目录

[一、 系统背景 3](#_Toc6398)

[1.1 Android系统概述 3](#_Toc15944)

[1.2 蓝牙概述 4](#_Toc13815)

[二、 系统分析 6](#_Toc3124)

[2.1 需求分析 6](#_Toc13771)

[2.2 技术基础 6](#_Toc18317)

[2.3 平台搭建 9](#_Toc27751)

[2.4 运行环境 12](#_Toc7788)

[三、 系统总体设计 13](#_Toc22996)

[3.1 设计思想 13](#_Toc22950)

[3.2 设计框架 14](#_Toc4285)

[四、 程序设计 17](#_Toc8640)

[4.1 设置蓝牙权限 17](#_Toc32273)

[4.3 设置蓝牙可见 17](#_Toc25724)

[4.4 查询已配对蓝牙设备 18](#_Toc725)

[4.5 扫描周围蓝牙设备 18](#_Toc10222)

[4.6 服务器端的连接 19](#_Toc6496)

[4.7 客户端的连接 20](#_Toc16198)

[4.8 数据通信线程 21](#_Toc18835)

[五、 系统测试 23](#_Toc13983)

[六、 设计体会 27](#_Toc3029)

1. **系统背景**

**1.1 Android系统概述**

自从2008年9月22日，美国运营商T-Mobile USA在纽约正式发布了第一款基于android安卓的手机后，更多的移动设备厂商看到了安卓android的光明前景，并纷纷加入其中，android甚至已经涉足上网本市场了其发展前景广阔涉及领域越来越大。随着android手机的普及，android应用的需求势越来越大，这个潜力巨大的市场，已经吸引无数软件开发厂商和开发者投身其中。android的开放性使其比苹果和黑莓对年轻人更具吸引力，更能吸引年轻人群的关注和喜爱。在18岁至24岁族群中，Android是最多人选用的操作系统。同时因为Android系统的开放性为Android平板电脑的快速发展奠定了基础，也才能促使其吸引着越来越多的用户接受和使用。同时Android还具有应用程序无界限、应用程序是在平的的条件下创建的、应用程序可以轻松的嵌入网络、应用程序可以并行运行等众多出色的品质。具有其他操作系统无法比拟的优势和发展空间。

安卓系统在不断升级完善，如今，平板电脑市场已成为Android与苹果IOS除了智能手机领域之外争夺的第二个焦点。一直以来，多数新闻仍然显示iPad在平板市场的霸主地位难撼。但是，以Android领头的平板电脑正以高速的增长对苹果iPad造成冲击。放眼整个平板市场，生产Android平板电脑的确是大势所趋。世界上所有的顶级设备制造商，三星、索尼、摩托罗拉、联想、LG、宏碁、HTC、华硕，几乎每一家都在卖Android平板电脑。

安卓给用户的触屏体验非常好，安卓手机在高端机领域泛滥的同时也在中低端机领域涉足，这将给安卓系统手机带来非常好的前景，从目前的市场上来看，安卓系统能够在短短的时间里面，占据着超大的市场份额，而且还有非常好的信誉口碑，绝对不是一两个人吹捧出来的，那是因为安卓手机真的能够让人体验到好，才能够有如此高的评价。所以说，虽然目前安卓手机还是存在一定问题上的不足，很多问题都有待需要改善，这就说明安卓系统处于一种发展状态，那么它的生命力就会很强，前景很广阔。

Android是一套真正意义的开放性移动设备综合平台，它包括操作系统、中间件和一些关键的平台应用。Android的Java程序运行环境包含一组Java核心函数库及Dalvik虚拟机，他们有效地优化了Java程序的运行过程。Android系统平台基于优化了的Linux内核，它提供诸如内存管理、进程管理、设备驱动等服务，同时也是手机硬件的连接层。

Android最大的优点在于它是一个开放的体系结构，具有非常好的开发和调试环境，而且还制成各种可扩展的用户体验，Android里面有非常丰富的图形系统，对多媒体的支持功能和非常强大的浏览器。 Android平台的开放性特点既能促进技术的创新，又有助于降低开发成本，还可以使运营商能非常方便地地址特色化产品，因此，它具有很大的市场发展潜力。

**1.2 蓝牙概述**

自从1998年提出蓝牙技术以来，蓝牙技术的发展异常迅速。蓝牙Bluetooth作为一种新的短距离无线通信技术标准，受到全世界越来越多工业界生产厂家和研究机构的广泛关注。成立了世界蓝牙组织Bluetooth SIG，采用技术标准公开的策略来推广蓝牙技术，现已发展成为一个相当大的工业界高新技术标准化组织，全球支持蓝牙技术的2000多家设备制造商都已经成为它的会员，一项公开的、全球统一的技术规范得到了工业界如此广泛的关注和支持在以往是罕见的。近年来，世界上一些权威的标准化组织，也都在关注蓝牙技术标准的制定和发展。例如，IEEE的标准化机构，也已经成立了802．15工作组，专门关注有关蓝牙技术标准的兼容和未来的发展等问题。IEEE 802.15.1 TG1就是讨论建立与蓝牙技术1.0版本相一致的标准；IEEE 802.15.2 TG2是探讨蓝牙如何与IEEE 802.11b无线局域网技术共存的问题；而IEEE 802.15.3 TG3则是研究未来蓝牙技术向更高速率(如10-20Mbits/s)发展的问题。国内的一些生产厂家与研究部门也准备开始组织蓝牙技术产品的开发。由来自国家主管部门、企业界、学术界以及研究生产机构的领导、专家、教授等权威人士发起成立的中国蓝牙技术发展与应用论坛，吸引了众多关注蓝牙技术的各界人士，还组织国内各界与世界蓝牙组织SIG的代表，就双方所关注的问题进行了认真的讨论。并就双方今后进一步加强联系、共享蓝牙技术信息资源、共同促进蓝牙技术在中国的推广与应用等问题达成共识。

