# 中国科学技术大学

# “科学与社会”新生研讨课研究报告

报告题目：基于 Arduino 的计步器设计

小组组长：林浩南

小组成员：张雨辰　胡佳琪　周靳裔

导师姓名：符传孩

2018 年 5 月 28 日

# 一、研究小组成员及其承担的主要工作

| 学号 | 姓名 | 所在学院 | 在研究和报告撰写中承担的主要工作 |
|---|---|---|---|
| PB17000156 | 林浩南 | 少年班学院 | 总体方案设计，搭建电路，提出和编写数据预处理代码、基准范围判定算法，撰写报告 |
| PB17000148 | 张雨辰 | 少年班学院 | 搭建电路，提出和编写峰值判定算法 |
| PB17000151 | 胡佳琪 | 少年班学院 | 编写数据预处理代码，试验过程中对数据进行处理，后期调试 |
| PB17000149 | 周靳裔 | 少年班学院 | 硬件部分设计，初期控制器和传感器选型，后期调试 |

# 二、进度安排

2017 年 12 月 21 日 定题
2018 年 1 月 讨论出核心算法，完成硬件部分设计与控制器、传感器的选型
2018 年 3 月 搭建电路，完成数据预处理代码和范围判定部分代码
2018 年 5 月 完成趋势判定代码，调试与测试，撰写报告

# 三、摘要

随着移动通信技术的发展，传统互联网已经在向移动互联网迁移，智能穿戴设备已成为一个关注热点，在提高人们生活品质、促进生活方式智能化方面将会起到很重要的作用。而其中最常见的一个功能即是步数检测。本文出于探索如何用传感器来获取并处理信息的目的，展开了对记步系统的设计。本文主要根据人体运动过程中的体态特征，设计了基准范围判定和趋势判定相结合的新算法，提出了基于 Arduino 和 mpu6050 的高精度记步方案。最终系统的记步准确率达 92.3%。

**关键词：**智能穿戴设备；计步器；传感器

# 四、研究报告

目　录

# 1 背景和目标

## 1.1 背景

随着社会经济的发展，人民生活水平正逐渐提高，伴随而来的是之前很少出现的各种慢性疾病。这些慢性疾病很多是因为缺乏必要的锻炼造成的。而步行，是锻炼中较为科学有效的一种方式。越来越多的人把步行锻炼变成了一种生活方式。人们在通过步行锻炼的同时，往往想知道自己锻炼的进度与成效，于是，各种智能穿戴设备应运而生，计步器便是其中的一种。

计步器最早是由意大利的伦纳德·达芬奇酝酿的，但现存的最早的计步器是在达芬奇之后 150 年，即 1667 年制作的。

## 1.2 分类及实现方法

现在市面上的计步器大体分两种：机械式计步器与电子式计步器。机械式计步器会内置一个机械装置，里面通常会有一个弹性小球或弹簧片，当佩戴者进行运动时，小球有规律的震动会产生规律性的脉冲，而计步器里面的判决器会通过判断电子脉冲的个数来实现计步功能。这类计步器的优点是成本低，缺点是往往受运动幅度的影响较大，灵敏度较低，因此机械式计步器在市场上基本被淘汰。

电子式计步器通常由控制器和三轴加速度传感器构成。先由传感器获取加速度信息，然后传递给控制器。控制器将数据简单地处理后，通过记步算法进行判定。主流的记步算法有以下几种：[1][2]

（1）波峰检测算法

根据人行走时加速度数据的周期性波动特点，峰值检测算法通过统计数据图像中峰值和谷值的数目来实现记步，当检测到一组连续的波峰和波谷时，便视为人走了一步。波峰和波谷的寻找一般用一组相邻数据的斜率正负突变作为判断条件。这种算法的优点是计算量小，但是缺点是易受高频扰动的干扰，因此这种算法对滤波的要求很高。有的文献[3]提出了一种自适应波峰检测算法，其核心思想是根据步行加速度峰值统计数据，用不同的加速度阈值和时间阈值作为计步判断的依据。

（2）动态阈值算法

动态阈值算法的核心思想是用动态可变的阈值来作为记步的判定条件。每次数据有更新时，系统会取一定时间窗口内的数据进行统计，计算得到新的阈值，然后将下一次的数

据拿来判断。这种算法的主要缺点十分明显：由于每一次的阈值都受之前数据影响，所以当人的运动状态突然改变时，会有一个过渡期，在过渡期内步数无法得到有效判定，产生较大的误差。[4]

（3）自相关算法

人连续运动时，加速度成有规律的周期性变化；而人在非连续运动时加速度不具有这样的规律。自相关算法就是利用当前数据周期和前一个数据周期的自相关系数来判断步数[5]。试验阶段会先通过数据统计的方式比较静止状态和步行状态自相关系数差异，然后选择合适阈值作为区分依据。这种算法中涉及到大量乘除、平方和开方运算，因此计算量很大，时间效率低。

### 1.3 目标

基于以上背景，**本文尝试设计一种新的记步算法，将范围判定和趋势判定相结合，在避免过大计算量的同时，有效保证准确率。**本文设计的计步器是一种基于 Arduino 和 MPU6050 的高精度计步器。以 ATmega328 低功耗微控制器作为核心处理器，数字输出三轴加速度传感器 MPU6050 作为运动检测模块。计步器利用传感器和一定的算法来精确检测真实的步伐。我们的目标是在正常步行过程中，记步准确率能够达到 90%。

## 2 系统总体设计方案

### 2.1 人体运动模型分析

人在行走过程中，左右腿交替支撑地面。通常，人体步行步态周期主要由两部分构成：站立期和迈步期。以右脚为例，站立期是指右脚与地面接触这段时间。而接下来的迈步期是指右脚离开地面的这段时间。之后便又是站立期与迈步期的交替，如此循环往复。

