

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

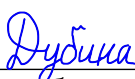
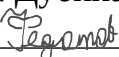
**Московский институт электроники и математики НИУ ВШЭ  
им. А.Н. Тихонова**

**ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС ОТСЛЕЖИВАНИЯ И  
АНАЛИЗА ДАННЫХ ДВИЖЕНИЙ ЧЕЛОВЕКА**

Выпускная квалификационная работа по направлениям подготовки  
11.04.02 «Инфокоммуникационные технологии и системы связи» и  
01.04.04 «Прикладная математика»


студентов образовательных программ  
«Интернет вещей и киберфизические системы» и  
«Системный анализ и математические технологии»


Студенты группы МСМТ212:

  
\_\_\_\_\_  
Д.О. Дубина  
  
\_\_\_\_\_  
Г.А. Федотов

Студент группы МИВ212:

  
\_\_\_\_\_  
К.Г. Кожакин

Руководитель  
Профессор департамента бизнес-  
информатики  
  
\_\_\_\_\_  
В.Ю. Попов

Рецензент  
звание, должность  
  
\_\_\_\_\_  
Профессор ДПИ ФКН  
Е.М.Гринкруг  
И.О.Фамилия

## Реферат

По мере того, как важность понимания пределов человеческого тела становилась все более очевидной, росла потребность отслеживать и документировать эти границы. Со временем методы, используемые для изучения движений человека, продвинулись от простых визуальных наблюдений и ручного подсчета до сложных аналоговых решений с использованием проводов и специализированного оборудования, которые были доступны только профессиональным спортсменам и ученым. В последние годы цифровые беспроводные решения стали более доступными, а точность их измерений возросла до такой степени, что они сравнимы с медицинскими приборами. Спортивная индустрия в настоящее время является основной движущей силой инноваций в отслеживании движений человека, ярким примером чего является популярность фитнес-браслетов. Однако эти устройства часто имеют закрытые программные системы, которые ограничивают их функциональность, кастомизацию и возможность стороннего сбора данных. В этой статье представлен прототип программно-аппаратного комплекса, который может отслеживать данные о двигательной активности человека, классифицировать стадии сна и определять типы физической активности. В документе приводится подробное описание конструкции и реализации системы. В нем описывается, как работает система, и объясняется каждый этап процесса разработки.

Работа содержит 88 страниц, 3 главы, 33 рисунка, 36 источников.

**Ключевые слова** — Bluetooth, гироскоп, акселерометр, машинное обучение, Android, STM32WB55.

## **Abstract**

As the importance of understanding the limits of the human body has become increasingly clear, there has been a growing need to track and document these boundaries. Over time, the techniques used to study human movement have advanced from simple visual observations and manual counting to complex analog solutions involving wires and specialized equipment that were only accessible to professional athletes and scientists. In recent years, digital wireless solutions have become more affordable and their measurement accuracy has increased to the point where they are comparable to medical devices. The sports industry is currently the main driver of innovation in tracking human movements, with the popularity of fitness bracelets being a notable example. However, these devices often have closed software systems that limit their functionality, customization, and third-party data collection capabilities. This paper presents a prototype of a software and hardware system that can track human motor activity data, classify sleep stages, and identify types of physical activity. The paper provides a detailed description of the system's design and implementation. It outlines how the system operates and explains each stage of the development process.

The work contains 88 pages, 3 chapters, 33 figures, 36 sources.

**Keywords** — Bluetooth, gyroscope, accelerometer, machine learning, Android, STM32WB55.

## Основные определения, термины и сокращения

**Bluetooth** - производственная спецификация беспроводных сетей, обеспечивает обмен информацией между разнообразными устройствами.

**USB** - последовательный интерфейс для подключения периферийных устройств к другим устройствам.

**ARM** - микропроцессорная архитектура с сокращенным набором команд (RISC), разрабатываемая компанией ARM.

**EEPROM** - это тип энергонезависимой памяти, который используется в электронных устройствах для хранения данных, которые могут быть прочитаны, удалены или записаны, а также который можно перепрограммировать и стереть.

**I2C** - последовательная асимметричная шина для связи внутри электронных устройств. Использует две двунаправленные линии связи и применяется для взаимодействия компонентов с микроконтроллерами.

**Набор данных** – это структурированные и обработанные данные представленные в табличном виде. Как правило, строки таблицы являются объектами, а столбцы признаками.

**Признак (фича, feature)** – это переменная, описывающая свойство или характеристику объекта.

**Модель (model)** представляет собой результат, который получается при обучении машинного алгоритма на данных. Обученная модель может быть применена для прогнозов и принятия решений по данным, которые она раньше не встречала.

## Оглавление

<b>Реферат</b>	<b>0</b>
<b>Abstract</b>	<b>1</b>
<b>Основные определения, термины и сокращения</b>	<b>2</b>
<b>Оглавление</b>	<b>3</b>
<b>Введение</b>	<b>6</b>
<b>Глава 1. Описание принципов работы и обзор аналогов</b>	<b>9</b>
1.1. Сон	9
1.2. Методы исследования сна	10
1.3. Существующие решения	10
1.4. Обоснование разработки нового программно-аппаратного комплекса	11
1.5. Выводы по главе	12
<b>Глава 2. Используемые методы, модели и инструменты</b>	<b>13</b>
<b>2.1. Аппаратная часть</b>	<b>13</b>
2.1.1. Основной модуль STM32WB55CG	13
2.1.2. Модуль передачи данных по беспроводной сети	13
2.1.3. Модуль питания	14
2.1.3.1. LD3985M33R	15
2.1.3.2. L6924D013	15
2.1.4. Модуль хранения данных	15
2.1.4.1. L6924D013	15
2.1.5. Модуль считывания данных	15
2.1.5.1. LSM6DSR	16
2.1.6. Модуль отладки и индикации	16
2.1.7. Модуль атмосферного давления	16
2.1.7.1. LPS22HBTR	16
2.1.8. Модуль NFC	17
2.1.8.1. ST25DV64K-JFR6D3	17
2.2. Программная часть	18
2.2.1. X-CUBE-EEPRMA1	18
2.2.2. X-CUBE-MEMS1	18
2.2.3. STM32_WPAN	19

2.2.4. HAL	19
2.2.5. Протокол Bluetooth	20
2.2.6. ПО ядра Bluetooth	22
2.2.7. Android Studio	22
2.2.8. Kotlin	23
2.2.9. Стандартная библиотека Android	23
2.2.10. MPAndroidChart	25
2.2.11. Midjourney	25
2.2.12. Выбор инструментария для разработки приложения компаньона	26
2.2.13. Стандартная библиотека Java	29
<b>2.3. Машинное обучение</b>	<b>31</b>
2.3.1. Язык программирования Python	31
2.3.2. Google Colab	31
2.3.3. NumPy	31
2.3.4. Pandas	32
2.3.5. Matplotlib	32
2.3.6. Sklearn	32
2.3.7. PyTorch	32
2.3.8. Optuna	33
2.3.9. Catboost	34
2.3.10. Полносвязная нейронная сеть	34
2.3.11. Оптимизатор Adam	35
2.3.12. CrossEntropyLoss	39
2.3.13. Матрица ошибок и метрика Accuracy	40
2.3.14. Кросс-валидация	40
2.4. Выводы по главе	41
<b>Глава 3. Разработка и архитектура приложения</b>	<b>42</b>
<b>3.1. Аппаратная часть устройства</b>	<b>42</b>
3.1.1. Общие сведения	42
3.1.2. Разработка аппаратной части	43
3.1.3. Архитектура аппаратной части	49
3.1.4. Средства и инструменты разработки аппаратной части	50
<b>3.2. Программная часть устройства</b>	<b>51</b>
3.2.1. Общие сведения	51
3.2.2. Средства и инструменты разработки программной части	52

3.2.3. Реализация профиля GATT	53
3.3. Приложение компаньон	54
3.3.1. Интерфейс приложения компаньона	54
3.3.2. Реализация работы с Bluetooth	56
3.3.3. Реализация работы с данными	59
<b>3.4. Машинное обучение</b>	<b>61</b>
3.4.1. Создание набора данных	61
3.4.2. Анализ набора данных	63
3.4.3. Создание модели CatboostClassifier и подбор ее гиперпараметров с помощью optuna	68
3.4.4. Создание полносвязной нейронной сети	73
3.4.5. Создание полносвязной нейронной сети и подбор ее гиперпараметров с помощью optuna	75
3.4.6. Внедрение модели в мобильное приложение	78
3.5. Примеры использования и тестирования	79
3.6. Репозиторий	81
3.7. Описание структуры проекта	81
3.8. Выводы по главе	82
<b>Заключение</b>	<b>83</b>
<b>Список литературы</b>	<b>85</b>
ПРИЛОЖЕНИЕ А	102
ПРИЛОЖЕНИЕ Б	108
ПРИЛОЖЕНИЕ В	120
ПРИЛОЖЕНИЕ Г	137

## **Введение**

Отслеживание физической активности человека, поиск пределов возможностей и их документирование были актуальны всегда, но начать документировать по-умному, человечество смогло только в прошлом столетии. Еще до восхода эпохи гаджетов появились первые носимые электронные устройства для спортсменов: мониторы сердечного ритма и велокомпьютеры. Их задачей было собирать данные об активности спортсмена во время тренировки для последующего анализа. И только спустя десятилетия на мировой рынок из профессионального спорта пришли потребительские устройства.

Сейчас рынок фитнес-браслетов изобилует моделями способными, не только отследить активность пользователя, но и оценить качество его сна, измерить давление, пульс на уровне приближенном к профессиональным медицинским приборам. Некоторые устройства даже способны заменить смартфон.

Однако подавляющее большинство компаний разрабатывают закрытые проекты, модификация которых для сбора данных своими силами невозможна, или нарушает закон. Если перед исследователем стоит задача собрать данные для исследования данных активности человека, то зачастую его единственное решение искать проект с открытым исходным кодом, либо реализовывать подобное устройство самому.

В этом документе описывается проект программно-аппаратного комплекса, который позволяет записывать данные двигательной активности посредством гироскопа и акселерометра и передавать их на



сопутствующее устройство по беспроводной сети, для дальнейшей обработки, посредством моделей машинного обучения.

Цель выпускной квалификационной работы – разработка проекта программно-аппаратного комплекса, считывающего двигательную активность человека, передающего на приложение-компаньон по беспроводной сети эти данные. А также реализовать две модели машинного обучения, для классификации двигательной активности и этапов сна.

Цель данного документа – описать методы, технологии и инструменты, которые были использованы в ходе работы над продуктом, а также описать результат и принципы его работы

Для достижения цели выпускной квалификационной работы были поставлены следующие задачи:

- Изучить аналоги;
- Изучить технологии реализации;
- Выявить основания для разработки;
- Разработать техническое задание и функциональные требования;
- Спроектировать и реализовать устройство;
- Спроектировать и реализовать приложение-компаньон;
- Собрать данные для обучения моделей машинного обучения;
- Обучить модели;
- Внедрить модели в приложение компаньон;
- Отладить программный код;
- Разработать техническую документацию.

Данный документ состоит из 3 глав:

В первой главе рассказывается об анализе существующих решений

Во второй главе были рассмотрены инструменты, которые использовались для разработки комплекса, а также некоторая теоретическая информация.

В третьей главе были рассмотрены практические подходы, которые использовались для разработки программно-аппаратного комплекса.

# **Глава 1. Описание принципов работы и обзор аналогов**

## **1.1. Сон**

Сон [1] является уникальным функциональным состоянием мозга и тела человека и животных, которое определяется генетически и характеризуется специфическими отличительными чертами в активности центральной нервной системы и соматической сферы, такими как ингибирование активного взаимодействия организма с окружающей средой и не полное прекращение сознательной психической деятельности. Физиологический сон различается от состояний, таких как кома, сопор, наркоз, спячка и гипноз, тем, что он наступает из-за внутренних, а не из-за внешних факторов, и при этом обеспечивается возможность пробуждения.

Существует две фазы сна: медленный (ФМС) и быстрый (ФБС). ФМС характеризуется расслаблением мышц, замедлением пульса и ровным дыханием, и включает в себя четыре стадии: первую, вторую, третью и дельта-сон. Общая длительность медленного сна у среднестатистического человека составляет 80%, при этом первая стадия длится 10%, вторая - 50%, дельта-сон - 20%. ФБС, в свою очередь, характеризуется быстрым движением глаз под закрытыми веками, учащенным дыханием и возможностью произношения речи во сне, и является стадией сновидений. Средняя продолжительность фазы быстрого сна составляет 20%.

Сон человека представляет собой периодические циклы продолжительностью от 1,5 до 2 часов, которые повторяются 3-5 раз за ночь и включают в себя фазы ФМС и ФБС. Несмотря на общее снижение двигательной активности, наблюдается многообразие движений человека

во время сна, включая как мелкие судороги конечностей, так и более крупные перемены положения тела. Эти движения характерны для всех стадий сна и часто предшествуют смене фаз.

## 1.2. Методы исследования сна

Основным средством изучения сна у человека является полисомнография (ПСГ) [2], метод регистрации жизнедеятельности организма в течение периода сна. Для этой цели измеряются несколько показателей, среди которых важнейшими являются электроэнцефалограмма (ЭЭГ), регистрирующая электрическую активность мозга, электроокулограмма (ЭОГ), фиксирующая движения глаз, и электромиограмма (ЭМГ), отображающая напряженность мышц. Для точного определения характеристик сна ПСГ также измеряет поток выдыхаемого воздуха, движения грудной клетки и брюшной полости при дыхании, шум дыхания, уровень насыщения крови кислородом, положение тела в постели и частоту сердечных сокращений.

## 1.3. Существующие решения

Основными конкурентами нашего программно-аппаратного комплекса являются уже существующие фитнес браслеты: Samsung Galaxy Fit, Honor Band 5, Xiaomi Mi Band 4 и другие [3, 4]. В основном модели имеют следующие возможности:

- Измерение пульса и шагомеры, что позволяет точно определить количество прошедших шагов и сожженных калорий за определенный промежуток времени.
- GPS-мониторинг, который позволяет фиксировать маршрут тренировки и общее расстояние.

- Уведомления о звонках, сообщениях и других уведомлениях, которые приходят на подключенный смартфон.
- Возможность контроля за качеством и продолжительностью сна.
- Использование в качестве будильника, который легко настраивается и позволяет вставать в момент, когда вы наиболее отдохнули.
- Возможность засечь время тренировки и определять время, затраченное на прохождение определенной дистанции.

Samsung Galaxy Fit: работает с приложением Samsung Health, предоставляющим доступ к большим объемам данных по тренировкам, сна и питанию. Имеет стильный дизайн и большой сенсорный дисплей.

Honor Band 5: имеет высокую точность измерения пульса и GPS-трекинг. Имеет возможность отображать информацию о силе и продолжительности тренировки, а также дает дополнительные рекомендации об упражнениях.

Xiaomi Mi Band: легкий и простой в использовании. Имеет очень долгий срок службы батареи и множество цветовых вариантов для выбора. Также мониторит сон и имеет удобную управляющую панель для отображения уведомлений.

## **1.4. Обоснование разработки нового программно-аппаратного комплекса**

Обоснованием разработки нового программно-аппаратного комплекса является малое количество open source решений в данной области, которые предоставляют возможность снимать.

## **1.5. Выводы по главе**

В первой главе была рассмотрена предметная область данной работы и существующие аналоги. Также было приведено обоснование разработки нового комплекса.

## **Глава 2. Используемые методы, модели и инструменты**

### **2.1. Аппаратная часть**

Устройство состоит из двух частей: аппаратной части и программного обеспечения к нему. Аппаратная часть состоит из абстрактных модулей, состоящих из компонентов, соединенных в соответствии с рекомендациями в их документации и разделенных по функционалу.

#### **2.1.1. Основной модуль STM32WB55CG**

Серия микроконтроллеров STM32WB55, и версия STM32WB55CG [5], в частности, выступает в роли аппаратной основы для беспроводных устройств интернета вещей. Микроконтроллер содержит два ядра. Первое ядро, которое необходимо для исполнения основной программы — это производительное ядро ARM Cortex-M4, работающее на частоте до 32 МГц. Второе ядро, которое реализует работу с одним из поддерживаемых беспроводных протоколов - менее скоростное ядро ARM Cortex-M0+ работающее на частоте до 32 МГц. Также, в микроконтроллере реализован набор периферии, доступ к которой регулируется благодаря аппаратным семафорам. Линейка микроконтроллеров STM32WB55 является самой актуальной среди продуктов компании STMicroelectronics, и была выбрана как основа устройства.

#### **2.1.2. Модуль передачи данных по беспроводной сети**

Для реализации приложения, удовлетворяющего потребностям и возможностям протокола Bluetooth [6] последней версии, ядро

микроконтроллера серии STM32WB55 CPU2, посредством API предоставляет:

- Сконфигурировать публичный адрес устройства BLE;
- Настроить мощность передатчика;
- Настроить профиль GATT;
- Сконфигурировать GATT в зависимости от выбранной роли устройства;
- В случае использования функционала безопасности позволяет задать требования к аутентификации;
- В случае работы как GATT-сервер, позволяет сконфигурировать набор предоставляемых сервисов.

Для реализации сопутствующих потребностей Bluetooth, с аппаратной точки зрения, используются компоненты MLPF-WB55-01E3, выступающий в роли фильтра частот и KN3216-A55, который является антенной. Оба компонента подключены в соответствии с документацией серии микроконтроллеров и собственно документацией.

