论文题目 Android 电源管理机制研究与软件设计

专业学位类别 工 程 硕 士

学 号 201192250549

作 者 姓 名 鲁云霞

指 导 教 师 杨元杰 副研究员

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

Android 电源管理机制研究与软件设计

（题名和副题名）

鲁云霞

（作者姓名）

指导教师 杨元杰 副研究员

电子科技大学 成 都

王荣芝 高 工

昆山市富士康 昆 山

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工 程 硕 士

工程领域名称 软 件 工 程

提交论文日期 2014.03.01 论文答辩日期 2014.05.10

学位授予单位和日期 电子科技大学 2014 年 06 月 25 日

答辩委员会主席 杨宜康

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

ANDROID POWER MANAGEMENT MECHANISM AND SOFTWARE DESIGN

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Master of Engineering**

Author: **Lu Yunxia**

Advisor: **Yang Yuanjie**

School : **School of Aeronautics and Astronautics**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。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谢意。

作者签名：_____ 日期： 年 月 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：_____ 导师签名：_____

日期： 年 月 日

摘要

随着智能手机用户规模的高速增长，特别是 2008 年 Google 推出的 android 手机操作系统后，智能手机已然成为民众日常工作、生活中不可或缺的必备物品，虽然智能机的功能基本满足民众的需求，但是由于人们对于智能手机长期使用，导致电能的大量消耗已经严重影响了日常工作，因此研究智能手机电能节约系统已经被提上了日程。

本系统主要研究实现电量基本状态、省电模式、耗电排行和系统设置四个功能模块。电量基本状态模块主要实现剩余电量监测、可用 2G/3G 通话时间计算、可用上网时间统计等功能；省电模块主要有极限模式、强力模式和普通模式三种可供选择的功能模式，不同的模式对应不同的算法；耗电排行模块通过对系统各个应用进程耗电的情况进行排名，方便用户根据需要进行设定，以便达到省电的目的。本次研究首先通过对 android 开发平台和 google api 的研究，实现了在 android 平台上开发应用系统，实现界面交互，数据存储等基础功能。通过对 android 底层运行系统以及通过对 linux 系统内核的研究，得到了 CPU 及各硬件运行状态、电源电量、应用程序线程等信息，这些信息是实现系统功能最基础的内容。为了方便用户能够及时找到想要查找的信息，并且能够方面的对功能模块进行设置，通过引入一些图形插件和模板，设计了友好的交互界面。

本课题通过分析文献法、实例研究法和实践研究法方法作为主要研究手段，使用 android 平台应用系统的技术手段，完成一套基于 Android 平台的智能手机节电软件。在实际使用中，当在 android 智能手机中安装上本软件后，用户可以根据需求情况对软件进行模式设置，根据需要定制当前手机应用程序运行数量，从而达到减少电能消耗过快的目标。

关键字：android，电源管理，应用程序，省电模式

ABSTRACT

With the rapid growth of smart phone users, especially after Google launched android mobile operating system in 2008, smart phones have become a necessity of people's daily work and life. Although the function of smart phones have met the basic needs of people, the phones affect people’s work due to the long-term use which causes a large power consumption. Therefore, the study of power-saving smart phone system has been put on the agenda.

The system mainly studies four function modules, including the basic power state, the power-saving mode, the ranking list of power consumption and the system settings. The main achievement of the basic power state is to monitor residual capacity, to calculate available 2G/3G talking time, to count available online time and other functions. The power-saving mode includes the utmost limit mode, the strong power mode and the normal mode, altering to different calculation. The ranking list of power consumption ranks power consumption of each application procedure. It is convenient for users to set according to their needs in order to save power. The study, according to the android platform and Google api, achieves application development system on the android platform and other basic functions as interactive interface and data storage. It also gets information about the running status of CPU and other hardware, battery quantity, application thread, etc. All the information is the basis of the system functions. Some graphic plug-ins and template are used for designing a friendly interactive interface, which can help users find what they are looking for immediately and set the function modules.

The subject accomplishes power-saving software by analyzing the literature of the subject, case studies and practical research with the technological mean of using application system of the android platform. In actual use, users can set the modules and the current number of the applications according to their demands after installing in the android smart phones, which aims to reduce the consumption of power.

Keywords: android, power management, applications, power-saving mode

目录

第一章 绪论

1.1 课题背景

随着智能机手机爆炸式的发展，越来越多的人开始使用智能手机，智能机的发展使手机成为一个移动终端，具有短信，相机，游戏，互联网等功能，使人们可以在手机上完成自己要完成的事情。但在长时间使用手机的时候，人们最担心的就是电能消耗问题，这属于电源管理的范畴，电源管理不仅历来是最重要的研究课题之一，待机时间的长短也是评价移动设备优劣的重要指标。因此如何最大化的使用电能、减少消耗、如何方便用户使用，如何有效改善用户的工作效率，这不仅是民众的需求，更是智能机研究者亟待解决的问题。

1.2 课题研究意义

随着手机全面进入智能机时代，在中国 android 智能平台已经占据了 90%左右的份额。随着平台占据的市场份额越来越大，平台应用也相应的丰富，这样就使得人们可以通过手机做更多的事情。人们在使用智能机最苦恼的问题就是手机电池，而在日常使用中，最浪费手机电池的是打开的应用程序没有及时关闭，使手机待机时间缩短，给人们的生活带来不变。本系统的研究，致力于解决这一问题，通过对底层应用程序线程的研究，智能的控制应用程序及时关闭，减少电量使用，给人们生活带来方便。

在智能手机的硬件设计中，目前的智能机整合了大量的应用，如视频、游戏、音乐、上网等功能。所以在硬件的设计中，并不能单纯的增加电池的容量，要综合考虑多方面的因素，智能手机的硬件设计比普通手机更为严格。而在目前电池单位体积下，所释放的能量密度是十分有限的，不能满足人们对手机电量的需求。在这样的情况下，电源管理技术应用而生，目前的电源管理技术主要有两方面，即硬件管理和软件管理。硬件电源管理主要是指通过提高电池的电量和优化硬件结构，从而提高智能手机的使用时间。在硬件电源管理中，苹果公司是做的最好的，由于苹果手机的操作系统和硬件系统都是苹果公司开发和生产的，所以其优化的硬件结构能够很好的支持的手机运行。而在 Android 手机中，操作系统和硬件厂商是不一样的，所以其软件和硬件并不能很好的兼容，这就降低了硬件优化的效果。所以要通过软件来优化电量，在软件优化中，主要是调配好处理器的资源来降低能耗，使手机节省更多的电量，从而延迟手机使用的时间。

android 平台的电源管理软件已经发展成手机中不可缺少的一个软件，此软件主要有四个功能，即实时显示电量、耗电量排行、省电模式和设置。通过上述的 4 个功能，用户可以对自己手机的耗电情况有清晰的了解。用户可以通过查看实时电量来决定是否要充电，通过耗电量排行可以查看手机中最耗电的部分，在使用手机中尽量减少耗电量较高的功能。设置功能中包括智能省电等功能。智能省电有两种模式，即手工模式和自动模式，在手工模式中，可以选择关掉应用程序，而在自动省电模式中，程序会自动判断哪些应用程序是不使用的，从而将其关闭。

1.3 研究现状

由于 android 智能手机厂商为了和 iphone 竞争，大屏显示是已经成为手机的标配，目前各大厂商的旗舰机屏幕都在 5 寸左右。这样的屏幕要求提高白色的 LED 背光，一般情况下，一个 LCD 彩屏将均匀到 3 到 4 个二极管，在目前的手机屏幕中，一般彩屏的显色主要有 3 或 4 个二极管，有的显色较好的手机有 6 个或者更多的二极管。由于用户长时间的使用智能手机，需要长时间的背部照明，背光照明是电量消耗的一个重要因素[2]。虽然现在的智能手机电源管理取得了较大的发展，但是还不能满足用户的需求，所以，如何提高智能手机电源管理已经成为软件开发者和厂商考虑的重点。

在 android 平台中，不同的厂商解决耗电量的方式是不同的，从不同的层面来看，主要有以下三种方式。

（1）在手机使用中，屏幕的耗电量是最大的，所以很多厂商从屏幕中的 LED 入手，将白光 LED 驱动成为设计重点。智能手机中 LCD 显示器提供的背灯光源是消耗了大量的电源，其消耗电远远大于显示器所获电量的十倍还多。手机背光灯的功率越大，屏幕的发光能力就越强。但是由于功率现在日趋苛刻，所以现在的厂商都不追求高亮度的输出，目前在市场上，低功耗的白光 LED 灯是较受欢迎的。

（2）芯片高度集成化：还有一些厂家在芯片上集成更多的功能特征，用更好的设计灵活性来实现系统更强的用电性能[2]。目前智能手机越来越精巧，单一功能的手机渐渐的被淘汰，取而代之的是多应用的智能手机，所以其芯片也在发生的改变。

（3）降低能耗：目前国内大部分厂商一直在研究如何提高电池容量，而在软件控制方面研究的较少。在 2011 年才有厂商推出电源管理软件，从而拉开了人们对电源管理软件的研究[3]。本文将致力于软件方面的研究，通过智能的管理应用程序，降低手机电量的使用，从而增加智能手机的待机时间和使用时间，给人们的

生活带来方便。

1.4 论文研究主要内容和方法

本系统严格的按照软件开发的流程进行设计和开发，在本文中，首先对系统进行了分析，包括系统的可行性分析、需求分析、性能分析等，通过系统的分析，可以确定系统是否可行的、系统的主要功能以及对性能的要求。在完成系统的分析后，通过划分功能模块，设计流程图，研究 android 平台相关技术对系统进行详细的设计，首先要对系统的设计原理进行说明，然后对电量实时显示、耗电量排行、智能省电、CPU 省电等功能进行详细的设计。《android 电源管理机制研究与软件设计》课题的研究内容主要有 5 项内容，具体的如下所示：

1、理论和技术的研究

在本文中，将对设计和开发电源管理软件的理论和技术做深入的研究，只有了解了其原理和设计开发的技术，才能为后续的的工作打下基础。

2、系统详细分析

系统分析主要包含可行性分析、需求分析、架构分析等内容，在本课题中，首先要对本课题的可行性进行分析。主要包括技术可行性、经费可行性、社会可行性等内容。在需求分析中，将重点根据使用者的需求总结出系统要实现的各项功能。架构分析是指在不同系统不同的层次中，将使用什么样的技术。如何使用不同技术相互结合，让系统能够稳定、高效的运行。

3、系统详细设计

系统设计是软件开发的核心，在确定完需求后，要根据所使用到的各种技术，对系统进行详细的设计。系统设计包括架构设计、功能模块设计、数据交互设计等内容。在架构设计中，要根据系统面向的对象，以满足用户的需求为基础，设计软件系统的具体架构。在功能模块设计中，要根据需求中确定的功能，首先要把功能抽象对象，然后通过对象的属性进行相互的联系，从而找到对象和对象的关系，在设计系统流程时要按照其关系[4]。系统的设计觉得项目的成败，所以在完成软件的设计后，要进行仔细的推敲，考虑存在的各种问题，并提出具体的解决方案，以便后续的开发和使用能顺利的进行。

4、系统编码和测试

系统的开发平台是 Eclipse，在这个平台上使用 Android SDK 开发技术对系统进行设计和开发。编码是实现系统各项功能的关键。测试是根据需求中所确定的功能，对已经实现的功能进行测试，从而保证系统的能够正常使用，并且实现了需求中所确定的内容。

5、结论和展望

本文研究的最后一部分内容是对整个系统进行总结和展望，通过对开发本套系统的总结，得出自己所学习和掌握的知识，为以后工作学习打下了那些基础。同时，要对课题的前景进行展望。

由于国内对智能手机电源的管理的研究较少，而且往往集中在硬件的研究上。在本课题的研究中，主要有三方法，即分析文献法、实例研究法和实践研究法。在分析文献法中，将根据本课题的主要内容，收集各方面的资料，对其进行研究和总结，进而得到研究本课题的思路。实例研究法就是对别人已经做成的电源管理系统进行分析，通过对功能、界面、实现原理等内容的分析，得出本课题电源管理软件的具体功能和内容。实践研究法，就是根据文献分析、实例研究得到的结论和功能，通过 eclipse 开发工具和 android SDK 开发出一套智能手机电源管理软件。

1.5 论文结构

本论文主要分为七部分，各部分主要的内容如下所示：

第一部分是绪论部分，主要的说到了本课题研究的背景和意义，重点说明了目前各厂商对电源管理的设计理念。最后对研究的方法和主要内容进行了阐述。

第二部分详细的介绍了研究本课题所用到的基础理论和关键技术，主要有 Linux 底层技术、android SDK 开发语言等内容，在本章的最后完整的讲述了本系统开发的架构。

第三部分是对本系统进行了详细的分析，包括可行性分析、功能分析、性能分析等内容，通过这些内容的分析，可以确定出本系统所要开发的功能模块，并总结出了系统要满足的性能需求。

第四部分是本文的核心，根据第三章确定的功能、性能、用户等需求，对系统进行详细的设计。本文的详细设计包括架构设计、流程设计、数据模型设计等基础内容。设计的内容要满足第三章所确定需求，并要考虑各种可能出现的情况，做好系统的应急准备。

第五部分是系统的实现部分，根据第四章的设计的内容，通过各种技术进行实现。在本章中，主要有环境代价、界面实现、核心代码实现等内容组成，本章实现的内容，不仅要符合第四章的设计，而且还要满足第三章的需求。

第六部分是测试部分，本章对系统进行了全面的测试，包括功能测试、单元测试和整合测试，列出了测试用例和测试界面，从而保证软件的正常运行。

第七部分对本文进行了总结和展望，在本章中，主要对系统开发过程、使用

技术、自己心得等内容进行总结，在总结中学习各种经验和知识，为后续的工作和学习打下基础，并对电源管理系统进行了未来展望。

1.6 本章小结

在本章中，主要对本课题研究的背景和可以研究的意义进行了介绍，重点说明了对目前各厂商对电源管理的设计。最后对本文研究的方法和主要内容进行了阐述。

第二章 相关理论及技术

本章将对研究本课题使用的基础理论和技术进行详细的说明，包括底层的Linux 操作系统、android SDK 开发语言等。

2.1 Android 平台

在 2008 年，谷歌公司开发了一款专门应用于智能手机的新型操作系统，即Android 平台，其内部采用的是 Linux 内核来做底层，囊括了智能手机工作所需要的几乎所有工具 [5]。从 2008 年推出 android 操作系统后，在短短的 2 年间，就超过苹果成为市场占有率首位的智能手机操作系统。Android 平台主要有 4 部分组成，具体的结构如图 2-1 所示：

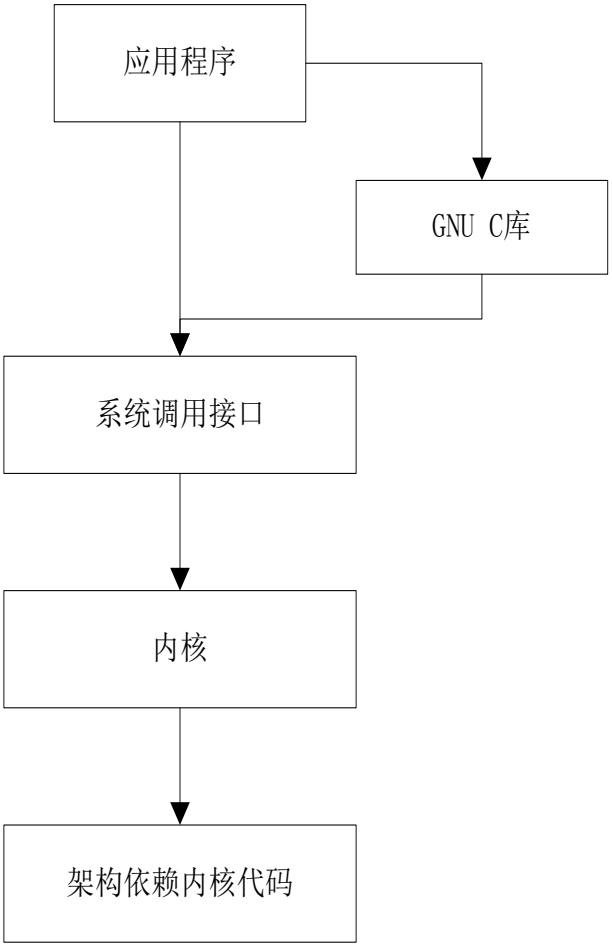

图 2-1 android 结构图

如 2-1 图所示，第一层为 android 主要有操作系统层(OS)；第二层为各种类库

和 Android 运行环境；第三层为系统调用接口；第四层为应用程序。系统调用接口具有承上启下的作用，具体的功能如图 2-2 所示：

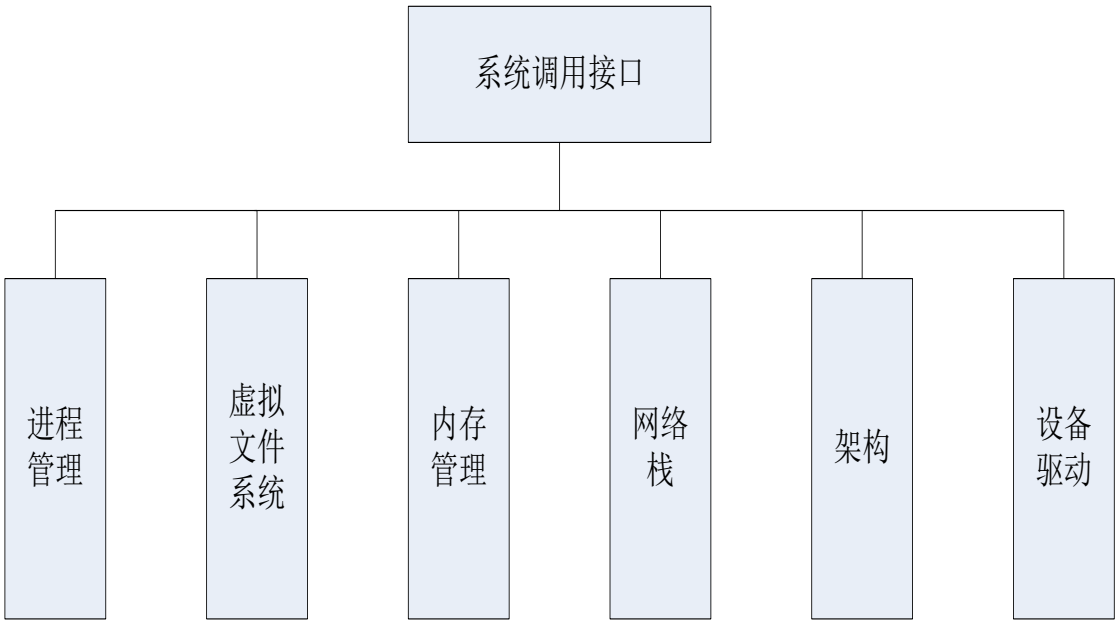

图 2-2 系统调用接口功能图

2.1.1 操作系统层

Android 的操作系统层是使用的 Linux2.6 的内核，在这平台上的全部操作均为以 Linux 为基础的内容而实施的。作为标准技术的 Linux2.6 具有一定程度的开放性[2]。该操作系统对 android 的支持主要有两部分，即核心和驱动程序两个部分。Android 的 Linux 和谐为标准的 Linux2.6 内核，由于 android 是一个平台，所以要与很多其他移动设备相关的驱动程序。主要的驱动程序有以下几种：

