

Минобрнауки России  
Юго-Западный государственный университет

Кафедра программной инженерии

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**  
**ПО ПРОГРАММЕ БАКАЛАВРИАТА**

**09.03.04 Программная инженерия**

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

**«Разработка программно-информационных систем»**

**Программное обеспечение для анализа и улучшения аудиозаписей**

(название темы)

**Дипломный проект**

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР

**И. О. Матвеев**

(подпись, дата)

(инициалы, фамилия)

Группа ПО-116

Руководитель ВКР

**И. Н. Ефремова**

(подпись, дата)

(инициалы, фамилия)

Нормоконтроль

**А. А. Чаплыгин**

(подпись, дата)

(инициалы, фамилия)

ВКР допущена к защите:

Заведующий кафедрой

**А. В. Малышев**

(подпись, дата)

(инициалы, фамилия)

Курск 2025 г.

**Минобрнауки России**  
**Юго-Западный государственный университет**

Кафедра программной инженерии

УТВЕРЖДАЮ:  
Заведующий кафедрой

---

(подпись, инициалы, фамилия)

«\_\_\_\_\_» 20\_\_\_\_ г.

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ  
РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА**

Студента Матвеев И. О., шифр 21-06-0160, группа ПО-11б

1. Тема «Программное обеспечение для анализа и улучшения аудиозаписей» утверждена приказом ректора ЮЗГУ от «04» апреля 2025 г. № 1696-с.
2. Срок предоставления работы к защите «9» июня 2025 г.
3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

- 1) захват и воспроизведение аудио в реальном времени;
- 2) загрузка и обработка аудиофайлов;
- 3) разделение сигнала на частотные полосы;
- 4) многополосная компрессия с индивидуальными настройками;
- 5) эквалайзация для коррекции тембра;
- 6) реверберация и шумоподавление;
- 7) оптимизация производительности (ЛТ, многопоточность);
- 8) управление состоянием и синхронизация модулей;
- 9) настройка параметров эффектов в реальном времени;
- 10) логирование и диагностика работы системы.

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для программной системы являются: аудиосигнал в виде цифрового потока, настройки эффектов, задаваемые пользователем (параметры эквалайзера, компрессора, реверберации и шумоподавление).

2) Выходными данными для программной системы являются: обработанный аудиосигнал, выводимый на устройство воспроизведения или сохраняемый в файл, возможность мониторинга и визуализации параметров обработки.

4. Содержание работы (по разделам):

4.1. Введение.

4.1. Анализ предметной области.

4.2. Техническое задание: основание для разработки, цель и назначение разработки, требования к программной системе, требования пользователя к интерфейсу.

4.3. Технический проект: общая характеристика решения задачи, обоснования выбора технологии, эффекты, архитектура программной системы.

4.4. Рабочий проект: спецификация классов программы, тестирование системы.

4.5. Заключение.

4.6. Список использованных источников.

5. Перечень графического материала:

Лист 1. Сведения о ВКРБ.

Лист 2. Цели и задачи разработки.

Лист 3. Диаграмма прецедентов.

Лист 4. Архитектура приложения.

Лист 5. Интерфейс программы.

Лист 6. Интерфейс вкладки компрессора и графиков.

Лист 7. Интерфейс вкладки реверберации и графиков.

Лист 8. Заключение.

Руководитель ВКР

И. Н. Ефремова

(подпись, дата)

(инициалы, фамилия)

Задание принял к исполнению

И. О. Матвеев

(подпись, дата)

(инициалы, фамилия)

## РЕФЕРАТ

Объем работы равен 104 страницам. Работа содержит 21 иллюстрацию, 0 таблиц, 20 библиографических источников и 8 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: аудиопроцессор, цифровая обработка сигналов, Python, многополосная компрессия, эквалайзация, реверберация, шумоподавление, многопоточность, JIT-компиляция, оптимизация, архитектура программной системы, аудиофильтры, обработка в реальном времени.

Объектом разработки является программная система — универсальный аудиопроцессор, предназначенный для обработки аудиосигналов в реальном времени с использованием современных методов цифровой обработки сигналов.

