRSSIVector = new Vector<Short>();
20     }
21 }
```

操作说明：该类表示蓝牙防丢器。其中 mName 和 mAddr 分别表示蓝牙防丢器的名字和地址；mSSIVector 用来存放一段时间检测到该蓝牙防丢器的 RSSI 值（之所以保留多组数据，是方便今后大家扩展）。

8） 采用同样的方法新建一个 Func_Draw.java 文件，并将文件修改为：

```
1  package com.example.first_test;
2
3  import java.util.Vector;
4
5  import android.graphics.Canvas;
6  import android.graphics.Color;
7  import android.graphics.Paint;
8  import android.graphics.Paint.Style;
9  import android.view.SurfaceHolder;
10
11 public class Func_Draw {
12     private static Vector<Paint> mPaint = new Vector<Paint>();
13     public static Integer times = 0;// 防丢搜索次数
14     public static float Bei = 200;// 绘制图形时放大倍数
15
16     public static void initPaint() {
```

```
17          Paint paint0 = new Paint();
18          paint0.setAntiAlias(true);
19          paint0.setStyle(Style.STROKE);
20          paint0.setColor(Color.RED);
21          mPaint.add(paint0);
22          Paint paint1 = new Paint();
23          paint1.setAntiAlias(true);
24          paint1.setStyle(Style.STROKE);
25          paint1.setColor(Color.GREEN);
26          mPaint.add(paint1);
27          Paint paint2 = new Paint();
28          paint2.setAntiAlias(true);
29          paint2.setStyle(Style.STROKE);
30          paint2.setColor(Color.BLUE);
31          mPaint.add(paint2);
32          Paint paint3 = new Paint();
33          paint3.setAntiAlias(true);
34          paint3.setStyle(Style.STROKE);
35          paint3.setColor(Color.YELLOW);
36          mPaint.add(paint3);
37          Paint paint4 = new Paint();
38          paint4.setAntiAlias(true);
39          paint4.setStyle(Style.STROKE);
40          paint4.setColor(Color.WHITE);
41          mPaint.add(paint4);
42          Paint paint5 = new Paint();
43          paint5.setAntiAlias(true);
44          paint5.setStyle(Style.STROKE);
45          paint5.setColor(Color.LTGRAY);
46          mPaint.add(paint5);
47          Paint paint6 = new Paint();
48          paint6.setAntiAlias(true);
49          paint6.setStyle(Style.STROKE);
50          paint6.setColor(Color.CYAN);
51          mPaint.add(paint6);
52      }
53
54      public static void draw(SurfaceHolder mHolder) {
55          Canvas canvas = mHolder.lockCanvas();
56          canvas.drawRGB(0, 0, 0);
57          for (int i = 0; i < UI_Main.mBTSArrayList.size(); i++) {
58              boolean find = false;
59              short rssi = 0;
60              for (int j = 0; j < UI_Main.mFuncBT.mAddrVector.size(); j++) {
```

