**​Android系统智能电视HAL层Sensor数据传输的一种实现**

胡济豪[[1]](#footnote-1)

（上海交通大学软件学院，上海 1130379118）

**摘要**：随着智能平板电视技术的飞速发展，智能电视的交互设计也在不断改进，其中电视体感交互就是其中的一个方向。当前许多厂商推出的智能电视都不约而同地选择Android系统。本文将讨论如何设计并实现Android系统下HAL层获取外置电视体感遥控器内传感器数据及手机第二屏虚拟传感器的数据。

**键词**：智能电视，Android系统，HAL，传感器

**THE IMPLEMENTION OF SENSOR HAL IN ANDROID SYSTEM**

Hu Jihao

(School of Software, Shanghai Jiaotong University, Shanghai 1130379118)

**Abstract**: With the rapid development of technology in smart flat-panel TV, smart TV interaction design has been improved steadily. For an Example，The TV somatosensory interaction is one area which is worthy of research. This article will introduces the design and implement of Sensor HAL in Android system to get real sensor data from real devices and virtual sensor data from a mobile phone.

**Keywords**: Smart TV, Android system, HAL, Sensor

**1.引言**

智能电视如今已经成为家电市场上的一个焦点，许多厂商推出的智能电视一般选择的都是Android操作系统，当然这更多的是因为Android的开放特性。在智能电视领域，随着其功能的日渐丰富和强大，人们对电视的使用不仅仅限于接收和观看电视节目。现在，中高端的智能电视一般都支持了体感游戏的功能，比如厂商TCL、海信、创维，等等都在自己的智能电视平台上集成了体感游戏这些功能模块。还有，现在广电总局推出的TVOS2.0系统也把人机交互作为一个卖点推出。市场也已经出了不少的体感游戏，比如1905互动出品的赛车类游戏、球类运动游戏、等等，在智能电视上通过外设体感设备进行电视版体感游戏的操控已经成为一种潮流。这些游戏的操控一般是通过集成了传感器芯片的外设遥控器进行完成。Android系统在framework层已经有了比较完备的获取传感器数据的实现，针对各个不同的厂商，他们要做的就是基于自己的硬件设备来完成HAL层的实现，这样传感器的数据就能传送给上层应用。本文的目标在于讨论如何设计和实现Android系统的HAL层，读取遥控器中的传感器数据以及虚拟传感器数据。

**2.Android系统Sensor数据传输的框架分析**

**2.1Android系统整体架构**

Android的系统架构和其他操作系统类似，采用了分层的架构。从架构图（图2.1.1）来看，Android分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和Linux核心层[1]。

(1) 应用程序

所有的应用程序都是使用java语言编写的，每一个应用程序由一个或者多个活动组成，

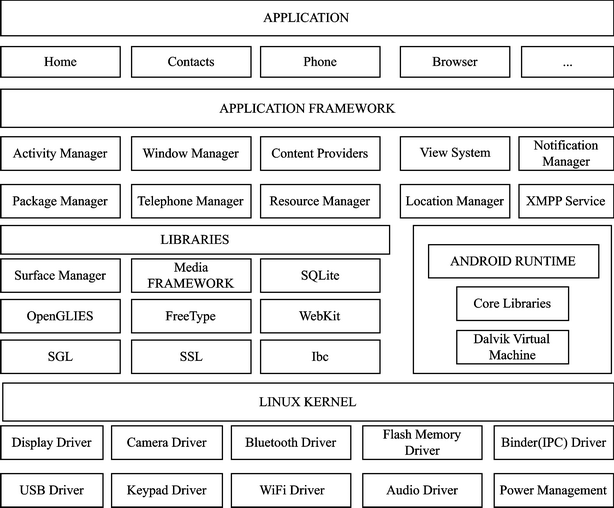


图2.1.1 Android系统整体架构图

活动必须以Activity类为超类，活动类似于操作系统上的进程，但是活动比操作系统的进程要更为灵活。与进程类似的是，活动在多种状态之间切换。利用java的跨平台特性，基于Android框架开发的应用程序可以不用编译运行于任何一台安装有Android系统的平台，这点正是Android的精髓所在。

(2) 应用程序框架

应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且其他任何应用程序都可以使用其所发布的功能模块(遵循框架的安全性限制)，帮助程序员快速的开发程序，并且该应用程序重用机制也使用户可以方便的替换程序组件。隐藏在每个应用后面的是一系列的服务和系统，其中包括：丰富而又可扩展的视图（Views），内容提供器（Content Providers），资源管理器（Resource Manager），通知管理器（Notification Manager），活动管理器（Activity Manager），等。

