МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Оценка определения местоположения в пространстве по показаниям инерциального датчика»

Обучающегося М.А. Матюнин
(И.О. Фамилия) (личная подпись)

Руководитель С.В. Митин
(ученая степень, звание, И.О. Фамилия)

Консультант И.Ю. Усатова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Аннотация

Название дипломной работы: «Оценка определения местоположения в пространстве по инерциального показаниям датчика».

Выпускная работа состоит из введения, трех глав, заключения, списка литературы из 25 источников, включая 20 зарубежных источников, и приложения.

Ключевым вопросом дипломной работы является проблема погрешностей акселерометра, которая не дает полноценно использовать его в навигационных системах, отдельно от других датчиков.

Целью работы является оценка погрешности акселерометра при двойном интегрировании и разработка мобильного приложение для проверки оценки и наглядного примера.

Дипломная работа может быть разделена на следующие логически взаимосвязанные части: построение теоретической модели и прогнозирование оценки отличий от реальных данных; применения акселерометра в системе определение местоположения как дополнительного датчика; возможные применения и тестирование.

В конце исследования представлена работа об успешной оценки погрешности акселерометра, в результате двойного интегрирования, и показали на сколько эта погрешность велика и как сильно она может изменять показания замеров.

Подводя итоги, можно подчеркнуть, что данная работа является хорошим помощником для будущих работ с акселерометром, что поможет минимизировать погрешности и более часто использовать этот инерциальный датчик в измерении дистанции.

Abstract

The title of the graduation work is "Evaluation of positioning in space based on the readings of the inertial sensor".

The graduation work consists of an introduction, three chapters, a conclusion, a list of 25 references, including 20 foreign sources, and an attachment.

The key issue of the thesis is the problem of accelerometer errors, which prevents it from being fully used in navigation systems, separately from other sensors.

The aim of the work is to estimate the error of the accelerometer with double integration and to develop a mobile application for checking the assessment and to give an illustrative example.

The graduation work may be divided into several logically connected parts which are: the construction of a theoretical model and a prediction of estimates of differences from real data; the use of the accelerometer in the location system as an additional sensor; possible applications and testing.

Finally, the report on the successful estimation of the accelerometer error, as a result of double integration, is presented and it shows how big this error is and how much it can change the readings of the measurements.

It can be concluded that this work is a good reference for the future work with the accelerometer, which will help to minimize errors and to use this inertial sensor more often in a distance measurement.

Содержание

Введение

В данной работе рассматривается вопрос возможности использования датчика акселерометра в качестве датчика линейного перемещения**,** и возникающие при этом проблемы при двойном интегрировании, для получения линейного перемещения. А также практическое использование данных датчика акселерометра в приложениях для мобильных устройств.

Для решения первого вопроса необходимо разработать математическую модель и показать возникающие погрешности при обработке данных с датчика акселерометра.

Так же необходимо показать, как везет себя погрешность на реальном устройстве акселерометра. Для этого нужно будет взять реальный акселерометр, и написать для него программу, для вывода нужных нам данных дистанции при измерение расстояния.

Определить основные проблемы, при использовании данных только датчика акселерометра для определения положения в пространстве.

В нынешнем мире уже нельзя представить жизнь без инерциальных датчиков. Они пропитали все слои отраслей, от телефонов, машин, до самолётов и ракет. Из всех инерциальных датчиков для определения местоположения подходит только акселерометр, о нём и будет идти речь.

Акселерометр - это датчик для измерения ускорения, который состоит в классе инерциальных датчиков. Если раньше этот датчик был достаточно большой, то сейчас их делают в размере от 1 микрометра до 100 микрометров.

Целью данной работы является поиск погрешностей акселерометра и возможными применениями датчика, примером датчика будет служить акселерометр ADXL345.

Предмет исследования: разработка модели получения данных акселерометра и исследование влияния частоты опроса акселерометра на погрешность измерения ускорения, возможные области применения акселерометра.

1 Построение теоретической модели и прогнозирование оценки отличий от реальных данных

1.1 Формирование задачи для дальнейшего исследование погрешностей.

Первым делом нужно разобраться, а какие вообще есть инерционные датчики. Первым является акселерометр. Акселерометр - это датчик для измерения ускорения, он был изобретен в конце 19 века и устанавливался в автомобили и паровозы для контроля за скоростью. Первые акселерометры были тяжелыми и громоздкими. Они основывались на использовании инерционной силы движущийся с ускорением массы и представляет собой маятник со спиральной пружиной, пример показан на рисунке 1. При ускорении или замедлении грузик стремился сохранить своё первоначальное положение, т.е. отстать или опередить корпус [2]. Одна из пружин при этом сжималась, а маятник совершал опережение. Величина этого перемещения и определялась акселерометром для вычисления ускорения.

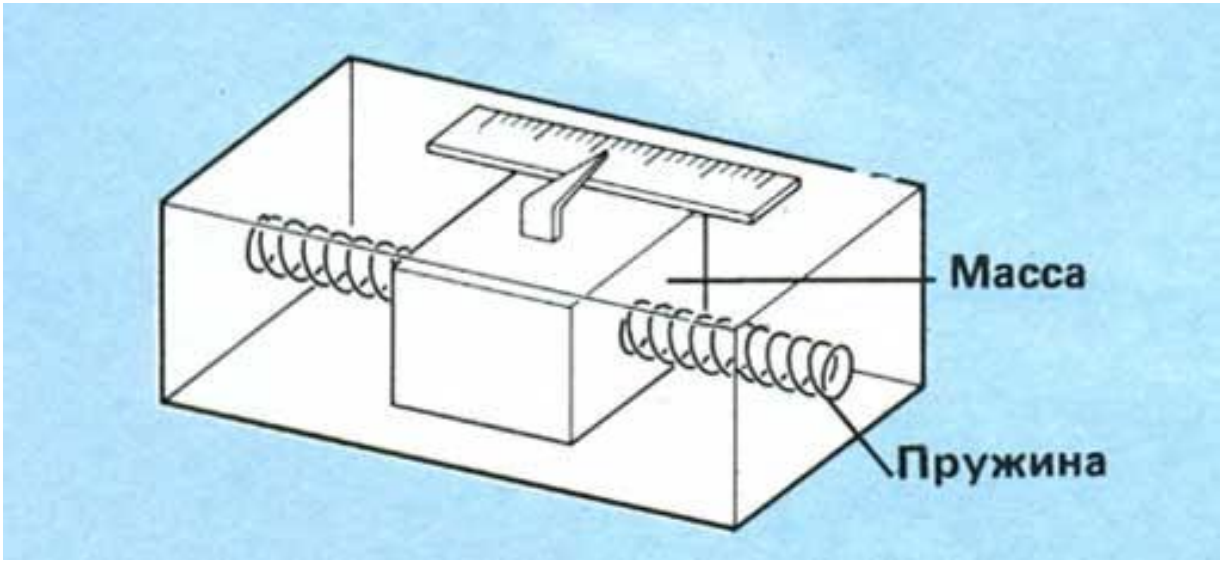


Рисунок 1 – Акселерометр

Сейчас же принцип действия акселерометра немного изменился, они стали меньше и легче. Сейчас он представляет из себя чип, в котором находится инертная масса, пример такого чипа показан на рисунке 2. Принцип такого чипа схож с работой классического акселерометра. Инертная масса меняет свою позицию во время ускорения [1], [6]. Что дает возможность получить данные о положении в пространстве. Сам чип представляет собой неподвижный корпус в котором на специальных упругих приставках крепится перегородка с отведенными в сторону проводниками. Эти отводы размещаются между контактами которые снимают показания. При перемещении отводов напряженность поля вокруг контактов меняет свои значения, это и служит показателями для измерения.

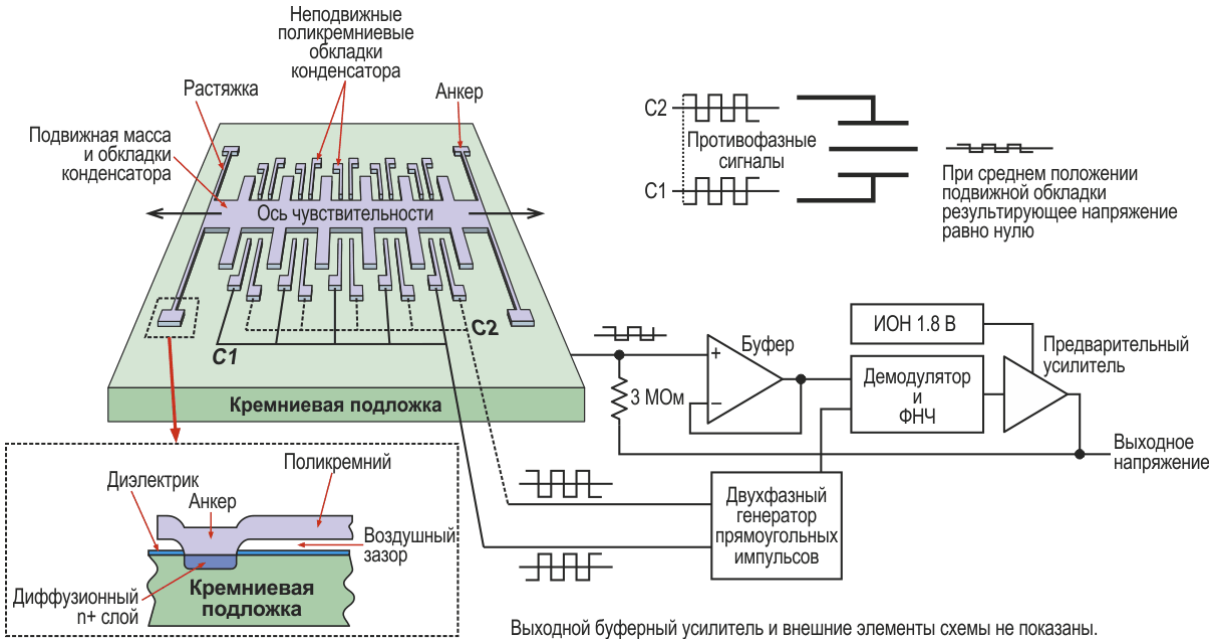
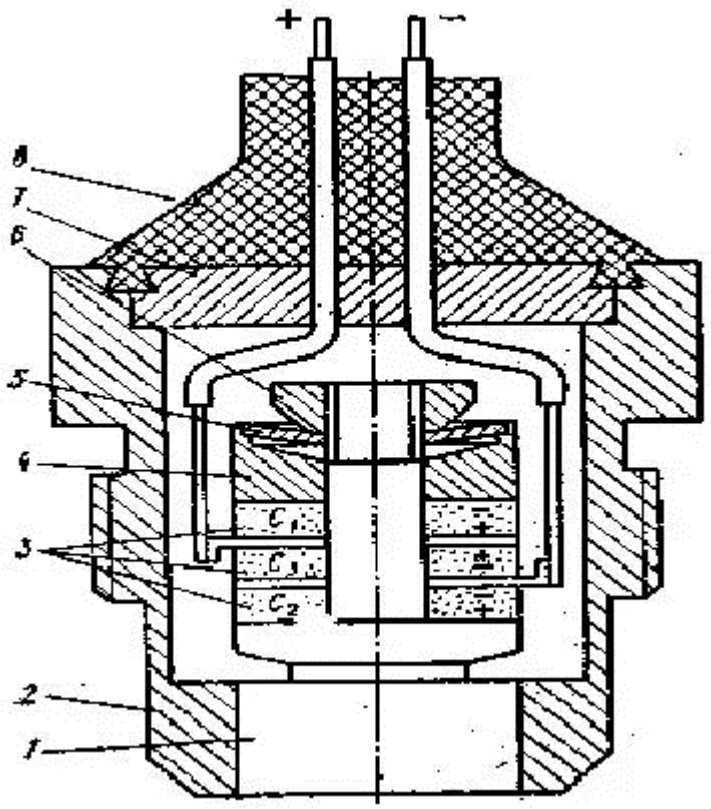