 蓝牙是取代数据电缆的短距离无线通信技术，可以支持物体与物体之间的通信，工作频段是全球开放的2.4GHz频段，可以同时进行数据和语音传输，传输速率可达到10Mb/s，使得在其范围内的各种信息化设备都能实现无缝资源共享。 蓝牙技术的应用被认为非常广泛而且极具潜力。它可以应用于无线设备(如PDA、手机、智能电话、无绳电话)、图像处理设备(照相机、打印机、扫描仪)、安全产品(智能卡、身份识别、票据管理、安全检查)、消费娱乐(耳机、MP3、游戏)汽车产品(GPS、ABS、动力系统、安全气袋)、家用电器(电视机、电冰箱、电烤箱、微波炉、音响、录像机)、医疗健身、建筑、玩具等领域。蓝牙行业对于市场的持续增长感到欣慰，现在没有人再质疑它的生命力。2002年，400余种蓝牙产品的销量总共达到了3000万件；而2003年的数字是2002年的2倍。In-Stat／MDR公司预测，蓝牙市场的规模在2007年将膨胀到6亿件。爱立信技术授权公司的总裁Maria Khorsand表示：“在如此艰难的经济环境下，蓝牙是少数仍在增长的产品之一”。

1. **系统分析**

**2.1 需求分析**

本系统设计的目标是开发出一个基于Android系统的短距离蓝牙通信软件。现在几乎每一部智能手机都有蓝牙连接，在安装本程序设计APP后，可以实现在蓝牙的有效距离内进行数据传输，并且不用耗费额外的流量和话费。

用户打开应用软件后首先要提醒用户允许打开自己的手机蓝牙，然后扫描附近的其他蓝牙设备，并选择其中一个设备对象进行连接并进行数据通信，所以该设计要有对其他蓝牙设备的扫描功能，连接功能和数据通信功能。另外，本设计还可以实现发送接受表情，显示在线用户，查看聊天记录，设置本机用户名，可以实现短距离蓝牙通信的基本操作。

本设计界面简洁明了，使用户使用方便快捷，并且加入了再按一次退出提醒功能，使操作更加人性化。

**2.2 技术基础**

本设计是基于Android平台的一个设计。Android是Google开发的基于linux平台的开源智能手机操作系统。Android平台的基本框架如图2.2.1所示。

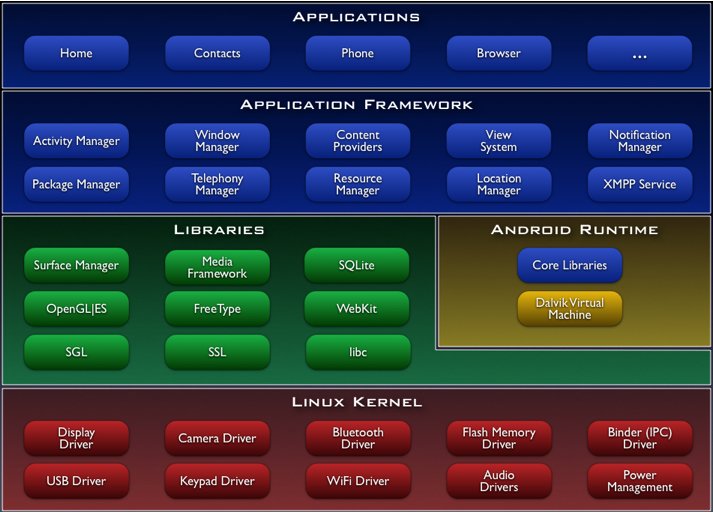


图2.2.1 Android基本框架图

(1)Applications  Application Android会同一个核心应用程序包一起发布，该应用程序包包括email客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是用JAVA编写的。

(2)Application FrameWork开发者完全可以访问核心应用程序所使用的API框架。该应用程序架构用来简化组件软件的重用；任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块（不过得遵循框架的安全性限制）。

(3)Libraries库  Android包括一个被Android系统中各种不同组件所使用的 C/C++ 库集。该库通过 Android 应用程序框架为开发者提供服务。

(4)Android Runtime  Android包括了一个核心库，该核心库提供了JAVA编程语言核心库的大多数功能。

(5)Linux内核  Android的核心系统服务依赖于Linux2.6内核，如安全性，内存管理，进程管理，网络协议栈和驱动模型。Linux内核也同时作为硬件和软件堆栈之间的硬件抽象层。

Android的四大基本组件：Activity(活动)，Service(服务)，BroadcastReceiver(广播)，ContentProvider(内容提供者)，各组件之间要进行切换需要一个意图或企图Intent类，通过启动StartIntent()方法来使各组件进行跳转 。

Android的五大布局方式：LinearLayout(线性布局)，FrameLayout(帧布局)，TableLayout(表格布局) ，AbsoluteLayout(绝对布局)和RelativeLayout(相对布局）。五种布局方式可以相互嵌套，Android各种漂亮的界面都是通过嵌套布局而实现的。