```
61                  if (UI_Main.mBTSArrayList.get(i).mAddr
62                          .equals(UI_Main.mFuncBT.mAddrVector.get(j))) {
63                      find = true;
64                      rssi = UI_Main.mFuncBT.mRSSIVector.get(j);
65                  }
66              }
67              if (find == false) {
68                  canvas.drawText(
69                          times + ": NOT_FIND "
70                                  + UI_Main.mBTSArrayList.get(i).mName, 5,
71                          i * 10 + 12, mPaint.get(i));
72              } else {
73                  float power = (float) ((Math.abs(rssi) - 59) / (10 * 2.0));
74                  float dis = (float) Math.pow(10, power);
75
76                  canvas.drawText(
77                          times + ": FIND " + UI_Main.mBTSArrayList.get(i).mName
78                                  + " dis: " + new Float(dis).toString()
79                                  + " rssi: " + rssi, 5, i * 10 + 12,
80                          mPaint.get(i));
81                  canvas.drawCircle(canvas.getWidth() / 2,
82                          canvas.getHeight() / 2, Bei * dis, mPaint.get(i));//画圆圈
83              }
84          }
85          times++;
86          mHolder.unlockCanvasAndPost(canvas);// 更新屏幕显示内容
87          UI_Main.mFuncBT.mRSSIVector.clear();
88          UI_Main.mFuncBT.mNameVector.clear();
89          UI_Main.mFuncBT.mAddrVector.clear();
90      }
91  }
```

操作说明：该类提供在 SurfaceView 上绘制功能。其中静态方法 initPaint 对画笔进行初始化，draw 函数负责绘制。

draw 函数的核心在于 canvas 绘图，canvas 绘图的过程和我们在白纸上绘绘图的过程很像，如：

- 第 55 行锁定 canvas 相当于得到一张纸；
- 第 56 行用 RGB 为 0 的颜色来刷新 canvas 相当于用橡皮擦把纸上原来的东西擦掉；
- 第 68 和 76 行 drawText 相当于在纸的相应位置写文字；
- 第 81 行 drawCircle 相当于在纸的相应位置绘制一个指定的圆；
- 第 86 行的 nlockCanvasAndPost 相当于你把绘制好的作品拿出来展示给别人看；

正是因为 canvas 的加锁和解锁这一机制，才保证了绘制过程中屏幕正确地显示。

接着再来理解这里 draw 函数的功能：mBTSArrayList 是一个 My_BTS 类型的数组，保存我们想要防丢的蓝牙防丢器设备的名称、地址等信息；mFuncBT 是一个可以实时搜索周边蓝牙设备的一个对象，其静态变量 mNameVector、mAddrVector、mRSSIVector 保存着实时搜索结果；这样核心部分的功能便

是通过两层循环遍历待防丢设备是否在本次搜索中，如果不在则显示“NOT_FIND”，如果在则由 RSSI 计算距离。(效果如图 8-4)

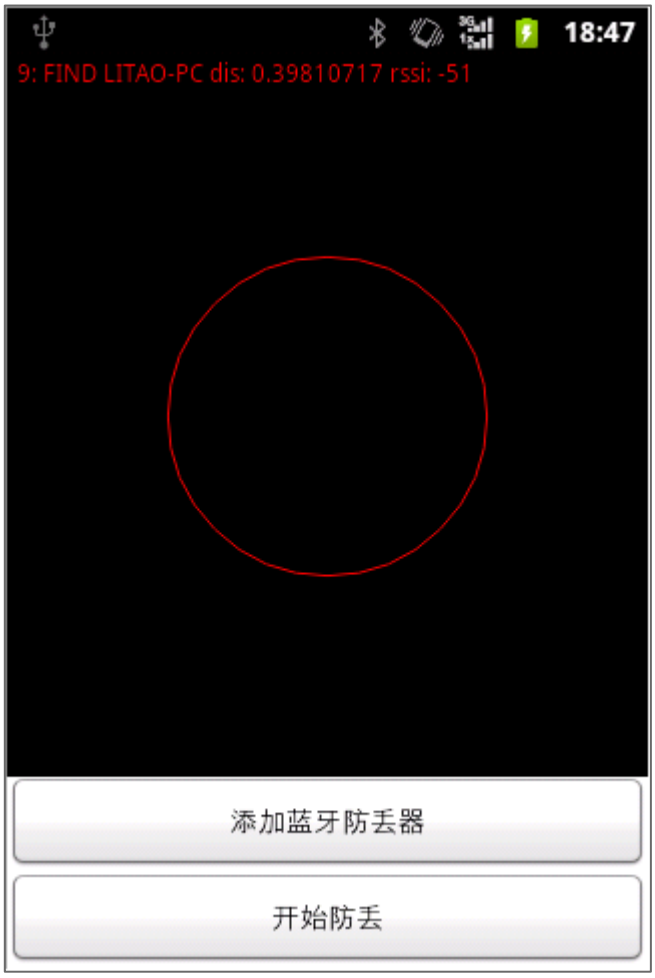


图 8-4 找到蓝牙设备图

9) 新建一个 Func_BT. java 文件，并修改为：

```
1  package com.example.first_test;
2
3  import java.util.Vector;
4
5  import android.app.Activity;
6  import android.bluetooth.BluetoothAdapter;
7  import android.bluetooth.BluetoothDevice;
8  import android.content.BroadcastReceiver;
9  import android.content.Context;
10 import android.content.Intent;
11 import android.content.IntentFilter;
12 import android.os.Bundle;
13 import android.os.Handler;
14 import android.os.Message;
```

