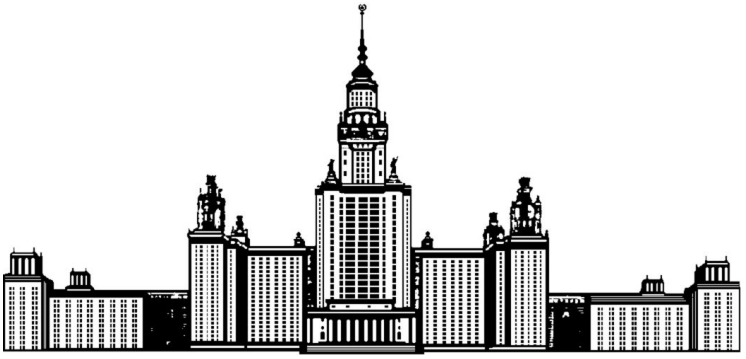

Московский государственный университет им. М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра автоматизации систем вычислительных комплексов

Чайчиц Даниил Александрович

Разработка и реализация клиентской части платформы для сбора и обработки физиологических данных о человеке

Курсовая работа

Научный руководитель:

Бахмуров Анатолий Геннадьевич

Консультант:

Любицкий Александр Андреевич

Москва 2020

Аннотация

В данной курсовой работе решается задача создания клиентской части платформы для сбора и обработки физиологических данных о человеке. Рассматривается умный жилет Hexoskin и смартфон с операционной системой Android, способы их взаимодействия между собой и с сервером хранения и обработки данных. В рамках практической части разрабатывается приложение для смартфона, реализующее передачу данных с жилета на сервер, готовится исследование его потребления энергии в зависимости от режима работы.

Содержание

1. Введение

С ростом частоты использования Интернета вещей в разных отраслях, медицинское пространство не стало исключением. На основе связной технологической инфраструктуры операторов беспроводной и проводной связи выстраиваются фрагменты глобального Internet of Medical Things (IoMT). Появляется всё больше и больше носимых устройств, способных считывать биометрические данные человека. Все эти данные попадают на сервера, а потом в руки врачей.

Крайне важно получать медицинские показатели не только во время посещения лечебных заведений, но и в течение некоторого непрерывного времени. Эти данные можно использовать для диагностирования заболеваний, которые невозможно выявить при кратком осмотре, прогнозирования состояния человека, составления персональных рекомендаций. Поэтому проблема мониторинга состояния человека является очень важной в наше время. Уже существуют носимые приборы, позволяющие измерять определенные характеристики, но их возможности ограничены лишь небольшим числом параметров, которые можно записать. Кроме того, порой требуется прикреплять к телу достаточно много датчиков, что затрудняет их ежедневное ношение.

В наше время появляется все больше функций слежения за здоровьем в различных умных часах, фитнес браслетах и других привычных для ежедневного ношения устройствах. Но основной проблемой является невозможность получения необработанных данных и закрытость исходного кода. Таким образом, создание открытой платформы сбора и обработки физиологических данных является актуальной задачей на сегодняшний день.

В данном проекте в качестве умного носимого устройства выступает умная одежда Hexoskin, которая имеет открытый программный интерфейс, и исходный код библиотек для работы с ней находится в общем доступе. В связи с новым по отношению к привычному формату форм-фактору, с помощью этого устройства имеется возможность считывать гораздо больше показателей.

Данная работа посвящена созданию мобильного приложения для сбора и обработки физиологических данных о человеке. Описано устройство, с помощью которого производится снятие показателей с человека, описан протокол Bluetooth Low Energy, проведен обзор способов хранения данных на устройстве, заданы требования для эксперимента по выявлению оптимального режима энергопотребления.

2. Цель и задачи

2.1. Цель

Требуется реализовать считывание данных с умного жилета Hexoskin по протоколу Bluetooth Low Energy, организовать их хранение на устройстве и отправку на сервер по протоколу MQTT (Рис. 1).