Android的线程以及线程之间通信的中介：Handler类。

Android配置文件：AndroidManifest.xml及res资源文件中各种属性布局方式引用的资源等。

Android应用程序由一些零散的有联系的组件组成，通过一个工程manifest绑定在一起，在manifest中，描述了每一个组件以及组件的作用。 然而，并不是所有的程序都需要全部部分，可以由上面的一个或几个来组建。使用某些组件构建Android应用程序时，需要在AndroidManifest．xml文件里进行声明。

1. Activities（活动） 在Android应用程序中，Activity一般代表手机的一屏．即用户可见的一个界面。An droid应用程序的运行过程就是在不同的Activity之间切换的过程。每个Activity都有自己的视图和事件对应。大多数应用程序都有多个Activity。每一个画面对应于你的应用程序，将会是Activity类的扩展。Activity使用Views去构建UI来显示信息和响应用户的行为。Activity的生命周期示意图如图2.2.2所示。

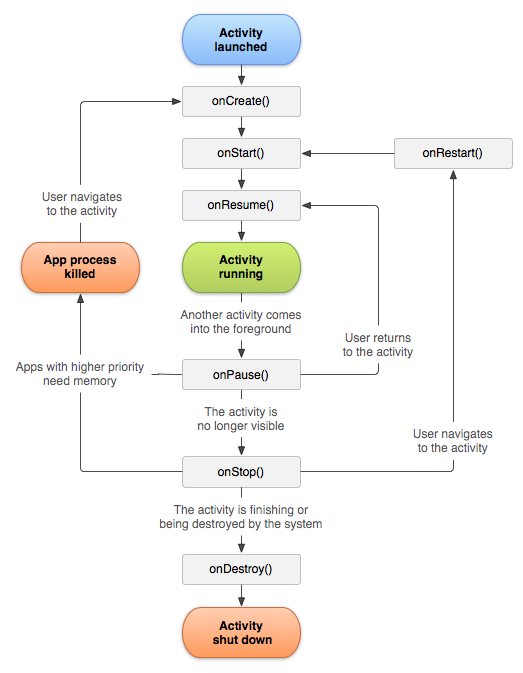


图2.2.2 Activity生命周期示意图

1. Services（服务） Service是具有长生命周期并且没有用户界面的代码 通过Start Service(Intent service)可以来启动一个Service，通过Context．bindService ()可以绑定一个Service。当已经连接上了一个服务，就可以使用服务提供的外部接口来和服务通讯了Android应用程序中不可见的“工人”。 Service组件运行时不可见，但它负责更新的数据源和可见的Activity，以及触发通知。它们常用来执行一些需要持续运行的处理，当你的Activity已经不处于激活状态或不可见。

（3）Content（内容） Content Provider（内容提供器）就是一个类。作为应用程序之间唯一的共享数据的途径，它实现了一个标准的方法集合，其他应用程序可以使用这些方法在应用程序之间共享数据。提供共享的数据存储。Content Provider用来管理和共享应用程序的数据库。在应用程序间，Content Provider是共享数据的首选方式。这意味着，你可以配置自己的Content Provider去存取其他的应用程序或者通过其他应用程序暴露的Content Provider去存取它们的数据。Android设备本身包含了几个Content Provider来访问像联系人信息等有用的数据库。

1. Intents（意图）简单的消息传递框架。使用Intent，你可以在整个系统内广播消息或者给特定的Activity或者服务来执行你的行为意图。系统会决定那个（些）目标执行适当的行为，Intent有显式和隐式两种方式。
2. Broadcast Receivers（广播接收器） Broadcast Receiver是一个接受广播消息并响应的组件。广播消息可以由系统产生，例如时区发生变化、电池电量发生变化等。同时也可以由其他应用程序产生，对于这两种广播。在Broadcast receiver里都可以设置相应的过滤规则来进行接收。Broadcast Receiver是Intent广播的“消费者”。通过创建和注册一个Broadcast Receiver，应用程序可以监听符合特定条件的广播的Intent。Broadcast Receiver 会自动的启动你的Android应用程序去响应新来的Intent。Broadcast Receiver是事件驱动程序的理想手段。
3. Notifications（通知） 用户通知的框架。Notification用来在不需要焦点或不中断它们当前Activity的情况下提示用户。它们是Service或Broadcast Receiver获得用户注意的首选方式。例如，当设备收到文本信息或外部来电时，它通过闪光，发声，显示图标或显示对话框信息来提醒你。

**2.3 平台搭建**

### （1）java JDK 安装

JDK是Java语言的原创者推出的原始开发环境。采用JDK开发Java程序能够很快理解程序中各部分代码之间的关系，有利于理解Java面向对象的设计思想。在此之前，我们就必须先完成此项工作及在Windows上安装JDK，首先访问Oracle的官方网站下载页面，也可以从其他网站下载，下载完成后，双击JDK的安装程序，装好后配置JDK的环境变量，在“我的电脑”|“属性”|“高级”|“环境变量”|“系统变量”中先添加两个环境变量：

JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_77（安装路径）

classpath=.;%JAVA\_HOME%\lib

之后修改环境变量PATH，在前面加上：%JAVA\_HOME%\bin;

完成后测试一下在命令提示符中输入java -version 显示如图 2.3.1所示则表示完成了。



图 2.3.1 Java安装测试

### (2)Eclipse安装以及ADT工具的安装

首先下载Eclipse，直接解压放到了C:\Program Files\Java\jre8\eclipse下，之后打开eclipse设置工作路径：

C:\Program Files\Java\jre8\eclipse\workspace

下面由于学校校园网不流畅我直接下载了ADT-26.0.6.zip,准备离线安装ADT插件。打开eclipse，选择Help > Install New Software...之后然后选择Add...在name项中随便取一个名字如ADT-22.0.6，之后点击Archiv...找到下载的ADT-26.0.6.zip 如图 2.3.2所示。