(3) 系统运行库

Android系统运行库包括程序库（Libraries）和运行时库（Android Runtime）。Android包含一些C/C++库，这些库能被Android系统中不同的组件使用。他们通过Android应用程序框架为开发者提供服务。其核心库主要包括Bionic系统C库、媒体库、Surface Manager、基于WebKit的浏览器、SGL 2D图形引擎、3D libraries、FreeType位图和矢量字体显示、功能强劲的轻型关系型数据库引擎SQLite，等等。

值得注意的是，这里还有一个硬件抽象层，即所谓的HAL。其实Android并非所有的设备驱动都放在linux内核里面，有一部分实现在用户空间，这么做的主要原因是可以避开Linux所遵循的GPL协议，一般情况下如果要将Android系统移植到其他硬件去运行，只需要实现这部分代码即可[2]。包括：显示器驱动，声音，相机，GPS，GSM，Sensor等等。在后边的章节中要讨论的Sensor HAL层的实现变是在这一层。

Android 包括了一个核心库，该核心库提供了JAVA编程语言核心库的大多数功能。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。Dalvik被设计成一个设备可以同时高效地运行多个虚拟系统。 Dalvik虚拟机执行(.dex)的Dalvik可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由JAVA编译器编译，然后通过SDK中的工具转化成.dex格式由虚拟机执行。Dalvik虚拟机依赖于linux内核的一些功能，比如线程机制和底层内存管理机制。

(4) Linux内核

Android的核心系统服务依赖于Linux内核，如安全性，内存管理，进程管理，网络协议栈和驱动模型。Linux内核也同时作为硬件和软件栈之间的抽象层。其还对其做了部分修改，主要设计两部分修改：另外实现了一套进程间通讯机制Binder和为手持设别的电源能耗做了较好的电源管理[3]。

**2.2Android Sensor架构**

现在的Android系统内置对传感器的支持已经很多，常见的是：加速度传感器（accelerometer）、磁力传感器（magnetic field）、方向传感器（orientation）、陀螺仪（gyroscope）、环境光照传感器（light）、压力传感器（pressure）、温度传感器（temperature）和距离传感器（proximity）等等[4]。Android实现传感器系统包括以下几个部分（如表2.2.1所示）：

|  |  |  |
| --- | --- | --- |
| 类别 | 名称 | 代码 |
| 用户空间 | Java应用程序 | 用户实现 |
| Java framework框架层 | SensorManager.java SensorListener.java SensorEvent.java … |
| JNI层 | android\_hardware\_SensorManager.cpp com\_android\_server\_SensorService.cpp |
| HAL硬件抽象层 | 用户实现(sensor.c) |
| 内核空间 | 设备驱动程序 | 用户实现 |
| 具体硬件 | 加速度传感器、陀螺仪传感器、压力传感器，等等 | 用户实现 |

表2.2.1 Android Sensor代码空间划分

Java应用程序指的是最上层的app，比如一些体感游戏等等，一般由第三方公司开发，运行于Android平台，通过SDK提供的framework层Java接口和系统交互；Java framework框架层为app提供了所需要的系统接口调用，app可以通过这些接口获取所支持的Sensor列表，以及实时的Sensor数据，比如获取系统服务getSystemService、获得传感器对象getDefaultSensor, 注册监听函数registerListener；JNI是Java程序调用C/C++接口的衔接部分；HAL硬件抽象层屏蔽了Android系统对驱动细节的具体的依赖。设备驱动程序和硬件部分由不同的厂商根据自己的产品而定。

各部分之间的层次结构如下图（图2.2.1）所示。

图2.2.1 Android Sensor层次结构图

和Android系统相关的Sensor框架主要分为三个层次：客户端、服务端、HAL

层，这三个层次的总体调用关系图如图2.2.2所示。服务端负责从HAL读取数据，并将数据写到管道中，客户端通过管道读取服务端数据。

客户端主要类：

SensorManager.java 从Android4.1开始，把SensorManager定义为一个抽象类，定义了一些主要的方法，该类主要是应用层直接使用的类，提供给应用层的接口；

SystemSensorManager.java 继承于SensorManager，客户端消息处理的实体，应用程序通过获取其实例，并注册监听接口，获取Sensor数据；

