



Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

Голубкова Мария Сергеевна

**Разработка имитатора датчика
электрокардиограммы в составе приложения
имитации датчиков интернета вещей**

Выпускная квалификационная работа

Научный руководитель:

к.ф.-м.н.

Бахмуро́в Анатолий Геннадьевич

Научный консультант:

Любицкий Александр Андреевич

Москва, 2024

Аннотация

**Разработка имитатора датчика
электрокардиограммы в составе приложения
имитации датчиков интернета вещей**

Голубкова Мария Сергеевна

Данная выпускная квалификационная работа нацелена на создание Android приложения, имитирующего поведение датчиков интернета вещей в части передачи данных посредством классического Bluetooth и Bluetooth Low Energy.

В ходе работы были изучены фрагменты исходного кода от производителя для работы с имитируемым кардиографом, реализованы работа с Bluetooth, генерация данных, их передача и запись этих данных в файл журнала. Также была исследована зависимость величины потерь данных при передаче по Bluetooth от частоты отправки данных и величины пакетов в клиентском приложении сервиса сбора и обработки медицинской телеметрии.

Abstract

Development of electrocardiogram sensor simulator as part of application
for simulating Internet of Things sensors

Golubkova Mariia

This graduation thesis aims to develop an Android application that simulates the behavior of Internet of Things sensors in terms of data transmission via classic Bluetooth and Bluetooth Low Energy.

During the work, the source code fragments for working with simulated cardiograph were analyzed, work with Bluetooth, data generation, data transfer and recording of this data to a log file were implemented. The dependence of the amount of data loss via Bluetooth on the frequency of data sending and size of packets in the client application of the service for collecting and processing medical telemetry was also studied.

Содержание

1 Введение	6
2 Приложение имитации датчиков интернета вещей	8
2.1 Пользовательский интерфейс	8
2.2 Генерация данных	9
2.3 Запись трассы	9
2.4 Необходимость доработки приложения	10
3 Постановка задачи	11
4 Актуальность задачи	12
4.1 Потребности тестирования	12
4.2 Актуальность	12
4.3 Обзор существующих решений	13
5 Система кардиореабилитации «Аккордикс»	14
5.1 Кардиорегистратор KP-2	14
5.2 Анализ исходного кода системы «Аккордикс»	15
5.3 RFC-1055	16
6 Bluetooth Classic	18
6.1 Bluetooth Classic vs BLE	18
6.2 Топология сетей Bluetooth	19
6.3 Основные протоколы	20
6.3.1 SDP	20
6.3.2 RFCOMM	21
6.4 Одновременное использование Bluetooth Classic и BLE	21
7 Реализация на языке Kotlin	22
7.1 Реализация работы Bluetooth Classic	22
7.2 Сборка пакетов	24
7.3 Пользовательский интерфейс	26

8 Экспериментальное исследование	27
8.1 Работа с приложением Аккордикс ЭПП	27
8.2 Оценка работоспособности клиентской части сервиса сбора и обработки медицинской телеметрии	29
9 Заключение	32
Список литературы	33

1 Введение

В последние годы интернет вещей (Internet of Things, IoT) стал неотъемлемой частью технологического прогресса. IoT представляет собой сеть устройств, соединенных между собой, которые могут обмениваться данными и функционировать автономно без человеческого участия. Одной из наиболее перспективных областей применения IoT является динамично развивающийся рынок интернета медицинских вещей (Internet of Medical Things, IoMT).

IoMT — это новый способ реализации медицинских услуг и внедрения инноваций в здравоохранении. Системы мониторинга, дистанционной диагностики, устройства для жизнеобеспечения и управление лекарствами — все это включено в понятие интернета медицинских вещей. С помощью IoMT предоставляются новые возможности для мониторинга здоровья пациентов, снижения затрат на медицинское обслуживание и повышения качества жизни пациентов и медицинского персонала.

Выпускные квалификационные работы Фролова Александра[1] и Князева Егора[2] посвящены развитию сервиса по сбору и обработке медицинской телеметрии. Серверная часть позволит медицинским работникам дистанционно наблюдать за состоянием пациентов и своевременно реагировать на серьёзные изменения в их показателях здоровья, а клиентская часть обеспечит сбор данных с носимых устройств, отображение информации о них пользователю и их передачу на сервер.

Применение IoMT включает разнообразные устройства, такие как умные часы, наушники, браслеты, датчики температуры, сердечного ритма и артериального давления, а также дистанционные приборы и машины для диагностики и лечения пациентов. Количество новых устройств на рынке стремительно растёт, например, в рамках обзора устройств в ходе данной работы были обнаружены такие интересные приборы, как сенсорные стельки для предотвращения диабетических язв стоп, серьга, которая отслеживает уровень сахара в крови и даёт обратную связь в реальном времени, или наушники, регистрирующие электрокардиограмму.

Сервис сбора и обработки медицинской телеметрии должен иметь возможность быстро адаптироваться под изменения рынка. Данная работа нацелена на разработку удобного инструмента, позволяющего развивать сервис.

Разработка приложения, имитирующего поведение датчика интернета вещей, может значительно помочь в развитии сервиса сбора и обработки медицинской телеметрии и других сервисов Интернета вещей. Такое приложение позволит имитировать работу реального датчика и генерировать тестовые данные, что поможет разработчикам проводить испытания и тестирование новых функций и алгоритмов обработки данных без необходимости использования настоящих устройств.

2 Приложение имитации датчиков интернета вещей

В ходе написания курсовой работы[3] было разработано мобильное приложение, имитирующее датчики Интернета вещей, посылающие небольшие объёмы данных с низкой частотой. Например, Smart часы, посылающие частоту сердцебиения (одно число) раз в секунду.

2.1 Пользовательский интерфейс

Начальный экран приложения позволяет пользователю выбрать тип имитируемого устройства. Список устройств был составлен в ходе выполнения курсовой работы на основе анализа литературы. На выбор предложены:

- Hexoskin[4] - смарт-рубашка с открытыми данными для мониторинга различных показателей здоровья человека (частота сердечных сокращений, частоты дыхания и т.д.) и других измерений активности, таких как подсчет шагов и частота вращения педалей.
- Ritmer[5] - смарт-браслет для мониторинга сердечной активности.
- Dexcom[6] - глюкометр с открытым исходным кодом.
- Smart весы - имитация поведения таких весов, как, например, Picooc[7] или Mi Body Composition Scale[8].
- Smart часы - имитация поведения таких часов, как Xiaomi Mi Smart Band[9].
- Универсальное устройство - возможность для пользователя составлять свой список передаваемых данных.

При выборе конкретного устройства (любого, кроме универсального) также выбирается и соответствующий конфигурационный файл, задающий список предоставляемых устройством данных. В настоящий момент такие файлы написаны для Hexoskin и Smart часов.