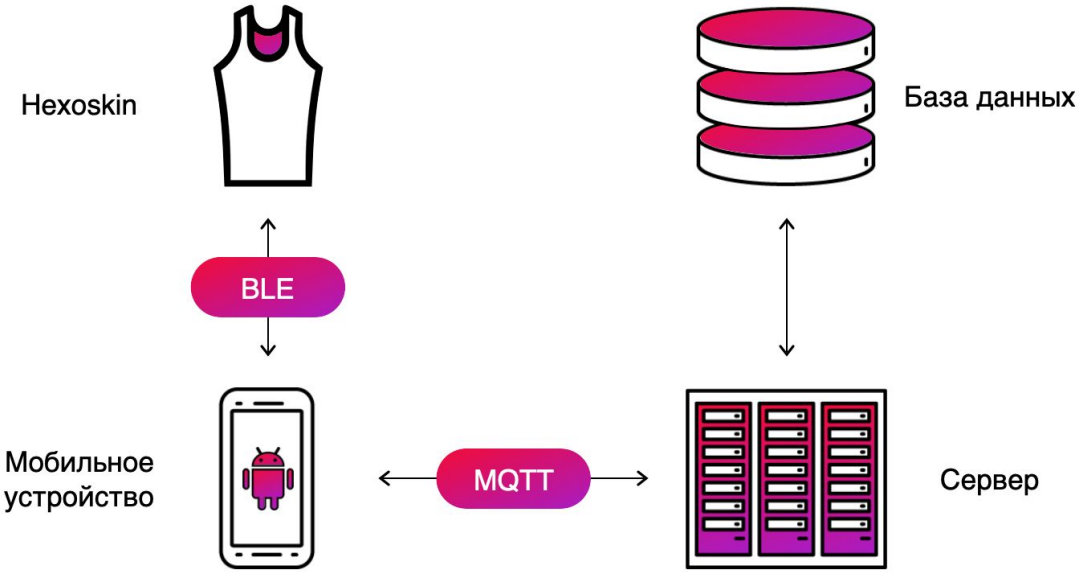

Рис. 1. Схема работы приложения

2.2 Задачи

1. Изучить умное носимое устройство, выяснить, какие данные могут быть с него получены.[3]
2. Изучить протоколы BLE и MQTT и порядок работы с ним в ОС Android.[1][2][9][11]
3. Сделать обзор способов хранения данных в ОС Android.[5][6]
4. Изучить алгоритмы аутентификации пользователя в ОС Android.[10]
5. Написать приложение на языке Java для мобильного устройства.[1][3][8]
6. Экспериментальным путем выяснить оптимальный режим работы для минимизации потребления заряда батареи.[7]

3. Носимое устройство и протокол BLE

3.1 Умная одежда Hexoskin

В качестве носимого устройства выступает умный жилет Hexoskin. Это устройство позволяет считывать с человека следующие показатели[3]:

- частоту сердцебиения;
- вариабельность сердечного ритма;
- число шагов;
- частота дыхания;
- VO2 max;
- объем легких;
- уровень активности;
- число шагов в минуту(каденс);

Рис. 2. Умный жилет Hexoskin

Жилет подключается к смартфону посредством протокола Bluetooth Low Energy.

3.2 Bluetooth Low Energy

Отличием этого протокола от обычного Bluetooth является сверхмалое пиковое энергопотребление, среднее энергопотребление и энергопотребление в режиме простоя.[11] Эти параметры являются очень важным критерием для мобильности системы мониторинга показателей человека. Достигается это следующим способом: вещание идет короткими сообщениями с разрывом подключения после передачи информации. Таким образом, большую часть времени устройства “спят”, не расходуя энергию на поддержание подключения.

Стек протоколов содержит 3 уровня: уровень приложения, хоста и контроллера. Уровень приложения — самый высокий уровень стека протоколов (Рис. 3.).

Уровень хоста содержит следующие подуровни:

- GAP (Generic Access Profile) — профиль общего доступа,
- GATT (Generic Attribute Profile) — профиль общих атрибутов,
- L2CAP (Logical Link Control and Adaptation Protocol) — протокол логического соединения и адаптации,
- ATT (Attribute Protocol) — протокол атрибутов,
- SM (Security Manager) — менеджер безопасности,
- HCI (Host Controller Interface) — интерфейс хост-контроллер, часть на стороне хоста,

Хост связан с контроллером протоколом HCI и имеет уровни:

- HCI — интерфейс хост-контроллер со стороны контроллера,
- LL (Link Layer) — канальный уровень,
- PHY — физический уровень.