SensorEventListener接口用于注册监听sensor事件；

SensorThread 是SystemSensorManager的内部类，开启一个新线程负责读取sensor数据，当注册了SensorEventListener接口的时候才会启动线程；

android\_hardware\_SensorManager.cpp是负责与java层通信的JNI接口；

SensorManager.cpp 是sensor在Native层的客户端，负责与服务端SensorService.cpp的通信；

SensorEventQueue.cpp是消息队列。

服务端主要类：

SensorService.cpp 服务端数据处理中心；

SensorEventConnection 从BnSensorEventConnection继承来，实现接口ISensorEventCon-

nection的一些方法，ISensorEventConnection在SensorEventQueue会保存一个指针，指向调用服务接口创建的SensorEventConnection对象；

Bittube.cpp 在这个类中创建了管道，用于服务端与客户端读写数据；

SensorDevice 负责与HAL读取数据

HAL层：

Sensor.h是google为Sensor定义的HAL接口。我们需要探讨的就是HAL接口的实现，具体的将在后边进行讨论，下边先关注一下服务端获取Sensor数据的调用时序图，如图2.2.3所示。在SystemServer进程中的main函数中，通过JNI调到com\_android\_server\_SystemServ-

er.cpp的android\_server\_SystemServer\_init1()方法，该方法又调用system\_init.cpp中的system\_init()。SensorService创建完之后，将会调用SensorService：：onFirstRef()方法，在该方法中完成初始化工作。首先获取SensorDevice实例，在其构造函数中，完成了对Sensor模块HAL初始化。在这里主要做了三个工作：调用HAL层的hw\_get\_module()方法，加载Sensor模块so文件；调用sensor.h的sensor\_open方法打开设备；调用sensor\_poll\_device\_t的activate()对Sensor模块使能。