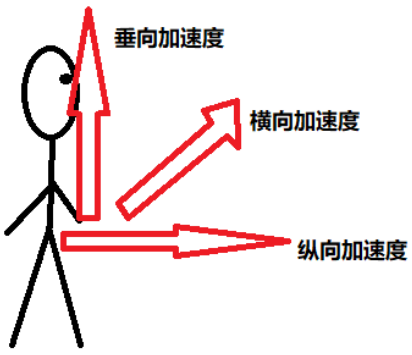


图 1 人体运动模型分析示意图

如图 1，人在正常行走过程中，人体的加速度往往会产生三个方向的分量，分别是纵向加速度、垂向加速度和横向加速度。在人体行走过程中，当脚蹬地离开地面时，脚会给地面一个作用力，同时也会受到地面的反作用力，此时人体的垂向加速度逐渐增大。当脚要达到最高点时，脚的垂向速度值最小，垂向加速度值达到最大，然后接着脚会向下移动，垂向加速度开始逐渐减小，最终脚落地时，垂向加速度减少到最小值，然后便进入到下一个步伐周期。在人每迈出一个步伐周期的过程中，人体的垂向加速度值都会像正弦函数一样周期性波动。一个周期对应着一步。

### 2.2 滤波算法

正常人步行的步频为 0.5Hz-2.0Hz[6]，所以当采样率远大于 2Hz 时，就会引入高频干扰，我们要通过滤波来滤除加速度数据中的高频干扰。

### 2.3 记步算法

目前计步器采用的常见记步算法有波峰检测算法、动态阈值算法、自相关算法(详见 1)等等，我们采用的是基准范围判定和趋势判定相结合的算法。

### 2.4 原理设计

根据人体在一个步伐周期内加速度的变化，采用加速度传感器对人体运动的加速度信息进行采集，对加速度信号作预处理，再由控制器通过计步算法准确计算出人体实际行走的步数。

根据系统原理，本文设计的原理框图的主要结构是微控制器与传感器的结合。

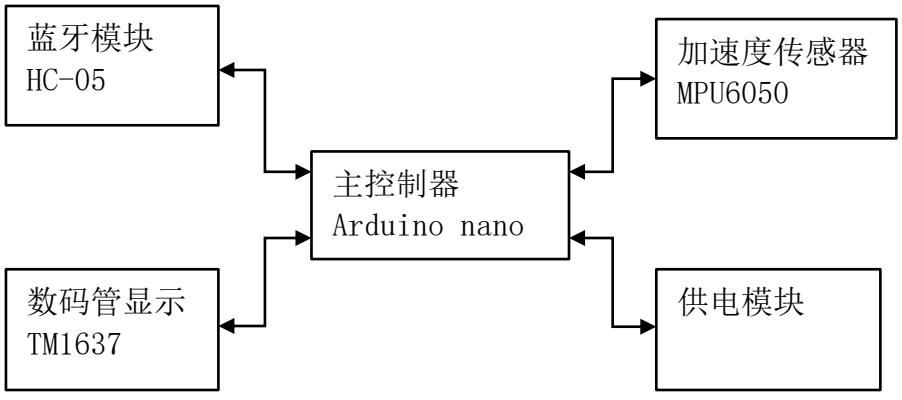


图 2 系统原理框图

主控制器主要负责协调各个模块之间的调度。三轴加速度传感器模块主要采集人体在步伐周期内的加速度信息。蓝牙模块主要实现数据的传输。显示模块采用的是 TM1637 四位数码管显示，用于步数显示。

# 3 系统技术设计

## 3.1 系统硬件设计

### 3.1.1 控制器与外设

本系统的硬件设计部分主要包括 Arduino 微控制器和由 MCU 控制的各种外设：

- IIC 接口的 MPU6050 三轴加速度传感器
- TM1637 四位数码管显示模块
- HC-05 蓝牙模块
- 供电控制电路

### 3.1.2 控制器选择

（1）Arduino 系列

Arduino 是一款便捷灵活、方便上手的开源电子原型平台。硬件部分可以用来做电路连接；软件部分 Arduino IDE 基于 processing IDE 开发，可以用 C++编写控制程序。[7]

（2）STM32 系列

STM32 系列 32 位闪存微控制器基于 Arm® Cortex®-M 处理器，主要特点有高性能、低成本、低功耗，适合嵌入式应用专门设计。[8]

Arduino 相较于 STM32 来说开发速度快，易于实现，有着足够的灵活性。而 STM32 则有更高的性能，二者可谓各有优劣。本文所设计的计步器对控制器的处理能力要求较低，Arduino 即满足需求，所以选用 Arduino nano 作为控制器。

### 3.1.3 传感器选择

（1）压电式加速度传感器[9]

压电式加速度传感器又称压电加速度计。它也属于惯性式传感器。压电式加速度传感器的原理是利用压电陶瓷或石英晶体的压电效应，在加速度计受振时，质量块加在压电元件上的力也随之变化。当被测振动频率远低于加速度计的固有频率时，则力的变化与被测加速度成正比。

（2）压阻式加速度传感器[9]

基于世界领先的 MEMS 硅微加工技术，压阻式加速度传感器具有体积小、低功耗等特点，易于集成在各种模拟和数字电路中，广泛应用于汽车碰撞实验、测试仪器、设备振动监测等领域。

（3）电容式加速度传感器[9]

电容式加速度传感器是基于电容原理的极距变化型的电容传感器。电容式加速度传感器是比较通用的加速度传感器。通常应用于安全气囊，手机移动设备等。电容式加速度传感器采用了 MEMS 工艺，在大量生产时保证了较低的成本。

每种加速度传感器都有优缺点。电容型加速度传感器通常设计成临界阻尼或过阻尼状态，适合做低频测试。其成本低，适合汽车、消费品等大批量的应用，但精度不高。压阻式传感器耐用，频率响应好，能够测静态加速度，进而准确的计算速度和位移，但信噪比不高，结构复杂。压电式传感器优点是结构简单，灵敏度高，信噪比高，缺点是会有电荷泄露。本文综合考虑各种加速度传感器的优缺点，考虑到体积、精度等问题，最终采用压电式加速度传感器 MPU6050。

