摘 要

本报告总结了一种用于检测物体移动的系统的设计过程，并对其特性、设计方案、工程问题分析、方案实现、系统测试等方面进行了讨论和阐述。该系统的检测端使用Flutter编写，配合传感器在安卓手机上运行；服务端使用Java编写的后端接收数据，并配合数据库进行数据的存储与分析；监控端同样使用Flutter编写，用于接收从检测端经服务端传递的动态。

**关键词：传感器，Flutter，Socket通信，移动检测**

# 目 录

第一章 针对复杂工程问题的方案设计与实现 1

1.1 针对复杂工程问题的方案设计 1

1.2 针对复杂工程问题的推理分析 2

1.3 针对复杂工程问题的方案实现 1

第二章 系统测试 2

第三章 知识技能学习情况 3

第四章 分工协作与交流情况 4

参考文献 5

致谢 6

# 第一章 针对复杂工程问题的方案设计与实现

1.1针对复杂工程问题的方案设计

1.1.1对于复杂工程问题的简析

本期的复杂软件工程问题是开发用于检测移动的软件系统。具体涉及到的问题有对物体移动的检测、对检测数据的汇报以及服务器通信的加密等等，需要为各个结点分别开发能实现相应功能的软件。总体结构设计上，我们采用客户端-服务器模式，分别开发用于检测的客户端和用于监控、接收检测动态的客户端，由服务器存储数据并负责和各客户端之间的通信。

具体详细设计上，检测客户端利用智能手机内置的各种传感器实现运动检测；接收检测动态的客户端和服务器建立长连接接收推送；服务端使用Socket进行和各客户端之间的通信和数据交换。

总体结构设计的客户端-服务器模式（C-S）中，我们的服务端基于国内公有云平台的计算资源进行搭建，并配备相应的PostgreSQL数据库等资源。服务端操作系统使用Ubuntu，采用Nginx作为代理，使用Java编写后端。而用于检测的客户端为安卓平台的应用程序，运行在具备特定硬件条件（加速度传感器和方向传感器）的手机设备上，使用flutter编写，在后台长期运行，通过收集相关传感器的数据，使用相应的算法和代码判断物体是否发生了移动。如果发生移动，则通过到服务端的连接回报状态。用于监控和接收检测动态的客户端同样是安卓平台上的应用程序，使用flutter编写，通过和后端服务器之间通信轮询或主动推送，获得当前动态并向用户展示，提供成组结果汇报、历史动态汇报等统计特性。有特别情况的，可以通过播放警报音、使用手机震动等提醒用户查看情况。

服务端选用Ubuntu系统，考虑到Linux系统长期运行时的稳定性和后期的可扩展性、目前服务端软件的兼容性，选用服务器的长期支持版本。服务器使用开源的异步服务器Nginx对后端程序进行隐藏，配合必要的鉴权提高安全性，并保证服务端性能。而后端开发采用Java则是考虑到安卓平台的开发语言与Java具有较高的相关性，采用Java编写服务端的话可以在部分双端通用的地方减少代码工作量和设计思路。数据库则是使用稳定且长期维护的对象-关系数据库PostgreSQL，使用ORM框架Mybatis连接到数据库，方便管理且在一定程度上避免了繁琐的SQL语句直接操作。

用于检测的客户端沿用了之前在移动端开发时使用的flutter，其采用dart语言，并且可以实现跨平台运行。考虑到硬件需求以及平台开放性，我们的主要开发硬件平台依然是安卓平台。Flutter的界面绘制简单直观，而dart语言编程也已经可以实现大部分的常用功能，涉及到dart不能完成的需求时，还可以使用各平台的原生代码嵌入到flutter应用程序中进行开发。Flutter的热重载特性也大大缩短了开发的时间，增强了软件开发过程中对于调试感受的直观性，让代码的改动能更快地反映为手机上的测试应用变化。依赖于dart的健全空安全体制，使用flutter编写应用时可以获得额外的安全性提升，以及更小的编译二进制文件，为软件分发和软件安全带来诸多好处。在与服务器的连接方面，应用程序使用socket连接，可以与服务器建立安全持久的连接，并保证动态及时通知服务器，相比于其他轮询等方法节省了资源的同时保证了通信效率。客户端在连接之前需要填写从服务端获取的唯一ID，用于标识客户端和防止外部攻击。

用于监控和接收检测动态的客户端在开发时可以很大参照已经有的服务端设计和检测客户端设计，从而减少新增代码量，实现代码复用，减少后期测试时的冲突等问题。监控和接收检测动态的客户端同样采用flutter编写，但是为其绘制不同的界面，并且实现和检测客户端不一样的功能。监控客户端主要涉及到对于动态的提醒，类似于接收新的消息，所以要在保证消息到达效率的前提下尽可能优化对于资源的消耗。在这方面的设计上可以考虑使用部分系统集成的推送服务，以实现推送时效和资源消耗最小化的双重目标。同时，对于如何排版向用户展示当前情况，也需要进行设计。检测客户端可能涉及到一个或多个客户端的情况，以及一次和多次汇报、历史动态等，需要设计相应的界面或者对于整体界面设计进行协调，以展示各方面数据。监控客户端的安全性要求相比于检测客户端更高，所以在服务器端配置相应的账号密码登录，以防止未授权访问，同时也可以基于账号系统进一步完善出相应的权限管理系统，实现不同用户角色的不同权限，对于检测客户端做细致分类。

对整个系统进行设计时，我们根据要实现的目的，即用于监测移动的软件系统，对于需要设计实现的要点做了大致的划分，然后确立了以包括服务器设计为完整结构的客户端-服务器-客户端三点架构，并对于其中各方和其中的链路进行设计和研究。对于其中各个要点可能的发展方向和需要实现的功能要点采用思维导图进行绘制，然后对于必须要实现的特性和改进特性做标记区分，着重开发重点功能。同时，在设计各部分应该具备的功能时参照其他部分的情况，做出相应的调整，使得整个系统之间的各个部分协同工作，契合运行。比如对于检测客户端，则应当着重实现优化检测功能，然后确保数据回报功能正常、稳定、可用。而服务器方面，核心要求事能稳定持续运行，并且正确处理和各个客户端、数据库之间的通信和数据处理，并且对于收集的数据进行可选的统计分析、对于不同用户和客户端的分级权限管理。监控客户端的推送和告警功能是核心功能，并且也需要对于数据的呈现有一定优化设计。

另：

随着人们对移动产品的熟悉，人们越来越倾向于喜欢简单的轻量级产品，这也是本综合设计主要的考虑方向，如何带个用户最简易的交互，最明了的图表数据支持。从产品设计之初就考虑是否能够创新的使用一种极简的交互方式完成测量，能够让用户通过一种新的交互操作去完成测量和监控任务，这种交互同时沿用实际生活中人们的习惯和尝试，带来一种新的体验。

1.1.2针对复杂工程问题需要实现的目标

在智能手机上利用传感器实现对设备移动的测量和计步的监控等；

结合的用户需求分析以及系统在实践过程中的实现难度，拟实现下面几个功能：

#### 利用加速度传感器实现设备A位移的测量

能够通过设备的位移测量出所测距离的结果：对距离的测量目前有很多方式，可以通过拍照，扫描或是利用激光。通过这些复杂的设备能够做到距离测量，但这些设备都较贵，不常用。在实际生活中对于短距离的测量我们通常是利用皮尺或者卷尺这样的工具进行测量，但又不是常常会带在身边，我们通过一个程序将每天随身携带的手机变成这样一个测量工具。同时也模拟实际测量时候的操作，利用手机直接对带测量物滑动一遍就完成了测量，同时也是对手机移动的一种检测，对防盗有一定用处。

#### 利用加速度传感器获取到设备A加速度的变化，然后通过算法来检测步数。

能通过设备的加速度计算出人的步数运动：现在的智能手机嵌入了一些微小的传感器，比如重力传感器、光传感器、声音传感器等。如何有效地利用这些传感器来开发一些应用，是一个值得深入研究的课题。比如开发医疗健康的应用、运动量监视器等。本次综合设计通过加速度传感器来开发一款监视步数的程序，程序的关键在于计步模型的建立。