点击OK，然后选择Select all之后一路的next和OK最后会根据提示重启下eclipse，则完成了ADT的安装，至于如何看是否完成则是选择 Window > Preferences…打开首选项面板看是否有Android，有的话就成功了，至于会跳出要求装SDK则在下面进行安装了。

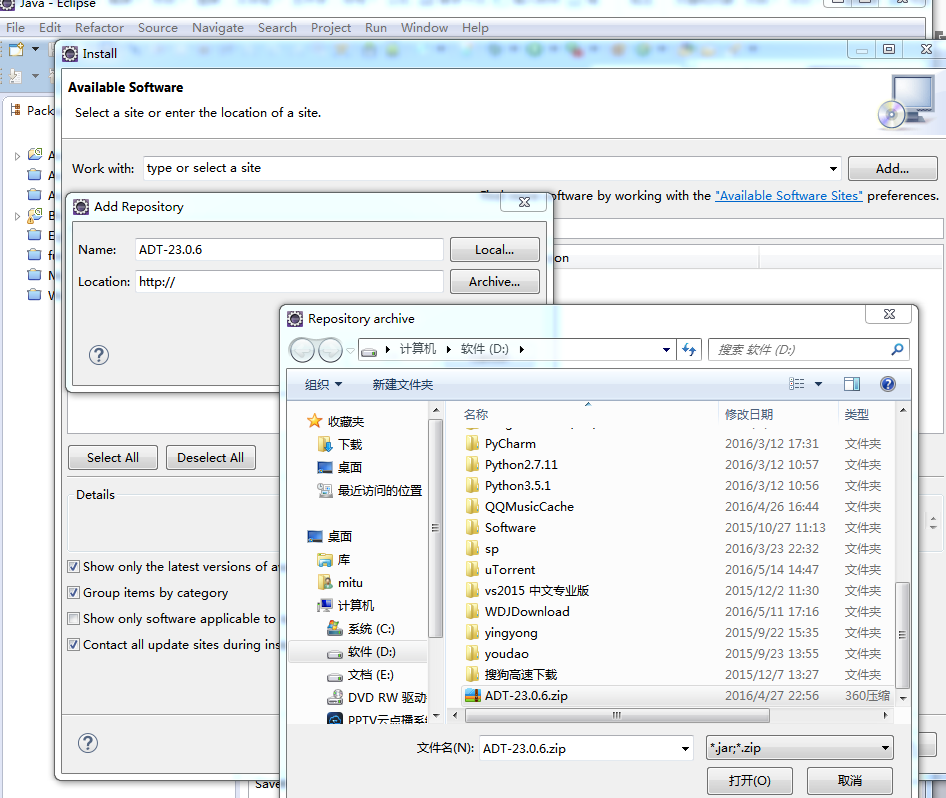


图 2.3.2 ADT安装步骤

### (3)Android SDK 的安装以及系统的设置

首先下载最新的SDK直接解压到C:\Program Files\Java\后修改环境变量PATH，在前面加上：

C:\Program Files\Java\android-sdk-windows\tools;

然后Window > Preferences…弹出“Preferences”对话框，单击左侧“Android”列表项，单击右侧“Browse”按钮，选择SDK路径（例如C:\Program Files\Java\android-sdk-windows），单击“apply”按钮，退出SDK安装提示对话框，单击“OK”按钮，退出“Preferences”对话框。

下面是打开Window > Android SDK Manager

选择：“Tools”下“Android SDK platform-tools”

“Extras”下“Google USB driver package”

“Android 2.3.3”

单击“Install 22 packages”按钮

选中“Accept All” （如果选择“Accept”，可能漏掉“Google USB driver package”）

单击“Install”按钮

之后就是无尽的等待，完成后关闭，如图2.3.3所示。

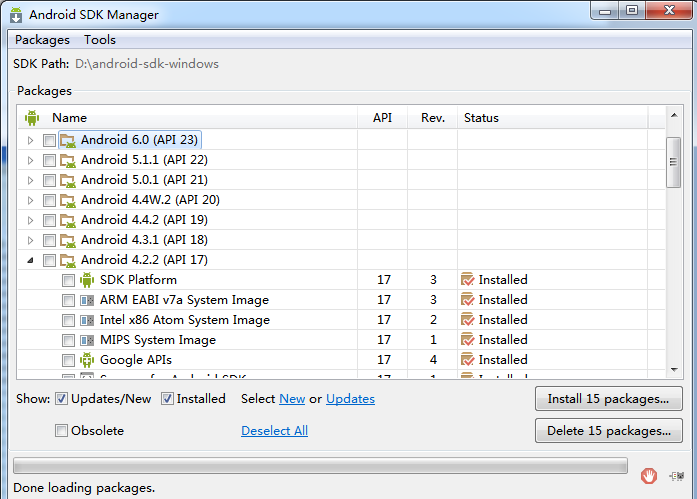


图2.3.3 Android SDK 更新界面

SDK装好后还要修改下环境变量PATH，在前面加上：

C:\Program Files\Java\android-sdk-windows\platform-tools;