Целью выпускной квалификационной работы является создание программного аудиопроцессора, обеспечивающего высокое качество обработки звука и возможность гибкой настройки параметров для различных пользовательских задач.

В процессе разработки аудиопроцессора были реализованы основные модули для захвата и воспроизведения аудиосигнала, разделения сигнала на частотные полосы, многополосной компрессии, эквалайзации, реверберации и шумоподавления. Для повышения производительности использованы технологии JIT-компиляции и многопоточности. Архитектура системы построена на принципах модульности и масштабируемости, что позволяет легко расширять функциональность и адаптировать продукт под различные сценарии использования.

Разработанный аудиопроцессор протестирован и готов к использованию для обработки аудиосигналов в реальном времени.

## ABSTRACT

The volume of work is 104 pages. The work contains 21 illustration, 0 tables, 20 bibliographic sources and 8 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The layout of the site, including the connection of components, is presented in annex B.

Keywords: audio processor, digital signal processing, Python, multiband compression, equalization, reverberation, noise reduction, multithreading, JIT compilation, optimization, software system architecture, audio filters, real-time processing.

The object of the development is a software system, a universal audio processor designed to process audio signals in real time using modern digital signal processing methods.

The purpose of the final qualification is to create a software audio processor that provides high-quality audio processing and the ability to flexibly adjust parameters for various user tasks.

During the development of the audio processor, the main modules for capturing and reproducing the audio signal, dividing the signal into frequency bands, multiband compression, equalization, reverberation and noise reduction were implemented. JIT compilation and multithreading technologies are used to improve performance. The architecture of the system is based on the principles of modularity and scalability, which makes it easy to expand functionality and adapt the product to various use cases.

The developed audio processor has been tested and is ready for use for processing audio signals in real time.

## СОДЕРЖАНИЕ

|   |    |
|---|----|
| <b>ВВЕДЕНИЕ</b>   | 11 |
| 1 Анализ предметной области   | 14 |
| 1.1 Понятия и основная терминология                                 | 14 |
| 1.1.1 Эквалайзер  | 15 |
| 1.1.2 Многополосный динамический компрессор                         | 16 |
| 1.1.3 Реверберация  | 17 |
| 1.2 Оптимизация   | 18 |
| 1.3 ЛТ-компиляция   | 19 |
| 1.4 История создания и развития аудиопроцессоров                    | 20 |
| 1.5 Классификация и технические особенности аудиопроцессоров        | 22 |
| 1.6 Технические проблемы и перспективы развития аудиопроцессоров    | 24 |
| 2 Техническое задание   | 26 |
| 2.1 Основания для разработки  | 26 |
| 2.2 Цель и назначение разработки                                    | 26 |
| 2.3 Требования к программно-информационной системе                  | 26 |
| 2.3.1 Требования к данным программно-информационной системы         | 26 |
| 2.3.2 Функциональные требования к программно-информационной системе | 27 |
| 2.3.2.1 Шумоподавление  | 28 |
| 2.3.2.2 Эквалайзер  | 29 |
| 2.3.2.3 Компрессор  | 30 |
| 2.3.2.4 Реверберация  | 33 |
| 2.3.2.5 Визуализация  | 34 |
| 2.3.2.6 Управление  | 35 |
| 2.3.3 Моделирование вариантов использования                         | 36 |
| 2.3.3.1 Вариант использования «Загрузка аудиофайла»                 | 36 |
| 2.3.3.2 Вариант использования «Воспроизведение аудио»               | 37 |
| 2.3.3.3 Вариант использования «Остановка воспроизведения»           | 37 |