Рисунок 2 – Принцип действия современного акселерометра

В чипе все элементы крайне малы и производятся на автоматизированных конвейерных линиях без участия человека, а при их изготовлении применяется реакции кремния с другими веществами.

Так же есть еще вид полноценных приборов, а точнее пьезоэлектрических, пример на рисунке 3 [13].



Обозначения: 1 – основание, 2 – корпус, 3 – пьезоэлементы, 4 – инерционный элемент, 5 – пружина, 6 – гайка, 7 – крышка, 8 – герметизация.

Рисунок 3 – Компенсационный акселерометр

В основе таких датчиков лежит грузик, который давлением воздействует на пьезокристалл. Благодаря давлению он вырабатывает электрический ток, что позволяет рассчитать искомое ускорение. Так же существует термальный акселерометр, он представлен на рисунке 4, архитектура которого предусматривает использование маятника, при ускорении маятника отклоняется от своего начального положения, что и фиксируется датчиками, а затем используется для расчёта ускорения [10]. Также стоить учесть то, что акселерометр всегда измеряет силу тяжести, т.е. показания не равны нулю, когда датчик неподвижен.

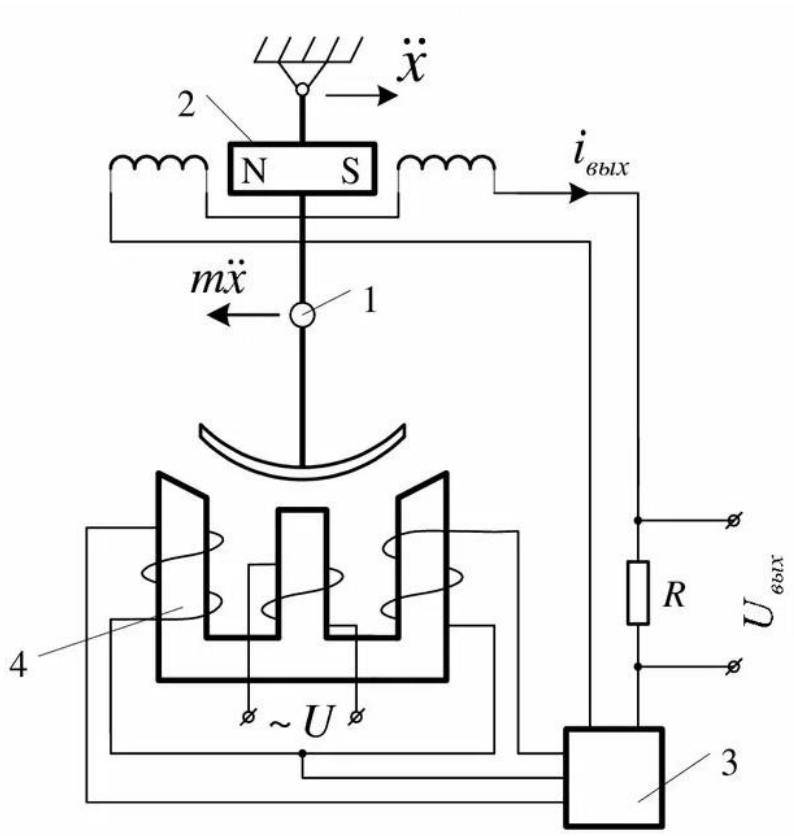


Схема: 1 – маятник, 2 – моментный датчик, 3 – усилитель, 4 – индуктивный датчик.

Рисунок 4 – Компенсационный акселерометр

Если акселерометр неподвижен, пружина растягивается только под действием силы тяжести [17]. Если акселерометр находится в динамическом состоянии, то будет сумма сил: вызванных инерцией веса ($F = m\vec{a}$) и гравитацией ($F_g = m\vec{g}$).

Использовать будем следующую модель для трех осевого акселерометра:

$$a_{XYZ} = m_a \times A_{XYZ} + b_a + n_a \quad (1)$$

Где a_{XYZ} - измененный вектор ускорения в координатах рамки тела XYZ;

m_a - матрица смещений осей и масштабных коэффициентов триады акселерометров;

A_{XYZ} - истинный вектор ускорения в координатах рамки тела XYZ;

b_a - вектор нулевого смещения акселерометров (смещения);

n_a - шум измерений.

Матрица смещения осей и масштабных коэффициентов имеет вид:

$$m_a = \begin{bmatrix} 1 + m_{a,1,1} & m_{a,1,2} & m_{a,1,3} \\ m_{a,2,1} & 1 + m_{a,2,2} & m_{a2.3} \\ m_{a,3,1} & m_{a,3,2} & 1 + m_{a,3,3} \end{bmatrix} \quad (2)$$

В данном уравнении элементы, расположенные на главной диагонали $(1 + m_{a,1,1}, 1 + m_{a,2,2}, 1 + m_{a,3,3})$, являются масштабными коэффициентами и их ошибками по трем осям акселерометров, а остальные элементы матрицы являются смещениями осей из триады акселерометров [23].

Объектом исследования в данной работе будет 3-х осевой акселерометр ADXL345, он показан на рисунке 5. Данный акселерометр способен измерять ускорение величиной до ±16 g, 13 бит максимального расширения, что дает максимальную частоту измерения в 3200 Гц. Диапазон питающего напряжения: от 2,0 В до 3,6 В. Широкий диапазон температур (от -40 °C до + 85 °C). Маленькие размеры чипа: 3 мм на 5 мм на 1 мм.

Рисунок 5 – 3-х осевой акселерометр ADXL345

Схема работы ADXL345 представлена на рисунке 6.

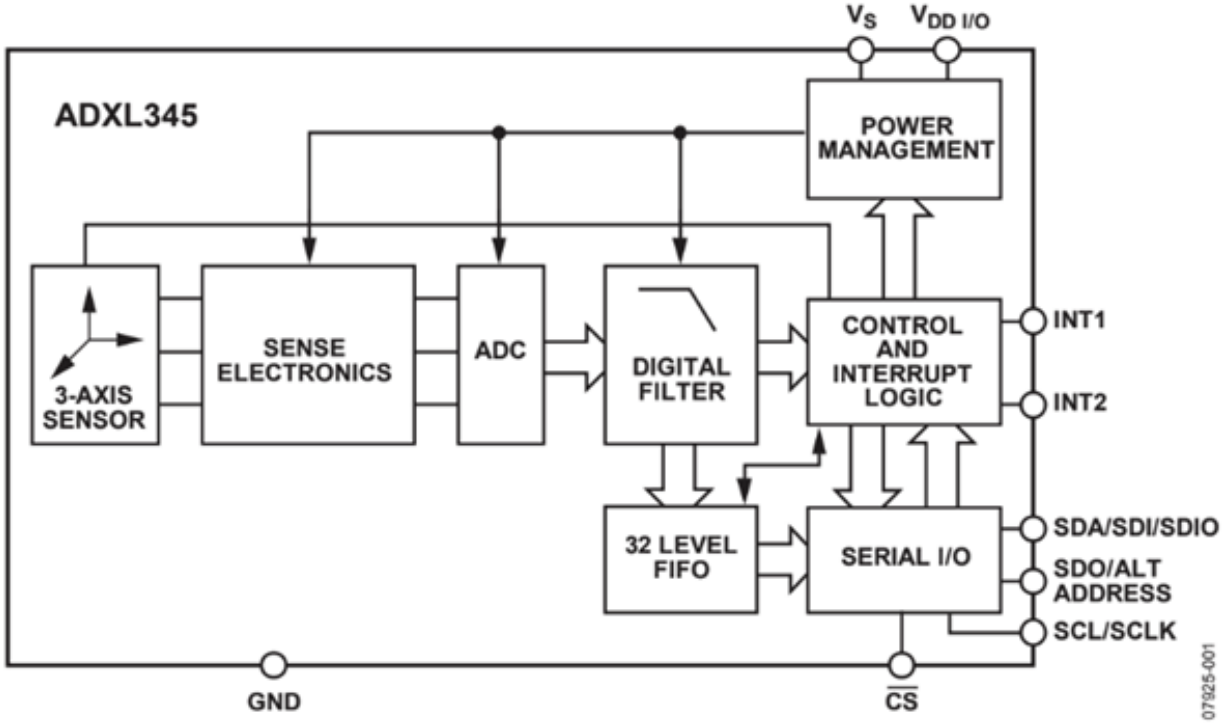


Рисунок 6 – Схема работы акселерометра ADXL345

1.2 Составление задачи для дальнейшей реализации

Нам необходимо измерить ошибку, и построить модель работы акселерометра и сравнить ее с идеальным вариантом.