3.1.4 蓝牙方案

采用 hc05 透传模块

3.1.5 供电方案

将 3.7v 锂电池升压至 5v 给系统供电

3.1.6 系统实物图

基于以上硬件设计，做出计步器系统的实物图如下：

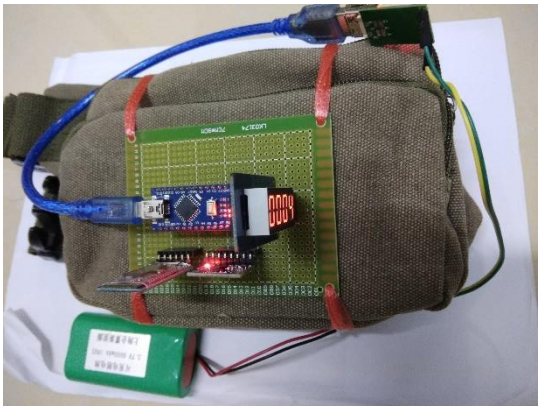

图 3 计步器系统的实物图

## 3.2 系统软件设计

### 3.2.1 数据预处理

为了减少高频扰动的影响，更好地获取有用的数据，提高后期利用数据的效率，我们要对数据进行预处理。

（1）三轴归一化处理

为了减少加速度传感器与人体相对方向发生变化带来的影响，我们采取三轴归一化处理。

通过 2.1 的分析可知，只有垂向加速度会有较大的变化。我们可以考虑将加速度传感器的方向保持和身体的相对固定，让传感器的三轴分别对应分析中的三个方向，这样有一轴的加速度变化较大，记为有效轴，而另两轴的加速度变化则较小。进而我们可以利用有效轴的数据来进行分析和步伐判断。但是，通常情况下，加速度传感器并不能被很好的固定。在记步过程中，若有效轴发生变化，则会造成计数点的丢失。为了避免上述情况，我们采用一种更为稳妥的方式来提高稳定性：三轴归一化处理。我们采用求三个数据平方和的二次方根，即 2-范数的方法。

$$\|a\| = \sqrt{a\_xAxis^2 + a\_yAxis^2 + a\_zAxis^2}$$

式中$a_xAxis$为$x$轴方向加速度值，$a_yAxis$为$y$轴方向加速度值，$a_zAxis$为$z$轴方向加速度值。

（2）滤波预处理

我们采用滤波预处理来滤除加速度数据中的高频抖动。

很多文献卡尔曼滤波算法，但是卡尔曼滤波算法是一种把多传感器的数据融合的最优估计算法，本文只使用了一种传感器，并不存在数据融合，所以本文采用一阶滞后滤波算法。公式如下：

$$y(n) = (1-\alpha)\cdot x(n) + \alpha \cdot y(n-1)$$

其中$x(i)$为输入信号，$y(i)$为输出信号，α是滤波系数，取 0 到 1 之间的值。我们选取$\alpha = 0.85$。

### 3.2.2 加速度波动基准值的获取

加速度传感器静止时测得的数据近似为重力加速度，而行走时加速度数据会在这个静止时数据上下波动，我们称静止时测得的数据为加速度波动基准值，简称波动基准值。

因为受到环境的各种影响，每一次系统初始化后，波动基准值并不会完全相同。所以，我们在系统初始化时，会先测量得出波动基准值。具体的方法是：人静止不动，传感器是在最初几秒的时间内进行取样，然后对这一系列的数据进行处理，得到波动基准值。下面我们以一组一般数据为例进行分析。

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10.90 | 10.91 | 10.89 | 10.90 | 10.90 | 10.91 | 10.90 | 10.89 | 10.89 | 10.89 |
| 10.92 | 10.90 | 10.91 | 10.92 | 10.91 | 10.89 | 10.90 | 10.91 | 10.90 | 10.91 |
| 10.89 | 10.89 | 10.91 | 10.91 | 10.89 | 10.92 | 10.91 | 10.90 | 10.92 | 10.89 |
| 10.89 | 10.91 | 10.92 | 10.92 | 10.92 | 10.89 | 10.91 | 10.92 | 10.91 | 10.90 |

**表 1 一组人静止时的加速度典型数据**

表 1 中数据在人静止时测得（加速度传感器放置在腰部位置），采样频率 30Hz，数据已经过三轴归一化处理与滤波预处理。

计算可得上述数据的平均值$\bar{a}$、标准差$\sigma_x$和极差$max - min$：

$$\bar{a} = 10.904m \cdot s^{-2}$$

$$\sigma_x = 0.011m \cdot s^{-2}$$

$$max - min = 0.03m \cdot s^{-2}$$

可见，这组数据的离散程度很小。这个结果具备一般性。试验过程中我们发现，一系列静止时的加速度数据（预处理后）的极差通常相当小，满足：

$$max - min < O(0.01m \cdot s^{-2})$$

式中，$max$表示这一系列数据中的最大值，$min$表示最小值。

$max$与$min$相差如此之小，以至于对上述数据过于复杂的处理是没有意义的，所以我们取

$$average = \frac{max + min}{2}$$

作为波动基准值。

3.2.3 计步器主程序流程

如图 4，系统启动后，主控制器、数码管模块、加速度传感器相继完成初始化。之后系统进行波动基准值的获取。当收到开始信号后，系统正式开始记步。控制器首先对传感器收集的数据进行预处理，然后判断是否满足记步条件。若满足，则步数更新。之后开始准备下一次的数据获取、处理、判定。这样的过程不断循环，直到系统收到结束信号。

