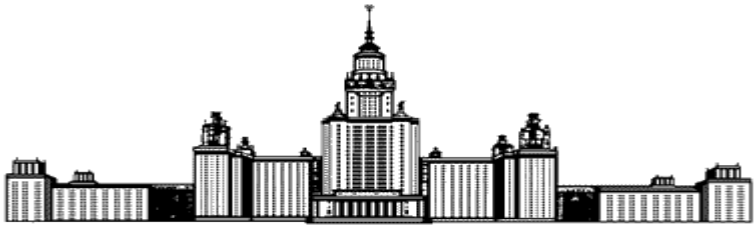

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М.В. ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

Кафедра автоматизации систем вычислительных комплексов Лаборатория открытых информационных технологий

Проект

Логгирование данных акселерометра в операционной системе Tizen

Выполнили:

Лебединская Дарья Игоревна студентка учебной группы №322,
Некраплённая Мария Николаевна студентка учебной группы №322

Научные руководители:

Ильюшин Евгений Альбинович,

Намиот Дмитрий Евгеньевич

Москва

2017

Оглавление

1. Введение

Акселерометр — прибор, измеряющий проекцию кажущегося ускорения (разности между истинным ускорением объекта и гравитационным ускорением). Как правило, акселерометр представляет собой чувствительную массу, закреплённую в упругом подвесе. Отклонение массы от её первоначального положения при наличии кажущегося ускорения несёт информацию о величине этого ускорения (рис 1).

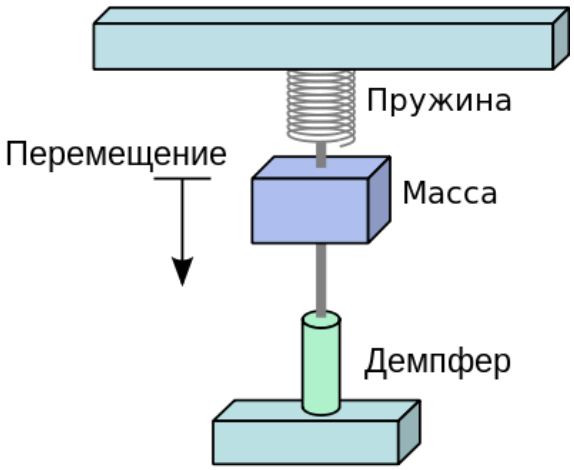

Рис 1. Устройство простейшего акселерометра.

При помощи акселерометр можно определить такие характеристики, как ориентация и наклон в пространстве, а также, что самое важное – фиксировать характер перемещений.

С развитием технологий акселерометрами стали оснащаться все приборы. Мобильное устройство, оснащенное таким датчиком, впервые было выпущено в 2005г году компанией Nokia. С тех пор акселерометры есть в устройствах всех производителей, в том числе и Samsung.

Ускорения считываются акселерометром по направлениям трех осей (рис 2).

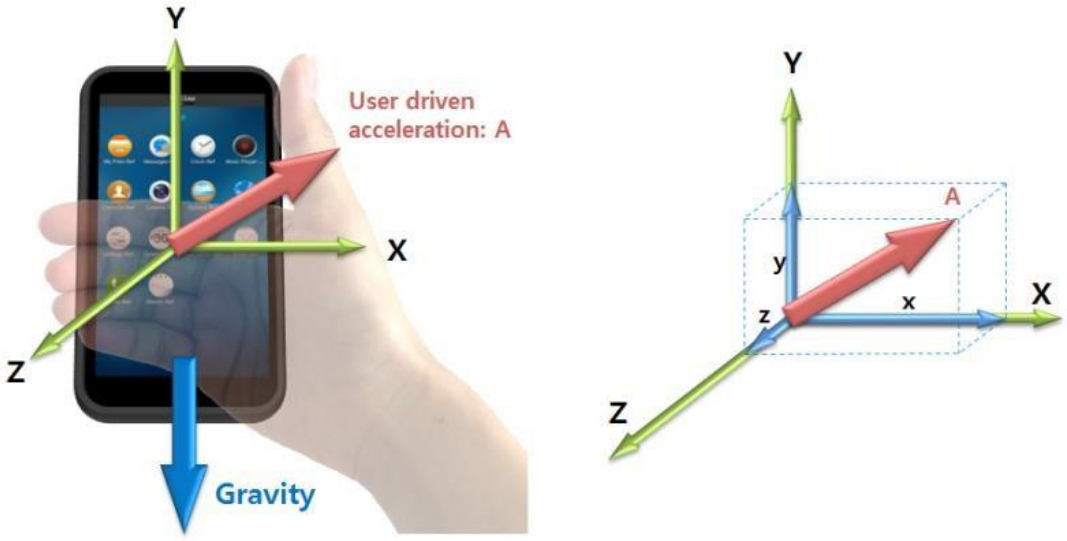

Рис 2. Направление осей координат акселерометра в телефоне [1].