#### 将位移数据和步数数据传输到设备B

能将设备A的位移数据和步数数据安全传输到设备B：通过认证和通信达到安全传输数据和监控的目的。

#### 能够直观的在屏幕上显示测量过程中的数据变化，以及图表变化。

能够直观的在屏幕上显示移动的数据变化，以及图表变化：在实际的测量中用户会关注测量过程中的一些中间数据，特此我们将其中的加速度变化以形图的方式呈现出来，同时也实时更新各个方向上的位移数据。

#### 方便快捷的实现测量的一些参数设置和开关

方便快捷的实现测量的一些参数设置和开关：在屏幕上我们设置了方便的设置按钮，分别能够控制几个量：去重力加速度、水平校正、滤波。通过这三个设置我们可以对测量进行测量前的修正，由于手机上的加速度传感器没有达到很髙的准确度，在测量经常会有部分误差，且不可控，所以水平校正没很有必要，测量过程中也会有部分噪声，需要滤除掉。

另外对于一个成熟的系统来说，还要有一些对应的系统性能相关的需求来保证系统高效准确的运行，因此除了需要满足以上提出的需求，我们还需要去提高性能，努力改善用户的体验。虽然智能手机的配置越来越高，各种性能也有很大的提升，但是手机的局限性也蛮大的，特别是作为一种智能设备。在架构方面，目前无法满足多程序的并发，发热和CPU资源方面也有待改善。所以设计系统的时候，需要多考虑下面几点：

1. Android系统目前装配在各种厂商的不同机型上，使得我们在设计系统的时候一定要考虑各种屏幕分辨率适配的问题，使我们的程序能够在各种机型上正确运行。

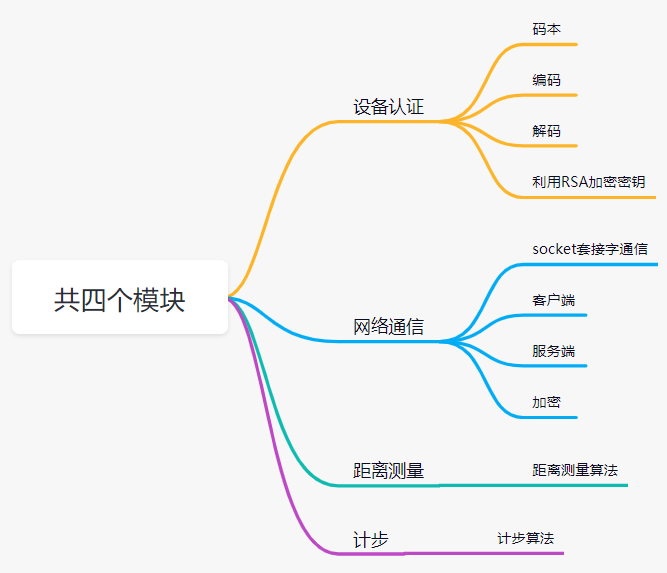
2. 本课题中能耗主要是由客户端自身和调用的传感器产生，程序中渉及的一些传感器的使用，可能会在后台一直运行它们，采集加速度的数据。所以需要考虑这些传感器一直运行时如何去节省电量，减少手机发热。

3. 测量结果是否准确也需要重点考虑，由于手机内置的加速度传感器自身的不精确，从而给本课题带来很大的挑战。

4. 可扩展性，移动感知环境在未来一定会有更多的用法以及功能，距离测量也是，以后一定会实现三维空间的测量。这样要求系统能够进行扩展，以备以后继续实现更多新功能。

1.1.3实现工程问题所需的模块化设计

#### 1.1.3.1模块划分



共四个模块：

1. 设备认证
2. 网络通信
3. 位移向量计算
4. 计步算法

四个模块之间都能独立验证，组合起来集成测试，各自又分成多个组件，能独立单元测试，也能组合进行集成测试。

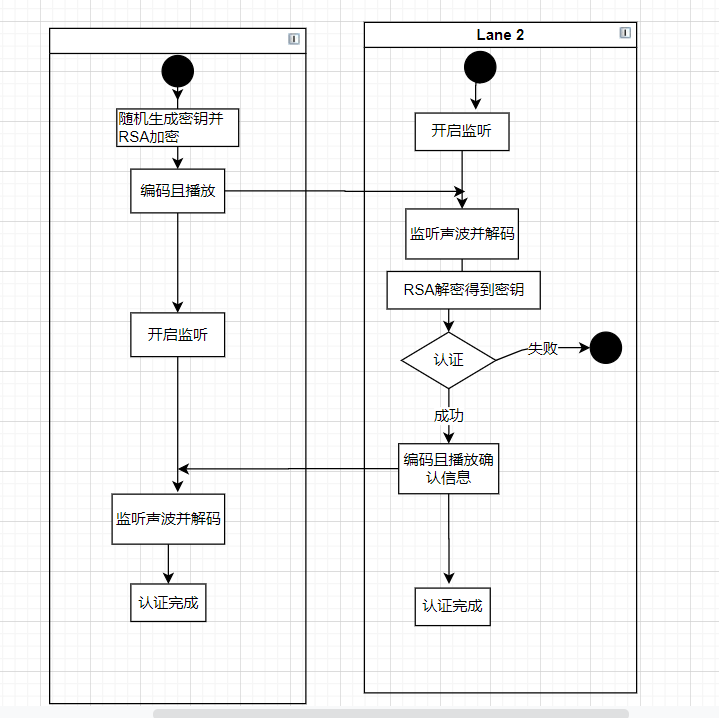
#### 1.1.3.2模块设计

1.1.3.2.1设备认证

采用声波通信的方式传递密钥和认证。

原理：声波通信的原理比较简单，主要是用单频率声音信号对数据进行编码，然后播放这些单频率声音，接收方在收到声音后，识别出频率，然后根据频率解码出数据。

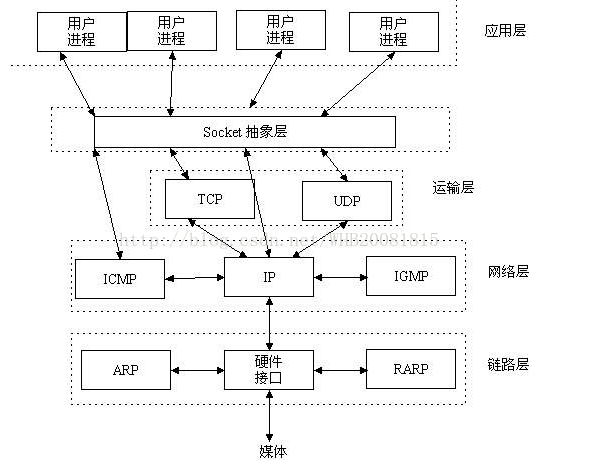
比如：我们可以将1500HZ的正弦波对应数字1，1600HZ的正弦波对应数字2，1700HZ的正弦波对应数字3。那么数字串3123就对应4段正弦波，规定每段正弦波持续100ms，则3123对应400毫秒的声音段。接收方录制声音，对收到的声音进行解析，识别出1700HZ，1500HZ，1600HZ，1700HZ四段正弦波频率，然后查找码本，解码出的数字就是3123。

下图演示进行认证：

1.1.3.2.2网络通信（socket+加密）

采用Socket套接字通信和对称加密算法AES（高级加密标准）和非对称加密算法RSA

由于AES具有计算量小、加密速度快、加密效率高，但安全性低的特点，以及RSA安全性高，但运算速度较慢的特点，最后确定了在设备认证时将AES的密钥通过RSA加密后发给对方，再用AES算法进行加密和传输，做到了对通信数据的加密，保证了通信的安全性以及高效性。