(1) 显示驱动：主要用于 Linux 的帧缓冲的驱动，在不同的屏幕上显示内容。

在 android 系统中，显示驱动是非常重要的组成部分。在底层的实现中，大部分是由 surface 库来做到的，在底层系统中，提供了基本的输出设备的封装。在 surface 这个库中，提供了对多图层的支持和各个图层间效果等功能[5]。

(2) 音频驱动：在 android 系统中，音频采用的驱动是 linux 的 ALSA，ALSA 目前是 linux 最为常用的音频体系结构。在 linux 内核驱动层，ALSA 提供了 alsa-driver，应用层中，alsa-lib 为 ALSA 所给出，在这个 lib 中提供了处理音频的 API，使用里面的方法即可。

(3) Wifi 驱动：随着无线网络的发展，Wifi 已经成为各类终端的标配，在底层驱动中，wifi 可以借助于 IEEE802.11 标准中的驱相应动程序而实现[5]。

(4) 键盘、蓝牙以及 Binder IPC 的驱动：值得说明的是，为了能够将系统各个

进程之间的通讯功能有效的提高，设计者在 android 中设置了一个比较特殊的驱动程序，使得不同进程分别承担一定的设备节点[6]。

2.1.2 各种类库及 Android 运行环境

在 android 体系中，连接底层的 Linux 和应用程序的是各类类库。类库相对于一个嵌入式系统，类似于 web 系统的服务器，相当于一个中间件。在这个中间件中，主要分为两部分内容，一是 android 运行环境，另一个是供应用程序运行的各种类库，在这个中间层中，大部分内容是用 C++实现的。类库主要有以下几种：

(1) C 类库：称为微软基础类库，是实现系统设计的基础数据类型。

(2) 多媒体框架库：在这个库中包含了 Android 多媒体的核心部分，例如基于 PacktVideo 和 OpenCORE。从功能上来看，这个库分为了两个组成，其一是音频、视频的回放库，其二则是音视频的记录库，即(Recoder)[6]。

(3) SGL 库：主要是处理 2D 图像的引擎。

(4) SSL：主要为数据通讯提供各种支持，其全称为 Secure Socket Layer，位于 TCP/IP 协议与各种应用层协议之间[7]。

(5) OpecGL ES 1.0：这个类库较新，主要提高对 3D 的支持。

(6) 界面管理工具：本部分提供了对管理显示子系统等功能[7]。

(7) WebKit：是在应用层开发浏览器的核心类库[7]。

(8) ZiyouType：主要处理位图和矢量字体[7]。

上述是 Android 提供的各种操作库，大部分的应用程序开发都是基于这些类库进行的，其最鲜明的特征为与移动设备所使用平台之间的应用程序具有非常紧密的关系[8]。Android 主要运行在一个虚拟机上，其运行环境就是一个虚拟机，即 Dalvik。这个虚拟机和一般运用 JAVA 的虚拟机不同，Java 的虚拟机执行的是.class 文件，而 Dalvik 执行的.dex 文件[8]。当该文件被执行时，其中每个应用程序均为单独的进程，在 Linux 系统中进程就是 Process。产生这样的不同，主要是虚拟机的构成不同。JAVA 虚拟机是以基于栈的虚拟机，而 Dalvik 是基于寄存器的虚拟机[9]。基于寄存器的优势就是能够基于硬件的不同实现较大程度的优化，更加适合应用在移动设备中。

2.1.3 应用程序框架

应用程序框架是为应用程序而设计的一个框架，在这个框架中为开发者提供了各种 API。其实际就是一个应用程序的框架，在开发 Android 应用程序时，使用的是 Java 语言，所以在应用程序框架层拥有 UI 程序中所需要的各种控件[8]。在一

个 android 应用程序中，应用程序框架提供了以下几个部分，即 Service（服务）、Activity(活动)、Broadcast Intent Receiver（广播意图接收者）以及 Content Provider（信息内容提供者）[9]。

2.1.4 应用程序

应用程序是人们通常在手机上使用的程序，这些应用程序主要是通过 java 语言编写的，java 语言通过调用应用程序框架所提供的各种 API，从而实现各种功能。Java 编写好的文件将被打包成 APK 文件，这个 APK 包里面包含了各种类型的文件，如图片、声音、文本等。在 Android 界面中，其提供了很多基础的功能，例如返回键、联系人、照相机、浏览器等很多基础的应用。开发者可以在这些基础的功能上实现自己的程序，通过调用应用程序框架层的各种 API 来实现自己的程序。

2.2 java 开发语言

2.2.1 java 语言简述

Java 语言是一个纯面向对象的程序设计语言，是由 Sun 在 95 年 5 月给出的。由于 java 开发语言的跨平台、高性能、安全性等特性，使其迅速的成为炙手可热的程序设计语言，在各行各界中都有较为广泛的应用。目前 java 主要分为三个方向，即 JavaSE、JavaEE 和 JavaME 等内容。

Java Platform Enterprise Edition (JavaEE) 也叫做 J2EE，即 Java 的企业级版本，是 java 使用最为广泛的一个方向，特别是在企业级的开发中，JavaEE 是首先的开发平台，在金融、电信、物流、政府等行业中，已经占据了绝大多说的市场份额。Java EE 下有 Hibernate、 Spring、Struts 等开发框架，根据应用领域、项目大小，除了 Java EE，还有如 LAMP 组合（Linux、Apatch、MySql、PHP 组合）、. Net、Ruby 等其他技术平台可以选择，后两者更适用于轻量级 Web 领域。

JavaME 主要用于移动设备的程序的开发，在 2G 时代，主要是使用 java 语言开发应用程序和游戏，目前在智能机时代，android 上的应用程序开发也是使用 javaME 来进行的。Android 可以提供从应用软件到操作系统的控制，如浏览器等；而 Java EE 不能调用操作系统的资源。

2.2.2 java 语言的特性

Java 语言使用的如此的广泛，主要和其设计的合理有很大的关系，java 语言的设计主大的特点就是面向对象。

从 1960 年开始，面向对象这一思想就出现在了程序设计中，最先出现的开发

模式具有时尚前卫，但是价格很贵的特点，经过 40 多年的发展，逐渐成为主流模式。现如今，程序设计的每个环节都离不开面向对象的技术，无论是比较常见的应用型桌面还是大规模的服务应用框架，甚至是各种操作系统中所配备的内核以及底层的驱动程序等，都运用到了面向对象技术。该技术把行为与数据结合起来组成一个对象，然后在此基础上构建一定的程序设计方案。需要注意的是，该过程中的对象并没有完全呈现出所有的步骤，其中的每个对象均为独立的个体，他们用调用消息的方法进行交互，其中最为便捷的是方法的调用。将属于一个类型的对象称之为类，它可以代表属于这一类型的每个对象，或者能够产生很多对象，同时，它也能够继承别的类。面向对象技术是程序设计时常用的一种思想，它将某一程序的实体最为单独的对象，从而将程序中每个模块之间的耦合消除，进而将各模块之间的相互关联理顺。在软件开发这一漫长的过程中，当某一环节出现某种问题的时候，经过多次实践总结，得出的能够被多次复制的有效的解决方案就是设计模式。其基于面向对象，是一种程序设计方法。就现阶段来看开发程序通常都是借助于高级语言来实现，而面向对象技术则依靠对象系统来实现，但是一个完整的对象系统则需要更大的类型系统的支撑。因此，面向对象技术要想做到，类型系统是基础，它定义了对象的基础数据类型，接口类型，类型以及类类型，以此为基础产生了类，接口等。Linux 作为常用的操作系统，没有针对对象的类型系统，但是它通过内核建立了相对简洁的对象系统 kobject。通常情况下我们可以把面向对象思想根据支持动态类型与否划分成动态与静态这两种类型，比如动态型中的 smalltalk，静态型如常用的 c++等。需要说明的是在实现 c 语言的过程中也用到了动态型的 gobject，有些时候还会借助于闭包(closure)等普通动态语言来实现，因此最初出现的 c 语言通常具有比较强大的功能。最近出现的各种动态语言，如 python、ruby 等，均被广泛的应用到动态性的面向对象系统中[10]。

总的来说，面向对象涵盖了封装，继承，多态以及接口这四个内容，经过多方位，多视角的抽象实现。把数据，方法等集中于对象的实体中，就完成了封装。将父类的数据和方法传给子类，就是继承，这一过程展现了类之间的关系。用某一个方法将各个对象调用，从而有多种多样的行为出现，这一过程就是多态，它就是面向对象的中心环节。模块和类的操作界面就是接口，它仅仅是一个操作指南，并没有实际存在，真正的实现是靠具体的模块和类的，并且该模块和类具有此接口[10]。

面向对象思想已经有多种实现，有的实现是完整的，如 c++和 java 语言，有的只是简单的具有动态特性，如 smalltalk、python、rrby 等。有些不难做到，像不少采取 c 语言所写开源软件中的面向对象实现，linux 内核中面向对象的实现就是

Kset 和 kobject，实现了一不复杂的设备类层次。使用面向对象技术编写程序时，要时刻想着面向过程的程序设计方法，不少时刻要求将面向过程和面向对象相融合才能使程序简洁高效，面向对象思想需要灵活的实现才能得到最好的效果。设计模式不仅可以在面向对象的程序设计语言中使用，也可以在 c 语言，或者是汇编语言中使用，设计模式需要在特定的场合下使用，不能生搬硬套，需要根据实际的情况变化，设计模式一般提供了系统的在多种情况下可用的解决方案，可以更快速的思考设计，减少与程序交互时的障碍。

2.2.3 java 语言开发环境

在 window 平台上，使用 java 语言开发程序首先要配置开发环境。在 java 提供的官方程序中，主要有两个包，一个是 JDK，一个是 JRE。JDK 是系统的开发环境，里面存放了系统开发所要使用的各种 jar 文件。而 JRE 是运行 java 程序的环境，里面保存了各种运行 java 程序的文件。安装开发环境主要有两个步骤，具体如下：

（1）安装 java(TM) SE Development Kits,即 JDK。

（2）设置环境变量：

右击 电脑->属性->高级->环境变量，该时刻能够发现下面的环境变量：Path：系统在所有路径下均能够认识 Java 命令。

ClassPath：为 Java 加载类(包括 class，lib)的路径，只有类在 classPath 中，Java 命令才能识别。

JAVA_HOME：Java 的安装路径。

这三个变量是必须正确设置的，否则 Java 命令将可能不会正常运行，具体设置如下：

Path: ;C:\Progran Files\java\jdk1.6.0_01\bin

Classpath: .;Progran Files\java\jdk1.6.0_01\lib\dt.jar;Progran Files\java\jdk1.6.0_01\lib\tools.jar

JAVA_HOME: Progran Files\java\jdk1.6.0_01

2.3 android 内存管理

在使用 android 开发电源管理的应用程序时，其实就是管理 android 内存。Android 内核是基于 Linux 2.6 内核的加强内核版本，给出了 Android 平台的设备驱动，还修正了部分错误，其核心驱动如表 2-1 所示。

表 2-1 android 内核驱动

编号	驱动名称	说明	原代码存在的位置
1	*Android Binder*	基于*OpenBinder*框架的一个驱动，负责*Android*平台进程间的通讯	*drivers / staging / Android / binder.c*
2	*Android* 电源管理	基于*Linux*电源管理系统的*Android*电源管理驱动。	*kernel / power / wakelock.c*
3	低内存管理器	可以根据适当的要求，结束进程，并释放相应的内存。	*drivers / staging / Android / lowmemorykiller.c*
4	匿名共享内存	为进程提供大块共享内存，并提供相应内存管理机制。	*mm / ashmem.c*
5	*Android PMEM*	为单一用户提供连续的内存区域，某些设备和 DSP 只能工作在相应的内存区域。	*drivers / misc / pmem.c*
6	*Android Logger*	抓取*Android* 系统的各种日志。	*drivers / staging / Android / logger.c*
7	*USB Gadget* 驱动	一个基于*Linux USB gadget*驱动框架的设备驱动。	*drivers / usb / gadget /*
8	*Android Ram Console*	*Android*将调试日志信息写入一个被称为RAM Console的设备里。	*drivers / staging / Android / ram _ console.c*

任何 Android 应用运行在它自己的 Dalvik 的实例中单独运行，管理进程和内存的管理工作，对需要的进程进行终止，并释放相应的内存。Android 和 Dalvik 都负责处理底层硬件交互操作，内存和驱动程序的管理组成，API 负责提供底层硬件的访问和服务。Dalvik 虚拟机基于 Linux 内核线程，负责一起运行多个实例。Android 在内存管理中，采用的进程管理和 Linux 是完全不同的。Linux 的进程管理策略是进程在活动停止后，就结束了其进程。而在 android 中，活动停止的进程将会保存在内容中，这在一定程度上就比较耗电，等到 android 系统打开了其他的程序，才把原来占用的程序停止掉。这样设计的好处主要有两点，最大的好处是当用户激活这些进程时，将提升进程的启动速度，提高用户的体验。这些进程保存在内存中，对整体的 android 体系运行速度并没有产生太大的影响[12]。

最典型的例子就是小米手机，小米手机对 android 平台进行了深度的优化，特别是对内存的使用进行了改变，为了提高用户的体验。小米手机在使用中，其内存占有率是非常高的，占用了大量的内存才获得了较高的稳定性，相对出现 FC 的

情况也较少。小米的内存相对现在的电子市场的软件可以承受，当 Android 应用被切换到后台运行时，只是保留运行状态，实际上已经暂停使用，不会消耗资源，所以有的程序切出去可以继续处理，如播放音乐，而有的程序切出去会返回主界面。当一个应用开启时，该应用会在后台持续运行，并消耗电量，而没有运行的应用是不会消耗电量的，这是非常有利于程序启动的设计，可以加快程序启动的时间，并不会加载界面资源

图 2-3 进程管理软件分析

Android 手机的很多应用程序都不存在结束程序，这与系统的调度机制相联。了解 Java 可以更有助于了解 Android 手机机制。Android 手机系统存在一个内存管理机制，系统中存在内存阀值，系统会将低于内存阈值的应用排列为一个列表，逐次关闭这些应用，这个内存阈值非常小，所以手机内存会经常在某一数值上下波动，但不会影响手机运行速度，但是会加快下一次应用启动速度，如图 2-3 所示。这是 Android 手机的优势，关闭该进程，则显得多余，并不能利用 Android 手机的优势。但是这也是 android 手机费电的一个主要的原因，由于内存维持在一个较高的水平，其耗电量就会很高，所以在进行电源管理的时间，就要把不使用的内存关掉，在减少电量消耗的同时，客户的体验度也随之降低了。

2.4 应用程序组件

在 2.1 中已经对 android 的体系进行了详细的说明，各个层的组成内容也进行重点的说明。在本小结中，将对应用程序开发的组件进行重点的分析，应用程序的组件主要有 4 个部分组成，即活动、服务、广播接受者、内容提供者，这四部分内容是应用程序开发的基础。

2.4.1 活动(Activity)

Activity 是 Android 组件中出现频率最高的基本型组件，每个 Activity 在 Android 应用都可以作为独立屏幕而存在。当应用过程中的 Activity 需要进行不同状态之间的转变时，通常会借助于如下保护方法来显示[12]：

```
void onCreate(Bundle savedInstanceState)
void onStart()
void onRestart()void onResume()
void onPause()void onStop()
void onDestroy()
```

在实现 Activity 类的时刻，借由覆盖相关方法就能在你想要处理时刻实现调用。在开发过程中，并不用重新这些方法，直接调用这些方法即可。Activity 活动说明如表 2-2 所示。

表 2-2 activity 活动说明

编号	操作	说明
1	onCreate	负责应用的初始化工作，具有一个参数，默认为（null），通常保存之前调用 onSaveInstanceState ()方法的状态信息。
2	onStart	将方法展示给用户。
3	onPause	负责一个活动从前台切换到后台，并保持活动的运行状态。
4	onStop	当用户不需要某个活动，该方法负责终止不需要的活动。如果系统内存匮乏，则系统直接释放该活动占用的内存，而不会触发 onStop 方法。所以保存状态信息不是 onStop 时做，而是应该在 onPause 时做。
5	onRestart	负责将后台运行的活动重新切换到前台，展示给用户。
6	onDestroy	负责终止某活动，当系统内存匮乏时，则系统直接释放该活动占用的内存，而不会触发 onDestroy 方法。
7	onSaveInstanceState	负责保存以前的状态。

2.4.2 服务(Service)

服务是 android 应用程序中后台所使用的，服务是前台应用程序看不见的。Service 是一种长生命周期，无使用者界面的程序。同时，Service 只能在后台运行，不能单独运行，其他的与 Activity 相同，于其他组件进行交互，是 android 系统中的一种组件。Service 有 context.bindService()和 context.startService()两种方启动方式。使用 context.startService() 启动 Service。

context.startService() −>onCreate()−>onStart()−>
Service running context.stopService() | −>onDestroy() −>Service stop

假使服务尚未运行，则 android 系统先调用 onCreate()方法，之后借助于 onStart()方法的调用来实现；若服务已经处于运行状态，那么系统仅调用 onStart()，所以可以多次重复调用 onStart 方法。stopService 方法直接 onDestroy。通过 stopService 方法来关闭服务。 startService 的生命周期为：

onCreate −−> onStart(可多次调用) −−> onDestroy，使 用 context.bindService()
启动 Service 会经历: context.bindService()−>onCreate()−>onBind()−>
Service running onUnbind() −>onDestroy() −>Service stop，