При переходе на промежуточный экран "Универсальное устройство" приложение устанавливает http-соединение с серверной частью сервиса сбора и обработки медицинской телеметрии и получает от сервера хранящиеся на нём конфигурационные файлы[10].

В соответствии с полученным списком на экране динамически[11] генерируются кнопки для выбора имитируемого устройства.

Таким образом для имитации устройства по индивидуальным запросам необходимо прежде всего написать конфигурационный файл, задающий список передаваемых данных. В настоящий момент приложение может получать только загруженные на сервер авторизованными пользователями файлы, однако в будущем можно будет предоставить возможность для загрузки конфигурационного файла, например, из памяти смартфона.

На втором экране в соответствии с конфигурационным файлом генерируется динамический список[11] всех предоставляемых данных (например, частота сердцебиения и/или частота дыхания), и для каждого показателя предоставляются поля для установки минимального и максимального значений и частоты их обновления.

2.2 Генерация данных

После настройки значений и частот для каждого передаваемого показателя в сопрограммах[12] запускаются функции, в которых асинхронно, с задержкой, определённой пользователем при настройке параметров, из промежутка значений, определённого пользователем, выбираются случайные числа. Число присваивается соответствующему показателю, после чего подключенному устройству отправляется уведомление о его обновлении. В данный момент в приложении реализована корректная генерация значений только для стандартного сервиса частоты сердцебиения.

2.3 Запись трассы

Для тестирования работы клиентского приложения необходимо сравнение полученных от датчика данных с данными, впоследствии поступившими на сервер, также интерес представляет временная задержка при передаче данных. Поэтому при подключении стороннего устройства к разрабатываемому приложению, все данные, отправляемые ему, записываются в текстовый файл вместе с названием характеристики и временем отправки. После окончания работы становится доступной кнопка SHARE LOGS, с помощью которой созданный файл можно отправить по почте или в любые мессенджеры.

2.4 Необходимость доработки приложения

Как уже было сказано ранее, приложение посылает небольшие объёмы данных с низкой частотой, но есть медицинские данные, которые обновляются непрерывно (точнее очень часто) в течение некоторого отрезка времени. Самым ярким примером подобных данных является электрокардиограмма (ЭКГ).

Использование ЭКГ в устройствах IoT предоставляет возможность эффективного мониторинга сердечного здоровья, раннего обнаружения проблем (таких как аритмия, ишемическая болезнь, а также профилактика инсультов и инфарктов) и улучшения качества жизни пациентов. В настоящее время множество производителей представляют кардиографы, для соединения с смартфоном использующие технологии беспроводной связи, такие как Bluetooth, Wi-Fi и другие[13].

Доработка приложения и добавление возможности работы с данными ЭКГ поможет в развитии сервиса сбора и обработки медицинской телеметрии и других сервисов Интернета вещей. Можно заново сформулировать цель разработки: создание приложения, позволяющего пользователям выбрать используемый протокол (Bluetooth Classic и BLE) и задавать параметры генерации и передачи данных по нему (такие, как интервалы числовых значений параметров, частота генерации данных, формат данных/пакетов). Также пользователи могут получить доступ к трассе сгенерированных данных для дальнейшего анализа работоспособности их приложений и сервисов, а также для оценки технических характеристик используемых протоколов, например задержек или потерь данных при передаче. Использование приложения требует участия пользователя только при начальной настройке его работы (т.е. выбор протокола, формата и параметров передачи), генерация данных, их передача и запись трассы происходят без вмешательства со стороны пользователя.

3 Постановка задачи

Целью данной работы было:

Разработать имитатор датчика ЭКГ в части передачи пакетов с данными посредством Bluetooth Classic в составе приложения имитации датчиков интернета вещей и с его помощью исследовать работоспособность клиентской части сервиса сбора и обработки медицинской телеметрии.

Для достижения поставленной цели необходимо выполнить следующие **задачи**:

- доработать ранее созданное приложение, имитирующее поведение датчика Интернета вещей, добавив поддержку интерфейса Bluetooth Classic
- создать на базе приложения генератор пакетов с данными, подобными данным электрокардиограммы
- с помощью доработанного приложения исследовать работоспособность клиентской части сервиса сбора и обработки медицинской телеметрии при увеличении частоты передавачи данных.

4 Актуальность задачи

4.1 Потребности тестирования

Использование реальных датчиков может быть недостаточным при исследовании работоспособности приложений, взаимодействующих с устройствами IoT.

Использование лишь реальных датчиков делает невозможным:

- имитацию датчиков, недоступных в настоящий момент
- тестирование приложения при работе с несколькими датчиками, нужного количества которых может даже не быть в наличии
- отслеживание возникновения ошибок на уровне приложений, т.к. мы не можем знать сколько пакетов фактически передали, например, Smart часы
- контроль передачи всех пакетов для устройства в разных режимах работы, например в спящем режиме
- имитацию некорректной работы датчика или же его новых возможностей, планируемых к реализации

4.2 Актуальность

Решение задачи позволит:

- имитировать поточную передачу данных посредством Bluetooth Classic
- создать возможности для нагружочного тестирования клиентской части сервиса сбора и обработки медицинской телеметрии
- исследовать возможности Bluetooth Classic, оценить задержки и потери данных при различных условиях

При формулировании целей работы и исследовании актуальности поставленной задачи были попытки найти подобные решения в открытом доступе, однако никаких удобных инструментов для подключения и наблюдения передачи данных посредством Bluetooth Classic обнаружено не было.

4.3 Обзор существующих решений

Учитывая требования к приложению, были рассмотрены следующие существующие решения:

HCI sniffer - это инструмент, позволяющий перехватывать пакеты данных между уровнями контроллера и хоста. Сниффер обеспечивает получение данных до их попадания на уровень приложения, но не предоставляет никаких возможностей для нагружочного и функционального тестирования. Подобный инструмент существует в каждом смартфоне Android[14], а также на рынке есть множество аппаратных наблюдателей такого типа, их цена варьируется от 50 до 40 000 долларов[15].

BlueZ - это официальная реализация стека протоколов Bluetooth для Linux. Позволяет устанавливать соединение между ноутбуком и смартфоном и управлять им с помощью командной строки. С его помощью можно, например, писать Python скрипты[16], управляющие соединением. Но недостатки подобного решения очевидны: использование компьютерных программ менее удобно для рядового пользователя, а главное - делает затруднительным параллельное тестирование, требующее несколько устройств.

Nordic nRF Connect for Android[17], мобильное приложение, рассматриваемое в курсовой работе, предоставляет пользователю много возможностей по управлению соединением посредством BLE с понятным пользовательским интерфейсом, но не поддерживает работу с устройствами Bluetooth Classic.