Реализация программы:

– Строим любой непрерывный график функции,

– Делаем выборку из графика по максимальным значениям акселерометра,

– По этим данным проводим двойное интегрирование и сравниваем с идеальным графиком.

Схема ошибки оценки углов ориентации акселерометра представлена на рисунке 7.

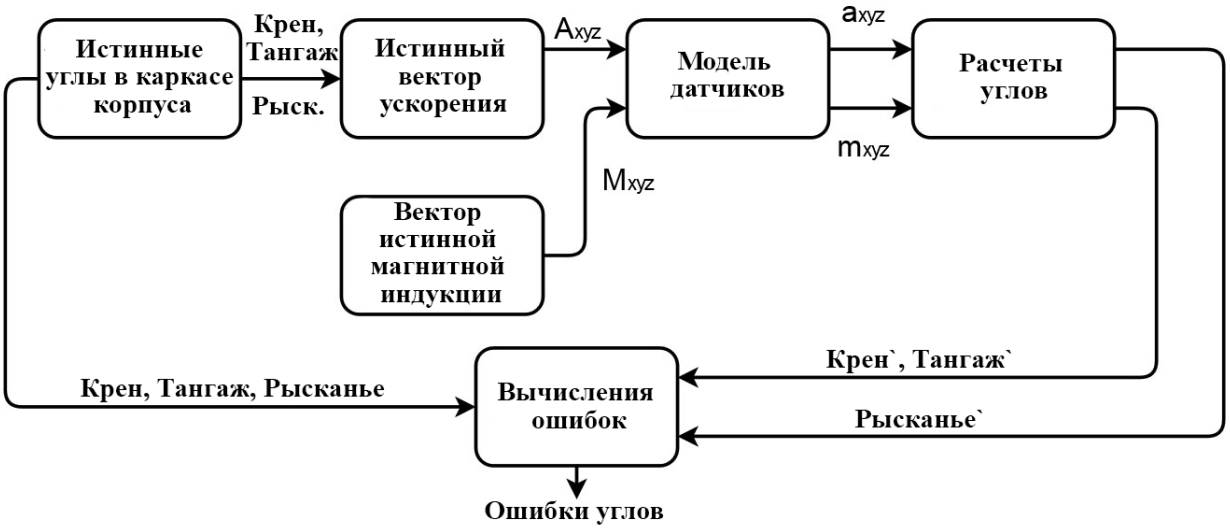


Рисунок 7 – Схема ошибок оценки углов ориентации.

Как видно, для конечных вычислений ошибок углов ориентации приходят только крен, тангаж и рысканье.

1.3 Проблема интегрирования данных акселерометра

МЭМС (англ. Микроэлектромеханическая система) датчики широко используются в последние годы. Низкая стоимость сенсорных чипов является основной причиной их использования во многих приложениях. Акселерометры MEMS часто используются в мобильные телефоны (смартфоны), жесткие диски, фотоаппараты, консольные игры и т.д.[15]. Однако в большинстве случаев акселерометры используются только для определения направления вектора силы тяжести.

В передовых приложениях, таких как определение ориентации воздушных судов и наблюдение за здоровьем людей, существует потребность в датчиках высокой точности и точности. Существуют также сложные задачи, такие как обнаружение автомобильных аварий, корректировка хирургической траектории или другие специализированные решения для микрорототехники [19].

Наиболее часто используемые системы, как правило, ориентированы на определение ориентации в статических условиях или на другие инерциальные измерения. Если вы используете видеокамеру с инерциальными измерениями, соответствующую обработку изображения можно улучшить многими способами [9]. Например, для улучшения качества результатов обработки, особенно при плохих условиях съемки, сдвиг изображения может быть уменьшен с помощью инерциальных измерений. Таким образом, разработка и использование надлежащего алгоритма объединения данных датчиков представляется важнейшей задачей.

Чтобы обеспечить точные результаты интеграции показаний акселерометра, мы должны проанализировать параметры и данные акселерометра для создания динамической модели применяемого акселерометра, пример вычислений показан на рисунке 8. Из-за хорошо известных проблем, связанных с интеграцией данных, существуют определенные ограничения для этого подхода. Поэтому мы предполагаем, что данные акселерометра должны быть интегрированы только на короткий промежуток времени. Чтобы обсудить наш подход к двойной интеграции, давайте сначала предположим, что все датчики установлены на устойчивом носителе или платформе [22]. Кроме того, датчики изолированы от внешнего вращательного движения, возникающего в результате перемещений применяемых карданов и применения подвижной робототехники или шагомеров [21].

Рисунок 8 – Замеры показаний акселерометра

Как результаты тестов с длительным временем интегрирования: Двойной интегрирование ошибки ускорения, равной 0.lg, будет означать ошибку определения местоположения более чем на 350 м в конце испытания [8]. Однако существуют решения (в стадии разработки), используемые в шагометрии и основанные на двойной интеграции данных акселерометра, которые не указывают на большие ошибки определения местоположения [18].

Существуют и другие проблемы с акселерометрами, которые следует принимать во внимание. Показания акселерометров страдают от потери точности данных, вызванной технологией MEMS. Однако на практике такие функции должны быть смоделированы, поскольку трудно не учитывать их влияние на ухудшение качества данных. Типичными проблемами, связанными с MEMS, являются их масштабный коэффициент и смещение [20]. Масштабный коэффициент, иногда называемый чувствительностью, представляет собой отношение изменения на выходе датчика к изменению на его входе, которое предназначено для измерения.

Смещение акселерометра можно определить, как ненулевое смещение выходного сигнала датчика от ожидаемого истинного значения в м/с2. Смещение не имеет никакой корреляции с входными данными. Он измеряется в течение определенного времени при заданных условиях эксплуатации. Обе характеристики на них влияют факторы, связанные с материалом и конструкцией, например нелинейность, гистерезис, эффекты поперечных осей и т.д. Существуют также факторы, связанные с условиями эксплуатации, такие как температура и давление. Смещение и чувствительность (масштабный коэффициент) могут быть улучшены с помощью надлежащей процедуры калибровки. Мерцающий шум возникает из-за определенного термомеханического шума, колебаний со скоростью, превышающей частоту дискретизации датчика [16]. Как следствие, смещение может изменяться с течением времени. Такие флуктуации можно смоделировать как случайное блуждание. Мерцающий шум создает случайное блуждание второго порядка по скорости (неопределенность растет пропорционально t3/2) и случайное

блуждание третьего порядка по положению (неопределенность растет пропорционально t5/2).

Таким образом, двойное интегрирование сигнала с флуктуирующим смещением вызывает ошибку положения, что видно на рисунке 9, которая квадратично растет со временем. Вышеупомянутые проблемы с шумом в MEMS, а также трудности двойного интегрирования учитываются при построении подходящей модели акселерометра MEMS и при синтезе соответствующих уравнений преобразования [24]. Для этих целей нам необходимо охарактеризовать силы, которые оказывают влияние на акселерометр.

t=t0 находится в положении *[x0 , y0 , z0]* и векторе скорости *[vx0 , vy0 , vz0]*.

У *t=t1* читается вектор ускорения *[ax1 , ay1 , az1]* (среднее ускорение от *t0* и *t1*).

Тогда вектор скорости *at t=t1* будет равен:

$$[vx1\,, vy1\,, vz1] = = [vx0 + ax1 \times (t1 - t0)\,, vy0 + ay1 \times (t1 - - t0)\,, vz0 + az1 \times (t1 - t0)] \quad (3)$$

Средняя скорость между *t0* и *t1* будет равна:

$$[vx01\,, vy01\,, vz01] = [(vx0 + vx1) / 2\,, (vy0 + vy1) / 2\,, (vz0 + vz1) / 2] \quad (4)$$

И позиция *at t=t1* будет равна:

$$[x1\,, y1\,, z1] = [x0 + vx01 \times (t1 - t0), y0 + vy01 \times (t1 - t0), z0 + vz01 \times (t1 - t0)] \quad (5)$$

Как видно, ошибка распространяется с t^2, поэтому инерциальные системы должны быть компенсированы внешней ссылкой, такой как GPS [14].

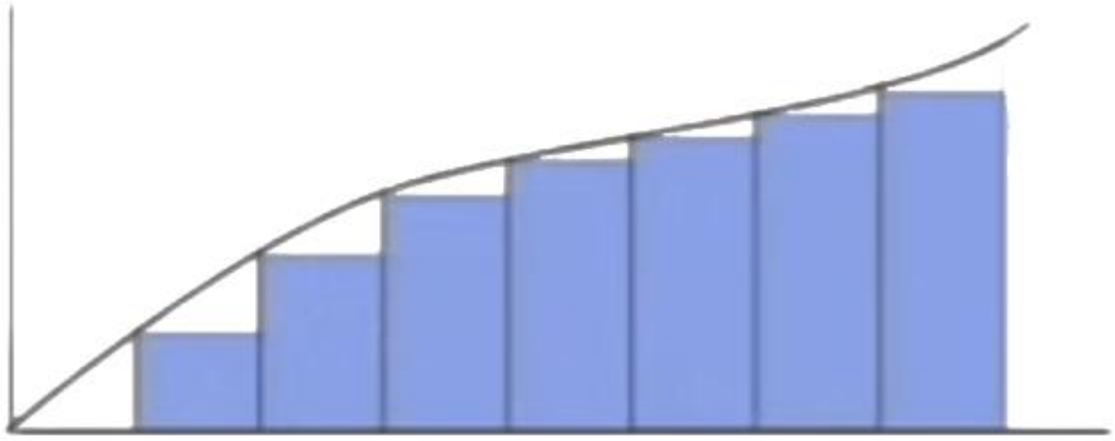


Рисунок 9 – Пример накопления погрешности

Данные по времени должны быть максимально точными. Если данные по времени отклонены например на 5%, это так же плохо, как если бы данные акселерометра были отклонены на 5% [7], [12].