```
15
16 public class Func_BT {
17     private BluetoothAdapter mBtAdapter;// 蓝牙适配器
18     private static final int ENABLE_BLUETOOTH = 1;
19     // 分别用于存储设备名地址名称和 RSSI 的向量
20     public Vector<String> mNameVector;
21     public Vector<String> mAddrVector;
22     public Vector<Short> mRSSIVector;
23
24     private Handler myHandler;
25     private Activity activity;
26
27     public Func_BT(Activity activity, Handler myHandler) {
28         this.myHandler = myHandler;
29         this.activity = activity;
30
31         mNameVector = new Vector<String>();// 向量
32         mAddrVector = new Vector<String>();
33         mRSSIVector = new Vector<Short>();
34
35         IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
36         activity.registerReceiver(mReceiver, filter);
37         filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
38         activity.registerReceiver(mReceiver, filter);
39         activity.registerReceiver(mReceiver, filter);
40
41         mBtAdapter = BluetoothAdapter.getDefaultAdapter();
42     }
43
44     private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
45         @Override
46         public void onReceive(Context context, Intent intent) {
47             String action = intent.getAction();
48             if (BluetoothDevice.ACTION_FOUND.equals(action)) {
49                 BluetoothDevice device = intent
50                         .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
51                 short rssi = intent.getExtras().getShort(
52                         BluetoothDevice.EXTRA_RSSI);
53                 mNameVector.add(device.getName());
54                 mAddrVector.add(device.getAddress());
55                 mRSSIVector.add(rssi);
56             } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED
57                     .equals(action)) {
58                 /*if (mNameVector.size() != 0) {
```

```
59                      Message msg = new Message();// 消息
60                      msg.what = 0x01;// 消息类别
61                      myHandler.sendMessage(msg);
62                  }*/
63              }
64          }
65      };
66
67      public void doDiscovery() {
68          if (mBtAdapter.isDiscovering()) {
69              mBtAdapter.cancelDiscovery();
70          }
71          mBtAdapter.startDiscovery();
72          new TimeLimitThread().start();
73      }
74
75      public void openBT() {
76          // 如果没有打开则打开
77          if (!mBtAdapter.isEnabled()) {
78              Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
79              activity.startActivityForResult(intent, ENABLE_BLUETOOTH);
80          } else {
81              doDiscovery();
82          }
83      }
84
85      protected void onActivityResult(int requestCode, int resultCode, Intent data){
86          if (requestCode == ENABLE_BLUETOOTH) {
87              if (resultCode == Activity.RESULT_OK) {
88                  doDiscovery();
89              }
90          }
91      }
92
93      public void setHandler(Handler myHandler) {
94          this.myHandler = myHandler;
95      }
96
97      public void setFunc_BT(Activity activity, Handler myHandler) {
98          this.myHandler = myHandler;
99          this.activity = activity;
100     }
101
102     class TimeLimitThread extends Thread{
```

```
103         public void run() {
104             try {
105                 sleep(3000);
106                 if (mBtAdapter.isDiscovering()) {
107                     mBtAdapter.cancelDiscovery();
108                 }
109                 Message msg = new Message();// 消息
110                 msg.what = 0x01;// 消息类别
111                 myHandler.sendMessage(msg);
112             } catch (InterruptedException e) {
113                 e.printStackTrace();
114             }
115         }
116     }
117 }
```

操作说明：该类用来实时搜索周边蓝牙设备，并把搜索结果保存在其静态成员变量 mNameVector、mAddrVector、mRSSIVector 中。此外这里还使用 Handler 用于向调用其搜索方法的 Activity 发送本次搜索完毕的消息。具体介绍如下：

- 第 27 到 42 行为构造函数，主要负责成员变量初始化、注册 filter、获取本地蓝牙设备；
- 第 44 到 65 行为 BroadcastReceiver，主要负责异步搜索周边蓝牙设备的广播的接收；
- 第 67 到 73 行为开始搜索周边蓝牙设备函数；
- 第 75 到 83 行为带有检查本地蓝牙是否开启并能够发出开启请求的搜索周边蓝牙设备函数；
- 第 85 到 91 行为接收用户是否授权打开蓝牙设备的 Activity 的结果函数；
- 第102到116行是一个线程类主要用于限定搜索周边蓝牙设备的最长时间(默认为12s左右)；

正是因为加了独立用于限定搜索时长的线程，才让搜索过程的时间长短便于我们控制，但是 sleep 的时间也不要设置的过小，否则会出现一个蓝牙设备也搜索不到的情况。(建议最好参照第七部分——“安卓蓝牙编程主要操作”来理解本部分的代码)

10) 修改 UI_Main.java 为：