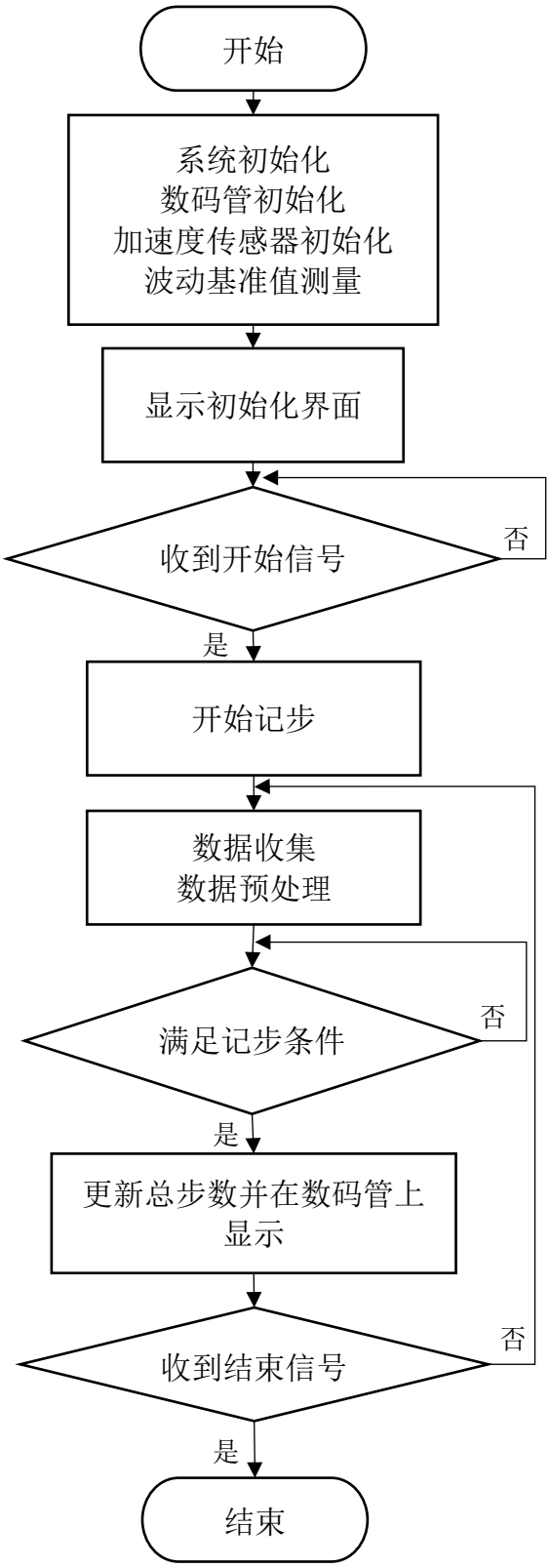


**图 4 计步器主程序流程图**

# 4 计步器系统核心算法实现

## 4.1 人正常行走时的步态分析

目前计步器采用的常见记步算法有波峰检测算法、动态阈值算法、自相关算法等等，本文结合以上算法，提出一种新的合理算法。根据 2.1 的分析，人在行走过程中，腰部会有一个上下起伏的过程，如图 5

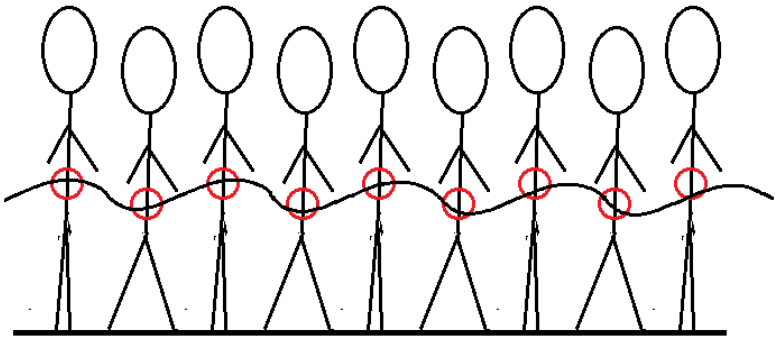

图 5 人行走时的腰部起伏示意图

可以看出，在人正常行走时，腰部的高度变化大致成周期函数，则其二阶导数即垂向加速度也大致成周期函数。为了验证我们的推测，我们绘制了图 6 所示的加速度-时间图像。

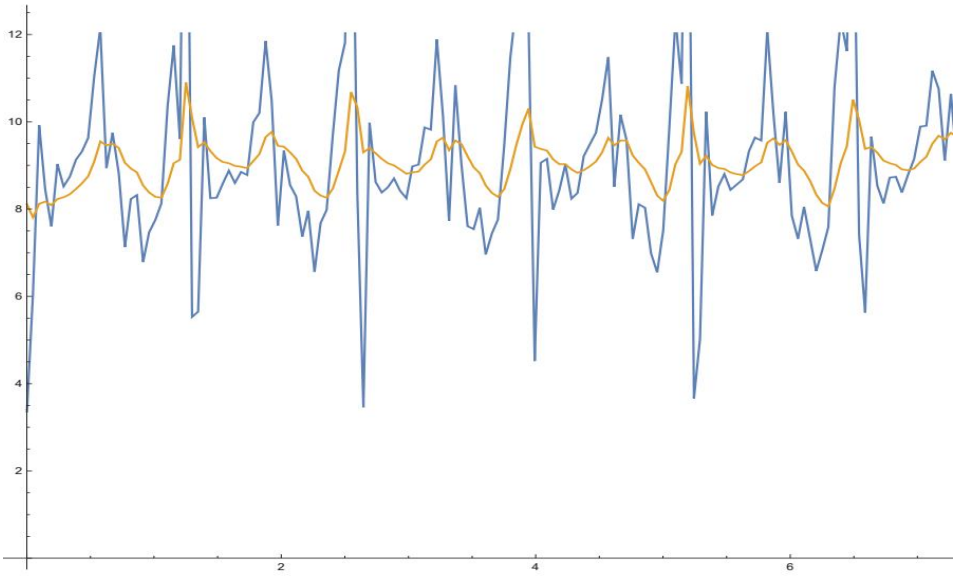


图 6 人正常行走时的加速度-时间图像

图中蓝线为原始数据，黄线为取 3.2.1 滤波公式中$\alpha = 0.85$得到的图像，可以看出，图像所呈现的大致周期性与我们的分析相符。

## 4.2 基准范围判定算法

在每个周期中加速度-时间图形会经过重力加速度$g$处 2 次。理论上，我们只需要检测图像经过$g$值（称为判定基准值）的次数，就能够得到人行走的步数。注意，这里的判定基准值与 3.2.2 中的波动基准值不同。波动基准值是指数据波动范围的近似中心，一定时间内是定值，而判定基准值只是我们选择的一个标准。当然现在我们取判定基准值为波动基准值。