|   |    |
|---|----|
| 2.3.3.4 Вариант использования «Изменения позиции воспроизведения»     | 37 |
| 2.3.3.5 Вариант использования «Применения шумоподавления»             | 38 |
| 2.3.3.6 Вариант использования «Применения эквалайзера»                | 38 |
| 2.3.3.7 Вариант использования «Применения компрессора»                | 39 |
| 2.3.3.8 Вариант использования «Применения реверберации»               | 39 |
| 2.3.3.9 Вариант использования «Сохранение обработанного файла»        | 40 |
| 2.3.3.10 Вариант использования «Визуализация данных»                  | 40 |
| 2.4 Требования пользователя к интерфейсу приложения                   | 40 |
| 2.4.1 Нефункциональные требования к программно-информационной системе | 42 |
| 2.4.1.1 Требования к надёжности                                       | 42 |
| 2.4.1.2 Требования к безопасности                                     | 43 |
| 2.4.1.3 Требования к программному обеспечению                         | 43 |
| 2.4.1.4 Требования к аппаратному обеспечению                          | 44 |
| 2.4.2 Требования к оформлению документации                            | 44 |
| 3 Технический проект  | 45 |
| 3.1 Общая характеристика решения задачи                               | 45 |
| 3.2 Описание используемых технологий и языков программирования        | 45 |
| 3.2.1 Кроссплатформенная библиотека FFmpeg                            | 48 |
| 3.3 Эффекты   | 48 |
| 3.3.1 Устройство шумоподавления                                       | 48 |
| 3.3.2 Устройство эквалайзера  | 49 |
| 3.3.3 Устройство многополосного компрессора                           | 50 |
| 3.3.4 Устройство реверберации   | 52 |
| 3.4 Архитектура программно-информационной системы                     | 53 |
| 3.5 Архитектура приложения  | 56 |
| 3.5.1 Многопоточная обработка   | 57 |
| 3.6 Проект данных программно-информационной системы                   | 58 |
| 3.6.1 Описание сущностей графического интерфейса                      | 59 |
| 3.6.2 Описание сущностей логики процессора                            | 60 |

|  |        |
|--|--------|
| 3.6.3 Описание сущностей системного уровня   | 61     |
| 3.7 Проектирования пользовательского интерфейса  | 61     |
| 4 Рабочий проект   | 63     |
| 4.1 Классы, используемые при разработке приложения   | 63     |
| 4.1.1 Класс AudioProcessor   | 63     |
| 4.1.2 Класс AudioApp   | 64     |
| 4.1.3 Класс Knob   | 65     |
| 4.1.4 Класс App  | 65     |
| 4.2 Описание элементов интерфейса пользователя   | 65     |
| 4.3 Тестирование программно-информационной системы   | 69     |
| ЗАКЛЮЧЕНИЕ   | 77     |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ   | 78     |
| ПРИЛОЖЕНИЕ А Представление графического материала  | 82     |
| ПРИЛОЖЕНИЕ Б Фрагменты исходного кода программы  | 91     |
| На отдельных листах (CD-RW в прикрепленном конверте)   | 104    |
| Сведения о ВКРБ (Графический материал / Сведения о ВКРБ.png)   | Лист 1 |
| Цели и задачи разработки (Графический материал / Цели и задачи разработки.png)                                   | Лист 2 |
| Диаграмма прецедентов (Графический материал / Диаграмма прецедентов.png)   | Лист 3 |
| Архитектура приложения (Графический материал / Архитектура приложения.png)                                       | Лист 4 |
| Интерфейс программы (Графический материал / Интерфейс программы.png)   | Лист 5 |
| Интерфейс вкладки компрессора и графиков (Графический материал / Интерфейс вкладки компрессора и графиков.png)   | Лист 6 |
| Интерфейс вкладки реверберации и графиков (Графический материал / Интерфейс вкладки реверберации и графиков.png) | Лист 7 |
| Заключение (Графический материал / Заключение.png)   | Лист 8 |

## ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АЧХ – амплитудно-частотная характеристика.

DSP (Digital Signal Processing) – цифровая обработка сигналов.

JIT-компиляция – Just-In-Time компиляция (компиляция «точно в нужное время»).

БИХ-фильтр (IIR) – фильтр с бесконечной импульсной характеристической.

GIL (Global Interpreter Lock) – глобальная блокировка интерпретатора.

GUI (Graphical User Interface) – графический интерфейс пользователя.

ФНЧ – фильтр низких частот.

ФВЧ – фильтр высоких частот.

FFT (Fast Fourier Transform) – быстрое преобразование Фурье.

## ВВЕДЕНИЕ

В современную эпоху цифровых технологий качество звука становится одним из ключевых факторов успешной коммуникации, развлечений и творчества. Развитие аудиотехнологий сопровождается постоянным ростом требований к обработке аудиосигналов как в профессиональной, так и в бытовой сфере. В связи с этим возрастаёт актуальность создания эффективных программных решений, позволяющих улучшать, корректировать и адаптировать звук под различные задачи и условия.

