学　　号　14284060xx
等　　第

苏州大学实验报告

Linux下计步器软件设计

院(系)名称：　电子信息学院

专业名称：　14通信工程(嵌入式培养)

学生姓名：　某某某

课程名称：　Linux操作系统

2015-2016学年第一学期

摘要

随着人们生活水平的不断提高，人们的健康意识得到了空前广泛的提高，越来越多的人以锻炼的方式来缓解自己在工作、学习和生活中的种种压力。在现代人们的生活节奏普遍提高的情况下锻炼的效果成为人们关心的话题，为了能使锻炼的效果达到人们预计的目标，计步器的使用成为人们的最佳选择。

本报告主要介绍在 Google Android 平台智能手机中计步器软件的开发及测试。从软件的概要设计到详细设计一一作了说明，并且着重介绍了软件界面的实现与计步原理。通过分析传感器数据得出计步算法，并对此算法提出了优化；同时，文章还介绍了计步器软件中其他涉及到的数据并作相应的处理。最后文章还从实践的角度出发，验证计步器软件在真机中的运行情况。

关键词：Google Android；计步器；智能手机；计步原理

合作者：　合作甲　14284060xx　14 通信工程 (嵌入式培养)
合作乙　14284060xx　14 通信工程 (嵌入式培养)
合作丙　14284060xx　14 通信工程 (嵌入式培养)

Abstract

With the continuous improvement of living standard, people's awareness of health have improved a lot, more and more people want to find a better way to ease their work, study and life pressures. Pace of life in the modern society, a general increase in the case of the effect of exercise become a topic of concern. In order to enable an effect way of exercise to achieve the objectives which was expected, pedometer becomes the best choice.

This report describes the Google Android platform in the pedometers smartphone software development and testing. Summary from software design to detailed design, explained one by one, and focuses on the realization of the software interface of step with the principles of dollars. Sensor data obtained by analyzing the total-step method, and optimization of this algorithm is presented; the same time, the article also introduces the pedometer software and other data relating to the treatment accordingly. Finally, the paper also from a practical point of view, verify pedometer software in real machine in operation.

Key words: Google Android; Pedometer; Smart phones; meter-step

目录

第 1 章　绪论

1.1　Google Android 平台的发展与应用

随着 3G 时代的到来，无线带宽越来越高，使得更多内容丰富的应用程序布置在手机上成为可能，如：视频通话、视频点播、移动互联网冲浪、在线看、听歌、内容分享等。为了承载这些数据应用及快速部署，手机功能将会越来越智能、越来越开放。为了实现这些需求，必须有一个好的开发平台来支持，在此由 Google 公司发起的 OHA 联盟走在了世界的前列，于 2007 年 11 月推出了开放的 Android 平台，任何公司及个人都可以免费获取到源代码及开发 SDK。由于其开放性和优异性，Android 平台得到了业界广泛的支持，其中包括各大手机厂商和著名的移动运营商等。继 2008 年 9 月第一款基于 Android 平台的手机 G1 发布之后，三星、摩托罗拉、索爱、LG、华为等公司都将推出自己的基于 Android 平台手机，中国移动也将联合各大手机厂商共同推出基于 Android 平台的 OPhone。按照目前发展的态势，我们有理由相信，Android 平台能够在短时间内跻身智能手机开发平台前列 [1, 2]。

2009 年 1 月 7 日，工业和信息化部为中国移动、中国电信和中国联通发放 3 张第三代移动通信 (3G) 牌照，此举标志着我国正式进入 3G 时代。中国 3G 正式商用和规模建网，App 商店在全球被运营商和终端厂商热捧；中国移动 App 商店将不久上线，这些背景，共同决定了中国基于 Android 的应用程序开发在未来处于一个爆发和蓬勃成长期，这也为软件开发者提供了一个淘金的好机会，也迫使大量此前不熟悉 Android 开发的程序员要迅速进入此领域，以便在经历了单机计算时代和传统互联网时代之后，能够在移动互联网时代的元年开始赶上这趟列车，成为了许多程序员的愿望。

1.2　计步器介绍

由于现代人们的生活节奏的加快和生活水平的不断提高，越来越多的人们开始关注自身的身体素质，在尽可能短的时间内达到锻炼的最佳效果，这时就需要一种计步装置。

计步器是一种内置传感器的计步装置。它一般用于帮助人们记录运动的步数以及所消耗的热量，从而达到使用者的锻炼目的。它携带于使用者身上，通过感应使用者的震动来记录步数。计步器的功能比较多，即可以选择不同的运动模式、传感器的精

度，还可以帮助人们记录运动所消耗的热量。

传统的计步器一般为专用计步器，其内部构造为硬件电路搭建而成，不同厂家也采用各自不同的计步算法，计步可以达到一定的精度。

由于手机硬件的飞速发展，现在的不少智能手机已内置的加速度计，在这种情况下，已没有必要购买单独的计步器。通过编写合适的软件即可达到相同的目的计步目的，并且根据使用者的身体状况来输出所消耗的热量等数据。

1.3　内容安排

本设计主要介绍在 Google Android 平台下设计一个计步器程序。报告从以下几个章节分别介绍了软件的实现：

第 1 章介绍了计步器软件开发的背景与实际意义；

第 2 章从程序概要设计方面介绍了软件开发的具体流程；

第 3 章开始软件的详细设计介绍，并从统一建模语言 UML (Unified Modeling Language) 角度规划出系统各类之间的关系 [3]；

第 4 章为程序设计核心内容，介绍了如何依据手机传感器数据来计步的几种算法，并比较了各算法之间的优劣性；

第 5 章介绍如何依据步数来计算出其他用户感兴趣的数据；

第 6 章测试程序并记录了数据。

第 2 章　程序概要设计

2.1　任务概述

本设计是在 Google Android 平台上开发手机计步器软件，通过利用 Android SDK 模拟器 [4] 来帮助软件的调试、运行，主要任务如下：

- 通过分析真机传感器的数据，能够提供较为精确的计步算法；
- 提供友好程序界面，并提供用户与手机交互信息的接口；
- 通过计步数据得出其他用户感兴趣的信息，如里程、速度、消耗热量等；
- 实现软件在 Android 智能手机上无 bug 运行，测试并记录数据。

2.2　系统用例图

Google Android 计步器软件提供人机交互界面，通过用户输入数据 (或程序默认值) 来运行，其用例图如图 2.1 所示。

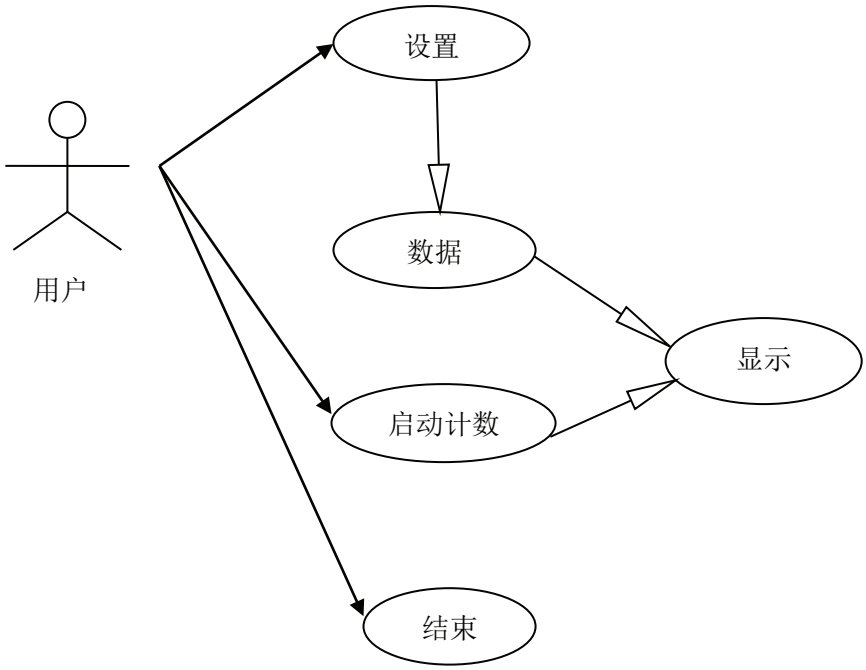

图 2.1: 计步器程序用例图

首先，用户打开程序界面，通过设置来保存自身数据。数据保存返回后，用户启动计步程序，同时，将设置数据与计步数据联合可以计算出其他用户感兴趣的信息，并在屏幕上显示出来。另外，用户可以通过退出菜单来结束程序。