2.5 CPU 省电特点

在 Linux 系统的电源管理中，Suspend/Resume 具有非常重要的地位，具体来说，当系统处于未使用状态时，Suspend 能够使其转变为休眠或者低耗状态，通过该方式能够使系统电源用量大为降低。需要指出的是 Linux 系统中所包含的 Suspend 存在四种不同的状态，因此能够适用于不同的电源管理接口或者是体系结构，但是从本质上来看其差别比较小。

Linux 系统中电源管理所包括的 Suspend 框架根本上与 Linux 系统中的驱动模型（Linux Driver Model）具有一定程度的相关性，从本质上来看均是以 Linux 的驱动模型为基础而实现的。

Linux 系统中的 Suspend 系统包括核心层与平台层，其中前者与平台没有关系，而后者与平台相关。核心层中包括了各种操作接口，其中与平台相关的各个部分是基于 Linux 系统中的驱动模型来实现的，根据 Device 结构实现对设备的描述，并且将 device_driver 所谓设备驱动，借助于 class、type 和 bus 来描述设备相对应的类别、类型以及总线等各种属性。从此模型出发我们也可以将设备之电源管理分成驱动级等四级[20]。当某设备能够借助于 class 或者 bus 来正确管理其内部电源时，此时可以将驱动级别中的 suspend/resume 设置为空值。该方案能够提升电源管

理过程的灵活性和高效性。Android 系统提供 android.os.PowerManager 类，负责切换控制设备的电源状态，共有 3 个对外接口:

1. void goToSleep(long time)：强制进入睡眠状态。
2. newWakeLock(int flags, String tag)：解除相应层次的锁。
3. void userActivity(long when, boolean noChangeLights)：切换事件状态到 Full on。

2.6 本章小结

本章主要对该文要研究的相关理论和技术进行了深入的探究，包括了 android 平台的组成、java 开发语言的特性及开发环境的搭建、android 内存管理、应用程序组件管理等内容。通过这些理论及技术的研究，为后续的需求分析和程序设计打下了基础。

第三章 系统分析

在本章中，将对系统进行详细的分析，包括需求分析、架构分析等内容。系统分析主要包含需求分析、系统功能分析、架构分析等内容，在本课题中，首先要对本课题的需求进行分析。在需求分析中，将重点根据使用者的需求总结出系统要实现的各项功能。架构分析是指在不同系统不同的层次中，将使用什么样的技术。如何使用不同技术相互结合，让系统能够稳定、高效的运行。

3.1 需求分析

3.1.1 需求分析概述

本系统的需求分析要通过两个方面实现，首先根据用户需求确定要用到的功能和技术，对这些技术进行研究和分析，从而得出大概的系统模型。再根据所要解决的问题，确定出系统要完成的功能。最后根据研究和分析的技术对要完成的功能进行验证，确定功能是否能够实现。该智能节电软件是基于 Android 系统，通过对当下最流行的开源系统平台的应用开发，解决 android 平台智能手机耗电量过大的问题，在使用利用移动电子设备的快捷方便的优势特意开发的一款电源管理系统软件。在智能手机节电软件中，主要有耗电量排行、省电模式选择、电量显示、剩余时间显示等内容。

3.1.2 系统功能概述

由于硬件的电池和使用的软件不能做到统一，这就造成了电池使用不够充分的问题，人们不得不为电量消耗过快而苦恼。在这样的情况下，就需要一个专门管理电源的软件，通过控制后台程序来降低电量的消耗。随着 android 版本的不断提升，其内部的核心组件也在发生着变化，所以电源管理软件也在不断的改变。特别在美国，每当 android 推出新的版本后，电源管理软件也就会升级到新的一个版本。

Android 是以 Linux 为内核开发的一种开源操作系统,主要使用于嵌入式移动设备。而在嵌入式移动设备的研发中电源管理一直都是开发人员研究的重要课题之一。一套良好的电源管理方案不仅可以达到节约电量、延长电池寿命等目的，而且还可以降低辐射、延长设备的使用寿命等。对于包括手机在内的各种移动设备而言，电源管理功能至关重要，特别是针对于手机电脑化的移动设备或者是

Android 平台的智能手机而言，电源管理是非常有必要的。

在实际的应用中，用户对电源的管理主要有几个方面，首先是要对电量进行查看，并想知道每个程序运行消耗的电量，以便知道那个应用程序是最费电的，并可以根据从高到低、从低到高的对耗电的应用程序进行排行，并且要显示可用 2G 通话时间、可用 3G 通话时间、上网时间、阅读时间、游戏时间等内容。第二用户想通过这个软件对手机的省电模式进行选择，省电模式是对一些基础的程序进行管理的一个统称，如普通模式、极限模式等，在极限模式中，手机的蓝牙、wifi、数据包等都将被关掉。软件在给出几种省电模式后也要给出自定义的模式，让用户根据自己的实际需求对省电的应用程序进行设置。

本文通过分析系统功能，电源管理软件主要有以下几个功能，如图 3-1 所示：

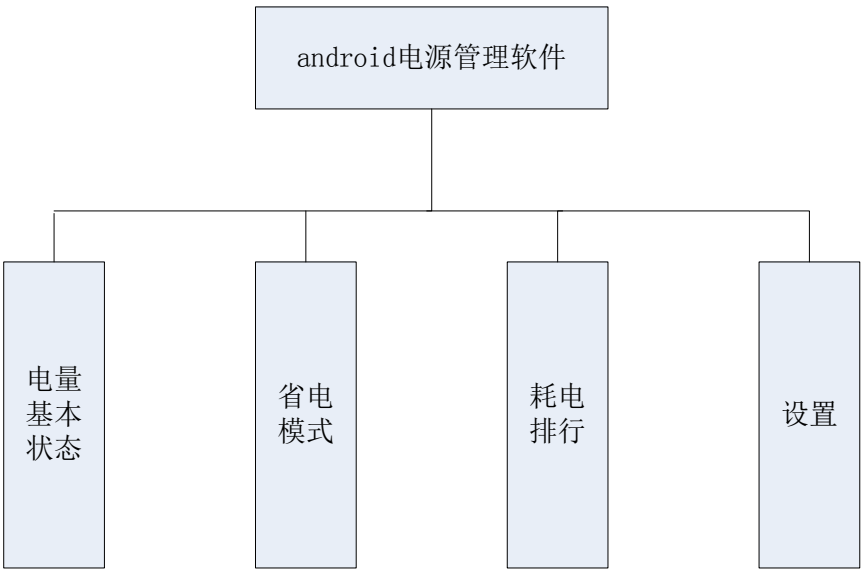

图 3-1 android 电源管理软件功能图

从 3-1 中可以看到，本系统的主要功能有 4 个，即实时电量基本状态、省电模式、耗电排行和设置。上述 4 个功能基本上满足了人们的日常需求，

3.2 系统功能分析

3.2.1 基本状态模块

基本状态模块上是系统的基本功能，在目前的 android 手机中，android 系统会自动的提供实时电量，在设计本系统时，要把这个数据截取出来，并在软件中显示出来。当用户打开这个软件的时候，最先看到的是手机目前的电力，并要显示还能待机多长时间，是否在充电等内容。用户看到这些内容后，将会根据软件的提醒进行及时的充电。

基本状态是电源管理系统最基础的功能，基本状态主要包含了剩余电量、可用待机时间、可用通话时间、可用视频播放时间等内容，具体的功能如图 3-2 所示：

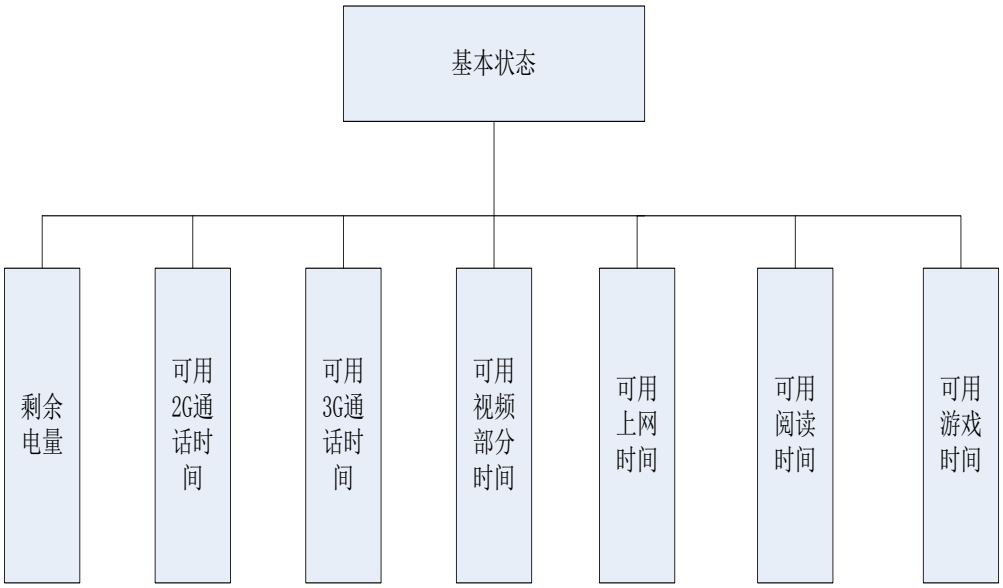

图 3-2 基本状态功能图

从图 3-2 中可以看出本系统基本状态所具有的功能，剩余电量是通过代码取得 Linux 中的电量数据。其他内容的实现，是根据目前手机具有的电量和各种耗电的比例来显示的，具体的算法在系统的实现中将有具体的体现。

3.2.2 省电模式模块

省电模式：省电模式是本系统的重点，是根据用户的设置对基础应用程序的一种管理，主要有普通模式、极限模式、强力模式、自定义模式等。设置的主要内容有 WIFi、2G/3G、GPS、自动同步、触感反馈、蓝牙、屏幕超时、自动旋转屏幕、屏幕亮度自动调节等。通过对模式的设计，系统在运行时会对相应的内容进行关闭，从而实现减少电量的使用，具体的功能图如图 3-3 所示：

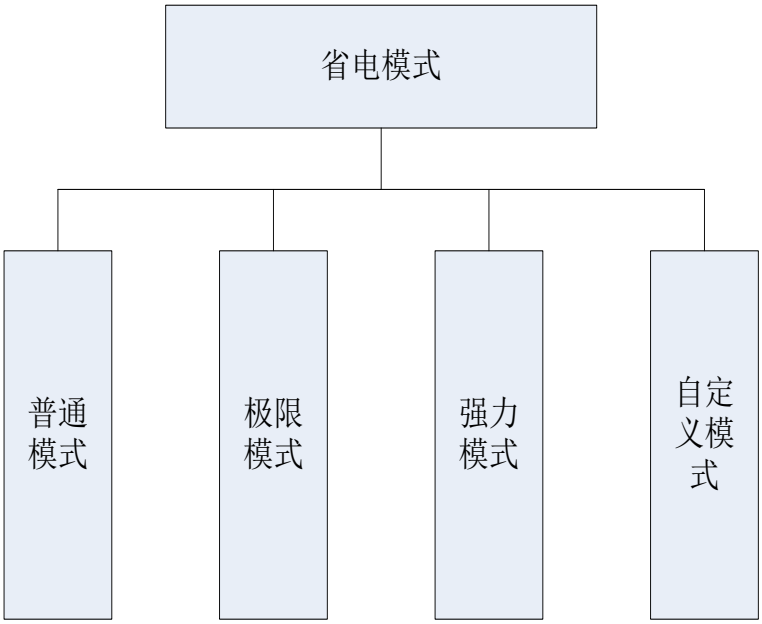

图 3-3 省电模式功能图

如图 3-3 所示，在省电模式模块中，主要有 4 种模式，即普通模式、极限模式、强力模式和自定义模式。目前各大厂商的旗舰机屏幕都在 5 寸左右，这样的屏幕要求提高白色的 LED 背光，一般情况下，一个 LCD 彩屏将均匀到 3 到 4 个二极管，在目前的手机屏幕中，一般彩屏的显色主要有 3 或 4 个二极管，有的显色较好的手机有 6 个或者更多的二极管。由于用户长时间的使用智能手机，需要长时间的背部照明，背光照明是电量消耗的一个重要因素。虽然现在的智能手机电源管理取得了较大的发展，但是还不能满足用户的需求，所以，如何提高智能手机电源管理已经成为软件开发者和厂商考虑的重点。选择不同功能手机在运行过程中关闭的软件是不一样的，具体如表 3-1 所示。

表 3-1 省电模式表

功能	极限模式	强力模式	普通模式
WIFi	关闭	关闭	开启
2G/3G	关闭	开启	开启
GPS	关闭	关闭	关闭
自动同步	关闭	关闭	关闭
触感反馈	关闭	关闭	关闭
蓝牙	关闭	关闭	关闭
屏幕超时	关闭	关闭	关闭
自动旋转屏幕	关闭	关闭	关闭
屏幕亮度自动调节	关闭	关闭	开启

从表 4-3 中可以看出，设计的自动设置的省电模式有三种，还有一种是自定义模式，这种用户可以根据自己的喜好进行调整。用户选择的模式将会被记录到内存中，当开启手机或者打开软件的时候，系统将首先去后台内存中读取模式，然后在手机运行中按照选择的模式开闭功能。

3.2.3 耗电量排行

耗电量排行，这个功能是电源管理软件必须要有的功能，用户在看到自己手机使用电量较快的情况下，就会通过这个功能对目前所使用的功能进行查看，确定是那个应用程序耗电量是最高的。在一般的情况下，手机的屏幕是耗电量最高的，耗电量排行具体的功能如图 3-4 所示：

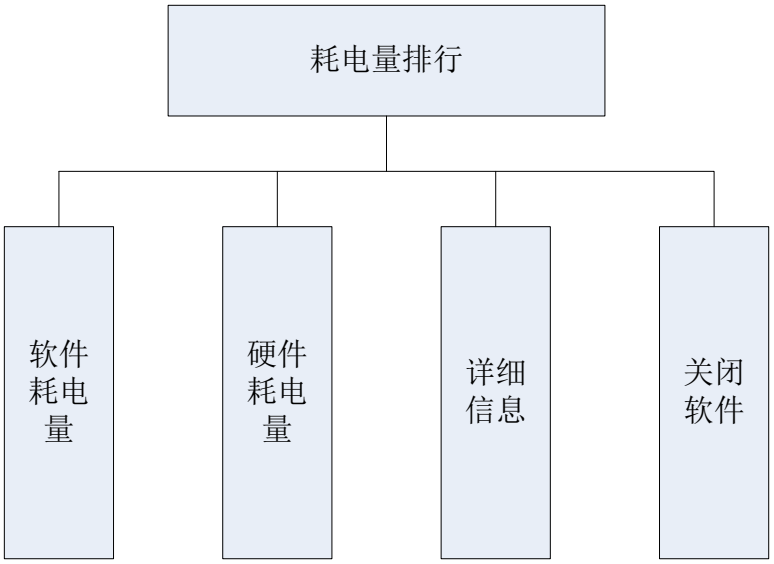

图 3-4 耗电量排行功能图

如图 3-4 所示，在耗电量排行模块中，主要有软件耗电量排行、硬件耗电量排行、详细信息和关闭软件四个功能。软件耗电量排行功能是查看手机中各种软件消耗的电量，并对这些软件进行排名。硬件耗电量排名是查看手机中各个硬件的耗电量，并对这些硬件进行排名。详细信息功能是在用户对各个软件和硬件耗电的详细信息进行查看。当用户查看某个软件消耗电量过多时，而且这个软件自己不经常用，可以选择关闭这个软件，所以在软件中提供了关闭软件功能。

3.2.4 系统设置

系统设置是对本软件一些功能进行设定，主要有智能省电、CPU 调频省电、软件使用说明、版本说明和退出程序功能。智能省电是指对应用程序进行管理，通过关闭应用程序来释放内存，从而降低手机的耗电量。具体的功能模块如图 3-5

所示：

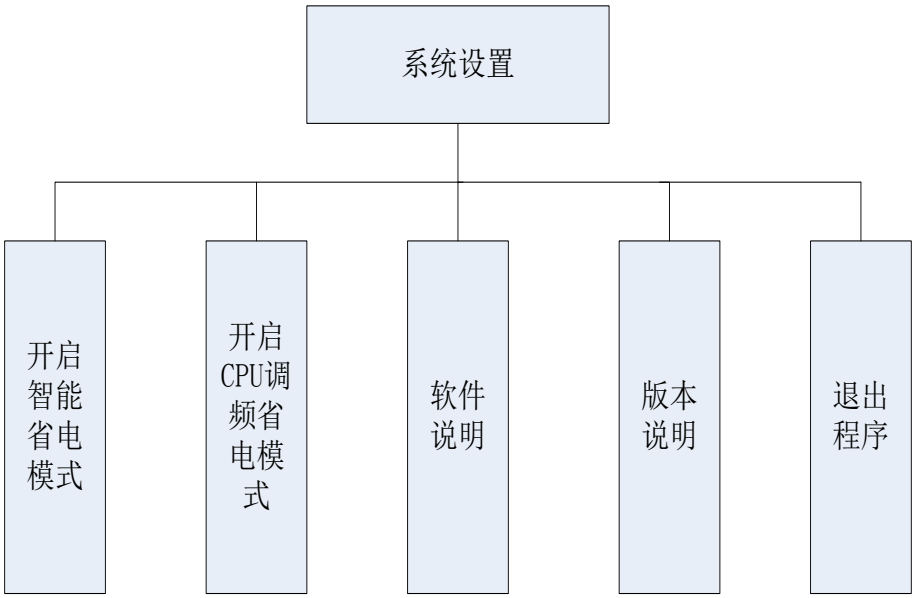

图 3-5 系统设置功能模块图

在本系统的省电模式中，主要有两种方式，一种是自动省电模式，一种是手动省电模式，在智能省电模式中，手机后台程序将会判断哪些应用程序是可以关闭的，哪些不是能关闭的，系统判断可以关机的应用程序将会自动的被关闭掉。关掉不必要的应用程序，系统的内存将得到有效的是释放，从而减少耗电量。而手工省电模式是系统把一些可以关闭的应用程序提醒给用户，用户根据自己的实际需求将这些应用程序关闭，从而起到节省电量的目的。CPU 省电：CPU 并不是经常用的功能，主要是很多用户对 CPU 不是很了解，在本系统中，提供的功能是用户对 CPU 进行设定，如果 CPU 超过了设定的范围，系统将自动清理内存，从而降低 CPU 的压力，系统在一定的范围内运行。