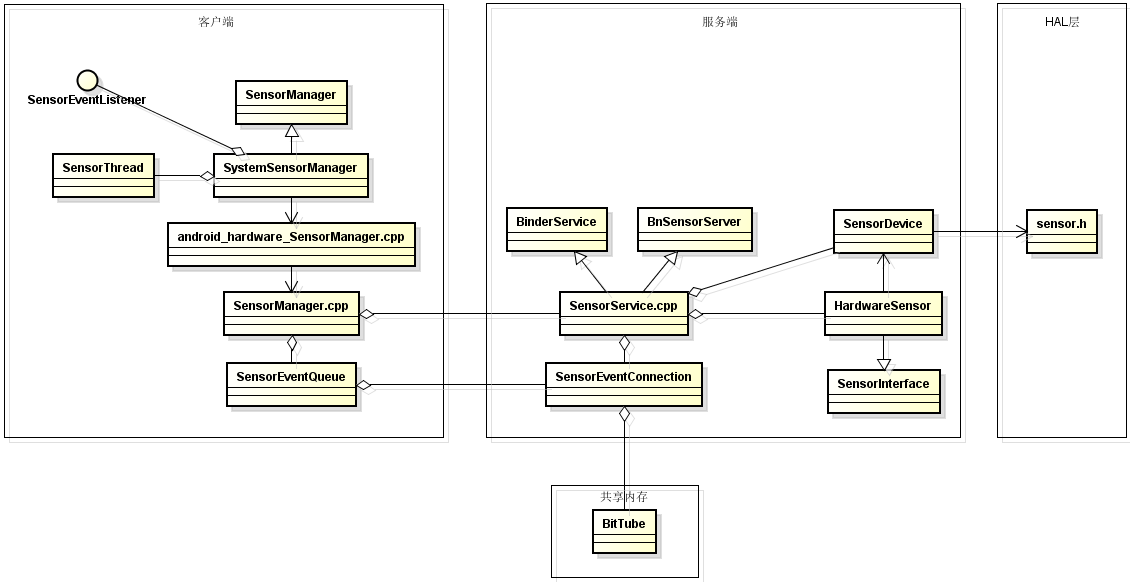


图2.2.2 Android Sensor层次类图

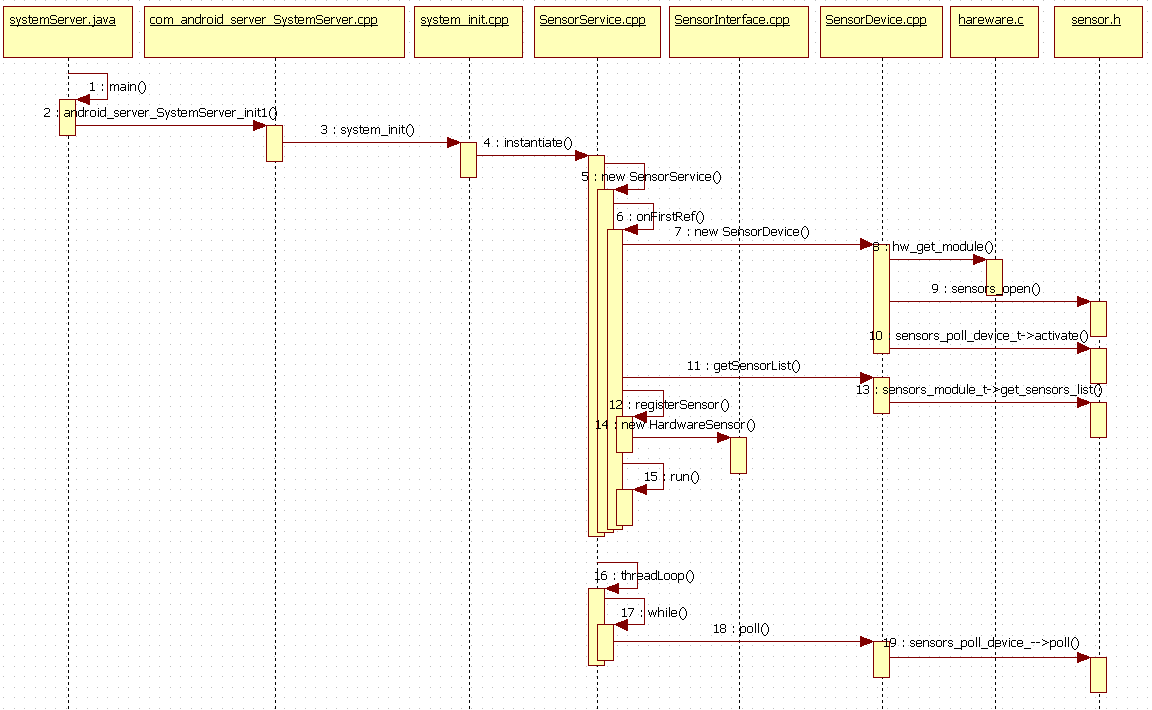


图2.2.3 Android Sensor服务端调用Sequence

**2.3 Sensor HAL分析**

HAL存在两种架构[5]：旧的HAL架构（位于libhardware\_legacy目录）和新的HAL架构（位于libhardware目录）。

过去的libhardware\_legacy，将\*.so当做shared library来使用，在runtime(JNI)部分以direct function call使用HAL module。通过直接函数调用的方式，来操作驱动程序。当然，应用程序也可以不需要透过JNI的方式进行，直接以加载动态库\*.so（dlopen）的做法来调用动态库里的符号(symbol)也是一种方式。之而言之是没有经过封装，上层可以直接操作硬件。如下图（图2.3.1所示）。

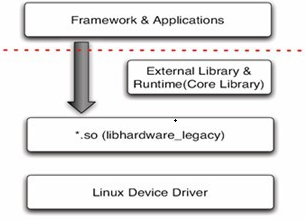


图2.3.1 旧HAL框架

现在的libhardware采用stub做法，如图2.3.2所示。HAL stub是一种代理人（proxy）的概念，stub虽然仍是以\*.so的形式存在，但HAL已经将\*.so隐藏起来了。Stub向HAL提供操作函数（operations），而runtime则是向HAL取得特定模块（stub）的operations，再callback这些函数。这种以indirect function call的框架，让HAL Stub变成是一种包含关系，即HAL里包含了许许多多的stub（代理人）。Runtime只要说明“类型”，即module ID，就可以取得操作函数。对于目前的HAL，可以认为Android定义了HAL层结构框架，通过几个接口访问硬件从而统一了调用方式。

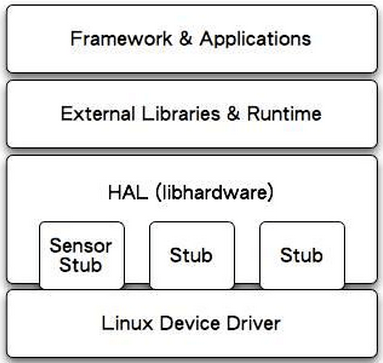


图2.3.2 新HAL框架

HAL\_legacy和HAL对比：

HAL\_legacy：旧式的HAL是一个模块，采用共享库形式，在编译时会调用到。由于采用function call形式调用，因此可被多个进程调用，但会被mapping到过个进程空间中，造成浪费，同时需要考虑代码能否安短重入的问题（thread safe）。