Для измерений всех показаний, как правило, нам нужно само устройство, после идут процессоры движения и в конце процессоры приложения. Как правило, устройство и процессор движения расположены сразу на одной кремниевой плате.

Измерить положение, двигая акселерометром из стороны в сторону очень сложно. Сейчас посмотрим на то, как это будет работать.

Нам нужно линейное ускорение, мы можем двигать устройством вверх/вниз, вправо/влево и вперед/назад.

Нужно взять данные акселерометра и убрать гравитацию, это называется гравитационной компенсацией.

Линейное ускорение = данные с акселерометра - гравитационное ускорение

Как только мы убрали гравитацию, все что осталось, это линейное движение. Затем нужно интегрировать его 1 раз, чтобы получить скорость.

$$v = \int adt \quad (6)$$

После снова интегрировать, чтобы получить положение. Это была сделана двойная интеграция.

$$x = \int vdt \quad (7)$$

Двойные интегралы создают ужасный дрейф, что показано на рисунке 10.

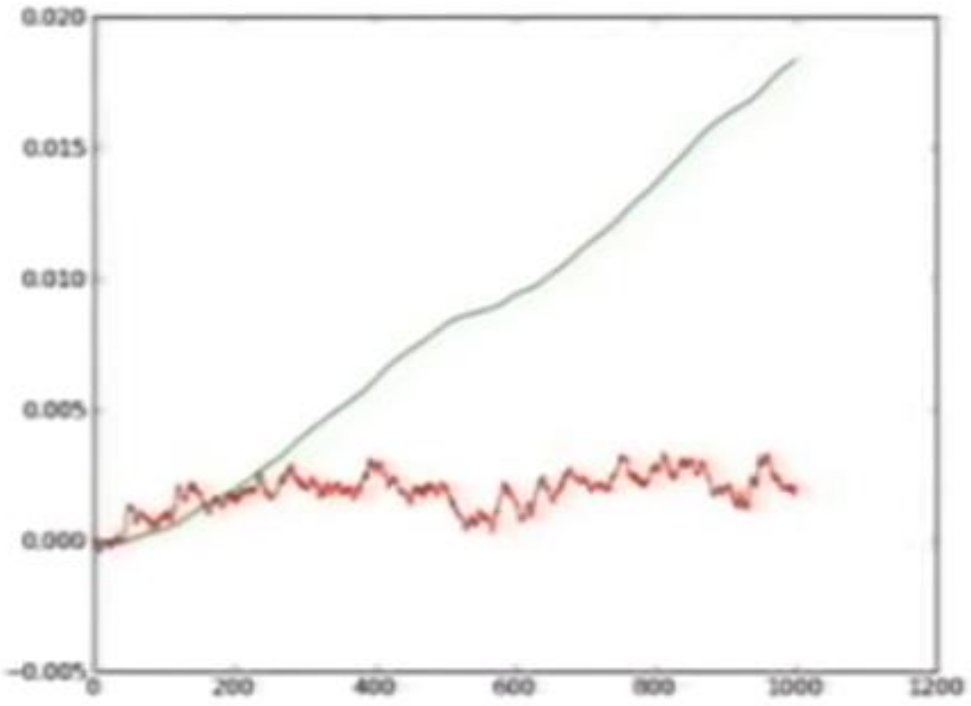


Рисунок 10 – одиночная и двойная интеграция

Красным показана одиночная интеграция, она как бы отклоняется из стороны в сторону, а зеленый сигнал, это двойная интеграция, он сильно взлетел.

Можно видеть, что это имитация акселерометра, ничего не делая, и он улетел на 20 сантиметров за 1 секунду из-за интегрирующего шума.

Проблема также может скрываться в том, что гравитацию убрать почти невозможно, можно только приближаться к этому, трудно сделать это идеально [11]. Предположим, что мы держали акселерометр при 37 градусах,

но на самом деле мы держали его при 38 градусах. Ошибка в оценки гравитации составила 1 градус, это видно на рисунке 11. Интегрируя это, мы получаем параболу. Мы дважды интегрируем константу вверх.

$$x = \frac{1}{2}at^2 \tag{8}$$

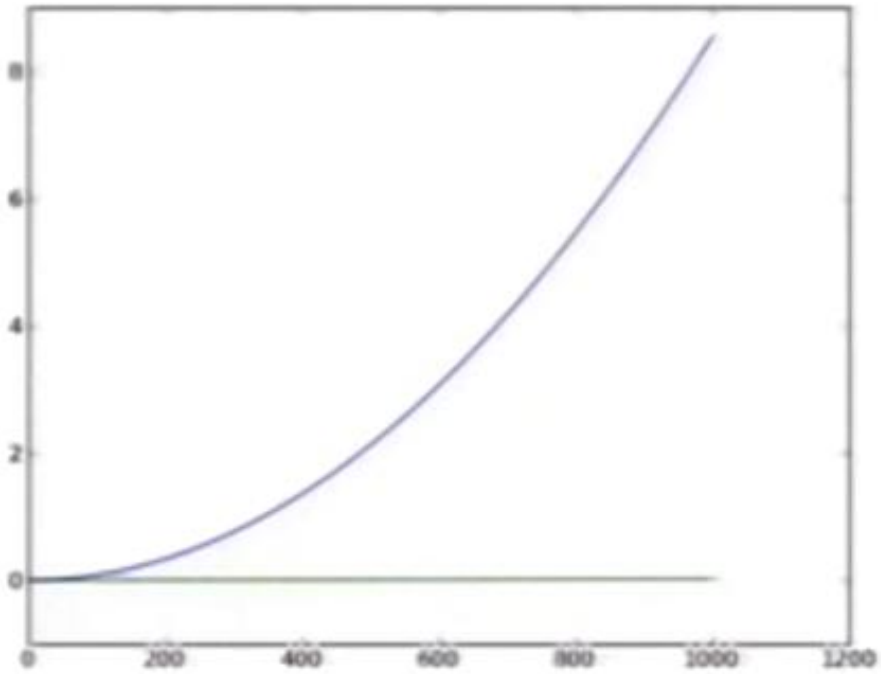


Рисунок 11 – Ошибка при смещение датчика на 1 градус

Зеленая линия, это линия с прошлого графика. Ошибаясь только на 1 градус, за 1 секунду, мы получаем ошибку в ориентации в 8.5 метров.

У нас есть оценка ориентации состоящая из ориентация, компас и сила тяжести, она представлена на рисунке 12.

Рисунок 12 – Модель оценки ориентации

Мы добавляем в нее линейное движение, схема на рисунке 13. Первое, что делаем, это компенсируем акселерометр с помощью гироскопа. Для того чтобы убедиться, что есть наилучшая оценка того, какой путь лежит вниз, мне нужны все эти датчики [25]. Тогда после использования всех этих датчиков у нас остается линейное ускорение, мы дважды интегрируем и результат должен быть лучше.

Рисунок 13 – Модель оценки ориентации с добавлением линейного ускорения

Использования фильтра высоких частот не работает в данном случае. Причина по которой это не работает, заключается в том, что мы предполагаем, что гравитация меняется медленно, но это не так. Гравитация меняется так быстро, как только быстро мы можем двигать устройством.

Таким образом была рассмотрена модель ошибки, выявлены все основные ее аспекты, и представлена модель возможного уменьшения этой погрешности, путем добавления новых датчиков.

2 Применения акселерометра в системе определение местоположения как дополнительного датчика

2.1 Определение положения с помощью датчика акселерометра

Чтобы подтвердить наши исследования из прошлой главы, было принято решение разработать приложение, которое на реальном устройстве покажет, на сколько большая может быть погрешность.

Был взят смартфон на операционной системе Android, и под него написано приложение в Android Studio, оно представлено на рисунке 14. Приложение взаимодействует только с акселерометром, не производит никаких уменьшений погрешностей и все данные являются реальными.

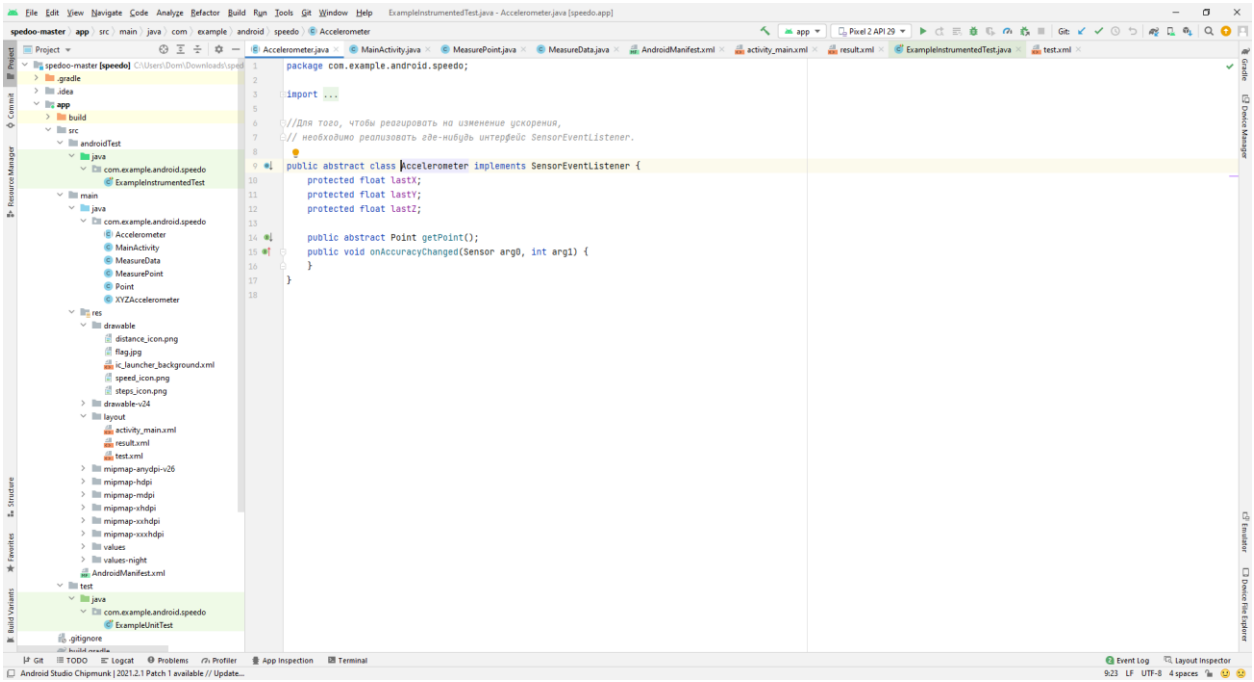


Рисунок 14 – Рабочая среда Android Studio

Почему не был взят смартфон на операционной системе iOS? Операционная система iOS является более закрытой, и получить доступ хотя бы к данным акселерометра является очень сложной задачей. А также сама

разработка под эту операционную систему очень сложна из-за факта, что на языке swift можно писать только в xcode, который так же доступен только на macOS и, следовательно, сама разработка является очень дорогой.