**2.4 运行环境**

1、硬件环境

 两台Android系统的智能手机

一台Windows7系统的电脑

2、软件环境

 Android4.0.0及其以上手机系统

Eclipse +JDK + ADT + SDK

1. **系统总体设计**

**3.1 设计思想**

在Android系统中对蓝牙的操作包括开启蓝牙，关闭蓝牙，设置蓝牙可见性，扫描周围蓝牙设备，对蓝牙设备进行配对，连接蓝牙设备，进行数据通信。其中对第一次扫描到的蓝牙设备进行配对过程是通过系统秘钥手动完成的，是为了保证数据通信的安全性。我们的技术开发主要的工作是设置蓝牙可见性，扫描周围的蓝牙设备，选择一台蓝牙设备进行连接并进行数据通信。并在此基础上完成数据库的读写操作，实现查看聊天记录的功能。

首先两个设备上都要有蓝牙设备或者专业一点叫蓝牙适配器，以手机和电脑为例我画了一个蓝牙工作流程图如图3.1所示。其次在手机上进行扫描，扫描周围的蓝牙设备，找到手机附近的电脑后给它发出一个信号需要进行蓝牙的配对，再次返回一个信号说明手机和电脑已经配对成功了，最后配对成功后就可以进行连接和数据通信了。这是一个最基本的流程。

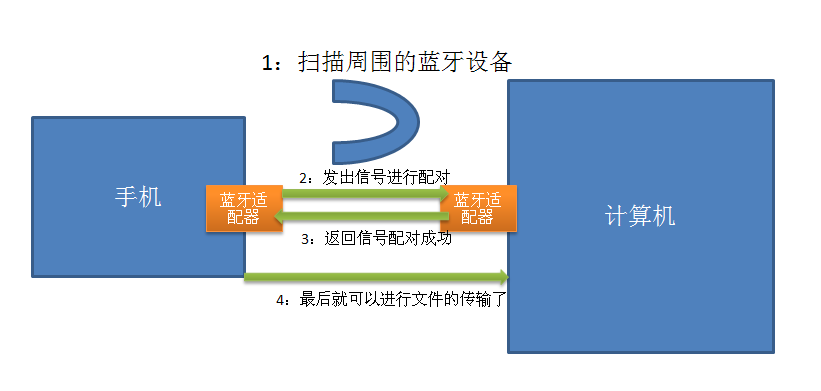


图3.1 蓝牙工作流程图

在Android中进行蓝牙开发的相关类和接口都位于Android.bluetooth包中。其中最主要的几个API如下所示。

BluetoothAdapter类：代表了一个本地的蓝牙适配器。它是所有蓝牙交互的的入口点。利用它你可以发现其他蓝牙设备，查询绑定了的设备，使用已知的MAC地址实例化一个蓝牙设备和建立一个BluetoothServerSocket（作为服务器端）来监听来自其他设备的连接。

BluetoothDevice类：代表了一个远端的蓝牙设备，使用它请求远端蓝牙设备连接或者获取远端蓝牙设备的名称、地址、种类和绑定状态。其信息是封装在bluetoothsocket中。

BluetoothSocket类：代表了一个蓝牙套接字的接口（类似于tcp中的套接字），它是应用程序通过输入、输出流与其他蓝牙设备通信的连接点。

BlueboothServerSocket类：代表打开服务连接来监听可能到来的连接请求（属于Server端），为了连接两个蓝牙设备必须有一个设备作为服务器打开一个服务套接字。当远端设备发起连接连接请求的时候，并且Server端已经接受连接时，BlueboothServerSocket类将会返回一个BluetoothSocket。

Bluetoothclass类：描述了一个蓝牙设备的一般特点和能力。它的只读属性集定义了设备的主、次设备类和一些相关服务。然而，它并没有准确地描述所有该设备所支持的蓝牙文件和服务，而是作为对设备种类来说的一个小小暗示。

**3.2 设计框架**

完整的蓝牙操作过程示意图如图3.2.1所示。

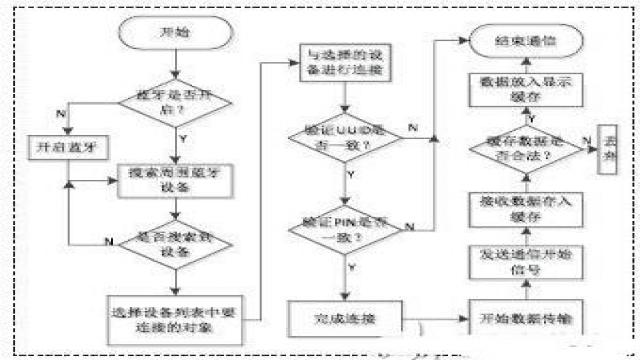


图3.2.1 蓝牙操作流程图

扫描周围蓝牙设备得到配对的设备列表后，需要从中选取一个设备进行连接并进行数据通信。蓝牙数据通信过程采用C/S模型，两台手机通过蓝牙进行数据传输，其中主动发起连接请求的设备视为客户端，监听并接受请求的设备视为服务器端。下面分别介绍蓝牙通信在服务器端和客户端上的连接操作过程如图3.2.2和图3.2.3所示。