HAL：新式的HAL采用HAL module和HAL stub结合形式，HAL stub不是一个share library，编译时上层只拥有访问HAL stub的函数指针，并不需要HAL stub。上层通过HAL module提供的统一接口获取并操作HAL stub，so文件只会被mapping到一个进程，也不存在重复mapping和重入问题。

**3. Sensor HAL的一种实现**

通过上面章节的分析，我们对Android系统下的Sensor HAL整体架构有了比较系统的了解。下边，我们将讨论和具体需求相关的HAL层的一种实现。

从需求来看，第一是需要实现从外部硬件设备中读取真实Sensor的数据，第二是需要开辟一个通道将手机第二屏Sensor的虚拟数据传输给Android系统。

**3.1 获取真实Sensor数据**

现在市场面上有许多的体感遥控器是通过2.4G技术进行无线连接与数据传输，在主机接收端插入一个类似USB端口的接收器，遥感数据可以从此USB接收端读出，图3.1.1是此类产品的一个外观图。我们将讨论此类产品Sensor数据的读取。



图3.1.1 2.4G无线遥控器产品示意图

将此设备的USB端子插入Android系统的智能电视USB接口，在dev下可以看到多出多出hidraw设备。

我们可以通过调用系统函数read()读取hidraw的原始数据，将其转换成Android Sensor说需要的数据结构即可，同时为了更好的兼容外设体感遥控器的Sensor数据，我们定义一套标准的数据结构以进行规格适配，如图3.1.2所示。

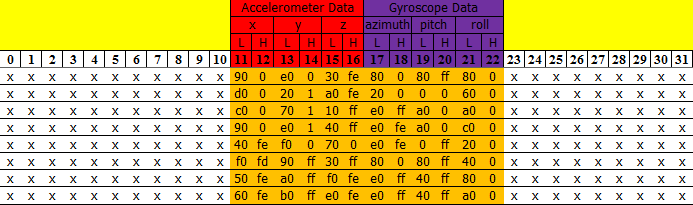


图3.1.2 Sensor数据格式

在图3.1.2中，每一行所示的是从hidraw读出的原始数据，每次读出数据32个字节，data[11~16]表示加速度传感器数据，data[17~22]表示陀螺仪数据。在加速度传感器数据中，x轴数据用[11][12]表示；其中[11]表示低字节，[12]表示高字节；y轴数据用[13][14]表示；其中[13]表示低字节，[14]表示高字节；z轴数据用[15][16]表示；其中[15]表示低字节，[16]表示高字节。其他类似。对于读出的原始数据，我们将其乘以一个系数，转换成合适的数据类型被Android读取。此系数可根据需求自行进项调整。提供参考如下：

acceleration.x = 原始数据 \* CONVERT\_A\_X;

其中CONVERT\_A\_X = (((9.80665f) / (64.0f)/4))

现在，还有一个问题没有解决：打开dev可以看到dev下不止一个hidraw设备，我们从哪个hidraw读取数据呢？这里就涉及到USB HID。USB HID （Human Interface Device，人机接口设备）是USB设备中常用的设备类型，是直接与人交互的USB设备，例如键盘鼠标与游戏杆等。在USB设备中，HID设备的成本较低。另外，HID设备并不一定要有人机交互功能，只要符合HID类别规范的设备都是HID设备。使用HID设备的一个好处就是，操作系统自带了HID类的驱动程序，而用户无需去开发复杂的驱动程序，只要直接使用API调用即可完成通信。所以很所简单的USB设备喜欢枚举成HID设备，这样就可以不用安装驱动直接使用。HID协议的详解可参看官网，此处不做深入讨论。

为了便于设计上的通用性，我们采用读报告描述符的方法来确定需要的hidraw，像定义Sensor数据格式那样，我们定义一个报告描述符，只要HID设备按照该报告描述符发送数据，我们就认为它是我们要读取的hidraw设备。报告描述符的定义如图3.1.3所示。