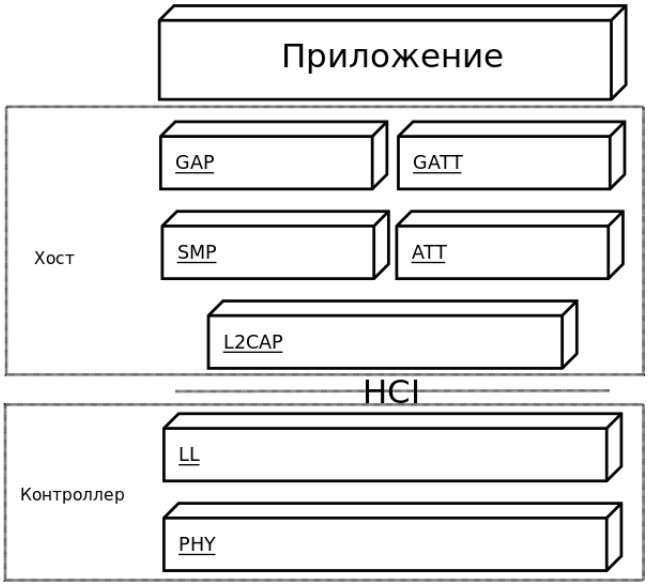

Рис. 3. Стек протоколов Bluetooth Low Energy

4. Хранение данных на мобильном устройстве

4.1 Обзор способов хранения

В связи с возможными разрывами в подключении к сети Интернет моментальная отправка данных на сервер после получения с жилета не является хорошей стратегией, так как велик риск потерять часть информации. Поэтому после того, как данные с носимого устройства будут получены, необходимо их сохранять до момента отправки. В ОС Android на выбор доступны следующие механизмы хранения данных: внутреннее хранилище, внешнее хранилище, SharedPreferences и база данных[6].

1. **Внутреннее хранилище**

   Каждое приложение Android имеет свой собственный внутренний каталог хранения, взаимодействующий с ним, в котором приложение может хранить текстовые и двоичные файлы. Файлы внутри этого каталога недоступны для пользователя или других приложений, установленных на устройстве пользователя. Они также автоматически удаляются, когда пользователь удаляет приложение. Таким образом, это обычный текстовый файл, в который можно записать данные и откуда их потом можно прочесть. Для чтения конкретного значения придется искать его в файле вручную. Объем вмещаемой информации будет ограничен объемом свободной памяти во внутреннем хранилище смартфона.

2. **Внешнее хранилище**

   Поскольку внутреннее хранилище устройств Android обычно фиксировано и часто довольно ограничено, некоторые устройства Android поддерживают внешние носители данных, такие как съемные microSD-карты. Это все еще будет файл, но в который можно будет записать гораздо больший объем данных.

3. SharedPreferences

Средства SharedPreferences позволяют приложениям сохранять настройки или другие данные в виде пар ключ-значение. Эти данные сохраняются даже когда пользователь закрывает приложение. Android хранит SharedPreferences в виде XML файла. SharedPreferences зависят от приложения, то есть сохраненные данные будут удалены из Android устройства после удаления приложения или после очистки данных приложения. Это также является файлом, но поиск и запись в нем потребуют меньше времени.

4. База данных

Если использовать базу данных, нужно быть уверенным, что каждое устройство будет её поддерживать. В этом случае в качестве единственного кандидата на выбор остается СУБД SQLite, так как она нативно поддерживается операционной системой android и уже есть у каждого в смартфоне. Каждое приложение для Android может создавать и использовать базы данных SQLite для хранения больших объемов структурированных данных. Размер индексируемой памяти для этой СУБД равен 140 Тб, а реальный размер будет ограничен размером внутренней памяти.

4.2 Результаты обзора

Таким образом мы получаем следующее:

	Размер	Быстрый доступ к данным	Исполнение запросов
Внутреннее хранилище	ограничен размером внутренней памяти	Нет	Нет
Внешнее хранилище	ограничен размером внешней памяти	Нет	Нет
SharedPreferences	ограничен размером внутренней памяти	Есть	Нет
SQLite	ограничен размером внутренней памяти	Есть	Есть

Таблица 1. Результаты обзора

Исходя из полученных результатов (Таблица 1.) оптимальным вариантом является использование СУБД SQLite. Данные, которые будут в ней храниться структурированы, что позволит производить предобработку показаний с помощью SQL запросов в будущем.

5. Аутентификация пользователей

Важным условием в работе системы является безопасность данных. Поэтому в приложении реализован механизм аутентификации и авторизации пользователей. Не пройдя этот процесс, человек не сможет получить возможность подключаться к носимому устройству.

В процессе получения доступа используются следующие понятия: идентификация - процесс определения, что за человек перед нами, часто он совмещается с аутентификацией - процессом проверки подлинности введенных данных. В случае успешной проверки происходит авторизация - предоставление доступа к функциональности.

При подключении к брокеру есть возможность указать в параметрах данные аутентификации - логин и пароль. Данные поступают на брокер, он отправляет запрос в базу данных, в которой лежит информация об учетных записях пользователей. В случае положительного ответа подтверждение придет и от брокера[10].