### **2.1.3. Модуль питания**

Для обеспечения устройства электричеством, применены Li-Ion аккумулятор с защитой от излишних разрядки и зарядки, и два компонента необходимых для управления питанием. LD3985M33R используется преобразования напряжения аккумулятора в необходимые для микроконтроллера 3.3 Вольта. L6924D используется для реализации алгоритмов зарядки разрядки аккумулятора.



### **2.1.3.1. LD3985M33R**

LD3985M33R - линейный стабилизатор с низким падением напряжения положительной полярности, 3.3 В, 0.15 А. Используется для понижения напряжения.

### **2.1.3.2. L6924D013**

L6924D013 - контроллер заряда батарей Li-Ion/Li-Pol, предоставляет возможность тонко настроить все фазы зарядки батареи, предоставляет два источника прерывания для индикации статуса зарядки.

## **2.1.4. Модуль хранения данных**

Для хранения данных в устройстве, независимого от доступа к электричеству в устройстве использована схема EEPROM памяти M24512-DFMC6TG, управление и обмен данными с которой осуществляется посредством I2C протокола.

### **2.1.4.1. L6924D013**

EEPROM память с I2C интерфейсом, объёмом 512 кбит, предоставляет 512 128 байтовых страниц для записи данных и одну идентификационную дополнительную 128 байтовую страницу. Имеет вход для управления блокировкой записи в память.

## **2.1.5. Модуль считывания данных**

Для считывания необходимых данных в устройстве используется схема LSM6DSR, посредством которой устройство, через протокол I2C получает данные гироскопа и акселерометра, для последующей обработки.

### **2.1.5.1. LSM6DSR**

LSM6DSR - инерциальный MEMS-датчик, который включает цифровой 3D-акселерометр и 3D-гироскоп. Датчик измеряет ускорения в диапазонах  $\pm 2 / \pm 4 / \pm 8 / \pm 16$  g и угловую скорость до  $\pm 125 / \pm 250 / \pm 500 / \pm 1000 / \pm 2000 / \pm 4000$  dps. Позволяет подключать к себе дополнительные датчики по I2C шине, единовременно собирать данные с датчиков гироскопа, акселерометра, температуры и внешних датчиков, и предоставлять I2C мастеру единовременно, цельным блоком данных.

### **2.1.6. Модуль отладки и индикации**

Для отладки и индикации в устройстве реализован светодиод и вибромотор. В рамках устройства заложены и другие модули, не используемые в реализации, но позволяющие расширить возможности устройства, путем модификации ПО.

### **2.1.7. Модуль атмосферного давления**

Для улучшения работы комплекса, и сбора дополнительных данных, в устройство включен датчик LPS22HBTR для измерения атмосферного давления, подключённый по I2C к датчику LSM6DSR, выступающим в роли Master и собирающего данные с него.

#### **2.1.7.1. LPS22HBTR**

LPS22HBTR, MEMS Датчик абсолютного давления с цифровым выход, 260-1260 Гпа. Передает данные по I2C.

## **2.1.8. Модуль NFC**

Для улучшения работы комплекса, увеличения способов обмена данных с приложением компаньоном, и для расширения будущих возможностей безопасного соединения устройств по Bluetooth, в устройство включен NFC чип ST25DV64K-JFR6D3, управляемый по I2C.

### **2.1.8.1. ST25DV64K-JFR6D3**

ST25DV64K-JFR6D3 представляет собой метку NFC RFID, предлагающую 64 КБ электрически стираемой программируемой памяти EEPROM. Имеет два интерфейса, последовательный канал I2C, RF-канал, который активируется, когда устройство действует как бесконтактная память, питаемая от принятой электромагнитной несущей волны.

## **2.2. Программная часть**

Для разработки программной части были использованы библиотеки поставляемые STMicroelectronics X-CUBE-EEPRMA1, X-CUBE-MEMS1 и STM32\_WPAN и HAL. X-CUBE-EEPRMA1 используется для работы с чипом памяти, X-CUBE-MEMS1 для работы с датчиком акселерометра и гироскопа, HAL используется для корректной работы устройства, а также реализует возможность работы с периферией. STM32\_WPAN предоставляет базовый функционал необходимый для настройки и взаимодействия с ядром Bluetooth микроконтроллера.

### **2.2.1. X-CUBE-EEPRMA1**

X-CUBE-EEPRMA1 [7] это программно-аппаратный пакет, содержит драйвер и набор функций для управления внешними энергонезависимыми модулями памяти (EEPROM) через шину I2C. Это позволяет программистам записывать и читать данные из внешних EEPROM без необходимости использования дополнительных компонентов или собственной низкоуровневой реализации.

### **2.2.2. X-CUBE-MEMS1**

X-CUBE-MEMS1 [8] - это программно-аппаратный пакет, содержит драйверы и библиотеки для работы с микро электромеханическими системами (MEMS), такими как акселерометры, гироскопы и магнитометры. Он предоставляет возможность легко и быстро обрабатывать данные из MEMS-датчиков и использовать их в своих приложениях. Внедрение подобных пакетов позволяет ускорить время разработки приложений и уменьшить затраты на разработку.

### 2.2.3. STM32\_WPAN

STM32\_WPAN [9] – фреймворк для разработки персональных беспроводных сетей по технологии Bluetooth. Включает в себя необходимые для сертифицированного стека BLE 5.0 компоненты:

- Сертифицированный стек BLE 5.0;
- Канальный уровень;
- Интерфейс хост-контроллер;
- L2CAP (Logical Link Control and Adaptation Protocol – протокол управления логическим подключением и адаптацией);
- ATT (Attribute Protocol) – протокол атрибутов;
- GAP (Generic Access Profile) – профиль общего доступа;
- GATT (Generic Attribute Profile) – профиль общих атрибутов.

Фреймворк предоставляет весь необходимый функционал [10], необходимый для реализации корректной работы Bluetooth на микроконтроллере.

### 2.2.4. HAL

HAL позволяет отказаться от работы с регистрами и является оберткой над операциями низкого уровня, что позволяет эффективней и быстрее разрабатывать ПО для микроконтроллеров. Для реализации проекта планируется использовать следующие модули этой библиотеки:

- HAL\_exti модуль для работы с прерываниями. Модуль нужен для работы с устройствами ввода и другими модулями HAL библиотеки, необходим для реализации условного параллельного выполнения заданий за счет того, что позволяет на аппаратном уровне управлять и проверять флаги периферии, не занимая ресурс микроконтроллера.

- HAL\_rtc модуль для работы с часами реального времени. Часы необходимы для полноценной работы протокола Bluetooth, перехода в спящий режим, а также для согласования длительности операций внешними часами. Модуль настраивает часы реального времени, инициализирует, предоставляет доступ к работе с часами другим модулям.
- HAL\_gpio модуль для работы с выходами микроконтроллера. Данный модуль нужен для конфигурирования выводов микроконтроллера, для последующего управления. Это позволит реализовывать индикацию и управление периферией.
- HAL\_hsem модуль для работы с семафорами микроконтроллера. Семафоры необходимы для взаимодействия двух ядер микроконтроллера, а именно для доступа к ресурсам. Использование семафоров позволяет работать ядрам независимо друг от друга, регламентируя доступ к общим ресурсам. Данный модуль необходим для корректной работы как Bluetooth, так и всего микроконтроллера в целом.
- HAL\_tim модуль для работы с таймерами микроконтроллера. Таймеры необходимы для совершения действий с заданным интервалом, не занимая ресурс микроконтроллера на время ожидания. Модуль производит настройку таймеров, их инициализацию и предоставляет функции для создания и управлениями таймерами.

### 2.2.5. Протокол Bluetooth

Протокол Bluetooth — это беспроводной стандарт связи между электронными устройствами. Он используется для передачи данных

между устройствами, такими как смартфоны, наушники, компьютеры и другие устройства. Для его работы необходимо сформировать GATT [11].

GATT (Generic Attribute Profile) - это профиль, который определяет, каким образом устройства могут использовать АТТ-атрибуты для обмена данными. АТТ (Attribute Protocol) - это протокол, использующийся в Bluetooth Low Energy (BLE) для передачи информации между устройствами. Он позволяет устройствам обмениваться данными в формате атрибутов — это некоторые значения, которые хранятся в устройстве и могут быть изменены либо прочитаны.

Устройство, реализующее GATT профиль, содержит в себе сервисы. Сервисы — это коллекции связанных характеристик, объединенных в отдельные уровни для удобства коммуникации между устройствами. Например, сервис может описывать датчик движения, который определяет текущее положение устройства. Характеристики — это особые атрибуты, которые описывают какую-то конкретную характеристику устройства, такую как его серийный номер, модель или версия программного обеспечения.

GATT профиль может реализовывать несколько сервисов, каждый из которых может иметь по несколько характеристик. Каждый профиль имеет свое название, каждый сервис и характеристика уникальный 128-битный UUID. Таким образом, можно удобно и эффективно организовать доступ к данным и к операциям взаимодействия с ними.

Допустимые разрешения на взаимодействия с характеристиками также настраиваются. Так можно выдавать следующие разрешения [12]:

- Разрешения доступа:
  - чтение;

- запись;
- чтение и запись.
- Разрешение аутентификации:
  - аутентификация требуется;
  - аутентификация не требуется.
- Разрешение авторизации:
  - авторизация требуется;
  - авторизация не требуется.

Дополнительно можно настраивать дескрипторы. Дескрипторы — это более дополнительные характеристики, которые определяют некоторые дополнительные свойства атрибутов. Например, дескриптор может указать масштабирование значения для определенной характеристики. Также он может описывать единицы измерения для определенной характеристики. Дескрипторы позволяют включить индикацию или нотификацию. Нотификация позволяет отправлять сообщения подключенному устройству и не требует подтверждения в получении со стороны клиента. Индикация позволяет делать тоже самое, но требует подтверждения.

## **2.2.6. ПО ядра Bluetooth**

Для работы Bluetooth, необходимо установить специальную прошивку [13] на второе ядро микроконтроллера. STMicroelectronics предлагает широкий выбор протоколов помимо Bluetooth и вариативные реализации ПО ядра, в зависимости от потребностей проекта.



### **2.2.7. Android Studio**

Android Studio - это интегрированная среда разработки (IDE) для создания приложений под операционную систему Android. Она предоставляет разработчикам все необходимые инструменты для создания и отладки приложений, включая поддержку языков программирования Kotlin и Java, графических редакторов для создания интерфейса пользователя, эмулятор Android-устройства и многие другие функции, упрощающие процесс разработки приложений для Android. Кроме того, Android Studio обладает различными инструментами отладки и тестирования приложения на устройствах с различной конфигурацией.

В целом, Android Studio является одним из самых распространенных и популярных средств разработки мобильных приложений для Android.

### **2.2.8. Kotlin**

Kotlin - это язык программирования, который разработан компанией JetBrains. Он предназначен для создания приложений для Android, взаимодействия с серверами и создания веб-приложений. Kotlin является статически типизированным языком, который работает в среде JVM и может взаимодействовать с Java. Он обладает рядом преимуществ перед Java, таких как более удобный и продуктивный синтаксис, меньше кода, более высокая безопасность, лямбда-выражения и многие другие. Kotlin также поддерживает функциональное программирование и объектно-ориентированную парадигмы. Поэтому, многие разработчики выбирают Kotlin в качестве языка программирования для своих проектов.

## 2.2.9. Стандартная библиотека Android

Стандартная библиотека Android предоставляет широкий спектр инструментов для разработки приложений на платформе Android. С помощью этой библиотеки можно создавать различные виды приложений, работающих с разными функциями системы, такими как:

- Графический интерфейс пользователя (GUI):
  - Создание различных макетов и интерфейсов с помощью компонентов View и ViewGroup;
  - Использование анимаций и переходов между экранами;
  - Работа с изображениями и графикой.
- Связь с внешними устройствами:
  - Взаимодействие с камерой и микрофоном;
  - Управление Bluetooth-устройствами и Wi-Fi;
  - Работа с GPS и другими датчиками на смартфоне.
- Работа с хранилищами данных:
  - Создание и управление базами данных SQLite;
  - Работа с файловой системой устройства;
  - Использование SharedPreferences для хранения настроек и других данных.
- Работа с сетью и интернетом:
  - Отправка и получение данных с сервера с помощью HTTP-запросов;
  - Работа с SOCKS-протоколом для обмена данными по сети;
  - Разработка приложений для социальных сетей с использованием API.

Кроме этих функций, стандартная библиотека Android содержит множество других возможностей для разработки приложений, таких как работа с событиями, создание сервисов и обработка ошибок. Зная все эти возможности, можно создавать высококачественные приложения для Android, адаптированные под различные устройства и типы экранов.

Данная библиотека была использована для работы с Bluetooth, а именно для запросов разрешений, поиска устройства и подключения к нему, сопряжения с ним и получения различных данных с устройства. Также с помощью этой библиотеки были реализованы различные Activities и Fragments.

### **2.2.10. MPAndroidChart**

MPAndroidChart [36] - это библиотека для отображения графиков и диаграмм на платформе Android. Она предоставляет различные типы графиков, такие как линейные, круговые, столбчатые, в виде свечной диаграммы и др. Также библиотека предоставляет множество опций настройки внешнего вида графиков, возможность добавления анимации и тултипов для интерактивного взаимодействия пользователя с графиком. MPAndroidChart является одной из самых популярных библиотек для визуализации данных на Android платформе и используется во многих приложениях. Данная библиотека использовалась для отрисовки графиков активности и сна.

### **2.2.11. Midjourney**

Midjourney - это нейросеть, разработанная компанией NVIDIA, для обучения машинного обучения. Она представляет собой глубокую

сверточную нейронную сеть, специально разработанную для обработки изображений. Она была создана для решения задачи классификации изображений и работает на основе архитектуры CNN (Convolutional Neural Networks, сверточные нейронные сети), которая позволяет эффективно и точно анализировать изображения. Midjourney используется в различных задачах компьютерного зрения, таких как распознавание объектов на изображении, определение настроения по фотографиям и обнаружение дефектов в изображениях. Она также широко применяется в различных индустриях, таких как здравоохранение, финансы и автомобильная промышленность. Использовалась для генерации изображений в приложении компаньоне.

### **2.2.12. Выбор инструментария для разработки приложения компаньона**

Функциональные требования

- Запрос необходимых разрешений для корректной работы приложения;
- Поиск подходящих устройств по Bluetooth;
- Подключение к устройству по Bluetooth;
- Подписка на изменение необходимых характеристик устройства;
- Экспорт собранных данных в формате csv;
- Отображение обработанных данных в виде графиков и численных значений;
- Сохранение данных в кеше приложения.

При разработке приложения-компаньона стояла задача реализовать мобильное приложение на базе Android. Существует несколько способов разработки для этой платформы:

- Разработка нативных приложений на Java/Kotlin. Для этого необходимо установить Android Studio, которая содержит в себе все необходимые инструменты для разработки, такие как Android SDK и эмуляторы устройств. В качестве IDE также можно использовать IntelliJ IDEA.
- Разработка гибридных приложений на различных фреймворках, таких как Flutter, React Native и других. Благодаря ним приложение может быть кроссплатформенным без написания специфического кода для различных систем.
- Разработка на Android NDK, с применением различных языков программирования, например: Python, C++, JavaScript и другие.
- Разработка веб-приложений на платформе Chrome, используя Android WebView. WebView позволяет запускать веб-страницы внутри приложения, что позволяет создавать веб-приложения с дополнительной функциональностью, такой как доступ к железу устройства.

Все эти способы имеют свои преимущества и недостатки, и выбор зависит от требований к приложению, бюджета и опыта разработчика.

Проанализировав эти параметры мы пришли к выбору между первыми двумя вариантами.

### **Нативные приложения на Java/Kotlin.**

Преимущества:

- Высокая производительность и скорость работы;
- Полный доступ к аппаратным возможностям смартфона: камера, геолокация, микрофон и т.д.;
- Максимальная оптимизация работы на мощности устройства;

- Нативный доступ к хранилищу данных устройства;
- Большие возможности по взаимодействию с другими приложениями.

Недостатки:

- Более сложный процесс разработки;
- Для каждой платформы необходимо писать отдельное приложение;
- Большое количество кода (особенно для достижения множественной платформенности).

### **Гибридные приложения на фреймворках.**

Преимущества:

- Одновременный запуск на нескольких платформах (iOS, Android и т.д.);
- Упрощенный процесс разработки и тестирования;
- Одновременное развитие этих приложений для всех платформ;
- Требуется меньше кода (в сравнении с нативными приложениями);
- Использование форматов, ориентированных на веб-разработку.

Недостатки:

- Невозможность использования API и функций, которые есть только у конкретной мобильной платформы;
- Низкая производительность (гибриды работают медленнее, чем нативные приложения);
- Проблемы с кэшированием и автономной работой, степень зависит от особенностей фреймворка;
- Могут потребовать дополнительных расходов на настройку и обслуживание.

Исходя из сравнения преимуществ и недостатков данных вариантов, было принято решение разрабатывать нативное приложение. В качестве языка программирования был выбран Kotlin.

### **2.2.13. Стандартная библиотека Java**

Некоторые из наиболее распространенных и полезных возможностей стандартной библиотеки Java включают в себя:

- Работа с вводом и выводом. Стандартная библиотека содержит классы, которые позволяют считывать и записывать данные из файлов, потоков и других источников.
- Работа с сетью. Java позволяет создавать клиент-серверные приложения, обмениваться данными между компьютерами и обрабатывать сетевые запросы.
- Работа с базами данных. С помощью стандартной библиотеки Java можно подключаться к различным базам данных и выполнять запросы на чтение и запись данных.
- Работа с многопоточностью. Java имеет мощную поддержку многопоточности, позволяющую создавать приложения, которые могут параллельно выполнять несколько задач.
- Работа с XML и JSON. Стандартная библиотека содержит классы, которые упрощают чтение, запись и обработку данных в форматах XML и JSON.
- Работа с коллекциями. Java имеет богатый выбор классов и интерфейсов для работы с различными типами коллекций данных, включая списки, стеки, очереди, карты и многие другие.