但是，这样的分析过于理想化。事实上，因为我们的数据点并非连续的，所以并不会有某个数据点恰好落在判定基准值上，因此我们需要一个范围（称为判定基准范围）来进行判定。当以下两式成立时，我们认为图像通过判定基准。

$$\begin{cases} a < average + sensitivity \\ a > average - sensitivity \end{cases}$$

式中$a$表示经过数据预处理得到的加速度，$average$为波动基准值，$sensitivity$表示判定范围的半径。

**4.3 仅依赖基准范围判定算法所面临的问题**

进行到这里，我们仍然面临着两个问题：1）在人静止不动时，加速度始终在判定基准范围内，我们将会陷入人静止时计步器仍然在计数的窘境；2）虽然加速度已经经过了滤波处理，但是无法完全排除偶然扰动情况。为了解决第一个问题，我们考虑将判定基准范围向下平移或为步伐判断添加新的判据。经过试验后我们发现平移判定基准范围的平移距离较难确定，具体表现为：平移距离过小无法解决静止计数问题；平移距离过大则无法对较轻的步伐进行有效计数。

**4.4 趋势判定算法**

为了解决上述问题，我们增加新的判据：利用趋势判定算法。我们采样的频率取为 30Hz（利用计时器中断实现），人快速行进时步频为 2Hz 左右，则加速度的每个周期至少有 15 个数据点，即在每次加速度上升或下降时，都至少经过 5 个数据点。据此我们采用以下判断方法。

$$\begin{cases} a(n-3) < a(n-2) \\ a(n-3) < a(n-1) \\ a(n-3) < a(n) \end{cases}$$

式中$a(n)$表示第$n$次的数据点。若上式成立，则认为数据图像刚刚经过了一个上升的趋势，为了排除人静止时随机扰动导致上式恰好成立，我们将上式修改为

$$\begin{cases} a(n-3)-a(n-2)<0 \\ a(n-3)-a(n-1)<0 \\ a(n-3)-a(n)<-gap \end{cases}$$

相应地，

$$\begin{cases} a(n-3)-a(n-2)>0 \\ a(n-3)-a(n-1)>0 \\ a(n-3)-a(n)>gap \end{cases}$$

若上式成立，我们认为数据图像刚刚经过了一个下降的趋势。当图像经过一个上升和下降的趋势后，我们说下一次记步是有效的。保险起见，我们采用一个变量来记录现在距离上一次有效记步的时间，当间隔大于$interval = 500ms$时才允许下一次记步。

**4.5 每一次记步判断的具体过程**

最终，我们每一次判断记步的具体过程如下：到达数据采样时间后，数据被采样和预处理。若距离上一次有效记步时间大于$interval$，则继续下面的判断。然后判断数据点是否在基准范围内。若是，则继续判断是否满足趋势判定条件。若是，则总步数$sum$加 1。然后本次对于是否计数的判断结束。若上面有任何一次判断为否，则$sum$不变，直接跳出过程。