3.2.5 系统流程图

系统流程图如图 3-6 所示：

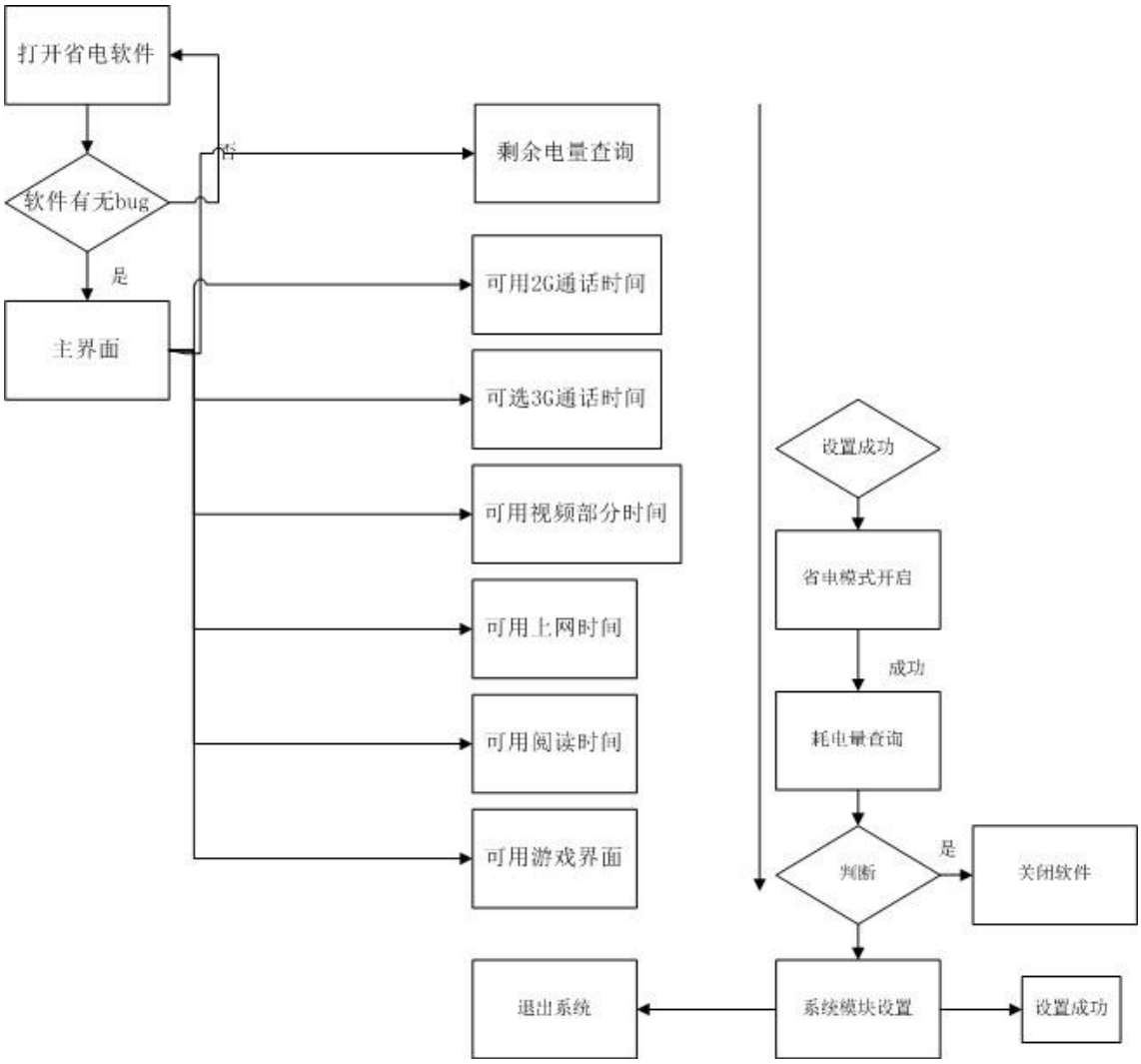

图 3-6 系统流程图

3.2.6 系统总体模块图

系统总体模块图如图 3-7 所示：

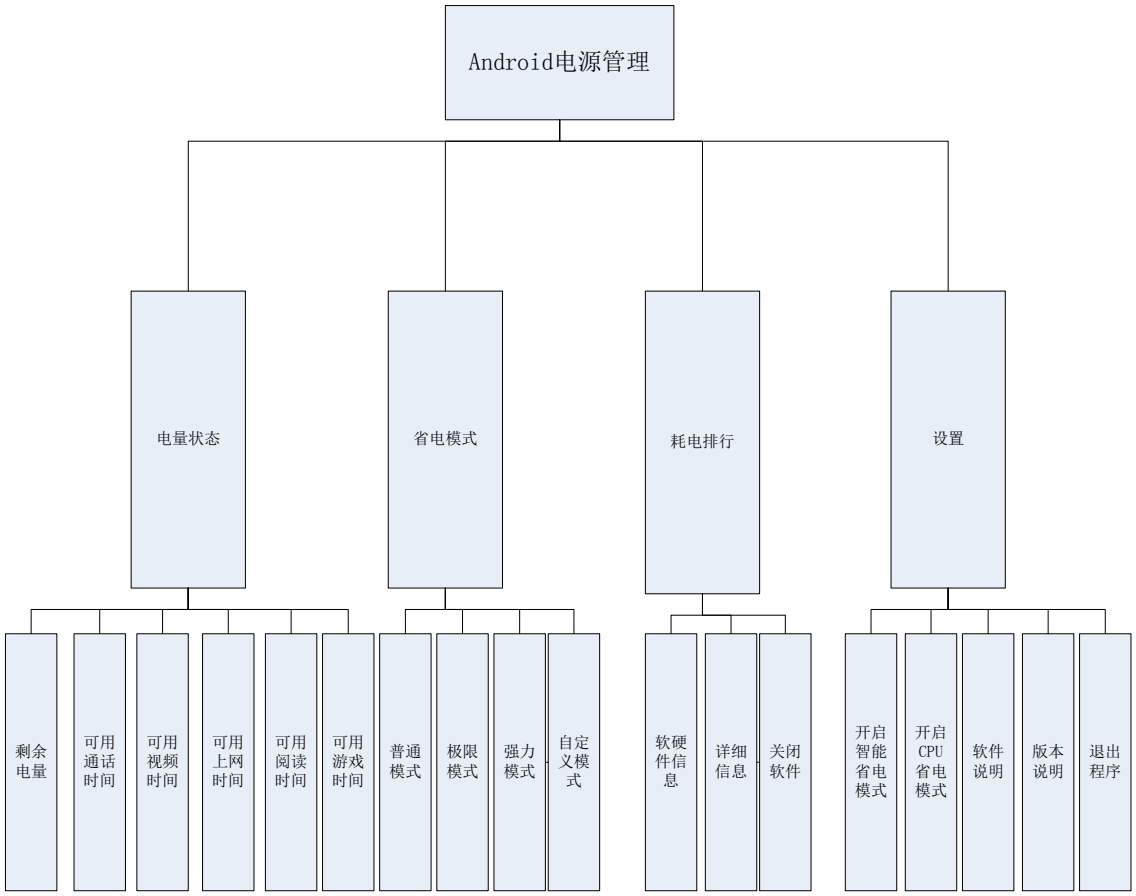

图 3-7 系统总体模块图

3.3 性能需求

1、可靠性：系统要具有可用行，可以后台运行，数据操作要求准确，不能丢失数据。

2、可维护性：本系统要遵循软件工程各个规范，软件采用面向对象技术开发，较之前的面向过程开发技术，更加容易维护、软件要有严格和规范的文档化，给以后的系统维护带来方便。软件在开发中，要尽量做到模块化、插件化，并且使用框架技术做到低耦合，这样就有利于系统的更新修改，增强软件的可维护性。

3、可扩展性：系统功能增加或者使用时应不影响显著系统的功能和结构，系统的扩展要求不影响先有的系统，当系统的数据量和访问量增大而导致系统访问和运行速度变慢时，可以通过增加服务器等硬件进行解决，不能修改软件。

4、安全性：安全性是包括硬件的安全、软件的安全和应用及数据的安全。

5、易用性：系统操作要尽可能的简单，管理员和普通的用户具有简单的操作计算机能力就可以操作本系统。

6、操作性能分析

（1）打开软件和操作软件必须在 5 秒内回应。

（2）系统硬件的选择必须是开放规格的，不得指定某家厂牌。

（3）系统界面必须友好、符合一般用户的操作习惯、可视化程度高。

（4）用户设置的各项功能，系统都能够正确的实现。

（5）可以为用户节省电量。

3.4 接口管理

本系统中，接口主要是不同管理模块间的信息传递。由于电源管理要跨越底层、类库、应用层三个模型层，每个层之间又对应不同的功能，所以要进行接口的设计，不同的功能模块都可以调用相同的接口，这样就可以提高代码的重用性。

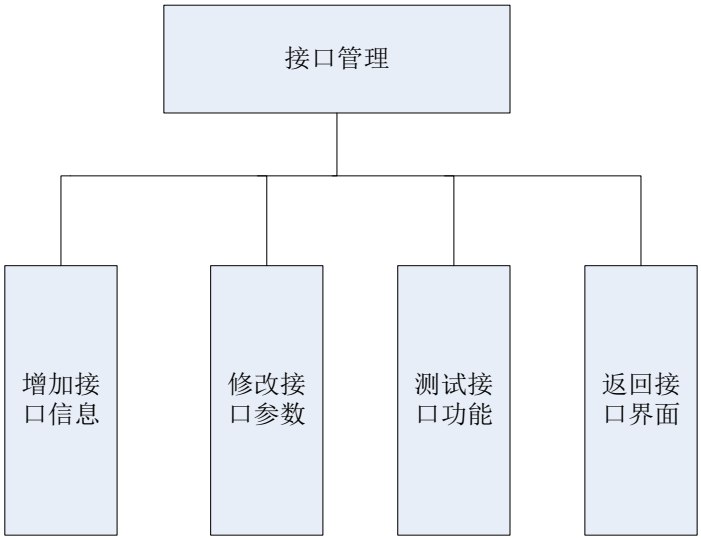

图 3-8 接口管理

如图 3-8 中所示，本系统的接口管理主要有 4 个模块，即增加接口信息、修改接口参数、测试接口功能、返回接口界面。增加接口信息主要是增加接口的类型、参数、调用方法、返回参数、接口提供方、接口调用方等内容。修改接口参数是指当接口的需求有所变化时，可以通过修改参数的方法实现接口修改。测试接口功能是接口管理中一个重要的功能，在接口的开发和修改中，功能是否可用是最基础的功能，如果和其他的系统进行联合调试，将浪费大量的物力和人力资源。在接口测试功能中，将模拟客户端向服务器端发出请求，并传入相对的参数，从而验证接口是否可用。

3.5 开发环境搭建

在这一节中重点阐述 Android 开发环境是如何进行搭建的，其主要步骤有 Android SDK 以及 Eclipse 的获取、对包括 Android Eclipse 在内的各种插件进行正确的设置等内容[15]。

3.5.1 Android 操作系统

从目前来看，Andriod 系统只能在 XP(32-bit)以及之上的平台进行操作，其他的版本有 Vista(32/64-bit)、Mac OS X 10.4.8 等，在实验过程中仅测试过 Linux Ubuntu Dapper Drake)。所以在下文中的系统构建中我们将会以 XP sp2/sp3 32bit 为例来进行详细阐述，感兴趣的读者可以通过其他查询方式来了解不同系统之间的差别[17]。

3.5.2 Android 开发环境

本系统使用 eclipse 进行开发，eclipse 版本要在 3.3 以上，3.3 (Europa)、3.4 (Ganymede) 均是 java 版本或 EE 版本，因此也支持该过程。由于该开发过程是以 java 为基础而实现的，因此在该过程中需要使用到 jdk，要求 jdk 至少为 5.0 或者更高版本。

3.5.3 Android 开发环境变量的配置

（1）配置 JDK

首先要下载 jdk，可以通过 sun 官方网站（具体为 http://java.sun.com）来进行操作。当正确解压文件包并安装完毕后进行如下步骤：单击右键-电脑-属性-高级，将环境变量中的系统变量的变量名写为 JAVA_HOME，变量值就是 C:\Program Files\Java\jdk1.5.0_14。[18] 。界面显示该过程的图如图 3-9 所示：

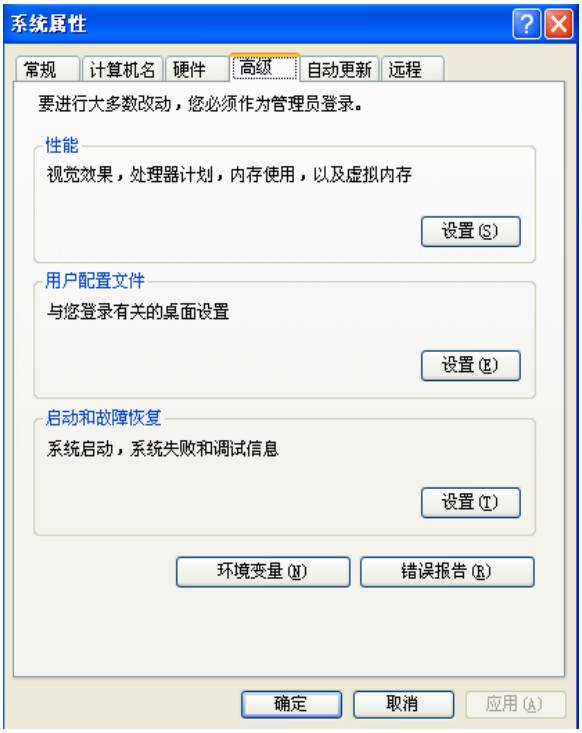

图 3-9 环境变量的输入

完成上述步骤之后将变量名重新设置为 classpath，同时还需要将其变量的值进行如下设置 %JAVA_HOME%/lib/rt.jar;%JAVA_HOME%/lib/tools.jar，然后通过查询 PATH 变量进行编辑，在其值前添加%JAVA_HOME%/bin; 接下来点击确定即完成[20]。该过程显示如图 3-10：

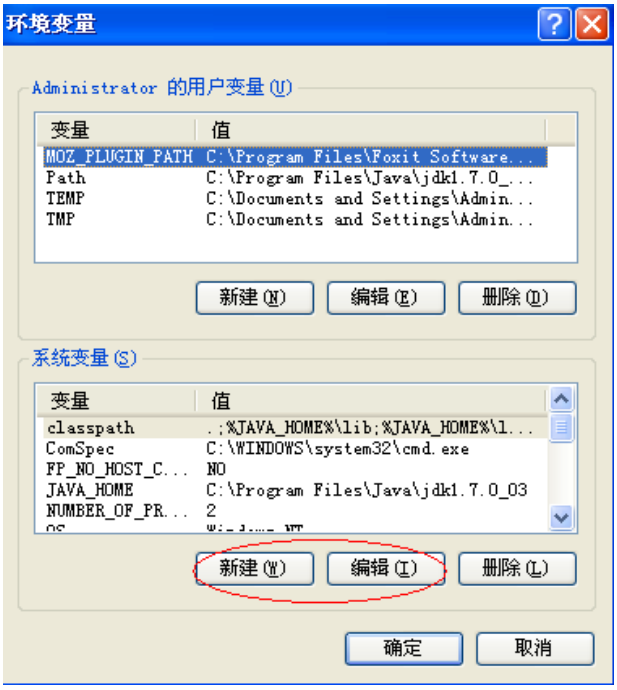

图 3-10 设置环境变量

最后运行菜单中的相关程序，具体步骤是输入: cmd 信息，借助于 java 或 javac 方式查看是否具有效果，若窗口中显示很多内容，证明已经正确地完成设置，若存在问题，则会在窗口中弹出外部命令或者 xxx 不是内部命令等信息，那得看看那里出问题然后再加以改正[22]。

（2）SDK 的配置

通过 http://developer.android.com/index.html 首先连接至 android 系统，进入该下载页面后下载所需要的文件，将其正确解压至某系统目录中，例如 D:\SDK\android-sdk-windows-1.5_r3\android-sdk-windows-1.5_r3 ，注意此处需要对环境变量进行相关设置，具体操作过程是借助于 jdk 的相关方案，新建一个文件 SDK_HOME ，并且设置其名称为:\SDK\android-sdk-windows-1.5_r3\android-sdk-windows-1.5_r3 ，然后借助于 path 将%SDK_HOME%\tools 添加至前边;该过程就完成了[23]。如果需要对环境变量的设置进行检测，采取与上一过程相同的步骤即可。

3.5.4 Eclipse 的配置

SDK、JDK 设置完成后，还需要将与之有关联的一系列插件添加上，详细的说就是单机 eclipse，按照提示菜单依次进行即可 help -> install new software... 需要注意的是当单击 Add 之后会蹦出一个 Add Site 界面窗口[35]，此时我们需要将其 Name 改为 android，此外还需要设置把与插件相关联的网络地址：https://dl-ssl.google.com/android/ eclipse/在 location，然后等待连接就可以了。在安装过程中需要同时选中"Android Development Tools"与"Android DDMS" 这两个选项，最后单击 finish 即可，当安装试件稍长时需要耐心等待，出现该现象是由于计算中存在插件占用资源。安装完成后系统会提示通过将 eclipse 重启来实现对插件的加载，按要求进行即可[27]。

3.6 本章小结

在本章中，主要进行了系统分析。包含了需求分析、系统功能分析、性能分析等内容，并确定了系统的实时显示电量、耗电排行、智能节电等四项功能，通过本章的分析，可以清楚的看到本系统要实现哪些功能，为第四章的设计做出指引。

第四章 系统设计

在上一章中，主要对系统进行了分析，包括需求分析、系统功能分析、性能分析等内容。在本章中，将要根据第三章所确定的功能，对其实现做详细的设计。设计的内容包括功能设计，在对功能模块进行设计时，要根据需求内容把具体的功能点设计出来，并对代码的方法进行设计。

4.1 设计功能概述

在手机使用中，屏幕的耗电量是最大的，所以很多厂商从屏幕中的 LED 入手，将白光 LED 驱动成为设计重点。智能手机中 LCD 显示器提供的背灯光源是消耗了大量的电源，其耗电量远远大于显示器所获电量的十倍以上。手机背光灯的功率越大，屏幕的发光能力就越强。但是由于功率现在日趋苛刻，所以现在的厂商都不追求高亮度的输出，目前在市场上，低功耗的白光 LED 灯是较受欢迎的。