Более того при доработке приложения была потребность быстро проверить работоспособность текущей версии, однако не было найдено ни одного мобильного приложения, которое могло бы подключиться к заранее заданному серверу Bluetooth Classic (об этом подробнее написано в разделе о реализации).

5 Система кардиореабилитации «Аккордикс»

«Аккордикс» — это система кардиореабилитации на основе технологии теле-ЭКГ[18]. Она состоит из следующих элементов:

- Двухканальный кардиорегистратор KP-2, который в процессе тренировки записывает данные ЭКГ, дыхания, а также темп движения и положение тела пациента. Полученные результаты передаются на смартфон с помощью заранее настроенного беспроводного канала Bluetooth.
- Смартфон с мобильным приложением. Пациент может видеть в приложении заранее созданный врачом курс тренировок. Данные обследований передаются по сети Интернет на удаленный сервер и сохраняются там. Просмотреть их можно в любое удобное время.
- Портал врача, содержащий базы данных пациентов и курсы тренировок. Врач может наблюдать за процессом тренировки в режиме реального времени или просматривать данные в записи.

Компания «Нейрософт» предоставила экземпляр кардиорегистратора, аккаунт для входа в приложение и портал, а так же фрагменты исходного кода, для изучения работы KP-2 в рамках развития проекта на кафедре. Именно поэтому при развитии нашего (кафедра АСВК факультета ВМК МГУ) сервиса для работы с данными ЭКГ за основу был взят KP-2.

5.1 Кардиорегистратор KP-2

KP-2, имея 5 электродов, позволяет проводить длительный мониторинг с использованием 2 ЭКГ-каналов, дополнительно регистрирует 3 канала двигательной активности и канал дыхания.

Электрокардиографы фиксируют напряжения, связанные с сердечной деятельностью, с помощью электродов, расположенных на определенных частях тела. Расположение электродов позволяет наблюдать за электрической активностью сердца под разными углами. Эти напряжения отображаются в виде «каналов» на записи ЭКГ.

Таким образом, каждый канал отображает дифференциальное напряжение между двумя электродами или дифференциальное напряжение между одним электродом и средним напряжением нескольких электродов. Различные комбинации измеряемых напряжений позволяют отображать больше каналов, чем имеется электродов.

5.2 Анализ исходного кода системы «Аккордикс»

На основе переданных изготовителем краткого описания и фрагментов исходного кода был проведён анализ протокола передачи данных между компонентами системы кардиореабилитации «Аккордикс».

Кардиограф может работать в одном из следующих режимов:

1. Ожидание команды
2. Режим мониторинга ЭКГ
3. Измерение импеданса электродов (но эта функциональность в данной работе не рассматривается)

После включения прибор переходит в режим ожидания команды. При переходе в режим измерения ЭКГ прибор передаёт пакеты с данными каналов ЭКГ, каналов акселерометра и дыхания.

Список рассматриваемых в работе команд, поступающих от смартфона к кардиорегистратору такой:

Цифровой код	Символьное имя (в исходном коде)	Назначение
0	ReadDeviceInformation	Чтение информации об устройстве
1	ReadDeviceStatus	Получить статус устройства
2	ReadComponentsStatus	Получить заряд батареи и температуру
4	MonitoringMode	Перейти в режим мониторинга ЭКГ (или завершить мониторинг ЭКГ)
10	PowerOff	Выключение устройства

Рис. 1

Вместе с командой MonitoringMode смартфон высылает частоту дискретизации (или сэмплирования), то есть частоту, с которой записываются значения непрерывного сигнала. Частота дискретизации принимает значение в 250, 500 или 1000 Гц, и от этого значения в дальнейшем будет зависеть размер приходящих пакетов. После команды

MonitoringMode приложение принимает и обрабатывает данные от KP-2, а затем снова отправляет команду 4, но с частотой равной 0 - признаком окончания записи ЭКГ.

KP-2 отвечает на приведённые команды (кроме PowerOff) следующим образом:

Цифровой код	Символьное имя (в исходном коде)	Назначение
0	ReadDeviceInformation	Чтение информации об устройстве
1	ReadDeviceStatus	Получить статус устройства
2	ReadComponentsStatus	Получить заряд батареи и температуру
254	Error	Сообщение об ошибке
255	MonitoringData	Высылка данных ЭКГ в режиме мониторинга (ответ на MonitoringMode)

Рис. 2

При отправке пакета с командой MonitoringData самим данным предшествует заголовок содержащий номер сэмпла аналого-цифрового преобразования (АЦП), статус, необходимый для детектирования обрывов электродов и вычисления частоты, сами обрывы, а также частота семплирования, отправленная со смартфона.

Сами пакеты имеют следующую структуру:

- Версия протокола - всегда 0x00
- Команда (один байт)
- Данные (массив байтов)
- Контрольная сумма пакета (сумма общей длины пакета в байтах и их значений)

5.3 RFC-1055

Перед отправкой пакет кодируется по принципу, описанному в стандарте RFC-1055, а после получения, соответственно, декодируется. Такой способ кодирования данных нужен для разделения потока байтов на пакеты.

RFC-1055 - это простейший способ инкапсуляции IP-дейтограмм для последовательных каналов связи. IP-дейтограмма должна завершаться специальным символом 0xC0. Во многих реализациях дейтограмма и начинается с этого символа. Если какой-то байт дейтограммы равен символу 0xC0, то вместо него передается двухбайтовая последовательность 0xDB, 0xDC. Октет 0xDB выполняет функцию ESC-символа. Если же

байт дейтограммы равен 0xDB, то вместо него передается последовательность 0xDB, 0xDD.

6 Bluetooth Classic

Как уже было сказано ранее приложение уже поддерживает передачу небольших объёмов данных, которые необходимо отправлять достаточно редко. Для этих целей используется Bluetooth Low Energy (BLE). Однако эта технология не подходит для передачи ЭКГ, при передаче непрерывных данных зачастую, как и в системе кардиореабилитации «Аккордикс», используется Bluetooth Classic.

6.1 Bluetooth Classic vs BLE

- Bluetooth Classic, используется в беспроводных громкоговорителях, автомобильных информационно-развлекательных системах и наушниках;
- Bluetooth Low Energy (BLE), т.е. Bluetooth с низким энергопотреблением. Он чаще всего применяется в приложениях, чувствительных к энергопотреблению или в устройствах, передающих небольшие объемы данных с большими перерывами между передачами (например, разнообразные сенсоры параметров окружающей среды или управляющие устройства, такие как беспроводные выключатели).