2.2 Реализация приложения для измерения дистанции при помощи акселерометра на Android

Сперва нужно разобраться с рядом ограничений, которые будут, если использовать в вычислениях только акселерометр. Первое, это то, что ориентация аппарата не должна изменяться в пространстве. Второе, что желательно откалибровать датчик перед тем, как будут проводиться измерения. И третье самое важное, это то, что движение должно быть обязательно прямолинейное.

Реализуем интерфейс SensorEventListener, код представлен на рисунке 15, чтобы была возможность реагировать на любые измерения ускорения.

```
public abstract class Accelerometer implements SensorEventListener {
    protected float lastX;
    protected float lastY;
    protected float lastZ;

    public abstract Point getPoint();
    public void onAccuracyChanged(Sensor arg0, int arg1) {
    }
}
```

Рисунок 15 – Интерфейс SensorEventListener

После необходимо реализовать класс, в котором будут находиться показания датчика, код представлен на рисунке 16.

```
public class Point {
    private float x = 0;
    private float y = 0;
    private float z = 0;
    private int cnt = 1;

    public float getX() { return x/(float)cnt; }

    public float getY() { return y/(float)cnt; }

    public float getZ() { return z/(float)cnt; }

    public Point(float x, float y, float z, int cnt) {
        this.x = x;
        this.y = y;
        this.z = z;
        this.cnt = cnt;
    }

}
```

Рисунок 16 – Показания датчика

В режиме SENSOR_DELAY_GAME, сенсор производит обновления примерно раз в 20 миллисекунд. Показания достаточно частые, что в принципе не требуется в нашей задаче, однако если их брать слишком редко, то могут попадаться “выбросы”, а они нам не нужны, так как пропадет точность, которая и так слабая.

Вот что получается дальше, создаем класс XYZAccelerometer и задаем размер буфера 500, это представлено на рисунке 17.

```
public class XYZAccelerometer extends Accelerometer{
    private static final int BUFFER_SIZE = 500;
```

Рисунок 17 – Класс XYZAccelerometer

Проводим калибровку, код представлен на рисунке 18.

```
private  float dX = 0;
private  float dY = 0;
private  float dZ = 0;
```

Рисунок 18 – Калибровка

Задаем буферные переменные, код представлен на рисунке 19.

```
private float X;
private float Y;
private float Z;
private int cnt = 0;
```

Рисунок 19 – Буферные переменные

Возврат последних значений SensorEvent представлен на рисунке 20.

```
public Point getLastPoint(){

    return new Point(lastX, lastY, lastZ,  cnt: 1);
}
```

Рисунок 20 – Последние значения SensorEvent

Берем параметры, которые возвращают среднее значение с использованием буфера в момент последнего вызова getPoint (), это представлено на рисунке 21.

```
public Point getPoint(){

    if (cnt == 0){
        return new Point(lastX, lastY, lastZ,  cnt: 1);
    }

    Point p =  new Point(X, Y, Z, cnt);

    reset();
    return p;
}
```

Рисунок 21 – Возврат средних значений

Сбрасываем все из буфера, код показан на рисунке 22.

```
public void reset(){
    cnt = 0;
    X = 0;
    Y = 0;
    Z = 0;
}
```

Рисунок 22 – Сброс буфера

И проводим все изменения с сенсором, калибруем датчики. Нужно снимать показания на протяжение времени и после установить у XYZAccelerometer нужные dX, dY, dZ [4], это показано на рисунке 23.

Сначала присваиваем x,y,z хранящиеся в массиве se.values с прибавлением к ним dX, dY, dZ. После присваиваем к каждой lant значения переменных x,y,z [5]. Прибавляем к уже имеющимся X,Y,Z, значения переменных x,y,z. Проверяем является ли значение cnt меньше чем BUFFER_SIZE. И в конце получаем значения dX, dY, dZ.

```
public void onSensorChanged(SensorEvent se) {
    float x = se.values[SensorManager.DATA_X] + dX;
    float y = se.values[SensorManager.DATA_Y] + dY;
    float z = se.values[SensorManager.DATA_Z] + dZ;

    lastX = x;
    lastY = y;
    lastZ = z;

    X+= x;
    Y+= y;
    Z+= z;

    if (cnt < BUFFER_SIZE-1) {
        cnt++;
    } else
    {
        reset();
    }
}

public  void setdX(float dX) { this.dX = dX; }

public  void setdY(float dY) { this.dY = dY; }

public  void setdZ(float dZ) { this.dZ = dZ; }
```

Рисунок 23 – Вычисление значений

Добавим класс, на котором можем хранить и вычислять параметры движения на интервале, код показан на рисунке 24.

В классе calc ускорение как проекция вектора тока в среднем, а так же используются формулы $\mathrm{v} = \mathrm{v0} + a_t$ и $\mathrm{s} = \mathrm{v0} \times \mathrm{t} + \mathrm{a} \times t^2/2$. В конце добавляем геттеры [3].

```java
public class MeasurePoint {
    private float x;
    private float y;
    private float z;
    private float speedBefore;
    private float speedAfter;
    private float distance;
    private float acceleration;
    private long interval;


    public MeasurePoint(float x, float y, float z, float speedBefore, long interval, float distance) {
        this.x = x;
        this.y = y;
        this.z = z;
        this.speedBefore = speedBefore;
        this.interval = interval;
        this.distance = distance;
        speedAfter = 0;
        calc();
    }

    private void calc(){
        System.out.println(this.x + "   "+ this.y+ "    "+this.z);
        acceleration = (float) Math.sqrt(this.x*this.x+this.y*this.y+this.z*this.z);
        float t = ((float)interval / 1000f);
        speedAfter = speedBefore + acceleration * t;
        System.out.println("Скорость до: "+ speedBefore + "   Ускорение: " + acceleration + "    Время: " + t + "   Скорость: " + speedAfter);
        distance += speedBefore*t + acceleration*t*t/2;
    }

    public float getSpeedAfter(){

        return speedAfter;
    }
    public float getDistance() { return  distance; }
    public float getSpeedBefore(){ return speedBefore;}
}
```

Рисунок 24 – Класс хранения и вычисления

А также для хранения данных обо всем эксперименте создадим класс MeasureData, он представлен на рисунке 25 и 26.

```
public class MeasureData {
    // точки от акселерометра
    private LinkedList accData;
    private LinkedList data;
    private TextView  checkAcc;

    // таймер интервала генерации точек
    private long interval;

    public MeasureData(long interval) {
        this.interval = interval;
        accData = new LinkedList ();
        data = new LinkedList ();
    }

    public void addPoint(Point p){
        accData.add(p);

    }

    public void process(){

        for(int i = 0; i < accData.size(); ++i){
            Point p = (Point)accData.get(i);
            float speed = 0;
            float distance = 0;

            if(i > 0){
                MeasurePoint mp = (MeasurePoint) data.get(i-1);
                speed = mp.getSpeedAfter();
                distance = mp.getDistance();
            }
            data.add(new MeasurePoint(p.getX(), p.getY(), p.getZ(), speed, interval,distance));
        }
    }
```

Рисунок 25 – Код, 1 часть

```
    String format(float k1, float k2, float k3) {
        return String.format("%1$.3f\t\t%2$.3f\t\t%3$.3f", k1, k2,
                k3);
    }
    public float getLastSpeed(){
        MeasurePoint mp = (MeasurePoint) data.getLast();
        return mp.getSpeedAfter();
    }
    public String getAcc(){
        Point k = (Point)accData.get(accData.size()-1);
        //System.out.println(k.getX()+" "+k.getY() + " " + k.getZ());
        return (""+ format(k.getX(), k.getY(), k.getZ()));
    }

    public float getDistanceM(){
        MeasurePoint mp = (MeasurePoint) data.getLast();
        return mp.getDistance();
    }
    public float getSpeedBeforeKm(){
        MeasurePoint mp = (MeasurePoint) data.getLast();
        return mp.getSpeedBefore();
    }
    public float getLastSpeedKm(){
        float ms = getLastSpeed();
        return ms*3.6f;
    }
}
```

Рисунок 26 – Код, 2 часть.

Далее нужно приступить к Activity, добавляем отображение скорости в км/ч, расстояние в метрах, параметры калибровки для возможности задать dx, dy, dz. Кнопку старт, чтобы начать замеры, обнуление результатов, чтобы начать повторный эксперимент и результаты измерения акселерометра, которые выводим отдельным loyout ниже основного поля, графическая часть activity показана на рисунке 27 и 28.