```
1  package com.example.first_test;
2
3  import java.util.ArrayList;
4  import android.app.Activity;
5  import android.content.Intent;
6  import android.os.Bundle;
7  import android.os.Handler;
8  import android.os.Message;
9  import android.view.SurfaceHolder;
10 import android.view.SurfaceHolder.Callback;
11 import android.view.SurfaceView;
12 import android.view.View;
13 import android.view.View.OnClickListener;
14 import android.widget.Button;
```

```
15
16 public class UI_Main extends Activity implements Callback {
17
18     public Func_Draw mFuncDraw;
19     public SurfaceHolder mHolder;
20     public static Func_BT mFuncBT;
21     // 防丢设备
22     public static ArrayList<My_BTS> mBTSArrayList = new ArrayList<My_BTS>();
23
24     // 消息句柄(线程里无法进行界面更新,
25     // 所以要把消息从线程里发送出来在消息句柄里进行处理)
26     public Handler myHandler = new Handler() {
27         @Override
28         public void handleMessage(Message msg) {
29             if (msg.what == 0x01) {
30                 Func_Draw.draw(mHolder);
31             }
32             mFuncBT.doDiscovery();
33         }
34     };
35
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
39         setContentView(R.layout.ui_main);
40
41         Func_Draw.initPaint();
42
43         SurfaceView mSurface = (SurfaceView) findViewById(R.id.surfaceView);
44         mHolder = mSurface.getHolder();
45         mHolder.addCallback(this);
46
47         mFuncBT = new Func_BT(this, myHandler);
48
49         Button mButton1 = (Button) findViewById(R.id.button_start);
50         mButton1.setOnClickListener(new OnClickListener() {
51             @Override
52             public void onClick(View v) {
53                 Func_Draw.times = 0;
54                 mFuncBT.openBT();
55             }
56         });
57
58         Button mButton2 = (Button) findViewById(R.id.button_add);
```

```
59          mButton2.setOnClickListener(new OnClickListener() {
60              @Override
61              public void onClick(View v) {
62                  startActivity(new Intent(UI_Main.this, UI_List.class));
63              }
64          });
65      }
66
67      @Override
68      public void surfaceCreated(SurfaceHolder holder) {
69          // TODO Auto-generated method stub
70
71      }
72
73      @Override
74      public void surfaceChanged(SurfaceHolder holder, int format, int width,
75              int height) {
76          // TODO Auto-generated method stub
77
78      }
79
80      @Override
81      public void surfaceDestroyed(SurfaceHolder holder) {
82          // TODO Auto-generated method stub
83
84      }
85  }
```

操作说明：上面第 2 步时我们修改了 ui_main.xml 并做出应用程序主页面的效果，而该类则是其对应的逻辑实现。

- 首先看第 37 到 65 行的 onCreate 函数：
  - 第 39 行设置要显示的页面为 ui_main.xml 所展示的效果；
  - 第 41 行调用 Func_Draw 的静态 initPaint()方法对画笔进行初始化；
  - 第 43 到 45 行负责获取并设置 SurfaceView；
  - 第 47 行实例化一个 Func_BT；
  - 第 49 到 56 行是给开始防丢按钮绑定一个监听器，一旦点击该按钮则执行 onClick 内代码；
  - 第 58 到 65 行是给添加蓝牙防丢器绑定一个监听器，一旦点击则启动另一个 Activity；
- 第 24 到 34 行实例化一个 Handler 用于接收 Func_BT 一次搜索结束时发回结束的消息。在这里一旦收到本次周边蓝牙设备搜索结束的消息就调用 Func_Draw.draw(mHolder)进行绘制，然后继续调用 mFuncBT.doDiscovery()实现周期性搜索/防丢。
- 第 67 到 84 行是自动生成的，暂且不用管（也不要删除!）。

11）新建一个 UI_List.java 文件，并修改代码为：