流程图如下：

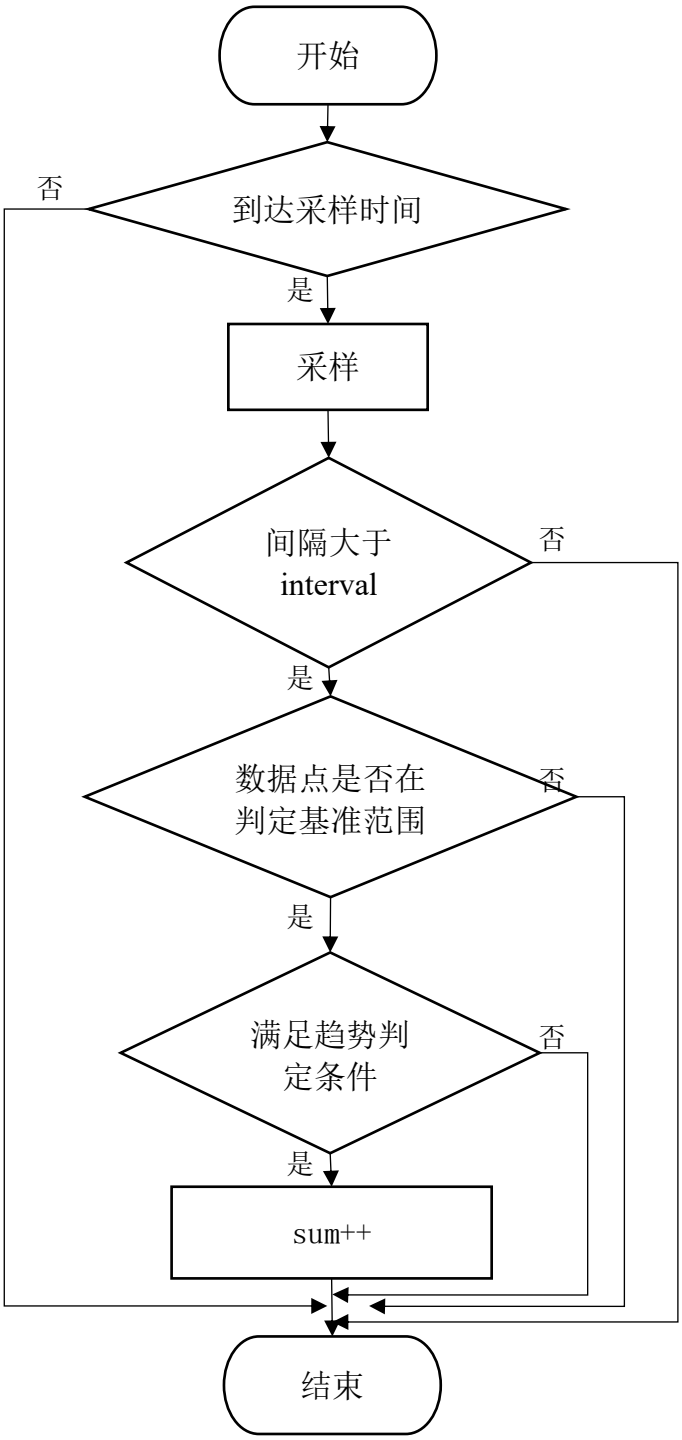


图 7 记步具体过程流程图

# 5 产品测试与实验结果分析

## 5.1 测试过程

在测试过程中，由一名实验者在腰部正前方佩戴计步器，然后正常行走，由计步器进行计数。我们通过有序地改变关键参数的值来探究关键参数对计步器准确度的影响，进而选取合适的参数。

## 5.2 关键参数值的影响探究与选取

在基准范围判定和趋势判定中，对计步器的准确程度起着较大作用的参数分别是$sensitivity$和$gap$。（$sensitivity$是范围判定中的范围半径，而$gap$是趋势判定中的$a(n-3)$和$a(n)$的最小距离。）因此我们认为$sensitivity$和$gap$是需要调试的关键参数。

为了更好的判断参数值的优劣，我们引入记步准确率

$$\eta = (1 - \frac{|N - N_0|}{N_0}) \times 100\%$$

其中，$N_0$为真实走的步数，$N$为计步器记录的步数

我们先来预测一下记步准确率和两个变量的关系。$sensitivity$增大时，即范围判定的半径增大，意味着记步更加灵敏，反之，记步灵敏度降低。$gap$增大时，意味着$a(n-3)$和$a(n)$的之间需要更大的距离，即灵敏度降低，反之，灵敏度升高。而一定范围内，高灵敏度往往意味着高记步准确率,但是过高的灵敏度反而会降低记步准确率。

我们将$sensitivity$在 0.1 到 1.5 中间隔 0.2 取值，将$gap$在 0.2 到 0.8 中间隔 0.2 取值，测试的过程较为繁琐，数据较多，这里仅取其中的一部分典型数据来进行观察分析。

步频约为 1Hz 时的记步准确率如下表

| η | | sensitivity | | |
|---|---|---|---|---|
| | | 0.3 | 0.7 | 1.1 |
| gap | 0.2 | 85% | 97% | 96% |
| | 0.4 | 82% | 99% | 98% |
| | 0.6 | 70% | 95% | 93% |

**表 2 步频为 1Hz 时的记步准确率**

将上面的数据绘制成折线图如下

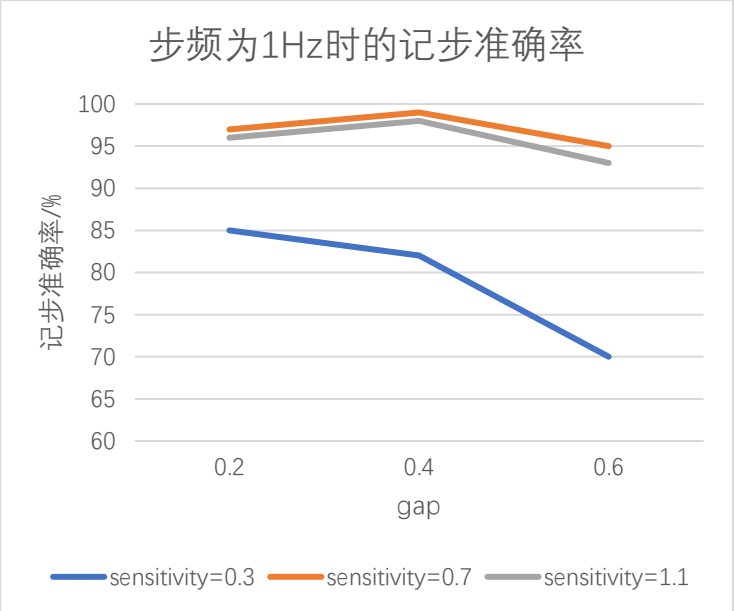


图 8 步频为 1Hz 时的记步准确率

步频约为 2Hz 时的记步准确率如下表

| η | | sensitivity | | |
|---|---|---|---|---|
| | | 0.3 | 0.7 | 1.1 |
| gap | 0.2 | 85% | 97% | 98% |
| | 0.4 | 81% | 92% | 92% |
| | 0.6 | 75% | 83% | 83% |