**1).Socket传输的优点：**

1.传输数据为字节级，传输数据可自定义，数据量小（对于手机应用讲：费用低）

2.传输数据时间短，性能高

3.适合于客户端和服务器端之间信息实时交互

4.可以加密,数据安全性强

**2).加密算法**

**2.1).对称加密：**

* 双方使用的同一个密钥，既可以加密又可以解密，这种加密方法称为对称加密，也称为单密钥加密。
* **优点：**速度快，对称性加密通常在消息发送方需要加密大量数据时使用，算法公开、计算量小、加密速度快、加密效率高。
* **缺点：**在数据传送前，发送方和接收方必须商定好秘钥，然后使双方都能保存好秘钥。其次如果一方的秘钥被泄露，那么加密信息也就不安全了。另外，每对用户每次使用对称加密算法时，都需要使用其他人不知道的唯一秘钥，这会使得收、发双方所拥有的钥匙数量巨大，密钥管理成为双方的负担。
* 在对称加密算法中常用的算法有：DES、AES等。
  + AES：密钥的长度可以为128、192和256位，也就是16个字节、24个字节和32个字节
  + DES：密钥的长度64位，8个字节。

**2.2).非对称加密：**

* 一对密钥由公钥和私钥组成（可以使用很多对密钥）。私钥解密公钥加密数据，公钥解密私钥加密数据（私钥公钥可以互相加密解密）。
* 私钥只能由一方保管，不能外泄。公钥可以交给任何请求方。
* 在非对称加密算法中常用的算法有：
* RSA、Elgamal、背包算法、Rabin、Diffie-Hellman、ECC（椭圆曲线加密算法）。
* 使用最广泛的是RSA算法，Elgamal是另一种常用的非对称加密算法。
* **缺点：**速度较慢
* **优点：**安全

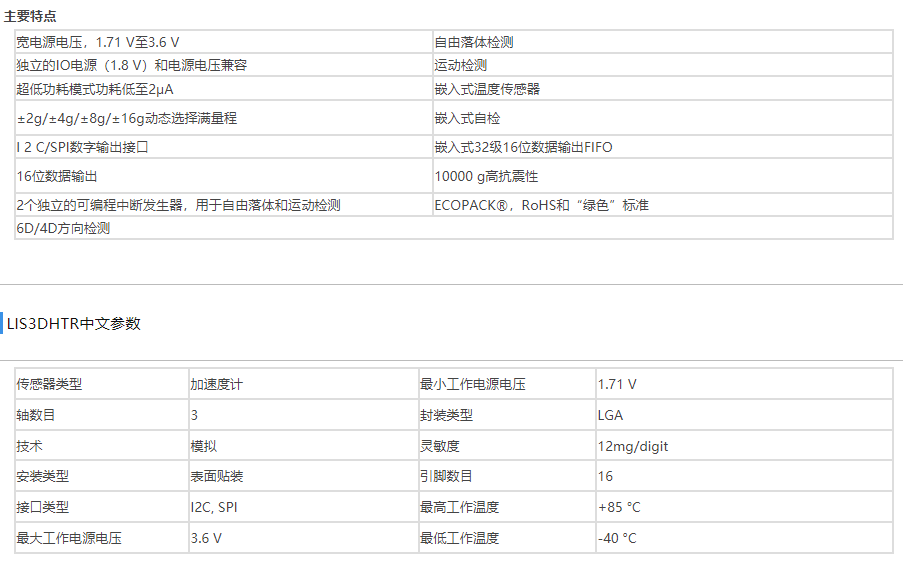
#### 1.1.3.2.3距离计算

**利用手机加速度传感器**

传感器技术在现代科学中起着越来越重要的作用，各种先进的设备，仪器中都大量用到传感器，使用这些传感器去采集数据进行科学分析，或者使用这些传感器去使我们的设备更智能。正是由于传感器技术的发展才为科学技术向前，社会发展打下了良好的基础。近年来在原先传统传感器技术的基础上，发展了新的传感器制作工艺，最后研制出的新型传感器，大幅提高了微电子技术。  
 对于测量加速度的传感器，分为以下几种电容式、谐振式、压阻式和压电式加速度传感器。

1. 电容式加速度传感器：它的基本工作原理是基于电容的瞬间变化来测量加速度变化，所以它具有灵敏度高，灵敏度受周围的环境影响小，线性度较好的优点；缺点：由于外界对电容的干扰的客观存在，所以精度受外界干扰较大。
2. 谐振式加速度传感器：它的原理是物体加速度变化时，谐振变化，这将影响物体固有的频率，通过固有频率的变化从而可以测得加速度的大小。特点是输出电路简单，还可以直接数字输出，缺点是复杂的制作工艺，以及受外界温度影响较大。
3. 压阻式加速度传感器：它的基本工作原理是加速度变化时导致动力机械装置产生的阻尼效应来测量加速度的变化。特点是工艺简单，测量方法也简单，同时保持了良好的线性度。
4. 压电式加速度传感器：它的工作原理是利用压电陶瓷或石英晶体的压电效应（压电效应：某些电介质在沿一定方向上受到外力的作用而变形时，其内部会产生棱化现象，同时在它的两个相对表面上出现正负相反的电荷。当外力去掉后，它又会恢复到不带电的状态，这种现象称为正压电效应。当作用力的方向改变时，电荷的极性也随之改变。相反，当在电介质的极化方向上施加电场，这些电介质也会发生变形，电场去掉后，电介质的变形随之消失，这种现象称为逆压电效应。依据电介质压电效应研制的一类传感器称为压电传感器。），优点是测量的频率范围大，缺点是受环境影响，同时分辨率低。

目前手机中加常见的加速度传感器是LIS3DH3-axis，该加大速度传感器由意法半导体(STMicroelectronics)公司生产。该传感器可以感应手机的运动，不停的获取空间三维坐标系中的X,Y,Z三个分量。该传感器具有体积小（长3mm，宽3mm，高1mm，功耗小的特点。所以比较适合使用在一些工号要求非常低的场合和一些对运动敏感度高的场合，例如性能较高的手持设备：手机、计步器、游戏机等等。测量范围：±2g/±4g/±8g/±16g。



对加速度传感器的三个参数处理得到位移输出。

#### 1.1.3.2.4计步算法

**监听传感器**

1. 取得SensorManager

* SensorManager sm=(SensorManager)getSystemService(SENSOR\_SERVICE);

1. 实现接口SensorEventListener

* public void onAccuracyChanged(Sensor sensor,int accuracy){}  
  public void onSensorChanged(SensorEvent event){}
* 在onSensorChanged(SensorEvent event)中,Accelerometer Sensor测量的是所有施加在设备上的力所产生的加速度的负值（包括重力加速度）。加速度所使用的单位m/sec2
* 数值是加速度的值,如下:
  + SensorEvent.value[0];加速度在X轴的值
  + SensorEvent.value[1];加速度在Y轴的值
  + SensorEvent.value[2];加速度在Z轴的值

1. 取得某种Sensor对象 ，Sensor.TYPE*TEMPERATURE填写为Sensor.TYPE*ACCELEROMETER，表示类型为 加速度传感器

* List<Sensor> sensor =sm.getSensorList(Sensor.TYPE\_ACCELEROMETER);

1. 注册SensorListener,该函数位于onResume()中 。

* sm.regesterListener(SensorEventListener listener,Sensor sensor,int rate)
* 需要说明的是第三个参数是延迟时间的精密度 。

1. 最重要的是使用完后一定记得注销掉SensorManager，同时手机在黑屏的时候是不会自动停掉 传感器的使用的 ， 所以从能耗的角度上 ， 不使用的时候需要通过函数unregesterListener()来释放， 该函数位于Activity的onStop()函数中

* sm.unregesterListener(this);