2.3　用户操作流程

用户操作流程图如图 2.2 所示。

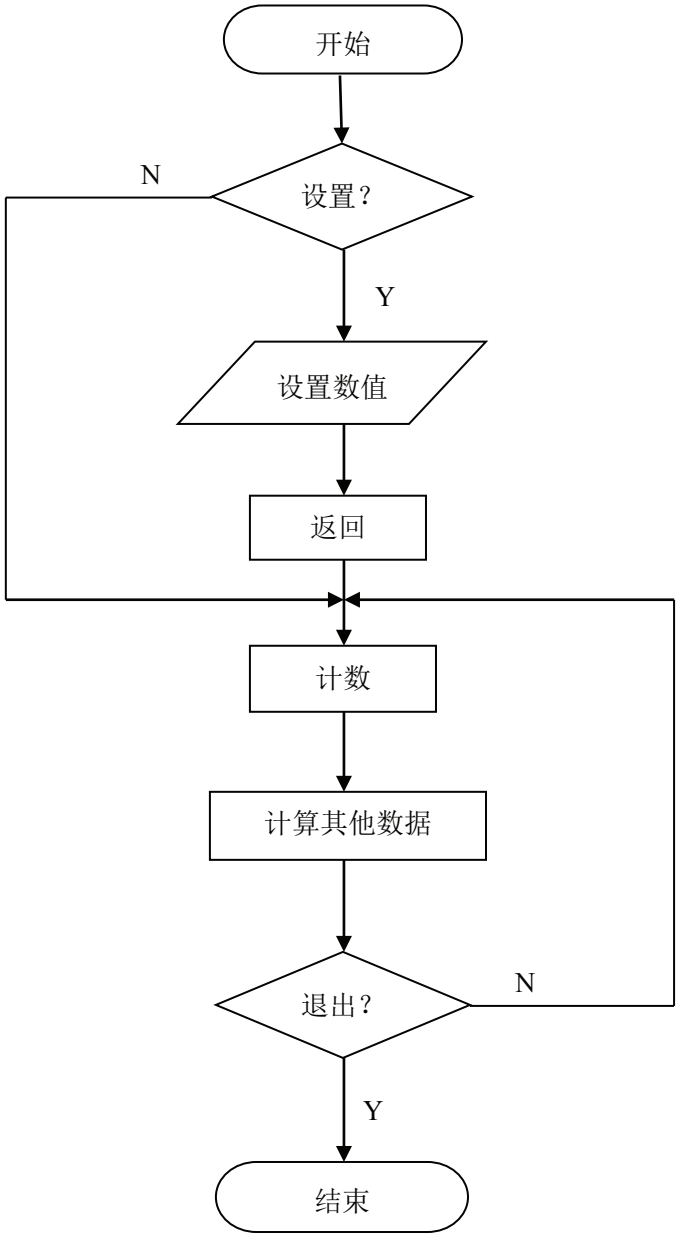

图 2.2: 用户操作流程图

从操作流程图可以看出，程序提供用户设置窗口，用户输入自身参数 (可选，若用

户直接启动程序计数，则参数为默认值)，如体重、步长等，通过程序内部数据传递得到用户参数。设置返回后程序启动计步功能，并将步数显示在屏幕上。另一方面，通过程序获得手机系统时间，计算得出步速、里程、时速等信息。当用户结束本次运动后计步结束，退出计步器程序。

2.4　程序内部数据流图

程序内部数据流图如图 2.3 所示。

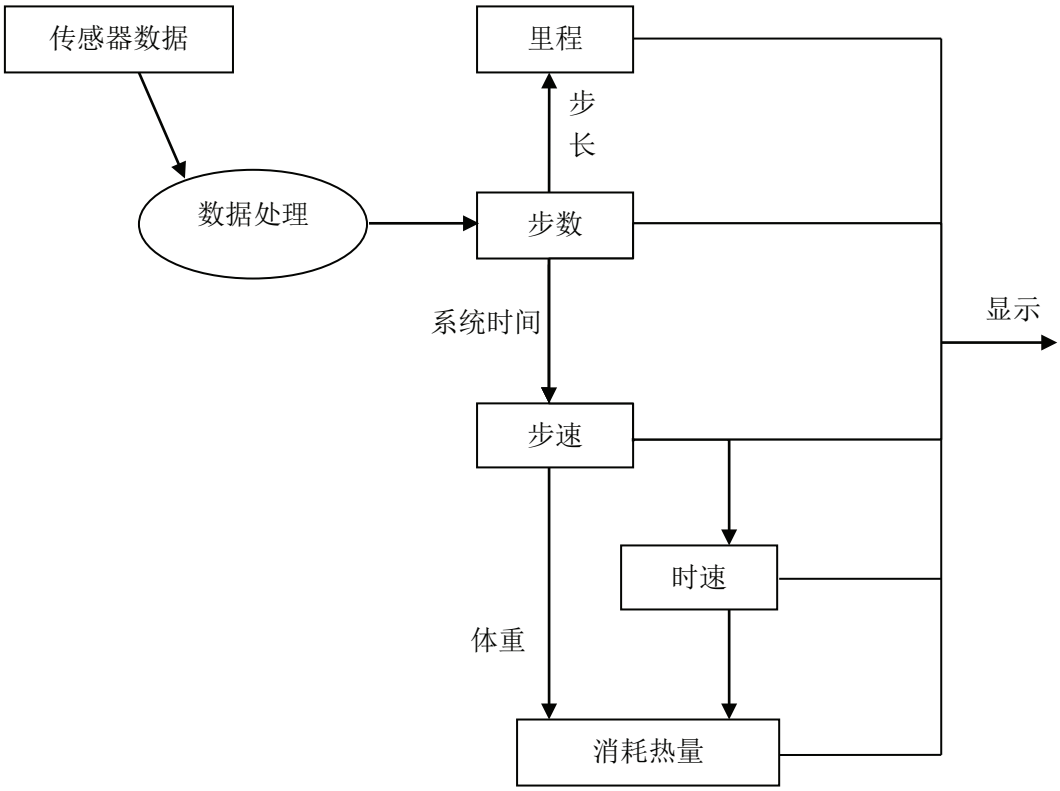

图 2.3: 程序内部数据流图

首先，程序读取手机内置传感器 (这里特指加速度计) 数值，经数据分析处理得到步数。用户步长与步数计算可以直接得到里程；步数依据系统时间，经延时可以计算出步速。热量消耗通过体重、时速可由公式得出。最终这些数据在屏幕上特定区域显示，通过不断的数据更新来刷新屏幕。

第 3 章　详细设计

上一章主要从软件概要设计方面介绍了软件需要完成的基本功能以及用户的操作流程。这一章从如何实现来详细说明软件的设计过程。计步器软件详细设计主要涉及到程序界面设计以及程序代码的设计。

3.1　界面设计

手机计步器软件要求提供友好界面，这就要求在界面的设计过程当中，不仅要完成程序的基本功能，而且要以用户体验的原则，设计出符合用户需求的界面。这要从界面的布局、颜色设置等方面予以考虑。

3.1.1　界面设计框架结构

程序设计中需提供人机交互界面，当用户在主界面状态下按下设置菜单时，界面需跳转到设置窗口。其结构如图 3.1 所示。

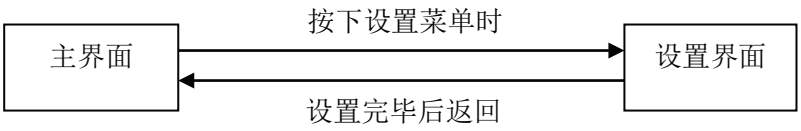

图 3.1: 界面结构框架示意图

其中，主界面是程序显示数据信息的窗口，包括步数、里程、速度、消耗热量等参数。并提供从主界面切换到设置界面的菜单。设置界面包含用户输入 (体重、步长) 和选择 (性别、运动模式以及传感器灵敏度) 区域，并提供返回按钮。

3.1.2　界面布局设计

界面布局 (Layout) 为活动构造了用户界面的结构，布局包含了要展现给用户的所有组件及各组件之间的结构，在 Google Android 设计中可以用以下两种方法来设计布局：

1. 用可扩展标记语言 XML(Extensible Markup Language) 申明用户界面的组件 — —相应于视图类及其子类，Android 提供了简单而明确的 XML 语法。

2. 在应用程序运行时来实例化布局 ——可以在应用程序中使用编程语句来创建视图和视图组件对象。

上述第一种方法可以实现大部分界面功能，如添加文本框 (TextView)、输入框 (EditView)、按钮 (Button) 等。而在实际设计中有些界面却也可以用上述第二种方法，即编写代码的方式来实现，如完成动态效果、菜单等。以上这两种方法在计步器程序界面的设计中均用到，以下逐一说明。

(一) XML 设计布局 [5]

用 XML 定义用户界面的好处是，它可以更好地把应用程序界面的展现部分和行为的控制代码分隔开来。用户界面的描述对应用程序代码来说是外在的，因此，我们可以很方便地修改界面而不需修改和重新编译代码。另外，在 XML 文件中定义布局，使用户界面更形象化，更易于调试应用程序。