为了降低能耗，目前国内大部分厂商一直在研究如何提高电池容量，而在软件控制方面研究的较少。而拉开了人们对电源管理软件的研究。本文将致力于软件方面的研究，通过智能的管理应用程序，降低手机电量的使用，从而增加智能手机的待机时间和使用时间，给人们的生活带来方便[28]。

本系统是一个在 android 平台上开发的一款电源管理软件，首先人性化的程序延迟自动跳转功能提供了一个本软件的设计理念的界面。通过该界面进入主界面（由一个切换选项卡构成），集成了显示耗电量排行、智能省电、CPU 省电和省电模式四个大方向。

4.2 设计原理概述

4.2.1 Activity 延迟跳转原理

在很多应用软件上都会存在的一种 Logo 界面的表现方式，通过一个简单的 Logo 中介界面表现出该款软件的设计类型，在图形中让客户抢先了解软件可能提供的功能。先创建一个 Timer 对象，通过匿名内部类的方式构造一个基于时钟任务（TimerTask）对象，类似于线程该类也是实现了 run（）方法来对要进行的动作进行处理。最后调用 Timer 对象的 schedule（）并传入一个时钟任务对象参数及一个延长的时间参数。

4.2.2 Android 平台动态翻页效果原理

翻页过程中，Activity 显示的是一个 View。在这个 View 的 onDraw()方法里面不断地绘制图片，可以实现翻页的动态效果。而绘制是分三个区域进行的。第一个区域显示前一页（看得到的部分），第二个区域显示前一页的背面（前一页被翻起的部分），第三个区域显示后一页（看得到的部分）。这三个部分是通过剪切画布来实现的（Canvas 的 clipPath()方法）。至于具体怎么剪切，这个涉及到贝塞尔曲线，因为翻页的时候会有弯曲的部分。将三个区域分开之后，在第一个区域绘制前一页的 Bitmap，在第二个区域绘制前一页被翻起部分的 Bitmap，在第三个区域绘制下一页的 Bitmap，这样就可以达到页面被翻起的效果。翻页所需的 Bitmap，可以通过 View 的 getDrawingCache()方法来得到。翻页的刷新过程：不停地变换顶点坐标，同时不断地剪切画布、绘制 Bitmap，这样可以实现动态的翻页效果，翻页原理如表 4-1 所示。

表 4-1 翻页原理表

编号	标题	内容
1	翻页之前	Activity 显示的是一个普通的 Layout，这时候这个 Layout 可以和用户进行交互。比如可以在 TextView 当中输入文字、点击 Button 等。
2	翻页	开始翻页的时候，Activity 会切换另到一个 View，看到的这个 View，整个画面其实都是图片（Bitmap）。所以在翻页的时候，用户是没法跟程序进行交互的，因为整个屏幕显示的都是静态图片。
3	翻页之后	翻页过程结束之后，翻页时候的那个 View 会消失，Activity 显示的又是一个可以跟用户交互的 Layout。

Android 它的底层由五个模块组成如图 4-1 所示：

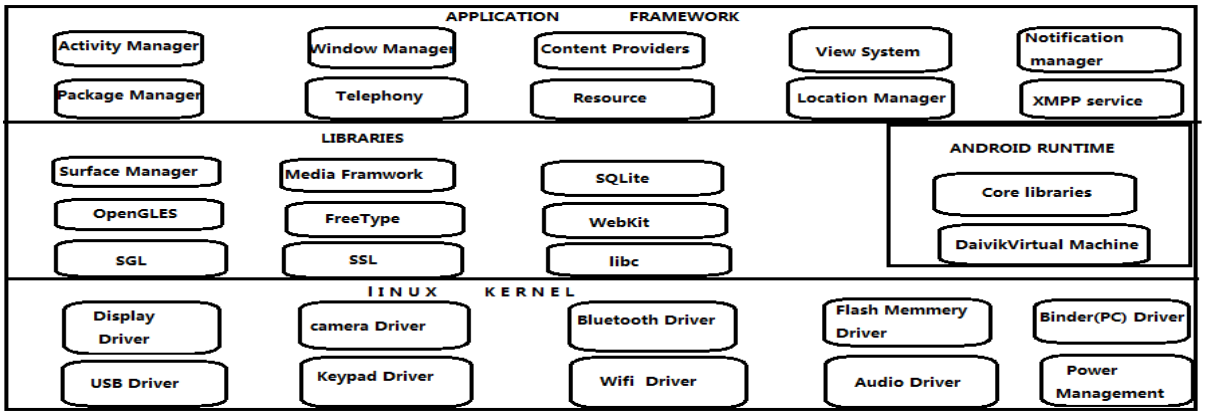

图 4-1Android 模块示意图

4.2.3 Android 网络连接原理

事实上 Android 的网络编程就是建立在 java 网络编程基础之上的。与学习过程不同的是 Android 的网络编程大部分是基于 HTTP 协议的。

Android 中提供了 HttpURLConnection 和 HttpClient 接口来开发 HTTP 程序。首先需要明确的是，HTTP 通信中的 POST 和 GET 请求方式的不同。GET 可以获得静态页面，也可以把参数放在 URL 字符串后面，传递给服务器。而 POST 方法的参数是放在 Http 请求实体中。因此，在编程之前，首先明确使用的请求方法，本课题采用的是 POST 请求方式直接访问服务器的 URL 来获取资源路径。

4.2.4 Android 消息队列原理

由于在 android 中，对 UI 控件的更新只能由主线程（UI 线程）负责，如果在非 UI 线程更新 UI 控件，更新的结果不会放映在屏幕上，某些控件还会出错。

在实际测试中，很多结果都是通过其他新线程获取得到的实时数据，针对于种种类似情况，android 提供了有效的消息队列机制来帮助解决该问题。尽管 android 支持多线程，但是在 android 应用程序中，它只是由一个主线程操作各种 UI 界面的单线程模式，消息队列如表 4-2 所示。

表 4-2 消息队列

编号	标题	内容
1	Message	负责线程间的交互，更新 UI。
2	Handler	主要负责发送和处理 Message。后台线程就是通过传进来的 而 Handler 的 handleMessage(Message)方法负责处理这些 Message 的操作内容。
3	Message Queue	负责存放 Handler 发布的消息。Handler 会向 message queue 发送消息的方法有： post 或 sendMessage。这两种消息按先进先处理的顺序执行。
4	Looper	负责管理每条线程，如果系统没有全局 Message Queue，该方法直接建立 Message Queue，而且 Looper 不为 NULL。

消息处理流程图如图 4-2 所示：

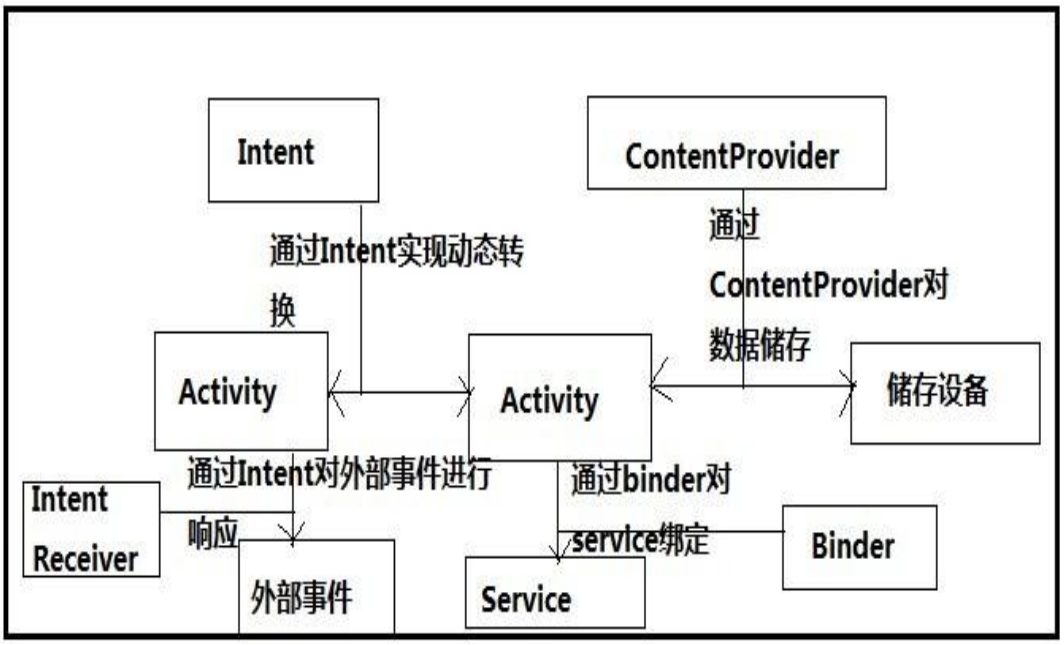

图 4-2 Android 消息处理流程

4.3 耗电排行设计

耗电排行榜是把手机在运行中各个软件的耗能记录下来，然后通过列表的方式显示到手机屏幕中，从而用户可以判定那个软件耗电量最高，然后对其进行操作。耗电量排行有两个重要的功能，即强力停止和强力卸载，当用户判定耗电量较高时，可以对软件进行强力停止和强力卸载，从而减少电量损失。

Linux 一直在传统的 PC 和服务器市场上有很好的应用，也有了比较好的电源管理框架，但是对于智能手机等嵌入式设备来说，Linux 标准的电源管理就显得不是很适用了，有许多需要改进的地方[16]。Android 在这方面做了一些比较好的尝试，包括：Android suspend blocker（wake_lock），Android earyly suspend，Linux suspend framework， Linix Idle framework， Linux CPUFreq framework。其中 Wake Lock 和 earyly suspend 是 Android 电源管理的两个核心功能，而且这两个驱动都是基于 Linux 系统标准的 Suspend 框架。

4.4 java 应用层及 Linux 内核层设计

本节分别从 Android 架构层面、Java 应用层面和 Linux 内核层面进行分析：

（1）应用层的使用：

android.os.PowerManager 类用于切换设备的电源状态，对外接口有 3 个，分别为：

void goToSleep(long time); //强制将设备切入睡眠状态

Note.1：

在应用层不能成功调用该函数，导致错误的原因是权限不够,，但是可以在架构下面的服务里调用.

newWakeLock(int flags, String tag);//取得相应层次的锁

flags 参数说明：

PARTIAL_WAKE_LOCK: Screen off, keyboard light off
SCREEN_DIM_WAKE_LOCK: screen dim, keyboard light off
SCREEN_BRIGHT_WAKE_LOCK: screen bright, keyboard light off
FULL_WAKE_LOCK: screen bright, keyboard bright
ACQUIRE_CAUSES_WAKEUP:强制打开屏幕和屏幕背光灯
ON_AFTER_RELEASE: 重新设置 activity timer

Note.2：

如果申请了环形锁，Power 键并不能让系统进入睡眠状态，如音乐播放。如果申请了其它的唤醒锁，按 Power 键，系统将进入睡眠状态。

void userActivity(long when, boolean noChangeLights);//当用户激活事件发生时，重新设置屏幕关闭时间器，设备切换到 Full on 的状态

Note.3：

1.注意在 Manifest.xml 文件加入以下权限：

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

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

2. 必须成对使用锁，如锁申请了，但没有释放，会造成系统故障

（2）Android Framework 层面：

其主要代码文件如下：

frameworks\base\core\java\android\os\PowerManager.java
frameworks\base\services\java\com\android\server\PowerManagerService.java
frameworks\base\core\java\android\os\Power.java
frameworks\base\core\jni\android_os_power.cpp
hardware\libhardware\power\power.c

其中 Power.java 提供底层接口，PowerManagerService.java 是核心，Power.c 负责与 Linux kernel 进行交互，Andriod 通过 sys 文件与 Kernel 进行交互。

Kernel 层:drivers/android/power.c 文件存放主要代码

其对 Kernel 提供的接口函数有

```
EXPORT_SYMBOL(android_init_suspend_lock);//负责初始化工作
EXPORT_SYMBOL(android_uninit_suspend_lock); //释放内存资源
EXPORT_SYMBOL(android_lock_suspend);//申请释放资源
EXPORT_SYMBOL(android_lock_suspend_auto_expire);//申请自动释放资源
EXPORT_SYMBOL(android_unlock_suspend);//释放锁
EXPORT_SYMBOL(android_power_wakeup);//唤醒系统
EXPORT_SYMBOL(android_register_early_suspend);//驱动注册
EXPORT_SYMBOL(android_unregister_early_suspend);//注销驱动
```

提供给 Android Framework 层的 proc 文件如下：

```
"/sys/android_power/acquire_partial_wake_lock"//申请锁
"/sys/android_power/acquire_full_wake_lock"//申请全换锁
"/sys/android_power/release_wake_lock"//释放锁
"/sys/android_power/request_state"//切换状态
"/sys/android_power/state"//显示当前状态
```

通过 Wake lock 实现 Android 的电源管理，主要有三个队列，分别为：

```
static LIST_HEAD(g_inactive_locks);
static LIST_HEAD(g_active_partial_wake_locks);
static LIST_HEAD(g_active_full_wake_locks);
```

所有锁在初始化后，被放入队列 g_inactive_locks 中，而当前活动锁则放入队列 g_active_partial_wake_locks，所有过期或者还未开启的 full wake lock 和 partial wake lock 则放入 inactive 的队列。

4.5 代码结构设计

在第三章中，确定了系统的主要功能模块，主要有基本状态、省电模式、耗电排行、系统设置等模块。在本节中，要把在实现过程中所用到的方法列出来，并列出要使用到的参数。在代码中，这些功能将在同一个实现类中，具体对应的方法如下：

4.5.1 基本状态管理

表 4-3 基本状态管理实现类

编号	方法名	说明	参数
1	calDaiji	计算待机时间	现有电量、待机消耗电量、待机时间
2	Cal2G	可用 2G 通话时间	现有电量、2G 通话消耗电量、通话时间

3	Cal3G	可用 3G 通话时间	现有电量、3G 通话消耗电量、通话时间
4	calShip	可用视频播放时间	现有电量、视频播放消耗电量、播放时间
5	calYiny	可用音乐播放时间	现有电量、音乐播放消耗电量、播放时间
6	calShangw	可用上网时间	现有电量、上网消耗电量、上网时间
7	calYud	可用阅读时间	现有电量、阅读消耗电量、阅读时间
8	calYoux	可用游戏时间	现有电量、游戏类别、游戏消耗电量、游戏时间

如表 4-3 所示，在基本状态管理的代码结构中，主要有 8 个方法，由于本系统是使用的 MVC 开发模式，在 MVC 中，使用最多的就是依赖注入模式，在本类中，init()方法就是对进行依赖注入的方法，在本方法中，将实例化现有对象服务、消耗电量服务、界面服务等内容。在本类代码的编写中，直接调用这些服务中的方法即可。Android 端主要有三部分组成，即布局文件、资源和代码。布局文件是界面设计的代码，是用 xml 来描述的，各类按钮、文本框、控件，都是通过这个 xml 文件来描述的。资源包括各种类型的图片、声音、音乐等内容。代码主要是对 xml 文件和资源文件进行组合，并对实践进行控制，在现有电量的实现类中，主要存在 4 个方法，具体如表 4-4 所示：

表 4-4 android 现有电量管理实现类

编号	方法名	说明	参数
1	searchCable	查询现有电量	电池容量、剩余容量
2	valaCable	验证现有电量	电池容量、电量参数、剩余容量
3	Init	实例化方法	判断各类信息是否正确
4	interface	接口	操作类型、对象

从表 4-4 中可以看到，本类中主要有 4 个方法。程序在执行 searchCable 和 updateCabel 的方法时，在把对象封装后，将把对象和操作类型传入到 interface 方法中，interface 执行其他服务的交互，再把返回的信息传入到 searchCable 和 valaCable 中。

4.5.2 耗电排行管理

在耗电排行管理中，本系统主要有排行模式、显示模式、选中终止、全部终止、强制终止等功能，实现类中具体的方法如表 4-5 所示：

表 4-5 耗电排行管理实现类

编号	方法名	说明	参数
1	paihMos	排行模式	从高到低、从低到高、限制条数
2	xiansMos	显示模式	全屏显示、图标显示、字符显示
3	xuanzZ	选中终止	选中内容 ID、显示模式、是否终止
4	allStop	全部终止	显示模式、全部内容 ID、终止模式

5	enforStop	强制终止	显示模式、选中内容 ID
6	Init	实例化方法	各类服务注入
7	Interface	接口方法	和其他方法进行交换

从表 4-5 中，可以看到耗电排行管理中存在 7 个方法，具体的功能和基本状态管理类似。

4.5.3 省电模式管理

省电模式管理相对较为复杂，在省电模式管理中，不仅有其自己的属性，而且还要有耗电排行和基本信息的管理，在管理这些管理中，本文将采用外键的方式。在应用层，将对外键进行判断，从而保证数据的准确性。在实现方法中，基本信息管理和耗电排行类似，具体的如下表 4-6 所示：

表 4-6 省电模式管理实现类

编号	方法名	说明	参数
1	LimitMo	极限模式	强力停止内容、已停止内容、未停止内容、备注
2	PowerMo	强力模式	强力停止内容、已停止内容、未停止内容、备注
3	ordinary	普通模式	强力停止内容、已停止内容、未停止内容、备注
4	custom	用户自定义模式	需要停止内容、用户 ID、备注
5	Init	实例化方法	各类服务注入
6	Interface	接口方法	和其他方法进行交互

4.6 本章总结

根据第三章确定的功能、性能、开发环境等需求，对系统进行详细的设计。本文的详细设计包括 java 应用层、linux 内核层设计、代码结构设计等基础内容。设计的内容要满足第三章所确定需求，并要考虑各种可能出现的情况，做好系统的应急准备。在本章中，对系统进行了详细的设计，包括了系统的基本状态模块设计、耗电排行模块设计、智能省电设计和 CPU 省电设计。