```
1  package com.example.first_test;
```

```
2
3  import java.util.ArrayList;
4  import android.annotation.SuppressLint;
5  import android.app.Activity;
6  import android.app.ProgressDialog;
7  import android.content.Intent;
8  import android.os.Bundle;
9  import android.os.Handler;
10 import android.os.Message;
11 import android.util.Log;
12 import android.view.View;
13 import android.view.View.OnClickListener;
14 import android.widget.ArrayAdapter;
15 import android.widget.Button;
16 import android.widget.ListView;
17
18 public class UI_List extends Activity {
19
20     private ArrayList<String> Items;
21     private ListView myListView;
22     private ArrayAdapter<String> aa;
23     private boolean getDIV;
24
25     @SuppressLint("InlinedApi")
26     protected void onCreate(Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.ui_list);
29
30         UI_Main.mFuncBT.setFunc_BT(this, myHandler);
31
32         // 获取 listview 并设置为多选
33         myListView = (ListView) findViewById(R.id.listView1);
34         myListView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
35         myListView.setTextFilterEnabled(true);
36         // 设置 listview 数组并绑定
37         Items = new ArrayList<String>();
38         aa = new ArrayAdapter<String>(this,
39                 android.R.layout.simple_list_item_checked, Items);
40         myListView.setAdapter(aa);
41
42         // 获取 OK 按钮，并遍历选择的设备，返回主 Activity
43         Button myButton1 = (Button) findViewById(R.id.button_ok);
44         myButton1.setOnClickListener(new OnClickListener() {
45             @Override
```

```
46              public void onClick(View v) {
47                  int num = 0;
48                  for (int i = 0; i < myListView.getCount(); i++) {
49                      if (myListView.isItemChecked(i)) {
50                          String item = myListView.getItemAtPosition(i)
51                                  .toString();
52                          String name = item.substring(0, item.indexOf("\n"));
53                          String addr = item.substring(item.indexOf("\n") + 1);
54                          Log.i("UI_LIST", name + " " + addr);
55                          UI_Main.mBTSArrayList
56                                  .add(num++, new My_BTS(name, addr));
57                      }
58                  }
59                  Intent mIntent = new Intent(UI_List.this, UI_Main.class);
60                  mIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
61                  startActivity(mIntent);
62              }
63          });
64          // 获取 Search 按钮，设置监听事件
65          Button myButton2 = (Button) findViewById(R.id.button_search);
66          myButton2.setOnClickListener(new OnClickListener() {
67              @Override
68              public void onClick(View v) {
69                  getDIV = false;
70                  UI_Main.mFuncBT.openBT();
71                  final ProgressDialog dialog = ProgressDialog.show(UI_List.this,
72                          "搜索蓝牙设备", "稍等一下~", true);
73                  new Thread(new Runnable() {
74                      public void run() {
75                          while (getDIV == false);
76                          dialog.dismiss();
77                      }
78                  }).start();
79              }
80          });
81      }
82
83      // 消息句柄(线程里无法进行界面更新,
84      // 所以要把消息从线程里发送出来在消息句柄里进行处理)
85      public Handler myHandler = new Handler() {
86          @Override
87          public void handleMessage(Message msg) {
88              if (msg.what == 0x01) {
89                  Items.clear();
```