- Работа с графическим интерфейсом. Java содержит набор классов и компонентов для создания интерактивных пользовательских интерфейсов, включая окна, кнопки, меню и многое другое.

Это только некоторые из возможностей, предоставляемых стандартной библиотекой Java. В целом, Java является мощным и гибким языком программирования, который предлагает множество инструментов для решения разнообразных задач. Данная библиотека была использована для работы с потоками и файловой системой, так как Java совместима с Kotlin.



## **2.3. Машинное обучение**

### **2.3.1. Язык программирования Python**

Python [14] - это высокоуровневый язык программирования, который регулярно и успешно применяется для решения задач в областях анализа данных и машинного обучения. За годы существования языка для него для него было написано много библиотек упрощающих процесс решения задач в подобных областях.

### **2.3.2. Google Colab**

Google Colab [15] - это бесплатный сервис от Google, который предоставляет возможность создавать и запускать Jupyter-ноутбуки в облаке, поддерживает написание кода на Python. Google Colab предоставляет доступ к различным графическим процессорам (GPU) для ускорения вычислений, что особенно полезно при работе с машинным обучением и глубоким обучением.

### **2.3.3. NumPy**

NumPy [16] - это открытая библиотека для языка программирования Python, которая содержит инструменты для работы с многомерными массивами, выполнения математических операций, линейной алгебры и многого другого. За счет оптимизированной работы с многомерными массивами скорость работы алгоритма с использованием NumPy приблизительно равна скорости эквивалентного кода, выполняемого в MATLAB (эквивалентные алгоритмы, написанные на чистом Python часто работают значительно медленнее). NumPy можно расширять с помощью других библиотек Python, таких как SciPy и Pandas, что позволяет выполнять более сложные задачи анализа данных.

### **2.3.4. Pandas**

Pandas [17] - это библиотека для языка программирования Python, которая предоставляет высокоуровневые структуры данных и инструменты для работы с ними. Она используется для анализа и выполнением различных операций с данными, включая чтение и запись данных из различных форматов файлов, фильтрацию, сортировку, группировку, агрегацию и многое другое. Pandas оптимизирован для работы с большими объемами данных, что позволяет обрабатывать их быстро и эффективно.

### **2.3.5. Matplotlib**

Matplotlib [18] - это библиотека для языка программирования Python, которая предназначена для создания графиков различных типов (например, для визуализации данных).

### **2.3.6. Sklearn**

Sklearn (или Scikit-learn) [19] - это библиотека машинного обучения для языка Python. Она содержит множество алгоритмов классификации, регрессии, кластеризации, анализа данных и прочих методов машинного обучения, а также инструменты для предобработки данных, выбора признаков и оценки моделей.

### **2.3.7. PyTorch**

PyTorch [20] - это одна из самых популярных библиотек для машинного обучения и глубокого обучения, которая предоставляет инструменты для создания и обучения нейронных сетей. Кроме того, существует расширение библиотеки - PyTorch Mobile [21],

которое позволяет разработчикам создавать и развертывать модели машинного обучения на мобильных устройствах.

### 2.3.8. Optuna

Optuna [22] - это библиотека для автоматизации поиска гиперпараметров модели машинного или глубокого обучений, что позволяет улучшить их качество. Optuna совместима с моделями множества библиотек для машинного и глубокого обучений, часто используется на соревнованиях Kaggle.

**trial** - это объект, который используется для определения значения гиперпараметров в каждой итерации оптимизации. Он содержит методы `suggest`, которые предлагают новые значения гиперпараметров для модели в каждой итерации, а также методы `report`, которые сообщают о точности модели при использовании определенных значений гиперпараметров. Каждый `trial` представляет собой одну итерацию оптимизации.

**study** - это объект, который представляет собой исследование оптимизации гиперпараметров. Он содержит информацию о прогрессе оптимизации, лучшем наборе гиперпараметров и точности модели, обученной на основе этих гиперпараметров.

Для использования Optuna для поиска гиперпараметров модели необходимо определить пространство поиска гиперпараметров, которые будут подбираться. Например, для Catboost можно задать диапазон значений для скорости обучения, глубины деревьев. Затем необходимо определить функцию оценки, которая будет использоваться для оценки качества модели на каждой итерации подбора гиперпараметров и провести исследование (`study`), которое будет состоять из  $N$  испытаний (`trial`).

После завершения процесса подбора гиперпараметров можно использовать лучшие значения для обучения модели и получения

наилучшего качества предсказаний. Более подробно о работе библиотеки можно прочитать в статье [23].

### **2.3.9. Catboost**

Catboost [24]- это библиотека градиентного бустинга, разработанная компанией Yandex, используется для решения различных классов задач.

Бустинг-подход основан на объединении нескольких слабых функций (которые имеют ограниченную обобщающую способность) в итеративном процессе, где каждая последующая модель обучается с использованием информации об ошибках предыдущих моделей [25]. В результате каждая новая модель учитывает ошибки предыдущих и становится более точной. Результирующая функция представляет собой линейную комбинацию базовых, слабых моделей. В случае Catboost каждое новое дерево строится на основе ошибок предыдущих деревьев, чтобы улучшить точность модели. Catboost показывает хорошую точность по сравнению с другими моделями градиентного бустинга и часто используется на соревнованиях по машинному обучению [26].

### **2.3.10. Полносвязная нейронная сеть**

Полносвязная нейронная сеть [27] - это тип нейронной сети, где каждый нейрон в слое соединен со всеми нейронами в предыдущем и следующем слоях.

Прямое распространение - это процесс передачи входных данных через слои нейронной сети, где каждый слой выполняет линейную трансформацию входных данных и применяет нелинейную функцию активации к результату. Результатом прямого распространения является выходной вектор, который может быть использован для принятия решений или предсказаний. После этого происходит оценки ошибки:

сравниваются предсказания модели с истинным ответом. После чего выполняется обратное распространение.

$$nn = \sum_{i=1}^{i=n} w_i \cdot x_i$$

Нелинейность нейронной сети добавляется с помощью активации, которая должна быть нелинейна и дифференцируема. ReLU - одна из самых популярных функций активации:

$$ReLU(x) = \max(x, 0)$$

Обратное распространение в полносвязной нейронной сети - это алгоритм обучения, который позволяет настраивать веса между нейронами, чтобы минимизировать ошибку предсказания модели. Ошибка распространяется назад через сеть, начиная с выходного слоя и двигаясь к входному слою. Каждый нейрон в сети получает часть ошибки, пропорциональную его вкладу в предсказание. Используя полученную ошибку, обновляются веса между нейронами таким образом, чтобы уменьшить ошибку при следующем прямом распространении. Частная производная ошибки по весам  $\omega_{ij}$ .

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial O_j} \frac{\partial O_j}{\partial nn_j} \frac{\partial nn_j}{\partial \omega_{ij}}$$

Dropout [27] - это метод регуляризации в нейронных сетях, который заключается в случайном исключении некоторых нейронов во время обучения с заданной вероятностью  $p$ . Это помогает снизить переобучение и улучшить обобщающую способность модели.

### 2.3.11. Оптимизатор Adam

Формула обновления весов модели:

$$\omega_{t+1} = \omega_t - \eta_t \nabla Q(\omega_t)$$

У классической вариации градиентного спуска есть несколько проблем, ниже представлены некоторые из них:

**Проблема:** алгоритм может найти локальный минимум функции, посчитать его за глобальным минимумом и застрять. При этом глобальный минимум найден не будет.

**Проблема:** седловая точка.

В направлении, соответствующем одному параметру найден локальный минимум, при этом во втором направлении, которое соответствует другому параметру, кривая будет находиться в локальном максимуме. Такая точка называется седловой. Область, окружающая ее, называется платом (она выглядит довольно плоско), что означает практически нулевые градиенты. Оптимизатор пытается найти решение в направлении первого параметра, колеблясь около седловой точки. Хотя для нахождения истинного решения ему необходимо спуститься по уклону ко второму параметру.

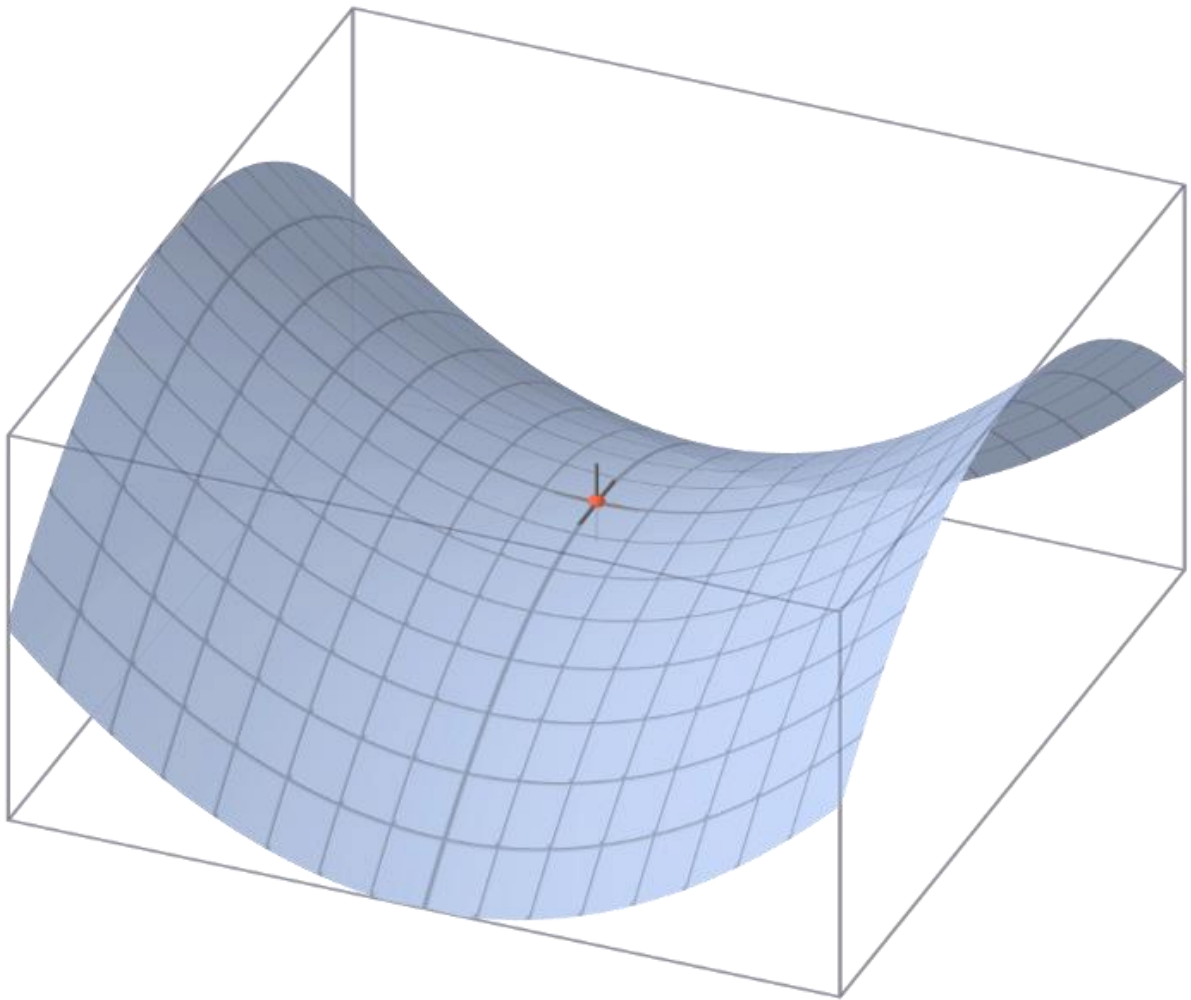


Рис. 1 Седловая точка

(Привод. по [28])

**Проблема:** стохастический градиентный спуск может образовывать сильные колебания при движении к точке минимума.

**Метод импульсов:**

Идея: усреднять градиент по шагам с помощью экспоненциального скользящего среднего.

$$v = \gamma v + (1 - \gamma) \eta_t \nabla Q(\omega_t)$$

$$\omega_{t+1} = \omega_t - v$$

$$N \approx \frac{2}{\gamma} - 1$$

$\nu$  - экспоненциальное скользящее среднее;

$\gamma$  - параметр экспоненциального скользящего среднего, для регулировки количества градиентов, которые будут учтены в формуле;

$N$  - приближенное количество усредненных градиентов, коэффициенты перед остальными равны почти нулю.

### **RMSProp:**

Из-за того, что в задачах многомерной оптимизации число подбираемых весовых коэффициентов может быть десятки или сотни тысяч.

Вычисленные частные производные по каждому из параметров могут сильно отличаться. В таком случае, один весовой коэффициент может меняться значительно медленнее других. Чтобы нормализовать скорость адаптации разных весовых коэффициентов Джеффри Хинтон в своем курсе по Глубокому обучению предложил делать следующую перенормировку шага обучения.

Вычисляем экспоненциальное скользящее среднее для квадрата градиентов:

$$G = \alpha G + (1 - \alpha) \nabla Q(\omega_t) \odot \nabla Q(\omega_t)$$

Обновляем веса:

$$\omega_{t+1} = \omega_t - \eta \frac{\nabla Q(\omega_t)}{\sqrt{G} + \epsilon}$$

$\epsilon > 0$ , чтобы избежать деления на ноль

### **Adam (adaptive momentum) [29]:**



Совмещает два подхода, описанных выше. в итоге:

$$v = \gamma v + (1 - \gamma) \eta_t \nabla Q(\omega_t)$$

$$G = aG + (1 - a) \nabla Q(\omega_t) \odot \nabla Q(\omega_t)$$

Выполняем нормировку, чтобы на первых итерациях алгоритма принимали более большое значение, а на последних поменьше:

$$\dot{v} = v / (1 - \gamma^{t+1})$$

$$\hat{G} = G / (1 - a^{t+1})$$

Используем параметры для корректировки весов:

$$\omega(t + 1) = \omega(t) - \eta \frac{\dot{v}}{\sqrt{\hat{G} + \epsilon}}$$

Оптимизатор Adam отлично себя зарекомендовал и получил широкое практическое применение для решения различных задач.

### 2.3.12. CrossEntropyLoss

CrossEntropyLoss [30] - это функция потерь, которая используется в задачах классификации для оценки разницы между предсказанными и истинными значениями целевой переменной. Она вычисляет ошибку между предсказанными вероятностями классов и истинными метками классов, используя формулу кросс-энтропии. Чем больше разница между предсказанными и истинными значениями, тем выше значение CrossEntropyLoss. Она является одной из наиболее распространенных функций потерь в задачах классификации. Для задачи классификации формула выглядит следующим образом:

$$L_{ce} = - \sum_{i=1}^n t_i \log(p_i), \text{ где}$$

n - число классов

$p_i$  - предсказанная вероятность

$t_i$  - индикатор

### 2.3.13. Матрица ошибок и метрика Accuracy

Матрица ошибок [31] - это таблица, которая показывает количество верно и неверно классифицированных примеров в задачах классификации. Она состоит из четырех ячеек:

- True Positive (TP) - количество верно классифицированных примеров положительного класса;
- False Positive (FP) - количество неверно классифицированных примеров положительного класса, является ошибкой I рода;
- False Negative (FN) - количество неверно классифицированных примеров отрицательного класса, является ошибкой II рода;
- True Negative (TN) - количество верно классифицированных примеров отрицательного класса.

Матрица ошибок позволяет оценить качество работы модели и вычислить различные метрики. Классическая матрица ошибок строится для бинарной классификации и имеет размерность 2x2. В случае, построения матрицы ошибок для N классов, матрица будет иметь вид NxN, все TP значения будут находиться на главной диагонали матрицы. Accuracy (точность) - это метрика, которая показывает долю правильных ответов модели относительно общего числа наблюдений. Она вычисляется как сумма правильных ответов, поделенная на общее число наблюдений:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

### **2.3.14. Кросс-валидация**

Кросс-валидация [32] - это метод оценки качества обучения модели на основе разбиения исходного набора данных на обучающую и тестовую выборки. В процессе кросс-валидации данные разбиваются на  $k$  равных частей (фолдов), из которых  $(k-1)$  фолд используется для обучения модели, а оставшийся - для тестирования. Этот процесс повторяется  $k$  раз, каждый раз используя один фолд для тестирования и остальные фолды для обучения модели. В результате получается  $k$  оценок качества модели, которые затем усредняются для получения окончательной оценки. Кросс-валидация позволяет более точно оценить качество модели и уменьшить вероятность переобучения.

## **2.4. Выводы по главе**

Во второй главе были рассмотрены инструменты, которые использовались для разработки комплекса, а также некоторая теоретическая информация.

## **Глава 3. Разработка и архитектура приложения**

### **3.1. Аппаратная часть устройства**

#### **3.1.1. Общие сведения**

При разработке устройства мы решили использовать актуальные компоненты. Выбор основы устройства производился среди линейки микроконтроллеров компании STMicroelectronics, ввиду наличия опыта работы с их продукцией, обильной документации и наличия нужного функционала. Компания также предоставляет инструменты разработки, которые и были использованы для ускорения реализации прототипа. Подбор прочих компонентов производился исходя из функциональных требований проекта, но с уклоном в продукцию компании STMicroelectronics, для упрощения закупки комплектующих, а также в результате изучения опыта создания аналогичных по функционалу решений на основе этих компонентов.