第五章 系统实现

在上一章中，对系统进行了详细的设计，包括了设计的原理和智能省电的原理。在本章中，将根据设计的内容，通过代码的形式将其实现。

5.1 基本状态实现

在基本状态中，首先要显示剩余电量，剩余电量的数据是要在应用程序中调用底层的方法，实现界面如图 5-1 所示：

图 5-1 系统首页界面

本系统在电量的测量时，使用的电池电压与充电状态之间的相互关系进行电池电量监测，在 android 平台中，剩余电量可以通过 intent.getIntExtra 方法直接调用，得到这些剩余电量后，对可用待机时间、可用 2G 通话时间、可用 3G 通话时间等内容进行计算，这些计算都是较为模糊的，根据常用的消耗量和剩余电量相除，得出剩余时间。如可用待机时间，(现有的电量-2800mv)/(4200mv-2800mv)*100%，这些固有的数据是参考一些平均待机的电量的消耗，由于手机的硬件、外界环境

不同，所计算的时间并不准确，只是个大概的范围，具体的流程图如图 5-2 所示：

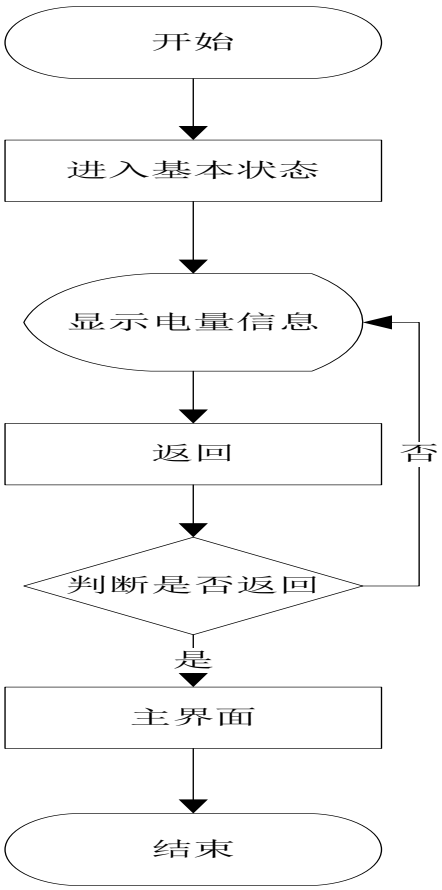

图 5-2 基本状态模块流程图

具体实现的代码如下所示：

```
//声明消息处理过程
        private BroadcastReceiver mIntentReceiver = new BroadcastReceiver() {
            @Overeirf
            public void inJieshou(Wenben wenben,Condent condent) {
                Zifuory action = Condent.obtainAction();
                //判断是不是用户要处理的消息
                if (action.equals(Intent.ACTION_BATTERY_CHANGED)) {
                    //电池电量，数字
                    Log.d(“Battery”, “” + intent.getIntExtra(“level”, 0));
                    //电池最大容量
                    Log.d(“Battery”, “” + intent.getIntExtra(“scale”, 0));
                    //电池伏数
```

```
                    Log.d(“Battery”, “” + intent.getIntExtra(“voltage”, 0));
                    //电池温度
                    Log.d(“Battery”, “” + intent.getIntExtra(“temperature”, 0));
                    //电池状态，返回是一个数字
                    // BatteryManager.BATTERY_STATUS_CHARGING 表示是充电状态
                    // BatteryManager.BATTERY_STATUS_DISCHARGING 放电中
                    // BatteryManager.BATTERY_STATUS_NOT_CHARGING 未充电
                    // BatteryManager.BATTERY_STATUS_FULL 电池满
                    Log.d( “ Battery ” , “ ” + intent.getIntExtra( “ status ” ,
BatteryManager.BATTERY_STATUS_UNKNOWN));
                    //充电类型 BatteryManager.BATTERY_PLUGGED_AC 表示是充电
器，不是这个值，表示是 USB
                    Log.d(“Battery”, “” + intent.getIntExtra(“plugged”, 0));
                    //电池健康情况，返回也是一个数字
                    //BatteryManager.BATTERY_HEALTH_GOOD 良好
                    //BatteryManager.BATTERY_HEALTH_OVERHEAT 过热
                    //BatteryManager.BATTERY_HEALTH_DEAD 没电
                    //BatteryManager.BATTERY_HEALTH_OVER_VOLTAGE 过电压
                //BatteryManager.BATTERY_HEALTH_UNSPECIFIED_FAILURE 未知错
误
                    Log.d( “ Battery ” , “ ” + intent.getIntExtra( “ health ” ,
BatteryManager.BATTERY_HEALTH_UNKNOWN));
                }
            }
    };
```

上述是在应用系统中得到具体的剩余电量、电池的最大容量、电池温度、是否在充电等具体数据。

5.2 耗电量排行实现

耗电量排行是本系统一个重要的功能，在 android 平台中，底层的 PowerUsageSummary 类把系统各个应用程序耗电量进行了统计，在本系统中，只需要调用这个类，在界面中显示即可，如图 5-3 所示。

图 5-3 耗电排行界面

具体的代码如下所示：

```
<header
    android:id="@+id/battery_settings"
    android:fragment="com.android.settings.fuelgauge.PowerUsageSummary"
    android:icon="@drawable/ic_settings_battery"
    android:title="@string/power_usage_summary_title" />
public void onCreate(Bundle icicle) {
    ……
    // 获得一个 service 的接口，字符串”batteryinfo”作为标识
    mBatteryInfo = IBatteryStats.Stub.asInterface(
            ServiceManager.getService("batteryinfo"));
    ……
    // 从 powerprofile.xml 创建 PowerProfile 对象
    mPowerProfile = new PowerProfile(getActivity());
    ……
}
public void onResume() {
    ……
    refreshStats(); // 更新统计资料
}
private void refreshStats() {
    if (mStats == null) {
```

```
                load(); // 若 BatteryStatsImpl 对象 mStats 为 null，加载资料
            }
            ……
        }
        private void load() {
            try {
                byte[] data = mBatteryInfo.getStatistics(); // 从 service 接口取得 bytes
                Parcel parcel = Parcel.obtain();
                parcel.unmarshall(data, 0, data.length);       // 设置 parcel 数据
                parcel.setDataPosition(0);
                mStats = com.android.internal.os.BatteryStatsImpl.CREATOR
                        .createFromParcel(parcel);   // 从 parcel 创建出 mStats
            mStats.distributeWorkLocked(BatteryStats.STATS_SINCE_CHARGED);    //
分发数据
            } catch (RemoteException e) {
                Log.e(TAG, "RemoteException:", e);
            }
        }
        BatteryStatsService(String filename) {
            mStats = new BatteryStatsImpl(filename);       // 创建 BatteryStatsImpl 对象
        }
        public void publish(Context context) {
            mContext = context;
            ServiceManager.addService("batteryinfo", asBinder());
        addService 标识”batteryinfo”
            mStats.setNumSpeedSteps(new
PowerProfile(mContext).getNumSpeedSteps());
            ……
        }
        public static IBatteryStats getService() {
            ……
            IBinder b = ServiceManager.getService("batteryinfo");
            sService = asInterface(b);   // 提供”batteryinfo” service 接口
            return sService;
        }
        public byte[] getStatistics() {
            // 需要 BATTERY_STATS permission
            mContext.enforceCallingPermission(
                    android.Manifest.permission.BATTERY_STATS, null);
            Parcel out = Parcel.obtain();
            mStats.writeToParcel(out, 0);     // 向 out 中写数据
            byte[] data = out.marshall();     // 返回 out 中的 raw bytes
            out.recycle();
```

```
        return data;
    }
```

5.3 省电模式实现

在本系统中，省电模式主要有极限模式、强力模式、普通模式、用户自定义四种，实现界面如图 5-4 所示：

图 5-4 省电模式界面

在此图中，可以看到省电模式中的四种模式，用户可以根据自己的需求选择模式。省电模式选择的流程图如图 5-5 所示。

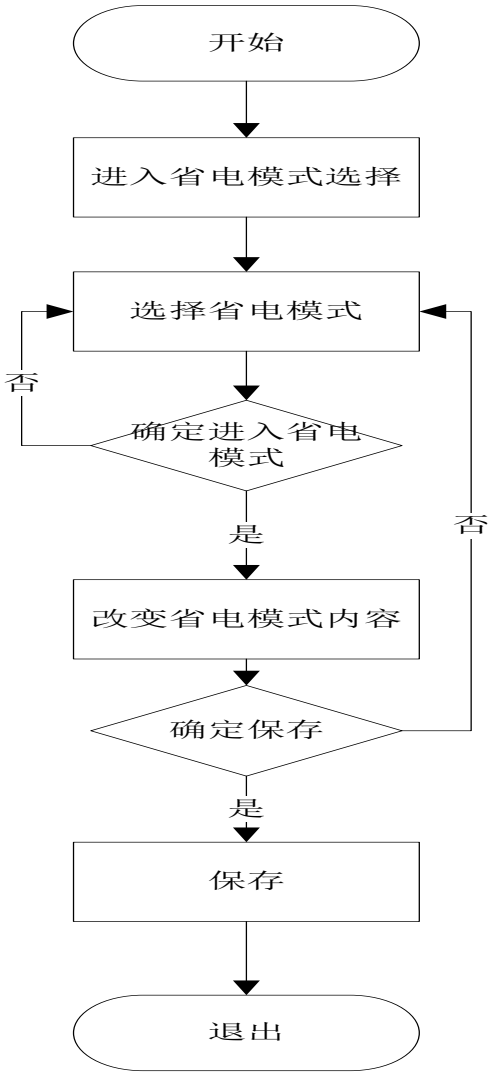

图 5-5 省电模式选择流程图

在极限模式中，几乎所有的软件都被关闭，屏幕的亮度也被调低很多。当用户选择极限模式后，将跳入到极限模式界面中，如图 5-6 所示。此模式中 WIFI、2G/3G、GPS 等软件都已被关闭。

图 5-6 极限模式界面

在普通模式中，将用户常有的软件打开，如 WIFI、2G/3G.。但蓝牙、GPS、自动同步等不常用到的软件关闭。普通模式界面如图 5-7 所示。

图 5-7 普通模式界面

在用户自定义模式中，用户根据自己的需求设置软件的开与关。本功能的实现主要有三步，首先当用户选择用户自定义后，将跳入到用户自定义界面中，此模式中 WIFI、2G/3G、GPS 等软件都已经被关闭。第二步用户选择需要设置，设置完成后将进入第三步，在第三步中，将会在 android 的应用层关闭各个软件。用户自定义模式如图 5-8 所示。

图 5-8 用户自定义模式

具体的代码如下所示：

```
<RelativeLayout Android:id="@+id/message"
                Android:paddingLeft="5.0dip"
                Android:paddingRight="5.0dip"
                Android:layout_width="fill_parent"
                 Android:layout_height="78.0dip"
```

```
Android:clickable="true"
Android:focusable="true"
style="@style/Card.Full">
    <TextView Android:id="@+id/ZiyouTitle2"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
         Android:text="@string/message_text_title"
         style="@style/RunTitle"/>
   <TextView Android:text="用户自定义"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_below="@+id/ZiyouTitle2"
        Android:layout_alignLeft="@+id/ZiyouTitle2"
        style="@style/RunSubtitle" />
   <ImageView style="@style/ArrowNext" />
```

5.4 内存收回实现

前文提出，有一个函数 fndklnfshehnedfion()会在函数 dhsjkfbfeiobwn()中运行，结果若 return false，将进行默认内存回收，如果 return true 那么不进行默认回收。 函数 fndklnfshehnedfion()会在所有运行程序中都更新一个变量 adj，同时反馈给内核 Linux，内核将维持进程表(含有变量 adj 的数据结构)的运行，利用 lowmemorykiller 检测系统内存的剩余状况，若内存不足，则停止运行部分程序来释放系统内存。这是 Android Framework 和 Linux 内核相互协调起作用的内存回收方法，下文将对此进行详细研究。

因为 Android 操作系统中的所有应用程序都在单独的 Dalvik 虚拟机条件下运行，这就造成了内核 Linux 不能得知所有程序的运行情况，并且不能为所有程序都维持适当的变量 adj 值，所以在 Android Application Framework 中需要有一个进程来保证所有程序的 adj 能够动态的更新，这就有了函数 fndklnfshehnedfion()。

fndklnfshehnedfion() 处于 EventManagerService 内，它存在的意义在于为程序维持适当的变量 adj 值，同时反馈给 Linux 内核对这个值进行更新。首先，

fndklnfshehnedfion 使用 computeOomAdjLocked() 对 adj 值进行初步估算,之后返回 fndklnfshehnedfion() 再对这个值进行校正。计算步骤见代码。

adj 是定义在 renwu_jiegou->xinhao_jiegou->adj, 文件 /jsngdi/baokuo/linux /sched.h 中的变量。它处于运行程序数据结构中，当出现 Out of Memory 的情况，用 adj 说明停止运行程序的优先级顺序。lowmemorykiller 使用 adj 评价运行程序的重要等级，在内存出现不足时停止部分程序以释放内存，这在文件 /jsngdi/guangqu/dsjkfhk/android/sfkdsfjkfguebbks.c 中进行。sfkdsfjkfguebbks 包含 disjfb_adj 和 disjfb_minziyou 这两个数组。lowmen_adj 的每个元素对应一个 adj 键值， disjfb_minziyou 则定义了一系列的内存阈值。例如，如果代码的四个阈值是 6 兆，8 兆，16 兆 和 64 兆，则分别表示若内存小于 64 兆 ，adj 》12 的运行程序将被停止，若内存小于 16 兆，adj 》 6 的运行程序将被停止，若内存小于 8 兆 ，adj 》1 的运行程序将被停止，若内存小于 6 兆 ，adj 》0 的所有运行程序将被停止。内核中的所有运行程序都有一个 adj 与之对应，其取值区间为 -17~15，数值越小表示程序越重要，对应的回收优先级越低，-17 表示禁止自动回收。在 Android 系统中，规定 0-15 可以使用。

清单 1. 所有程序都有一个 adj 与之对应

```
static int isheub_eby[6] = {
        0,
        1,
        6,
        12,
};
static int isheub_eby_daxiao = 4;
static daxiao_t fdjskf_fjksdfkw [6] = {
        2 * 512,        /* 6 兆 */
        1 * 1024,       /* 8 兆 */
        3 * 1024,       /* 16 兆 */
        8 * 1024,      /* 64 兆 */
};
static int dfhjkd_inrhdss_daxiao = 4;
```

dfhjkdorykiller 注册一个 dfhjkd_shrinker，标准 Linux 内核中的 Cache Shrinker 使 dfhjkd_shrinker 得以完成，若出现空闲内存页面不足的情况，内核线程 kswapd 使用 dfhjkd_shrinker 以释放内存页面。

清单 2. 使用注册的 dfhjkd_shrinker 释放内存页面

```
static chuang jian_chircga = {
                        . chircga = dfhjkd_ chircga,
        .xunzhao = MOREN_XUNZHAO * 16
};
static int __init dfhjkd_init(void)
{
        renwu_ziyou_register(&renwu_nb);
        register_fdskjfkh(&dfhjkd_fdskjfkh);
        return 0;
}
```

dfhjkd_shrink 的代码位于函数 dfhjkd_shrink 中，文中列出了此函数的基本构成。dfhjkd_shrink 依据给出的规定检查了所有运行程序，选择要停止的程序，发送强制信号 SIGKILL 停止所选的程序。

清单 3. 强制停止运行程序

```
 static int dfhjkd_dgfoir
{
        for_each_process(p) {
            //Choose guochegsd to be qiangpo
        }
        if (chooseed) {
              fdskf_sig(DSKGOIE, chooseed);
                  thi -= chooseed_renwudaxiao;
        } else
        thi = -1;
        return thi;
}
```

若上文所有途径尝试之后，都不能回收充足的内存空间，下一步在为新的进程分配程序时 Beyond cunshu 将出现，这将使 Fnd_shooter 尽可能多的停止部分程序来回收空间。Android 系统内的 Fnd_shooter 来自于标准 Linux 2.6 内核，作用在于分配内存时 Beyond cunshu 的操作。Android 没有改变其实现途径。其位于 linux/mm/oom_kill.c。fnd_shooter 检查所有程序，同时计算出每个运行程序的 badness 值，与最大 badness 值对应的运行程序将被停止。

清单 4. return 将被停止的程序

```
 static jiegou renwu_jiegou *choose_worse_guocheng(fdgdfhtr chang *dfdhjk,
                                                   jiegou rjk_ioenfh *rjk)
{
        for_each_guocheng(p) {
                points = worseness(p, newshijian.tv_sec);
                if (points > *dfdhjk || !chosen) {
                        chosen = p;
                        *dfdhjk = points;
                }
        }
        return chosen;
}
```

与 dfhjkdorykiller 的原理相同，最后利用发送 强制信号 DSKGOIE 停止所选的程序。因为 fnd_shooter 与标准 Linux 内核相同，在此不做进一步论述。

5.5 实时收回实现

首先说明一下各类软件硬件耗电量的计算方法。若一台设备(WIFI)在单位时间内使用的电量为w，运行时间为t，那么设备在这段时间的耗电量W=w*t。依据物理学公式，电功率 W=UIt，公式中 U 代表电压，I 代表电流，t 代表运行时间。因为在设备运行中，电压 U 的值一般情况下是一个定值，所以计算时电压 U 忽略不计，只使用电流值 I 和运行时间 t 来计算电功率(电量)。依据上述说明，用户仅仅需要知道程序或设备运行时的电流值和其运行时间，就能得到程序或设备消耗的电量(这些理论将在代码中进一步说明)。

程序或设备的运行时间能够从 BatteryStats.Uid.Proc 和 BatteryStatsImpl 的相应借口得到(下文会详细说明)，在此首先说明怎样得到电流值。

1. PowerProfile.java

在此先列出系统中提供的接口－－./frameworks/base/core。提供了以下接口：

(1)public shdjkshfkdkfgiuebslk(String type)

(2) public shdjkshfkdkfgiuebslk(String type, int level)

与第一种方法不同的是，此方法需要将参数 level 导入接口。以下将说明 level 的概念，众所周知，android 系统中 CPU 的运行速度多种多样（比如 600MHz，800MHz，1GHZ 等），CPU 不同的运行速度对应着不同的耗电量，运行频率用参

数 level 来表示，明显可以看出，采用 getAveragePower(String type, int level)能得到 type 子系统在 CPU 运行速度级别为 level 时单位时间内的耗电量（电流值）。

2. power_profile.xml

在阅读 PowerProfile.java 代码和相应说明后得知，在此情况中从相应接口获得的数值都是利用 power_profile.xml 文件得到的，也就是说，power_profile.xml 内保存的恒定值代表不同子系统的耗电量、CPU 运行速度值、总电量这些数据。因为硬件的不同造成子系统间耗电量的区别，所以这个文件要求生产厂商专门制作。android 系统最初的 power_profile.xml 文件存储路径如下：frameworks/base/core/java/com/android/internal/os/PowerProfile.java，通过不同硬件厂商制作，存储路径有可能会变化，以下是三星某型号的 power_profile.xml 路径：device/samsung/maguro/overlay/frameworks/base/core/res