Цель данной работы состоит в разработке приложения для считывания данных акселерометра в среде операционной системе Tizen.

Tizen — открытая операционная система на базе ядра Linux, предназначенная для широкого круга устройств (смартфонов, планшетов, wearable-устройств и т.д.). Её преимущество перед другими операционными системами состоит в том, что разработка может быть целиком произведена с использованием только связки HTML/JavaScript/CSS. С другой стороны, начиная с версии 2.0 в Tizen добавлены средства нативной разработки с использованием C++. Это делает возможным нативную разработку производительных приложений с использованием технологий OpenGL ES, OpenAL и OpenMP и библиотек Glibc, libstdc++ и libxml2 [5].

API Tizen предоставляет доступ к различным сенсорам, в том числе и к акселерометру.

2. Постановка задачи

Задачи данной исследовательской работы:

- выяснить принцип работы датчика акселерометра.

- научиться обращаться к данному датчику в ОС Tizen.
- изучить API социальной сети Twitter.
- изучения языка JavaScript и связки HTML/JavaScript/CSS.
- изучение процесса разработки клиент-серверных приложений.
- написать клиентскую часть приложения, которая будет извлекать показания датчика и отправлять серверу, либо сохранять их в собственной базе данных при отсутствии подключения к сети Интернет.
- написать серверную часть приложения, собственно логгер, которая будет отправлять полученные данные в Twitter.
- проанализировать ход работы и возникшие в процессе трудности.

3. Основная часть

3.1. Исследование и выбор средств разработки

Основное средство разработки для ОС Tizen — это Tizen SDK, который включает в себя IDE на основе Eclipse, набор инструментов (web-симулятор, эмулятор, дизайнер интерфейсов и другие), компилятор и документацию. Доступны сборки Tizen SDK для Ubuntu (x32/x64), Windows XP и Windows 7 (x32/x64), Apple Mac OS X 10.7 Lion и 10.8 Mountain Lion (x64).

Поскольку архитектура данного приложения подразумевает разделение клиентской и серверной части, были рассмотрены различные инструменты разработки.

Так, для организации хранилища данных на стороне клиента была выбрана IndexedDB [6]. Это объектная база данных, позволяющая хранить большие структурированные данные на клиенте. Она достаточна проста в использовании и позволяет быстро обращаться к локальному хранилищу данных в небольших приложениях.

Для разработки сервера была использована платформа Node.js [8].

Node или Node.js — это программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Для отправки данных от клиента к серверу используется API XMLHttpRequest[2].

Для отправки данных акселерометра в социальную сеть Twitter в рамках архитектурного стиля REST используется открытая асинхронная клиентская библиотека 'twitter' [4].

3.2. Структура приложения

Приложение включает в себя клиентскую и серверную части. Визуальная часть написана на `html` и `css`. При запуске приложения пользователю предлагается начать считывание данных акселерометра при нажатии на кнопку (рис 3).