Программные аудиопроцессоры занимают важное место в индустрии обработки звука, обеспечивая широкий спектр возможностей: от базовой фильтрации и эквалайзации до сложной динамической обработки, реверберации и подавления шумов. Современные программные комплексы должны не только обеспечивать высокое качество обработки, но и быть гибкими, масштабируемыми, работать в реальном времени и иметь удобный интерфейс для пользователя.

Целью данной дипломной работы является разработка программного аудиопроцессора на языке Python, способного выполнять многополосную обработку аудиосигнала в реальном времени с применением современных эффектов, таких как эквалайзация, компрессия, реверберация и шумоподавление. Особое внимание уделяется модульности архитектуры, оптимизации производительности и возможности гибкой настройки параметров обработки.

*Цель настоящей работы - разработка аудиопроцессора с графическим интерфейсом для эффективной обработки звука с поддержкой основных эффектов. Для достижения поставленной цели необходимо решить следующие задачи:*

- провести анализ предметной области и существующих решений;
- спроектировать архитектуру программно-информационной системы, обеспечивающую модульность, расширяемость и эффективное взаимодействие всех компонентов;

- реализовать модуль обработки и воспроизведения аудиосигнала с поддержкой работы в реальном времени, реализовать основные эффекты обработки звука;
- создание удобного и понятного интерфейса приложение с отображение графиков формы волны и АЧХ в реальном времени и в соответствии с обработанными данными;
- обеспечить оптимизацию вычислений и минимизацию задержек.

*Структура и объем работы.* Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 104 страницам.

*Во введении* сформулирована цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

*В первом разделе* проведён анализ предметной области, включающий основные понятия цифровой обработки звука, историю развития аудиопроцессоров, их классификацию, технические проблемы и перспективы развития.

*В втором разделе* представлено техническое задание, включающее основания для разработки, цели, требования к системе, функциональные и нефункциональные требования, а также моделирование вариантов использования.

*В третьем разделе* описан технический проект, включающий общую характеристику решения, используемые технологии, устройство эффектов (шумоподавление, эквалайзер, компрессор, реверберация), архитектуру системы и проектирование пользовательского интерфейса.

*В четвертом разделе* приведён рабочий проект, содержащий описание классов приложения, элементов интерфейса и результаты тестирования программной системы.

В заключении излагаются основные результаты работы, полученные в ходе разработки.

В приложении А представлен графический материал. В приложении Б представлены фрагменты исходного кода.

# 1 Анализ предметной области

## 1.1 Понятия и основная терминология

Аудиопроцессор - это электронное устройство или программный комплекс, предназначенный для обработки и управления звуковыми сигналами с целью улучшения качества звучания и адаптации аудиосистемы под конкретные условия. Мощный инструмент для обработки, анализа и улучшения звука.

Аудиопроцессор полезен как звукорежиссёрам, так и обычным пользователям, которые работают со звуком. Он включает в себя:

- модификацию аудио (изменять громкость, регулировать частоты);
- добавление эффектов (реверберация);
- анализирование звука (визуализация частот и формы волны);
- оптимизирование записи (компрессия, нормализация).

Преимущества аудиопроцессоров:

1. Гибкость обработки: возможность применять разные эффекты (эквалайзер, компрессор, ревёрб) в любом порядке.
2. Режим реального времени: обработка звука без задержек.
3. Визуализация звука: осциллограммы, спектрограммы и другие графики помогают анализировать аудио.
4. Доступность: программные аудиопроцессоры не требуют дорогого оборудования.

Недостатки аудиопроцессоров:

1. Задержка: в режиме реального времени возможны задержки из-за сложных вычислений.
2. Требовательность к ресурсам: некоторые эффекты (например, реверберация с большим буфером) нагружают процессор.
3. Качество зависит от алгоритмов: дешевые или плохо настроенные процессоры могут ухудшать звук (артефакты, искажения).
4. Ограниченная совместимость: некоторые форматы аудио могут не поддерживаться, например, редкие кодексы.