**利用计步算法计算步数**

1.2针对复杂工程问题的推理分析

检测物体移动方面，需要实现对于物体移动的检测。而具体实现时有多种方法可供选用，其中既包括使用硬件的UWB技术、陀螺仪、红外、磁力等传感器，也可以使用基于图像的识别等方法。考虑到不同手机配备的传感器存在差异，而图像成像素质也各有不同，需要找到一个能适应绝大部分情况并且拥有良好和稳定检测率的检测方法。同时，可以考虑在耗电、算法效率等多方面进行优化。最终考虑到不同平台特性之间的差异，选用加速度传感器和方向传感器作为用于检测的传感器，并以此开发检测方法。

与服务器通信方面，监测用的客户端不必一直和服务器保持连接，而只需要在检测到相应状态时向服务器发起通信；而换一种思路的话，也可以和服务器保持一个长连接，并在产生变动时及时向服务器发送数据，从而保证通信的及时性，减少了每次建立连接的步骤，类似于现代移动聊天软件和服务器之间获取新消息的逻辑。同时，为了保护相应动态免受可能的网络攻击，需要对通信进行一定的加密保护。选用低延时的通信协议和高性能的加密方法也是需求的一部分。通过在多种方案之间比较，TCP/Socket和HTTP通信等具有不同特色的通信方案之间比较后，我们选择使用socket通信来完成客户端到服务器的数据回报。

服务器架构设计方面，考虑到客户端的数量不固定，需要为服务器的可扩展性进行相应的设计。同时，由于需要并发处理需求，期间的数据库操作、死锁预防以及并发处理的性能都需要各种优化。对此，根据实际规模考虑是否要对服务端进一步进行分布式设计、负载均衡等以优化实际性能。后端开发和检测软件既可以采用相同的语言，也可以采用有更加适合满足解决实际问题需要的后端设计特性的语言。因此我们使用易于扩展的Nginx作为网络服务器，并且在其后运行的采用Java编写的后端可以添加多进程和多线程支持，以满足突发性能需要。数据库也足够高效，可以支撑整个系统的运行。

监控客户端方面，涉及到对数据的定期更新甚至实时刷新，所以应该考虑与服务器长期保持一个活跃连接的形式，并且服务器与客户端之间通过管道进行通信和状态推送。有关如何实现这种长连接的技术也有很多，以及推送相关机制的实现和改进。应该考虑在长时间使用时，如何保持消息的及时性的同时尽可能降低能耗，防止监控客户端耗电过快从而导致监控失效等情况。针对这个问题，我们考虑使用手机自带的推送服务，以在尽量减少额外资源消耗的情况下获得最及时的推送和告警。同时，对于统计方面，客户端如果能给出检测的状态统计、历史记录则会提升终端用户的使用体验，这也作为我们的研究方向之一进行考虑。

1.2.1设备认证

已有上学期综设完成的基于音频的安全D2D通信系统，经过比较和权衡，决定采用开源的sinvoice的算法，sinvoice主要有以下优点：

1. 识别效率更高，几乎达到100%，完全可以达到商业用途标准，比chirp，支付宝，茄子快传等软件的识别效率更高。
2. 能支持更多复杂场景的识别，在有嘈杂大声的背景音乐，嘈杂的会议室，食堂，公交车，马路，施工场地，小汽车，KTV等一些复杂的环境下，依然能保持很高的识别率。
3. 能支持更多token的识别，通过编码可以传送所有字符。
4. 通过定制可以实现相同字符的连续传递,比如“234456”。
5. 支持自动纠错功能，在有3个以内字符解码出错的情况下可以自动纠正。
6. 程序运行效率非常高，可以用于智能手机，功能手机，嵌入式设备，PC，平板等嵌入式系统上。
7. 声波的频率声音和音量可定制。

1.2.2网络通信

#### 1.2.2.1 Server 服务端

**步骤一：**建立一个服务器 Socket

常见的一个服务器 Socket 类是 ServerSocket，ServerSocket 类常用三个方法：binder 、accept、close。

bind 方法为 ServerSocket 绑定 IP 地址和端口号，并开始监听该端口；accept 方法为 ServerSocket 接收请求并返回一个Socket 对象，accept 方法调用之后将一直阻塞，直到有请求达到；close 方法关闭一个 ServerSocket 对象。

初始化是一般需要设定 IP 地址和端口号

**步骤二：** 通过监听获取一个用于通信的 Socket 对象

这一步直接通过上述 accept 方法的执行就可实现

**步骤三：**在一个新线程中，通过对 Socket 对象进行封装

分别得到输入、输出流的引用对象，通过这两个对象向 Client 端发送或者从 Client 端接收数据，进而实现 Socket 通信。

一般选择在循环中读取 Client 发送过来的信息，并作出对应的处理，比如反馈 Client 端：自己已成功收到相应的消息。

**步骤四：** 在适当的时机关闭 Socket 连接

#### 1.2.2.2 Client 客户端

**步骤一：**初始化 Socket 对象

客户端一般选择在一个新的线程中，初始化一个 Socket 对象，初始化是需要设置 IP 和端口号，以帮助低层网络路由找到相应的服务端进程。

**步骤二：**获取与 Server 端通信的引用

此步和 Server 端建立连接后的步骤类似：根据步骤一中获取的 Socket 对象，进行封装，得到相应的输入、输出流对象，这些输入、输出流对象就是和 Server 端进行通信的引用。

**步骤三：**通过步骤二中得到的引用，循环的读取（在新线程中） Server 端发送过来的消息，并做相应的处理

**步骤四：**在合适的时机关闭与 Server 端的 Socket 连接

#### 1.2.2.3加密

**1）客户端**

客户端加密过程主要分为以下三个步骤：

1. 客户端随机产生AES的密钥；
2. 对身份证信息（重要信息）进行AES加密；
3. 通过使用RSA对AES密钥进行公钥加密。

这样在传输的过程中，即时加密后的AES密钥被别人截取，对其也无济于事，因为他并不知道RSA的私钥，无法解密得到原本的AES密钥，就无法解密用AES加密后的重要信息。  
 我们这里模拟客户端和服务端，对算法逻辑进行验证。

**2）服务端**

服务端使用RSA+ AES对重要信息进行解密

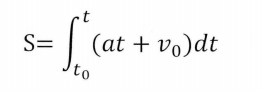
服务端解密过程主要分为以下两个步骤：

1. 对加密后的AES密钥进行RSA私钥解密，拿到密钥原文；
2. 对加密后的重要信息进行AES解密，拿到原始内容。

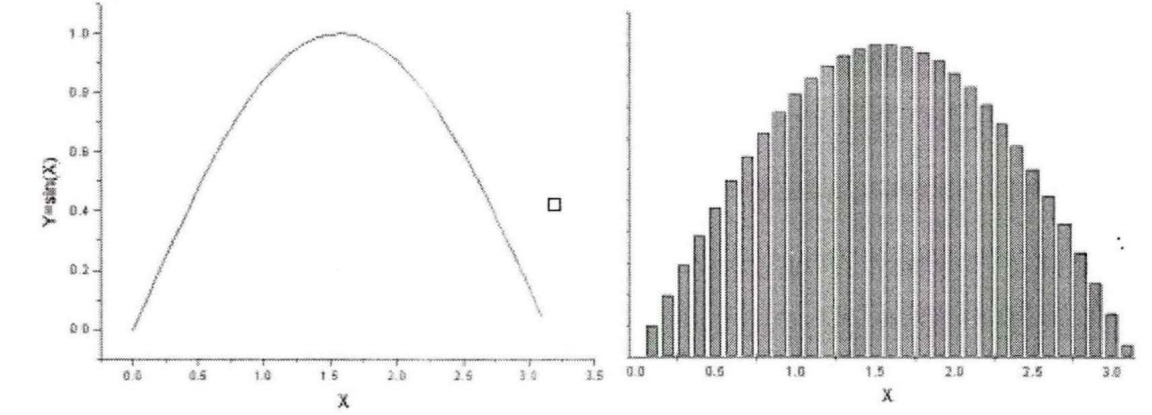
1.2.3位移向量计算

**原理**

主要是将之前加速度传感器采集到的数据进行处理，通过时间轴上的一系列加速度序列计算出该段时间的位移量。  
 通过加速度传感器测量空间位移，根据牛顿第二定理，基于加速度测量值，可以得到速度和距离.