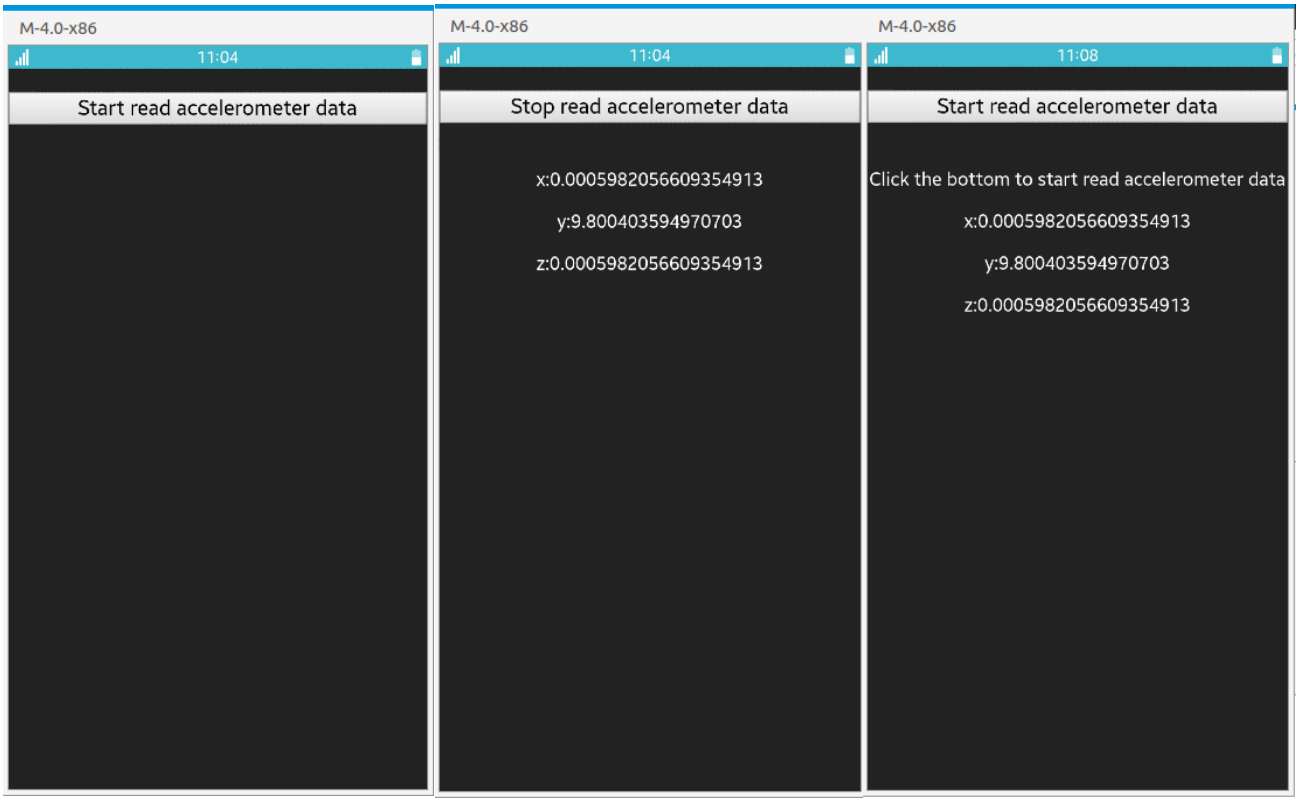

Рис 3. (a) начало работы приложения; (b) приложение в процессе считывания данных; (c) прекращение считывания данных.

После нажатия генерируется событие `onclick` на которое установлен обработчик. Далее все действия происходят в `javascript`.

ОС Tizen предоставляет функции управления датчиками и данными датчиков. Основными характеристиками API-интерфейса датчиков являются: слушатели датчиков, информация о датчиках, типы датчиков, URI датчиков. Слушатель позволяет обнаруживать датчик и контролировать его доступность. Он получает событие от датчика и обрабатывает это событие. После создания слушателя для определенного датчика и подписки на события этого датчика, мы можем управлять этим датчиком. Приложение может получать данные только при изменении показаний датчика.

Также мы можем получить доступ к некоторым данным аппаратного обеспечения датчика:

- Имя датчика
- Поставщик датчика
- Тип датчика
- Разрешение
- Интервал срабатывания
- Диапазон измерений URI датчика определен в виде:

http://<vendor>/sensor/<category>/<sensor-type>/<sensor-name>

Датчик акселерометра относится к категории healthinfo. Поэтому для получения соответствующего дескриптора датчика приложение должно иметь привилегию http://tizen.org/privilege/healthinfo.

С помощью `tizen.systeminfo.getCapability('http://tizen.org/feature/sensor.accelerometer')` проверяем доступность акселерометра.

Затем получаем доступ к датчику `tizen.sensorservice.getDefaultSensor("ACCELERATION")`

Запускаем акселерометр, используя функцию `start()`. В случае успеха, с помощью метода `getAccelerationSensorData`, получим данные датчика ускорения.

Устанавливаем слушателя `setChangeListener`. При изменении показаний акселерометра слушатель будет реагировать на событие через заданный промежуток времени. Слушатель предоставит нам готовые для дальнейшего использования данные акселерометра.

После прочтения данных акселерометра необходимо проверить наличие подключения к сети Интернет, чтобы в случае отсутствия подключения данные помещались в локальную базу данных IndexedDB.

Схема работы с базой данных:

1. Открыть базу данных.
2. Создать хранилище объектов в базе данных, над которой будут выполняться наши операции.
3. Запустить транзакцию и выдать запрос на выполнение какой-либо операции с базой данных, например, добавление или извлечение данных.
4. Ждать завершения операции, "слушая" событие DOM, на которое должен быть установлен наш обработчик.
5. Обработать результат.