Несмотря на обилие технической документации. Возникали множественные проблемы во время аппаратной реализации. Некоторые ключевые примеры реализации, предоставленные разработчиком, не работали в полной мере, или работали со сбоями, особенно это коснулось модуля питания, который нестабильно работал в реализации предлагаемой STMicroelectronics. Малое количество сторонних проектов, и публикаций по некоторым компонентам, существенно осложняло понимание принципов их работы на практике. На основе трудоемкого анализа материалов все же удалось установить принципы работы требуемых компонентов, но часть дополнительного функционала, который хотелось

реализовать дополнительно, была отложена для реализации в следующей итерации проекта.

### 3.1.2. Разработка аппаратной части

При разработке аппаратной части устройство был поделено на модули:

**Модуль USB** - комплекс резисторов и защитных диодов, разъема Type-C, подключен напрямую к микроконтроллеру и к шине питания.

Предоставляет возможность передавать данные и электрическую энергию модулю питания. Схема представлена на рисунке 0.

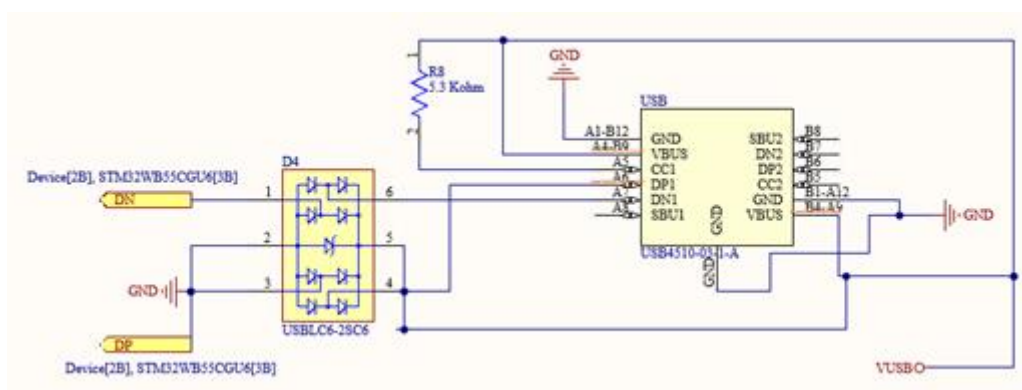


Рис. 2 Схема USB

**Модуль питания**, который представляет из себя комплекс резисторов, конденсаторов, аккумулятора, и 2 компонентов, которые регулируют питание устройства. LSM6DSR контролирует по заложенным в него алгоритмам разряд-заряд Li-ion аккумулятора, осуществляет индикацию процесса зарядки разрядки и передает данные на микроконтроллер. LD3985M33R преобразует напряжение с аккумулятора в напряжение +3.3V нужное для работы микроконтроллера и модулей. Схемы представлена на рисунке 0 и 0.

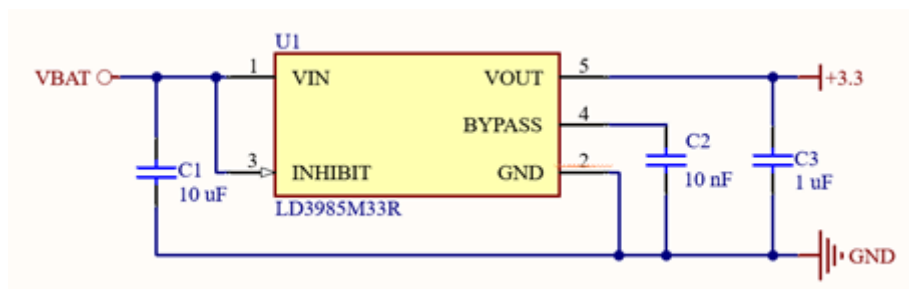


Рис. 3 Схема LD3985M33R

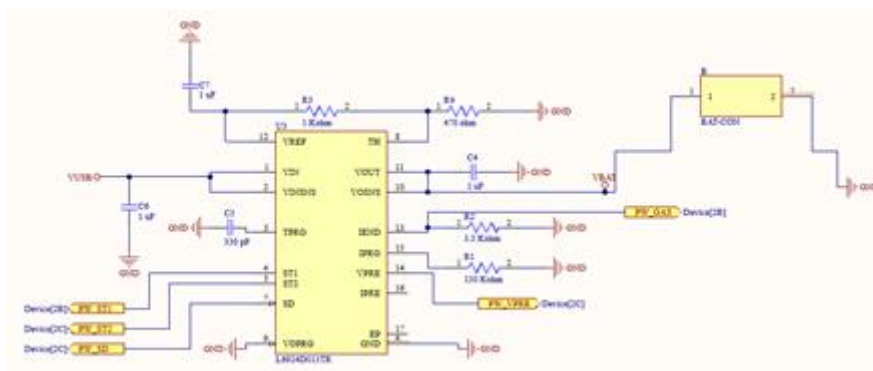


Рис. 4 Схема LSM6DSR

**Модуль памяти**, который представляет из себя M24512-DFMC6TG который посредством шины I2C позволяет записывать и считывать данные в энергонезависимую память. Схема представлена на рисунке 0.

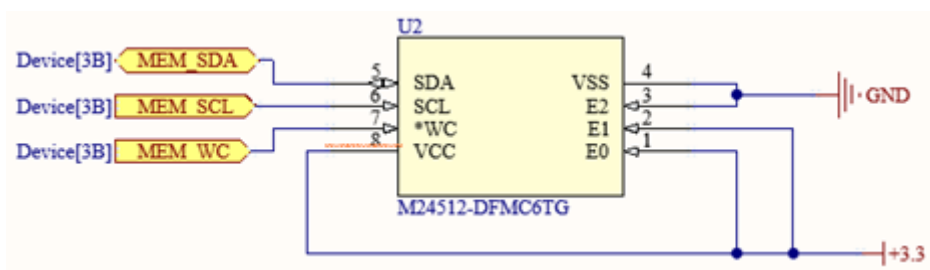


Рис. 5 Схема M24512-DFMC6TG

**Модуль вибромотора**, который реализован непосредственно из самого вибромотора, а также обвеса в виде резисторов, диода и n-канального

транзистора. Управляется посредством генерации ШИМ сигнала на управляющем входе. Схема представлена на рисунке 0.

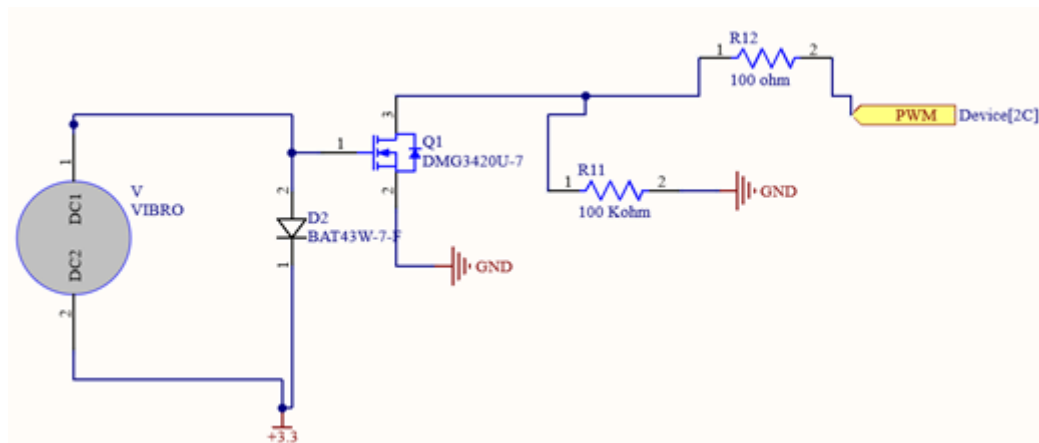


Рис. 6 Схема вибромотора

**Модуль SWD**, для отладки и загрузки прошивки в прототип. Схема представлена на рисунке 0.

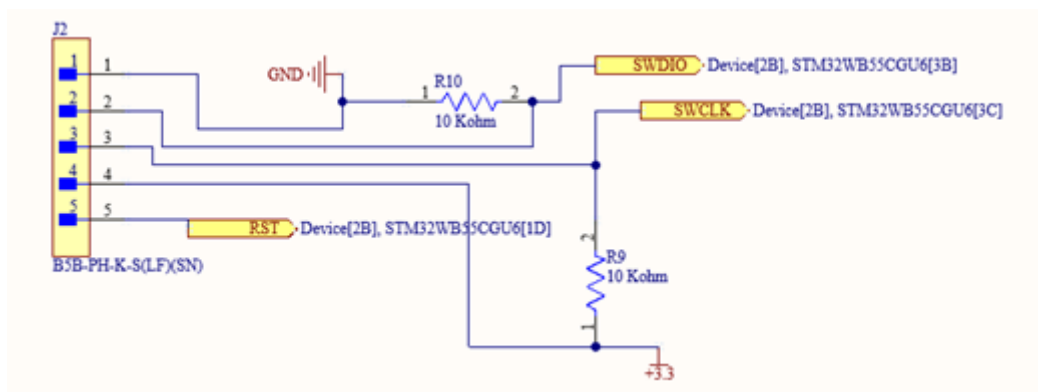


Рис. 7 Схема SWD

**Модуль NFC**, дополнительный модуль на базе ST25DV64K-JFR6D3 для дальнейшей разработки ПО. Модуль посредством шины I2C позволяет записывать и считывать данные в энергонезависимую память чипа, для отправки данных по NFC. Схема представлена на рисунке 0.

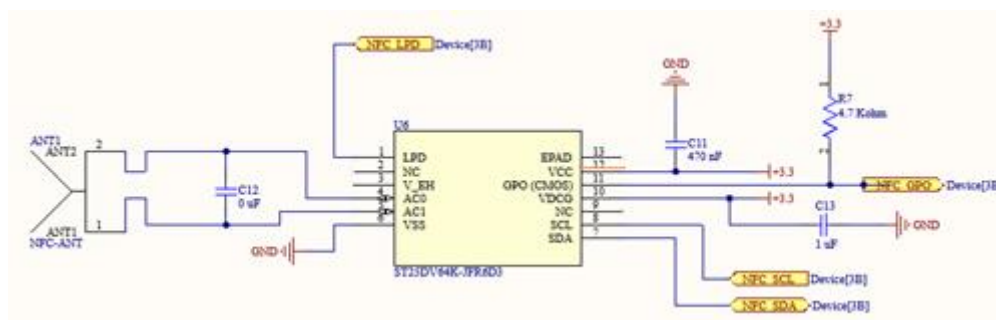


Рис. 8 Схема ST25DV64K-JFR6D3

**Модуль Датчиков**, представляет из себя LSM6DSRTR который посредством шины I2C позволяет отправлять данные акселерометра и гироскопа на микроконтроллер. Дополнительно к датчику по вторичной I2C шине подключен LPS22HBTR для получения данных атмосферного давления. Для дальнейшей разработки и улучшения ПО. Схемы представлены на рисунке 0 и 0.



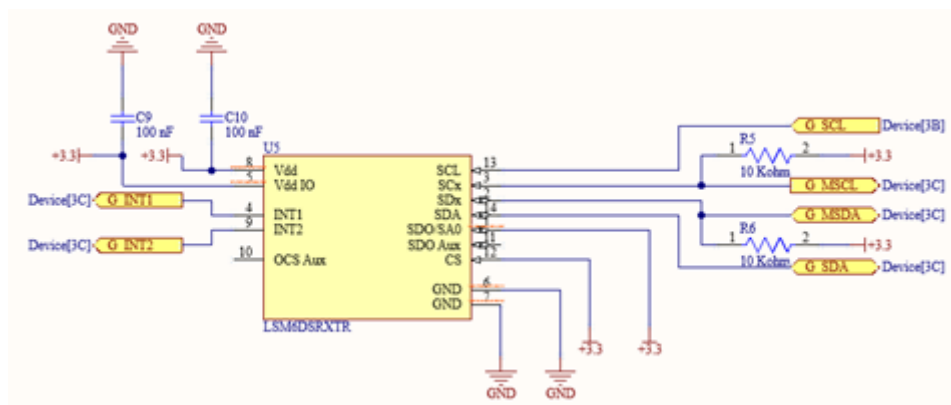


Рис. 9 Схема LPS22HBTR

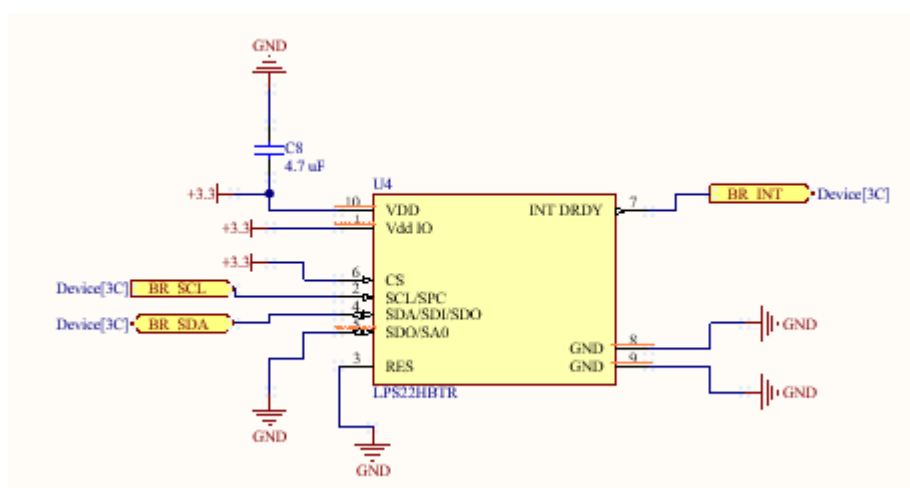


Рис. 10 Схема LSM6DSRTR

**Основной модуль**, представляет из себя микроконтроллер STM32WB55CG, комплекс резисторов, конденсаторов, кварцевого резонатора, часового кварца, и блока из RF фильтра MLPF-WB55-02E3 специализированного для использования с сетями Bluetooth и антенны. Посредством этого модуля устройство функционирует и через ядро Bluetooth осуществляет подключение к другим устройствам в рамках сети. Схема представлена на рисунке 0.

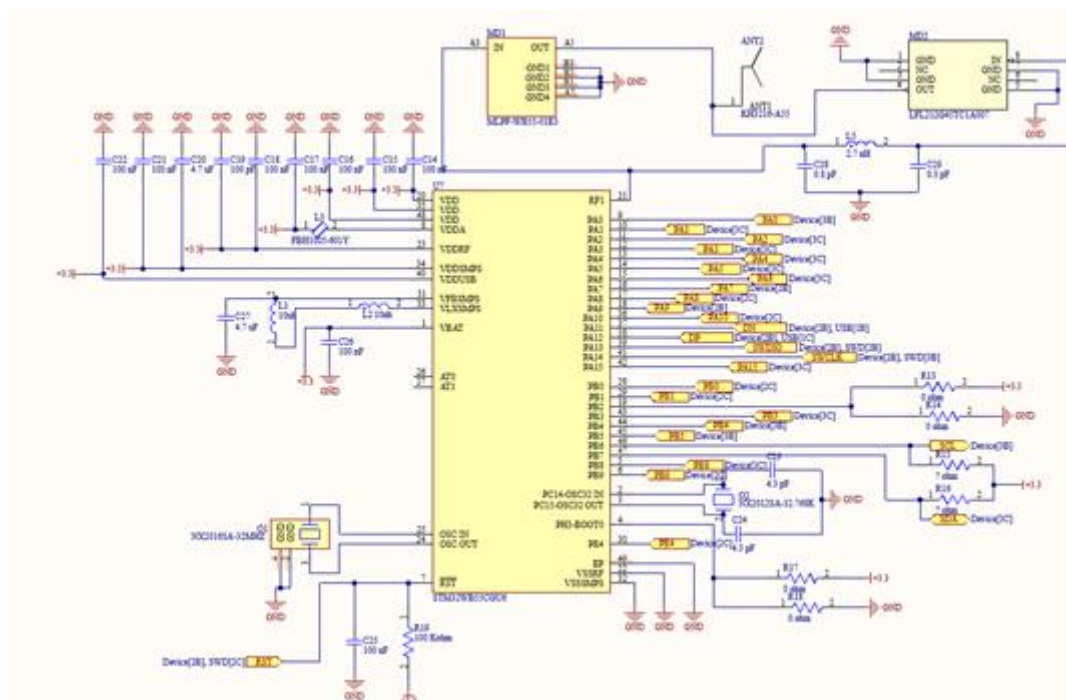


Рис. 11 Схема основного модуля

Схемы сторон платы, верхней и нижней, представлены на рисунке 0 и рисунке 0 соответственно:

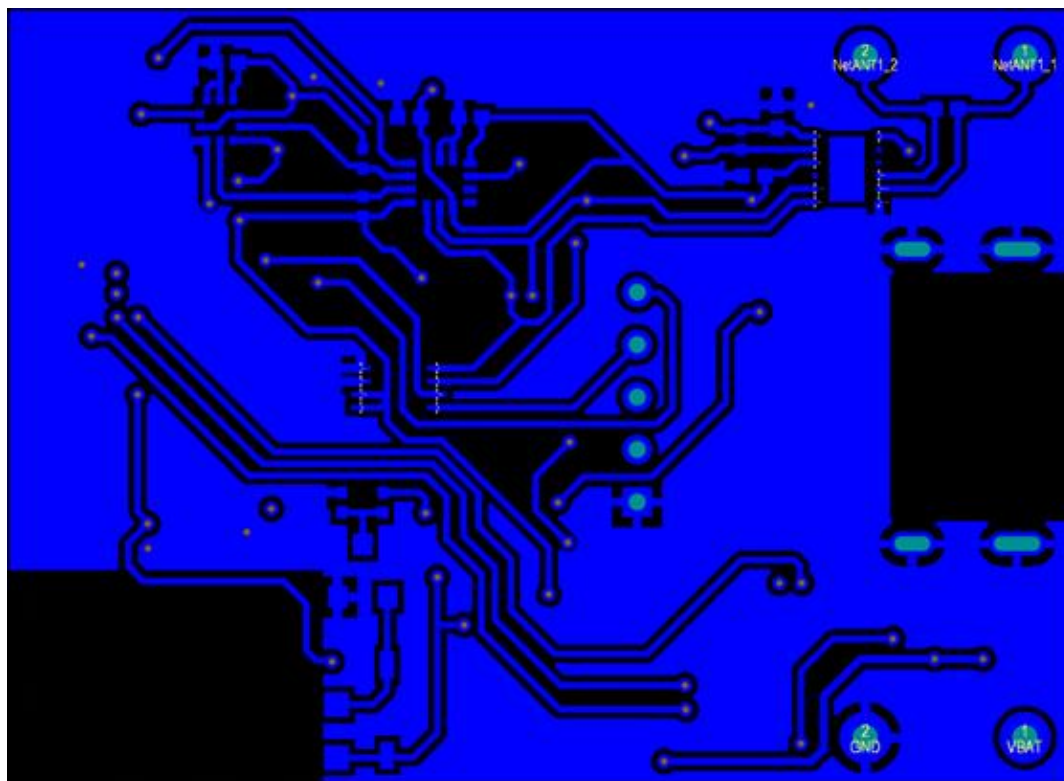


Рис. 12 - Схема нижней части платы устройства

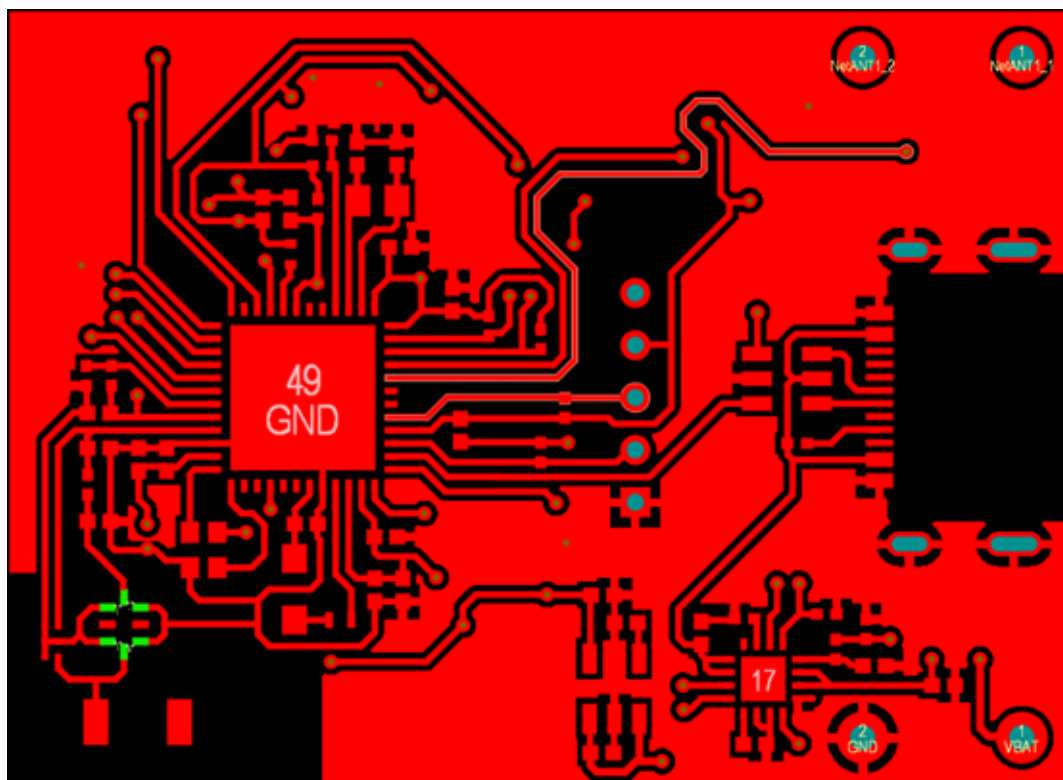


Рис. 13 - Схема верхней части платы устройства

### 3.1.3. Архитектура аппаратной части

Архитектура аппаратной части приведена в виде блок-схемы на рисунке 0 и объединяет все перечисленные выше модули:

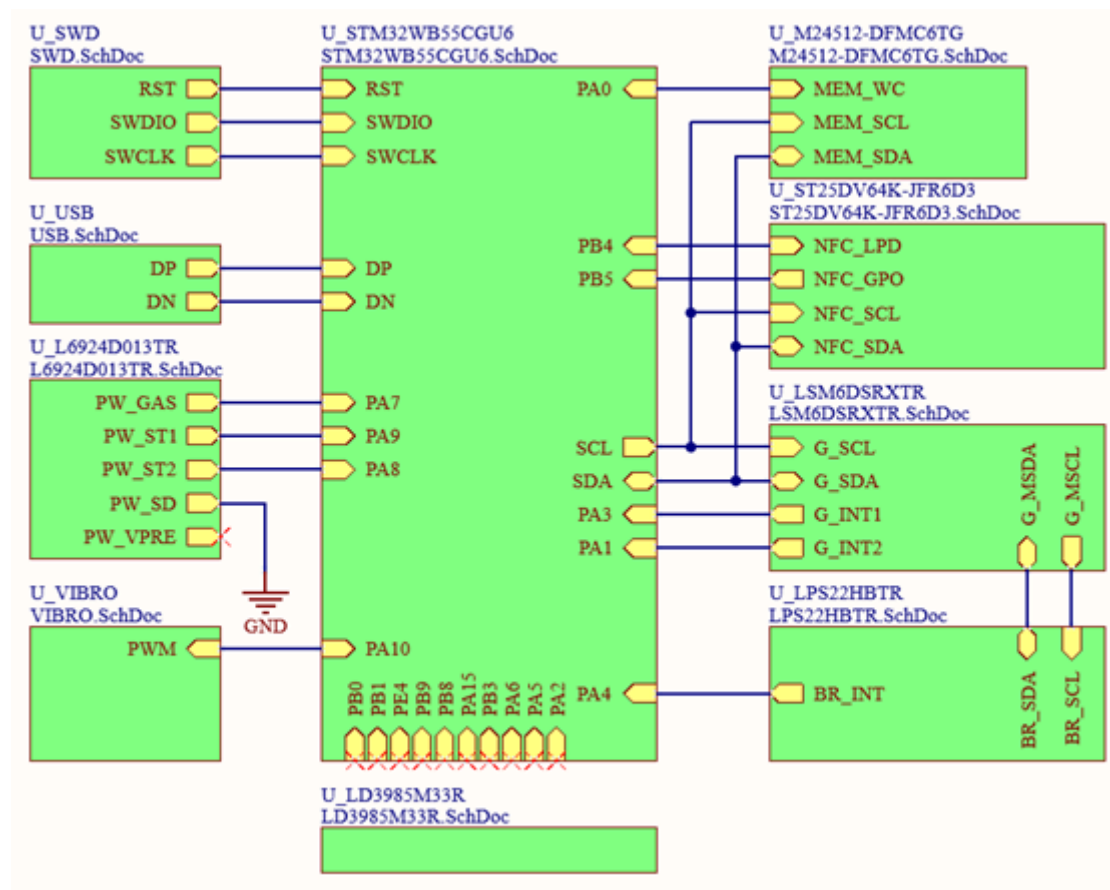


Рис. 14 Архитектура устройства

### 3.1.4. Средства и инструменты разработки аппаратной части

Аппаратная часть проекта разрабатывалась с помощью инженерного программного обеспечения САПР Altium Designer и документации от производителей комплектующих.

## 3.2. Программная часть устройства

### 3.2.1. Общие сведения

Разработка программной части устройства потребовала ориентироваться на документацию, предоставляемую STMicroelectronics, примеры [33], вебинары, и другие проекты на GitHub. Возникли проблемы с качеством найденных материалов. Многие примеры, которые были предоставлены STMicroelectronics, представляли собой исходные коды, не совместимые с конфигуратором встроенным в IDE, код генерируемый IDE претерпел существенные изменения, и не всегда соответствовал вебинарам. Все это существенно замедляло разработку. Малое количество сторонних проектов, и публикаций, усложняло понимание системы продуктов компании STMicroelectronics. Некоторые библиотеки, не представляли полный функционал, предоставляемый датчиками. Но на основе анализа материалов все же удалось установить принципы работы Фреймворков и библиотек. Структурно по итогам разработки, функционирование приложения разделено на следующие модули:

- Модуль инициализации всех компонентов библиотеки HAL, планировщика, сервера времени, службы беспроводной связи, которые используются в программном обеспечении устройства, в соответствии с конфигурациями в IDE, на основании аппаратной части.
- Реализованный посредством HAL I2C протокол, общается с датчиками устройства, настраивает их и считывает данные. Оптимизировано время, затрачиваемое на обмен данными между микроконтроллером и подчиненными I2C устройствами.

- Планировщик, отвечает за планирование и выполнение задач фоне в заданной последовательности. Позволяет содержать до 32 задач, управлять ими и алгоритмом их исполнения. Планировщик необходим для корректной работы беспроводного стека и используется для организации передачи данных, процессом публикации данных в сети Bluetooth.
- Реализованный посредством STM32\_WPAN, Bluetooth стек, управляет работой протокола по спецификации. Библиотекой реализован GATT сервер, содержащий три сервиса для хранения характеристик и организации доступа к ним. Урегулированы процессы отключения, повторного подключения и отключения, отправка уведомлений и доступа к чтению и записи характеристик.
- Сервер времени на базе Bluetooth ядра, который предоставляет таймеры для регулирования времени выполнения некоторых операций на основании показателей часов реального времени, которым необходима задержка или цикличность, например процесс публикации данных в сети Bluetooth.

Использовались и другие программные компоненты библиотеки HAL, для организации работы ШИМ, семафора, управления GPIO, прерываний.

### **3.2.2. Средства и инструменты разработки программной части**

При разработке программной части аппаратной составляющей проекта использовались приложения на android от компании производителя ST BLE Profile, ST BLE Sensor для отладки, а также STM32CubeProgrammer, STM32CubeIDE, для разработки, также

использовались прочие публичные источники информации от производителя микроконтроллеров и от сторонних разработчиков.

### **3.2.3. Реализация профиля GATT**

В соответствие со спецификацией Bluetooth в качестве идентификаторов сервисов и характеристик мы выбрали 128 битные UUID. Для работы протокола Bluetooth был реализован следующий GATT профиль:

- Сервис данных датчиков, содержит в себе одну характеристику, для передачи данных приложению компаньону. Состоит из 17 байт. 4 байта выделены для номера отправляемого пакета, по 2 байта на три показателя гироскопа и три показателя акселерометра, 1 байт для будущих показаний барометра. Характеристика реализует дескриптор для осуществления уведомлений (нотификаций), для реализации возможности подписки на данные приложением компаньоном.
- Сервис батареи, состоит из 2 характеристик, первая хранит показания аккумулятора устройства, и доступно только для чтения. Вторая хранит статус зарядки и разрядки батареи, и так же доступна только для чтения.
- Сервис управления, состоит из 3 характеристик, каждая из которых доступна для чтения и записи и выступает в роли флага для управления устройством. Это характеристики для сброса, обновления и синхронизации устройства с приложением компаньоном.

## 3.3. Приложение компаньон

### 3.3.1. Интерфейс приложения компаньона

При запуске приложение запрашиваются все необходимые характеристики: метод `checkPermissions` проверяет, есть ли у приложения необходимые разрешения. Если разрешения отсутствуют, то происходит запрос разрешений с помощью метода `requestPermissions`. Если разрешения есть, то метод возвращает `true`, иначе – `false`.

Список необходимых разрешений:

- Разрешение на использование Bluetooth-модуля устройства (BLUETOOTH) - для возможности подключения к другим Bluetooth-устройствам, передачи данных и управления устройством.
- Разрешение на администрирование Bluetooth-модуля (BLUETOOTH\_ADMIN) - для доступа к дополнительным функциям управления устройством (например, для изменения настроек и состояния Bluetooth-модуля).
- Разрешение на подключение к Bluetooth-устройствам (BLUETOOTH\_CONNECT) - для возможности установления соединения с другим устройством.
- Разрешение на поиск Bluetooth-устройств (BLUETOOTH\_SCAN) - для поиска доступных Bluetooth-устройств в окружающей среде.
- Разрешение на доступ к грубой геолокации (ACCESS\_COARSE\_LOCATION) - для определения приблизительного местоположения устройства при использовании некоторых функций приложений.
- Разрешение на доступ к точной геолокации (ACCESS\_FINE\_LOCATION) - для определения точного



местоположения устройства при использовании некоторых функций приложений.

- Разрешение на запись во внешнее хранилище (WRITE\_EXTERNAL\_STORAGE) - для сохранения данных, создания файлов и доступа к файлам на внешнем устройстве хранения.

Далее пользователь попадает на экран подключения к новому устройству (DeviceConnectionActivity), если ранее не было сопряжено устройство, иначе он попадает сразу на главный экран (GeneralActivity). При нажатии на кнопку, вызывается метод newConnectionOnClick который инициирует переход в окно выбора устройства (SelectDeviceActivity). На экране выбора устройства происходит сканирование устройств Bluetooth и дается возможность пользователю выбрать устройство, с которым он хочет установить соединение, после чего пользователю предлагается привязать устройство к телефону и подключиться к нему.

После привязки пользователь переходит к главному экрану приложения для работы с устройством. На главном экране в виде графика отображаются обработанные данные с устройства по активности, также отображаются некоторые численные параметры. Свайпом с правой стороны можно перейти в аналогичный фрагмент, отображающий данные по фазам сна. Также на этом окне находится боковое выдвижное меню (навигационное меню), которое представляет собой меню с настройками, которое можно открыть при помощи свайпа с левой стороны экрана.

Для отображения данных активности была выбрана столбчатая диаграмма, показывающая обработанные данные по количеству шагов. Данную статистику можно посмотреть в разрезе различных периодов

(день, неделя, месяц, год). Также на данном экране в текстовом формате показаны пройденное расстояние, количество шагов и затраченное время.

Для отображение данных сна была выбрана горизонтальная столбчатая диаграмма, показывающая обработанные данные по фазам сна. Данную статистику также можно посмотреть в разрезе различных периодов (день, неделя, месяц, год). На данной вкладке отображается время, проведенное в различных фазах сна.

Из данного меню можно перейти в окно настроек персональных данных, где можно указать персональные данные пользователя, и на экран управления устройством, где можно посмотреть информацию о подключенном устройстве и послать на него некоторые команды.

Для генерации логотипа и иллюстраций приложения была использована Midjourney.

### 3.3.2. Реализация работы с Bluetooth

Класс **SelectDeviceActivity** предназначен для сканирования и выбора устройств Bluetooth, с которыми пользователь может установить соединение. Внутри класса есть несколько переменных и методов. Переменные отвечают за определение параметров и настроек для сканирования устройств Bluetooth и для отображения списка доступных устройств для пользователя в интерфейсе приложения. Функции определяют логику работы приложения и включают в себя проверки наличия Bluetooth на устройстве, запросы разрешений на использование Bluetooth, сканирование Bluetooth-устройств, отображение списка устройств в приложении, обработку результатов сканирования и т.д.

Сервис **BluetoothLE** используется для установления соединения между устройствами BluetoothLE и передачи данных. Класс **BluetoothLeService** является потомком класса **Service**, что позволяет работать с приложением в фоновом режиме. Переменные класса включают в себя объект **BluetoothManager**, который управляет Bluetooth-адаптером на устройстве; **BluetoothAdapter**, который обеспечивает доступ к Bluetooth-адаптеру; **BluetoothDeviceAddress**, который представляет адрес Bluetooth-устройства, с которым происходит соединение; **BluetoothGatt**, который предоставляет интерфейс для совершения транзакций BluetoothLE; **ConnectionState**, который отслеживает текущее состояние соединения; **ExecutorService**, который предоставляет базовую реализацию **Executor**, и т.д. Внутренний класс **BluetoothGattCallback** берет на себя управление соединением BluetoothLE и реагирует на события, связанные с соединением, такие как изменение состояния соединения и обнаружение сервисов и реализует следующие функции:

- Функция **initialize()** инициализирует **BluetoothLeService**, проверяя, что **BluetoothManager** и **BluetoothAdapter** инициализированы.
- Функция **connect()** используется для установления соединения с определенным устройством BluetoothLE.
- Функция **disconnect()** используется для разрыва текущего соединения.
- Функция **close()** закрывает текущее соединение и удаляет объект **BluetoothGatt**.
- Функция **readCharacteristic()** используется для чтения значения характеристики **BluetoothGatt**.
- Функция **writeCharacteristic()** используется для записи заданного значения характеристики **BluetoothGatt**.

- Функция `setCharacteristicNotification()` используется для настройки оповещения о изменении характеристики `BluetoothGatt`.

Класс `BluetoothLeService` также содержит несколько констант, которые определяют действия и данные, используемые при передаче сообщений между устройствами `BluetoothLE`.