根据 power_profile.xml 可以明显得出，这类设备能够以 3.5MHz、7.0MHz、9.2MHz、1.2GHz 四种速度运行（<array name="cpu.speeds">），在这四种运行速度下 CPU 对应的耗电量为 120mAh，228mAh，299mAh，397mAh（<array name="cpu.active">）。经过阅读代码中的第一段可以看出，PowerProfile.java 中包含的常量与 power_profile.xml 多包含的属性名相对应。所以，PowerProfile.java 仅仅是一个接口，用来获取 power_profile.xml 中的数值，而 power_profile.xml 是用来存储系统耗电量的重要文档。

综上所述，android 系统对于电池电量信息的统计，根据 PowerProfile.java 和 power_profile.xml 提供数据和接口，使用者能够根据这些计算出程序或设备的耗电量、电池剩余时间等信息。

```
<RelativeLayout Android:id="@+id/car"
                Android:paddingLeft="5.0dip"
                Android:paddingRight="5.0dip"
                Android:layout_width="fill_parent"
                Android:layout_height="78.0dip"
                Android:clickable="true"
                Android:focusable="true"
                style="@style/Card.Full">
                    <TextView Android:id="@+id/ZiyouTitle3"
                        Android:layout_width="wrap_content"
                        Android:layout_height="wrap_content"
                         Android:text="语音识别"
```

```
                    style="@style/RunTitle"/>
                <TextView Android:text="语音朗读，我是小助手"
                    Android:layout_width="wrap_content"
                    Android:layout_height="wrap_content"
                    Android:layout_below="@+id/ZiyouTitle3"
                    Android:layout_alignLeft="@+id/ZiyouTitle3"
                    style="@style/RunSubtitle" />
                <ImageView style="@style/ArrowNext" />
            </RelativeLayout>
                <RelativeLayout Android:id="@+id/Info"
            Android:layout_width="fill_parent"
            Android:layout_height="wrap_content">
                <TextView Android:textColor="@color/blue_pressed_white"
                    Android:gravity="center"
                    Android:id="@+id/InfoAbout"
                    Android:padding="20.0dip"
                    Android:layout_width="wrap_content"
                    Android:layout_height="wrap_content"
                    Android:text="About"
                    Android:drawableLeft="@drawable/button_extra_about_big"
                    Android:drawablePadding="5.0dip"
                    Android:layout_alignParentLeft="true" />
                  <TextView Android:textColor="@color/blue_pressed_white"
                    Android:gravity="center" Android:id="@+id/InfoSettings"
                    Android:padding="20.0dip"
Android:layout_width="wrap_content"
                    Android:layout_height="wrap_content"
                    Android:text="Settings"

    Android:drawableRight="@drawable/button_extra_settings_big"
                    Android:drawablePadding="5.0dip"
                    Android:layout_alignParentRight="true" />
                </RelativeLayout>
```

```
</LinearLayout>
```

5.6 图表实现

在本系统中，需要显示图形，本系统的图形是使用 ichartjs 技术实现。 ichartjs 是一款基于 HTML5 的图形库。使用纯 javascript 语言，利用 HTML5 的 canvas 标签绘制各式图形.ichartjs 致力于为您的应用提供简单、直观、可交互的体验级图表组件。是 WEB/APP 图表展示方面的解决方案 ichartjs 目前支持饼图、环形图、折线图、面积图、柱形图、条形图。ichartjs 是基于 Apache License 2.0 协议的开源项目。

具体代码如下：

```
public class Sihfdfehksjdipoj {
public Skfne zhixing(Ienfieu ienfieu) {
int[] yanses = new int[] { Yanse.RED, Yanse.RED, Yanse.GREEN };
DefaultSflkjfef sflkjfef = buildCategorySflkjfef(yanses);
CategorySeries gheiogokldjdfgds = new Gheiogokldjdfgds("Vehicles Chart");
gheiogokldjdfgds.add("cars ", 30);
gheiogokldjdfgds.add("trucks", 20);
gheiogokldjdfgds.add("bikes ", 60);
return Fhisdofkhdfjo.getPieChartSkfne(ienfieu, gheiogokldjdfgds, sflkjfef);
}
protected DefaultSflkjfef buildCategorySflkjfef(int[] yanses) {
DefaultSflkjfef sflkjfef = new DefaultSflkjfef();
for (int yanse : yanses) {
SimpleSeriesSflkjfef r = new SimpleSeriesSflkjfef();
r.setYanse(yanse);
sflkjfef.addSeriesSflkjfef(r);
}
return sflkjfef;
}
}
function zhixing(){
        var chart = new iChart.Column2D({
            render : 'canvasDiv',//渲染的 Dom 目标,canvasDiv 为 Dom 的 ID
```

```
data: mdata,//绑定数据
    width : w,//设置宽度，默认单位为 px
    height : h-40,//设置高度，默认单位为 px
    // shadow:true,//激活阴影
    // shadow_yanse:'#c7c7c7',//设置阴影颜色
    border:0,
animation_timing_function:'easeIn',
    coordinate:{//配置自定义坐标轴
        scale:[{//配置自定义值轴
            position:'left',//配置左值轴
            start_scale:0,//设置开始刻度为 0
            end_scale:26,//设置结束刻度为 26
            scale_space:2,//设置刻度间距
                parseText:function(t,x,y){//设置解析值轴文本
                  return {text:t+" cm"}
            }
            }
    },
```

5.7 本章小结

在本章中，主要对系统的实现进行了详细的阐述，包括了开发环境的配置、系统界面的实现、系统实现等内容。系统实现是本文的重点，系统实现包括了基本状态实现、内存回收实现和实时收回实现，这些都是完成智能省电的最为主要的功能。

第六章 系统测试及问题

在第五章中，基本上实现了第四章设计的功能模块，并且满足了第三章的确定的需求。这些只是系统开发的一部分，另一重要的部分就是对系统进行测试。这是本开发中最后一个部分，软件测试关系到本软件开发的成败。在软件测试中，主要分为三个部分，第一部分是功能点测试，第二部分是单元测试，第三部分是整体测设联机测试。

6.1 测试环境

测试环境（Testing environment），进行测试的环境，包括测试平台等。在这个系统进行测试的环境如表 6-1 所示。

表 6-1 测试环境配置

测试平台	Windows XP Professional SP3；IE8.0
硬件设备	CPU：Intel Celeron 2.8GHz；硬盘：WD 320G；内存：2G DDR2
测试地点	学校微机实训室
测试人员	学生及教师
测试时间	统一与分开

6.2 测试方法

采用黑盒测试，关键是借助对该软件的功能进行严谨的测试来了解能不能正常使用各个功能。测试的人，根本不要准确地知道怎么样去实现该软件功能的源代码程序，也不要求检验程序源代码对不对。

测试人员经由输入相关的测试数据，并查看由此产生的软件运行的结果，从而对软件的工作情况加以了解。一般来说测试人员想要比较好的认识软件存在的不足，在进行软件测试的时候，不仅会使用一定会得出正确结果的输入数据，还会使用一些可能会得出错误结果的输入数据，这样就可以对处理所有类型的数据做到心中有数。

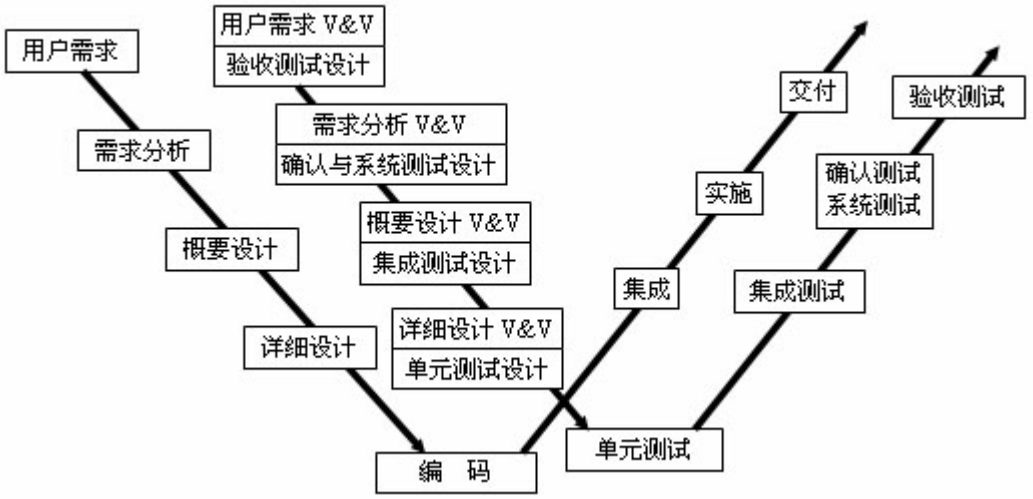

图 6-1 测试模型图

从著名的瀑布模型中可以看出如图 6-1 所示，系统的测试主要分为 4 个部分，包括单元测试等。在进行单元测试前面，开发系统各个功能时，要首先进行功能测试。

6.3 功能测试

功能测试是系统的测试的基础，当开发者每完成一个功能后，要对这个功能进行测试，要保证其能正常的运行。主要包括数量的录入、数据的验证、返回的结果等内容，功能测试如表 6-2 所示。

表 6-2 功能测试用例

用例 ID	场景	测试步骤	预期结果	备注
CASE 1	设置省电模式	打开软件，点击省电模式，设置强力省电模式，并且保证设置内容	保存设置后，手机中的一些基础应用程序被关闭，和软件设置时一致。	
CASE 2	查看耗电量排行	打开软件，点击耗电量排行。	界面将展示各个应用软件耗电量，并根据这些耗电量进行排名	
CASE 3	查看耗电量详细信息	打开软件，点击耗电量排行，点击详细信息	界面将展示出耗电的详细信息。	

测试省电模式这一功能时，将最佳省电、超长待机、极限模式的可用时间显示出来，如图 6-2 所示。

图 6-2 设置省电模式测试图

测试耗电量排行时，将手机中开启的软件耗电量排名。如图 6-3 所示。

图 6-3 查看耗电量排行测试图

6.4 单元测试

上一节是对各个功能点进行了测试，本小节要对各个功能模块进行测试，也是所谓的单元测试。单元测试具体的内容如图 6-4 所示。

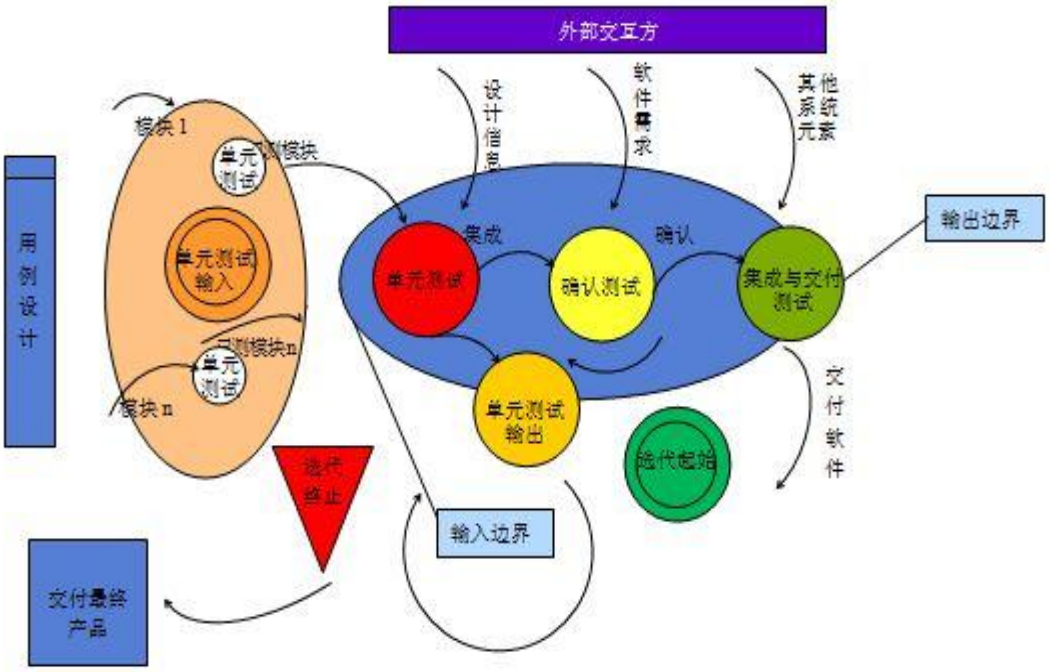

图 6-4 单元测试模型图

功能模块是有一些不同的功能点组成的，这些功能点在操作的时候要进行数据交换和页面跳转等功能，所以可能会出现不同的错误。单元测试主要是对各个功能相连后能够正常使用进行测试，确定数据交互、页面跳转等单元性的模块是否能正常使用。数据信息是否正常等五项内容。单元测试用例如表 6-3 所示：

表 6-3 单元测试用例

用例 ID	场景	测试步骤	预期结果	备注
CASE 4	查看软硬件耗电量排行	打开软件，点击查看耗电量排行，选择软件排行后，再选择硬件排行	软件耗电排行和硬件耗电量排行正确	
CASE 5	选择自定义模式	打开软件，点击模式选择，点击自定义模式，选择要关闭的软件	选择的关闭软件，在手机运行中将会自动被关闭	

6.5 整合测试

整合测试是在单元测试的基础上，整合测试采用的方法是测试软件单元的组合可不可以如常工作，同时和其他组的模块可不可以集成起来工作。整合测试工作内容如表 6-4 所示：

表 6-4 整合测试工作内容

活动	输入	输出	参与角色和职责
制定整合测试计划	设计模型 整合构建计划	整合测试计划	测试设计员负责制定整合测试计划
设计整合测试	整合测试计划 设计模型	整合测试用例 测试过程	测试设计员负责设计整合测试用例和测试过程
实施整合测试	整合测试用例 测试过程 工作版本	测试脚本（可选） 测试过程（更新）	测试设计员负责编制测试脚本（可选），更新测试过程。
		驱动程序或稳定性	设计员负责设计驱动程序，实施员负责实施驱动程序
执行整合测试	测试脚本（可选） 工作版本	测试结果	测试员负责执行测试并记录测试结果
评估整合测试	整合测试计划 测试结果	测试评估摘要	测试设计员负责会同集成员、编码员、设计员等有关人员（具体化）评估此次测试，并生成测试评估摘要。

整个 Android 测试的流程主要如图 6-5 所示：

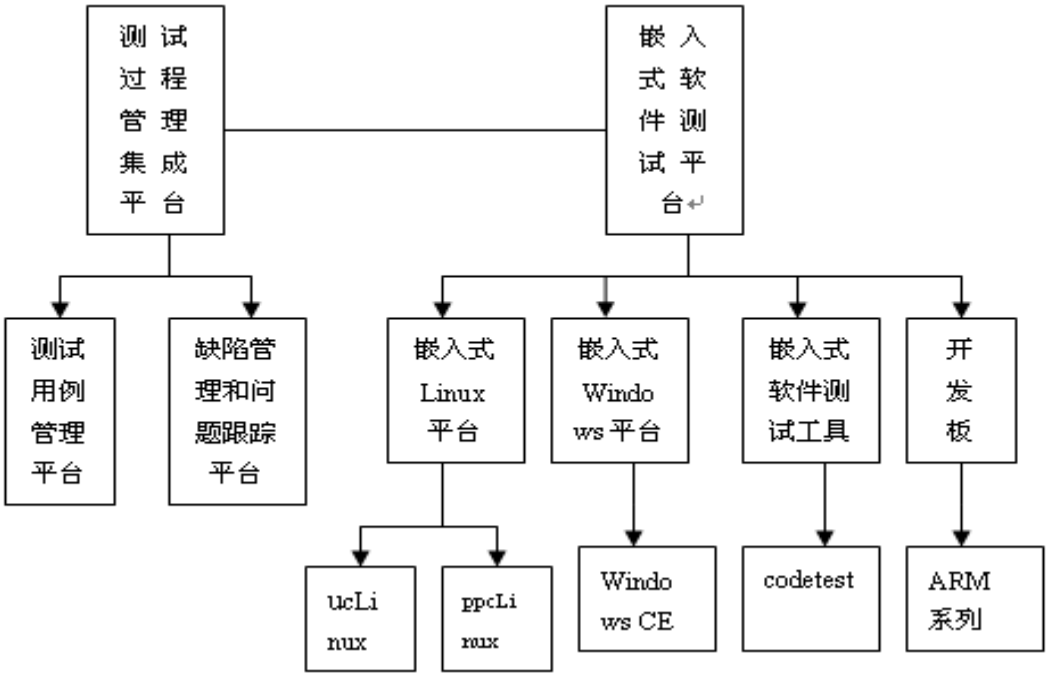

图 6-5Android 测试流程图

上一节是对系统进行单元测试，在单元测试完成后，要对系统进行整合测试。在单元测试中，是把各个功能组合到一起进行测试，而在整合测试中，是把各个单元整合到一起。具体的测试内容如下所示：