Таким образом, Bluetooth Classic позволяет посыпать данные в больших объемах и с большей частотой, однако это требует больших энергетических трат, поэтому используется, например, в беспроводных громкоговорителях, автомобильных информационно-развлекательных системах и наушниках. BLE чаще всего применяется в приложениях, чувствительных к энергопотреблению или в устройствах, передающих небольшие объемы данных с большими перерывами между передачами (например, разнообразные сенсоры параметров окружающей среды или управляющие устройства, такие как беспроводные выключатели). Поэтому в рамках данной работы было принято решение использовать Bluetooth Classic.

Подробнее о технологии BLE и реализации приложения имитатора датчиков можно прочитать в моей курсовой работе[3].

6.2 Топология сетей Bluetooth

Основные принципы[19]:

- Обмен информацией может осуществляться только между ведущим и подчиненным устройствами, при этом каждое устройство может быть как ведущим, так и подчиненным.
- Основным элементом организации сетей Bluetooth является пикосеть, состоящая из одного ведущего устройства и от 1 до 7 активных подчиненных устройств.
- В одну пикосеть может входить неограниченное количество устройств, находящихся в неактивном режиме.
- Подчиненное устройство может общаться только с ведущим, причем только тогда, когда это разрешает ведущее устройство.
- В каждый момент времени обмен данными может идти только между двумя устройствами в одном направлении.
- Любое устройство одной пикосети может также входить в другую пикосеть в качестве как подчиненного, так и ведущего.

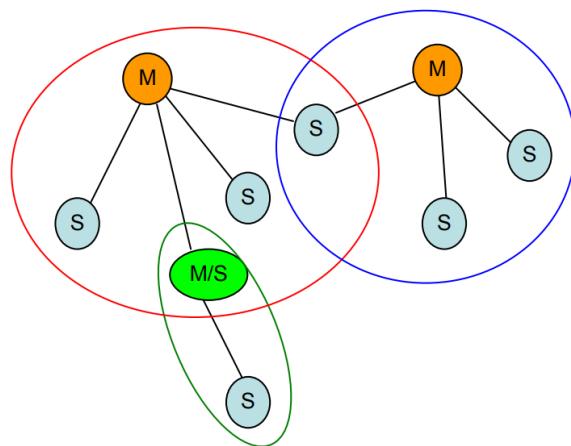


Рис. 3: Пример топологии пикосети

Такое чёткое описание топологии однозначно определяет роли устройств, рассматриваемых в данной работе: КР-2 и разрабатываемое приложение являются подчинёнными, а клиентское приложение - ведущим, так как именно оно инициирует соединение и передачу данных ЭКГ.

6.3 Основные протоколы

Устройства Bluetooth могут быть одного из этих двух типов:

- Одиночный режим — поддерживает профиль BR/EDR или LE.
- Двойной режим — поддержка профилей BR/EDR и LE.

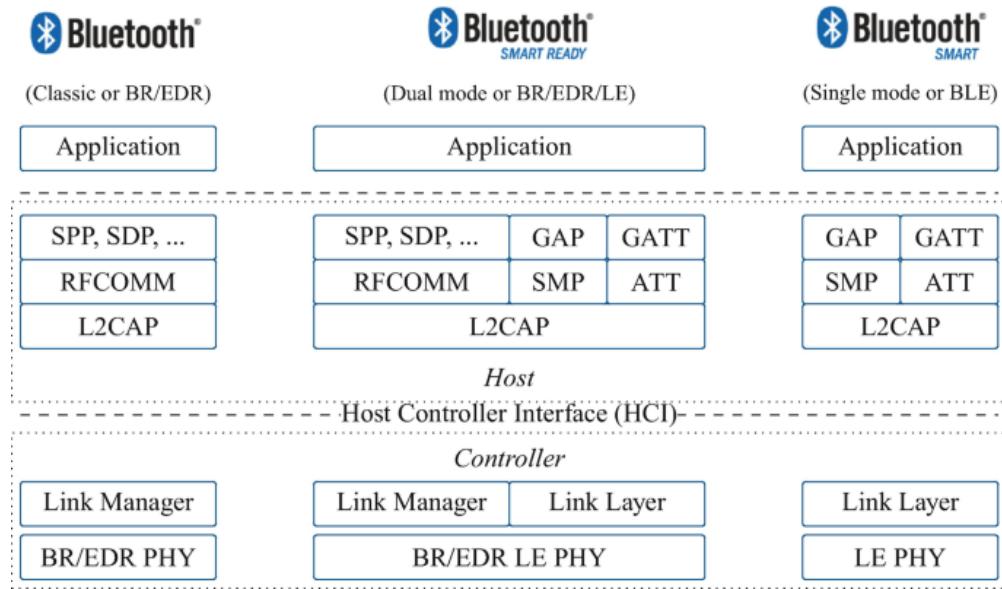


Рис. 4: Основные протоколы[20]

При разработке Bluetooth приложения на ОС Android необходимо обратить внимание на два протокола: SDP и RFCOMM[21].

6.3.1 SDP

Service discovery protocol (SDP) - протокол служб обнаружения, который предоставляет приложениям возможность запрашивать службы и характеристики службы,

после чего может быть установлено соединение между двумя или более устройствами Bluetooth.

При создании сервиса в системе с помощью SDP регистрируется запись со всей информацией о сервисе, в том числе об автоматически назначенному канале, который затем будет использован для подключения.

6.3.2 RFCOMM

Bluetooth device rfcomm protocol tdi (The Radio Frequency Communication Protocol Transport Driver Interface) предоставляет возможность установления виртуального последовательного RS232 порта между двумя Bluetooth-устройствами и осуществления обмена данными между ними.

Сервер открывает виртуальный последовательный порт, а клиент подключается к этому порту. После установления соединения между сервером и клиентом, они могут обмениваться данными с помощью последовательных команд и ответов.

Rfcomm предоставляет надежную передачу данных, используя протокол Bluetooth. Он осуществляет проверку целостности данных и обеспечивает организацию потоков передачи данных. RFCOMM также обеспечивает несколько одновременных подключений к одному устройству и позволяет подключаться к нескольким устройствам.

6.4 Одновременное использование Bluetooth Classic и BLE

Одновременное использование Bluetooth Classic и BLE возможно, так как классический Bluetooth и BLE — это почти две разные технологии, поэтому их параллельное использование в одном приложении не должно быть проблемой. Единственное ограничение, которое есть при работе с обеими технологиями, заключается в том, что нельзя одновременно сканировать классические устройства и устройства LE. Это упоминается в Руководстве разработчика Android BLE[20].