Открытие базы запускается запросом

```
var request = indexedDB.open('AccelerometerBase', 1);.
```

Вызов функции `open()` вернет объект `IDBOpenDBRequest`, содержащий результат (в случае успеха) или ошибку, которую можно обработать как событие. Результат функции всегда возвращает экземпляр объекта `IDBDatabase.`

Второй параметр метода `open()` - это версия базы данных. Версия определяет схему базы данных - хранилище объектов и их структуру.

На запрос `open()` устанавливаем три обработчика: `onerror, onsuccess, onupgradeneeded.` При создании новой базы данных или при увеличении версии существующей запускается событие `onupgradeneeded.` В обработчике этого события создается хранилище объектов, необходимое для этой версии базы данных. Хранилище создается так:

```
createObjectStore("AccelerometerStore", { keyPath: " time" }).
```

В качестве ключа используется время, когда произошло измерение показаний датчика акселерометра.

Если всё в порядке , то инициируется событие успеха (это событие DOM, свойство type которого выставлено в "success"). Это вызывает запуск функции `onsuccess(event)` с событием успеха в качестве аргумента. Теперь мы можем получить экземпляр класса `DBDatabase` через `event.target.result.`

Чтобы записать или прочитать данные из базы данных нужно создать транзакцию. Функция `transaction()` принимает два аргумента и возвращает объект транзакции. Первый аргумент - это список объектов, которые транзакция будет охватывать. Второй аргумент – это режим открытия транзакции. Транзакции имеют три режима (только для чтения, для чтения и записи и для изменение версии). Транзакции могут принимать события DOM трех разных типов: `error, abort` и `complete`. В случае успеха вызывается функция `oncomplete()` внутри которой можно считывать `get()` или записывать `add()` данные.

Каждый раз при получении новых данных от акселерометра выполняется проверка подключения с сети Интернет. При наличии подключения в начале проверяется база данных. В случае наличия данных в базе необходимо их считать для отправки в социальную сеть Twitter. Поскольку необходимо считать все значения из хранилища объектов можно использовать курсор.
При использовании курсора не требуется знать ключ.

Функция `openCursor()`генерирует событие `event.` В случае успеха мы получим курсор `event.target.result.` Для дальнейшего чтения из базы необходимо вызвать метод `continue().` До тех пор, пока хранилище не

пусто, `event.target.result` будет возвращать данные, иначе - `undefined`.

После прочтения данных из базы их необходимо отправить в Twitter. Для этого необходимо на клиенте сформировать запрос к серверу.

Для формирования запроса со стороны клиента используется API XMLHttpRrequest. Этот API пзволяет осуществлять HTTP-запросы к серверу без перезагрузки страницы.

План работы с объектом XMLHttpRequest можно представить следующим образом:

1. Создание экземпляра объекта XMLHttpRequest
2. Открытие соединения
3. Установка обработчика события
4. Отправка запроса.

Вначале создается экземпляр объекта `XMLHttpRequest`. После создания объекта `XMLHttpRequest` необходимо вызвать метод `open()` для инициализации. Метод `open()` принимает три параметра: тип запроса (в нашем случае `POST`), адрес запроса и третий необязательный параметр - логическое значение `true или false,` указывающее, будет ли запрос осуществляться в асинхронном режиме. В нашем приложении запрос осуществляется в асинхронном режиме.

При асинхронном запросе объект `XMLHttpRequest` использует свойство `readyState` для хранения состояния запроса. Состояние запроса представляет собой число. Для получения ответа от сервера необходимо установить

обработчик на событие `readystatechange`. Событие `readystatechange` возникает каждый раз, когда изменяется значение свойства `readyState`.

После инициализации запроса необходимо его отправить с помощью метода `send(body)`. После этого начинает работать вышеуказанный обработчик событий. В обработчике обычно происходит перехват всех возможных кодов состояния запроса и вызов соответствующих действий, а также перехват возможных ошибок.

Для написания сервера использовалась программная платформа Node.js. Node.js использует модульную систему. При написании сервера использовались такие модули, как express, body-parser, twitter. Для загрузки модулей применяется функция `require()`, в которую передается название модуля. После получения модуля становится доступным весь определенный в нем функционал.

Для того, чтобы воспользоваться модулем twitter, необходимо получить API ключи. Для получения ключей в разделе для разработчиков Twitter (https://apps.twitter.com/) регистрируется новое приложение, после чего мы получаем доступ к ключам. Создаем клиента с заданными ключами и отправляем post-запрос в Twitter.