5. Сложность настройки: для тонкой настройки эффектов нужен опыт, например, подбор параметров компрессора.

### 1.1.1 Эквалайзер

Эквалайзер (англ. Equalizer, EQ) – это аудиопроцессор, который регулирует амплитуду (громкость) звука в разных частотных диапазонах. Он используется для: коррекции тонального баланса, компенсации акустики помещения, творческой обработки. Эквалайзер работает на принципе частотной фильтрации звукового сигнала. Звук как физическое явление представляет собой колебания воздуха с определенными частотными характеристиками. Человеческое ухо воспринимает частоты в диапазоне 20 Гц - 20 кГц, и эквалайзер позволяет управлять амплитудой этих частотных составляющих.

Основные физические параметры: частота в Герцах, амплитуда и фаза.

Эквалайзер состоит из: Полос (bands) частотные диапазоны, которые можно регулировать (низкие, средние и высокие); фильтры – электронные или цифровые схемы, изменяющие уровень определённых частот; регуляторы усиления (Gain) – позволяют увеличивать или уменьшать громкость в выбранной полосе; частота среза (Cutoff Frequency) – граница, на которой фильтр начинает действовать; добротность (Q-factor) – ширина полосы воздействия.

Принцип работы эквалайзера:

- ФНЧ (фильтр низких частот) – пропускает только низкие частоты;
- ПФ (полосовой фильтр) – выделяет средние частоты;
- ФВЧ (фильтр высоких частот) – пропускает только высокие частоты.

Каждый диапазон частот обрабатывается с помощью: усиления – увеличение громкости выбранной частоты, ослабление – уменьшение громкости.

После обработки всех полос сигналы суммируются, и на выходе получается модифицированный звук.

Эквалайзер - мощный инструмент, требующий понимания как технических аспектов, так и особенностей слухового восприятия. Грамотное ис-

пользование позволяет значительно улучшить качество звучания, в то время как неправильное применение может ухудшить звук.

### 1.1.2 Многополосный динамический компрессор

Многополосный динамический компрессор — это аудиоэффект, который разделяет входящий звуковой сигнал на несколько частотных полос с помощью кроссоверных фильтров и применяет к каждой полосе отдельное сжатие динамического диапазона с независимыми настройками (порог, соотношение, атака, релиз и усиление). В результате компрессия воздействует не на весь спектр целиком, а избирательно на определённые частотные диапазоны, что позволяет точнее контролировать динамику и сохранить естественность звучания. Такой компрессор широко используется в микшировании и мастеринге для более гибкой и прозрачной обработки аудиосигнала.

Компрессор состоит из:

1. Частотные диапазоны (полосы) — входящий аудиосигнал разделяется на несколько отдельных частотных полос с помощью кроссоверных фильтров. Каждая полоса охватывает определённый участок частотного спектра и обрабатывается отдельно.
2. Кроссоверные фильтры — фильтры, которые разделяют сигнал на полосы, задавая точки пересечения (кроссоверы) между ними. Они обеспечивают плавное разделение спектра, чтобы минимизировать артефакты на границах полос.
3. Отдельные компрессоры для каждой полосы — каждая полоса имеет собственный компрессор с независимыми настройками: порог (threshold), коэффициент сжатия (ratio), время атаки (attack), время релиза (release), усиление компенсации (makeup gain).
4. Параллельная обработка — компрессия каждой полосы происходит параллельно, после чего полосы суммируются обратно в один выходной сигнал.

5. Регуляторы параметров — для каждой полосы можно настраивать параметры компрессии, а также точки кроссоверов, что позволяет гибко управлять динамикой в разных частотных диапазонах.

Эти параметры настраиваются отдельно для каждой полосы в много-полосном компрессоре, что позволяет гибко управлять динамикой в разных частотных диапазонах и добиваться высокого качества звука.

Таким образом, многополосный компрессор — это совокупность нескольких компрессоров, каждый из которых работает на своей частотной полосе, разделённой кроссоверами, что даёт более точный и гибкий контроль динамического диапазона по всему спектру звука

### 1.1.3 Реверберация

Реверберация — это акустическое явление, возникающее при много-кратных отражениях звуковых волн от поверхностей помещения. В отличие от эха, отдельные отражения при реверберации сливаются в непрерывный затухающий звуковой "хвост".