Данный процесс изображен на схеме ниже:

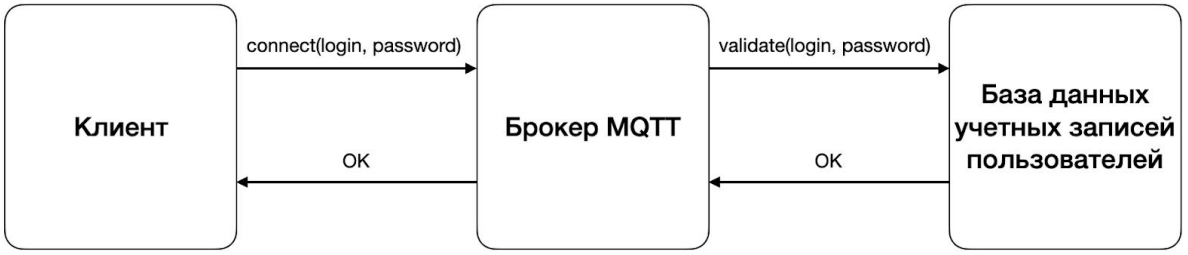

Рис. 4. Аутентификация пользователя

6. Архитектура приложения

После запуска приложения перед пользователем открывается окно входа в систему. Получив логин и пароль, приложение использует их для создания соединения по MQTTS, после чего в случае ответа от сервера о правильности данных, брокер отправляет подтверждение устройству. В случае ошибки пользователю будет предложено ввести данные заново. После успешного входа приложение переходит в работы с носимым устройством.

Пользователь имеет возможность выбирать, к какому носимому устройству подключаться. После выбора такового приложение устанавливает соединение и начинает приём данных. Параллельно с этим начинает жизнь второй процесс - Heartbeater. Из название следует его роль: периодически с некоторым интервалом он оживает и проверяет есть ли соединение с сервером и возможность принятия данных. В случае наличия он отправляет содержимое базы данных и очищает последнюю. Если же нет возможности что-то отправить, то процесс снова засыпает до следующего “удара”.

Архитектуру и основные алгоритмы приложения можно представить в виде диаграммы:

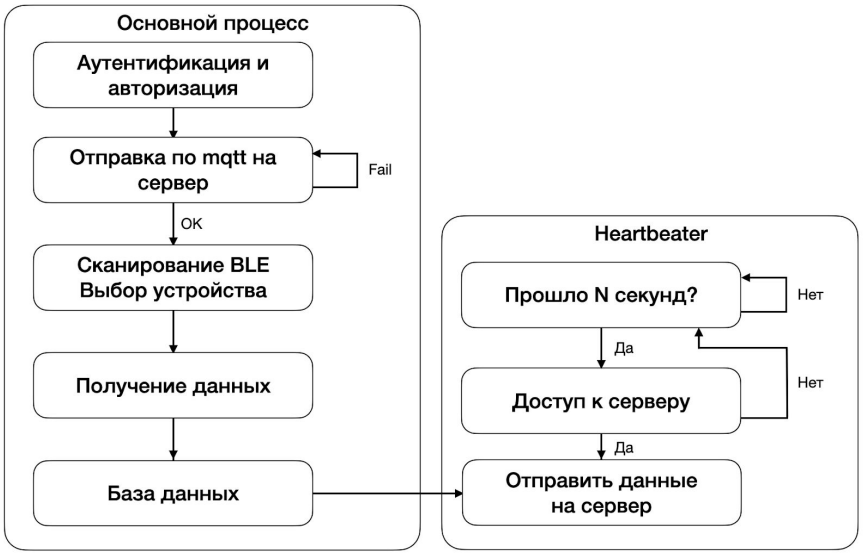

Рис. 5.Диаграмма работы приложения

Важным пунктом является протокол передачи данных между смартфоном и сервером. Для его выбора необходимо определить, каким требованиям должен удовлетворять этот протокол.

Известность - нужен достаточно популярный формат, проверенный на деле, чтобы избежать ошибок при создании собственного.

Неизбыточность - чем меньше служебных данных, тем лучше и быстрее будет происходить передача.

Поддерживаемость - средства разработки, а именно Java и Python3, языки на которых написаны клиент и сервер, должны поддерживать этот формат, чтобы воспользоваться готовыми библиотеками для работы с ним.

Кроссплатформенность - так как используются разные языки программирования, то необходимо, чтобы не возникло конфликтов с типами данных.

Поэтому в качестве формата передачи данных был выбран JSON. В нем практически нет избыточности, и Python3, и Java имеют библиотеки для легкой работы с ним, кроме того этот формат повторяет тип dictionary в Python3, что сильно упрощает работу с ним на стороне сервера. На слайде вы можете видеть пример сообщения с данными. В сообщении отсутствует идентификатор пользователя, так как все сообщения от конкретного пользователя будут посылаться в топик с его идентификатором на сервере-брокере.