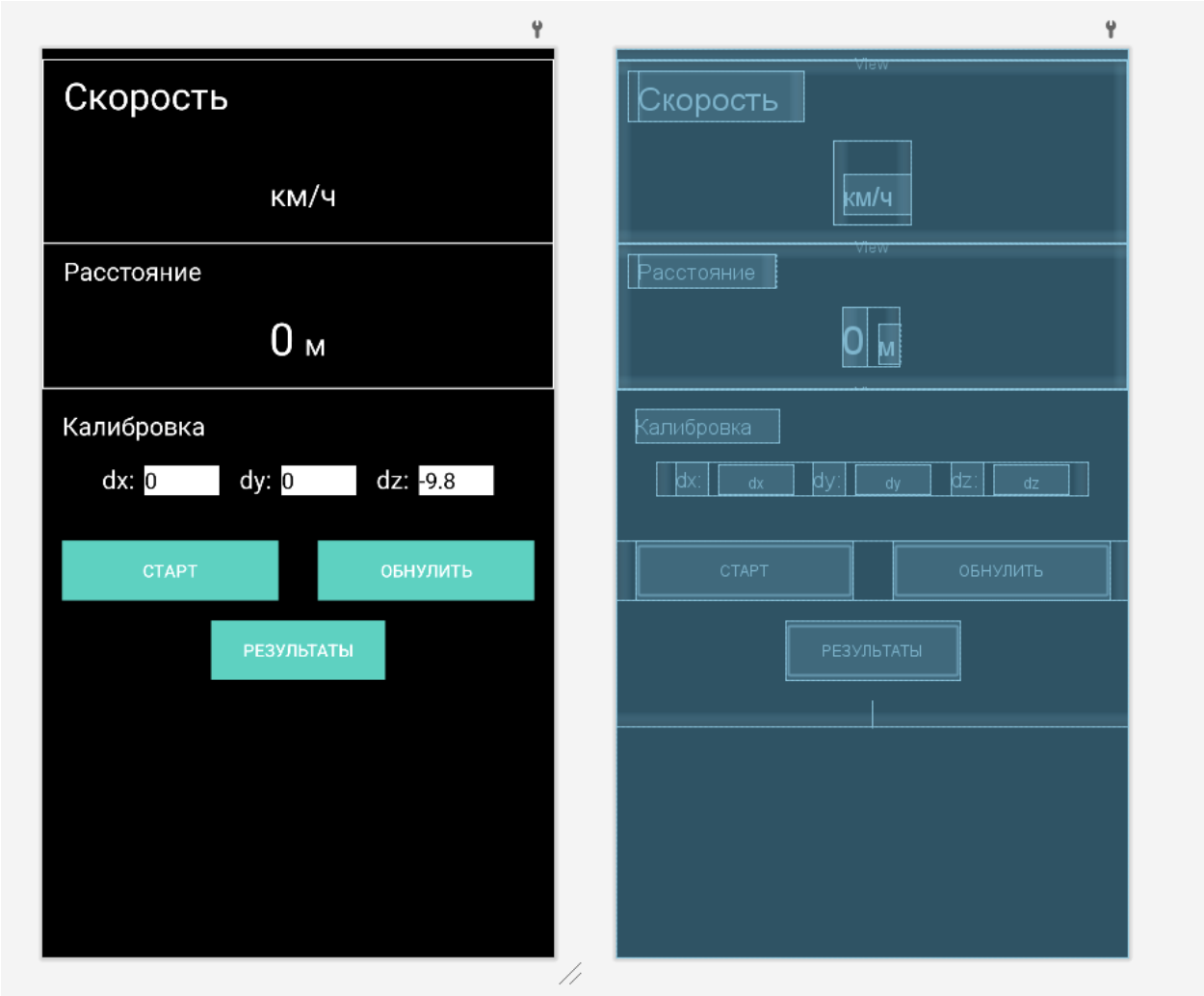


Рисунок 27 – Первая часть Activity приложения

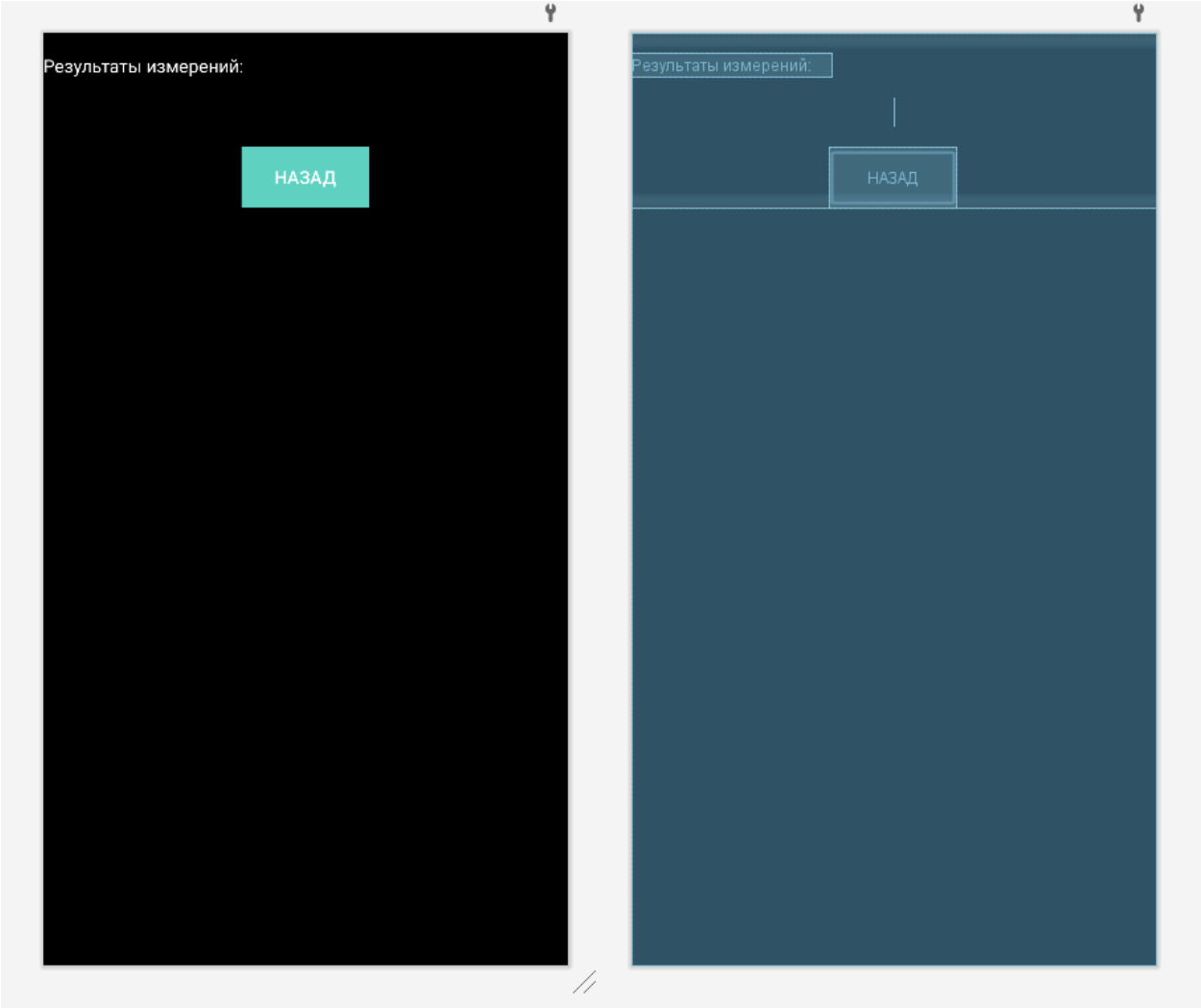


Рисунок 28 – Вторая часть Activity приложения

Приложение написано, теперь можно его тестировать и проводить замеры данных с акселерометра, сборка представлена на рисунке 29.

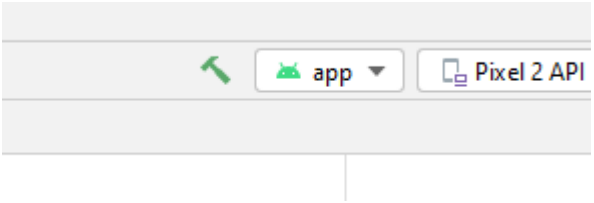


Рисунок 29 – Сборка приложения

Так как тестирование внутри приложения Android Studio может проводиться без финальной сборки приложения, это допускает проблему того, что с тем же apk файлом на устройстве приложение не допустит установку, или будет работать не корректно, поэтому после всех тестирований в Android Studio нам необходимо провести финальную сборку приложения, и только тогда устройство на android разрешит установку и будет работать нормально.

В данном разделе было разработано приложение для телефона на операционной системе Android, на котором присутствует датчик акселерометра, который и использовался, для сбора данных для работы приложения. Приложение должно собирать данные с акселерометра, обрабатывать из, после нажатия кнопки старт, проводить вычисления и показывать какая была скорость и дистанция при начале эксперимента. Приложение запустилось на виртуальном устройстве, далее остается только протестировать его на реальном.

3 Возможные применения, тестирование

Приложение готово, можно запускать. Сначала запустим в эмуляторе, что видно на рисунке 30, чтобы выявить возможные проблемы и неполадки, которые недопустимы на реальном устройстве.