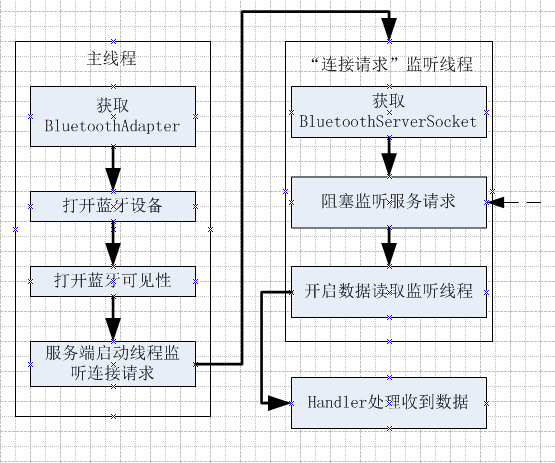


图3.2.2 服务器端连接过程

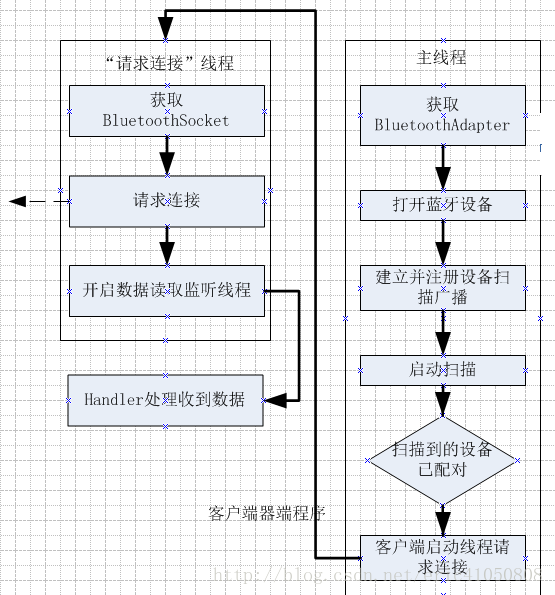


图3.2.3 客户端连接过程

在服务器端，注意如下几点：

1.一定要保证蓝牙开启且通信双方已经手动完成匹配；

2.服务端要开启蓝牙可见性，以保证能被客户端搜索到；

3.用BluetoothServerSocket的listenUsingRfcommWithServiceRecord()方法获取ServerSocket对象，再使用BluetoothServerSocket的accept()方法阻塞接收请求者的连接请求，成功后该方法返回BluetoothSocket对象；

4.使用获取到的socket对象的输入输出流方法操作数据通信，接收到的数据一定要通过handler处理显示。

在客户端，注意如下几点：

1.同样一定要保证蓝牙开启且通信双方已经手动完成匹配；

2.通过广播扫描周围可用设备，并确定该设备已与本机完成匹配；

3.通过调用BluetoothAdapter的startDiscovery()方法异步扫描到周围的BluetoothDevice对象，调用createRfcommSocketToServiceRecord()方法，获取BluetoothSocket对象，并用该对象调用connect()方法想服务器提出连接请求。注意此连接请求过程要循环请求，直至完成连接；

4.使用此连接的socket对象的输入输出流方法操作数据通信，接收到的数据一定要通过handler处理显示。

1. **程序设计**

**4.1 设置蓝牙权限**

在AndroidManifast.xml配置文件中设置蓝牙操作权限。代码如下：

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

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/>

**4.2 启动蓝牙功能**

首先通过调用静态方法getDefaultAdapter()获取蓝牙适配器BluetoothAdapter，如果返回为空，说明该设备没有蓝牙适配器。代码如下：

BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

if (mBluetoothAdapter == null) { }

其次，调用isEnabled()来查询当前蓝牙设备的状态，如果返回为false，则表示蓝牙设备没有开启，接下来你需要封装一个ACTION\_REQUEST\_ENABLE请求到intent里面，调用startActivityForResult()方法开启蓝牙设备。代码如下：

if (!mBluetoothAdapter.isEnabled()) {

Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE);

startActivityForResult(enableBtIntent, REQUEST\_ENABLE\_BT);

}

**4.3 设置蓝牙可见**

设置蓝牙设备可见，使其他设备可以搜索到本机蓝牙设备，然后将ACTION\_REQUEST\_DISCOVERABLE动作封装在Intent中然后通过调用方法startActivity(Intent)可以实现在应用程序不退出的情况下使你的设备能够被发现。缺省情况下的使能时间是120秒，可以通过添加EXTRA\_DISCOVERABLE\_DURATION字段来改变使能时间（最大不超过300秒，这是出于对你设备上的信息安全考虑）。代码如下：

Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_DISCOVERABLE);

discoverableIntent.putExtra(BluetoothAdapter.EXTRA\_DISCOVERABLE\_DURATION, 300);

startActivity(discoverableIntent);

**4.4 查询已配对蓝牙设备**

在建立连接之前必须先查询已配对的蓝牙设备列表（你周围的蓝牙设备可能不止一个），以便你选取某一个设备进行数据通信。代码如下：

Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();

if (pairedDevices.size() > 0) {

for (BluetoothDevice device : pairedDevices) {

mArrayAdapter.add(device.getName() + "\n" + device.getAddress()); // 添加设备名和MAC地址到设备列表中

}

**4.5 扫描周围蓝牙设备**

BluetoothAdapter调用startDiscovery()方法异步扫描周围蓝牙设备，这个扫描的过程大概持续12秒，每扫描到一个BluetoothDevice系统就会广播ACTION\_FOUND动作。BluetoothAdapter会注册一个BroadcastReceiver来获取该设备富的设备名和MAC地址。代码如下：

private final BroadcastReceiver mReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (BluetoothDevice.ACTION\_FOUND.equals(action)){

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

stView mArrayAdapter.add(device.getName() + "\n" + device.getAddress()); }

}

};

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

registerReceiver(mReceiver, filter);

**4.6 服务器端的连接**

首先用listenUsingRfcommWithServiceRecord(String, UUID)方法来获取BluetoothServerSocket对象，参数String代表了该服务的名称，UUID代表了和客户端连接的一个标识（128位格式的字符串ID，相当于PIN码），UUID必须双方匹配才可以建立连接。其次调用accept（）方法来监听可能到来的连接请求，当监听到一个请求后，返回一个连接上的蓝牙套接字BluetoothSocket。最后，在接受到一个连接以后，调用close（）方法来关闭监听程序，因为一般蓝牙设备之间是点对点的传输的。