通过以上的数值积分我们就可以在确定初速度和初始位移的情况下计算得到一定时间内从初始位移到终点位移的位移大小。  
 对于这个积分函数的计算，我们是通过矩阵积分进行近似计算的。釆用两点之间的中间值乘上他们的时间间隔，得到的矩形面积近似等于函数自身的面枳.



从上图可以看出,我们通过这两个图的对比，当我们划分区间的值愈来愈大的时候，右边的图形将无限接近于左边。当n趋近无穷的时候，左边的图形和右边的图形完全相等。所以我们也可以通过求取右边图形的面积来近似求取左边图形的面积，这就是矩阵积分法的原理。

在手机中我们加速度传感器的采样频率非常高，一般的手机都有上千的采样频率。这这足够高的采样频率下，我们的积分会有较好的近似度。

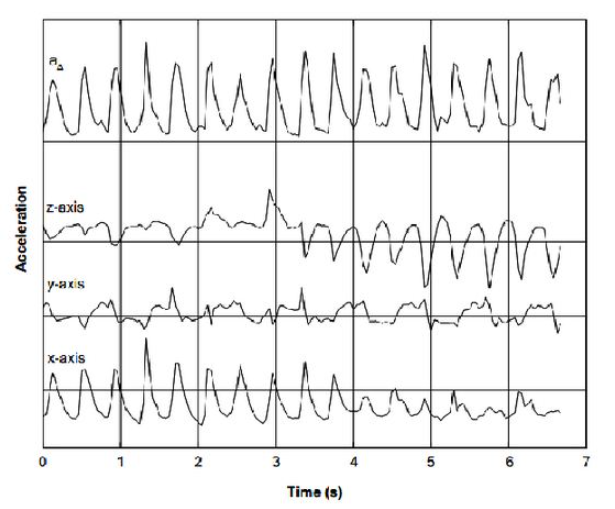
1.2.4计步算法

#### 1.2.4.1先要摸清模型的特征

目前，大部分设备都提供了可以检测各个方向的加速度传感器。以iOS设备为例，我们利用了其三轴加速度传感器（x，y，z轴代表方向如图）的特性来分析。分别用以检测人步行中三个方向的加速度变化。

用户在水平步行运动中，垂直和前进两个加速度会呈现周期性变化，如图所示。在步行收脚的动作中，由于重心向上单只脚触地，垂直方向加速度是呈正向增加的趋势，之后继续向前，重心下移两脚触底，加速度相反。水平加速度在收脚时减小，在迈步时增加。

反映到图表中，可以看到，在步行运动中，垂直和前进产生的加速度与时间大致为一个正弦曲线，而且在某点有一个峰值。其中，垂直方向的加速度变化最大，通过对轨迹的峰值进行监测计算和加速度阀值决策，即可实时计算用户运动的步数，还可依此进一步估算用户步行距离。



#### 1.2.4.2.计步的合理算法

因为用户在运动中可能用手平持设备，或者将设备置于口袋中。所以，设备的放置方向不定。为此，通过计算三个加速度的矢量长度，我们可以获得一条步行运动的正弦曲线轨迹。

第二步是峰值检测，我们记录了上次矢量长度和运动方向，通过矢量长度的变化，可以判断目前加速度的方向，并和上一次保存的加速度方向进行比较。如果是相反的，即是刚过峰值状态，则进入计步逻辑进行计步，否则舍弃。通过对峰值的次数累加，可得到用户步行的步伐。

最后，就是去干扰。手持设备会有一些低幅度和快速的抽动状态，或是我们俗称的手抖，或者某个恶作剧用户想通过短时快速反复摇动设备来模拟人走路，这些干扰数据如果不剔除，会影响记步的准确值，对于这种干扰，我们可以通过给检测加上阀值和步频判断来过滤。

1.3针对复杂工程问题的方案实现

**开发环境**

我们使用的是Android 10.0进行的系统开发，由于这个版本比较高。在开发测试中，我们涉及到的环境包括：

|  |  |
| --- | --- |
| 系统环境 | 简介 |
| eclipse集成开发环境 | eclipse是一个开放源代码的、基于Java的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。在幵发Android程序时是需要在幵发中安装环境。 |
| Android SDK 10.0 | 软件开发工具包：被软件开发工程师用于为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的开发工具的集合。用来开发Android应用。 |
| Android Studio 4.2.2 | Android Studio 是谷歌推出的一个**Android集成开发工具**，基于 IntelliJ IDEA. 类似 Eclipse ADT ，Android Studio 提供了集成的 Android 开发工具用于开发和调试。 |
| window10操作系统 | 以上的这些插件和工具都能运行在window操作系统上，他们在上window操作系统都能稳定运行 |

**硬件环境**

由于时间和财力限制，目前仅在 小米10（操作系统10.0）上实验。

**具体实现**

1.3.1设备认证

在sinvoice的demo中使用SinVoicePlayer.play(text)方法就可以播放出我们想要传输的数字的对应音频信号，然后进行解析即可。在SinVoicePlayer中，是通过调用PcmPlayer的start()方法，进行播放操作的，而在pcmPlayer中，则是又调用AudioTrace来实现最终的音频播放功能。通过对AudioTrace的一层层封装，最终实现了SinVoicePlayer的简单调用。

通过Encoder类、SinGenerator类和Buffer类完成对信息的编码、存储。其中，Buffer这个类是对字节数据进行存储的类，用来保存编码完成之后的字节数据。

1.3.2网络通信

#### 1.3.2.1客户端加密

客户端加密过程主要分为以下三个步骤：1.客户端随机产生AES的密钥；2.对身份证信息（重要信息）进行AES加密；3.通过使用RSA对AES密钥进行公钥加密。

这样在传输的过程中，即时加密后的AES密钥被别人截取，对其也无济于事，因为他并不知道RSA的私钥，无法解密得到原本的AES密钥，就无法解密用AES加密后的重要信息。

我们这里模拟客户端和服务端，对算法逻辑进行验证。

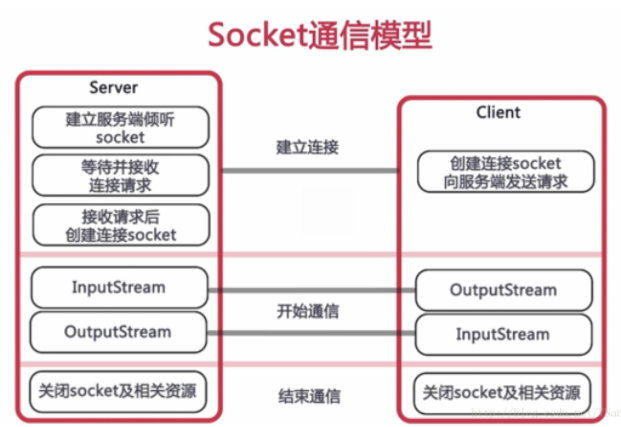
|  |
| --- |
| package RSAAES; import sun.security.krb5.internal.crypto.Aes128; public class TestEncrypt {  public static void main(String[] args) throws Exception {  String key = AESUtils.generateKey();  String name = "wujj";  String idCard = "444200002939402";  String encryptName = AESUtils.encryptData(key, name);  String encryptCard = AESUtils.encryptData(key, idCard);  String encryptKey = RSAUtils.encryptByPublicKey(key);  System.out.println("使用RSA加密后的AES密钥： " + encryptKey);  System.out.println("使用AES加密后的Name " + encryptName + " 使用AES加密后的Card :" + encryptCard);  } } |