Класс **`GeneralActivity`** содержит реализации различных методов жизненного цикла `Activity`, таких как `onCreate()`, `onPause()` и `onDestroy()`. Внутри класса определяются несколько полей и методов, некоторые из которых являются статическими полями и методами, доступными для красивого отображения списка в `NavigationDrawer` и возможности экспорта и очистки данных, оставшихся от устройства. Также класс реализует `ServiceConnection` и `BroadcastReceiver`, которые позволяют приложению подключиться к `BluetoothLeService` и обновлять интерфейс при получении сообщений об изменениях состояния соединения. При получении соответствующих сообщений, вызывается методы `updateConnectionState()` или `displayData()`. Класс содержит следующие важные функции:

- Функция `notifyChars()` проходит по набору сервисов, получает их `uuid` и имя, затем проходит по характеристикам каждого сервиса. Если `uuid` текущей характеристики совпадает с нужной, то вызывается метод `setCharacteristicNotification()` класса `BluetoothLeService`, для уведомления об изменении данных в характеристике.
- Функция `exportOnClick()` экспортирует данные в файл CSV на устройстве, а функция `clearOnClick()` очищает текущие данные об устройстве.

### 3.3.3. Реализация работы с данными

Для работы с данными, реализованы следующие функции:

- Функция `'save_data'` отвечает за сохранение данных в файл хранилища. В начале функции устанавливается имя файла хранилища и режим сохранения, который по умолчанию равен `'Context.MODE_APPEND'`. Затем создается объект `'FileOutputStream'`, который используется для записи данных в файл. Созданный объект передается в конструктор `'OutputStreamWriter'`, который используется для записи данных в поток в формате символов. Затем создается объект `'BufferedWriter'`, который используется для более эффективной записи данных. Далее происходит запись переданных данных в файл с помощью `'writer.write(data)'` и перевод в новую строку с помощью `'writer.newLine()'`. По завершению записи данных в файл, функция закрывает поток записи `'writer'`. Если в ходе работы функции возникает исключение, то оно обрабатывается с помощью `'e.printStackTrace()'`.
- Функция `'load_data'` отвечает за загрузку данных из файла хранилища и возвращает их в виде списка строк `'ArrayList<String?>'`. В начале функции происходит установка имени файла хранилища и создание объекта `'FileInputStream'`, который используется для чтения данных из файла. Полученный поток передается в конструктор `'InputStreamReader'`, который используется для чтения данных из потока в формате символов. Затем создается объект `'BufferedReader'` с переданным в конструктор `'InputStreamReader'` объектом, который используется для более эффективного чтения данных. Далее происходит

считывание данных из файла в цикле, пока не достигнут конец файла. Каждая прочитанная строка добавляется в список строк ``content``. По завершению чтения файла, функция возвращает список списка строк ``content``.

- Функция ``clear_data`` выполняет очистку данных в файле с именем `DATA_FILENAME`. Сначала открывается поток для записи в файл, используя функцию `openFileOutput()`. Затем создается объект `BufferedWriter`, который будет использоваться для записи данных в файл. Далее в файл записывается пустая строка, которая в результате очищает файл. Если в процессе выполнения возникает ошибка `IOException`, то она выводится на экран.
- Функция `savePersonalDataOnClick` сохраняет персональные данные пользователя (имя, фамилию и возраст) в локальном хранилище приложения (`SharedPreferences`) по нажатию на соответствующую кнопку. Сначала она получает доступ к файлу настроек `"user"`, используя метод `getSharedPreferences()`, и создает объект `Editor` для редактирования этого файла. Затем она вызывает методы `putString()` для добавления значений имени, фамилии и возраста, которые были введены пользователем в соответствующие поля на экране приложения с помощью `findViewById()`. Наконец, функция вызывает метод `commit()` объекта `Editor` для сохранения изменений в файле настроек. Таким образом, при следующем запуске приложения эти данные будут доступны для использования.

## 3.4. Машинное обучение

### 3.4.1. Создание набора данных

В результате сбора данных получилось большое количество файлов в формате .csv, которые необходимо было обработать и собрать в единый набор данных. Данные собирались разными людьми, с разной антропометрией. Для задачи о классификации активности файлы были поделены на четыре типа: сборщик данных стоял неподвижно, сборщик данных стоял на месте и с помощью небольших движений рук, ног и поворотов тела создавал шумы, сборщик данных шел в различном темпе, сборщик данных бежал. Название файла соответствовала типу активности сборщика данных.

Когда данных накопилось достаточное количество их было необходимо собрать в единый набор для последующего обучения моделей. Для этого использовалась библиотека pandas. С помощью функционала библиотеки csv файлы превращались в объекты класса DataFrame. Далее, в ручную были просмотрены csv файлы, в которых были представлены типы активности шаг и бег. Дело в том, что в первую и последнюю секунды сборщик данных включал прибор и, таким образом, эти секунды не соответствовали заявленному типу активности. Эти секунды были удалены перед конкатенацией csv файлов в единый набор данных. После чего очищенный DataFrame конкатенировался с финальным DataFrame и ему присваивалось численное значение в столбец target, которая соответствовала типу активности сборщика данных по следующим правилам:

- Отсутствие активности - 0;
- Шаги - 1;
- Бег - 2.

	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	time	target
0	181	-3007	2892	5	-7	-2	2023-05-10T21:23:17.026	0
1	185	-3010	2878	6	-7	-2	2023-05-10T21:23:17.103	0
2	187	-3012	2892	6	-7	-2	2023-05-10T21:23:17.180	0
3	186	-3009	2887	6	-7	-2	2023-05-10T21:23:17.379	0
4	184	-3011	2888	6	-7	-2	2023-05-10T21:23:17.393	0
...	...	...	...	...	...	...	...	...
639	-3829	-760	1455	1	-6	-7	2023-05-10T21:27:34.315	0
640	-3814	-759	1429	4	-12	-1	2023-05-10T21:27:34.390	0
641	-3836	-788	1404	2	-5	22	2023-05-10T21:27:34.468	0
642	-3800	-774	1379	0	5	12	2023-05-10T21:27:34.541	0
643	-3850	-791	1396	12	18	26	2023-05-10T21:27:34.613	0

644 rows x 8 columns

Рис. 15 Один из csv файлов представленный в виде DataFrame

	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z	time	target
0	181	-3007	2892	5	-7	-2	2023-05-10T21:23:17.026	0
1	185	-3010	2878	6	-7	-2	2023-05-10T21:23:17.103	0
2	187	-3012	2892	6	-7	-2	2023-05-10T21:23:17.180	0
3	186	-3009	2887	6	-7	-2	2023-05-10T21:23:17.379	0
4	184	-3011	2888	6	-7	-2	2023-05-10T21:23:17.393	0
...	...	...	...	...	...	...	...	...
6182	-775	-1804	2267	-1219	-1231	-1747	2023-05-10T17:49:40.425	2
6183	-453	-4694	1039	-1587	-341	-1874	2023-05-10T17:49:40.463	2
6184	-404	-4056	1755	-1059	292	-2099	2023-05-10T17:49:40.509	2
6185	74	-3137	941	-432	734	-1580	2023-05-10T17:49:40.689	2
6186	-40	-2834	504	-1540	1146	-1299	2023-05-10T17:49:40.694	2

6187 rows x 8 columns

Рис. 16 Одна из версий финального набора данных для решения задачи о классификации активности

Для получения данных о фазах сборщик данных использовал наше устройства и устройство Xiaomi, которое выводило временные диапазоны различных фаз сна. Когда данных накопилось достаточное количество,



данные, полученные с помощью нашего устройства были размечены метками соответствующих фаз сна:

- Быстрый сон - 0;
- Медленный сон - 1;
- Бодрствование - 2.

### 3.4.2. Анализ набора данных

Итоговый набор данных для классификации активности состоит из 6187 показаний данных датчика, собранных с акселерометра и гироскопа с интервалом 80 мс. Описание столбцов:

**acceleration\_x, acceleration\_y, acceleration\_z** - показатели акселерометра по соответствующей оси;

**gyro\_x, gyro\_y, gyro\_z** - показатели гироскопа по соответствующей оси.

**time** - время, в которое было получено показание данных датчика;

**target** – метка активности, *целевая переменная*:

- Отсутствие активности - 0
- Шаги - 1
- Бег - 2

Построим корреляционную матрицу для нашего набора данных:

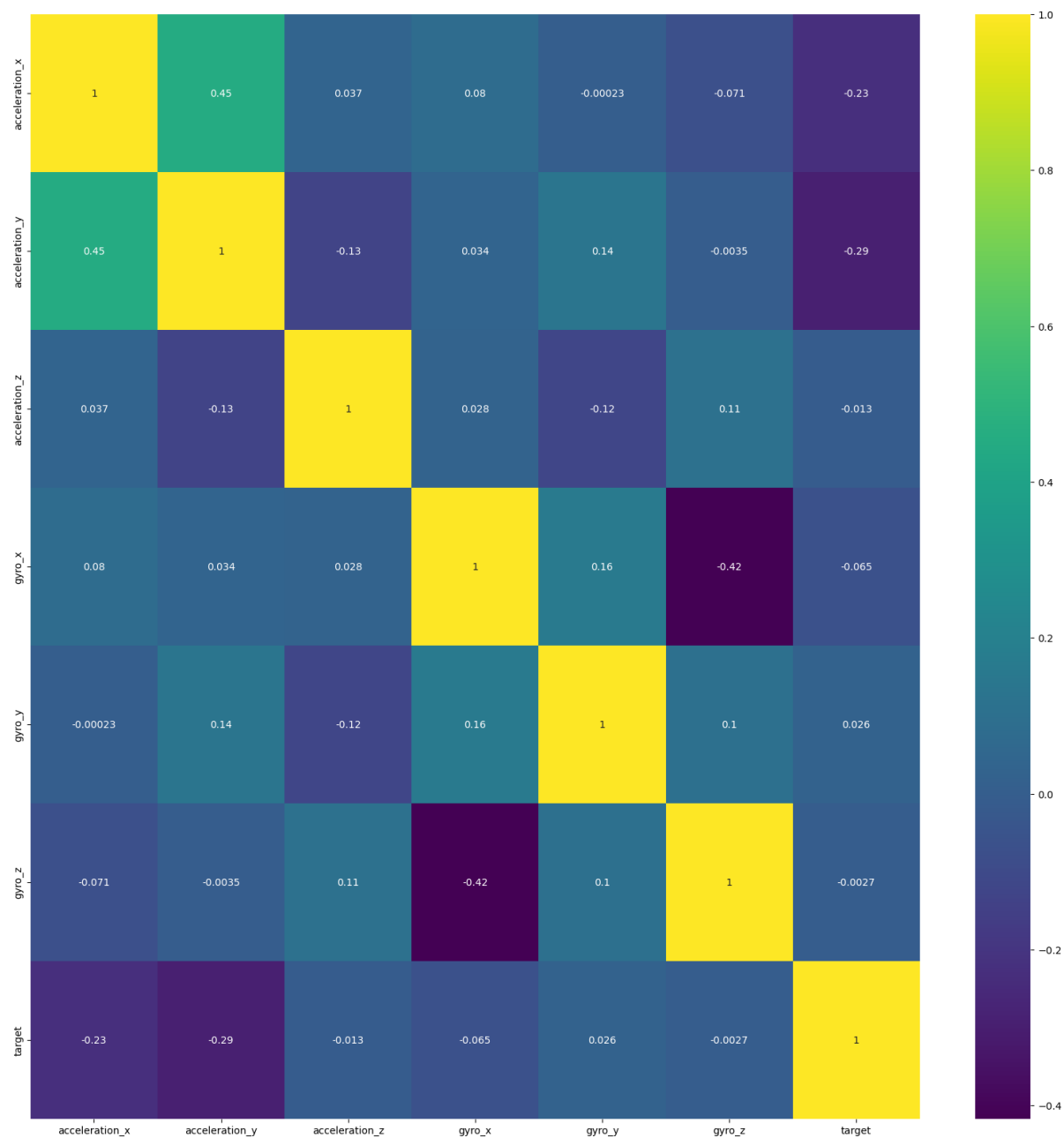


Рис. 17 Корреляционная матрица набора данных для классификации активности

На графике мы видим умеренную корреляцию **acceleration\_x** и **acceleration\_y** с целевой переменной.

Посмотрим на распределения оставшихся столбцов:

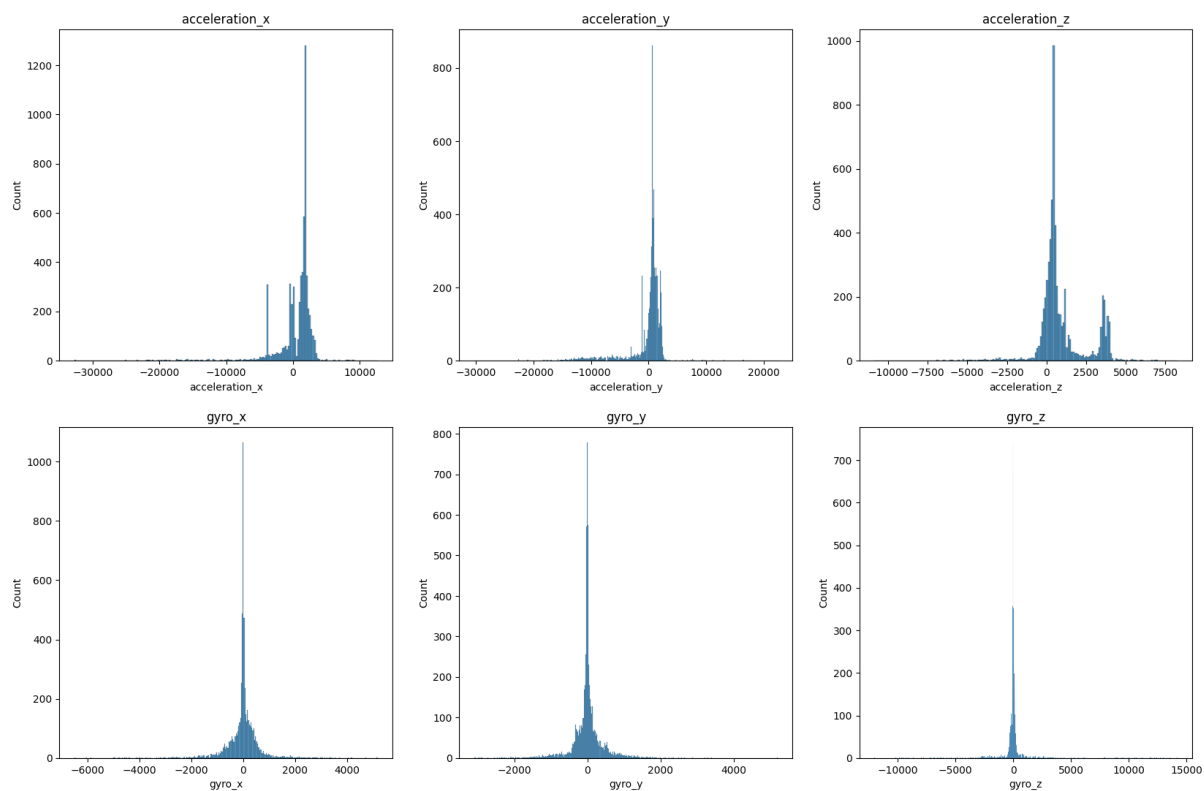


Рис. 18 Распределения показаний акселерометра и гироскопа

Все 6 распределений похожи на дельтаобразную функцию.

Пропущенных значений в наборе данных выявлено не было.

Итоговый набор данных для классификации фаз сна состоит из 3900000 показаний данных датчика, собранных с акселерометра и гироскопа с интервалом 80 мс.