Пример протокольной единицы:

```
{
    ”Clitime”: ‘2020-05-10 12:35:38’,
    ”Millisec”: 341,
    ”Steps”:146,
    ”RespRate”:2,
    ”Cadence”:212,
    ”Insp”:1895.833,
    ”Exp”: 1907.265,
    ”Activity”:4.437366,
    ”HeartRate”:71
}
```

7. Требование к эксперименту

В предыдущих главах неоднократно было упомянуто время между отправкой данных на сервер, но нигде не было указано конкретное значение. Дело в том, что требуется подобрать оптимальный временной интервал для того, чтобы снизить энергопотребление. Увеличение частоты ведет к увеличению потребления за счет времени, когда по окончании передачи радиоинтерфейс потребляет энергию перед переходом в режим экономии (tail-time по терминологии [4]). В то же время, если следующая передача состоится в пределах времени tail-time, дополнительных затрат энергии не возникнет. Если передавать данные редко, но большими фрагментами, влияние накладных расходов на включение и выключение радиоинтерфейса снижается[4].

Из доступных нам показателей расхода имеется только процент заряда аккумулятора, поэтому придется ориентироваться на него.
Эта величина меняется в зависимости от очень многих факторов[7].

С целью уменьшения погрешности измерения, необходимо придерживаться следующих правил:

- использовать одну и ту же часть шкалы (дело в том, что изменение на один процент заряда на разных частях шкалы означает изменение на разный заряд, например, от 100% до 99% обычно разряжается быстрее, чем от 90% до 89%);
- перед каждым тестом перезагружать устройство;
- останавливать все ненужные процессы
- во время одного теста разряжать устройство более чем на 10%, чтобы погрешность измерения была не очень велика;
- не загружать и не запускать никаких новых приложений во время серии тестов, они в фоне могут потреблять батарею;
- не менять настройки устройства: яркость экрана, настройки сети и т.д.;

- не перемещать устройство на существенные расстояния (более нескольких метров) во время теста или при каждом тесте перемещать его по одной и той же траектории.

Главными потребителями в нашем случае будут

- работа CPU (один из главных потребителей);
- подсветка экрана (сравнима с потреблением CPU);
- работа с сетью (отправка данных);
- частые получения GPS-координат (необходимо для работы Bluetooth)

Для исследования потребления можно использовать специальный профилировщик, но для минимального исследования достаточно воспользоваться встроенной утилитой в Android, показывающей расход батареи на каждое приложение. Параметром, который нам нужно будет оптимизировать является время накопления показателей в базе данных.

8. Заключение

В рамках курсовой работы была поставлена цель разработать клиентскую часть платформы для сбора и обработки физиологических данных о человеке.

В ходе работы был изучен язык программирования Java, исследованы такие протоколы связи с другими устройствами, как Bluetooth Low Energy и MQTT, изучены механизмы работы с базой данных в ОС Android, придуман алгоритм работы клиентской части и реализован в виде приложения. Также была подготовлена база для экспериментального исследования.

В рамках дипломной работы будет необходимо провести экспериментальное исследование, проработать все сценарии и механизмы защиты приложения, встроить предобработку данных и улучшить дизайн.

9. Список литературы

1. Android Developers Community: https://developer.android.com/
2. Eclispe MQTT Android: https://github.com/eclipse/paho.mqtt.android
3. Hexoskin SDK: https://bitbucket.org/carre/hexoskin-smart-demo/src/master/hexoskin-smart-android/
4. Niranjan Balasubramanian, Arun Venkataramani. “Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications”. 2009. Available: https://people.cs.umass.edu/~arun/papers/TailEnder.pdf
5. SQLite Android: https://startandroid.ru/ru/uroki/vse-uroki-spiskom/74-urok-34-hranenie-dannyh-sqlite.html
6. Хранение данных в Android: https://code.tutsplus.com/ru/tutorials/android-from-scratch-how-to-store-application-data-locally--cms-26853
7. Профилирование работы приложений Android: https://habr.com/ru/company/mailru/blog/263413/
8. Beautiful Android Login and Signup Screens with Material Design: https://sourcey.com/articles/beautiful-android-login-and-signup-screens-with-material-design
9. Bluetooth Low Energy on Android: http://developer.alexanderklimov.ru/android/theory/ble.php
10. Authentication with Username and Password - MQTT Security Fundamentals https://www.hivemq.com/blog/mqtt-security-fundamentals-authentication-username-password/
11. Bluetooth Low Energy https://ru.wikipedia.org/wiki/Bluetooth_с_низким_энергопотреблением