图3.1.3 HID报告描述符定义

我们解决了Sensor数据格式的定义以及读取数据的方法，至此，通过编码实现和调试，就可以实现读取真实Sensor数据了。

一般来说，当前智能电视中体感游戏说需要的传感器主要包括加速度传感器和陀螺仪传感器，底层支持了这两个传感器，就满足了当前市场上大多数的体感游戏对传感器数据的要求。

**3.2 获取虚拟Sensor数据**

上边讨论了如何获取真实Sensor的数据，下边看第二个需求：获取虚拟Sensor的数据。虚拟Sensor的数据来源于手机第二屏遥控器，这里解释下“手机第二屏遥控器”的含义，手机第二屏遥控器指手机或pad上安装相应的遥控app软件，将手机模拟成电视/机顶盒的软遥控器。

我们要做的是提供相应的接口，便于app调用此接口写入Sensor数据到系统。为此，我们创建一个命名管道，Sensor HAL从命名管道中读取虚拟Sensor数据，app侧调用接口写入虚拟Sensor数据。在读取端，随系统启动的时候Sensor HAL调用mkfifo在系统目录data下创建一个命名管道，然后调用open打开将其做为读入端，接着调用read和poll读取和等待数据。在app侧，我们提供数据写入的接口，主要包括三个：openSensor()，writeData(const sensor\_event\_t \*data)，和closeSensor()。在openSensor中我们打开命名管道作为写入端，在writeData中进行数据的写入，其数据结构采用android原生的结构体sensor\_event\_t，最后在数据写入完毕的时候调用closeSensor进行关闭。

手机第二屏遥控器在使用的时候将从本机获取的Sensor数据或者模拟出的虚拟数据转换成sensor\_event\_t，直接调用写入接口即可。

此项设计及实现的方案并不复杂，主要涉及的是命名管道的使用，及注意写入时的同步操作。

**4. 测试与验证**

测试与验证，可采取操控体感游戏的方式进行，比如赛车游戏及体感运动。本文测试过的游戏包括两款：“真实赛车”和“运动加加”中的乒乓球。下边简单描述了真实赛车的测试流程和结果。

**4.1真实Sensor数据**

插入体感遥控器USB端子到智能电视USB端口，安装测试游戏“真实赛车”，打开真实赛车游戏，挥动体感手柄，可以看到赛车可以流畅地进行左转右转，并且游戏操作体验流畅。

**4.2虚拟Sensor数据**

安装，并打开“真实赛车”游戏，接入手机第二屏遥控器app，挥动手机进行控制，可以发现赛车已经被操控随手机倾斜而左转右转。

以下（图4.2.1）是游戏过程中操控不同车型在不同场地比赛的截图。



图4.2.1 测试画面截图

**5. 结束语[[2]](#footnote-2)**

本文较为系统的阐述了Android系统下Sensor数据传输的框架和流程，设计和实现了Sensor数据传输的一种方案，解决了真实Sensor数据和虚拟Sensor数据的传输需求。在文中提供了Sensor数据结构和HID报告描述符的标准定义，任何厂家的体感设备只要符合该标准，都可以进行良好的兼容与适配。

当然，对于Android HAL层的实现方案不止一种，各个厂商可根据自己的需求进行定制实现，本文只是提供了一个方式进行讨论。此方案也存在某些不足，比如同时连接多个遥控器操作，这是本文没有深究的，可作为以后的课题深入探讨。

最后，向此论文完成过程中给予支持和鼓励的学校导师，公司领导、同仁和各位同学表示感谢，谢谢你们耐心的指导和中肯的建议。

**参考文献**

[1] 汪永松,Android平台开发之旅(第2版)[M]，北京，机械工业出版社，2012，10-13

[2] 王振丽，Android底层开发技术实战详解：内核、移植和驱动[M]，北京，电子工业出版社，2012，84-98

[3] 苏健，[Android智能手机平台电源管理](http://xueshu.baidu.com/s?wd=paperuri%3A%2810001ac99ef4932646e3f0abf230693e%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fd.wanfangdata.com.cn%2FThesis%2FD195021&ie=utf-8)[J].微机处理，2011(5)：66-69

[4] Google Inc，Sensors Overview[EB/OL]. http://developer.android.com/guide/topics/sensors/

sensors\_overview.html，2016-3-20

[5] 罗升阳, Android系统源代码情景分析[D]，北京，电子工业出版社，2016，13-43

1. 作者简介：胡济豪（1985.6-）男，工程硕士，研究方向：软件工程 [↑](#footnote-ref-1)
2. 作者联系方式：邮箱[hujihaoxinxiang@163.com](mailto:hujihaoxinxiang@163.com) 电话13524535860 [↑](#footnote-ref-2)