#### 1.3.2.2服务端解密

服务端解密过程主要分为以下两个步骤：1.对加密后的AES密钥进行RSA私钥解密，拿到密钥原文；2.对加密后的重要信息进行AES解密，拿到原始内容。

|  |
| --- |
| package RSAAES;  public class Demo {  public static void main(String[] args) throws Exception {  String encryptKey = "W42a322SBoRZKTSLaczF74uUyAxGMFZSgvZWeWnEf+v7AlfTYo8Ie0JLUJpXMMfCjt3i+AlX+8M+Wd1FiEgwgkXwaObvqhqLdwYEybHdyYLE6XcUjn/vqSl/Z3jhKrYi5H+X6RZ70BVYDXki2q4vLPOfL8rECZywRPnl1+tAI7E=";  String encryptName = "Y5zy9Pf6h8jx/xTETt/2BA==";  String encryptCard = "7XVmAnoZsqq44aEOySVlsg==";  //解密  String key = RSAUtils.decryptByPrivateKey(encryptKey);  String name = AESUtils.decryptData(key,encryptName);  String idCard = AESUtils.decryptData(key,encryptCard);  System.out.println("使用RSA解密后的AES密钥： " + encryptKey);  System.out.println("使用AES解密后的Name " + name + " 使用AES解密后的Card :" + idCard);  } } |

#### 1.3.2.3客户端和服务端



1.3.3位移向量计算

在程序的实现过程中，我们定义了一个新的类DataHistory来封装数据，记录每个采样时间点的加速度的值，速度值以及位移值。

表格

描述已自动生成

时问戳可以通过System.currentTimeMillis()函数获取，该方法的作用是返回当前的计算机时问，时间的表达格式为当前计算机时问和GMT时间（格林威治时间）1970年1月1号0时0分0秒所差的毫秒数。

DataHistory的主要数据处理函数是append(float accel,long timestamp)，下面我们介绍其的实现过程:

|  |
| --- |
| public void append(float accel,long timestamp) {  double ms= （fla）（imestamp — lastTimestamp）/1000.0； //i计算出时间间隔  double midAccel = (lastAccel+accel)/2;//计算中间值  double deltaSpeed =midAccel\*ms://计算增长值  double newSpeed = lastSpeed + deltaSpeed;  double midSpeed - (lastßpeed+newSpeed)/2;   double deltaDistance = midSpeed ms;  double newDistance - lastDistance + deltaDistance;   lastAccel accel;  lastSpeed = newSpeed;  lastDistance = newDistance;   lastTimestamp = timestamp; } |

在系统中，我们分别定义了X、Y、Z三个方向的DataHistory对象，分别对三个方向进行处理，最后通过各个分矢量来计算出总位移量，即我们需要测量的距离。通过封装的函数getTotalDistance()来实现这一功能，其中使用到了Math.sqrt()这个函数，它是用来返回数值的平方根的数学函数。

|  |
| --- |
| double getTotalDistance(){  double dx=historyX.lastDistance;//获得X轴的位移量  double dy=historyY.lastDistance;//获得Y轴的位移量  double dz=historyZ.lastDistance;//获得Z轴的位移量  double d=(dx\*dx+dy\*dy+dz\*dz); //获得总的位移量  d=Math.sqrt(d);  return d; } |

水平校正的实现，是对静止的手机进行三个方向的加速度校正，其原因是加速度传感器的固有误差，可能是由于硬件自身缺陷也有可能是由于软件缺陷导致，每次测试都有大致不变的误差，所以在程序中我们通过校正的方式进行抵消。定义了一个float型数组来保存三个方向的校正值。

同样数据处理传感器采集回来的数据的时候，需要减去保存在修正数组中的各个数值之后再做处理。[1]

1.3.4计步算法[2]

#### 1.3.4.1均值滤波器——滤波

均值滤波器实现均值滤波，其实很简单就是拿到多组x,y,z三轴数据，求平均值。最后的平均值作为输出结果，使输出结果更加平滑。完成初步滤波。

|  |
| --- |
| #define FILTER\_CNT 4  typedef struct {  short x;   short y;   short z;  }axis\_info\_t;  typedef struct filter\_avg{  axis\_info\_t info[FILTER\_CNT];  unsigned char count; }filter\_avg\_t;  //读取xyz数据存入均值滤波器，存满进行计算，滤波后样本存入sample static void filter\_calculate(filter\_avg\_t \*filter, axis\_info\_t \*sample) {  unsigned int i;  short x\_sum = 0, y\_sum = 0, z\_sum = 0;  for (i = 0; i < FILTER\_CNT; i++) {  x\_sum += filter->info[i].x;  y\_sum += filter->info[i].y;  z\_sum += filter->info[i].z;  }  sample->x = x\_sum / FILTER\_CNT;  sample->y = y\_sum / FILTER\_CNT;  sample->z = z\_sum / FILTER\_CNT; } |

#### 1.3.4.2动态阈值

动态阈值： 系统持续更新3轴加速度的最大值和最小值，每采样50次更新一次。平均值(Max + Min)/2称为"动态阈值"。

对于动态阈值其实就是我们在采样过程中拿到的每个轴的最大值和最小值的平均值，它是每取50个样本就更新一次，是动态变化的。所以我们只需每50次样本，取出其中的最大最小值就行了。最大最小值，还有一个重要的功能，判断最活跃轴，因为我们最后判断步伐，也是根据哪个轴加速度变化最大认为哪个是活跃轴。

|  |
| --- |
| #define MAX(a,b) ((a) > (b) ? (a) : (b)) #define MIN(a,b) ((a) < (b) ? (a) : (b)) #define SAMPLE\_SIZE 50  typedef struct {  axis\_info\_t newmax;  axis\_info\_t newmin;  axis\_info\_t oldmax;  axis\_info\_t oldmin; }peak\_value\_t;  static void peak\_value\_init(peak\_value\_t \*peak);  //在动态阈值结构体初始化时，一定要将max的值都赋值为最小值，min赋值为最大值，这样才有利于动态更新。 static void peak\_update(peak\_value\_t \*peak, axis\_info\_t \*cur\_sample) {  static unsigned int sample\_size = 0;  sample\_size ++;   if (sample\_size > SAMPLE\_SIZE) {  /\*采样达到50个，更新一次\*/  sample\_size = 1;  peak->oldmax = peak->newmax;  peak->oldmin = peak->newmin;  //初始化  peak\_value\_init(peak);  }  peak->newmax.x = MAX(peak->newmax.x, cur\_sample->x);  peak->newmax.y = MAX(peak->newmax.y, cur\_sample->y);  peak->newmax.z = MAX(peak->newmax.z, cur\_sample->z);    peak->newmin.x = MIN(peak->newmin.x, cur\_sample->x);  peak->newmin.y = MIN(peak->newmin.y, cur\_sample->y);  peak->newmin.z = MIN(peak->newmin.z, cur\_sample->z); } |

#### 1.3.4.3动态精度

动态精度：这其实是由一个线性移位寄存器加上一个我们预设定的动态变化值来消除高频噪声。其实它的操作很简单，移位寄存器中有两个样本，一个是new*sample,一个是old*sample.只要我们采集到一个样本，当前样本cur*sample.当cur*sample到来，old*sample无条件更新为new*sample的值，但是new*sample就不是无条件更新了，假设我们设定动态精度DYNAMIC*PRECISION=10，当|cur*sample.x-new*sample.x|，如果X轴变化量大于DYNAMIC*PRECISION，那么new*sample.x就更新为cur\_sample.x，否则不变。

|  |
| --- |
| #define ABS(a) (0 - (a)) > 0 ? (-(a)) : (a) #define DYNAMIC\_PRECISION 30 /\*动态精度\*/   /\*一个线性移位寄存器，用于过滤高频噪声\*/ typedef struct slid\_reg{  axis\_info\_t new\_sample;  axis\_info\_t old\_sample; }slid\_reg\_t;  static char slid\_update(slid\_reg\_t \*slid, axis\_info\_t \*cur\_sample) {  char res = 0;  if (ABS((cur\_sample->x - slid->new\_sample.x)) > DYNAMIC\_PRECISION) {  slid->old\_sample.x = slid->new\_sample.x;  slid->new\_sample.x = cur\_sample->x;  res = 1;  } else {  slid->old\_sample.x = slid->new\_sample.x;  }  if (ABS((cur\_sample->y - slid->new\_sample.y)) > DYNAMIC\_PRECISION) {  slid->old\_sample.y = slid->new\_sample.y;  slid->new\_sample.y = cur\_sample->y;  res = 1;  } else {  slid->old\_sample.y = slid->new\_sample.y;  }     if (ABS((cur\_sample->z - slid->new\_sample.z)) > DYNAMIC\_PRECISION) {  slid->old\_sample.z = slid->new\_sample.z;  slid->new\_sample.z = cur\_sample->z;  res = 1;  } else {  slid->old\_sample.z = slid->new\_sample.z;  }  return res;  } |