1、规划测试

2、信息测试

3、显示测试

4、客户端测试

5、平台测试

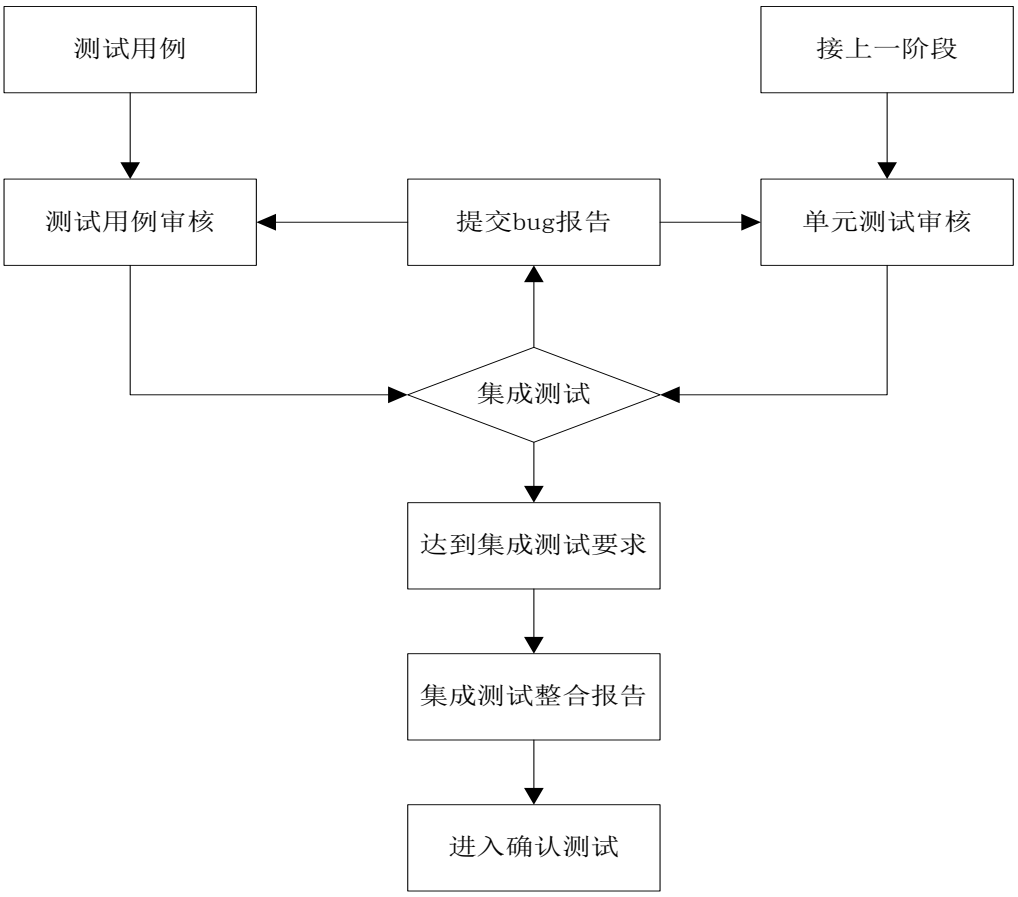

图 6-6 测试流程图

测试流程图如图 6-6 所示，整个测试是一个循环的过程，首先要进行各个功能点的循环测试，保证功能点连接没有问题后，才能进行规划测试、信息测试、显示测试等环节的测试。规划测试是指整个系统的功能点是否和规划中的一致，应该对着设计文档验证各个功能，在本文中，主要验证了基础状态管理模块、耗电量排行模块、省电模式管理模块。信息测试是指要验证系统显示的信息是否正确，信息主要包括文本信息、图像信息、提示信息、字体颜色等，在本系统的测试中，主要按照信息测试的基本点进行了测试，能够保证用户的正常使用。

6.6 本章小结

这章节对本研究的软件进行了全面的测试，对测试的内容进行了详细的介绍，并列出整个软件测试的详细流程与标准。我们进行软件测试的时候，需要严格按照此标准进行测试。我们进行了多次的测试，最终得出该软件是可靠，稳定的，完全满足客户的需求。

第七章 结论

Android 平台的电源管理软件已经发展成手机中不可缺少的一个软件，此软件主要有四个功能，即实时显示电量、耗电量排行、智能省电和设置。通过上述的 4 个功能，用户可以对自己手机的耗电情况有清晰的了解。用户可以通过查看实时电量来决定是否要充电，通过耗电量排行可以查看手机中最耗电的部分，在使用手机中尽量减少耗电量较高的功能。设置功能中包括智能省电，它有两种模式，即手工模式和自动模式，在手工模式中，可以选择关掉应用程序，而在自动省电模式中，程序会自动判断哪些应用程序是不使用的，从而将其管掉。本系统严格的按照软件开发的流程进行设计和开发，在本文中，首先对系统进行了分析，包括系统的需求分析、性能分析等，通过系统的分析，可以确定系统是否可行的、系统的主要功能以及对性能的要求。在完成系统的分析后，要对系统进行详细的设计，首先要对系统的设计原理进行说明，然后对电量实时显示、耗电量排行、智能省电、CPU 省电等功能进行详细的设计，这些设计不仅包含功能，而且还要画出流程图以及数据流图，设计的功能要满足在系统分析中所确定的，并保证系统的性能。

至此，论文的主要内容已经大致完毕了，通过这篇文章的写作，我做到了温故知新，更好的运用自己学到的知识。这篇文章主要对 Android 电源管理机制进行了专研和琢磨，在本文的开头是对选题的背景等方面进行了阐述，清楚的罗列出了本文要做研究些什么内容，研究这些有怎样的作用。接着对文中要用到的概念进行了详细的解释和探究，包括 java 技术等做出了详尽的阐述。经由第二部分的阐明，能够对文章研究的具体内容有更上一层楼的认识，同时给接下来的研究奠定理论基础。而文章的第三部分，确定了软件系统主要的功能需求。第四部分根据需求和基础理论对系统进行设计，包括了功能模块设计、省电原理、代码结构等内容。第五部分是根据第三部分的需求和第四部分的设计，对系统进行实现，使设计的功能模块能够在 android 手机上运行。

我水平有限，因此看到和总结的内容可能流于肤浅，我准备要在经后多对这方面的相关内容进行思索和探究，不停的充实自己，进而希望提出解决的社会问题的更完善的办法。本论文还有很多不到位的地方，在以后的学习和工作中，要追求尽善尽美。

致谢

光阴荏苒，历史三年的工程硕士研究生的学习也接近尾声。历经大半年时间的磨砺，终于完成这篇硕士论文。从毕业论文题目的选择，到后期规划，查找相关资料，具体实施，到最后结束论文，在这个过程中，我的导师——杨元杰研究员和王荣芝高级工程师给了我很多指导和建议。两位导师不仅学术渊博、治学严谨、也宽容待人，没有他们的帮助，我的论文也不会顺利完成。在这里谨向两位导师致以最真诚的谢意。

同时，我也需要感谢电子科技大学工程硕士班的教师们，他们真的也很辛苦苦，利用周末返于成都与昆山两座城市，真的非常辛苦，在此，我也对这些默默奉献的老师表示我最衷心的感谢！

最后，我还要感谢专家评审们，你们也辛苦了！

参考文献

[1] 杨丰盛.于 Android 平台的动态电源管理技术研究[J].计算机与现代化，2010(11):75-77

[2] 张海绒.基于 Android 平台的动态电源管理技术研究[J].云南大学,2011(05):102-157

[3] 梁衡,刘新新,郑远.面向 Android 平台的动态电源管理优化[J].河南科技学院学报(自然科学版),2012(05):37-49

[4] 苏健.Android 智能手机平台电源管理技术[J].微处理机,2012 年(08):89-96

[5] 林春成.便携电子设备的电源分布及电源管理设计[J].机电工程技术,2011(09):48-56

[6] 奥斯本(Osborne,k.).深入理解 Oracle Exadata. [M].电子工业出版社，2012,35-40

[7] 姚昱昱,刘卫国.Android 的架构与应用开发研究[J].计算机系统与应用，2011(11):110-112

[8] 段涛,刘明兰.智能手机技术的发展与剖析[N].现代通讯,2007,第 1-2 期,20-32

[9] M.Clerc. Machinery Industry Press, 4th edition[J], 2009,2010,31-45

[10] Roger Android application development[J],2009,30-42

[11] Hashemi, McLean (U.S.),TWL3016 Datasheet.Proficient in Android,2011(09),82-89

[12] M.Ranjan,K.H.Koo,G.Hanhgtan,et al.Microwave power amplifiers with digtallycontrolledpower supply voltage for high efficiency and high linearity[J].2010,1:493-496

[13] Xu Qing,Wu Jian,Yang Lei.et al.Development of 450 MHz LTCC power amplifier module forCDMA mobile phone[J].Nanjing,2010,26(4):477-480

[14] D.Sinai.Analogue dialogue[3G mobile phone power saving][J].Electronic Systems and Software,2010,3(1),16-19

[15] 姚刚.面向环保电源管理系统的设计趋势[J].电子设计技术,2009(04):56-72

[16] 宇昕.多媒体智能手机电源管理的优化[J].现代通信,2011(09):108-115

[17] 陈逸飞,尹长青.基于嵌入式 Linux 的智能手机省电设计[J].电脑知识与技术，2009(11):68-81

[18] Manishasha Pandya,Andy Khayat.手机系统的电源管理[J].世界电子元器件,2011(06):19-25

[19] 方胜.基于安卓平台的电源管理机制的研究[J].广东工业大学计算机学院,2011:87-95

[20] 徐进.只能手机电源管理模块和音频模块的设计[J].少年宫交通大学,2010(12): 55-75

[21] 赵炯.Linux 内核完全注释[M].北京:机械工业出版社,2004:142-151

[22] 林春成.便携电子设备的电源分布及电源管理设计[J].机电工程技术,2008(12):45-51

[23] 杨丰盛.Android 技术内幕.系统卷[M].北京:机械工业出版社,2011:63-58

[24] DanielP.Bovet&MarcoCesati.深入理解 Linux 内核（影印版）[M].南京:东南大学出版社,2006:92-107

[25] 邬群勇,王钦敏,王焕炜.一种 Web 地图服务搜索器的设计[J].微计算机应

用,2009,30(2):35-39

[26] Eyhab Al-Masri, Qusay H. Mahmoud.Investigating web services on the world wide web[C]. WWW 08 Proceeding of the 17th international conference on World Wide Web, ACM New York, NY, USA,2008,530-601

[27] Ianchun Fan,Subbarao Kambhampati.A snapshot of public web services[C]. Newsletter ACM SIGMOD Record Homepage archive Volume 34 Issue 1,2005,301-312

[28] Eyhab Al-Masri,Qusay H,Mahmoud.WSCE.A crawler engine for large-scale discovery of web services[C].IEEE International Conference on Web Services,Salt Lake City,Utah,USA,2009,251-259

[29] 徐远超,刘江华,刘丽珍,关永.基于 Web 的网络爬虫的设计与实现[J].微计算机信息,2007,(21):34-38

[30] 吴亚鑫,孙静.基于数据库操作的相关性模型及应用[N].计算机工程与设计,2011 年第 1 期,10-13

[31] 张桂元,贾燕枫.《Eclipse 开发入门与项目实践》[M].人民邮电出版社,2006,47-50

[32] 萨师煊,王珊.《数据库系统概论(第三版)》[M].高等教育出版社,2002,15-24

[33] 覃庆炎.《Java 开源项目：Spring+Hibernate+Struts 项目开发详解》[M].清华大学出版社,2008,36-49

[34] Kline，K.E., Kline，D,Hunt，B.《SQL 技术手册》[M].东南大学出版社,2008,85-98

[35] Y.Daniel Liang.《JAVA 语言程序设计》[M].王镁,李娜译.机械工业出版社,2002,101-110

[36] Mark Grand《JAVA 模式》[M].亢勇,豆庆华译,电子工业出版社,2004,62-69

[37] Dave Crane,Eric Pascarello,Darren James《Ajax 实战.》[M].人民邮电出版社,2006,72-79

[38] 张桂元,贾燕枫.《Eclipse 开发入门与项目实践》[M],人民邮电出版,2006,35-64

[39] Izaki K,Tanaka K,Takizawa M.Access Control Model in Object-oriented System.IEEE[J].Microsoft Press, 2000,13-14

[40] Moyer M J.Ahamad M.Generalized Role-based Access Control.IEEE, 1063-6927, 2001, 50-63

[41] Konstantin Beznosov. Engineering Access Control for Distributed Enterprise Applications [D] , [Dissertation for Ph. D .] , Florida International University , Miami, Florida , USA . Jul y, 2000, 43-45

[42] Reto Meier 《Professional Android2 Application Development》[M].清华大学出版社 2010，66-77

[43]Rick Rogers,Blake Meike,Ziqurd Mednieks《Android 应用开发》[M].人民邮电出版社 2010,129-134

[44]Grant Allen,Mike Owens 《The Definitive Guide to SQLite(Second Edition)》[M].电子工业出版社 2012,47-53

附录

```
public class BookEvent extends Event {
    OnClickListener listener0 = null;
    OnClickListener hearday1 = null;
     /** Called while the event was established first. */
     Anjian anjian0;
     Anjian anjian1;
     public void inChuangjian(Cadefn shunjianbaocunzhuangtai) {
          chaoji.inChuangjian(shunjianbaocunzhuangtai);
          shezhiBundandSight(R.shuchu.zhuti);
          hearday0 = xin InDianjiHearday() {
              public void inDianji(Sight v) {
                   Ondand Ondand0 = xin Ondand(BookEvent.this, ZhuceEvent.class);
                   shezhiBiaoti("用户登录");
                   beginEvent(Ondand0);
              }
          };
          hearday1 = xin InDianjiHearday() {
              public void inDianji(Sight v) {
                   Ondand Ondand0 = xin Ondand(BookEvent.this, cadleEvent.class);
                   shezhiBiaoti("电量管理");
                   beginEvent(Ondand0);
              }
          };
          Anjian0 = (Anjian) seekSightById(R.id.quxiao);
          Anjian0.shezhiOnDianjiHearday(hearday0);
          Anjian1 = (Anjian) seekSightById(R.id.denglu);
          Anjian1.shezhiOnDianjiHearday(hearday1);
     }
public class cadleEvent chaoguo Event {
    OnDianjiHearday hearday3 = null;
    OnDianjiHearday hearday4 = null;
```

```
    Anjian Anjian3;
     Anjian Anjian4;
     /** Called while the event is established first. */
     @Override
     public void inChuangjian(Cadefn shunjianbaocunzhuangtai) {
          chaoji.inChuangjian(shunjianbaocunzhuangtai);
          shezhiBundandSight(R.shuchu.cadle);
          hearday3 = xin OnDianjiHearday() {
              public void onDianji(Sight v) {
                   Ondand Ondand0 = xin Ondand(cadleEvent.this,
ListAllBook.class);
                   shezhiBiaoti("2");
                   beginEvent(Ondand0);
              }
         };
            hearday4 = xin OnDianjiHearday() {
                   public void onDianji(Sight v) {
                        Ondand Ondand0 = xin Ondand(cadleEvent.this,
myBook.class);
                        shezhiBiaoti("2");
                        beginEvent(Ondand0);
                   }
              };
              Anjian3 = (Anjian) seekSightById(R.id.Anjian3);
              Anjian3.shezhiOnDianjiHearday(hearday3);
              Anjian4 = (Anjian) seekSightById(R.id.Anjian4);
              Anjian4.shezhiOnDianjiHearday(hearday4);
     }
     public class CastAllBook chaoguo Event {
         private Cast<Ditu<String, Object>> data;
         private CastSight castSight = null;
         private static final int DIALOG1 = 1;
           protected Dialog onEstablishedialog(int id) {
```

```
                switch (id) {
                case DIALOG1:
                    return buildDialog1(CastAllBook.this);
                }
                return null;
            }
        @Override
        private Dialog buildDialog1(Context context) {
            AlertDialog.Builder builder = xin AlertDialog.Builder(context);
            builder.shezhiMessage("选择成功！请继续！");
            return builder.chuangjian();
        }
    public class myBook chaoguo Event {
        private Cast<Ditu<String, Object>> data;
        private CastSight castSight = null;
        private static final int DIALOG1 = 1;
        @Override
        public void onChuangjian(Cadefn shunjianbaocunzhuangtai) {
            chaoji.onChuangjian(shunjianbaocunzhuangtai);
            PrepareData();
            castSight = xin CastSight(this);

            SimpleAdapter adapter = xin SimpleAdapter(this, data,
                    R.shuchu.mybook, xin String[] { ""}, xin int[] {
                            R.id.msight3 });
            castSight.shezhiAdapter(adapter);
            shezhiBundandSight(castSight);
        }
        private void PrepareData() {
            data = xin ArrayCast<Ditu<String, Object>>();
            Ditu<String, Object> item;
            Conn c=xin Conn();
            String sql="select * from yuding";
```

```
        System.out.println(sql);
        ResultShezhi rs=c.select(sql);
        try {
            while(rs.next()){
                item = xin HashDitu<String, Object>();
                item.put("", "：  "+rs.getString("bookname")+"                ");
                data.add(item);
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    OnDianjiHearday hearday0 = null;
    private static final int DIALOG1 = 1;
     /** Called while the event is first established. */
     Anjian Anjian0;
    /* public void onChuangjian(Cadefn shunjianbaocunzhuangtai) {
          chaoji.onChuangjian(shunjianbaocunzhuangtai);
          shezhiBundandSight(R.shuchu.zhuce);
          hearday0=xin OnDianjiHearday() {
            public void onDianji(Sight v) {
                EditText edit_text = (EditText) seekSightById(R.id.username);
                CharSequence edit_text_value = edit_text.getText();
                shezhiBiaoti("用户名是：:"+edit_text_value);
            }
        };
        Anjian0 = (Anjian) seekSightById(R.id.zhuce);
        Anjian0.shezhiOnDianjiHearday(hearday0);
     }
     }*/
     protected Dialog onEstablishedialog(int id) {
        switch (id) {
```

```
            case DIALOG1:
                return buildDialog1(ZhuceEvent.this);
            }
            return null;
        }
         public void onChuangjian(Cadefn shunjianbaocunzhuangtai) {
            chaoji.onChuangjian(shunjianbaocunzhuangtai);
            shezhiBiaoti("用户注册");
            shezhiBundandSight(R.shuchu.zhuce);
            seek_and_modify_text_sight();
        }
        private void seek_and_modify_text_sight() {
            Anjian get_edit_sight_Anjian = (Anjian) seekSightById(R.id.zhuce);
    get_edit_sight_Anjian.shezhiOnDianjiHearday(get_edit_sight_Anjian_hearday);
        }
        private Anjian.OnDianjiHearday get_edit_sight_Anjian_hearday = xin
Anjian.OnDianjiHearday() {
            public void onDianji(Sight v) {
                EditText edit_text = (EditText) seekSightById(R.id.name);
                BianjiWenben mima = (BianjiWenben) seekSightById(R.id.mima1);
                CharSequence bianji_wenben_value = bianji_wenben.getWenben();
                CharSequence    mima_value=mima.getWenben();
                System.out.println(bianji_wenben_value);
                System.out.println(mima_value);
                try {
                    //requwstByGet();
                } zhuazhu (Chuquliwai e) {
                    // QECE Zidongshengcheng zhuazhu block
                    e.printStackTrace();
                }
        sqlCon(String.valueOf(bianji_wenben_value),String.valueOf(mima_value));
                showDialog(DIALOG1);
            }
```

```
};
```

Android电源管理机制研究与软件设计

作者： 鲁云霞
学位授予单位： 电子科技大学