注意：accept（）方法不应该放在主Acitvity里面，因为它是一种阻塞调用，在没有监听到连接请求之前程序就一直停在那里，应该新建一个AcceptThread线程来管理。代码如下：

 private class AcceptThread extends Thread {

private final BluetoothServerSocket mmServerSocket;

public AcceptThread() {

BluetoothServerSocket tmp = null;

try {

tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME, MY\_UUID);

} catch (IOException e) { }

mmServerSocket = tmp;

 }

public void run() {

BluetoothSocket socket = null;

while (true) {

try {

socket = mmServerSocket.accept();

} catch (IOException e) { break; }

if (socket != null) {

manageConnectedSocket(socket);

mmServerSocket.close();

break; }

}

}

public void cancel() {

try {

mmServerSocket.close();

} catch (IOException e) { }

}

}

**4.7 客户端的连接**

通过调用BluetoothDevice的createRfcommSocketToServiceRecord(UUID)方法来获取BluetoothSocket。UUID就是匹配码。然后调用connect（）方法来请求数据连接。如果服务器端接受了该连接，他们将在通信过程中共享RFFCOMM信道。

注意：conncet()方法也是阻塞调用，一般建立一个独立的ConnectThread线程来调用该方法。在设备搜索过程中不应该发起连接connect()方法，这样会明显减慢速度以至于连接失败。且数据传输完成只有调用close()方法来关闭连接，这样可以节省系统内部资源。代码如下：

private class ConnectThread extends Thread {

private final BluetoothSocket mmSocket;

private final BluetoothDevice mmDevice;

public ConnectThread(BluetoothDevice device) {

BluetoothSocket tmp = null;

mmDevice = device;

try {

tmp = device.createRfcommSocketToServiceRecord(MY\_UUI;

} catch (IOException e) { }

mmSocket = tmp;

}

public void run() {

mAdapter.cancelDiscovery();

try {

mmSocket.connect();

} catch (IOException connectException) {

try {

mmSocket.close();

} catch (IOException closeException) { }

return;

}

  manageConnectedSocket(mmSocket);

}

**4.8 数据通信线程**

当两台设备连接上以后，每个设备都拥有各自的BluetoothSocket。现在就可以实现设备之间的数据通信了。首先通过调用getInputStream()和getOutputStream()方法来获取输入输出流。然后通过调用read(byte[]) 和write(byte[])方法来读取或者写数据。

 注意：读取和写操作都是阻塞调用，需要建立一个单独的线程ConnectedThread来管理。

private class ConnectedThread extends Thread {

private final BluetoothSocket mmSocket;

private final InputStream mmInStream;

private final OutputStream mmOutStream;

public ConnectedThread(BluetoothSocket socket) {

mmSocket = socket;

InputStream tmpIn = null;

OutputStream tmpOut = null;

try {

tmpIn = socket.getInputStream();

tmpOut = socket.getOutputStream();

} catch (IOException e) { }

mmInStream = tmpIn;

mmOutStream = tmpOut;

}

public void run() {

byte[] buffer = new byte[1024];

while (true) {

try {

bytes = mmInStream.read(buffer);

mHandler.obtainMessage(MESSAGE\_READ,bytes,-1,buffer).sendToTarget();

} catch (IOException e) { break; }

}

}

public void write(byte[] bytes) {

try {

mmOutStream.write(bytes);

} catch (IOException e) { }

}

public void cancel() {

try {

mmSocket.close();

} catch (IOException e) { }

}

}

1. **系统测试**

打开BluetoothChat应用APP，会出现询问是否打开蓝牙的窗口，如图5.1所示。



图5.1 是否允许打开蓝牙

点击允许按钮后进入主界面，然后点击Menu键出现菜单栏如图5.2所示。

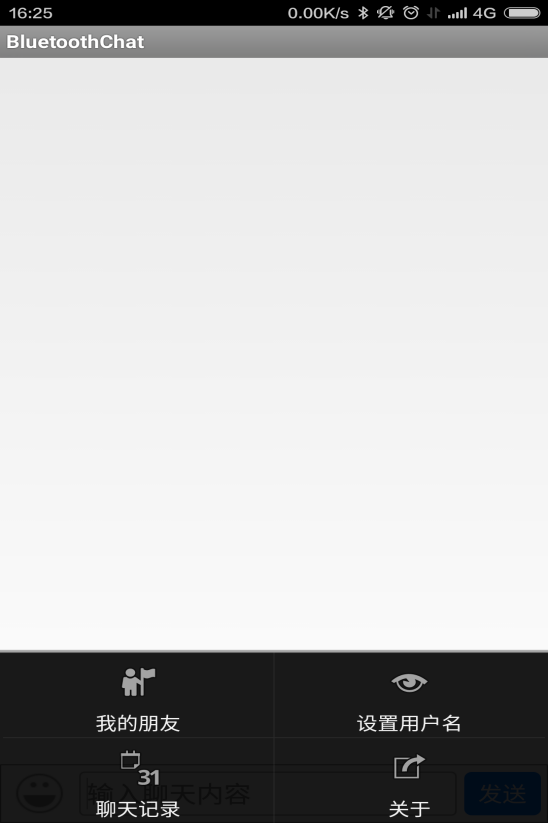


图5.2 Menu菜单栏

点击设置用户名，设置两台手机用户名分别为Jack和Rose，如图5.3所示。



图5.3 设置用户名

回到菜单栏点击我的朋友，进入扫描蓝牙设备界面，点击Scan Bluetooth Device对周围的设备进行扫描，得到设备列表中显示了扫描到的蓝牙设备的设备名和MAC地址，如图5.4所示。



图5.4 扫描周围蓝牙设备

在得到的蓝牙设备列表中点击Jack，即可与名为Jack的设备进行连接，连接成功后，聊天主界面会弹出Jack已经上线，此时两台手机就可以进行数据通信了，并且不同用户接受消息会有不同的声音提示。聊天界面如图5.5所示。



图5.5 聊天界面

此时返回Menu菜单点击聊天记录可以查看刚才的聊天记录，显示聊天内容和当时的时间记录，如图5.6所示。

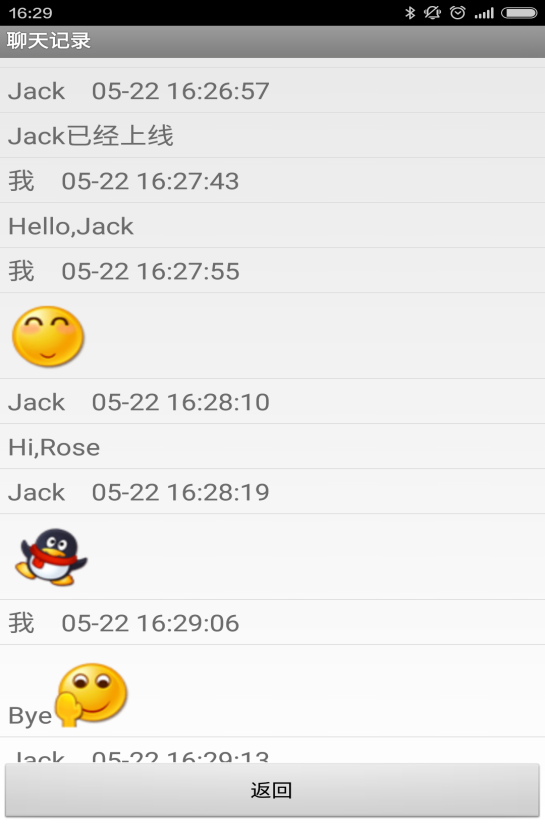


图5.6 聊天记录界面

最后点击Back返回键退出该应用APP时，会提示再按一次退出，使得该APP操作更加人性化，如图5.7所示。

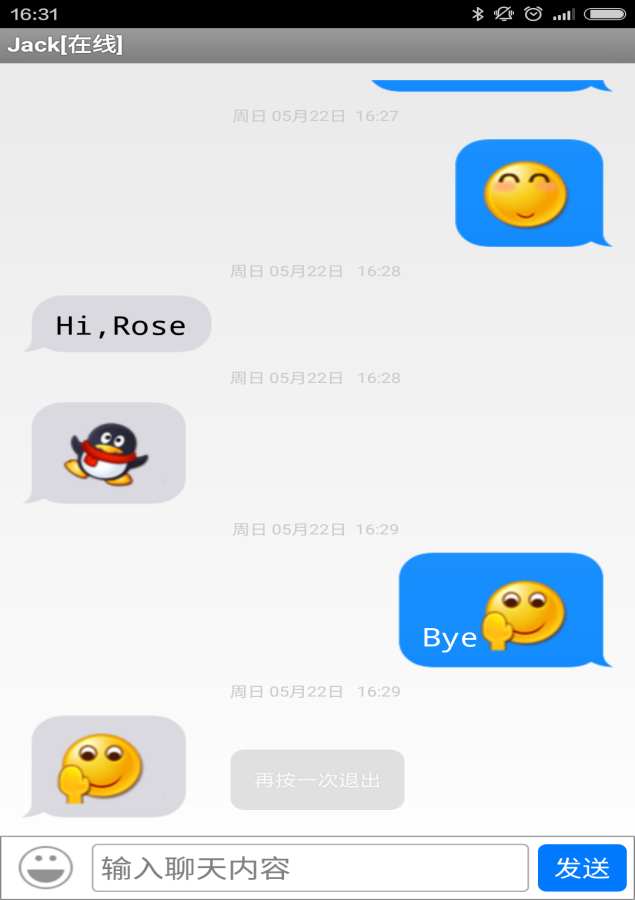


图5.7 提示再按一次退出

1. **设计体会**

本次系统设计是一个基于Android的短距离蓝牙数据通信设计，需要使用包括Java语言、安卓基础、蓝牙通信在内的相关知识，可以实现简单的安卓蓝牙数据通信，并且，本设计还可以实现发送表情，显示在线用户名，查看配对设备，查看聊天记录，设置本机用户名，再一次退出提醒等功能。

在此期间所做的工作包括：

1. 设计初期收集Java语言，安卓知识方面相关资料，对安卓开发有一定的了解。
2. 学习蓝牙通信，加深了对蓝牙操作，蓝牙之间通信方式的了解。
3. 参考官网上蓝牙的程序源码，确定最终的设计方案
4. 根据要实现的需求，在eclipes开发环境下应用Java语言进行编程，并且通过真机来测试手机界面程序。

从整个设计的过程来看前期的准备学习过程十分重要，经过前期充分的准备和对软件开发的学习，避免了很多在后续的程序设计中可能出现的问题。另外，此次设计还有很多可以完善的功能，比如可以进一步实现对文件和图片的传输并保存到手机上，实现用户注册登录的功能，以及实现多点对多点的多人聊天功能等，在后期时间允许的条件下将会对该程序进一步完善。