使用 Android 的 XML 语法 [6]，可以快速地设计用户界面的布局以及其中包含的屏幕元素，每一个布局文件都要明确地包含一个根元素，这个根元素必须是 View 或 ViewGroup 对象。我们可以用树状图来描述和定义一个活动的用户界面，其结构图如图 3.2 所示。当定义好根元素后，就可以把额外的布局对象或根组件作为元素的子元素 (如 TextView、Button、EditView 等) 添加进来，这样，就逐渐创建了一个可以定义整个布局的视图层次结构。

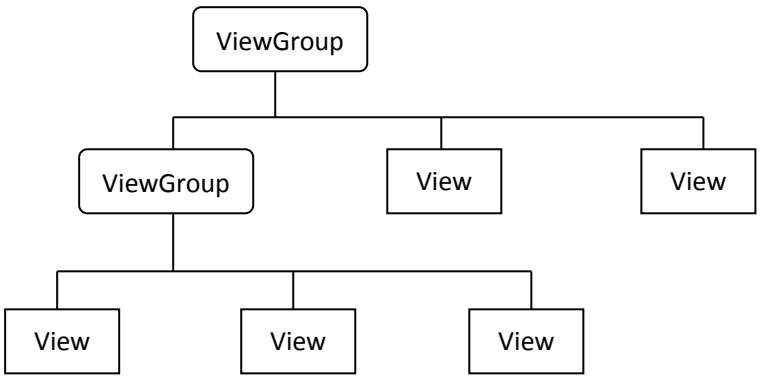

图 3.2: View 与 ViewGroup 结构图

图 3.2 中，View 为视图，是 android.view.View 的一个实例，它是一个存储有屏幕上特定的矩形内布局和内容属性的数据结构。视图负责处理它所代表的屏幕布局、测量、绘制等。ViewGroup 为视图组，是 android.view.ViewGroup 的一个实例。它是一个特殊类型的视图，其功能是装载和管理一组下层的视图和其他视图组，可以为界面增加结构，创建复杂界面元素。应用上述方法创建如图 3.3 为程序主界面结构示意图。

其中：

(1) 代表界面一个视图组 ViewGroup，负责装载其他视图 (组)；

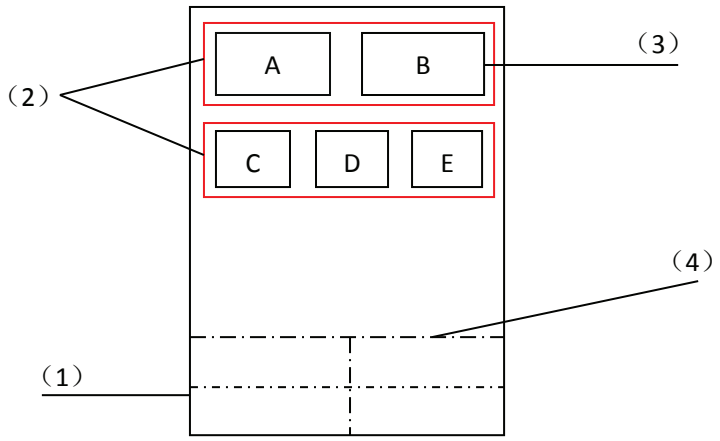

图 3.3: 主界面结构示意图

(2) 表示嵌于 (1) 内的两个 ViewGroup；

(3) 为文本显示区 TextView，包括区域 A、B、C、D、E，分别显示当前步数、里程、步速、时速、消耗热量。设计中可以通过设置相应的参数，以改变各区域的颜色：(1) 区位黑色背景，A、B、C、D、E 均设置成绿色。

(4) (图中虚线部分) 为菜单弹出区域，它的设计可以采用 (1)(2)(3) 的 XML 设计方法，也可以采用通过应用程序来实例化布局，在计步器程序的设计中使用了第二种方法，这在后面会进一步说明。

通过图 3.3 可以看出，主界面追求一种简约风格，不仅满足了程序的需求，而且也使得界面美观大方。

同样，设置界面结构示意图如图 3.4 所示，其中：

(1) 代表界面一个视图组 ViewGroup，负责装载其他视图 (组)；

(2) 表示控件 Button，用于设置完毕后返回至主界面；

(3) 为 3 个视图组件，用于布局其内部视图。

A1、A2 区域为文本提示框 (固定不变)，A1 为 “请输入步长”，A2 为 “请输入体重”；

B1、B2 为用户编辑窗，分别对应于步长和体重的值；

C1、C2 为两个选择 (Spinner) 窗口，分别为运动模式选择和传感器灵敏度选择，这里采用第二种布局方法，即通过程序来实例化布局来实现待选参数的设置。

从图 3.4 可以看出，设置界面与主界面一样，在追求简约的风格的同时得到了程序运行所必须的数据，同时使界面看上去美观大方，符合了用户和设计者的初衷。另外，在用户点击 C1、C2 的过程中还设计了动态效果 (见下文)，大大减少了枯燥了程序界面。

(二) 程序实例化布局

- Menu 菜单类

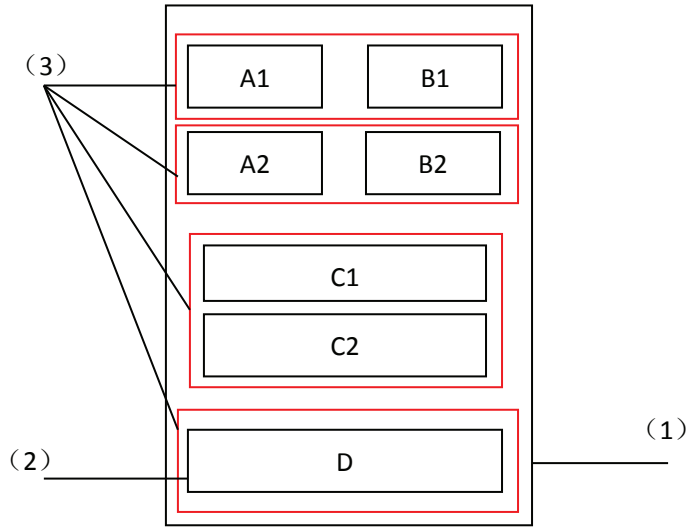

图 3.4: 设置界面结构示意图

图 3.3 中 (4) 中虚线部分为为当用户按下 Menu 键时弹出的，它用来管理菜单上的条目，这也是程序提供丰富功能的一条重要途径，同时对于编写友好界面有着举足轻重的作用。Menu 菜单实现的具体类图如图 3.5 所示。

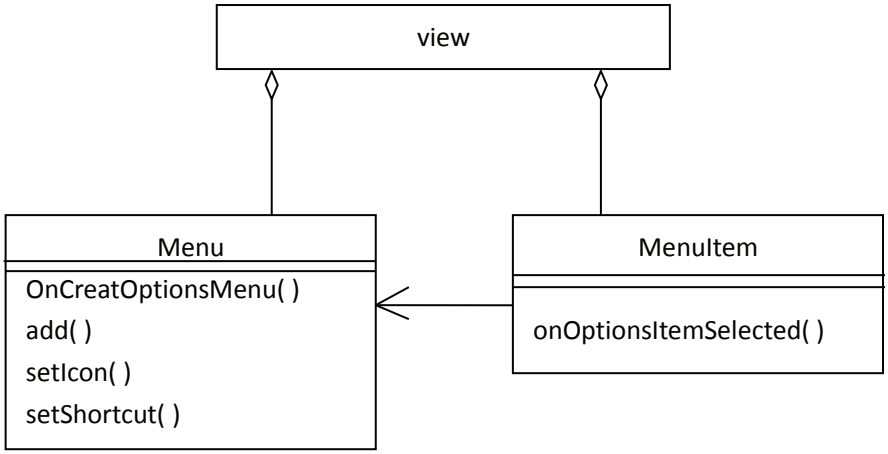

图 3.5: Menu 菜单类类图

从图 3.5 可以看到，Menu 类和 MenuItem 类均继承于 view 类，view 类是一个视图类，用于实例化视图产生可视化界面。MenuItem 类与 Menu 类之间有着关联关系，Menuitem 中的 onOptionsItemSelectd () 方法需要找到 Menu 实例化后的名字，因此，必须在先实例化 Menu 后才能实例化 MenuItem。Menu 菜单其具体实现步骤 [7] 如图 3.6 所示。