7 Реализация на языке Kotlin

После рассмотрения всех практических и теоретических аспектов, можно сформулировать список вновь добавляемых функциональных возможностей дорабатываемого приложения:

1. Установка соединения и передача данных посредством Bluetooth Classic
2. Регулирование частоты отправки пакетов (у КР-2 фиксировано 10 раз в секунду)
3. Генерация данных в соответствии с форматами данных, передаваемых прибором КР-2
4. Упаковка данных в пакет, соответствующий протоколу взаимодействия устройства с приложением системы «Аккордикс»
5. Кодирование полученного пакета по RFC-1055
6. Обмен командами с клиентским приложением
7. Запись трассы

7.1 Реализация работы Bluetooth Classic

Ниже приведена функция, устанавливающая соединение посредством Bluetooth Classic с приложением сбора медицинской телеметрии. На 3 строке создаётся прослушивающий защищенный сокет RFCOMM с заданным именем сервиса и идентификатором последовательного порта (SPP), который затем будет использоваться клиентским приложением для поиска нашего сервиса. В этот же момент создаётся запись SDP, содержащая это имя, идентификатор, а также автоматически назначенный канал, который другие устройства находят по идентификатору и используют для подключения. Эта запись SDP будет удалена при закрытии сокета или при неожиданном закрытии этого приложения[22].

При вызове метода `accept()` приложение получает входящих соединений от прослушиваемого сервера, а когда соединение устанавливается, метод возвращает сокет, от которого можно получить информацию о подключенном устройстве, а также потоки ввода и вывода для управления соединением.

```

1 private fun startBluetoothWork() {
2     Thread {
3         try {
4             prevName = bluetoothAdapter.name
5             bluetoothAdapter.name = "HS ECG SN00000001"
6             if (bluetoothAdapter.scanMode != 23) {makeDiscoverable()}
7             while (bluetoothAdapter.scanMode != 23) {continue}
8             serverSocket = bluetoothAdapter
9                 .listenUsingRfcommWithServiceRecord("My KR-2", SPP_UUID)
10            var shouldLoop = true
11            while (shouldLoop) {
12                rfcommSocket = try {
13                    serverSocket?.accept()
14                } catch (e: IOException) {
15                    Log.e("BluetoothClassic", "accept() method failed", e)
16                    shouldLoop = false
17                    null
18                }
19                rfcommSocket?.also {
20                    Log.d("BluetoothClassic", "accepted socket")
21                    val device = rfcommSocket!!.remoteDevice
22                    this@BluetoothClassicActivity.runOnUiThread {
23                        fastLayout.phoneName.text = device?.name
24                        fastLayout.phoneAddr.text = device?.address
25                    }
26                    connectedDevice = device
27                    inputStream = rfcommSocket!!.inputStream
28                    outputStream = rfcommSocket!!.outputStream
29                    jobCommand = scopeCommand.launch { manageSocket(
30                        rfcommSocket!!) }
31                    serverSocket!!.close()
32                    shouldLoop = false
33                }
34            } catch (e: IOException) {
35                Log.d("BluetoothClassic", e.stackTraceToString())
36            }
37        }.start()
38    }

```

Обмен данными с клиентским приложением происходит в функции manageSocket, а полный код приложения можно посмотреть по ссылке в списке литературы[23].

Так же, приведённый отрывок кода хорошо демонстрирует описываемое в п.6.2 распределение ролей.

7.2 Сборка пакетов

Для иллюстрации имитации работы кардиорегистратора KP-2 приведён код функции, вызываемой при получении от приложения команды ReadComponentsStatus.

В ответ устройство отправляет информацию об ошибках, уровень заряда батареи, напряжение и температура, которые генерируются случайным образом. Промежутки значений для генерации напряжения и температуры взяты таким образом, потому что приложение при получении этих данных напряжение делит на 1000, а температуру - на 10.

```
1 private fun sendComponentStatus() {
2     var p = Packet(CommandDict["ReadComponentsStatus"]!!)
3     var status = ComponentsStatusData(ResultDict["OK"]!!)
4     status.Level = (30..100).random().toByte()
5     status.VoltageData = (100..9999).random()
6     status.TemperatureData = (240..450).random()
7     var data = (byteArrayOf(status.Result, status.Level)
8                 + shortToBytes(status.VoltageData.toUShort())
9                 + shortToBytes(status.TemperatureData.toUShort()))
10    p.Data = data
11    p.CRC = CalcCRC(p)
12    data = EncodeToRFC1055(p)
13    outputStream!!.write(data)
14 }
```

Здесь минимальное возможное значение уровня заряда status.Level установлено на 30, так как при уровне заряда меньше 30 приложение производителя отказывает в обмене данными с имитатором и просит зарядить устройство.

Сгенерированные данные упаковываются в массив байтов и затем инкапсулируются в поле data пакета. Для пакета вычисляется контрольная сумма, далее пакет кодируется по RFC1055 и отправляется в поток вывода, т.е. клиентскому приложению.

```
1 private fun CalcCRC(p: Packet): Byte {
2     var crc = (2 + p.Data.size).toByte()
3     crc = (crc + p.ProtocolVersion).toByte()
4     crc = (crc + p.Command).toByte()
5     if (p.Data != null)
6         for (i in 0..p.Data.size) crc = (crc + p.Data[i]).toByte();
7     return crc;
8 }
```



```
1 private fun EncodeToRFC1055(p: Packet): ByteArray {
2     var buffer = ByteArray(3 + (p.Data.size)){0.toByte()}
3     buffer[0] = p.ProtocolVersion
4     buffer[1] = p.Command
5     if (p.Data != null) p.Data.copyInto(buffer, 2)
6     buffer[buffer.size - 1] = p.CRC;
7     return RFC1055.encode(buffer);
8 }
```

7.3 Пользовательский интерфейс

На главном экране появилась кнопка "KP-2 по которой запускается сценарий работы приложения с Bluetooth Classic. Кнопка ведёт на экран настройки параметров работы, на нём можно установить количество пакетов, отправляемых в секунду. После настройки частоты можно устанавливать соединение, передавать и записывать данные, а также поделиться файлом с данными.