#### 1.3.4.4步伐判断

步伐判断：通过资料图，我们可以看到，判断为一步的条件是，先找到最活跃轴，然后最活跃轴的old*sample > 动态阈值，new*sample < 动态阈值。满足上述条件，认为走了一步。在我们运动过程，可以认为连续运动大于5步才认为是走步，这样可以过滤一些不必要的错误步数，同时资料所说的最快大约每秒5步，相当于200ms一步，我们是利用每40ms取一次数据，200ms得到5组数据，滤波得到一个样本，这样可能更加准确一点。

|  |
| --- |
| #define MOST\_ACTIVE\_NULL 0 /\*未找到最活跃轴\*/  #define MOST\_ACTIVE\_X 1 /\*最活跃轴X\*/  #define MOST\_ACTIVE\_Y 2 /\*最活跃轴Y\*/  #define MOST\_ACTIVE\_Z 3 /\*最活跃轴Z\*/    #define ACTIVE\_PRECISION 60 /\*活跃轴最小变化值\*/   /\*判断当前最活跃轴\*/ static char is\_most\_active(peak\_value\_t \*peak) {  char res = MOST\_ACTIVE\_NULL;  short x\_change = ABS((peak->newmax.x - peak->newmin.x));  short y\_change = ABS((peak->newmax.y - peak->newmin.y));  short z\_change = ABS((peak->newmax.z - peak->newmin.z));    if (x\_change > y\_change && x\_change > z\_change && x\_change >= ACTIVE\_PRECISION) {  res = MOST\_ACTIVE\_X;  } else if (y\_change > x\_change && y\_change > z\_change && y\_change >= ACTIVE\_PRECISION) {  res = MOST\_ACTIVE\_Y;  } else if (z\_change > x\_change && z\_change > y\_change && z\_change >= ACTIVE\_PRECISION) {  res = MOST\_ACTIVE\_Z;  }  return res; }   /\*判断是否走步\*/ static void detect\_step(peak\_value\_t \*peak, slid\_reg\_t \*slid, axis\_info\_t \*cur\_sample) {  static step\_cnt = 0;  char res = is\_most\_active(peak);  switch (res) {  case MOST\_ACTIVE\_NULL: {  //fix  break;  }  case MOST\_ACTIVE\_X: {  short threshold\_x = (peak->oldmax.x + peak->oldmin.x) / 2;  if (slid->old\_sample.x > threshold\_x && slid->new\_sample.x < threshold\_x) {  step\_cnt ++;  }  break;  }  case MOST\_ACTIVE\_Y: {  short threshold\_y = (peak->oldmax.y + peak->oldmin.y) / 2;  if (slid->old\_sample.y > threshold\_y && slid->new\_sample.y < threshold\_y) {  step\_cnt ++;  }  break;  }  case MOST\_ACTIVE\_Z: {  short threshold\_z = (peak->oldmax.z + peak->oldmin.z) / 2;  if (slid->old\_sample.z > threshold\_z && slid->new\_sample.z < threshold\_z) {  step\_cntt ++;  }  break;  }  default:  break;  } } |

# 第二章 系统测试

软件只是基于计算机大系统的一部分。最终，软件要与其它系统成分（如硬件、信息和人）相结合，并执行一系列集成测试和确认测试。

2.1传感器灵敏度测试

2.1.1测试环境搭建

在硬件部分，运动检测使用手机传感器中的加速度传感器和方向传感器，使用安卓手机接口获取加速度传感器和方向传感器数据，用运动算法得到运动的类型、方向、距离。

在Android2.3 gingerbread系统中，google提供了11种传感器供应用层使用。其中我们需要用到加速度传感器和方向传感器，分别是：  
 #define SENSOR\_TYPE\_ACCELEROMETER 1 //加速度

#define SENSOR\_TYPE\_ORIENTATION 3 //方向

对硬件部分的测试设计为对传感器灵敏度的测试，传感器灵敏度包含两个部分，分别是影响传感器正常工作的干扰因素、以及正常情况下传感器最大感应范围。

搭建测试环境之前我们先简要介绍加速度传感器和方向传感器的原理：

安卓平台提供了2个传感器用于让我们判断设备的位置，分别是地磁场传感器（the geomagnetic field sensor）和方向传感器（the orientation sensor）。关于Orientation Sensor在官方文档中的概述里有这样一句话：

The orientation sensor is software-based and derives its data from the accelerometer and the geomagnetic field sensor. （方向传感器是基于软件的，并且它的数据是通过加速度传感器和磁场传感器共同获得的）

通过对方向传感器源码的阅读，发现**计算设备的方向基于旋转矩阵**，而旋转矩阵的数据源来自于从加速度传感器和地磁场传感器获取的参数值。

经过学习了解，影响传感器正常工作的主要因素为磁场。

2.1.2测试用例设计

对影响传感器正常工作的干扰因素的测试：

用例：设计强磁场干扰，确定影响结果精确度的磁场强度等因素，制定相应方案进行纠错，如受强磁场干扰时提示校准，或进行冗余检测等。

对正常情况下传感器最大感应范围的测试：

由于高精度加速度测量的不易实现，采用本地重力加速度与静止时加速度传感器三轴和加速度数值进行比对。

2.1.3测试结果

多次重复实验后，正常情况下加速度传感器静止时三轴和加速度数值变化范围为9.56~10.10m/s2。可知加速度传感器的精度时十分低的。

为提高精度，对相关论文进行了阅读学习，最终采用针对加速度传感器低精度的问题,提出一种基于手机多姿态获得MEMS加速度传感器系统误差的模型。通过求解不同姿态下的观测数据n维非线性方程组得到误差值,经该模型校正后，加速度数据精度相较单姿态模型的方法有较大提高。校正前静止状态下加速度传感器的3轴合加速度数值变化范围为9.56~10.10 m/s2，校正后静止状态下加速度传感器下3轴合加速度数值变化范围为9.78~9.86 m/s2 。

2.1.4手机加速度传感器系统误差校正方法研究

1．已有单姿态校正模型的不足：

目前已有的MEMS加速度传感器系统误差的计算方法相对粗糙,包括三星、小米等手机厂商在内,大多采用单姿态模型的方法进行加速度传感器系统误差的测量与计算。实现方法为将手机静止放置于水平面上,并获取加速度传感器3轴数据的平均值,单姿态模型的不足在于:①由于实际环境中无法快速确定真正的水平面﹐且也无法验证手机与其所搭载MEMS加速度传感器是否存在倾角。因而.x 、y轴的加速度数据会存在偏差。假设设备沿α轴存在0角度的偏差,可知y轴加速度存在g sin 0的偏差。当0=1°时,偏差为0.17 m/s2。由此可知，微小的角度偏差便会对x 、y轴的加速度数据产生较大影响。2 MEMS加速度传感器自身精度有限,因此MEMS加速度传感器理论上所能测得的重力加速度数据与真实的重力加速度数据会存在偏差﹐从而导致计算误差。