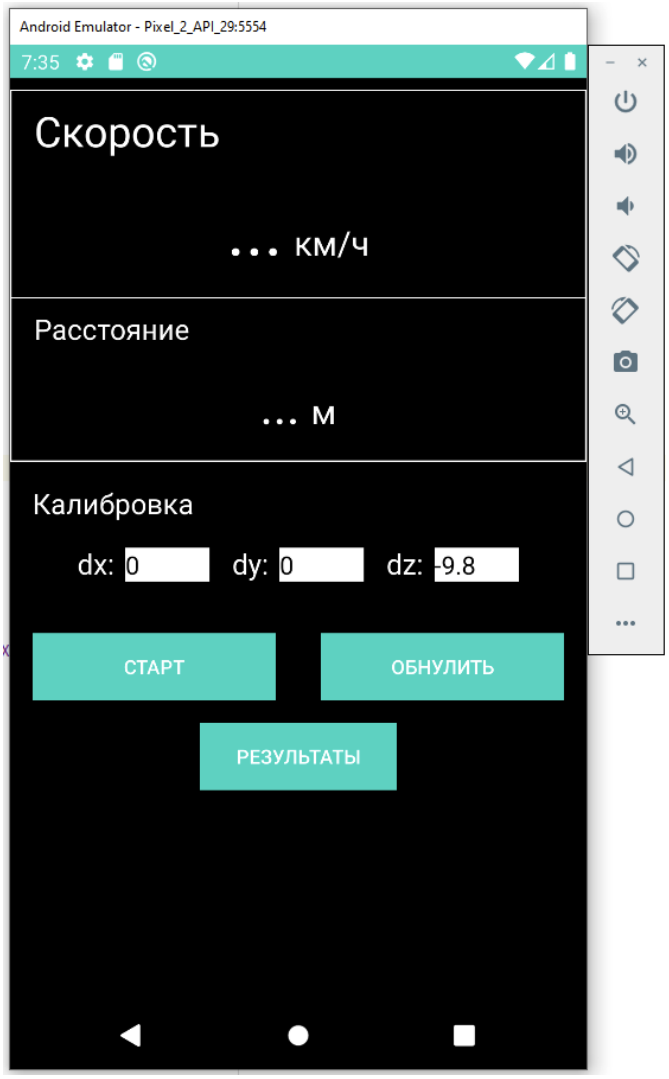


Рисунок 30 – Android приложение в эмуляторе

Приложение нормально запустилось, все кнопки и надписи ровно на тех местах, на которых и должны. Попробуем запустить. Результат запуска показан на рисунке 31.

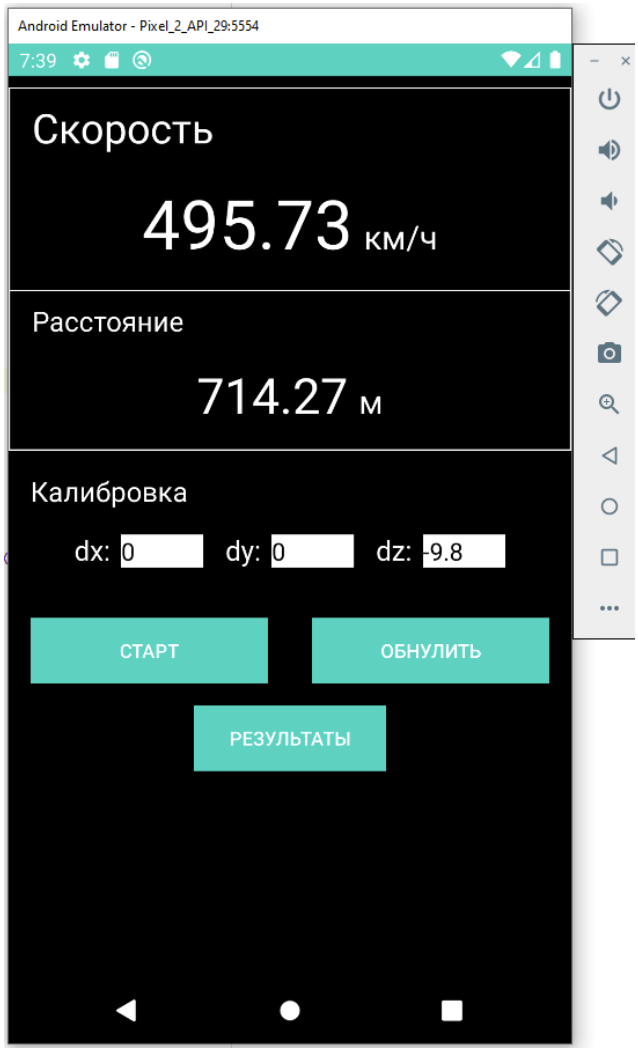


Рисунок 31 – Запуск программы

Данные не совсем корректны, так как запускается на эмуляторе, где нет акселерометра. Теперь можно приступать к тестам на реальном устройстве.

В эксперименте было использован смартфон Samsung Galaxy s9, с операционной системой Android 10. Подробные сведения об устройстве можно увидеть на рисунке 32.

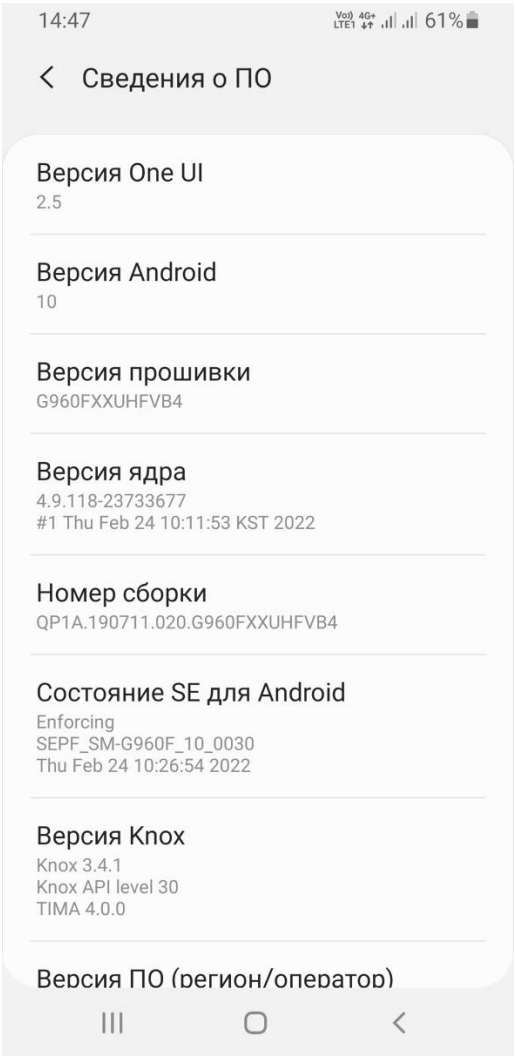


Рисунок 32 – Сведения о устройстве

Теперь приступаем непосредственно к тестированию на реальном устройстве. Скачиваем apk файл на устройство, подтверждаем, что apk файл нормальный, без вирусов, ведь на современных устройствах множество проверок, чтобы не допустить проблем с самим устройством. Запускаем приложение, запуск приложения показан на рисунке 33.

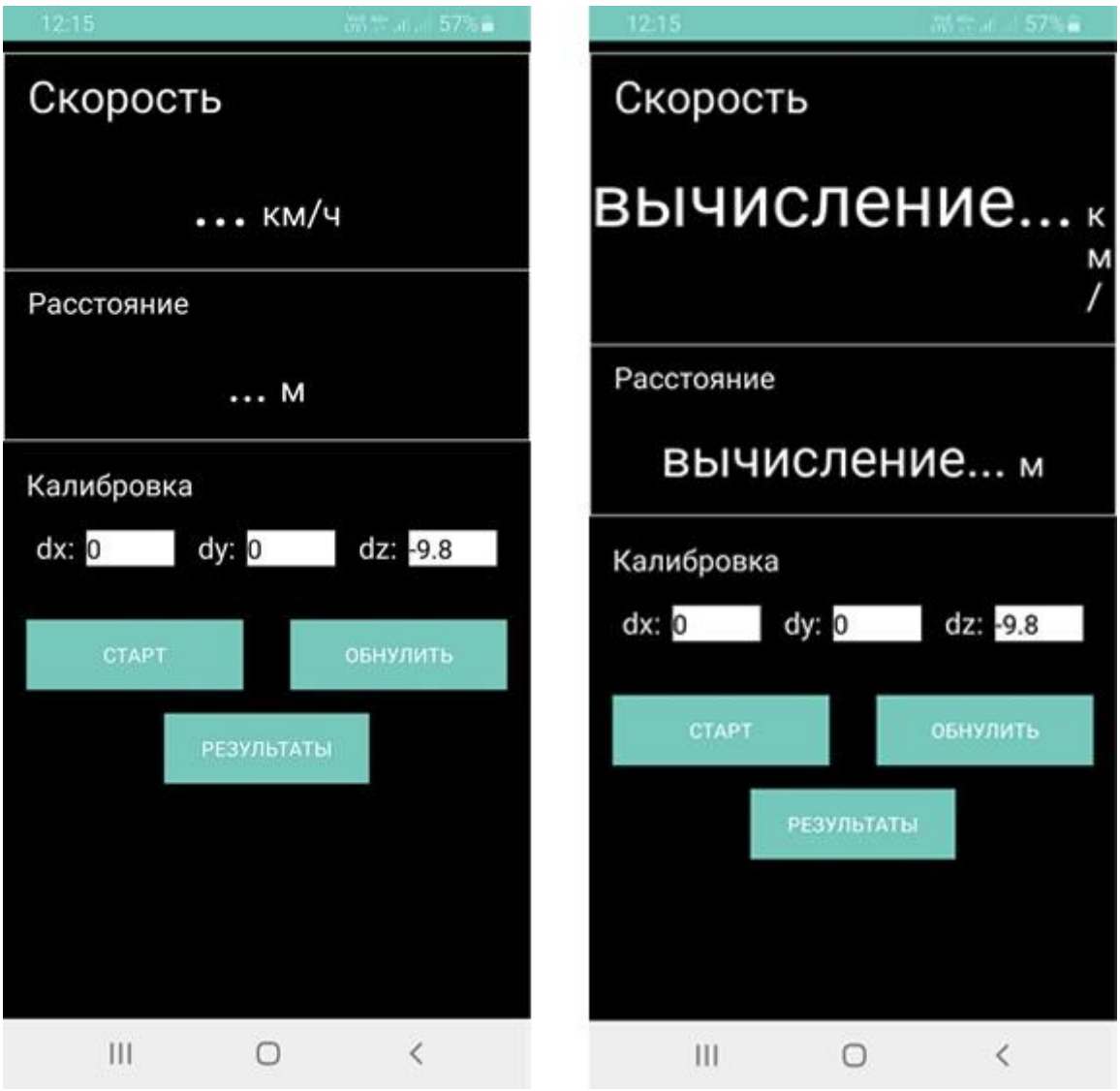


Рисунок 33 – Запуск приложения на устройстве Android

Было собрано финальное приложение, чтобы оно могло установиться с помощью apk файла на смартфон. Приложение запустилось нормально, работает стабильно, с установкой не было проблем, теперь можно приступать к запуску измерения скорости и дистанции.