从图 3.6 可以看出，当用户按下 Menu 时，程序才实例化菜单，即首先调用父类函数来构建菜单视图，然后在菜单视图里添加相应的选项，如帮助、设置、退出等，最后

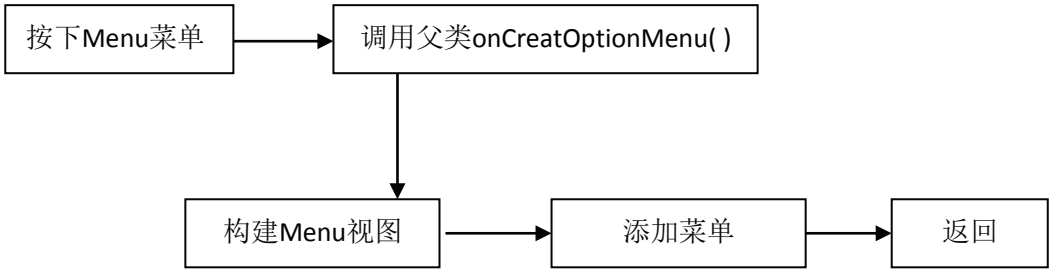

图 3.6: Menu 实现结构框图

再返回一个值 (True)，这样完整的菜单就创建完成了。

当菜单创建完毕，用户选择相应的菜单时便执行相应的功能。Menu 执行框图如图 3.7 所示。

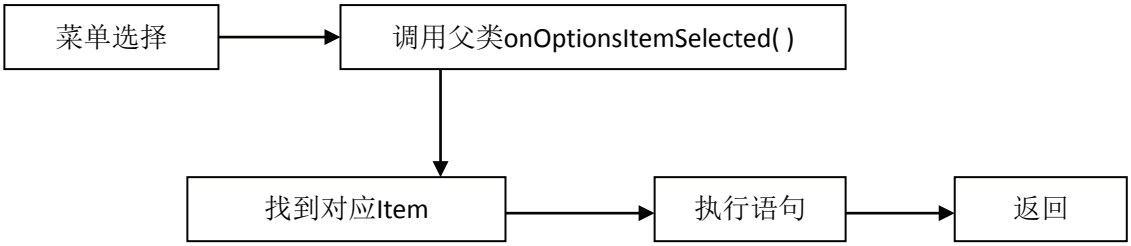

图 3.7: Menu 菜单执行框图

- 动态效果的设置

Android 平台提供了两类动画：一类是 Tween [8] 动画，即通过对场景里的对象不断做图像变换 (平移、缩放、旋转) 而产生动画效果；第二类是 Frane 动画，即顺序播放事先做好的图像。在上面的界面设计中使用到了第一类 Tween 中的 Animation 类来实现其动态效果。Animation 类关系如图 3.8 所示，TranslateAnimation、RolateAnimation、AlphaAnimation 等都是 Animation 的子类，分别实现了平移、旋转、和改变 Alpha 等值动画。

图 3.4 中 C1、C2 选择的实现是以子类 TranslateAnimation 来实现，在实现过程中，为了方便，将动态效果的设置单独存放于 `res\anim\my_anim.xml` 文件下。这样做不仅使界面与代码适当分离，增强代码的独立性，而且使它的实现相对简单，即只需导入 (loadAnimation) 即可。

3.2　代码设计

计步器程序代码分为两部分，一部分为用户数据设置部分，另一部分为主程序 (在 Android 程序中也称为主程序)。这两部分之间的关系结构如图 3.9 所示。

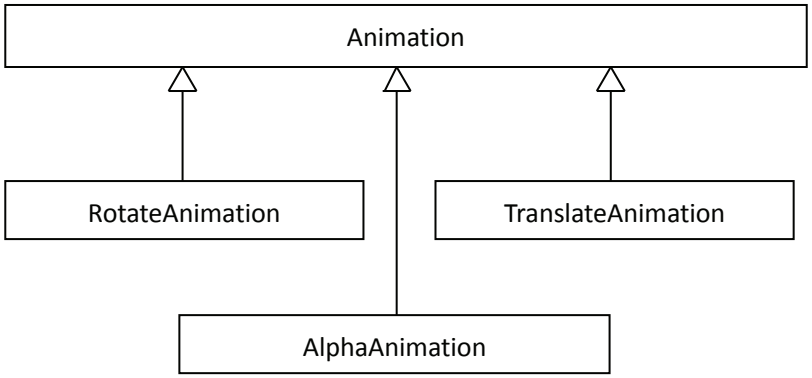

图 3.8: Animation 类关系图

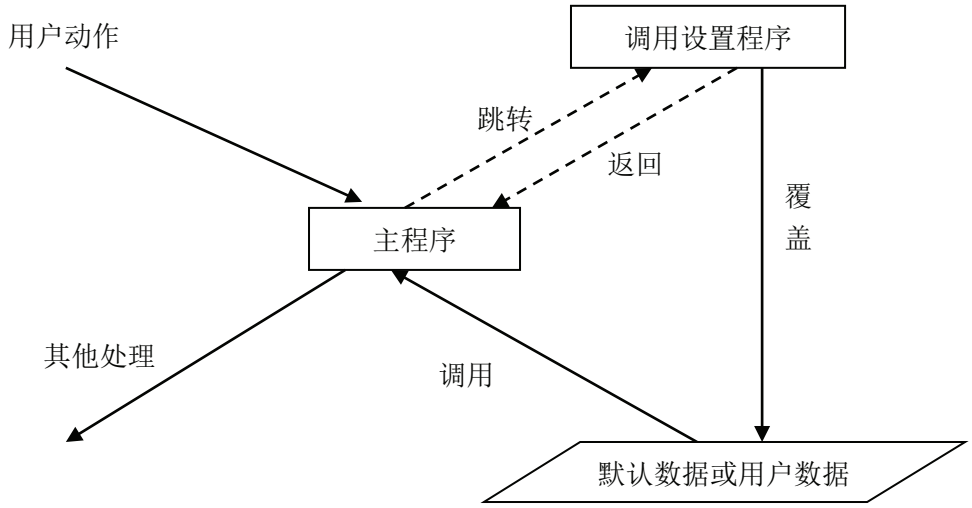

图 3.9: 主程序与设置程序关系图

图中虚线部分为可选部分，即若用户设置信息则调用，不设置则按默认运行。主程序与设置程序之间主要是完成数据的覆盖，用户数据与默认数据使用同一存储区。

3.2.1　设置程序

设置程序继承于 Activity 类。Activity 类是 Android 中最基本的一个类，有其固定的生命周期。如图 3.10 所示为设置程序类关系图。

以上类 Button、TextView、Animation、EditText、Spinner 都是继承于 view 的类。其中 Button 类用于实例化产生按钮，TextView 用于实例化产生文本显示区，Animation 用于实例化产生动态效果，EditText 用于实例化产生文本编辑区，Spinner 用于实例化产生单选按钮。Settings 类成员 `step_distance_setting`、`body_weight_setting`、`select_gender`、`select_move_mode`、`select_sensitivity` 是用户设置的数据，这些数据将会覆盖程序默认数据。若用户直接运行程序，则按默认数据来执行。并且，这些数据和主程序中声明的变量共用同一个存储区域。

3.2.2　主程序

任何一个 Android 应用程序都会有一个主程序来负责整个程序的协调运作，同样，在计步器软件中，主程序负责与设置程序交换 (覆盖) 数据、执行计步功能、程序的启动与退出等。

主程序中由于涉及到传感器，因此必须使用接口 SensorListener 来使程序能够读取传感器信息。另外，主程序可以设计为 Activity、也可以设计为 Service 的形式，两者的区别在于前者不可以后台运行，后者可以后台运行。在这里我们为了方便程序的调试，也将其设计为 Activity 的形式。

图 3.11 为主程序类关系图。主程序和设置程序有着相似之处，主程序中也包含着类似图 3.10 中的结构，但没有完全画出来。由于主程序是通过接口 SensorListener 得到，因此程序中必须含有 `onAccuracyChanged (int sensor, int accuracy)` 和 `onSensorChanged (int sensor, float[] values)` 函数，分别用于当要求传感器精度改变时和当传感器的数值发生变化时。这两个函数自动生成，程序员只需修改其内部实现代码即可。

程序何时根据传感器数值计步，我们可以将其划分在 `onSensorChanged(int sensor, float[] values)` 函数里。当传感器的数值随时间变化时，计步程序即可检测到运动的步数。关于如何依据传感器数值计步将在第 5 章进行讨论，其他数值信息都是依据步数计算出来，将在第 6 章进行讨论。