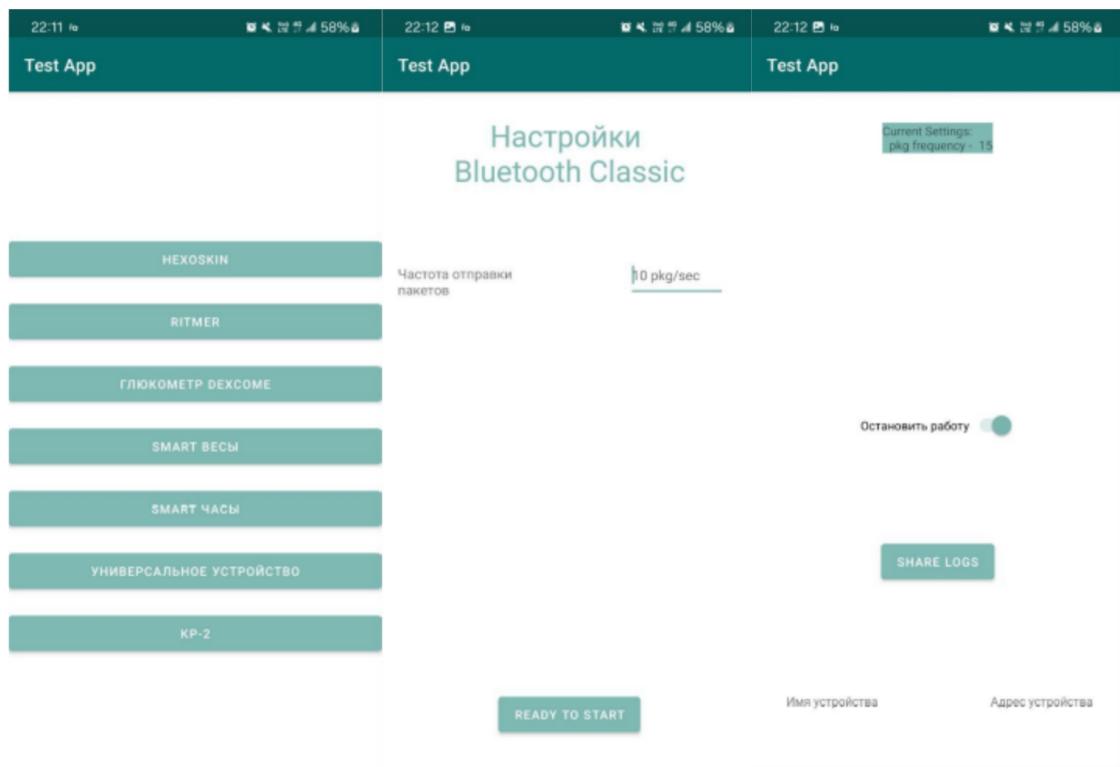


Рис. 5: Новый интерфейс

8 Экспериментальное исследование

8.1 Работа с приложением Аккордикс Эпп

Аккордикс Эпп - это клиентское приложение производителя системы кардиореабилитации «Аккордикс», компании «Нейрософт». Для проверки корректности работы разрабатываемого имитатора было необходимо подключиться к данному клиентскому приложению и отработать передачу данных ЭКГ.

Анализ фрагментов исходного кода приложения, предоставленных компанией Нейрософт показал, что для того, чтобы приложение увидело имитатор при поиске устройств Bluetooth поблизости, его имя должно удовлетворять определённому шаблону: включать строку "HS ECG", а далее, через пробел, должен быть указан серийный номер с префиксом "SN". Так, имя имитатора было изменено на "HS ECG SN000001".

После изменения имени устройства получилось успешно установить соединение между имитатором и клиентским приложением и начать передачу пакетов. Сведения о состоянии устройства (команда ReadComponentsStatus) были успешно приняты и отображены в пользовательском интерфейсе, однако далее, во время диагностики системы, приложение Аккордикс Эпп выдавало ошибку.

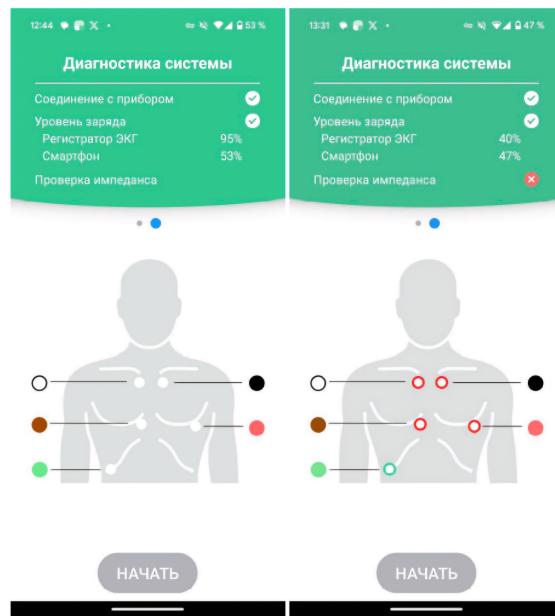


Рис. 6: Ошибки при подключении

Анализ информационных сообщений, выводимых при отладке имитатора, позволил понять, что клиентское приложение запрашивает у имитатора те данные, которые он генерировать не способен, а именно, требует ответа на команды FlashRead и ImpedanceMode. (В начале работы над имитатором было принято ограничиться только списком команд, описанным в разделе 5.2, так как именно эти команды необходимы для регистрации ЭКГ в рамках работы над сервисом сбора и обработки медицинской телеметрии). Было принято решение прочитать соответствующие данные с реального прибора КР-2 и выдавать их статично на запросы Аккордикс Эпп.

Так были реализованы ответы на команды FlashRead и ImpedanceMode. FlashRead - команда с кодом 7, запрашивает данные калибровки импеданса. ImpedanceMode - команда с кодом 3, запрашивает данные импеданса. Импеданс - это комплекское значение сопротивления между электродами кардиографа, представляет собой 8 пар значений - вещественных и мнимых частей. Получив данные импеданса, клиентское приложение проверяет, контактируют ли электроды с телом пациента, и в случае ошибок подсказывает, какие электроды нужно закрепить лучше.

На Рис. 6 первое состояние интерфейса представляет собой ситуацию, когда нет корректного ответа ни на одну из выше описанных команд, Аккордикс Эпп получило ответ только на команду ReadComponentsStatus и отобразило информацию о заряде КР-2 на экране. Второе состояние интерфейса отображает ситуацию, когда приложение получила корректные ответы на ReadComponentsStatus, FlashRead, однако ответ на ImpedanceMode говорил о том, что только один электрод из пяти был закреплён правильно.

После подбора удовлетворительного ответа на ImpedanceMode (считывание с прибора при замкнутых через кожу пациента электродах), имитатору удалось успешно пройти диагностику и начать передавать данные ЭКГ. Так как в настоящий момент в имитаторе реализована только генерация данных ЭКГ случайным образом, график отображал только прямую линию (возможно, в приложении реализована фильтрация неправдоподобных данных ЭКГ). Этот факт не является существенным недостатком, так как основной сценарий применения имитатора - исследование потерь при передаче данных на разрабатываемое на факультете ВМК приложение сбора данных медицинской телеметрии, здесь конкретная форма ЭКГ не важна.

Успешное подключение к приложению производителя указывает на то, что задача

по имитации КР-2 в части обмена пакетами посредством Bluetooth Classic выполнена.