Чтобы придать измерениям максимально чистые измерения, был выбран ровный участок, длиной 3 метра, само устройство под ровными углами было закреплено на плоскую поверхность.

Во время начала измерений, устройство постепенно передвигалось, без изменения углов и средней скорости. Испытаний проводилось 10 попыток, на рисунке 34 показан средний результат всех измерений.

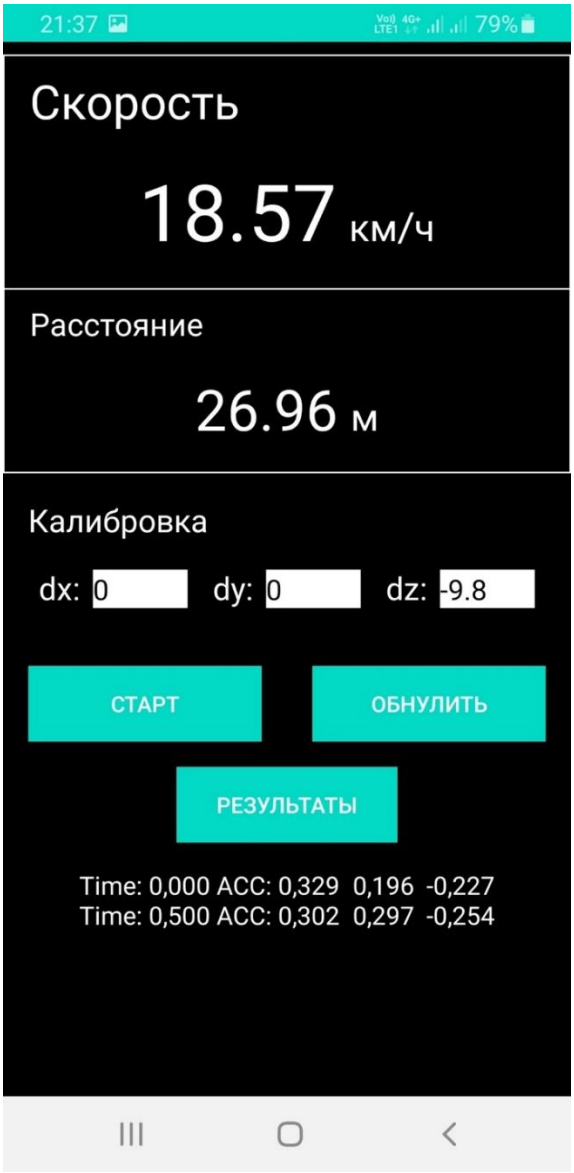


Рисунок 34 – Работа приложения на устройстве Android

Как видно по результатам измерения, погрешность является слишком большой и подтверждает, что она есть и на реальных устройствах, что доказывает то, что использование только акселерометра в измерении линейного движения является невозможным. Тем самым было показано на реальном устройстве акселерометра, как себя показывает эта погрешность.

Заключение

В данной работе мы рассмотрели вопрос об использовании датчика акселерометра в качестве датчика линейного перемещения, и возникающие при этом проблемы при двойном интегрировании, для получения линейного перемещения. Была представлена математическая модель, данные для которой были искусственно созданы, вывели графики, на которых видно, что погрешность накапливается очень быстро, и что по факту, при реальных замерах, погрешность может быть еще больше, так как не учтены многие внешние факторы.

Далее было принято решение в разработке мобильного приложения, чтобы наглядно показать на устройстве, где есть акселерометр, что при реальных измерениях есть эта погрешность и на сколько она большая.

При разработке мобильного приложения использовалась среда разработки Android Studio и язык программирования Java. В ходе разработки был написан код, который взаимодействовал с акселерометром внутри телефона на операционной системе Android. Важно было добавить в него только замеры с самого акселерометра и вычисление скорости и расстояния без каких-либо уменьшений погрешности, чтобы показать, как на самом деле обстоят дела с этим датчиком. Мобильное приложение было разработано, в его интерфейсе было выведено отображение скорости перемещения, дистанции пройденной с прибором и сами показания с акселерометра. В ходе проведения эксперимента, было сделано несколько замеров, чтобы усреднить данные и показать максимально точный результат.

Как и ожидалось после построения математической модели и оценки всех условий погрешности, реальные данные эксперимента, очень разнились с теми, что были показаны на устройстве. Таким образом можно сделать вывод, что использование только датчика акселерометра в определении дистанции является невозможным, без использования дополнительных датчиков, либо фильтров данных.

Список используемой литературы

1. Дао Ван Ба Динамический метод исследования погрешностей триады акселерометров [Текст]: дис . канд. техн. наук: 05.11.03: защищена 22.01.15: утв. 15.07.14 / Дао Ван Ба. – Санкт-Петербург – 2015, – 113 с.
2. Денисенко В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. : Горячая Линия - Телеком, 2009. 606 с.
3. Медникс З., Дорнин Л., Мик Б., Накамура М. П78 Программирование под Android. 2-е изд. — СПб.: Питер, 2013. — 560 с.
4. Сатия Коматинени, Дейв Маклин. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов. ; под ред. Ю.Н. Артеменко. : Вильямс 2012. 880 с.
5. Фелкер, Донн. Ф38 Android: разработка приложений для чайников. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2012. — 336 с.
6. A. Albarbar, A. Badri, J. K. Sinha, and A. Starr, “Performance evaluation of MEMS accelerometers,” Measurement: Journal of the International Measurement Confederation, vol. 42, no. 5, pp. 790–795, 2009
7. B. Barshan and H. F. Durrant-Whyte, “Inertial navigation systems for mobile robots,” vol. 11, pp. 328–342, June 1995
8. D. H. Titterton and J. L. Weston, Strapdown inertial navigation technology. The Institution of Electrical Engineers, 2004.
9. D. R. P. R. B. M. Joseph M. Cooke, Michael J. Zyda, “Npsnet: Flight simulation dynamic modelling using quaternions,” Presence, vol. 1, pp. 404–420, 1994
10. E. Foxlin, “Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter,” in Proc. Virtual Reality Annual International Symposium the IEEE 1996, pp. 185–194,267, Mar. 30–Apr. 3, 1996.
11. E. Foxlin, “Pedestrian tracking with shoe-mounted inertial sensors,” IEEE Comput. Graph. Appl., vol. 25, no. 6, pp. 38–46, 2005

12. E. R. Bachmann, X. Yun, and C. W. Peterson, “An investigation of the effects of magnetic variations on inertial/magnetic orientation sensors,” in Proc. IEEE International Conference on Robotics and Automation ICRA ’04, vol. 2, pp. 1115–1122, Apr. 2004.

13. F. Mohd-Yasin, D. J. Nagel, and C. E. Korman, “Noise in MEMS,” Measurement Science and Technology, vol. 21, no. 1, pp. 12–21, 2009.

14. G. A. Aydemir and A. Saranl, “Characterization and calibration of MEMS inertial sensors for state and parameter estimation applications,” Measurement, vol. 45, no. 5, pp. 1210–1225, Jun. 2012.

15. G. Pang and H. Liu, “Evaluation of a Low-cost MEMS Accelerometer for Distance Measurement,” Journal of Intelligent and Robotic Systems, vol. 30, no. 3, pp. 249 – 265, 2001.

16. H. J. Luinge and P. H. Veltink, “Inclination measurement of human movement using a 3-d accelerometer with autocalibration,” vol. 12, pp. 112–121, Mar. 2004.

17. H. J. Luinge, P. H. Veltink, and C. T. M. Baten, “Estimation of orientation with gyroscopes and accelerometers,” in Proc. First Joint [Engineering in Medicine and Biology 21st Annual Conf. and the 1999 Annual Fall Meeting of the Biomedical Engineering Soc.] BMES/EMBS Conference, vol. 2, p. 844, Oct. 13–16, 1999.

18. M. Haid and J. Breitenbach, “Low cost inertial orientation tracking with kalman filter,” Applied Mathematics and Computation, vol. 153, pp. 567–575, June 2004.

19. O. J. Woodman, “An introduction to inertial navigation,” University of Cambridge, Computer Laboratory, Cambridge, UK, Tech. Rep. 696, 2007.

20. R. A. Hyde, L. P. Ketteringham, S. A. Neild, and R. J. S. Jones, “Estimation of upper-limb orientation based on accelerometer and gyroscope measurements,” vol. 55, pp. 746–754, Feb. 2008.

21. R. E. Kalman, “A new approach to linear filtering and prediction problems,” Journal of Basic Engineering, vol. 82, pp. 35–45, 1960.

22. S. K. Hong, “Fuzzy logic based closed-loop strapdown attitude system for unmanned aerial vehicle (uav),” Sensors and Actuators A: Physical, vol. 107, no. 2, pp. 109 – 118, 2003

23. W. T. Ang P. K. Khosla, and C. N. Riviere, “Design of All-Accelerometer inertial Measurement Unit for Tremor Sensing in Handheld Microsurgical Instrument,” Proc. IEEE Intl. Conf Rob. and Auto., pp.1781-1786, Sep 2003.

24. Y. Maki, S. Kagami, and K. Hashimoto, “Localization and tracking of an accelerometer in a camera view based on feature point motion analysis,” in Proceedings of SICE Annual Conference, 2012, pp. 293 – 294

25. Y. Wang, S. Adams, J. Thorp, N. MacDonald, P. Hartwell, and F. Bertsch, “Chaos in MEMS, parameter estimation and its potential application,” in IEEE Transactions on Circuits and Systems I: Funda- mental Theory and Applications, vol. 45, 1998, pp. 1013 – 1020