```
90                  for (int i = 0; i < UI_Main.mFuncBT.mNameVector.size(); i++) {
91                      Items.add(i, UI_Main.mFuncBT.mNameVector.get(i) + '\n'
92                              + UI_Main.mFuncBT.mAddrVector.get(i));
93                      aa.notifyDataSetChanged();// 通知数据变化
94                  }
95                  getDIV = true;
96                  UI_Main.mFuncBT.mRSSIVector.clear();
97                  UI_Main.mFuncBT.mNameVector.clear();
98                  UI_Main.mFuncBT.mAddrVector.clear();
99              }
100         }
101     };
102 }
```

操作说明：这个类和 ui_list.xml 也是配套的。因此：

- 在 onCreate 中：
  - 第 28 行设置要显示的页面为 ui_list.xml 所展示的效果；
  - 第 30 行负责将周边蓝牙搜索对象的 Handler 设置为本 Activity 内的 Handler；
  - 第 32 到 40 行是 list 相关；
  - 第 42 到 80 行是两个按钮点击事件监听相关；
- 第 85 到 101 行的 Handler 则同 UI_Main.java 中的 Handler 类似负责接收周期性蓝牙搜索结束消息。在这里当接到蓝牙搜索结束的消息后是将搜索到的设备信息放入 list 中待用户选择。（没有再次调用蓝牙搜索）
- 此外两个按钮监听中执行部分要特别说明下：
  - 当点击 OK 按钮时程序将用户在 list 中选择的设备的信息存放在 UI_Main.mBTSArrayList 中，然后结束当前 Activity 并启动 UI_Main 所对应的 Activity。
  - 当点击 search 按钮时会启动周边蓝牙设备搜索并打开一个等待对话框，其中的 Thread 负责等待直到搜索完毕。

12）最后（如图 8-5）找到 AndroidMainFest.xml 文件修改为：

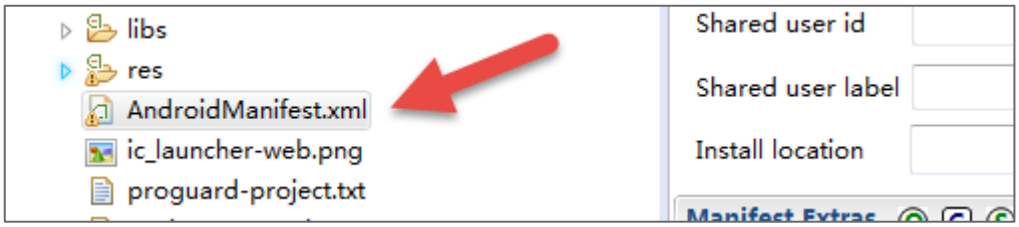


图 8-5 AnDroidManifest.xml 文件

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.first_test"
4      android:versionCode="1"
5      android:versionName="1.0" >
```

```
6
7       <uses-sdk
8           android:minSdkVersion="8"
9           android:targetSdkVersion="19" />
10
11  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
12  <uses-permission android:name="android.permission.BLUETOOTH" />
13
14      <application
15          android:allowBackup="true"
16          android:icon="@drawable/ic_launcher"
17          android:label="@string/app_name"
18          android:theme="@style/AppTheme" >
19          <activity
20              android:name=".UI_Main"
21              android:label="@string/app_name" >
22              <intent-filter>
23                  <action android:name="android.intent.action.MAIN" />
24                  <category android:name="android.intent.category.LAUNCHER" />
25              </intent-filter>
26          </activity>
27
28          <activity
29              android:name=".UI_List"
30              android:label="@string/app_name" >
31              <intent-filter>
32                  <action android:name="android.intent.action.UI_List" />
33                  <category android:name="android.intent.category.DEFAULT" />
34              </intent-filter>
35          </activity>
36
37      </application>
38
39
40  </manifest>
```

操作说明：主要是在 11、12 行添加蓝牙权限以及在 28 到 35 行添加一个 activity。

9 最终成果检查

到目前为止我们已经全部完工，下面是成果自评的时候啦（请参照下列条目给自己打分）：

- 你做的蓝牙防丢器可以被搜到（+ 20 分）
- 大致读懂了第 7 部分的内容（+ 10 分）
- 亲自按照第 8 部分实现了防丢器的安卓应用（+ 40 分）
- 对软件或者硬件有加入自己想法改进的（+ 20 分）

- 将自己制作的防丢器推广给别人用的（+ 20 分）
- 搜索资料做出蓝牙室内导航的（+ 80 分）
- ……

及格分 70 分，不够的要注意打好基础哦！