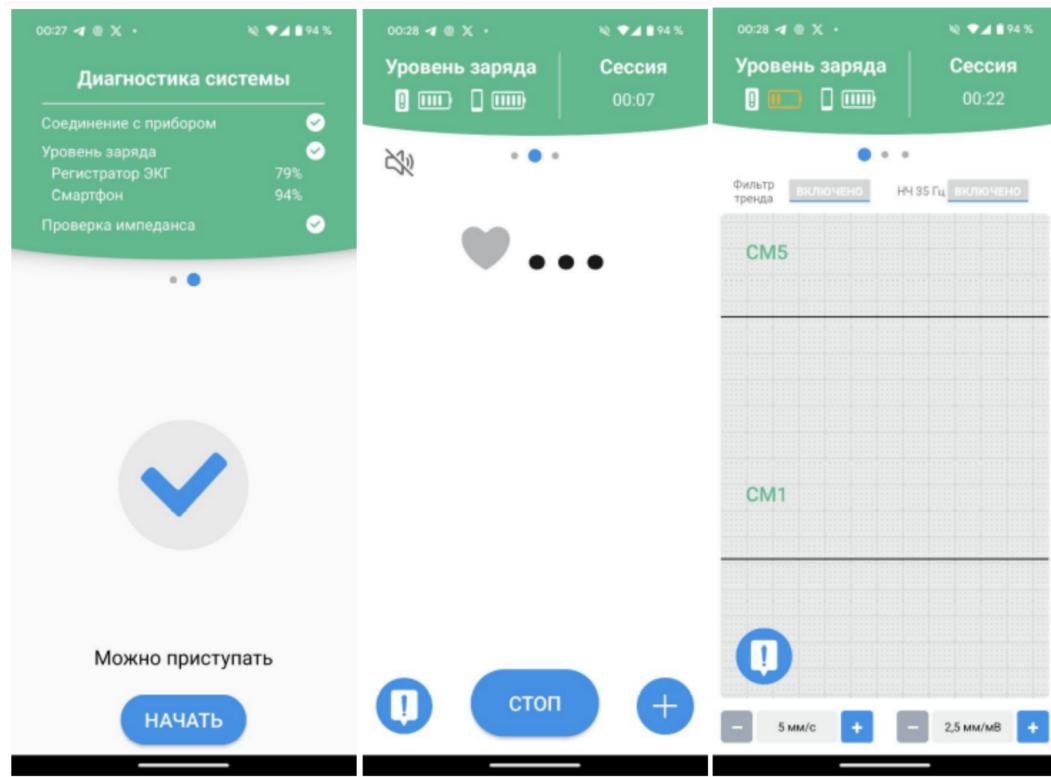


Рис. 7: Успешное подключение к Аккордикс Эпп

8.2 Оценка работоспособности клиентской части сервиса сбора и обработки медицинской телеметрии

Разработанный в ходе данной выпускной квалификационной работы имитатор КР-2 был использован для оценки работоспособности клиентской части сервиса сбора и обработки медицинской телеметрии, разрабатываемый в этом году студенткой третьего курса Бурдюговой Марией.

Клиентское приложение должно управлять кардиографом КР-2 посредством описанных в разделе 5.2 команд, принимать данные ЭКГ, отображать обработанные данные на экране в виде графика и отправлять их в базу данных на сервере.

В ходе исследования было проведено 18 сессий по обмену данными между клиентским приложением и имитатором с двумя параметрами: частота отправки пакетов настраивалась со стороны имитатора и принимала значения в 1, 5, 10, 20, 50 и 100

пакетов в секунду, и количество отсчетов ЭКГ настраивалось со стороны клиентского приложения и принимало значения в 25, 50 и 100 отсчетов в одном пакете (этот параметр передаётся кардиографу вместе с командой MonitoringMode, как было описано в разделе 5.2).

Здесь стоит отметить, что у кардиографа КР-2 частота отправки пакетов с данными ЭКГ, согласно документации, равна 10 пакетам в секунду.

В результате каждой сессии были получены по 2 текстовых файла: со стороны имитатора список отправленных данных с временем отправки каждого пакета и со стороны клиента список полученных данных с временем получения каждого пакета. Файлы состоят из строк с временем и значениями отсчетов для двух каналов.

С помощью скрипта, написанного на языке Python, были обработаны все 18 пар текстовых файлов, оценены потери данных и задержки при передаче данных. Так как данные были сгенерированы случайным образом, поиск потерь данных происходил просто по совпадению передаваемых байтов.

Для каждой сессии был получен файл с результатами, где для каждого потерянного отсчета ЭКГ был указан номер строки в файле от имитатора и значение. По номерам строк и общему количеству потерянных данных был сделан вывод о том, что при передаче были потеряны пакеты целиком: например при передаче пакета, содержащем 50 отсчетов, были потеряны данные с номерами строк от 4451 до 4550, то есть в этом промежутке потеряны 2 пакета, а также общее количество потерянных строк всегда было кратно количеству отсчетов в одном пакете.

Вывод скрипта в формате "частота отправки, количество отсчетов - количество совпадающих отсчетов, количество потерянных отсчетов":

1, 25 - 1325, 0	5, 25 - 5175, 125	10, 25 - 6100, 450
20, 25 - 12375, 850	50, 25 - 17575, 1125	100, 25 - 14250, 1475
1, 50 - 2700, 0	5, 50 - 12050, 150	10, 50 - 15650, 650
20, 50 - 22700, 1250	50, 50 - 20200, 1900	100, 50 - 24950, 1300
1, 100 - 4800, 0	5, 100 - 17600, 300	10, 100 - 18100, 600
20, 100 - 25900, 1000	50, 100 - 18900, 1000	100, 100 - 18700, 600

Величина задержек в данной работе не рассматривается, так как проведённые эксперименты не показали зависимости величины задержки от размера пакета и частоты отправки пакетов. Вне зависимости от этих параметров значение задержки чуть меньше одной секунды.

По полученным результатам была построена теплограмма (Рис. 8), отображающая процент потерянных данных в зависимости от частоты отправки пакетов и размера пакетов.