Описание столбцов:

**acceleration\_x, acceleration\_y, acceleration\_z** - показатели акселерометра по соответствующей оси;

**gyro\_x, gyro\_y, gyro\_z** - показатели гироскопа по соответствующей оси.

**time** - время, в которое было получено показание данных датчика;

**target** – метка активности, *целевая переменная*:

- Быстрый сон - 0
- Медленный сон - 1
- Бодрствование - 3

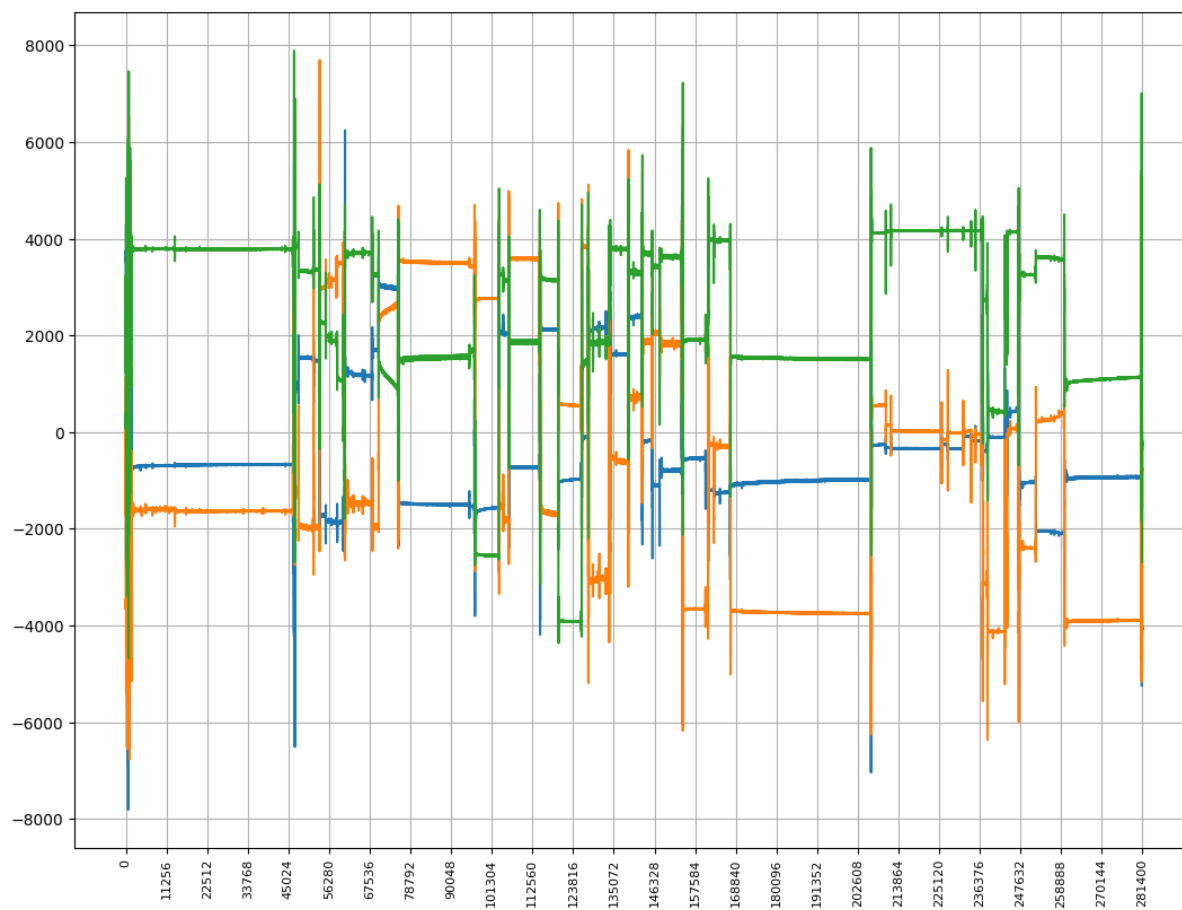


Рис. 19 Визуализация показаний гироскопа для одной ночи

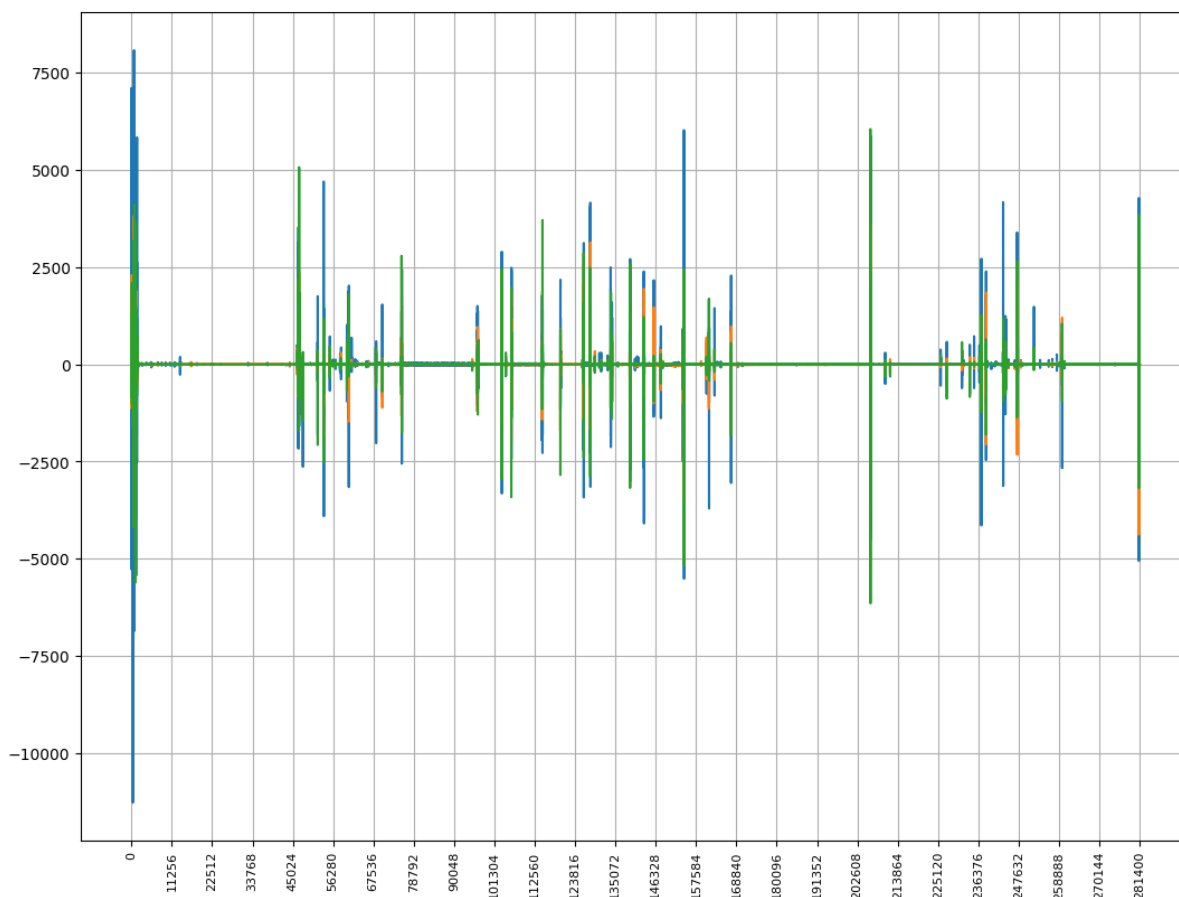


Рис. 20 Визуализация показаний акселерометра для одной ночи

### 3.4.3. Создание модели CatboostClassifier и подбор ее гиперпараметров с помощью optuna

В качестве первой модели была выбрана модель CatBoostClassifier из библиотеки CatBoost. Для подбора её гиперпараметров использовалась библиотека optuna. Некую теоретическую часть о работе этих инструментов вы можете прочитать в разделе 2.

Предварительно данные были разбиты на тренировочную и тестовую выборки (для окончательной оценки качества модели) в соотношении 70-30. Затем 25% тренировочной выборки было выделено на подбор гиперпараметров модели (валидационная выборка). Таким

образом, набор данных поделился на тренировочную-валидационную-тестовую выборки в соотношении 52,5-17,5-30.

После чего была определена сетка гиперпараметров, оптимальные значения которых будут подбираться с помощью библиотеки `optuna`.

Выбор именно этих параметров для оптимизации обусловлен рекомендациями по подбору гиперпараметров в документации Catboost [34], просмотр соревновательных решений на платформе Kaggle, а также личным опытом. Список параметров для подбора можно увидеть ниже:

- `learning_rate`
- `colsample_bylevel`
- `boosting_type`
- `bootstrap_type`
- `iterations`
- `l2_leaf_reg`
- `border_count`
- `random_strength`
- `od_type`
- `od_wait`
- `depth`
- `min_data_in_leaf`
- `leaf_estimation_iterations`
- `bagging_temperature`
- `subsample`

В процессе подбора гиперпараметров тренировочная выборка была поделена на 5 фолдов с помощью кросс-валидации. Использование валидационной выборки (`eval_set`) внутри кросс-валидации позволяет контролировать переобучение модели на каждом фолде.

Optuna подбирала гиперпараметры для максимизации метрики Accuracy,  $n\_trials = 100$ . После окончания подбора гиперпараметров функционал библиотеки позволяет вывести график оптимизации, который можно увидеть ниже:

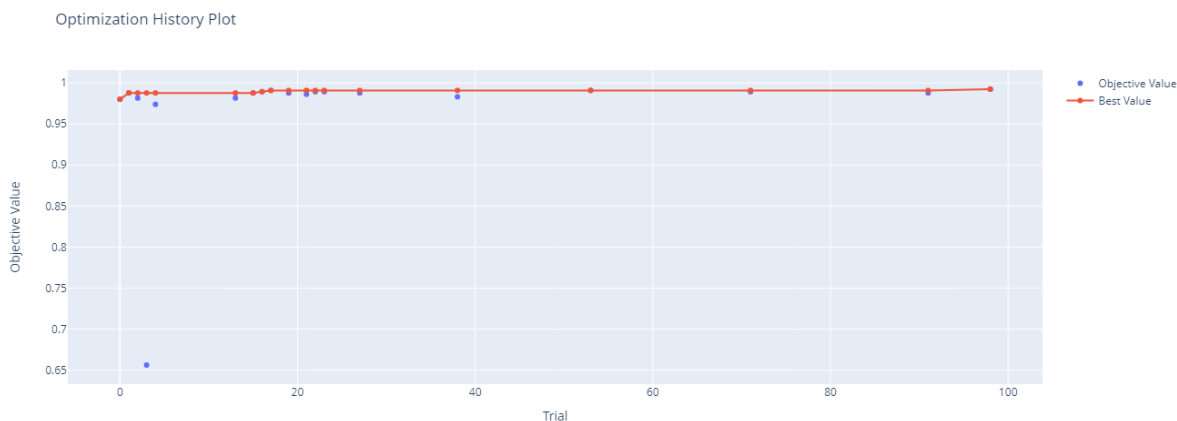


Рис. 21 График оптимизации модели CatboostClassifier

Также функционал библиотеки позволяет вывести график оценки важности гиперпараметров на основе завершенных испытаний в данном исследовании:

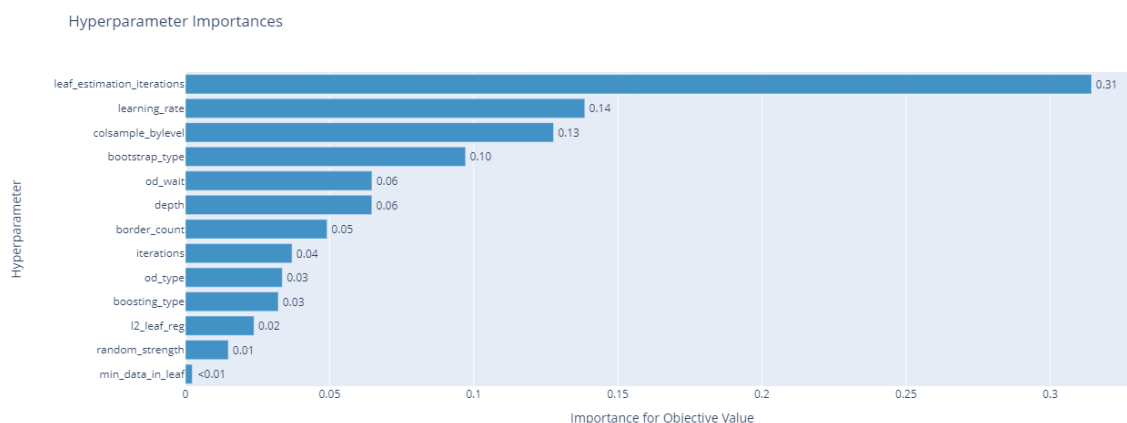




Рис. 22 Важность гиперпараметров CatboostClassifier в результате исследования

Кроме этого, можно вывести словарь с лучшими гиперпараметрами на основе проведенного исследования:

```
{'learning_rate': 0.09623757481665342,  
'colsample_bylevel': 0.09504648729531737,  
'boosting_type': 'Plain',  
'bootstrap_type': 'MVS',  
'iterations': 429,  
'l2_leaf_reg': 0.1094979180039778,  
'border_count': 123,  
'random_strength': 1.2186739347053031e-06,  
'od_type': 'IncToDec',  
'od_wait': 49,  
'depth': 12,  
'min_data_in_leaf': 7,  
'leaf_estimation_iterations': 8}
```

Рис. 23 Лучшие гиперпараметры на основе проведенного исследования

Точность на валидационном наборе данных составила 0.9925%

После этого, обучим модель CatboostClassifier с использованием «лучших» гиперпараметров и проверим ее качество на тестовой выборке. Accuracy составила 0.992. Проверим качество модели с помощью матрицы ошибок на тестовой выборке:

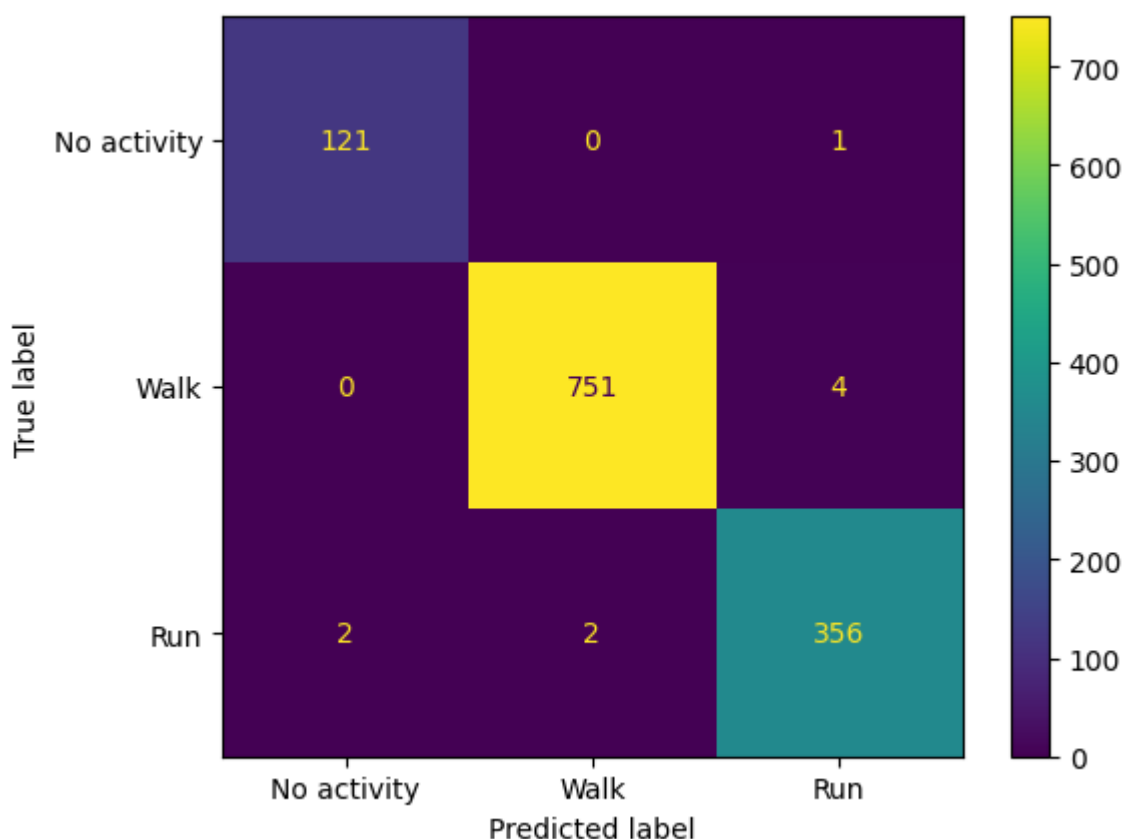


Рис. 24 Матрица ошибок обученной модели CatboostClassifier

Обучение модели классификации фаз сна проходило по тому же пайплайну, что и классификация активности. Точность на валидационном наборе данных составила 70%. Точность на тренировочном наборе данных составила 67%. Подобную точность можно объяснить тем, что в работе не используются специализированные приборы, а только акселерометр и гироскоп. На данном ресурсе можно посмотреть множество статей и примеров кода для детекции фаз сна [35]. Для формирования набора данных там использовалась ЭЭГ (электроэнцефалограмма), при этом точность варьируется от 80 до 86%. В связи с этим, мы считаем, что выжали из наших данных максимум.

### 3.4.4. Создание полносвязной нейронной сети

В данном разделе будет описано обучение полносвязной нейронной сети с помощью Pytorch. Некую необходимую теоретическую информацию о ней вы можете найти в главе 2.1.10. Гиперпараметры полносвязной нейронной сети в этом разделе подбирались эмпирическим путем. Об обучении полносвязной нейронной сети и подборе ее гиперпараметров с помощью optuna вы можете прочитать в следующем разделе.

Прежде всего разобьем данные на тренировочную, тестовую и валидационные выборки так же как и в разделе 3.4. «Создание модели CatboostClassifier и подбор ее гиперпараметров с помощью optuna». Кроме того, нам нужно произвести специфичные для PyTorch преобразования данных (например, тензорный вид). Для этого реализуем класс Dataset, который позволит представлять набор данных для обучения в необходимом для нас формате, а также обеспечивать доступ к отдельным элементам набора. Кроме того нам необходимо создать DataLoader для каждого набора данных, который будет использоваться для загрузки данных из класса Dataset и создания пакетов (batch) данных для обучения модели. Кроме того, он позволит эффективно загружать данные в память, перемешивать их и распределения на несколько процессов.

```
NN(  
    (linear1): Linear(in_features=6, out_features=25, bias=True)  
    (linear2): Linear(in_features=25, out_features=50, bias=True)  
    (dropout): Dropout(p=0.2, inplace=False)  
    (linear3): Linear(in_features=50, out_features=25, bias=True)  
    (linear4): Linear(in_features=25, out_features=3, bias=True)  
)
```

Рис. 25 Архитектура полносвязной нейронной сети

Слои активации в данной сети - relu.