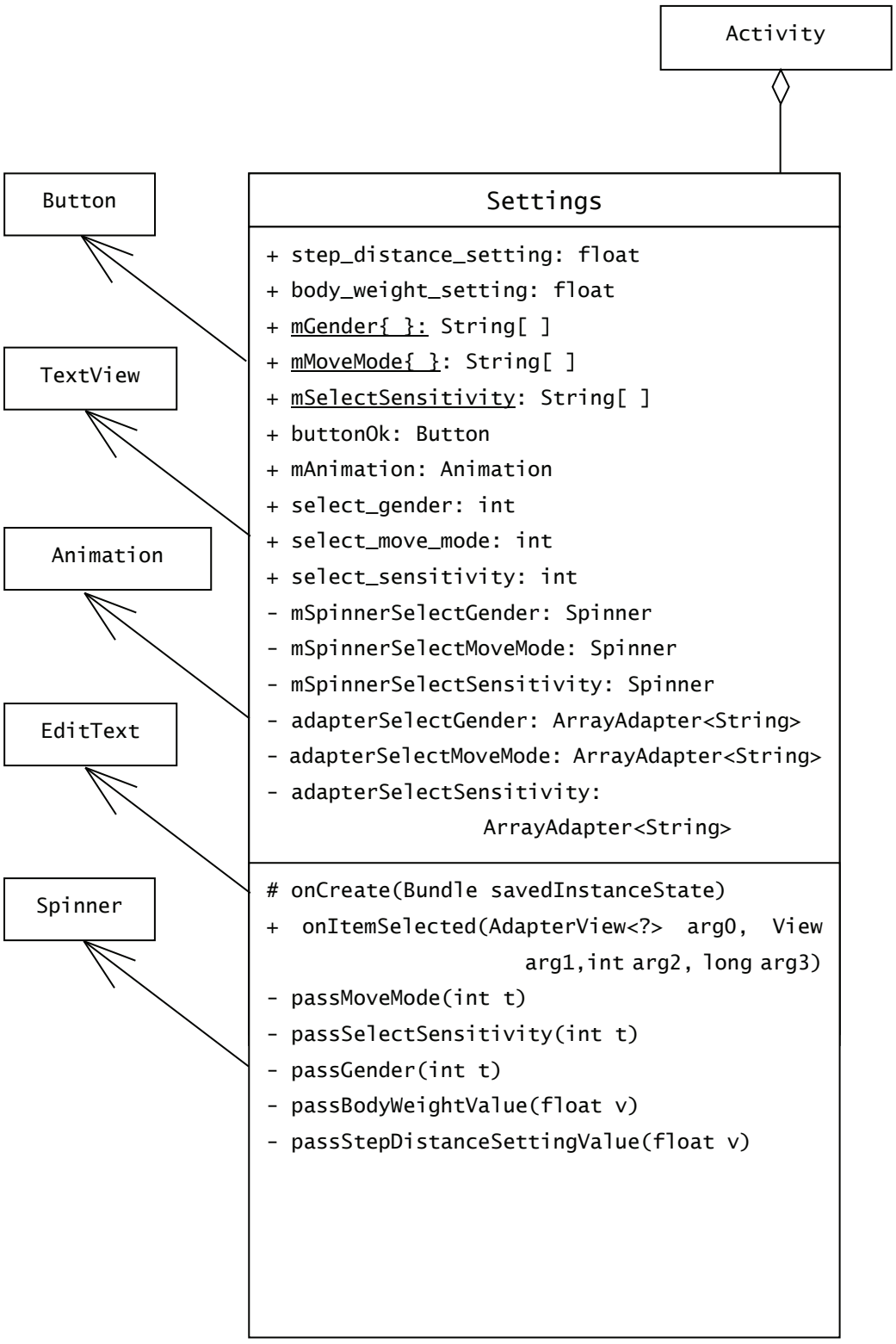

图 3.10: 设置程序类关系图

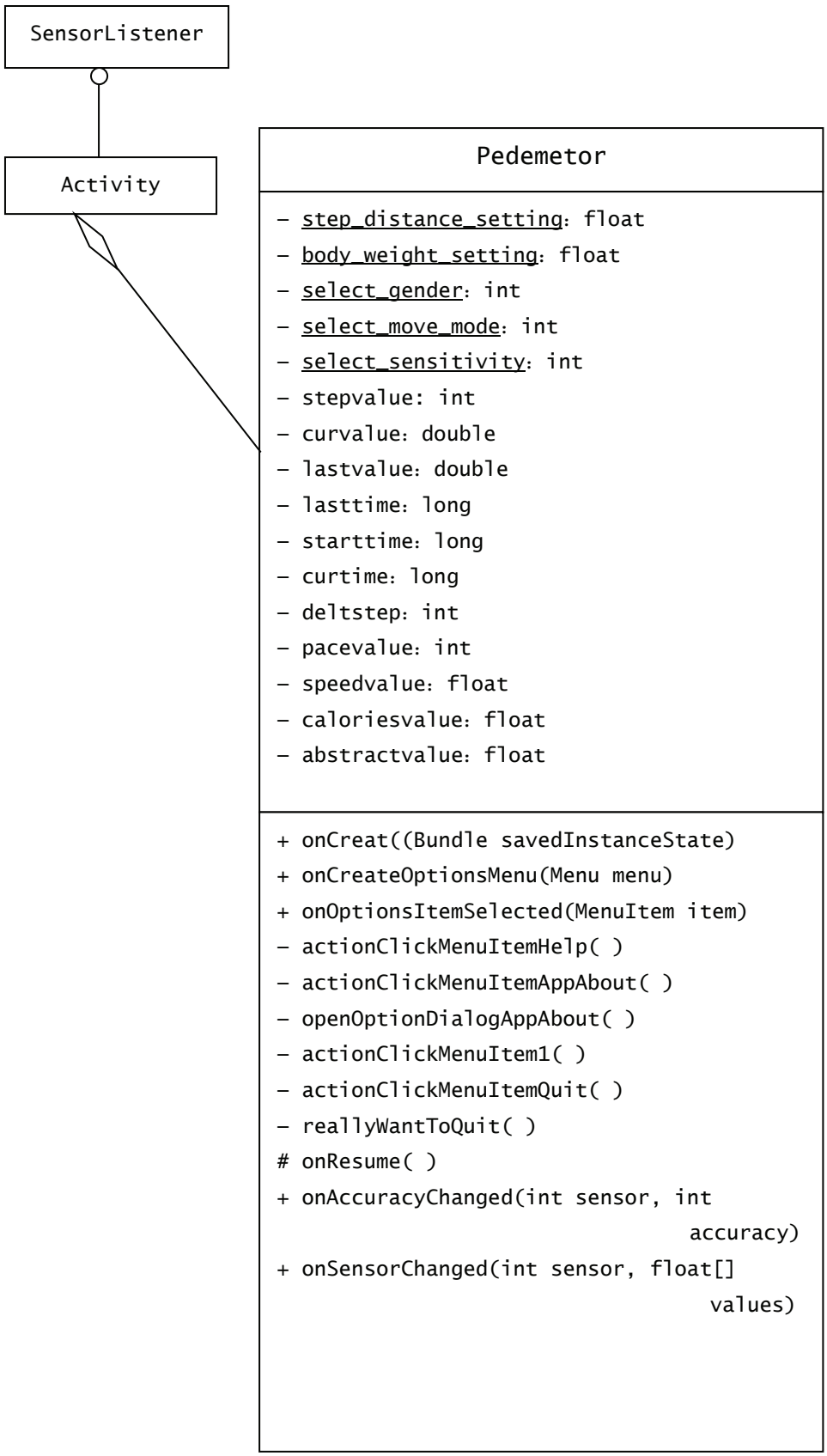

图 3.11: 主程序类关系图

第 4 章　计步原理

4.1　算法分析

一般专用计步器是直接由硬件平台实现，同样，手机计步器软件的实现是根据手机内置加速度计来感应手机的运动情况，并且能够根据加速度的数据，通过特定的算法来计步。为此，我们设计一个小测试程序来记录手机传感器的数据。图 4.1 所示为当手机震动 8 次加速度计某一参数所记录下的数据经 Matlab 分析所画图形，横坐标表示时间点数，纵坐标代表不同时刻的值 (由于离散图形不易于观察，现将其画成模拟形式)。

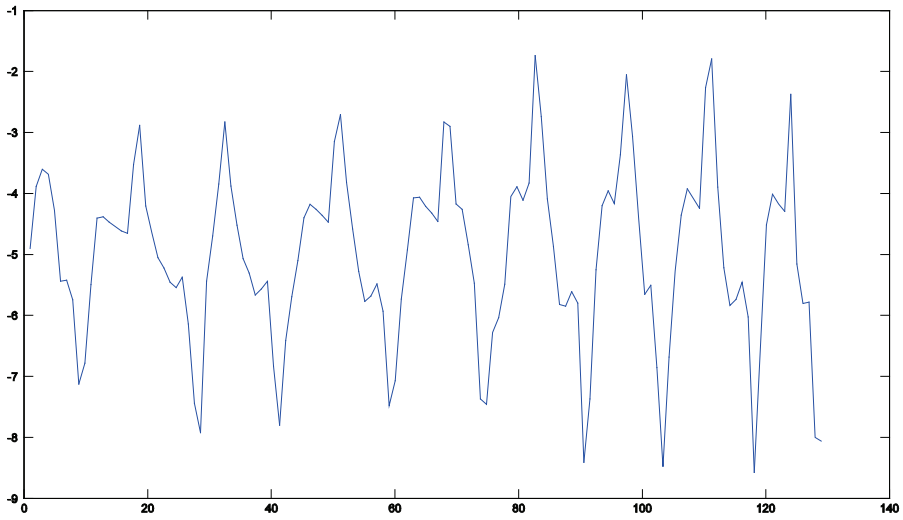

图 4.1: 真机传感器数据分析

可以看出，手机在震动时呈现出这样一个特点：每震动一下，其波形将会相应的波动一下，出现一个相应的尖峰 (谷)。为此，我们设想了以下几种计步算法及相应的优缺点：

- 峰 (谷) 值检测法

峰 (谷) 值检测法的意义是通过程序分析数据，将数据存进数组 array 里，每当检测到 `array[i-1] < array[i]`，并且当 `array[i] > array[i+1]` 时，则在第 i 个数据处为峰值，此时步数加 1。这种方法在实现时计算量相对较大，增加了手机运行时的负