2．多姿态校正模型的建立：

2.1 MEMS加速度传感器原理及误差来源：

对MEMS加速度传感器而言,其主要由质量块及悬挂在加速度传感器周边框架结构上的弹性元件(弹簧)构成。当加速度传感器连同外界物体一起加速运动时,质量块受到惯性力的作用向相反的方向运动,加速度就可以通过质量块的位移检测出来[11-12]。其中弹簧质量块系统的固有频率越高,则获取的数据频率越高,而较低的弹性系数能获得更大的质量块位移﹐从而提高灵敏度与精度。基于MEMS传感器的测量原理,加速度传感器的3轴数据会存在由弹簧初始位移量引起的3轴初始误差﹐该初始误差是MEMS加速度传感器的系统误差。其在室内定位中的影响表现为:在步态分析与识别中由于该系统误差的存在,静止时3轴合加速度的数值并不等于真实的重力加速度,且不同姿态下的3轴合加速度数值也不尽相同,差距较大,对单步波形的波长判断存在影响。3轴加速度传感器在手机设备中对应的3轴方向如图1所示。x 、y、z轴分别代表沿手机短轴向右的方向、沿手机屏幕垂向上的方向,沿手机长轴向前的方向。

* 1. 误差模型构建：

文本

描述已自动生成首先写出误差方程：

式中xi、yi、zi;为手机不同姿态静止状态下的加速度传感器的观测数据。

文本, 信件

描述已自动生成利用差分法得到相邻方程组的共n-1个差分方程：

图片包含 表格

描述已自动生成得到系数矩阵B：



另设x=(△x ,△y ,△z)T，v=(1/2(x i2+y i2+Zi2 - xi+12- yi+12- Zi+12))T，通过使用平差方法，求解v= Bx＋l;其中l为随机误差向量,可求得向量x，解算出MEMS加速度传感器的系统误差。

2.2功能测试：

测试工具选择Robotium，测试用例主要包含开始认证功能测试，查看移动检测信息功能测试以及返回初始界面功能测试。具体设计如下：

开始认证功能测试用例:

点取“开始认证”按钮后跳转认证信息反馈界面，系统输出提示“认证成功”，显示按钮“查看移动检测信息”或者系统输出提示“认证失败”，显示按钮“返回初始界面”。

查看移动检测信息功能测试用例：

认证成功后点取“查看移动检测信息”按钮，跳转到移动检测界面。对于发送方，界面输出提示“正在检测移动发送传感器信息”；对于接收方，界面显示是否正在移动、当前加速度、到起始点的距离。

返回初始界面功能测试用例：

认证失败后点取“返回初始界面”按钮，跳转到初始界面，显示“开始认证”按钮。

2.3性能测试:

性能测试主要为负载测试和压力测试，测试工具选择Jmeter。

分别为“开始认证”功能点和“查看移动检测信息”功能点设计负载和压力测试用例，“开始认证”功能点的负载测试用例如表1，“查看移动检测信息”功能点的压力测试用例如表2，剩下两个用例与给出的两个完全类似。

表1“开始认证”功能点的负载测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例ID | 001 | | | | | |
| 业务名称 | 开始认证 | | | | | |
| URL | 本地http代理 | | | | | |
| 前置条件 | 测试用户可以进入软件首页 | | | | | |
| 测试步骤 | 进入app首页  点击“开始认证”按钮 | | | | | |
| 脚本操作 | | | | | | |
| 参数设置 | 无 | | | | | |
| 事务设置 | 事务名称 | | 起始位置 | | 结束位置 | |
| 开始认证 | | 在点取“开始认证”按钮之前 | | 点取按钮后跳转认证信息反馈界面 | |
| 场景设计 | | | | | | |
| 测试场景 | | 100个并发用户登录系统，在10秒内完成开始认证，持续40秒。测试记录系统的各项指标数据。 | | | | |
| 期望结果 | | | | | | |
| 测试指标 | | 平均事务响应时间 | | 95%响应时间 | | 事务成功率 |
| 期望结果 | | <=3s | | <=3s | | >95% |

表2“查看移动检测信息”功能点的压力测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例ID | 002 | | | |
| 业务名称 | 查看移动检测信息 | | | |
| URL | 本地http代理 | | | |
| 前置条件 | 认证成功后进入认证信息反馈界面 | | | |
| 测试步骤 | 进入认证信息反馈界面  点击“查看移动检测信息”按钮 | | | |
| 脚本操作 | | | | |
| 参数设置 | 无 | | | |
| 事务设置 | 事务名称 | | 起始位置 | 结束位置 |
| 查看移动检测信息 | | 在点取“查看移动检测信息”按钮之前 | 点取按钮后跳转移动检测页面 |
| 场景设计 | | | | |
| 测试场景 | | 500个并发用户登录系统，在1秒内完成“查看移动检测信息”，持续100秒。测试记录是否有错误响应出现。按二分查找算法，找出没有出现错误响应的最大并发用户数。 | | |
| 期望结果 | | 通过测试，获得系统能承受的最大并发用户数 | | |

# 第三章 知识技能学习情况

要完成本期的复杂工程问题，需要对于Java编程相关的知识有较好的学习和使用。在本次综设中我们学习了Java中对于传感器的使用，以及服务端建立Socket连接的方法。

手机传感器是手机内部设置的感应器，一般专门指的是安卓手机中的可用感应器。能够通过感应光照度、热量、距离等，来调节手机的工作状态，从而使人们能够更加方便地使用手机。一些比较高档的手机，常常会配备有光纤传感器、热力传感器、温度传感器等等，这些传感器可以感应到手机使用者的使用状态的变化，从而使手机做出自动的调整，不需要人们去进行手动的、人为的调整，这是非常方便的，而且这些调整一般都是在极短的时间之内完成的，不会浪费过多的时间，使人们第一时间感受到方便，这对于使用者来说，是非常有利的。

于是我们学习了如何用安卓手机接口获取加速度传感器和方向传感器数据以及如何用运动算法得到运动的类型、方向、距离，以及手机加速度传感器系统误差校正的方法。

基于socket传输数据为字节级，传输数据可自定义，数据量小；传输数据时间短，性能高；适合于客户端和服务器端之间信息实时交互；可以加密，数据安全性强的特点，我们也学习了服务端建立Socket连接的方法，通过Socket，我们能使用TCP/IP协议。用socket可以创建tcp连接，也可以创建udp连接，

在设备认证以及加密算法方面，我们还额外学习了对称加密算法AES（高级加密标准）和非对称加密算法RSA，由于AES具有计算量小、加密速度快、加密效率高，但安全性低的特点，以及RSA安全性高，但运算速度较慢的特点，最后确定了在设备认证时将AES的密钥通过RSA加密后发给对方，再用AES算法进行加密和传输，做到了对通信数据的加密，保证了通信的安全性以及高效性。

在app的界面绘制方面，我们也学习了能够跨平台的编程解决方案Flutter。其使用Dart作为编程语言。它支持跨Android、iOS等平台开发软件。考虑到移动端平台整体特性十分接近，硬件属性差别不大，Flutter提供的跨平台特性应该足以让我们完成软件的开发工作，并且比较方便地实现同时在Android和iOS两大平台上运行，做到真正的低门槛、易用。

# 第四章 分工协作与交流情况

根据本次复杂工程问题的具体设计要求，团队中的几位成员分工如下：

冉江皓：作为组长，统筹工程问题的解决和具体安排，参与算法设计

邓亚辉：开发涉及到通信和检测算法的部分，编写前后端程序

官宏：进行系统测试，验证系统完整性和可用性

冯一凡：设计界面，编写检测端和监控端程序

罗驰宇：实现前后端功能，编写检测端和监控端程序

# 参考文献

1. http://cdmd.cnki.com.cn/Article/CDMD-10013-1015586114.htm 基于智能手机传感器的空间位移测量软件的设计与实现
2. https://blog.csdn.net/Dancer\_\_Sky/article/details/81504778 基于三轴加速度传感器的计步算法

# 致谢

本报告的工作是在我们的指导教师陈大江老师的悉心指导下完成的，在定期的进度交流中，老师对我们的课题提出了诸多建议，也给了我们很多帮助。在此，我们小组的全体成员向陈大江老师表示感谢。