Принципы работы реверберации:

1. Временная иерархия: прямой звук -> ранние отражения -> поздние отражения -> диффузный хвост. Каждая фаза формирует определённые аспекты пространственного восприятия.

2. Частотная зависимость: высокочастотные компоненты затухают быстрее низкочастотных. Материалы поверхностей влияют на спектральный баланс отражений.

3. Плотности отражений: количество отражений растёт экспоненциально со временем. Достигение критической плотности формирует диффузное поле.

4. Пространственной дисперсии: отражения приходят со всех направлений. Корреляция между каналами уменьшается со временем.

5. Decay Time (RT60) - время затухания энергии на 60 дБ.

6. Dry/Wet Mix - баланс прямого и обработанного сигнала.

## 1.2 Оптимизация

Оптимизация кода — это процесс модификации программы для повышения её эффективности по одному или нескольким параметрам: скорости выполнения, потреблению памяти, энергоэффективности или отзывчивости в реальном времени. В контексте аудиообработки оптимизация играет ключевую роль, поскольку позволяет обрабатывать звуковые сигналы с минимальной задержкой, что критически важно для профессиональных аудиоприложений.

Методы оптимизации:

1. JIT-компиляция (Numba). Just-In-Time компиляция преобразует Python-функции в машинный код во время выполнения, обходя интерпретатор. Она позволяет: ускорять математические операции в несколько раз, поддержка SIMD-инструкций процессора, автоматическая оптимизация циклов и математики.
2. Векторизация операций (NumPy). Замена циклов Python на матричные операции NumPy, которые выполняются на предварительно скомпилированном С-коде. Она позволяет: исключить медленные циклы Python, использовать CPU-кэши, параллелизм на уровне инструкций.
3. Многопоточность (ThreadPoolExecutor). Распределение задач между ядрами CPU с помощью пула потоков. Она позволяет: загружать все ядра процессора, создать отзывчивый интерфейс во время обработки, распараллелить независимые задачи.
4. Кэширование фильтров. Сохранение предвычисленных коэффициентов цифровых фильтров. Это позволяет: исключить повторные расчёты БИХ-фильтров, снизить нагрузку при изменении параметров, быстрое переключение пресетов.
5. Буферизация данных (Queue). Организация конвейера обработки через очереди фиксированного размера. Она позволяет: защищать от переполнения память, минимизирует блокировки.

6. Алгоритмические оптимизации. Выбор эффективных алгоритмов обработки сигналов. Она позволяет: снизить вычислительную сложность, минимизировать задержки, выполнить качественную обработку.

Оптимизация — это баланс между скоростью, потреблением памяти и читаемостью кода. В реальных проектах часто комбинируют несколько методов.

### 1.3 ЛТ-компиляция

ЛТ-компиляция (Just-In-Time компиляция) — это технология динамической компиляции кода во время выполнения программы, которая сочетает в себе преимущества интерпретируемых и компилируемых языков. В отличие от традиционной компиляции (АОТ — Ahead-Of-Time), когда весь код преобразуется в машинные инструкции до запуска программы, ЛТ-компилятор переводит фрагменты кода (например, часто выполняемые функции или циклы) в оптимизированный машинный код непосредственно в процессе работы приложения. Это позволяет адаптироваться к текущим условиям выполнения и применять специфичные для конкретной ситуации оптимизации.

Назначение ЛТ-компиляции:

1. Ускорение выполнения кода: интерпретируемые языки (например, Python, JavaScript) выполняются медленно, так как каждая инструкция обрабатывается построчно. ЛТ-компиляция преобразует участки кода, которые выполняются многократно, в машинный код, что значительно ускоряет их работу. Например в аудиопроцессоре, реализованном в моей работе, ЛТ-компиляция применяется для функций обработки звука (компрессии, реверберации), что позволяет обрабатывать аудиопоток в реальном времени без задержек.

2. Оптимизация под конкретное оборудование: ЛТ-компилятор может анализировать характеристики процессора и генерировать код, максимально эффективно использующий ресурсы системы.

3. Гибкость и портативность: программы на языках с ