西安工业大学
XI'AN TECHNOLOGICAL UNIVERSITY

本科毕业设计(论文)

题目：野外营地安全监控终端-软件设计

院　（系）：　电子信息工程学院

专　　业：　通信工程

班　　级：　120412

学　　生：　杨福磊

学　　号：　120412113

指导教师：　王鹏

2016 年 06 月 01 日

西安工业大学
XI'AN TECHNOLOGICAL UNIVERSITY

本科毕业设计(论文)

题目：野外营地安全监控终端-软件设计

院　（系）：　电子信息工程学院

专　　业：　通信工程

班　　级：　120412

学　　生：　杨福磊

学　　号：　120412113

指导教师：　王鹏

2016 年 06 月 01 日

野外营地安全监控终端-软件设计

摘　要

户外野宿,俨然成为当代人们接触自然和享受生活的一种重要方式。但是,户外宿营由于其不可预知动物侵袭和自然灾害等危险,使得人们不能更好的亲近大自然。因此，本课题基于 zigbee 蓝牙及 android 上位机设计了一套功能完整、使用方便的的野外智能安防系统—营地安全卫士。本课题所使用的核心技术主要是 zigbee 上蓝牙模块的使用以及 nadroid 上位机手机 app 的开发,硬件部分主要是操作蓝牙发送相应的传感器数据和接受相应的数据并作出反应,实现上位机对于下位机的控制作用,所用蓝牙模块为 HC-06 蓝牙模块.并采用 zigbee 模块作为协调器和终端节点，终端节点负责数据的采集和发送给协调器，而协调器则负责汇总数据，组织数据格式并通过蓝牙发送给上位机。

上位机主要是实现对于蓝牙数据的接收、展示，接收主要采用的是 android 系统自带的蓝牙 socket 流，而展示数据除了用到了基本的用户界面，还用到了开源框架 linechart 进行展示。在进行展示的同时，也可以发送控制命令给蓝牙模块，实现对于下位机的功能控制。本课题基于蓝牙、android 上位机设计的营地安全卫士，能够较好的监测周围移动物体，且具有监测天气、烟雾、温湿度等功能，实现了较好的监测及预警功能。

关键词：zigbee；HC_06蓝牙；Android；Linechart

Field camp security monitoring terminal and the software design

Abstract

Outdoor wild nights, has become the contemporary people contact with nature and enjoy the life of a kind of important ways. However, outdoor camping due to its unpredictable animal attacks and natural disasters such as dangerous, make people can't better nature. Therefore, this topic is based on bluetooth and zigbee android PC designed a set of complete function, easy to use field intelligent security systems - the camp security guards. This topic the main core technology is used on the use of bluetooth module and zigbee nadroid PC app development, hardware part mainly operating the bluetooth to send the corresponding sensor data and accept the corresponding data and respond, to realize the upper machine under the control function of a machine used by the bluetooth module for HC - 06 bluetooth module. And USES the zigbee module as the coordinator and terminal node, terminal node is responsible for data collection and sent to the coordinator, the coordinator is responsible for summary data, organize data format and sent to the PC via bluetooth.

Upper machine mainly for bluetooth data receiving and display, receive mainly adopts is the bluetooth android socket stream, and display data in addition to the basic user interface, also use the open source framework linechart to display. On display at the same time, also can send control commands to the bluetooth module, implementation for the function of the machine under control. This topic is based on bluetooth, the android camp security guards of the upper machine design, can better monitoring around a moving object, and has the functions such as monitoring the weather, smoke, temperature and humidity, better monitoring and early warning function is realized.

Key words: zigbee; HC_06 bluetooth; Android. Linechart

目录

1 绪论

1.1 营地安全卫士简介

ZigBee是基于IEEE802.15.4标准的低功耗局域网协议。根据国际标准规定，ZigBee技术是一种短距离、低功耗的无线通信技术。这一名称（又称紫蜂协议）来源于蜜蜂的八字舞，由于蜜蜂(bee)是靠飞翔和“嗡嗡”(zig)地抖动翅膀的“舞蹈”来与同伴传递花粉所在方位信息，也就是说蜜蜂依靠这样的方式构成了群体中的通信网络。其特点是近距离、低复杂度、自组织、低功耗、低数据速率。主要适合用于自动控制和远程控制领域，可以嵌入各种设备。简而言之，ZigBee就是一种便宜的，低功耗的近距离无线组网通讯技术。ZigBee是一种低速短距离传输的无线网络协议。

Android则是由google公司开源的一款智能手机操作系统，其在手机市场上有着最大的用户群体。

蓝牙则是一种短距离的无线技术标准，用来让固定设置同移动设备间进行信息的交互。其采用的是短波特高频（UHF）无线电波。经由2.4-2.485Ghz的ISM评断进行通信。

随着计算机技术的不断发展和人们对于野外宿营安全认识的不断提高，开发这样一种安全、可靠的智能终端用来保障人们的宿营安全已经是大势所趋。而传感器网络在其他领域的成熟应用也为我们的课题提供了技术保证。总之，基于zigbee、蓝牙、android的营地安全卫士野外智能监控终端有着广泛的应用市场和广阔的发展前景。

1.2 户外安全研究概况

Zigbee在智能家居市场上有着广泛的应用，其开发最初就是用来在智能家居上发力的，当然其在农业、工业的监控及测量中也有着广泛的应用，但是就目前来看，其较少的使用在户外宿营安全问题的解决方案上。也就是说，在野外宿营安全的领域内这里是智能设备的一片真空领域。而智能硬件同手机之间的交互则要更为成熟一些，有很多公司都推出了自己的手机app同智能硬件绑定的产品，比较典型的如小米公司智能家居系列的智能手环，其就是基于蓝牙进行的底层通信。还有例如kiba等智能相机，你可以使用app控制哪些场景是需要拍摄和保存的。

总的来说，zigbee、蓝牙及android技术的应用层面广泛，是比较成熟的技术，但是在野外安全方面还是处于真空状态，并没有一款领导性的产品。国内外的研究概况既是如此。

1.3 本文研究主要内容

本课题设计的野外营地安全监控终端-软件设计主要是进行zigbee对于蓝牙的控制及接收蓝牙的控制信息，并且利用android平台强大的兼容及展示功能将底层的数据接入到android平台上进行展示，通过蓝牙实现上位机及下位机之间信息的无缝交换。

技术指标：

1）android蓝牙app的用户界面设计。

2）采集蓝牙发送的相关数据，并且进行展示，相对误差不超过1%。

3）熟练掌握linechart开源框架的使用。

4）zigbee协调器汇总终端节点传送的数据，并制定相应的便于上位机解析的数据格式。

5）使用蓝牙模块发送已经汇总的数据，至少1fps/s。

1.4 本设计的重难点

课题研究重点：根据系统功能要求，选择系统设计方案，包括系统硬件设计和系统软件设计。硬件主要涉及到蓝牙模块的选型，软件部分包括zigbee部分蓝牙控制及蓝牙信息接收及android手机接收数据及展示数据，最终android手机采集数据并成功展示及android手机发送控制指令给蓝牙模块，实现zigbee与android上位机之间的无缝数据交换。

难点：本课题的难点就在于android手机app的设计，如何保证在一边接收数据的情形下还能够发送数据，还有就是对于数据的展示，如何将接收到的数据实时的更新到用户界面上去。Zigbee与android手机之间蓝牙通信协议的制定等。

1.5 本论文的章节安排

本章开头主要讲的是本设计的背景以及本设计在国内外的发展状况，并且对涉及到的相关技术做了一些简单的介绍，说明了本设计的主要设计目标以及主要的技术指标，相关的研究内容也列举了出来。接下来则设计系统的硬件及其软件。在硬件的设计中，着重考虑的是HC-06蓝牙模块与zigbee模块的协同使用，详细的叙述各模块的功能和所选用的模块的介绍。在软件设计中，由于本设计比较特殊，故而首先需要采用用户界面设计软件设计app的大致界面，确立技术选型和基础的软件架构，目前确认的是app作为客户端，zigbee作为下位机，学习相关需要用到的语言及技术等。接下来绘制主要软件的流程图及分支的流程图，在开发完成之后进行测试，最终将硬件及软件部分进行联合调试，找出问题后进行修改，最终交付。

本论文共由6个章节组成，主要内容及结构安排如下：

第1章，绪论。首先介绍了基于zigbee、蓝牙、android营地安全监控终端的背景和意义，并对该课题在国内外的发展现状和研究概况有一定的了解。

第2章，系统总体设计方案。首先介绍了zigbee组网技术及蓝牙通讯技术，其次介绍了android蓝牙通讯的原理，在这些基础上绘制系统方框图，讲解系统

的构成及基本的技术架构。

第3章，系统硬件设计。说明所选择的蓝牙模块，以及选择该模块的原因。

介绍了所选择的蓝牙模块的技术参数、连接特性、基本原理，并介绍了zigbee模块的详情，最后给出具体设计方案。

第4章，系统软件设计。进行下位机蓝牙发送及接收数据的代码编写，上位机端蓝牙及用户界面相关代码的编写，并给出具体的软件设计的方案及详细的流程图介绍。

第5章，android程序与下位机进行联调。联调的目的是为了检测设计的程序的稳定性已经容错性。本章简单介绍了联调时候需要做的准备，并写出给出了联调的结果及与前面的技术指标进行对比，发现不足，进行修改。

第6章，设计总结。对本设计的过程和设计结果做出客观的总结和评价，主要包括设计结论，设计收获与体会，设计不足和需要改进的地方。

2 系统方案论证

常用的下位机与android上位机进行数据交互的方法主要有串口、蓝牙、wifi通信，本课题采取的是蓝牙进行相关通信链路的设计。下位机采取zigbee模块作为协调器作为整个系统的主要硬件，利用其控制蓝牙模块实现我们想要的功能。软件部分采取的技术架构是：上位机作为客户端，是同用户交流的窗口，其上使用Java进行应用层的编写，而下位机的蓝牙部分采取c语言进行编写。

下面，我们一起来看看，android系统的基本框架：

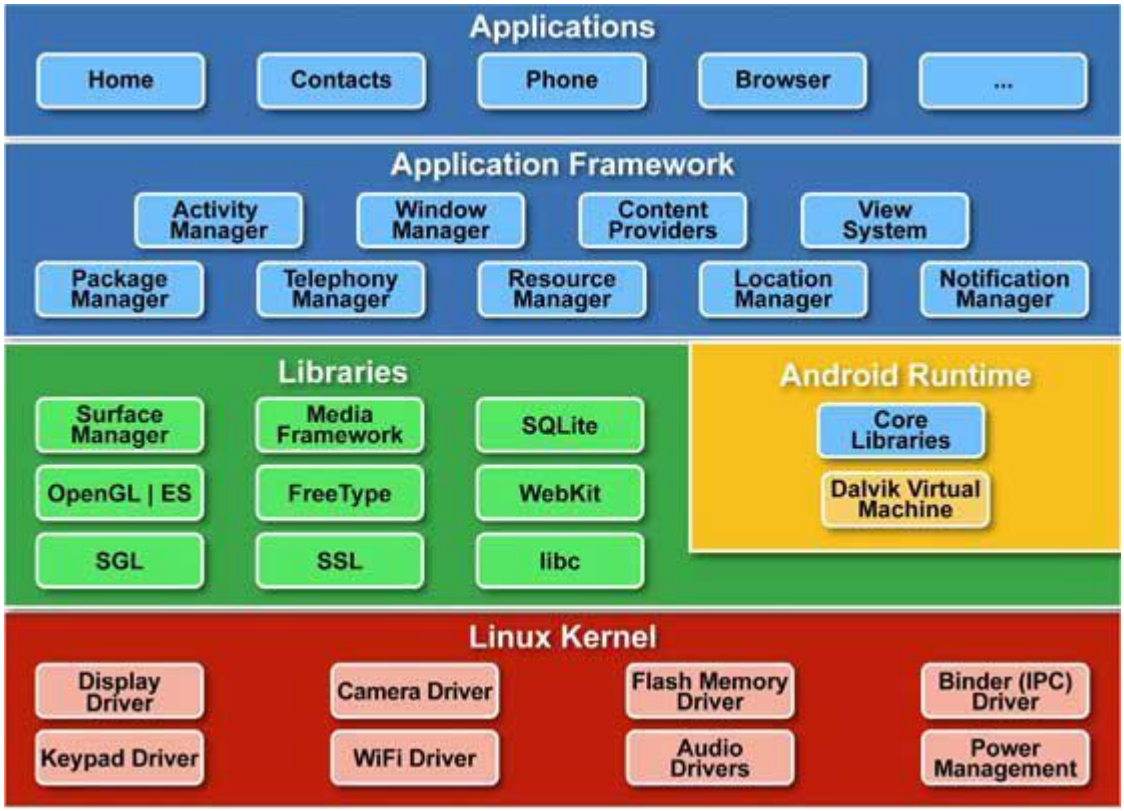


图2.1 android系统基本架构

从上图可以清晰的看到，android系统的基本架构主要分为四个部分，如下：

1） Linux Kernel。

本部分主要是基于Linux的内核提供系统服务，例如：安全、内存管理、进程管理、网络堆栈、驱动模型等。其也作为软件与硬件之间的抽象层，它隐藏了具体的硬件细节而为上层提供统一的服务。

本层是高内聚、低耦合的代表。

2） Android Runtime。

本部分主要是提供核心库的集合，提供大部分在java编程语言核心库中所用到的功能，每一个android应用程序都是dalvik虚拟机中的实例。

3） Libraries。

此部分主要包含c/c++库的集合，供android系统的各个组件进行调用。这些功

能通过Android的应用程序框架暴露给开发者。

4） Application Framework。

通过提供开放的开发平台，Android使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等。

5） Applications。

Android装配一个核心应用程序集合，包括电子邮件客户端、sms程序、日历等，所有应用程序都是采用java语言编写的，当然我们的主要工作也在这一层。

2.1 HC-06 模块分析

HC-06 模块发送和接收数据主要是通过串口进行的,我们并不关心其内部实现,关心的是它对外提供的接口.

串口分为全双工、半双工、单工等多种类型。

而经过资料的查找,HC-06 蓝牙模块的串口为半双工串口，也就是在同一时刻，要么是在发送，要么是在接收。

下面，简述蓝牙串口的工作原理：

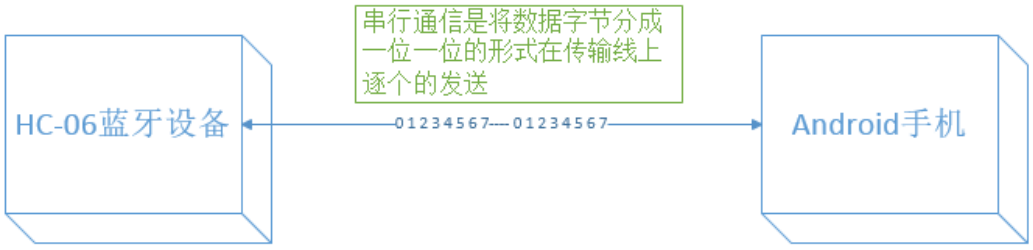


图 2.2 蓝牙串口发送原理图

通过上图我们可以看出，串口的发送一次只能发送一位数据，而且必须是按照一定的时序进行发送的，且蓝牙串口采用的是异步的通信格式，也就是说，在 android 上位机和 zigbee 下位机之间，没有一个统一的时钟。

在数据进行发送的时候，其是以字符（构成的帧）为单位进行传输，字符与字符之间的时间间隔是任意的，但每个字符中的各位是以固定的时间发送的，即字符之间不一定有”位间隔”的整数倍的关系,但同一字符内的各位之间的距离均为”位间隔”的整数倍。在后面的 android 上位机接收的时候我们就遇到了这样的问题。

串口的发送是可能出错的，如果出现了错误我们怎么能知道呢？

串口协议的标准为我们提供了这样几种校验数据的方式，如奇偶校验、代码和校验、循环冗余校验等。当然，串口还有着更多的技术参数，如传输速率、传输距离等。

常见的串口标准有：RS-232、RS-485、USB 等。

2.2 android 蓝牙分析

上面我们在方案的论证阶段已经看到了 android 操作系统的大体架构，我们具体的操作就是 Applications 层，在这一层上我们可以自由的构建我们的应用程序。

而 android 官方的 sdk 就已经为我们封装好了操作 android 设备底层设备的接口，在 java 里面的话大致提供了如下几个类，掌握了这些，我们就可以操纵蓝牙。如图 2.3 所示：

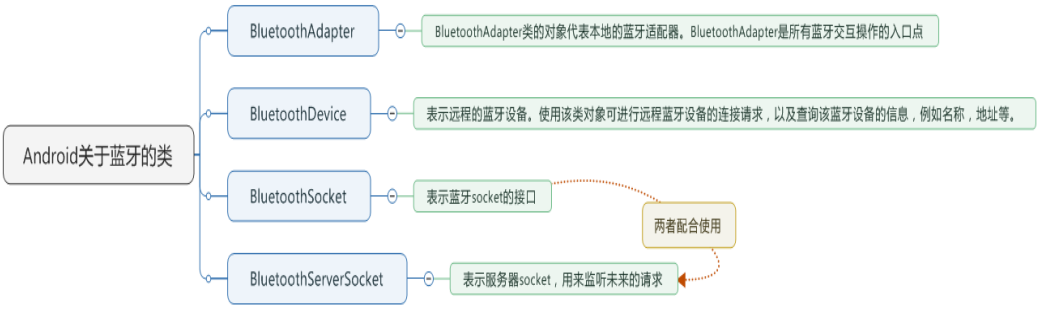


图 2.3　android 系统操作蓝牙所需的类

从上图我们可以看出,主要有这样几个类,那就是 BlutoothAdaptor、BlutoothDevice、BlutooothSocket 和 BlutoothServerSocket.

而 BluetoothAdaptor 是所有交互的起点，主要有如下功能：

功能编号	功能
1	发现其他蓝牙设备
2	查询已配对的设备
3	通过已知的 MAC 地址实例化远程蓝牙设备
4	创建 BluetoothServerSocket 类进行监听

表 2.1 BluetoothAdaptor 功能对照表

综合上面的图表我们可以得出这样的一个结论，那就是 android 蓝牙的底层通信就是 socket 通信，不过是把基本的 socket 通信绑定到了具体的蓝牙设备之上，接下来，我们看一下 socket 通信的概念：

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个 socket。

Socket 的英文原义是“孔”或“插座”。作为 BSD UNIX 的进程通信机制，取后一种意思。通常也称作"套接字"，用于描述 IP 地址和端口，是一个通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。在 Internet 上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个 Socket，并绑定到一个端口上，不同的端口对应于不同的服务。Socket 正如其英文原意那样，像一个多孔插座。一台主机犹如布满各种插座的房间，每个插座有一个编号，有的插座提供 220 伏交流电， 有的提供 110 伏交流电，有的则提供有线电视节目。 客户软件将插头插到不同编号的插座，就可以得到不同的服务。

这是对于 socket 的描述，下面我以一幅图来说明 socket 处理信息的过程：

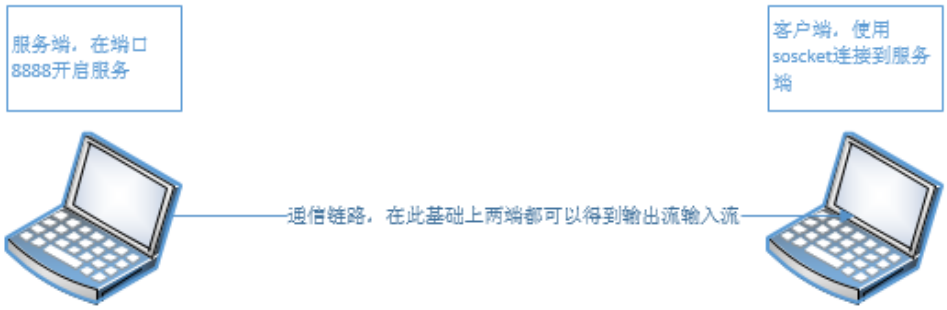


图 2.4 系统硬件框图

如上图所示，socket 通信中有两个重要的角色，分别是服务端和客户端，服务端主要是开启服务，其在 javase 中使用的是 Serversocket 类，而客户端使用的主要是 socket 类，这两个类配合使用可以实现客户端与服务端之间数据链路的建立。

上面提到，我们可以从数据链路中得到数据流，分为输出流和输入流两种，输入流和输出流起着不同的作用。下面分别予以介绍。

输入流我们可以从里面读取数据，上面的数据链路就相当于在服务端和客户端之间架立了一个数据的通道，我们的数据可以在这个通道之中自由的传输，在传输的过程中某一个时刻，一定是一端在发，一端在读，输入流就是从对方传送过来的数据进行读取。

输出流就是我们可以在里面写入数据，另一端即可以读取我们写入的数据，但是输出流和输入流的概念是相对的，也就是说，服务端的输入流对应的数据就是客户端的输出流。

综上所述，我们可以得到：如果我们要在 HC-06 和 android 手机之间建立通信链路，我们需要知道谁扮演着客户端，谁扮演着服务端的角色。经过资料的查询，得到的是，HC-06 在连接中扮演的是服务端的角色，所以我们的 android 上位机主要就是客户端。

我们就可以利用上面提到的 android 操作系统为我们提供的访问底层硬件设

备的接口进行设备的访问。

在得到了具体的通信链路之后，我们就可以利用这个通信链路得到输入流和输出流实现对于蓝牙数据的接收和对蓝牙发送我们的控制指令。

当然，在这个过程中我们需要制定相关的通信协议，因为数据只有在有背景的情况下应用才是有意义的。

2.3 设计系统框图

本设计方案如图 2.3 所示，主要由三个终端节点（zigbee 检测节点）及一个主控模块（zigbee 协调器）组成。检测节点上主要放置各种传感器，如红外热释电、sht11、yl83、烟雾等传感器。 协调器主要负责采集这些数据并将检测节点发送的数据进行汇总，并组织为有规律的格式后发送给 android 手机，方便 android 手机的解析。蓝牙在接收数据的过程中需要和 android 上位机之间约定好通信协议，这样在蓝牙接收到相关指令的时候蓝牙就可以做出相应的动作。

Android 手机主要负责的是对下位机的控制及所收取数据的展示。

Android 手机上负责数据展示的是开源的 linechart 框架，后面会有详细的介绍。

我主要负责的是红色标识的部分,主要是 zigbee 下位机蓝牙和上位机 app 的设计.如下图所示:

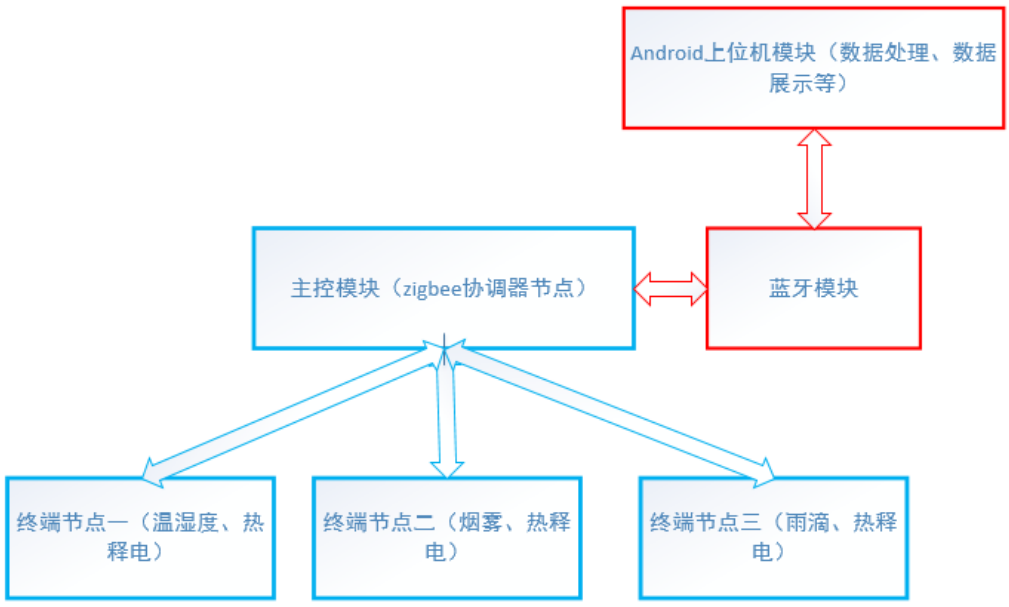


图 2.5 系统硬件框图

3 zigbee 蓝牙部分设计

3.1 蓝牙模块设计

3.1.1 HC-06 蓝牙模块介绍

蓝牙模块的核心采用 HC-06 从模块，引出接口包括 VCC、GND、TXD、RXD，预留 LED 状态输出脚，单片机可通过该脚状态判断蓝牙是否已经连接。LED 指示蓝牙连接状态，闪烁表示没有蓝牙连接，常亮表示蓝牙已连接并打开了端口，底板 3.3V LDO，输入电压 3.6~6V，未配对时电流约 30mA，配对后约 10mA，输入电压禁止超过 7V；接口电平 3.3V，可以直接连接各种单片机；配对以后当全双工串口使用，无需了解任何蓝牙协议，如图 3.1 所示：

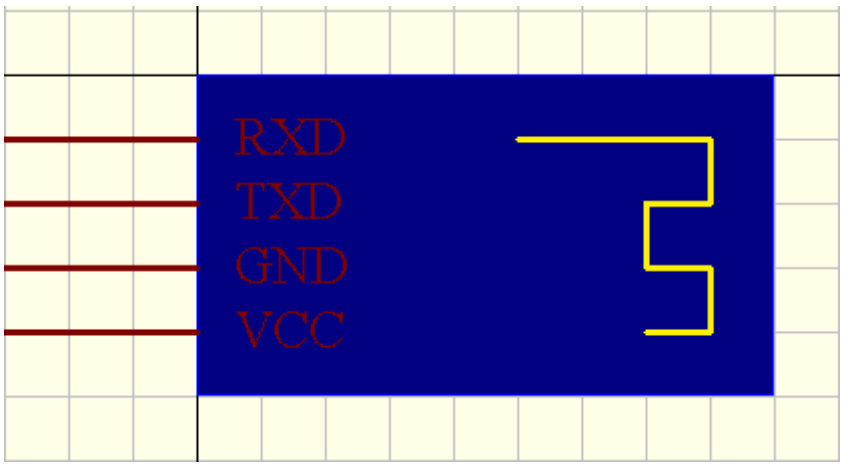


图 3.1 HC-06 模块管脚图

管脚号	名称	功能
1	VCC	电源端（3.6~6V）
2	GND	接地端
3	TXD	串行数据发送端
4	RXD	串行数据接收端

表3.1 HC-06管脚图对应表

手机蓝牙与 HC-06 蓝牙模块进行连接通信，进而通过蓝牙模块控制主控模块进行功能调试。Andriod 手机上安装用来控制蓝牙模块进行通信的 App，不需要了解复杂的蓝牙通信协议，只需了解单片机与蓝牙进行通信时的全双工串行通信形式即可完全会使用蓝牙模块。
本次的全双工串行通信形式采用串行通信方式 1，利用串行中断接收和发送数据。

3.1.2 HC-06 模块技术参数

Ø 灵敏度(误码率)达到 －80dBm

Ø -4 -> 6dBm 功率可调输出

Ø 蓝牙 2.0 带 EDR， 2Mbps-3Mbps 调制度

Ø 内置 2.4GHz 天线, 用户无需调试天线

Ø 外置 8Mbit FLASH

Ø 低电压 3.3V 工作（3.1V~4.2V）配对时 30～40MA 波动，配对完毕通信 8MA

Ø 可选 PIO 控制

Ø 标准 HCI 端口(UART or USB)

Ø USB 协议: Full Speed USB1.1, Compliant With 2.0

Ø 模块可以作为 SMD 贴片工艺

Ø RoHS 制程

Ø 引脚半孔工艺

Ø 数字 2.4GHz 无线收发射

Ø CSR BC04 蓝牙芯片技术

Ø 自适应跳频技术

Ø 体积小,(27mm×13mm×2mm)

Ø 简单的外围设计电路

Ø 蓝牙 Class 2 功率级别

Ø 存储温度：-40 至+85 度，工作温度：-25 至+75 度

Ø 协波干扰：2.4MHz，发射功率 3 dBm

Ø 误码率：0，但会在传输链路产生信号衰变，才有误码，如 RS232 和 TTL 线路处理线路

3.1.3 HC-06 模块的测试和 AT 指令

在刚拿到蓝牙模块的时候，本人需要对蓝牙模块进行简单的测试，这个时候需要用到简单的一个测试方法，那就是回环测试。

回环测试的具体方法是：

1：将蓝牙设备的RXD和TXD管脚短接，即实现自发自收。

2：将蓝牙设备用串口助手连接到电脑上，

3：采用手机串口助手连接我们的蓝牙设备，并发送数据。

4：观察接收区是否收到相应的数据，如果收到，则视为测试成功，蓝牙设备正常。

常用的AT指令如下：

进入AT指令的方法：给模块上电 ，不配对的情况下，就是 AT模式了。指令间隔

1S左右
出厂参数波特率 9600 名字 linvor ，密码1234
1 、 测试通讯
发送：AT（返回 OK，一秒左右发一次）
返回：OK
2 、 改蓝牙串口通讯波特率
发送：AT+BAUD1
返回：OK1200
发送：AT+BAUD2
返回：OK2400
3 、改蓝牙名称
发送：AT+NAMEname
返回：OKname
4 、改蓝牙配对密码
发送：AT+PINxxxx
返回：OKsetpin
参数 xxxx：所要设置的配对密码，4 个数字，此命令可用于从机或主机。从机是适配器或手
机弹出要求输入配对密码窗口时， 手工输入此参数就可以连接从机。 主蓝牙模块搜索从机后，
如果密码正确，则会自动配对，主模块除了可以连接配对从模块外，其他产品包含从模块的
时候也可以连接配对，比如含蓝牙的数码相机，蓝牙 GPS，蓝牙串口打印机，等等。

3.2 zigbee 模块设计

3.2.1 zigbee 模块的简介

一种典型的WSN网络协议就是ZigBee网络。ZigBee是基于IEEE802.15.4标准的低功耗局域网协议。根据国际标准规定，ZigBee技术是一种短距离、低功耗的无线通信技术。这一名称（又称紫蜂协议）来源于蜜蜂的八字舞，由于蜜蜂(bee)是靠飞翔和“嗡嗡”(zig)地抖动翅膀的“舞蹈”来与同伴传递花粉所在方位信息，也就是说蜜蜂依靠这样的方式构成了群体中的通信网络。主要适合用于自动控制和远程控制领域，可以嵌入各种设备。
。ZigBee协议层从下向上分别为物理层（PHY）、媒 体 控 制 层（MAC）、网 络 层（NWK）、应 用 层（APL）。其中MAC和PHY由IEEE 802.15.4标准定义，ZigBee联盟定义了NWK和APL。一般ZigBee网络支持星型、树型和网状型网络拓扑结构，
每个ZigBee网络必须有一个协调器作为父节点，其他的路由节点和终端节点作为

子节点加入网络。协调器是整个网络的中心，它的功能包括建立、维持和管理网络，分配网络地址等。所以可以将ZigBee网络协调器认为是整个ZigBee网络的“大脑”。路由器主要负责路由发现、传输消息、允许其他节点通过它接入到网络。终端节点通过ZigBee协调器或者ZigBee路由器接入到网络中，ZigBee终端节点主要负责数据采集或控制功能，但不允许其它节点通过它加入到网络中。

ZigBee提供了三级安全模式，分别是：无安全设定，使用访问控制清单防止非法获取数据和采用高级加密标准的对称密码。

ZigBee开放的频段有欧洲868MHz（带宽2MHz），915MHz（带宽0.6MHz）和2.4GHz（带宽5MHz），他们都属于免费开放的ISM（工业科学医疗）频段。

ZigBee具备每个网络65000个节点的组网规模以及自组织网络的能力。

总结起来就是：低功耗、低成本、低速率、近距离、短时延、自组织大规模、高安全和免费频段。

3.2.2 zigbee 的应用场景

ZigBe 广泛应用于家庭，工业自动化，交互式设备以及遥控、遥测设备中，常见的有：智能家居、火灾监测、粉尘监测、工业照明以及遥控飞行器和惯性导航定位

而随着我国物联网正进入发展的高峰期，ZigBee 正在一点一点的被市场所接受。该项技术也已经在部分智能传感器场景中进行了实际应用。例如北京地铁 9 号线隧道施工过程中的考勤定位系统便采用的是 ZigBee，ZigBee 取代传统的射频考勤系统实现了无漏读、准确的方向判断、准确的定位轨迹和可查询性，提高了整个地铁隧道安全施工的管理水平；在某些高档老年公寓中，使用 ZigBee 网络的无线定位技术设计制作的简单、可靠便携式移动报警器，可以在老人遇到险情的时候帮助老人及时获得救援。

随着我国物联网技术的飞速发展，ZigBee 技术的应用将会越来越多。

3.2.3 zigbee 模块与蓝牙连接

Zigbee 作为一个通信平台,是我们本次设计的重点,而我们最重要的工作是如何将蓝牙模块整合到 zigbee 上去,通过前面的分析我们知道,HC-06 蓝牙模块的特点是使用串口来进行读写,也就是说,我们可以将 HC-06 蓝牙模块简单的想象为一个串口最终实现蓝牙数据的发送和接收.

而任何一个单片机系统在连接任何一个需要使用串口的外设部件的时候都需要进行初始化,初始化的部分我们放在后面软甲设计的部分来讲.

首先给出 zigbee 底板的原理图如图 3.2 所示：

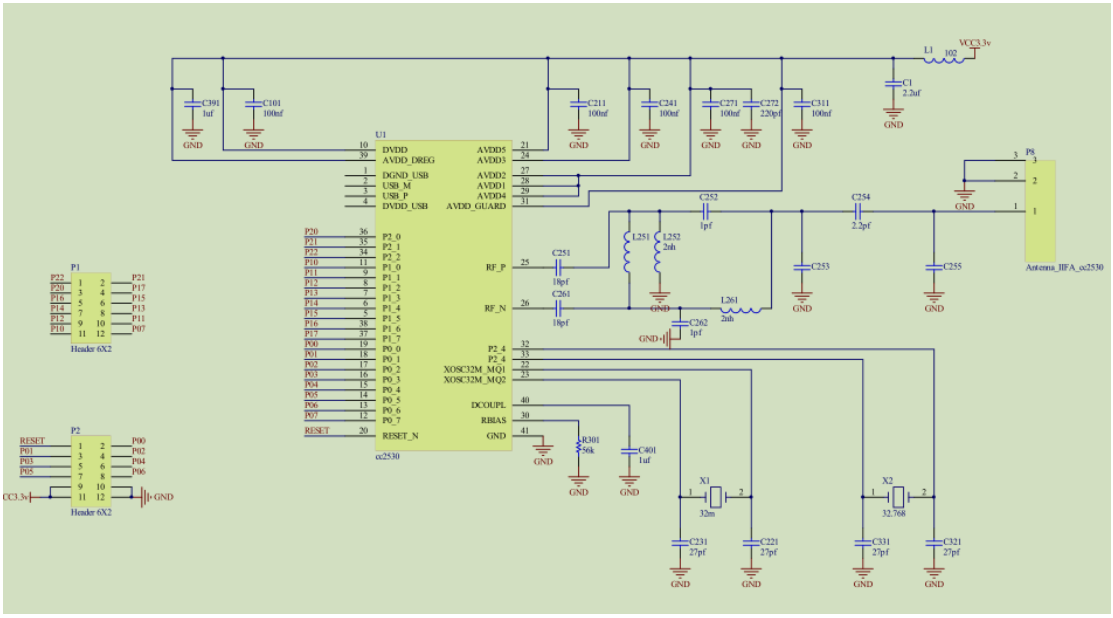


图 3.2 zigbee 模块原理图

给出了 zigbee 模块的原理图之后,我们需要确认的是蓝牙模块如何与 zigbee 模块进行连接,如图 3.3 所示：

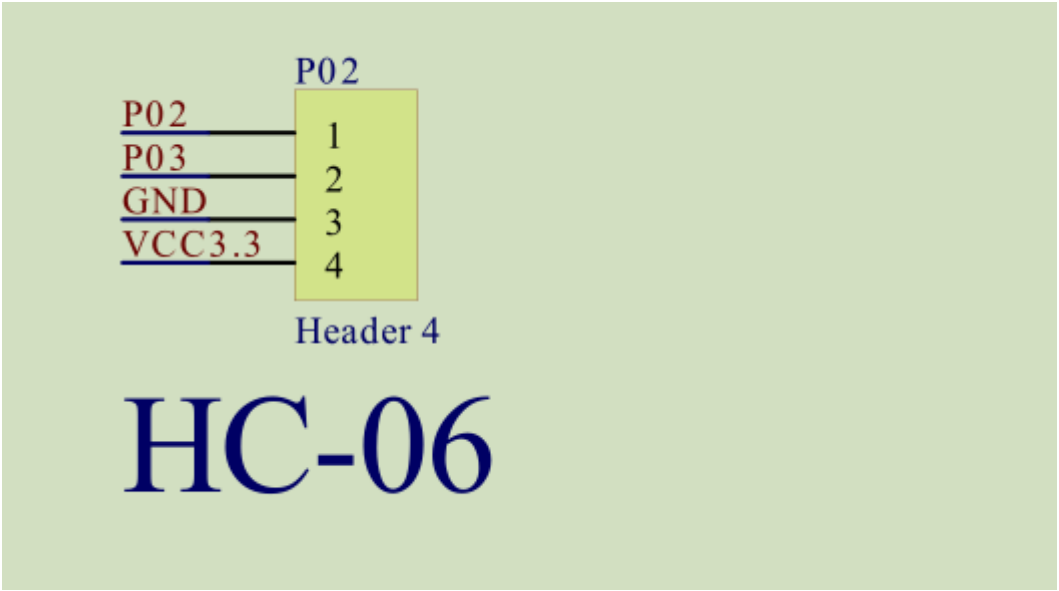


图 3.3HC-06 蓝牙模块端口对应图

从上面我们可以看出,H-06 与蓝牙之间的连接对应关系,在板子上我们需要这样连接,实现蓝牙数据的发送与接收.

3.3 蓝牙通信协议

3.3.1 必要性

制定通信协议是一件非常有必要的事情，举一个简单的例子来说，我们听不懂英语，但是借助翻译软件我们就能懂。

而如果没有有效的通信协议的话，zigbee和android上位机之间就存在着相互之间无法了解对方确切想要表达的信息，因此制定一个完善的通信协议是非常重要的。

有了通信协议之后，我们就可以实现在蓝牙设备和android上位机之间通畅的交流信息。

3.3.2 通信协议

通信协议的制定主要包含两个部分，第一个部分是蓝牙设备发送给 android 上位机的数据的数据格式的制定，如下表 3.2 所示：

DHT11 节点													
mac 地址								红外标志	红外数据	温度高八位	温度低八位	湿度高八位	湿度低八位
0x2B	0x28	0x2B	0x06	0x00	0x4B	0x12	0x00	0xAB	0xCD(没人)	0x21	0x00	0x17	0x00
									0xEF(有人)	,,,			
烟雾与雨滴节点													
mac 地址								红外标志	红外数据	烟雾标志	烟雾数据	下雨数据标志	下雨数据
0x92	0xD7	0x60	0x02	0x00	0x4B	0x12	0x00	0xAB	0xCD(没人)	0xFF	0xEE(没烟)	0xBB	0xAA(没雨)
									0xEF(有人)		0xDD(有烟)		0xCC(有雨)

表 3.2 蓝牙发送数据数据格式对照表

第二个部分是 android 上位机给下位机发送的控制指令的列表，如表 3.3 所示：

命令名称	具体指令
开启灯光	On
关闭灯光	Off
开启报警	Alarmon
关闭报警	Alarmoff
开启娱乐	exerciseon
关闭娱乐	exerciseoff

表 3.3 上位机命令一览表

如上表 3.2、表 3.3 将蓝牙发送的数据格式以及 android 上位机发送的命令即对应的数据格式制定出来，让蓝牙设备同 android 上位机之间的相互通信变得毫无阻碍。通信协议的制定是要遵守一定的规范的，比如说：

1:给上位机发送的数据消息要有标志位将各部分的数据进行分割,方便上位机进行有效的数据分割和数据处理,如果没有有效的标志位的话,上位机又如何进行对数据的有效处理呢?
2:上位机给下位机发送的数据应该尽可能的简单,方便下位机的处理,因为下位机使用的是 c 语言并没有字符串这类的基本变量.
综上,通信协议的制定在这里扮演着非常总要的作用.

3.4 本章总结

本章内容主要介绍了蓝牙模块、zigbee 模块，这两个模块的联合应用在本课题中起着很重要的角色，在最后介绍了蓝牙设备同 android 上位机之间的通信协议。

4 软件设计

4.1 上位机界面设计

Android上位机的编写主要涉及两个部分,一部分是逻辑代码的编写,另一部分是界面的设计,传统的界面设计就是在android中设计完成后看是否合适,不合适的话再次修改,但是这样的反馈速度未免太慢.新的设计方式是这样的,那就是强大的原型设计工具,axure.首先,简单的介绍一下axure:

4.1.1 axure 介绍

Axure RP是一个专业的快速原型设计工具。Axure（发音：Ack-sure），代表美国Axure公司；RP则是Rapid Prototyping（快速原型）的缩写。

Axure RP是美国Axure Software Solution公司旗舰产品，是一个专业的快速原型设计工具，让负责定义需求和规格、设计功能和界面的专家能够快速创建应用软件或Web网站的线框图、流程图、原型和规格说明文档。作为专业的原型设计工具，它能快速、高效的创建原型，同时支持多人协作设计和版本控制管理[1] 。Axure RP已被一些大公司采用。Axure RP的使用者主要包括商业分析师、信息架构师、可用性专家、产品经理、IT咨询师、用户体验设计师、交互设计师、界面设计师等，另外，架构师、程序开发工程师也在使用Axure。

那么,我们使用axure设计的优势都有哪些呢?

1:试错成本低,原有的开发方式只有当产品的开发周期结束之后,才可以进行检验,而现在,原型图帮你解决这个烦恼.

2:各部门协同方便,再也不用担心话说不清楚了.

3:axure支持多种插件库,丰富的插件库一定让你爱不释手.

4：axure设计出来的原型图支持多种展示方式，甚至可以制作高保真原型图，让你一开始就能做出最终产品的样子来。

4.1.2 设计流程

主要的设计流程如下：

1：安装 axure 软件，现在的 axure 软件已经更新到了 8.0.

2：打开 axure 后新建工程，注意，我们这里需要设计的是 android 的原型图，而 axure 自带的是网页设计的原型模型，我们需要自己载入 android 的库，但是库需要我们自己下载。

3：装载我们下载的 android 原型模型。

4：开始绘制产品的草图。

5：完成草图的绘制后我们可以进行精致的修正，确保我们输出的图与最后成型产品差距要小。

6：完成设计。

4.2 上位机蓝牙部分设计

4.2.1 确认蓝牙状态

在开始正式的编写蓝牙代码之前，我们必须首先确认本地的手机是否拥有蓝牙和蓝牙是否开启，如果蓝牙未开启的话我们需要提示用户开启蓝牙。而开启蓝牙的话,我们必须要获取具体的系统权限才可以,因为 android 操纵系统对于处于应用层的程序有着严格的权限限制.

下面,我们需要进行的是从判断蓝牙设备是否启用到启用蓝牙设备,如图 4/。1 所示：

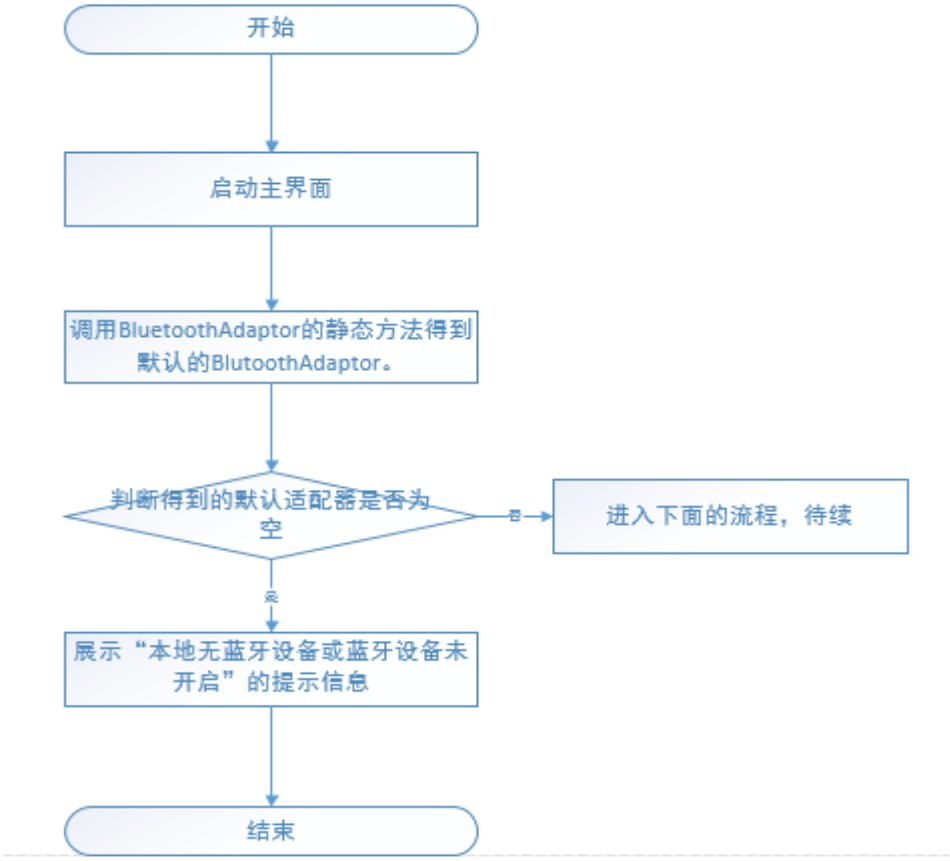


图 4.1 验证设备蓝牙流程图

在上面讨论蓝牙的四层技术架构的时候我们已看到 BlutoothAdaptor 是我们编写蓝牙程序的起点,通过这个类我们可以看到本地是否拥有蓝牙设备以及蓝牙设备是否已经启动.下面给出具体的代码展示:

```
//  获取本地的蓝牙适配器
mBluetoothAdapter  =  BluetoothAdapter.getDefaultAdapter();
```

```
//  判断蓝牙设备是否已经启用
if  (mBluetoothAdapter  ==  null)  {
Toast.makeText(MainActivity.this,  "蓝牙设备未启用或本地无蓝牙设备!",
Toast.LENGTH_SHORT).show();
finish();
return;
}
```

4.2.2 建立 socket 连接

上面我们已经通过 BlutoothAdaptor 类确认了蓝牙设备的开启状态和本地是否有蓝牙设备，下面我们就要以具体的设备为例来进行 socket 连接，建立通信信道.

但是，在这里又出现了一个问题，那就是我们的界面其实是运行在一个线程之上的，而这个线程就是我们俗称的主线程，蓝牙的连接是一个十分耗时的操纵，我测试了一下，建立 socket 信道的平均时间都在一秒以上，也就是说，在这一秒，我们的界面会处于一种假死状态，是因为耗时操作-蓝牙 socket 信道的建立暂时阻塞了.所以我们必须想出一种方法来即让我们的操作和界面的状态更新处于并行状态.这就用到了并发技术，即多线程技术.

在多线程技术使用的过程中我们一定要注意一点，并发有可能导致死锁，而一旦死锁这将对我们的产品的体验带来最糟糕的体验。

那么也就是说，我们需要在线程之上另外启动一个甚至多个线程，负责我们耗时处理的操作.如图 4.2 所示：

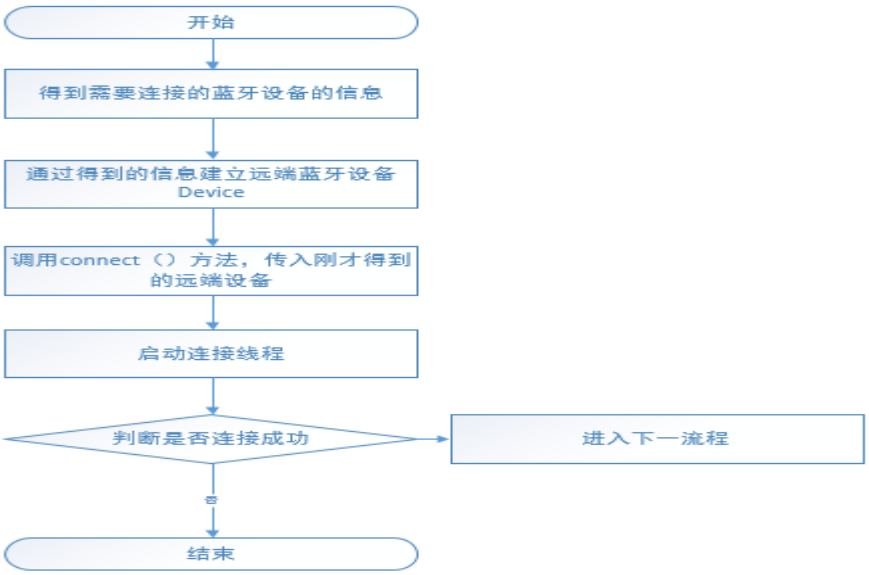


图 4.2 连接蓝牙主流程图

通过上面我们的流程我们可以看到获取远端蓝牙设备的基本信息是我们建立连接的关键,而比较重要的信息都包括那些信息呢?

首先,mac 地址是我们必须知道的,这是对一个蓝牙设备唯一标示的地址,其次,蓝牙的名字我们也需要知道,因为后期的话我们需要做一下展示.

下面是具体的代码:

```
String info = data.getExtras().getString(
DeviceListActivity.EXTRA_DEVICE_ADDRESS);
// 蓝牙设备名称
name = info.substring(0, info.length() - 17);
System.out.println(name);
// 根据 MAC 地址获得蓝牙设备
address = info.substring(info.length() - 17);
BluetoothDevice device = mBluetoothAdapter
.getRemoteDevice(address);
connect(device);
```

Connect()方法如下:

```
/**
 * 根据 MAC 地址连接设备
 *
 * @param device
 *                          根据蓝牙 MAC 地址得到的硬件设备
 */
private void connect(BluetoothDevice device) {
Log.d(TAG, "connect to " + device);
// 耗时操作另外开启一个线程
new ConnectThread(device).start();
}
```

当建立连接以后,我们就可以通过连接得到我们需要的数据流,实现 android 上位机和 zigbee 下位机之间的信息交流.代码如下:

```
public ConnectedThread(BluetoothSocket socket) {
Log.d(TAG, "create ConnectedThread");
mmSocket = socket;
InputStream tmpIn = null;
OutputStream tmpOut = null;

// 得到输入流和输出流
try {
tmpIn = socket.getInputStream();
```

```
tmpOut = socket.getOutputStream();
} catch (IOException e) {
Log.e(TAG, "temp sockets not created", e);
}

mmInStream = tmpIn;
mmOutStream = tmpOut;
}
```

Socket 连接的代码写到这里我们就可以得到输入流和输出流,实现与下端蓝牙设备的数据通信.

4.2.3 远端蓝牙设备的展示

想要连接蓝牙，我们首先要打开本地的蓝牙设备，开始对有效距离内的蓝牙进行搜索，得到的搜索结果的展示也是一个问题，但是 android 为我们提供了一个现成的控件，那就是 ListView，通过 ListView，我们可以很方便的将我们的蓝牙设备展示到屏幕上同用户进行交互，当然，我们还需要为 listview 设置事件的监听得到用户的反馈，比如说用户选中了那个蓝牙设备，我们再返回上一级主界面的时候就必须将用户选择的结果同时返回回去。

如图 4.3 所示：

图 4.3 listview 蓝牙展示

上面，我们可以看到，蓝牙设备已经被展示在了具体的 ListView 中去，下面，我们来看看具体的流程：

4.3 数据处理部分设计

Android上位机部分我们采用的是Java语言，java是一种面向对象的高级语言，一般情况下使用java是不需要关注底层实现的，但是，我们从前面制定的通信协议来看,我们的底层传输过来的数据格式是16进制的数据,而在传输的底层,其是作为字节的形式一字节一字节的发送的,java的字节流取到的数据都是字节,我们要做的就是将其转为我们方便进行分辨的格式并根据标志位对其进行解析.而对数据的解析也就成为了我们重要工作.下面,展示解析的具体流程:

我们从socket连接中取到的字节流输入流中取得的都是字节流,当我们一个一个字节将读出来的数据放到一个字节数组中的时候,我们首先将其解析成了十六进制的字符串,之后,在进行进一步的解析,当然,只是部分数据需要进一步的解析,如温度湿度等等.因为16进制的数据显得很不直观。

如图4.4所示：

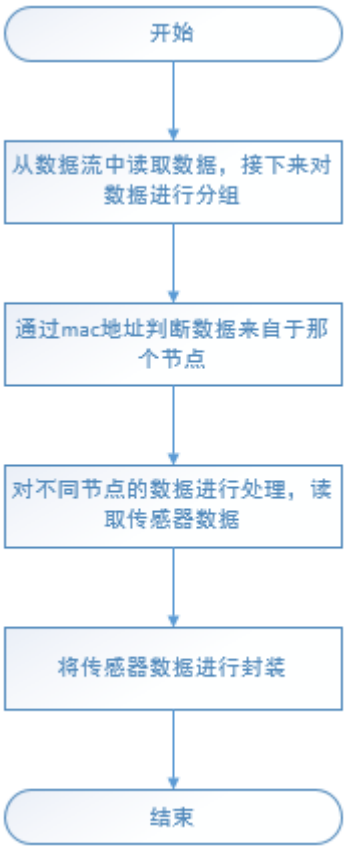


图 4.4 数据处理流程图

但是要将16进制的数字转换为十进制的数字，java当中并没有线程的类库，于是我们就只能自己编写一个方法了。下面是具体的接收信息的代码：

```
int bytesLength = 0;
StringBuffer res = new StringBuffer();
byte[] buffer = new byte[1];
while (true) {
try {
for (int i = 0; i < 14; i++) {
mmInStream.read(buffer);
res.append(CommonUtil.bytesToHexString(buffer));
}
// System.out.println("数据为:" + res.toString());
String macAdress = res.substring(0, 16);
String result = res.substring(16);
// 判断是哪个节点的数据，对应节点进行对应的解析
if (macNode1.equalsIgnoreCase(macAdress)) {
bluData.setTemperature(CommonUtil.hexToTen(result
.substring(8, 10)));
bluData.setHumidity(CommonUtil.hexToTen(result
.substring(4, 6)));
// 有无人是两个节点都要更新的
bluData.setHavePerson(result.substring(2, 4) + "-"
+ macAdress);
} else if (macNode2.equalsIgnoreCase(macAdress)) {
bluData.setHaveRain(result.substring(10, 12));
bluData.setHaveSmoke(result.substring(6, 8));
// 有无人是两个节点都要有更新的
bluData.setHavePerson(result.substring(2, 4) + "-"
+ macAdress);
}
// System.out.println(result+"-----"+macAdress);
System.out.println(bluData.toString());
Thread.sleep(100);
// 性能问题
res = new StringBuffer();
} catch (IOException e) {
e.printStackTrace();
} catch (InterruptedException e) {
e.printStackTrace();
```

```
}
}
```

通过上面的代码我们可以看出，按照字节流->16进制->10进制的方法转换过后，我们将最终得到的结果保存到了一个类的内部，而这个类就是我们储存数据的地方，下面我们会进行详细说明。

4.4 数据存储及展示部分

当我们接收到从 zigbee 下位机传输过来的数据之后，我们可以直接将数据经过一定规则的处理之后就将其展示给用户，当然，更好的方法是将数据保存起来，间接的供用户使用，为什么要这样做呢？原因是，这样可以降低代码的耦合性，试想，如果我们的这一部分数据同时在两个地方被使用到的话，我们没有抽取出来，那我们就会面临这样的尴尬，无法复用，这就和我们设计代码是一样的，尽量不做重复的工作。

首先，我们需要设计出来保存数据的一个实体类，在我们这个程序里面，只有一个地方引用到了这些数据，所以首先我们需要设计一个保存数据的实体类，如下：

```
package  com.oracle.moudle;
/**
  *  实现数据的存储的类,界面在更新时所取到的数据就来源于此
  *
  *  @author  Administrator
  *
  */
public  class  BluetoothData  {
//  红外数据  有人为 0xef  无人为 0xcd
private  String  havePerson  =  "";
//  烟雾标志  有烟为 1  无烟为 2
private  String  haveSmoke  =  "";
//  下雨标志  下雨为 1  不下雨为 2
private  String  haveRain  =  "";
//  温度
private  int  temperature  =  0;
//  湿度
private  int  humidity  =  0;

public  String  getHavePerson()  {
return  havePerson;
}
```

```
public void setHavePerson(String havePerson) {
this.havePerson = havePerson;
}

public String getHaveSmoke() {
return haveSmoke;
}

public void setHaveSmoke(String haveSmoke) {
this.haveSmoke = haveSmoke;
}

public String getHaveRain() {
return haveRain;
}

public void setHaveRain(String haveRain) {
this.haveRain = haveRain;
}

public int getTemperature() {
return temperature;
}

public void setTemperature(int temperature) {
this.temperature = temperature;
}

public int getHumidity() {
return humidity;
}

public void setHumidity(int humidity) {
this.humidity = humidity;
}

@Override
```

```
public String toString() {
return "BluetoothData [havePerson=" + havePerson + ", haveSmok
e="
+ haveSmoke + ", haveRain=" + haveRain + ", temperature="
+ temperature + ", humidity=" + humidity + "]";
}
}
```

上面的实体类的成员变量类型全部都是静态的，静态的好处是全局只有一个，这保证了数据的唯一性。但是一定需要考虑数据在并发处理时的死锁问题。如图 4.5 所示：

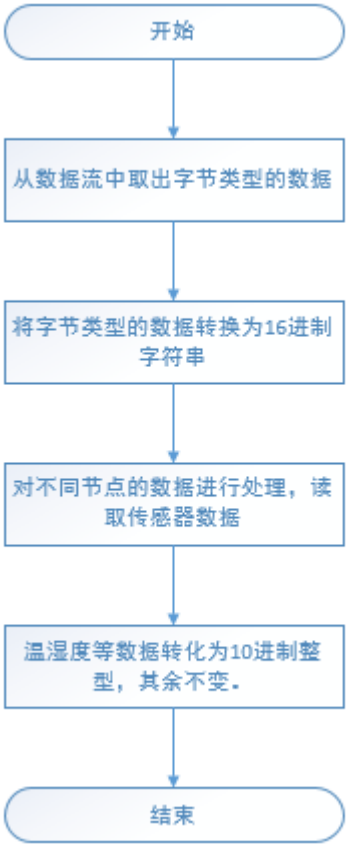


图 4.5 数据存储及展示

4.4.1 接收数据函数

上面我们已经看到了保存数据的实体类，那我们的数据到底是如何接收过来的呢？当然是有专门的数据接受函数，本部分是 zigbee 下位机与上位机连接的最重要的部分之一,上面我们谈到，我们的数据是建立在数据链路之上的，当我们需要取出数据的时候，我们就需要直接从输入流中进行读取。代码展示如下：

```
for (int i = 0; i < 14; i++) {
mmInStream.read(buffer);
res.append(CommonUtil.bytesToHexString(buffer));
}
// System.out.println("数据为:" + res.toString());
String macAdress = res.substring(0, 16);
String result = res.substring(16);
```

4.4.2 字节转 16 进制

我们都知道，计算机的底层数据的传输都是按照字节的形式进行传输的，也就是说，我们从上面也可以看到，我们接收的数据都是字节的形式，因此，我们需要对其做一个转换，而 java 没有为我们提供现成的 api，也就是说，我们需要自己编写一个这样的方法，那么方法如下：

```
public static String bytesToHexString(byte[] src) {
StringBuilder stringBuilder = new StringBuilder("");
if (src == null || src.length <= 0) {
return null;
}
for (int i = 0; i < src.length; i++) {
int v = src[i] & 0xFF;
String hv = Integer.toHexString(v);
if (hv.length() < 2) {
stringBuilder.append(0);
}
stringBuilder.append(hv);
}
return stringBuilder.toString();
}
```

我们可以看到，上面的方法是一个静态的方法，静态的方法我们可以直接用类名调用。方便我们的调用。

4.4.3 16 进制转 10 进制

在上面的步骤中，我们将字节转换为了 16 进制字符串，这样的话方便我们对数据进行分割处理，但是温度和湿度我们不能直接以 16 进制的形式展示给用户，因此我们需要将 16 进制的字符串其切换为 10 进制的字符串。

方法如下：

```
/**
```

```
 * 将 16 进制转化为 10 进制的数
 */
public static int hexToTen(String hexString) {
int result = 0;
String r1 = hexString.substring(1, 2);
String r2 = hexString.substring(0, 1);
r1 = formatString(r1);
r2 = formatString(r2);
result = Integer.valueOf(r2) * 16 + Integer.valueOf(r1) * 1;
return result;
}

private static String formatString(String r) {
if ("a".equals(r)) {
r = "10";
}
if ("b".equals(r)) {
r = "11";
}
if ("c".equals(r)) {
r = "12";
}
if ("d".equals(r)) {
r = "13";
}
if ("e".equals(r)) {
r = "14";
}
if ("f".equals(r)) {
r = "15";
}
return r;
}
```

上面的方法完成了我们最终的步骤，我们就可以将处理后的数据展示到前台的 app 界面上去。

4.5 下位机蓝牙发送设计

下位机的蓝牙发送设计也是我们需要关注的一个点，前面我们已经提到过，蓝牙的发送其实就是建立在串口之上的，如图 4.6 所示：

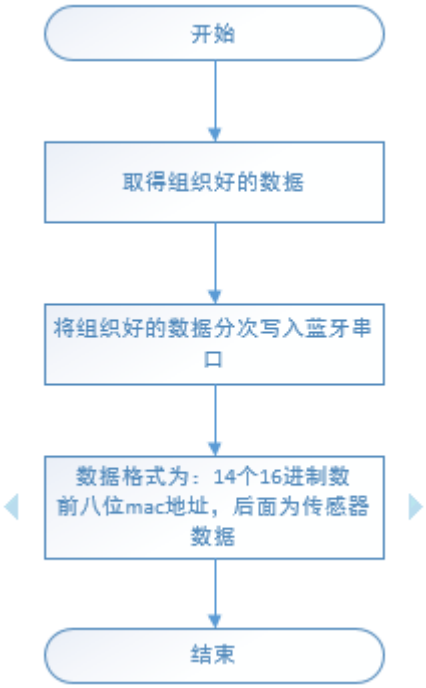


图 4.6 下位机蓝牙数据发送

如下面代码所示：

```
/*********************************************************************
**********
 * @fn      HalUARTWriteDMA
 *
 * @brief   Write a buffer to the UART.
 *
 * @param   buf - pointer to the buffer that will be written, not freed
 *          len  - length of
 *
 * @return  length of the buffer that was sent

*********************************************************************
********/
static uint16 HalUARTWriteDMA(uint8 *buf, uint16 len)
{
  uint16 cnt;
  halIntState_t his;
  uint8 txIdx, txSel;
```

```
// Enforce all or none.
if ((len + dmaCfg.txIdx[dmaCfg.txSel]) > HAL_UART_DMA_TX_MAX)
{
  return 0;
}

HAL_ENTER_CRITICAL_SECTION(his);
txSel = dmaCfg.txSel;
txIdx = dmaCfg.txIdx[txSel];
HAL_EXIT_CRITICAL_SECTION(his);

for (cnt = 0; cnt < len; cnt++)
{
  dmaCfg.txBuf[txSel][txIdx++] = buf[cnt];
}

HAL_ENTER_CRITICAL_SECTION(his);
if (txSel != dmaCfg.txSel)
{
  HAL_EXIT_CRITICAL_SECTION(his);
  txSel = dmaCfg.txSel;
  txIdx = dmaCfg.txIdx[txSel];

  for (cnt = 0; cnt < len; cnt++)
  {
    dmaCfg.txBuf[txSel][txIdx++] = buf[cnt];
  }
  HAL_ENTER_CRITICAL_SECTION(his);
}

dmaCfg.txIdx[txSel] = txIdx;

if (dmaCfg.txIdx[(txSel ^ 1)] == 0)
{
  // TX DMA is expected to be fired
  dmaCfg.txDMAPending = TRUE;
}
HAL_EXIT_CRITICAL_SECTION(his);
```

```
  return cnt;
}
```

4.6 蓝牙接收消息函数设计

蓝牙的接受也是十分重要的一个点，因为蓝牙的接收函数是否正常工作直接就决定了我们的下位机能否正常响应上位机的命令。本人可以再前面约定的通信协议中看到，我们必须在下位机的代码中匹配上位机发送过来的相应数据。如图 4.7 所示：

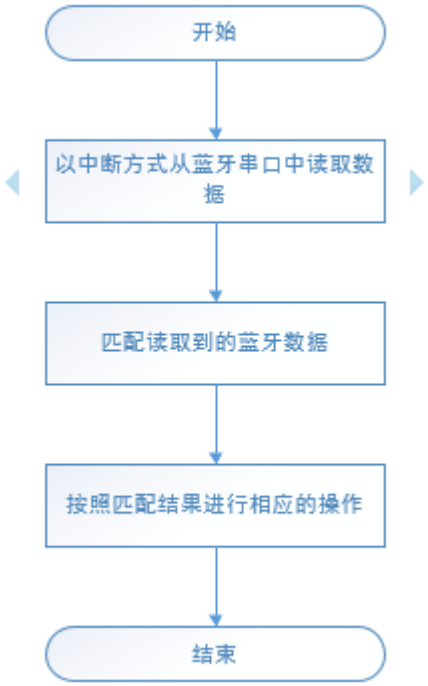


图 4.7 下位机数据接受处理

代码如下：

```
/*********************************************************************
*********
 * @fn      HalUARTReadDMA
 *
 * @brief   Read a buffer from the UART
 *
 * @param   buf  - valid data buffer at least 'len' bytes in size
 *          len  - max length number of bytes to copy to 'buf'
 *
 * @return  length of buffer that was read

*********************************************************************
********/
static uint16 HalUARTReadDMA(uint8 *buf, uint16 len)
```

```
{
  uint16 cnt;

  for (cnt = 0; cnt < len; cnt++)
  {
    if (!HAL_UART_DMA_NEW_RX_BYTE(dmaCfg.rxHead))
    {
      break;
    }
    *buf++ = HAL_UART_DMA_GET_RX_BYTE(dmaCfg.rxHead);
    HAL_UART_DMA_CLR_RX_BYTE(dmaCfg.rxHead);
    if (++(dmaCfg.rxHead) >= HAL_UART_DMA_RX_MAX)
    {
      dmaCfg.rxHead = 0;
    }
  }
  PxOUT &= ~HAL_UART_Px_RTS;  // Re-enable the flow on any read.

  return cnt;
}
```

4.7 本章小结

本章主要进行了软件部分的设计，包括的部分主要有下位机的蓝牙部分设计和上位机的界面以及蓝牙接收及发送代码的设计，在设计的过程中遇到的技术点主要有以下几个部分：

1：下位机蓝牙函数的编写。

2：上位机的蓝牙连接需要使用 socket，但是对 socket 编程并不是十分熟悉，也搞不清楚上位机到底应该作为服务端还是客户端，这个问题困扰了我很久。

3：展示的时候需要展示动态的图表，但是动态的图表 linechart 并没有实现，于是就只能上网查找资料，最后通过不断的 ui 刷新实现了动态的图表展示。

4：数据的是一大难点，因为底层的数据传输和高级语言的兼容性并不是很好，于是就遇到了很多的问题，比如说数制转换啊，之类的东西。

5：ui 界面的无缝切换，一开始想着分界面实现，后来利用 viewpager 实现了所有的功能，当然，实在 github 上找的一些代码的基础上修改的。

6：蓝牙和 ui 界面进行组装的时候出现了问题，一开始的话两者之间的数据不同步，但是后来利用线程进行了同步。

5 系统调试

5.1 开发环境-Eclipse 简介

Eclipse 是著名的跨平台开源集成开发环境（IDE）。最初主要用来 Java 语言开发，目前亦有人通过插件使其作为 C++、Python、PHP 等其他语言的开发工具。

Eclipse 的本身只是一个框架平台，但是众多插件的支持，使得 Eclipse 拥有较佳的灵活性，所以许多软件开发商以 Eclipse 为框架开发自己的 IDE。

Eclipse 最初是由 IBM 公司开发的替代商业软件 Visual Age for Java 的下一代 IDE 开发环境，2001 年 11 月贡献给开源社区，现在它由非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理。2003 年，Eclipse 3.0 选择 OSGi 服务平台规范为运行时架构。2007 年 6 月，稳定版 3.3 发布；2008 年 6 月发布代号为 Ganymede 的 3.4 版；2009 年 6 月发布代号为 Galileo 的 3.5 版；2010 年 6 月发布代号为 Helios 的 3.6 版；2011 年 6 月发布代号为 Indigo 的 3.7 版；2012 年 6 月发布代号为 Juno 的 4.2 版；2013 年 6 月发布代号为 Kepler 的 4.3 版；2014 年 6 月发布代号为 Luna 的 4.4 版；2015 年 6 月发布代号为 Mars 的 4.5 版。

Eclipse 的基础是富客户机平台（即 RCP）。RCP 包括下列组件：

核心平台（启动 Eclipse，运行插件）

OSGi（标准集束框架）

SWT（可移植构件工具包）

JFace（文件缓冲，文本处理，文本编辑器）

Eclipse 工作台（即 Workbench，包含视图（views）、编辑器（editors）、视角（perspectives）、和向导（wizards））

Eclipse 采用的技术是 IBM 公司开发的（SWT），这是一种基于 Java 的窗口组件，类似 Java 本身提供的 AWT 和 Swing 窗口组件；不过 IBM 声称 SWT 比其他 Java 窗口组件更有效率。Eclipse 的用户界面还使用了 GUI 中间层 JFace，从而简化了基于 SWT 的应用程序的构建。

但是，仅仅有 eclipse 是不行的，android 平台的所属权是 Google 公司，其为 android 定制开发了在 eclipse 使用的插件 ADT，安装 ADT 插件后安装 androidSdK 后即可进行 android 程序的开发。

5.2 Eclipse 编程及调试

5.2.1 软件编程

a 创建工程文件：单击Eclipse菜单中File-New-Android Project菜单项，输

入工程名字后创建一个新的工程。新建工程要使用独立的文件夹。工程文件建立后，

为工程配置一些基本属性，如编译后的代码放到哪个位置等。

b 创建新的包名并添加在工程中，创建新的源文件，值得注意的是，在java中，不同的功能的代码需要放到不同的包中去，需要遵守一定的代码规范，如表5.1所示：

包名	
Service	服务类的代码
Dao	访问数据库相关的代码
Test	测试相关的代码
Entity	实体类相关的代码

表 5.1 包名对应功能代码对照表

在建立完成对应的代码源文件后，本人需要正式开始编写程序。

c 程序编写：在程序设计界面输入JAVA代码。

当然，在程序编写的过程中一定要注意编写程序的规范。

5.2.2 调试

在我们已经完成的项目上右键点击 debug as android application,这个时候只要连接上我们的手机，我们就可以看到后台打印出的控制信息了。

当然，在调试的过程中也遇到了一些问题，那就是我的手机的安全限制不允许来自电脑的连接，于是就在网上查找了相关的解决方法。

网上给出的解决方法是将手机的某项参数设置为可调试的就可以了。

5.3 调试工具简介

Androidlogcat控制台其实就是一般的程序控制台一样用来获取调试的相关信息的一个工具，特别的是这个工具只为android平台服务罢了。

当然，这个控制台的功能不仅限于调试，它还有着很多强大的功能，下面一一做出列举。

1：可以实现对与来自不同界面的信息的过滤。

2：实现来自不同标志的调试信息的过滤。

3：支持调试信息的导出，这一点十分有用，因为我们在调试的过程中可能有许多的重要的特性都要保存下来，方便我们日后的分析。

4:会对我们的调试信息作出十分准确的描述,比如说属于哪个进程之类的。

5.4 调试结果及分析

图 5.1 调试结果图

看到上面的结果，我们可以看到，我们已经成功的接入了蓝牙发送给我们的数据，我们也成功的对数据做好了我们的处理，得到了我们想要的格式。

这样的话就可以很方便的对我们的数据进行处理，想把它显示到哪里就可以显示到那里去，我们可以看到，数据的发送时按照一定的时间间隔发送过来的，但是实际上，我们上位机的接收部分代码在没有数据的时候是处于阻塞状态的，所以并不会非常消耗系统的资源。

但是我们也可以发现，我们的代码还存在这许多需要优化的地方，而这些点，则是我们需要注意的地方。

6 结论与展望

6.1 结论

本文根据设计要求，考虑产品的性价比，进行系统的整体方案设计，实现一款能够应用于户外营地安全的产品。

本文研究的内容有：zigbee下位机与HC-06蓝牙模块的衔接、android开发的大致流程、android的开发标准、android的界面设计、android app的调试、linchart框架的使用、linechart动态图表的设计、android viewpager的使用等。

软件设计中，充分发挥java语言的优势，结合axure设计的UI界面，使整个软件系统模块功能完美响应软件系统调用，在展示动态图标温度数据的时候，做到了ui界面不卡顿，这一切都归功于多线程。在多界面展示中，采用了典型的利用viewpager进行分屏的方法。

本文从本设计的方案引入，分别介绍了zigbee模块、HC-06蓝牙模块、android上位机开发所需要的环境、用到的开源框架、以及大致的软件流程，证实了本项目开发的可行性。

本设计的所完成的产品，适用于各种复杂的野外宿营场合，其稳定性是十分有益的，可以说，这是一款充分考虑了实际应用的产品，是一款有前景的产品。

6.2 展望

本次设计所开发出来的产品,具有美观的界面、完整的功能、良好的设计交互体验，目前市场上并没有相同类型的产品，所以说这对于抢夺市场先机是十分重要的。

本次设计的上位机部分开发流程规范，代码结构性强，便于后期的维护和扩展，后期还可以不断的进行优化。

可以说，本次设计完成了设计的指标，同时也做出了不错的成绩，在后期，对于产品的进一步打磨将带给我们更多更好的体验！

致　谢

在此毕业设计课题完成之际，特此向为此毕业设计做出指导的王鹏老师表示深深地敬意和谢意。指导老师王鹏在此课题的研究方向、收集资料、设计过程中进行了悉心的指导，尽其所能的帮助。本设计从选题到完成，每一步都是在王鹏老师的悉心指导下完成的，倾注了王老师大量的心血。王老师学识渊博，治学严谨，在工作中兢兢业业，辅导学生时循循善诱、极其认真耐心，让本人深刻地体会到真正的为人师表的风范。。

在本课题的研究过程中，也得益于学校的各方面资源，图书馆可以查找到许多相关的有用的书籍资料、图书馆网站上的数据库也给我提供了不少参考，这些都给我们创造一个良好的学习、设计环境。在此向为学生服务的各位老师和管理员表示由衷的谢意。

另外，要感谢在大学期间所有传授本人知识的老师，是你们的悉心教导使本人有了良好的专业课知识，这也是论文得以完成的基础。同时，论文的顺利完成，离不开其它各位老师、同学的关心和帮助。在整个的论文写作中，各位老师、同学积极的帮助我查找资料并提供有利于论文写作的建议和意见，在他们的帮助下，论文得以不断的完善，最终成为一篇合格的毕业论文。

最后，再次向所有关心和帮助本人的老师、同学表达真诚的谢意。

参考文献

[1] 熊茂华.杨震伦.ARM9嵌入式系统设计与开发应用【M】.清华大学出版社2008

[2] 罗亚非等 32位嵌入式微处理器原理及应用【M】北京航空航天大学出版社2010

[3] 李明亮 蒙洋 康辉英 例说ZigBee【M】北京航空航天大学出版社 2013 282-294

[4] 青岛东合信息技术有限公司 Cortex-M3开发技术及实践 青岛东合信息技术有限公司【M】西安电子科技大学出版社 2013 7-12

[5] 吴培亚 王钢 基于ZigBee的智能家居远程监控系统的设计与实现

[6] 向忠宏 智能家居【M】人民邮电出版社 2002-06

[7] 侯海涛;国内外智能家居发展现状[J];建材发展导向;2004年05期

[8] 郑毅;刘润华;;基于CC2430的低功耗智能家居数据采集系统[J];电子设计工程;2011年22期

[9] 李皓;;基于ZigBee的无线网络技术及应用[J];信息技术;2008年01期

[10] 张周;周剑扬;闫沫;;ZigBee在智能家居系统中的应用研究[J];工业控制计算机;2006年12期

[11] 刘礼建;张广明;;基于ZigBee无线技术的智能家居管理系统设计[J];计算机技术与发展;2011年12期

[12] [10]张维勇,冯琳,魏振春.ZigBee实现家庭组网技术的研究[J].合肥工业大学学报,2005,7,28-7.

[13] Gutierrez J A,CallawayE,Barrett R.Low-rate wireless personal area networds;Enab- ling wireless sensorswith IEEE 802.15.4[M].New York:Institute of Electrical & Eledr-onicsEnginee,2003,62-74.

[14] 周游,方滨,王普.基于ZigBee技术的智能家居无线网络系统[J].电子技术应用,2005,9

[15] 尚丽丽;基于ZigBee的智能家居系统设计[D];大连理工大学;2010年

[16] EGAND． The emergence of Zig Bee in building automa-ion and industrial control[M].Computing ＆ ControlEngineering Journal，2005，16(2) : 14-19．

[17] RAPPAPORT T S.Wireless communications principlesand practice［M].2nd ed．Beijing: Publishing Houseof Electronics Industry，2006．

毕业设计（论文）知识产权声明

本人完全了解西安工业大学有关保护知识产权的规定，即：本科学生在校攻读学士学位期间毕业设计（论文）工作的知识产权属于西安工业大学。本人保证毕业离校后，使用毕业设计（论文）工作成果或用毕业设计（论文）工作成果发表论文时署名单位仍然为西安工业大学。学校有权保留送交的毕业设计（论文）的原文或复印件，允许毕业设计（论文）被查阅和借阅；学校可以公布毕业设计（论文）的全部或部分内容，可以采用影印、缩印或其他复制手段保存毕业设计（论文）。

（保密的毕业设计（论文）在解密后应遵守此规定）

毕业设计（论文）作者签名：

指导教师签名：

日期：

毕业设计（论文）独创性声明

秉承学校严谨的学风与优良的科学道德，本人声明所呈交的毕业设计（论文）是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，毕业设计（论文）中不包含其他人已经发表或撰写过的成果，不包含他人已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了致谢。

毕业设计（论文）与资料若有不实之处，本人承担一切相关责任。

毕业设计（论文）作者签名：

指导教师签名：

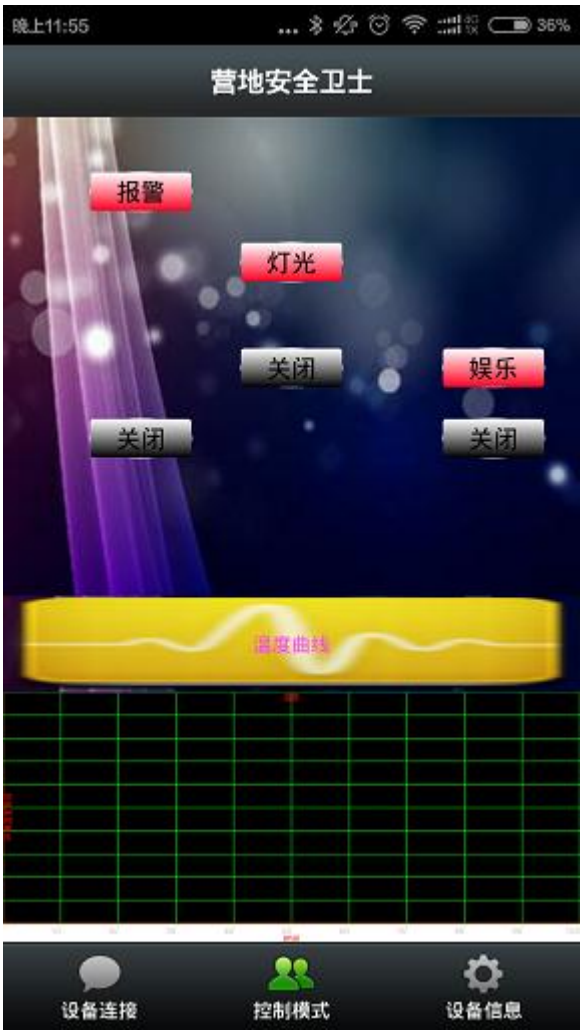
日期：

附录Ⅰ app 界面展示

主引导界面展示

登录界面展示

蓝牙连接界面展示



控制界面展示

蓝牙设备展示界面

数据展示界面

附录Ⅱ 软件设计清单

```
//主界面代码展示：
package com.example.bluetooth;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;
import org.achartengine.GraphicalView;

import android.R.integer;
import android.R.string;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v4.view.ViewPager.OnPageChangeListener;
import android.support.v7.app.ActionBarActivity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup.LayoutParams;
import android.view.ViewGroup;
```

```
import android.view.Window;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import com.example.service.ChartService;
import com.example.service.CommonUtil;
import com.oracle.moudle.BluetoothData;

/**
 *
 * @author Henry
 * @version 1.0
 * @since 2015/12/18
 *
 */
public class MainActivity extends ActionBarActivity implements OnClickListener {
    // 蓝牙连接需要用到的常量
    private static final int REQUEST_ENABLE_BT = 0;
    private static final int REQUEST_CONNECT_DEVICE = 1;
    private static final int MSG_NEW_DATA = 3;
    // 蓝牙数据&&mac 地址
    private BluetoothData bluData = new BluetoothData();
    private String macNode1 = "2b282b06004b1200";
    private String macNode2 = "92d76002004b1200";
    // 蓝牙设备
    private BluetoothAdapter mBluetoothAdapter;
    public ConnectedThread mConnectedThread;
    // 蓝牙所取得的结果
    private int result;
    // 蓝牙连接需要用到的 UUID
    private static final String SPP_UUID = "00001101-0000-1000-8000-00805F9B34FB";
    // 打印输出信息的标示
    public static String TAG = "MAINACTIVITY";
    String s = new String();
    // tab 界面上需要用到的控件
    private ViewPager mViewPager;
    private PagerAdapter mAdapter;
    private List<View> mViews = new ArrayList<View>();
    // TAB
    private LinearLayout mTabWeixin;
    private LinearLayout mTabFrd;
    private LinearLayout mTabSetting;
```

```
private ImageButton mWeixinImg;
private ImageButton mFrdImg;
private ImageButton mSettingImg;
// view
private View tab01;
private View tab02;
private View tab04;
// 蓝牙设备的基本信息
private String name;
private String address;
private String connectResult = "连接失败";
// 界面一用到的控件,按照从上到下的顺序进行
private TextView txtnNme;
private TextView txtAddress;
private TextView txtConnectMotion;
private Button btnScan;

// 界面二用到的控件
private Button btnOpenClock;
private Button btnCloseClock;
private Button btnOpenLight;
private Button btnCloseLight;
private Button btnOpenExerise;
private Button btnCloseExerise;
// 折线图
private LinearLayout right;
private GraphicalView rightview;
private GraphicalView leftview;
private ChartService rightservice;
private ChartService leftservice;
private Timer timer;

// 执行响应的命令的按钮实例
// 界面三用到的控件
private TextView txtTempature;
private TextView txtHavePerson;
private TextView txtHaveRain;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 设置主界面样式
    requestWindowFeature(Window.FEATURE_NO_TITLE);
```

```
        setContentView(R.layout.activity_main);
        // 初始化主界面
        initView();
        // 初始化事件处理
        initEvents();
        // 初始化折线图
        initChart();
        // 获取本地的蓝牙适配器
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        // 判断蓝牙设备是否已经启用
        if (mBluetoothAdapter == null) {
            Toast.makeText(MainActivity.this, "蓝牙设备未启用或本地无蓝牙设备!",
                    Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
}

/**
 * 初始化折线图
 */
private void initChart() {
        // 得到 view
        right = (LinearLayout) tab02.findViewById(R.id.id_linechart);
        // 开始设置右边的图表
        rightservice = new ChartService(this);
        rightservice.setXYMultipleSeriesDataset("右温度曲线");
        rightservice.setXYMultipleSeriesRenderer(100, 40, "温度", "时间", "环境温度曲线",
                Color.RED, Color.RED, Color.GREEN, Color.GREEN);
        rightview = rightservice.getGraphicalView();
        // 设置左边的图表
        // 将右边的图表添加到布局容器中去
        rightview.setFocusable(false);
        rightview.setBackgroundColor(Color.BLACK);
        rightview.setSelected(false);
        rightview.setClickable(false);
        rightview.setEnabled(false);
        right.addView(rightview, new LayoutParams(LayoutParams.MATCH_PARENT,
                LayoutParams.MATCH_PARENT));
        // 定时器
        timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
```

```
                handler.sendMessage(handler.obtainMessage());
            }
        }, 10, 100);
        handler = new Handler() {

            @SuppressLint("HandlerLeak")
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                //  更新温湿度等
                // txtTempature.setText("");
                // txtTempature.setText("当前温度为：" + bluData.getTemperature()
                // + "--当前湿度为：" + bluData.getHumidity());
                // Toast.makeText(MainActivity.this, "当前温度为：" +
                // bluData.getTemperature()
                // + "--当前湿度为：" + bluData.getHumidity(),
                // Toast.LENGTH_SHORT).show();
                // send data
                int addY = bluData.getTemperature();
                rightservice.updateChart(addY);
            }
        };
    }

    /**
     * 更新界面的线程
     */
    private class updataUserInterface extends Thread {

        @Override
        public void run() {
            super.run();
            while (true) {
                uiHandler.sendMessage(uiHandler.obtainMessage());
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }

    }
```

```
    /**
     * 更新绘图界面,更新界面
     */
    private Handler handler;

    /****************************************                                    initView()
*************************************************************/
    private void initView() {
        mViewPager = (ViewPager) findViewById(R.id.id_viewpager);
        // tabs
        mTabWeixin = (LinearLayout) findViewById(R.id.id_tab_weixin);
        mTabFrd = (LinearLayout) findViewById(R.id.id_tab_frd);
        mTabSetting = (LinearLayout) findViewById(R.id.id_tab_settings);
        // ImageButton
        mWeixinImg = (ImageButton) findViewById(R.id.id_tab_weixin_img);
        mFrdImg = (ImageButton) findViewById(R.id.id_tab_frd_img);
        mSettingImg = (ImageButton) findViewById(R.id.id_tab_settings_img);

        LayoutInflater mInflater = LayoutInflater.from(this);
        tab01 = mInflater.inflate(R.layout.tab01, null);
        /**** **********得到界面一的 4 个控件 ****************************************/
        txtnNme = (TextView) tab01.findViewById(R.id.id_device_name);
        txtAddress = (TextView) tab01.findViewById(R.id.id_device_address);
        txtConnectMotion = (TextView) tab01
                .findViewById(R.id.id_device_connectmotion);
        tab02 = mInflater.inflate(R.layout.tab02, null);
        btnScan = (Button) tab01.findViewById(R.id.id_device_scan);
        /**********      得    到    界    面    6    个    的    控    件
*********************************************/
        btnOpenClock = (Button) tab02.findViewById(R.id.id_openClock);// 开启报警
        btnCloseClock = (Button) tab02.findViewById(R.id.id_closeClock);// 关闭报警
        btnOpenLight = (Button) tab02.findViewById(R.id.id_openlight); // 开启灯光
        btnCloseLight = (Button) tab02.findViewById(R.id.id_closeLight); // 开启灯光
        btnOpenExerise = (Button) tab02.findViewById(R.id.id_openExerise);// 开启娱乐模式
        btnCloseExerise = (Button) tab02.findViewById(R.id.id_closeExerise);// 关闭娱乐
        tab04 = mInflater.inflate(R.layout.tab04, null);
        // 得到界面 3 的三个控件
        txtTempature = (TextView) tab04.findViewById(R.id.id_tempature);
        txtHavePerson = (TextView) tab04.findViewById(R.id.id_haveperson);
        txtHaveRain = (TextView) tab04.findViewById(R.id.id_haverain);
        // 设置背景图片
        tab01.setBackgroundResource(R.drawable.ic_back_1);
        tab02.setBackgroundResource(R.drawable.ic_back_2);
        tab04.setBackgroundResource(R.drawable.ic_back_3);
```

```
        mViews.add(tab01);
        mViews.add(tab02);
        mViews.add(tab04);
        mAdapter = new PagerAdapter() {

            @Override
            public void destroyItem(ViewGroup container, int position,
                    Object object) {
                container.removeView(mViews.get(position));
            }

            @Override
            public Object instantiateItem(ViewGroup container, int position) {
                View view = mViews.get(position);
                container.addView(view);
                return view;
            }

            @Override
            public boolean isViewFromObject(View arg0, Object arg1) {
                return arg0 == arg1;
            }

            @Override
            public int getCount() {
                return mViews.size();
            }
        };
        // 设置适配器
        mViewPager.setAdapter(mAdapter);
    }

    /**********************************************                initView();
************************************************************/
    /**********************************************               initEvents();
************************************************************/
    private void initEvents() {
        // 设置主界面三个按钮的监听
        mTabWeixin.setOnClickListener(this);
        mTabFrd.setOnClickListener(this);
        mTabSetting.setOnClickListener(this);
        mViewPager.setOnPageChangeListener(new OnPageChangeListener() {

            @Override
```

```
            public void onPageSelected(int arg0) {
                int currentItem = mViewPager.getCurrentItem();
                resetImg();
                switch (currentItem) {
                case 0:
                    mWeixinImg.setImageResource(R.drawable.tab_weixin_pressed);
                    break;
                case 1:
                    mFrdImg.setImageResource(R.drawable.tab_find_frd_pressed);
                    break;
                case 2:
                    mSettingImg
                            .setImageResource(R.drawable.tab_settings_pressed);
                    break;
                }
            }

            @Override
            public void onPageScrolled(int arg0, float arg1, int arg2) {

            }

            @Override
            public void onPageScrollStateChanged(int arg0) {

            }
        });
        // 设置界面一控件的监听事件
        btnScan.setOnClickListener(this);
        // 设置界面二控件的监听事件
        btnOpenClock.setOnClickListener(this);
        btnCloseClock.setOnClickListener(this);
        btnOpenLight.setOnClickListener(this);
        btnCloseLight.setOnClickListener(this);
        btnOpenExerise.setOnClickListener(this);
        btnCloseExerise.setOnClickListener(this);
    }

    /****************************************                                    initEvents();
*********************************************************/
    @Override
    protected void onStart() {
        super.onStart();
        if (!mBluetoothAdapter.isEnabled()) {
```

```
            // 启动蓝牙的意图
            Intent bluIntent = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
            // 如果进入启动界面前没有启动蓝牙,用户一旦启动蓝牙,会直接进入到蓝牙的搜索界
面
            startActivityForResult(bluIntent, REQUEST_ENABLE_BT);
        }
    }

    /**
     * 匹配蓝牙搜索界面返回的信息,进行相应的操作
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        // 根据返回的信息处理请求
        switch (requestCode) {
        // 匹配用户进入 app 打开蓝牙的行为:
        case REQUEST_ENABLE_BT:
            if (resultCode == Activity.RESULT_OK) {
                // 启动蓝牙搜索界面
                Intent serverIntent = new Intent(MainActivity.this,
                        DeviceListActivity.class);
                // 方便后面的参数进行匹配
                startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE);
            } else {
                Log.d(TAG, "bluetooth not enabled!");
                finish();
                return;
            }
            break;
        case REQUEST_CONNECT_DEVICE:
            if (resultCode != Activity.RESULT_OK) {
                return;
            } else {
                String info = data.getExtras().getString(
                        DeviceListActivity.EXTRA_DEVICE_ADDRESS);
                // 蓝牙设备名称
                name = info.substring(0, info.length() - 17);
                System.out.println(name);
                // 根据 MAC 地址获得蓝牙设备
                address = info.substring(info.length() - 17);
                BluetoothDevice device = mBluetoothAdapter
                        .getRemoteDevice(address);
                connect(device);
```

```
            }
            break;
        default:
            // 无匹配信息
            break;
        }
    }

    /**
     * 根据 MAC 地址连接设备
     *
     * @param device
     *                  根据蓝牙 MAC 地址得到的硬件设备
     */
    private void connect(BluetoothDevice device) {
        Log.d(TAG, "connect to " + device);
        // 耗时操作另外开启一个线程
        new ConnectThread(device).start();
    }

    /**
     * 内部类,负责连接的线程
     */
    private class ConnectThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final BluetoothDevice mmDevice;

        public ConnectThread(BluetoothDevice device) {
            mmDevice = device;
            BluetoothSocket tmp = null;
            // 得到蓝牙 socket 连接
            try {
                tmp = device.createRfcommSocketToServiceRecord(UUID
                        .fromString(SPP_UUID));
            } catch (IOException e) {
                Log.e(TAG, "create() failed", e);
            }
            mmSocket = tmp;
        }

        @Override
        public void run() {
            super.run();
            // 停止搜索
```

```
                mBluetoothAdapter.cancelDiscovery();
                try {
                    mmSocket.connect();
                    connectResult = "连接成功";
                    /******************  使 用 Handler 更 新 页 面 状 态
**************************************/
                    myHandler.sendMessage(myHandler.obtainMessage());
                } catch (IOException e) {
                    e.printStackTrace();
                    Log.d(TAG, "蓝牙设备连接失败!");
                }
                // 开始监听来自蓝牙设备的数据
                mConnectedThread = new ConnectedThread(mmSocket);
                mConnectedThread.start();
            }
        }

        /**
         * This thread runs during a connection with a remote device. It handles all
         * incoming and outgoing transmissions.
         */
        private class ConnectedThread extends Thread {
            private final BluetoothSocket mmSocket;
            private final InputStream mmInStream;
            private final OutputStream mmOutStream;

            public ConnectedThread(BluetoothSocket socket) {
                Log.d(TAG, "create ConnectedThread");
                mmSocket = socket;
                InputStream tmpIn = null;
                OutputStream tmpOut = null;

                // 得到输入流和输出流
                try {
                    tmpIn = socket.getInputStream();
                    tmpOut = socket.getOutputStream();
                } catch (IOException e) {
                    Log.e(TAG, "temp sockets not created", e);
                }

                mmInStream = tmpIn;
                mmOutStream = tmpOut;
            }
```

```
public void run() {
    Log.i(TAG, "BEGIN mConnectedThread");
    @SuppressWarnings("unused")
    int bytesLength = 0;
    StringBuffer res = new StringBuffer();
    byte[] buffer = new byte[1];
    while (true) {
        try {
            for (int i = 0; i < 14; i++) {
                mmInStream.read(buffer);
                res.append(CommonUtil.bytesToHexString(buffer));
            }
            // System.out.println("数据为:" + res.toString());
            String macAdress = res.substring(0, 16);
            String result = res.substring(16);
            // 判断是哪个节点的数据，对应节点进行对应的解析
            if (macNode1.equalsIgnoreCase(macAdress)) {
                bluData.setTemperature(CommonUtil.hexToTen(result
                        .substring(8, 10)));
                bluData.setHumidity(CommonUtil.hexToTen(result
                        .substring(4, 6)));
                // 有无人是两个节点都要更新的
                bluData.setHavePerson(result.substring(2, 4) + "-"
                        + macAdress);
            } else if (macNode2.equalsIgnoreCase(macAdress)) {
                bluData.setHaveRain(result.substring(10, 12));
                bluData.setHaveSmoke(result.substring(6, 8));
                // 有无人是两个节点都要有更新的
                bluData.setHavePerson(result.substring(2, 4) + "-"
                        + macAdress);
            }
            // System.out.println(result+"-----"+macAdress);
            System.out.println(bluData.toString());
            Thread.sleep(100);
            // 性能问题
            res = new StringBuffer();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
    /**
     * 发送数据
     *
     * @throws InterruptedException
     */
    public void writeChar(String message) throws InterruptedException {
        try {
            mmOutStream.write(message.getBytes());
            Thread.sleep(100);
        } catch (IOException e) {
            Log.e(TAG, "Exception during write");
        }
    }

    /**
     * 关闭 socket 连接
     */
    public void cancel() {
        try {
            if (mmSocket != null) {
                mmSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/**
 * 释放占用的资源
 */
@Override
protected void onDestroy() {
    super.onDestroy();
    if (timer != null) {
        timer.cancel();
    }
}

@Override
protected void onPause() {
    super.onPause();
}
```

```
//通用工具类代码展示
package com.example.service;

import java.util.HashMap;
import java.util.Map;

/**
 * 公共的工具类
 *
 * @author fulei.yang
 *
 */
public class CommonUtil {
	private static String noPer = "cd";
	private static String havePer = "ef";
	private static String noSmo = "ee";
	private static String haveSmo = "dd";
	private static String noRain = "aa";
	private static String haveRa = "cc";
	private static String resultHavePerson;
	private static String resultHaveRain;
	private static String resultHaveSmoke;
	private static String finalResult;
	private static String macAddress;

	/**
	 * 将字节转换为 16 进制
	 *
	 * @param src
	 * @return stringBuilder
	 */
	public static String bytesToHexString(byte[] src) {
		StringBuilder stringBuilder = new StringBuilder("");
		if (src == null || src.length <= 0) {
			return null;
		}
		for (int i = 0; i < src.length; i++) {
			int v = src[i] & 0xFF;
			String hv = Integer.toHexString(v);
			if (hv.length() < 2) {
				stringBuilder.append(0);
			}
			stringBuilder.append(hv);
```

```
        }
        return stringBuilder.toString();
    }

    public static Map<String, String> disposeData(String havePerson,
            String haveRain, String haveSmoke, int tem, int hum) {
        Map<String, String> map = new HashMap<String, String>();
        // 温湿度
        String resultTem = "当前温度为:" + tem + "    ----    " + "当前湿度为:" + hum;
        // 红外传感器
        resultHavePerson = "";
        if (havePerson.equals("")) {
            resultHavePerson = "传感器出现故障";
        } else {
            macAddress = havePerson.substring(3, 19);
            if (!havePerson.equals("")
                    && havePer.equalsIgnoreCase(havePerson.substring(0, 2))) {
                resultHavePerson = "mac 地址为" + macAddress + "的节点有人侵入！";
            } else if (!havePerson.equals("")
                    && noPer.equalsIgnoreCase(havePerson.substring(0, 2))) {
                resultHavePerson = "无物体侵入!";
            } else {
                resultHavePerson = "传感器出现故障！";
            }
        }

        // 烟雾和雨滴传感器
        resultHaveRain = "";
        resultHaveSmoke = "";
        if (haveRa.equalsIgnoreCase(haveRain)) {
            resultHaveRain = "是";
        } else if (noRain.equalsIgnoreCase(haveRain)) {
            resultHaveRain = "否";
        } else {
            resultHaveRain = "传感器出现异常";
        }
        if (haveSmo.equalsIgnoreCase(haveSmoke)) {
            resultHaveSmoke = "是";
        } else if (noSmo.equalsIgnoreCase(haveSmoke)) {
            resultHaveSmoke = "否";
        } else {
            resultHaveSmoke = "传感器出现异常";
        }
        finalResult = "";
```

```
            finalResult = "是否下雨:" + CommonUtil.resultHaveRain + " ---- " + "是否有烟雾:"
                    + resultHaveSmoke;
            map.put("tem", resultTem);
            map.put("person", resultHavePerson);
            map.put("smoke", finalResult);
            return map;
        }

        /**
         * 将 16 进制转化为 10 进制的数
         */
        public static int hexToTen(String hexString) {
            int result = 0;
            String r1 = hexString.substring(1, 2);
            String r2 = hexString.substring(0, 1);
            r1 = formatString(r1);
            r2 = formatString(r2);
            result = Integer.valueOf(r2) * 16 + Integer.valueOf(r1) * 1;
            return result;
        }

        private static String formatString(String r) {
            if ("a".equals(r)) {
                r = "10";
            }
            if ("b".equals(r)) {
                r = "11";
            }
            if ("c".equals(r)) {
                r = "12";
            }
            if ("d".equals(r)) {
                r = "13";
            }
            if ("e".equals(r)) {
                r = "14";
            }
            if ("f".equals(r)) {
                r = "15";
            }
            return r;
        }
    }
```