3.3. Анализ возникших сложностей

Основной проблемой при разработке данного приложения стало взаимодействие с Tizen SDK.

В первую очередь мы столкнулись с невозможностью установки данной среды разработки (Tizen Studio 2.0), либо сопутствующего ей программного обеспечения. Предпринимались попытки установить программу на такие устройства как:

Ноутбук Lenovo ideapad 320S-151 с техническими характеристиками:

- Model Name: 80Y9
- System: Ubuntu 16.04.3 LTS x86_64
- Kernel: 4.10.0-42-generic DE: Unity Session: ubuntu
- Processor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz
- Memory (Gb): 3.8
- Video: 00:02.0
- VGA compatible controller: Intel Corporation Device 1921 (rev 0a)
- Subsystem: Lenovo Device 39cc
- Kernel driver in use: i915 — 01:00.0
- 3D controller: NVIDIA Corporation Device 134f (rev a2)
- Subsystem: Lenovo Device 39cc

Ноутбук HP серии CND439C0B5

- System: Ubuntu 16.04.3 LTS x86_64
- Kernel: 4.10.0-42-generic DE: Unity Session: ubuntu
- Processor: Intel(R) Celeron(R) CPU N2840 @ 2.16GHz
- Memory (Gb): 3.8
- Video: 00:02.0
- VGA compatible controller: Intel Corporation Atom Processor Z36xxx/Z37xxx Series Graphics & Display (rev 0e)
- Subsystem: Hewlett-Packard Company Atom Processor Z36xxx/Z37xxx Series Graphics & Display Kernel driver in use: i915

Установка производилась в соответствии с информацией, указанной на сайте разработчиков Tizen - https://developer.tizen.org/node/22675.

После того, как на два компьютера удалось установить программное обеспечение в полном объёме и работать с ним, не сталкиваясь с ошибками, возникла проблема большой ресурсоёмкости этой SDK. Даже самые простые действия по редактированию текста кода обрабатывались в Tizen SDK слишком долго. Вследствие этого нормально продолжать разработку удалось лишь на одном из компьютеров, а именно на ноутбуке HP.

Следующей большой проблемой стала невозможность пользоваться веб симулятором данной SDK. Все обращения к системным функциям самой ОС Tizen, и в частности обращения к датчику акселерометра квалифицировались веб-симулятором как ошибка. Никакого пути обхода данной проблемы, кроме

использования “родного” эмулятора устройств Tizen. Это в значительной степени осложнило отладку той части приложения, в которой происходит взаимодействие с собственным сервером. Так, несмотря на то, что написанный сервер вполне работоспособен при обращении к нему через программу Postman (набор инструментов тестирования API), если обратиться к нему же из клиента, запускаемого в Tizen, запросы клиента получены не будут.

Также возникли проблемы при работе с базой данных IndexedDB. Для считывания данных из базы можно пользовать так называемый курсор. Для этого, среди прочего нужна функция continue(), которая распознается синтаксическим анализатором Tizen как ошибка, поскольку за таким ключевым словом в языке JavaScript должен следовать идентификатор.

Нужно отметить, что у среды разработки Tizen есть и определённые преимущества. Среди них можно выделить простоту разработки, для тех, кто не обладает специальными знаниями о внутреннем устройстве операционной системы. Кроме того, доступна хорошая документация с подробными объяснениями и примерами кода.

4. Заключение

Было проведено исследование различных средств разработки самих по себе, а также средств разработки в ОС Tizen. Результатом работы стало приложение, считывающее данные с датчика акселерометра и отправляющее их в социальную сеть Twitter.

Код разработанного приложения представлен в открытом репозитории по адресу https://github.com/MariaSev/TizenAccelerometerApp.

В перспективе приложение может быть расширено за счёт реализации алгоритма распознавания жестов и расширения пользовательского интерфейса, относящегося к взаимодействию пользователя, приложения и социальной сети.

5. Список литературы

1. AJAX [В Интернете] // https://metanit.com/web/javascript/13.1.php.
2. Device Sensors [В Интернете] // https://www.tizen.org/.
3. Tizen Developers [В Интернете] // https://developer.tizen.org.
4. Twitter Open Node Library [В Интернете] // https://www.npmjs.com/package/twitter.
5. Знакомство с Tizen [В Интернете] // https://habrahabr.ru/.
6. Использование IndexedDB [В Интернете] // https://developer.mozilla.org/ru/docs/IndexedDB/Using_IndexedDB.
7. Современный учебник Javascript [В Интернете] // https://learn.javascript.ru/.
8. Документация Node.js [В Интернете] //https://nodejs.org/en/docs/.