Эмпирическим путем были подобраны следующие параметры:

```
criterion = nn.CrossEntropyLoss()
```

```
optimizer = torch.optim.Adam(NN.parameters(), lr = 0.001)
```

```
NUM_EPOCHS = 20
```

Обученная модель оценивалась метрикой Accuracy на валидационной и тестовой выборках и показала 99.5 и 99,19 соответственно. Кроме того была составлена матрица ошибок для оценки качества модели на тестовой выборке:



Рис. 26 Матрица ошибок для обученной полносвязной нейронной сети

### 3.4.5. Создание полносвязной нейронной сети и подбор ее гиперпараметров с помощью optuna

В данном разделе будет описано обучение полносвязной нейронной сети с помощью Pytorch и подбор ее гиперпараметров с помощью optuna. Разделение на тренировочный-валидационный и тестовый наборы проходила аналогично описанным в разделе 3.5. «Создание полносвязной нейронной сети». С помощью optuna планировалось подбирать следующие параметры:

- Количество скрытых слоев;
- Количество нейронов в скрытых слоях;
- Вероятность Dropout;
- Оптимизатор;
- learning rate.

Optuna подбирала гиперпараметры для максимизации метрики Accuracy, n\_trials = 100. После окончания подбора гиперпараметров функционал библиотеки позволяет вывести график оптимизации, который можно увидеть ниже:

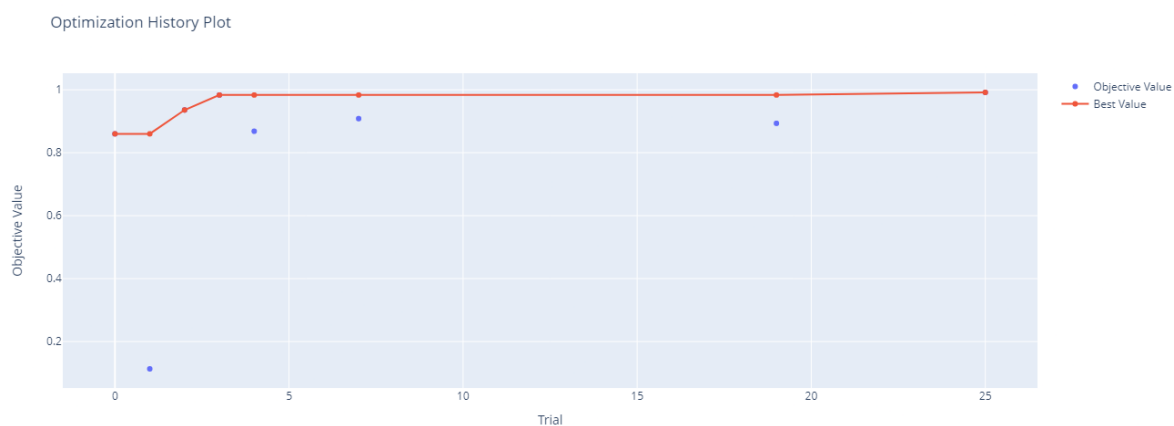


Рис. 27 График оптимизации модели полносвязной нейронной сети

Также функционал библиотеки позволяет вывести график оценки важности гиперпараметров на основе завершенных испытаний в данном исследовании:

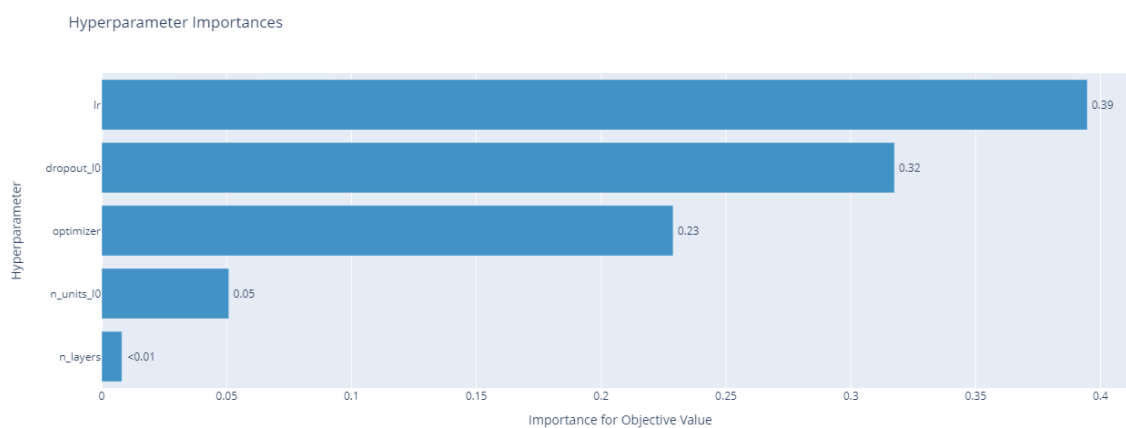


Рис. 28 Важность гиперпараметров полносвязной нейронной сети в результате исследования

Кроме этого, можно вывести словарь с лучшими гиперпараметрами на основе проведенного исследования:

```
{'n_layers': 1,  
'n_units_l0': 96,  
'dropout_l0': 0.1615111315593779,  
'optimizer': 'Adam',  
'lr': 0.0008268650397462426}
```

Рис. 29 Лучшие гиперпараметры на основе проведенного исследования

После этого, обучим модель полносвязной нейронной сети с использованием «лучших» гиперпараметров и проверим ее качество на тестовой выборке. Ассигасу составила 99,2 %. Проверим качество модели с помощью матрицы ошибок на тестовой выборке:



Рис. 30 Матрица ошибок для обученной полносвязной нейронной сети

### 3.4.6. Внедрение модели в мобильное приложение

В первую очередь необходимо было оптимизировать работу модели на устройствах с ограниченными ресурсами. Для этого было необходимо:

1. Конвертировать модель созданную в PyTorch в формат, который может быть использован на мобильных устройствах;
2. Применить к модели ряд оптимизаций, которые позволяют уменьшить ее размер и ускорить ее работу на мобильных устройствах;
3. Сохранить модель.

Для интеграции PyTorch Mobile в проект Android были добавлены зависимости в файл `'build.gradle'`.

Была создана папка `'assets'`, куда была сохранена экспортированная ранее модель.

Для использования модели ей необходимо передавать входные данные в корректном формате, после чего модель вернет выходное значение.



### 3.5. Примеры использования и тестирования

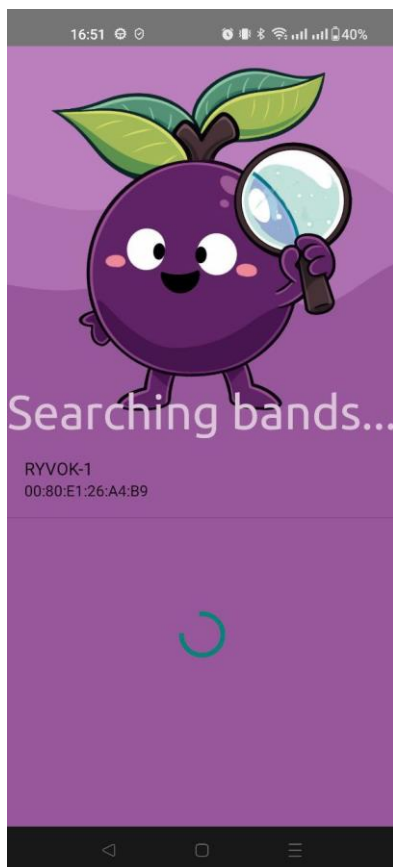


Рис. 31 Соединение с устройством по Bluetooth



Рис. 32 Определение фаз сна

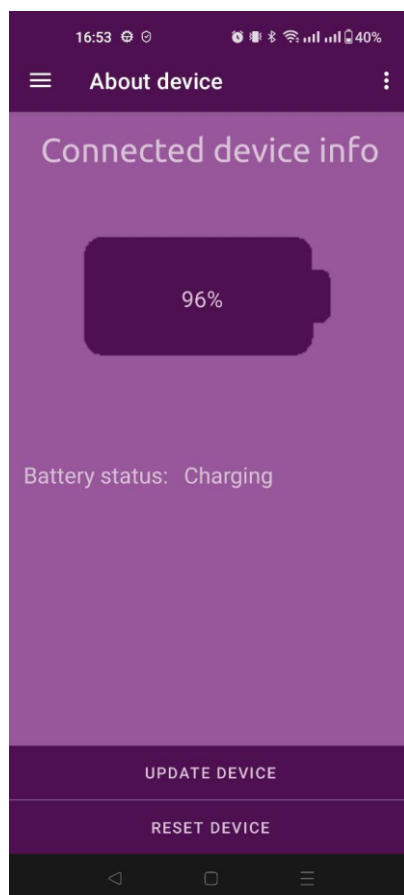


Рис. 33 Информация об устройстве

Подробнее про тестирование можно прочитать в приложении «Программа и методика испытаний».

### 3.6. Репозиторий

Весь код проекта можно найти по ссылке:

<https://github.com/DoubleDoo/PLUMPY-CW2023>

### 3.7. Описание структуры проекта

Структурно проект делится на следующие директории:

- / DevicePCB – папка со схемой аппаратной части
- / DeviceApp – папка с исходным кодом устройства
- / AndroidApp – папка с исходным кодом приложения компаньона
- / Modeling– папка с моделями машинного обучения
- / Docs – папка с документацией проекта

### **3.8. Выводы по главе**

В третьей главе были рассмотрены практические подходы, которые использовались для разработки программно-аппаратного комплекса.

## **Заключение**

В данной выпускной квалификационной работе был создан программно-аппаратный комплекс, который включает устройство и приложение-компаньон на базе Android. Для разработки использовались актуальные средства и решения, применяемые в разработке мобильного ПО и киберфизических устройств.

Первая глава описывает предметную область работы и демонстрирует существующие программно-аппаратные решения, что помогает определить направление проектирования и выбор инструментов.

Вторая глава посвящена выбранным технологическим решениям и обоснованию их использования в комплексной разработке.

Третья глава раскрывает особенности реализации программно-аппаратного комплекса, включая устройство, его аппаратную и программную составляющие, а также приложение-компаньон. Описывается архитектура проекта и тонкости реализации важных узлов взаимодействия между составляющими комплекса.

В процессе выполнения выпускной квалификационной работы были проведены исследования, изучены аналоги и выявлены основания для разработки проекта, а также определены конкурентные преимущества. Кроме того, были разработаны техническое задание и функциональные требования, спроектированы и реализованы устройство и приложение-компаньон, а также отлажен программный код и проведено тестирование проекта. Наконец, была разработана техническая документация в соответствии с ГОСТ.

В качестве дальнейшего развития проекта предполагается расширение функционала устройства, за счет уже внедренных аппаратных модулей NFC и барометра, после предполагается изготовление улучшенной платы, для уменьшения размеров устройства.

Для улучшения показателей моделей машинного обучения, планируется собрать больше данных для обучения, и внедрить в аппаратную часть датчик акселерометра и гироскопа с возможностью машинного обучения, для реализации обработки данных на устройстве, а не на приложении компаньоне.

## Список литературы

1. В.В. КОВЗОВ, В.К. ГУСАКОВ, А.В. ОСТРОВСКИЙ  
ФИЗИОЛОГИЯ СНА 2005.
2. ПОЛИСОМНОГРАФИЯ. [Электронный ресурс] // URL:  
<http://medsna.ru/o-sne/metody-issledovaniya> (дата обращения:  
12.02.23).
3. Фитнес-браслеты для бега: какие бывают и как выбрать  
[Электронный ресурс] // URL: <https://marathonec.ru/fitness-brasleti/>  
(Дата обращения 10.03.2023)
4. Лучшие фитнес-браслеты 2023 [Электронный ресурс] // URL:  
<https://www.fotosklad.ru/expert/articles/lucsie-fitness-braslety-2022/>  
(Дата обращения 10.03.2023)
5. STM32WB55CGU6 [Электронный ресурс] // URL:  
<https://www.st.com/en/microcontrollers-microprocessors/stm32wb55cg.html> (Дата обращения: 25.02.23)
6. Обзор архитектуры Bluetooth 5.0 [Электронный ресурс] // URL:  
<https://itechinfo.ru/content/bluetooth-50> (Дата обращения: 02.03.23).
7. X-CUBE-EEPRMA1 [Электронный ресурс] // URL:  
<https://www.st.com/en/embedded-software/x-cube-eeprma1.html>  
(Дата обращения: 20.04.23).
8. X-CUBE-MEMS1 [Электронный ресурс] // URL:  
<https://www.st.com/en/embedded-software/x-cube-mems1.html>  
(Дата обращения: 20.04.23).

9. STM32CubeWB [Электронный ресурс] // URL:  
[https://github.com/STMicroelectronics/STM32CubeWB/blob/master/Middlewares/ST/STM32\\_WPAN/stm32\\_wpan\\_common.h](https://github.com/STMicroelectronics/STM32CubeWB/blob/master/Middlewares/ST/STM32_WPAN/stm32_wpan_common.h)

(Дата обращения: 20.04.23).

10. Разработка решений на базе STM32WB55 для работы в беспроводных сетях [Электронный ресурс] // URL:  
<https://www.electronshtik.ru/news/show/13376?from=terraelectronica>

(Дата обращения: 18.03.23).

11. Что такое GATT и как он работает? [Электронный ресурс] // URL:  
<http://ru.feasywifi.com/info/what-is-gatt-and-how-does-it-work-78829953.html>

(Дата обращения: 12.02.23).

12. BLE под микроскопом (АТТы GATTы...) [Электронный ресурс] // URL: <https://habr.com/ru/articles/505078/>

(Дата обращения: 01.05.23).

13. STM32WB Firmware Upgrade Service [Электронный ресурс]: // URL:  
[https://wiki.st.com/stm32mcu/wiki/Connectivity:STM32WB\\_FUS](https://wiki.st.com/stm32mcu/wiki/Connectivity:STM32WB_FUS)

(Дата обращения: 01.05.23).

14. Python. Official website. [Электронный ресурс] // URL:  
<https://www.python.org/>

(Дата обращения 10.03.2023)



15. Google Colab. Official website. [Электронный ресурс] // URL:  
<https://colab.research.google.com/>  
(Дата обращения 10.03.2023)
16. NumPy. Official website. [Электронный ресурс] // URL:  
<https://numpy.org/>  
(Дата обращения 10.03.2023)
17. Pandas. Official website. [Электронный ресурс] // URL:  
<https://pandas.pydata.org/>  
(Дата обращения 10.03.2023)
18. Scikit-learn. Official website. [Электронный ресурс] // URL:  
<https://scikit-learn.org/>  
(Дата обращения 15.03.2023)
19. PyTorch. Official website. [Электронный ресурс] // URL:  
<https://pytorch.org/>  
(Дата обращения 11.04.2023)
20. PyTorch Mobile. Official website. [Электронный ресурс] // URL:  
<https://pytorch.org/mobile/home/>  
(Дата обращения 20.04.2023)
21. Optuna. Official website. [Электронный ресурс] // URL:  
<https://optuna.org/>  
(Дата обращения 15.03.2023)
22. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta and Masanori Koyama  
Optuna: A Next-generation Hyperparameter Optimization Framework. //  
KDD '19: Proceedings of the 25th ACM SIGKDD International  
Conference on Knowledge Discovery & Data Mining 2019 C. 2623–2631

23. Catboost. Official website. [Электронный ресурс] // URL:

<https://catboost.ai/en/docs/>

(Дата обращения 15.03.2023)

24. Gradient boosting. [Электронный ресурс] // Wikipedia URL:

[https://en.wikipedia.org/wiki/Gradient\\_boosting](https://en.wikipedia.org/wiki/Gradient_boosting)

(Дата обращения 20.03.2023)

25. Catboost. Benchmarks. [Электронный ресурс] // URL:

<https://catboost.ai/#benchmark%7C>

(Дата обращения 20.03.2023)

26. Perceptron [Электронный ресурс] // URL:

<https://en.wikipedia.org/wiki/Perceptron>

(Дата обращения 10.04.2023)

27. Dropout — метод решения проблемы переобучения в нейронных сетях [Электронный ресурс] // URL:

<https://habr.com/ru/companies/wunderfund/articles/330814/>

(Дата обращения 10.04.2023)

28. Neural Network Optimizers Made Simple: Core algorithms and why they are needed [Электронный ресурс] // URL:

<https://towardsdatascience.com/neural-network-optimizers-made-simple-core-algorithms-and-why-they-are-needed-7fd072cd2788>

(Дата обращения 10.04.2023)

29. Stochastic gradient descent [Электронный ресурс] // URL:

[https://en.wikipedia.org/wiki/Stochastic\\_gradient\\_descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent)

(Дата обращения 10.04.2023)

30. Understanding binary cross-entropy / log loss: a visual explanation

[Электронный ресурс] // URL:

<https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>

(Дата обращения 11.04.2023)

31. Confusion matrix [Электронный ресурс] // URL: [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)

[learn.org/stable/modules/generated/sklearn.metrics.confusion\\_matrix.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)

(Дата обращения 10.03.2023)

32. Cross-validation [Электронный ресурс] // URL: [https://scikit-](https://scikit-learn.org/stable/modules/cross_validation.html)

[learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)

(Дата обращения 10.03.2023)

33. Александр Шауэрман [Электронный ресурс] // URL:

<https://www.compel.ru/lib/134472>

(Дата обращения: 01.05.23).

34. Catboost. Parameter tuning. [Электронный ресурс] // URL:

<https://catboost.ai/en/docs/concepts/parameter-tuning>

(Дата обращения 15.03.2023)

35. Sleep Stage Detection. [Электронный ресурс] // URL:

<https://paperswithcode.com/task/sleep-stage-detection>

(Дата обращения 20.04.2023)

36. PhilJay/MPAndroidChart [Электронный ресурс] // URL:

<https://github.com/PhilJay/MPAndroidChart>

(Дата обращения: 01.04.23).

Дубина Д.О., Кожакин К.Г., Федотов Г.А., Программно-аппаратный комплекс отслеживания и анализа данных движений человека