荷，并且运行时内存开销也较大，可见这种方法是不经济的。另外，从图 4.1 可以看出，真机在震动时，每震动一下，出现的尖峰不止一个，这实际上增大了计步的步数，为计步带来误差。

- 数值放大法

由于不同时刻的数值较为接近，如果将这些数值放大 (如通过 log 函数)，再作差值，当差值大于某一固定数值时步数加 1。通过测试，这种方法可以相对精确地计步，但由于涉及到较为复杂的计算函数，系统开销同样较大。

- 波形整形法

这种方法类似于硬件电路中的波形整形。若用硬件平台实现，则可以将这些数值通过比较器，当数值大于阈值时，为高电平，低于阈值时则为低电平，再检测上升沿即可计步，其硬件实现等效电路如图 4.2 所示。在软件实现的情况下，同样可以参照此方法，虚线框内为软件实现等效，即设定一阈值，当传感器的数值大于阈值时，设定一整数 i 为 1，否则为 0。可以设想，当某一序列数值经“整形”后 i 为 1111000111100111 0，只需检测到由 0 跳变为 1 即可计为 1 步。这种方法实现时只需两个变量即可，减少了手机运行时占用的内存，并且没有涉及到复杂的函数，可见这种方法是较为经济的。

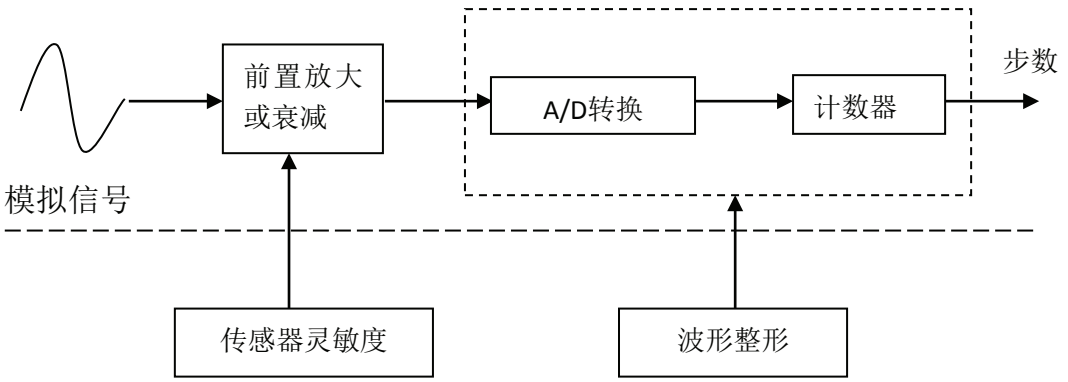

图 4.2: 硬件电路实现计数功能等效框图

通过上面的介绍，我们选择波形整形法。这种方法还有另外一个优点，用户可以设定不同的阈值来改变运动模式 (或传感器的灵敏度)。采用波形整形法，将阈值设为 -5，则波形整形为图 4.3 示。

从图 4.3 可以看出，每当程序检测到一个上升沿时，程序计为 1 步，精确度可以得到保证。

如果将阈值设为 -4，则图 4.1 转化后的波形如图 4.4 示。

比较图 4.3 和图 4.4，可以看出，当阈值设定为 -4 时，可能发生漏检。而当阈值设定为过大或过小时，又增大了检测的步数。这就要求必须设定合理的阈值。当要求灵敏度较高时，可以适当降低阈值，相反，当灵敏度要求较低时，可以适当增大阈值标准。在设计程序的过程中，为方便测试程序逻辑的正确性，采用了模拟传感器，两者

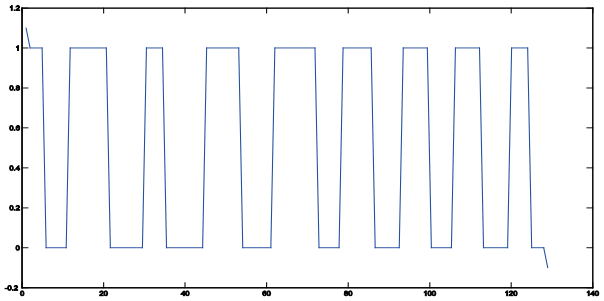

图 4.3: 阈值设为 -5 时整形后波形

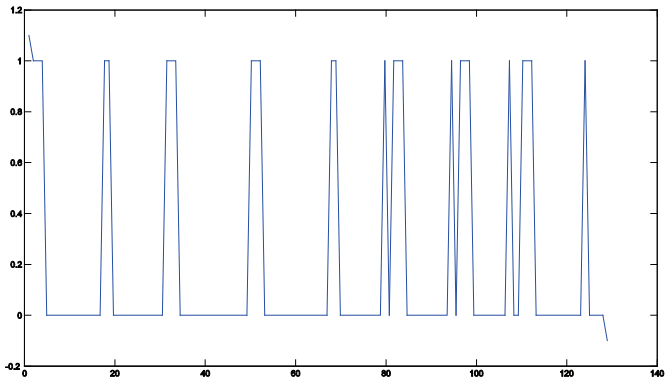

图 4.4: 阈值设为 -4 时整形后波形

的区别仅是阈值的设置不同，而程序内部实现机理是一样的。

4.2　传感器类型的选择

传感器的数据有三种，分别为加速度计 (accelerometer)，磁场 (magnetic field) 和方向 (orientation)。在设计计步器程序时，为了简化程序设计而又不失去实际意义，选择利用加速度计的数值来分析，而忽略磁场和方向的数值。这是因为在一特定区域，磁场和方向的变化不会影响计步效果。

传感器的每一种类型都有三个参量，为了能够获得较为精确的计步数据，现考虑加速度的三个参数。从加速度的三个参数波形可以看出，它们每震动 1 次，都会出现对应的峰值。理论上讲，检测出一次震动周期中的最大值时，计步加 1，即可以正确得出步数。

为了能够较为精确的计步数据，可以采用以下几种方式：

1. 直接法。比较以上三个参数的波形，可以发现，每当手机震动 1 次，三个参数都相应地出现了峰值，用前面介绍的波形整形法，设定好合适的阈值，即可得到较为精确的计步数值。这种方式实现起来简单，没有涉及到复杂的运算，较为经济。
2. 间接法。由于加速度是一个矢量，如图 4.5 示，其中原点即为手机所在位置。单

一地考虑一个参数可能会对计步带来误差，现将这三个参数的数值经过运算计算出加速度模值大小，再通过波形整形得到更可靠的加速度信息。

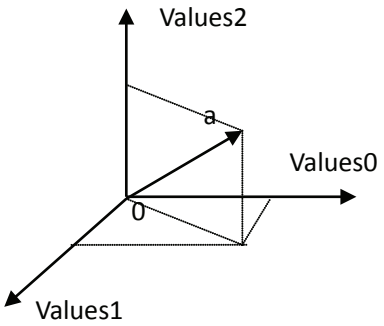

图 4.5: 加速度矢量示意图

虽然直接法的使用较为简单，没有涉及到复杂的数据运算，但由于手机在使用过程中的震动时十分复杂的，单一地采用分析加速度某一个参数显然是不合适的。间接法的使用虽然较为复杂，手机的运算量也较大，但这样可以提高计步的精确度。因此，我们在程序设计中采用的是间接法，提高了程序运行的可靠性。

第 5 章　其他数据设计与算法

计步器程序不光要提供用户运动时的步数，而且要能够提供其他一些相应的信息，如里程、步速、时速以及所消耗的热量。用户可以参考这些这些数据来实时调整运动情况，以达到最佳锻炼效果。

根据上一章的分析，我们已经可以得到较为精确的计步数据，在计步数据已知的情况下，就可以较为方便地计算出我们所感兴趣的信息。

5.1　里程计算

里程计算框图如图 5.1 所示。

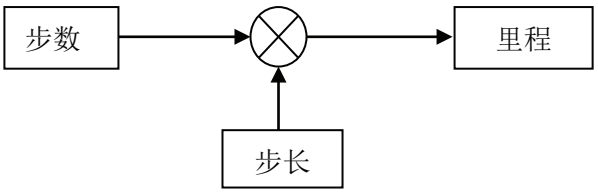

图 5.1: 里程计算框图

其中步长是由用户设置，其默认值为 50 cm。通过步数与步长的乘积即可得出大致的里程。

5.2　步速、时速的计算

步速和时速的计算涉及到手机系统时间 (格式为：时. 分. 秒. 毫秒)，即由单位时间内所运动的步数和里程得出。如图 5.2 所示，为步速、时速设计框图。框图设计的总体思想是：通过得出的手机系统时间，计算出相隔固定时间 T 内所运动的步数差值 ΔS，则 $\Delta S/T$ 即为步速，再由步长即可得出时速。