表 3 步频为 2Hz 时的记步准确率

绘制折线图如下：

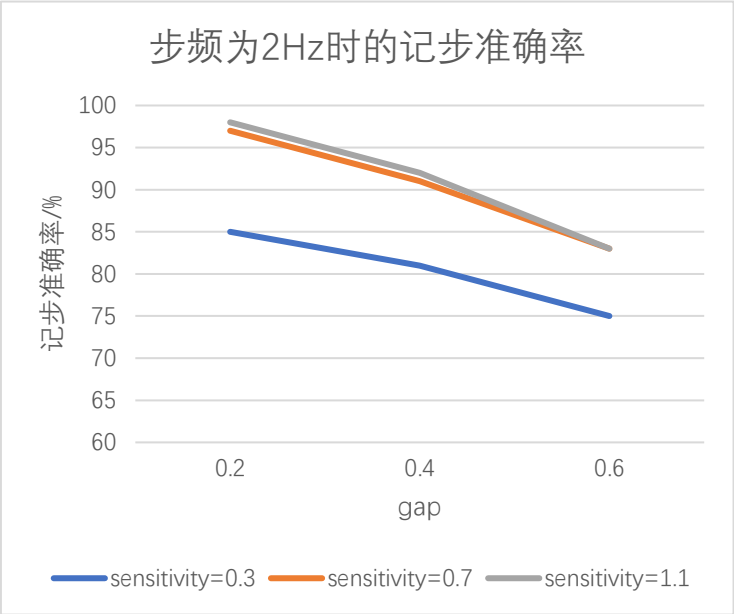


**图 9 步频为 2Hz 时的记步准确率**

表 2 和表 3 中的数据每一组均是取$N_0 = 100$进行实验，测两组数据算出$\eta$取其平均值得到。

观察图 8 和图 9 可以看出，我们之前的分析大体上是正确的，即一定范围内的记步准确率随着$sensitivity$的增大而增大，随着$gap$的增大而减小。但是，这并不意味着，我们应当取尽可能大的$sensitivity$和尽可能小的$gap$。观察图 8，在步频为 1Hz，$sensitivity = 1.1$时，当我们取$gap = 0.2$时，记步准确率反而比$gap = 0.4$时低，这是因为计步器的灵敏度过高，可能误把一些微小扰动记成一步，引入较大误差。

经过不断地尝试，我们最终选取$sensitivity = 0.7$，$gap = 0.55$。取这个解能保证相当的记步准确率。下面我们针对这个解作更加详细的测试。

### 5.3 记步准确率的测量

我们选取$sensitivity = 0.7$，$gap = 0.55$。经过数次测试绘制出如下表格：

| 组数 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 步频 | 1Hz | | | 2Hz | | |
| 步数测量值 | 102 | 101 | 103 | 86 | 89 | 85 |
| 步数真实值 | 100 | 100 | 50 | 100 | 100 | 100 |
| 记步准确率 | 98.0% | 99.0% | 97.0% | 86.0% | 89.0% | 85.0% |

| | |
|---|---|
| 记步准确率 | 92.3% |

**表** 4 sensitivity=0.7,gap=0.55 **时的记步准确率**

92.3%已经达到了我们开始所设立的目标。可见$sensitivity = 0.7$，$gap = 0.55$是一组相当好的解。

# 6 总结

智能穿戴设备一直是近年来人们日常生活中关注的热点。而计步功能又是智能穿戴设备中必不可少的功能。本文首先分析人体实际步行过程中步态运动的特征，找到加速度的变化规律。之后在进行器件选材时，首先考虑该器件是否能支持系统所需基本功能，然后综合考虑器件的体积大小、稳定性、操作的难度与灵活性。接下来便是软件设计部分，这部分首先对三轴加速度数据进行归一化处理，然后使用滤波等去噪方案，去除噪声对加速度信号的干扰，最后提出了一种合理算法，结合基准范围和趋势进行步伐判断。经过多次实地步伐测试，本计步器系统的计步准确率达 92.3%。

**项目主要创新点：**

（1）设计了一种基于基准范围检测和趋势检测的相结合的改进计步算法。

（2）采用的多维数据归一化处理有效避免了因计步器佩戴方向不同对实验造成的干扰。

主要缺陷：

（1）硬件的集成度较低

（2）软件功能还不够完善

（3）行进速度较快时记步准确率较低

待改善的方面：

（1）增加动态阈值，对多场景进行适应

（2）增强对不同佩戴位置的记步优化

（3）利用 Arduino 提供的丰富接口来实现多样化的实用功能

（4）提高整个系统的集成度

**参考文献**

**[1]张童飞.基于 BLE4.0 的计步系统设计与实现设计与实现[D].南京:东南大学,2016.**

[2]刘兵. 基于 ADXL362 的低功耗计步器的设计与实现[D].大连:大连理工大学,2017.
[3]彭丁聪.卡尔曼滤波的基本原理及应用[J].软件导刊,2009,(11):32-34.
[4]谢如花.步数检测方法及在手腕式计步器中的应用研究[D].兰州:兰州交通大学,2013.
[5]陈国良，张言哲，杨洲.一种基于手机传感器自相关分析的计步器实现方法[J].中国惯性技术学报，2014,22(6):794-798.
[6]陈银溢.基于 CC2541 和 LIS3DSH 的计步器设计[J].机械过程与自动化,2014,(6):96-98.
[7]*What is Arduino?*. Retrieved May 26, 2018 from: https:// www.arduino.cc/en/Guide/Introduction?setlang=en
[8]*STM32 32-bit ARM Cortex MCUs*. Retrieved May 26, 2018 from: http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html
[9]捷配电子市场网.加速度传感器工作原理及分类[EB/OL].http://www.dzsc.com/data/2016-11-22/111093.html,2016-11-22.
[10]卢文. 基于STM32和LIS3DSH的高精度计步器的研究与设计[D].宜昌:三峡大学,2016.