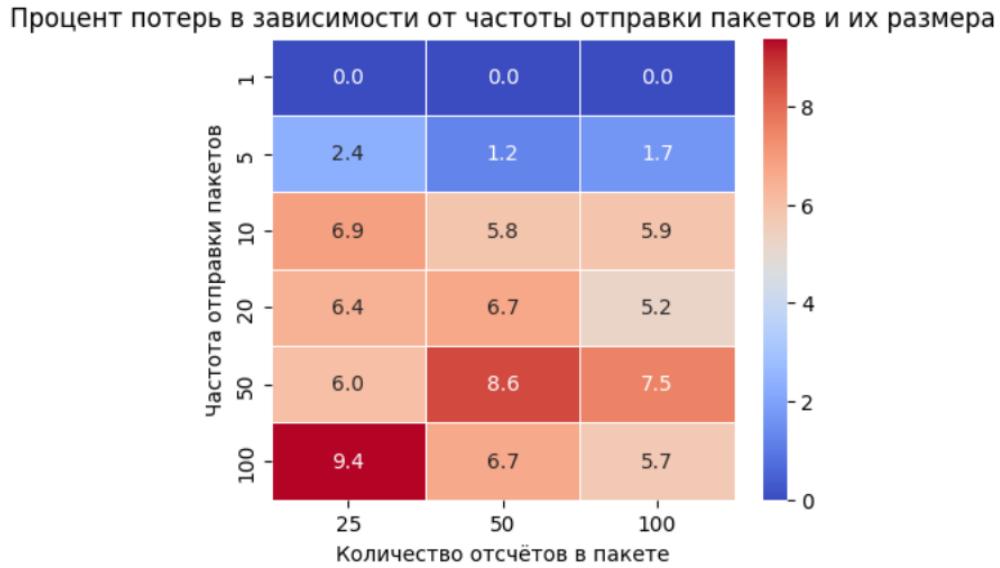


Рис. 8: Результат оценки работоспособности клиентской части сервиса

Исходя из полученных результатов и учитывая, что реальное устройство передаёт данные с частотой 10 пакетов в секунду, можно сделать вывод о том, что клиентское приложение нуждается в доработке и оптимизации, так как потери данных наблюдаются даже при меньшей частоте отправки пакетов, чем у реального КР-2.

Помимо этого Протокол обмена данными системы кардиореабилитации «Аккордикс» предполагает передачу номера отсчёта в пакете с данными ЭКГ для детектирования потерь пакетов со стороны клиента, однако на момент проведения экспериментов, данная функциональность реализована не была. Хотя экспериментальное исследование показало, что это необходимо.

9 Заключение

В ходе работы было доработано созданное ранее автором Android приложение, которое имитирует устройства интернета вещей посредством передачи данных по Bluetooth Low Energy, проведён анализ предоставленных фрагментов исходного кода для работы с имитируемым кардиографом, разработан пользовательский интерфейс, реализована работа с Bluetooth Classic, генерация данных, подобных данным кардиографа кр-2, их передача по Bluetooth Classic и запись в файл журнала.

Успешное подключение к клиентскому приложению производителя устройства компании «Нейрософт» позволяет Сделать вывод о корректной работе разработанного имитатора кардиографа КР-2.

Было проведено исследование доли потерь пакетов при передаче данных по Bluetooth в зависимости от частоты отправки данных. Исследование показало, что клиентская часть сервиса сбора и обработки медицинской телеметрии, разрабатываемого на кафедре АСВК факультета ВМК, нуждается в серьёзных доработках.

Разработанное приложение может упростить развитие сервиса сбора и обработки медицинской телеметрии и других сервисов Интернета вещей, позволяя разработчикам проводить тестирование их работоспособности без использования реальных устройств.

Список литературы

- [1] *B., Фролов А.* Разработка приложения сбора данных с нескольких устройств интернета вещей, подключенных одновременно, на примере медицинской телеметрии / Фролов А. Б. — 2023. — https://github.com/IoMT-LVK/papers/blob/main/client/4.%20Frolov_Diploma_Paper.pdf.
- [2] *И., Князев Е.* Разработка серверной части приложения интернета вещей на основе принципов передачи репрезентативного состояния / Князев Е. И. — 2023. — https://github.com/IoMT-LVK/papers/blob/main/server/4.%20Knyazev_Diploma_Paper.pdf.
- [3] *С., Голубкова М.* Разработка Android приложения, имитирующего датчики интернета вещей / Голубкова М. С. — 2023. — https://github.com/IoMT-LVK/papers/blob/main/client/5.Golubkova_Course_Paper.pdf.
- [4] *Hexoskin.* Smart Shirts. <https://www.hexoskin.com/>.
- [5] *Ritmer.* Монитор сердечной активности. <https://www.ritmer.ru/>.
- [6] *Dexcom.* Continuous Glucose Monitoring. <https://www.dexcom.com/global>.
- [7] *Picooc.* Умные весы. <https://picooc.ru/>.
- [8] *Mi.* Весы Mi Body Composition Scale. <https://ru.buy.mi.com/ru/item/3194100001>.
- [9] *Xiaomi.* Mi Smart Band. <https://www.mi.com/ru/product/mi-smart-band-6/>.
- [10] *B., Фролов А.* Развитие клиентской части сервиса сбора и обработки медицинской телеметрии / Фролов А. Б. — 2022. — https://github.com/IoMT-LVK/papers/blob/main/client/3.%20Frolov_Couse_Paper.pdf.
- [11] *Developers, Android.* Create dynamic lists with RecyclerView. <https://developer.android.com/develop/ui/views/layout/recyclerview>.
- [12] *Help, Kotlin.* Coroutines guide: Kotlin. <https://kotlinlang.org/docs/coroutines-guide.html>.

- [13] Varma N. Cygankiewicz I., Turakhia M. Российский кардиологический журнал. Контроль аритмий с помощью технологий мобильного здравоохранения: цифровые медицинские технологии для специалистов по сердечному ритму. Консенсус экспертов 2021. / Turakhia M. Российский кардиологический журнал. Varma N., Cygankiewicz I. — 2021. — <https://doi.org/10.15829/1560-4071-2021-4420>.
- [14] Medium. Bluetooth LE packet capture on Android. — 2020. <https://medium.com/propeller-health-tech-blog/bluetooth-le-packet-capture-on-android-a2109439b2a1>.
- [15] Bits, Novel. BLE Sniffer Basics + Comparison Guide. — 2023. <https://novelbits.io/bluetooth-low-energy-ble-sniffer-tutorial/>.
- [16] Website, Bluetooth® Technology. Bluetooth® Technology for Linux Developers. — 2023. <https://www.bluetooth.com/bluetooth-resources/bluetooth-for-linux/>.
- [17] Semiconductor, Nordic. nRF Connect for Mobile. <https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-mobile>.
- [18] «Нейрософт», Компания. Система кардиореабилитации «Аккордикс». <https://neurosoft.com/ru/catalog/ecg/2035#overview>.
- [19] ННГУ, Лаборатория физических основ и технологий беспроводной связи радиофизического факультета. курс «Основы проектирования беспроводных сетей». <https://wl.unn.ru/study/?page=5>.
- [20] (SIG), Bluetooth Special Interest Group. "Bluetooth for Developers." Version 5.2. <https://www.bluetooth.com/>.
- [21] MathWorks. Documentation. <https://www.mathworks.com/help/bluetooth/ug/bluetooth-protocol-stack.html>.
- [22] Developers, Android. listenUsingRfcommWithServiceRecord. <https://developer.android.com/reference/android/bluetooth/BluetoothAdapter>.
- [23] C., Голубкова М. Репозиторий с полным кодом описываемого приложения. <https://gitlab.asvk.cs.msu.ru/golubkova.m.s/testapp>.