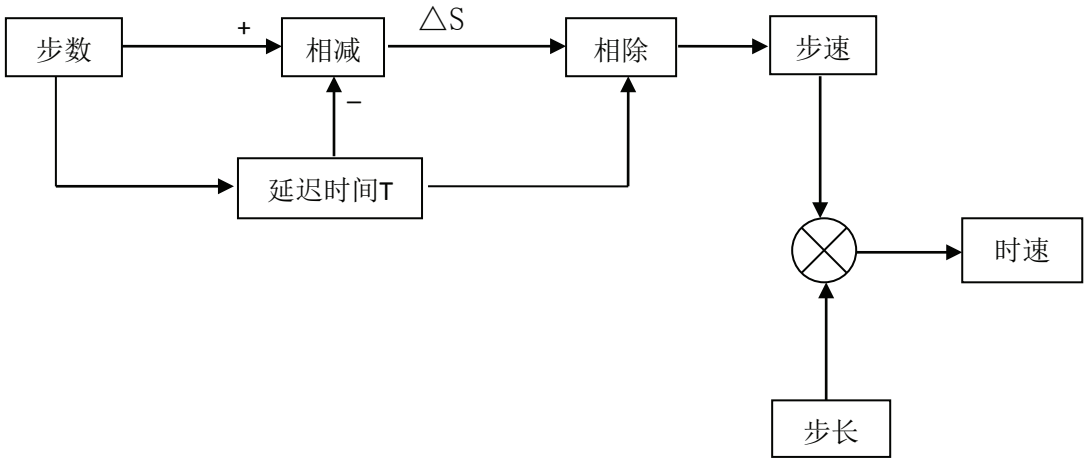

图 5.2: 步速、时速设计框图

5.3　热量计算

热量的计算是非常复杂的，且没有固定公式，程序中只能估计运动所消耗的热量。以下给出几种运动消耗卡路里的计算方式 (参考网页：http://www.boohee.com/assessment/activity_calory)：

(1) 已知体重 m 千克、时间 t 小时，速度 v (分钟/400 米)，则跑步消耗热量 W(kcal) 为：

$W = m \times t \times k$

式中，k 为运动指数，$k = 30 \div v$。

(2) 已知体重 m 千克、距离 s 千米，则

$W = m \times s \times 1.036$

(3) 已知体重 m 千克、速度和时间 (分钟)，则

$W = m \times t \times k$

式中，k 为运动指数，其具体数值见表 5.1

表 5.1: 运动指数 k 与速度的关系

运动速度	每小时 8 千米	每小时 12 千米	每小时 15 千米
k	0.1355	0.1797	0.1875

需要指出的是，以上公式都很粗略，因为它们忽视了年龄、性别、体质和基础代谢率等因素，只是给大家提供一个参考。在计步器程序设计中采用第一种计算方法，其具体设计框图如图 5.3 所示。

图中运动指数 k 的数值经方法 (1) 换算成 1.25× 时速 (千米/小时)，时速的数值已由第 5.2 节中介绍的方法求得。

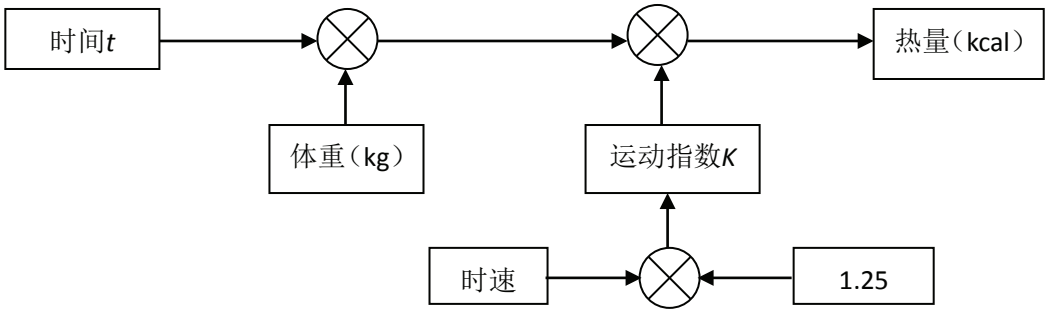

图 5.3: 运动消耗热量计算框图

通过以上各种算法，将步数作为中间变量间接求出其他信息。因此，必须在计步数据相对精确的情况下才有实际参考意义。另外，由于程序中设计的延时参数 T 较大(若 T 较小，则难以捕捉到短时间内的步数，有可能造成一段时间内速度始终为 0 的情况)，界面显示有一定的延时，但不会影响程序的正确运行。

第 6 章　程序测试

6.1　权限设置

一个应用程序可由多个 Activity 组成，其中至少有一个是主 Activity。当 Activity 在 AndroidManifest 文件的 <Activity> 标签中进行了声明时，应用程序才有权限来启动相应的 Activity，其他的应用程序为了能够启动这个 Activity，需要在自己的 AndroidManifest 资源文件中的 <uses-permission> 元素中进行声明。

6.2　程序安装

计步器程序调试完毕后即可以直接安装，安装文件 `*.apk` 在 `\bin` 目录下，直接拷贝到 Android 手机安装即可。该软件测试是在 Android 2.1 平台上进行的。

6.3　测试结果

以下是程序在 Google Android2.1 平台手机上测试结果，见表 6.1，数据仅供参考。

在记录过程中，当运动速度较快时，记录的精确度比较可靠。而当运动较慢时 (如散步) 时，有时运动却不计步，有时运动 1 步计为 2 步，存在较大的误差，如图 6.1 所示。

这是因为当运动较慢时，加速度值非常小。通过观察，当慢速运动时，其波形犹如随机噪声，运行 1 步的波形波峰不是很明显，甚至会出现好多波峰，这对步数检测带来一定的困难。因此，该计步器程序用于慢速运动时还需要设计改变传感器灵敏度。

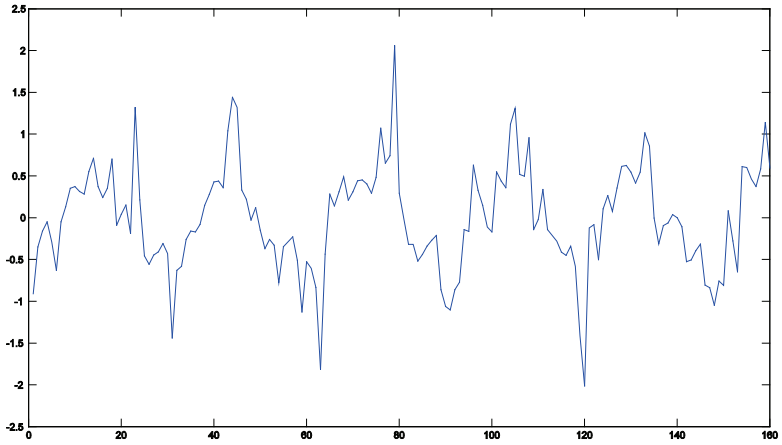

图 6.1: 极慢速运动时加速度计模值图

表 6.1: 计步器软件在 Google Android2.1 平台上运行结果

测试项目 / 测试结果 / 运行参数		实际运行步数	计步器记录运动步数	计步器记录里程(米)	计步器记录消耗热量(卡路里)
步长：63cm 体重：75kg 灵敏度：中	快跑	130	133	83.79	6510.483
		91	98	61.74	4797.198
		50	54	34.02	2643.354
	散步	30	43	27.09	2104.893
		36	34	21.42	1664.334
步长：56cm 体重：55kg 灵敏度：中	快跑	60	61	34.16	1946.4369
		50	50	28	1595.4
	慢跑	22	16	8.96	510.541
		34	37	20.72	1180.626

第 7 章　结论

基于 Google Android 智能手机的软件的开发过程中借助 Android SDK (模拟器) 来编译源程序，在验证程序正确性过程中起到了很大的帮助作用。

计步器软件的设计完成了最初设定的目标：

- 提供友好程序界面，并提供用户与手机交互信息的接口；
- 通过计步数据得出其他用户感兴趣的信息，如里程、速度、消耗热量等；
- 实现软件在 Android 智能手机上无 bug 运行，测试并记录数据。
- 通过分析真机传感器的数据，能够提供较为精确的计步算法；

第 4 个目标的实现得益于传感器数据经仿真，使得计步精确地得到了很大的提高。并且从理论与实际上验证了计步算法的精确性。该软件在 Android 手机上可以实现无 Bug 运行，并提供了用户使用手册。

由于该计步器软件使用的是 Activity 来实现的，这就意味着程序不能后台运行。因此今后的改进方向是将程序设计为 Service，以提高软件使用的方便性。

参考文献

[1] 姚尚朗, 靳岩. Google Android 开发入门与实战 [M]. 北京 : 人民邮电出版社, 2009.