## 附录

程序源码

```
1. #include <MPU6050.h>
2. #include <MsTimer2.h>
3. #include <TM1637.h>
4. MPU6050 mpu;
5. #define CLK 3
6. #define DIO 2
7. TM1637 tm1637(CLK, DIO);
8. const float alpha = 0.85;//滤波时的因子（代表原数据占比）
9. const float sensitivity = 0.7, skewing = 0; //有效值范围半径，基准偏离值
10. const int interval = 500;//计次间隔
```

```
11. const int frequence = 33;//测量取样间隔（不完全准确）
12. const float gap = 0.55;
13. float average;//重力加速度
14. float trana;
15. int sum = 0;
16. const int inittime = 2000;
17. char c;
18. boolean shouldmeasure = 0;//是否可以采样
19. double sa = 0, sa1 = 0, sa2 = 0, sa3 = 0;
20. boolean flag = 1;
21. boolean abled = 0;//有效步数
22. void setup() {
23.   MsTimer2::set(frequence, measure_sign);
24.   Serial.begin(9600);
25.   while (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G)) {
26.     Serial.println("Could not find a valid MPU6050 sensor, check wiring!");
27.     delay(500);
28.   }
29.   tm1637.init();
30.   tm1637.set(BRIGHT_TYPICAL);
31.   Serial.println("OK!");
32.   pinMode(13, OUTPUT);
33.   while (1) {
34.     if (Serial.available()) {
35.       c = Serial.read();
36.       if (c == 32) { //收到开始测重力加速度的信号
37.         Serial.println("ready to start?");
38.         digitalWrite(13, HIGH);
```

```
39.          delay(1000);
40.          digitalWrite(13, LOW);
41.          goto sign2;
42.        }
43.      }
44.    }
45.    sign2: average = gravity_get(); //测量重力加速度
46.    digitalWrite(13, HIGH);
47.    delay(1000);
48.    digitalWrite(13, LOW);
49.    while (1) {
50.      if (Serial.available()) {
51.        c = Serial.read();
52.        if (c == 32) {
53.          Serial.println("start now!");
54.          digitalWrite(13, HIGH);
55.          goto sign4;
56.        }
57.      }
58.    }
59.    sign4:;
60.    MsTimer2::start();
61.}
62.void loop() {
63.    boolean  measurenow;//用来读取 shouldmeasure，这里 shouldmeasure 不宜直接放入 if 语句，
     因为有中断函数对其写入
64.    static int timer = 0;
65.    noInterrupts();//关中断
```

```
66.   measurenow = shouldmeasure;
67.   shouldmeasure = 0;
68.   interrupts();//开中断
69.   static boolean sumabled = true;//是否可以计数
70.   float a;
71.   static float smootheda = trana;
72.   if (measurenow == 1) { //可以采样
73.     Vector normAccel = mpu.readNormalizeAccel();
74.     a = sqrt( normAccel.XAxis * normAccel.XAxis + normAccel.YAxis * normAccel.YAxis +
    normAccel.ZAxis * normAccel.ZAxis );//三轴归一化处理
75.     smootheda = ( alpha * smootheda ) + ( 1 - alpha ) * a;//滤波处理
76.     sa3 = sa2; sa2 = sa1; sa1 = sa; sa = smootheda;
77.     if ((sa3 > sa2) && (sa3 > sa1) && (sa3 - sa > gap) && (flag == 1)) { //下降趋势
78.       abled = 1;
79.       flag = 0;
80.     }
81.     if ((sa3 < sa2) && (sa3 < sa1) && (sa - sa3 > gap))flag = 1; //上升趋势，变量 flag
    表示经过上升趋势
82.     measurenow = 0;
83.   }
84.   if (sumabled == false) { //即不可计数
85.     timer = timer + frequence ;
86.     if (timer >= interval) { //interval 表示允许的两次计数的最小间隔
87.       sumabled = true;
88.     }
89.   } else { //如果可以计数
90.     if (a < average + sensitivity - skewing && a > average - sensitivity - skewing) {
91.       if (abled == 1) {
```

```
92.          sum++;
93.          sumabled = false;
94.          timer = 0;
95.          abled = 0;
96.          Serial.println(sum);
97.          tm1637.display(3, sum % 10);
98.          tm1637.display(2, (sum % 100) / 10);
99.          tm1637.display(1, (sum % 1000) / 100);
100.             tm1637.display(0, sum / 1000);
101.           }
102.         }
103.       }
104.       if (Serial.available()) {
105.         Serial.println("in");
106.         c = Serial.read();
107.         if (c == 32) { //收到停止信号
108.           MsTimer2::stop();
109.           Serial.println("stop!");
110.           digitalWrite(13, LOW);
111.           while (1) {
112.             if (Serial.available()) {
113.               c = Serial.read();
114.               if (c == 32) { //收到开始信号
115.                 Serial.println("start now!");
116.                 digitalWrite(13, HIGH);
117.                 timer = 0;
118.                 sum = 0;
119.                 sumabled = true;
```

```
120.                MsTimer2::start();
121.                goto sign3;
122.              }
123.            }
124.          }
125.        }
126.      }
127.      sign3:;
128.    }
129.    void measure_sign() {
130.      shouldmeasure = 1;
131.    }
132.    float gravity_get() { //测量重力加速度的函数
133.      MsTimer2::start();
134.      Vector normAccel;
135.      float a;
136.      float smootheda = 0, maximum = 0, minimum = 100;
137.      int timer = 0;
138.      int i = 2000 / frequence;
139.      boolean measurenow;
140.      while (i) {
141.        noInterrupts();
142.        measurenow = shouldmeasure;
143.        shouldmeasure = 0;
144.        interrupts();
145.        if (measurenow == 1) {
146.          normAccel = mpu.readNormalizeAccel();
147.          a = sqrt( normAccel.XAxis * normAccel.XAxis + normAccel.YAxis *
```

```
   normAccel.YAxis + normAccel.ZAxis * normAccel.ZAxis );
148.           smootheda = ( alpha * smootheda ) + ( 1 - alpha ) * a;//这里只进行了一次滤波
149.           i--;
150.           measurenow = 0;
151.         } else {
152.           delay(1);
153.         }
154.       }
155.       i = inittime / frequence;
156.       while (i) {
157.         noInterrupts();
158.         measurenow = shouldmeasure;
159.         shouldmeasure = 0;
160.         interrupts();
161.         if (measurenow == 1) {
162.           normAccel = mpu.readNormalizeAccel();
163.           a = sqrt( normAccel.XAxis * normAccel.XAxis + normAccel.YAxis *
   normAccel.YAxis + normAccel.ZAxis * normAccel.ZAxis );
164.           smootheda = ( alpha * smootheda ) + ( 1 - alpha ) * a;
165.           maximum = ( maximum > smootheda ) ? maximum : smootheda; //记录一段时间内
   smootheda 最大值
166.           minimum = ( minimum < smootheda ) ? minimum : smootheda; //记录一段时间内
   smootheda 最小值
167.           i--;
168.           measurenow = 0;
169.         }
170.       }
171.       trana = smootheda;
```

```
172.        Serial.print(maximum);
173.        Serial.print(" ma|mi ");
174.        Serial.print(minimum);
175.        Serial.println(" ");
176.        average = 0.5 * (maximum + minimum); //重力加速度
177.        MsTimer2::stop();
178.        return average;
179.    }
```

# 五、导师评审意见

导师签字：　　日期：