[2] 余志龙, 陈昱勋, 郑名杰, 等. Google Android 开发入门与实战 [M]. 北京 : 人民邮电出版社, 2009.

[3] DEITEL H M, DEITEL P J. Java: How to Program[M]. 5th ed. [S.l.] : Prentice Hall, Inc., 2002.

[4] GOOGLE. Android[EB/OL]. Google Inc., 2010. http://www.android.com/.

[5] eoe Android 社区 [EB/OL]. 2010. http://www.eoeandroid.com/.

[6] GOOGLE. API Guides - Android Developers[EB/OL]. Google Inc., 2010. https://developer.android.com/guide/index.html.

[7] HASEMAN C. Books for Professionals by Professionals : Android Essentials[M]. 1st ed. [S.l.] : Apress, 2008.

[8] DIMARZIO J F. Android A Programmer’s Guide[M]. 1st ed. [S.l.] : McGraw-Hill Education, 2008.

附录 A　主程序源代码

```
////////////////////////////Pedemetor.java//////////////////////////
package xu.pedemeter;

import android.app.Activity;
import android.app.AlertDialog;
//import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;
import android.app.Activity;
import android.hardware.SensorListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
import android.os.Bundle;
import java.lang.Math.*;
import java.text.DecimalFormat;
import java.util.Calendar;
import android.app.*;
import android.content.*;
import android.view.Menu;
import android.view.MenuItem;
import android.content.Intent;
import android.view.View.OnClickListener;

public class pedemeter extends Activity implements SensorListener{
    private SensorManagerSimulator mSensorManager;//用于模拟机测试
    //private SensorManager mSensorManager;//用于真机测试
    private int stepvalue=0;
    private double curvalue;
    private double lastvalue;
    private long lasttime=System.currentTimeMillis();
    private long starttime=System.currentTimeMillis();
    private long curtime;
    static float step_distance_setting = (float)0.5;//50cm
    static float body_weight_setting = (float)50;//50kg
```

```
    static int select_gender=0;
    static int select_move_mode=0;
    static int select_sensitivity=2;
    private int i,j;
    private int deltstep=0;
    private int pacevalue;
    private float speedvalue;
    private float caloriesvalue=0;
    float abstractvalue;
    DecimalFormat formatnum = new DecimalFormat("##0.00");
    TextView mStepValue;
    TextView mDistanceValue;
    TextView mPaceValue;
    TextView mSpeedValue;
    TextView mCaloriesValue;
    public static final int SETTINGS = Menu.FIRST;
    public static final int APPABOUT = Menu.FIRST + 1;
    public static final int QUIT = Menu.FIRST + 2;
    public static final int HELP = Menu.FIRST + 3;
    Calendar mCalendar = Calendar.getInstance();
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //time = System.nanoTime();
        mStepValue = (TextView) findViewById(R.id.step_value);
        mDistanceValue = (TextView) findViewById(R.id.distance_value);
        mPaceValue =(TextView)findViewById(R.id.pace_value);
        mSpeedValue = (TextView) findViewById(R.id.speed_value);
        mCaloriesValue = (TextView) findViewById(R.id.calories_value);
        mSensorManager=SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
        mSensorManager.connectSimulator();
//以上用于模拟机测试
//mSensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        //用于真机
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu){
        super.onCreateOptionsMenu(menu);
        //添加设置窗口
        menu.add(0,SETTINGS,0," 设置")
        .setIcon(android.R.drawable.ic_menu_preferences)
```

```
        .setShortcut('2', 'r');
        //添加“关于”菜单
        menu.add(0,APPABOUT,1," 关于")
        .setIcon(android.R.drawable.ic_menu_info_details)
        .setShortcut('2', 'r');
        //添加帮助菜单
        menu.add(0,HELP,1," 帮助")
        .setIcon(android.R.drawable.ic_menu_help)
        .setShortcut('2', 'r');
        //添加退出菜单
        menu.add(0,QUIT,1," 退出")
        .setIcon(android.R.drawable.ic_lock_power_off)
        .setShortcut('2', 'r');

        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item){
        super.onOptionsItemSelected(item);
        //Intent intent = new Intent();
        switch(item.getItemId()){
        case SETTINGS:
            finish();
            Intent intent=new Intent(pedemeter.this,settings.class);//跳转至设置界面
            startActivity(intent);
            break;
        case QUIT:
            actionClickMenuItemQuit();
            break;
        case APPABOUT:
            actionClickMenuItemAppAbout();
            break;
        case HELP:
            actionClickMenuItemHelp();
            break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void actionClickMenuItemHelp() {
        // TODO Auto-generated method stub
        openOptionDialogHelp();
    }
```

```
    private void openOptionDialogHelp() {
        // TODO Auto-generated method stub
        new AlertDialog.Builder(this)
        .setTitle(R.string.help)
        .setIcon(android.R.drawable.ic_menu_help)
        .setMessage(R.string.help_msg)
        .setPositiveButton(R.string.str_ok,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                    }
                }).show();
    }
    private void actionClickMenuItemAppAbout() {
        // TODO Auto-generated method stub
        openOptionDialogAppAbout();
    }
    private void openOptionDialogAppAbout() {
        // TODO Auto-generated method stub
        new AlertDialog.Builder(this)
        .setTitle(R.string.app_about)
        .setMessage(R.string.app_about_msg)
        .setPositiveButton(R.string.str_ok,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                    }
                }).show();
    }
    private void actionClickMenuItem1() {
        // TODO Auto-generated method stub
        setTitle(" 设置用户参数");
    }
    private void actionClickMenuItemQuit() {
        // TODO Auto-generated method stub
        reallyWantToQuit();
    }
    private void reallyWantToQuit() {
        // TODO Auto-generated method stub
        new AlertDialog.Builder(this)
        .setTitle(" 友情提示!")
```

```
        .setIcon(R.drawable.warning)
        .setMessage(" 确定要退出吗?")
        .setPositiveButton(R.string.str_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                        finish();
                        System.exit(0);
                    }
                })
                .setNegativeButton(R.string.str_cancel,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                // TODO Auto-generated method stub
                            }
                        }).show();
    }
    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, SensorManager.SENSOR_ACCELEROMETER
                | SensorManager.SENSOR_MAGNETIC_FIELD
                | SensorManager.SENSOR_ORIENTATION,
                SensorManager.SENSOR_DELAY_FASTEST);
    }//注册传感器
    @Override
    protected void onStop() {
        mSensorManager.unregisterListener(this);
        super.onStop();
    }//注销传感器
    public void onAccuracyChanged(int sensor, int accuracy) {
    }//当灵敏度调整时
    public void onSensorChanged(int sensor, float[] values) {
        j=i;
        switch(sensor){
        case SensorManager.SENSOR_ACCELEROMETER:
            //以下程序计步
            abstractvalue=values[0]*values[0]+values[1]*values[1]+values[2]*values[2];
            abstractvalue=(float)Math.sqrt(abstractvalue);
            if(select_move_mode==0){
                if(abstractvalue>10){i=1;}
                else{i=0;}
```

```
            }
            if(select_move_mode==1|select_move_mode==2){
                if(abstractvalue>10.4){i=1;}
                else{i=0;}
            }
            if(j-i==1){stepvalue++;deltstep++;}
            mStepValue.setText(""+stepvalue);
            CalDistance(stepvalue);
            CalStepSpeed();
            CalDistanceSpeed();
            CalCalories();
            break;

        }
    }
    private void CalCalories() {
        // TODO Auto-generated method stub
        caloriesvalue=(float)(body_weight_setting*stepvalue*step_distance_setting*1.036);
        mCaloriesValue.setText(""+caloriesvalue);
    }
    private void CalDistanceSpeed() {
        // TODO Auto-generated method stub
        speedvalue=pacevalue*step_distance_setting;
        mSpeedValue.setText(""+speedvalue);
    }
    private void CalStepSpeed() {
        // TODO Auto-generated method stub
        curtime=System.currentTimeMillis();
        if(curtime-lasttime>1500){
            pacevalue=(int)(deltstep*60000/(curtime-lasttime));
            mPaceValue.setText(""+pacevalue);
            lasttime=System.currentTimeMillis();
            deltstep=0;
        }

    }
    private void CalDistance(int stepvalue) {
        // TODO Auto-generated method stub
        float distance=stepvalue*step_distance_setting;
        mDistanceValue.setText(""+formatnum.format(distance));
    }
}
```

附录 B　Animation 自定义文件

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate
  android:fromXDelta="0"
  android:toXDelta="-100%p"
  android:duration="300"
  >
  </translate>
  <alpha
  android:fromAlpha="1.0"
  android:toAlpha="0.0"
  android:duration="300">
  </alpha>
</set>
```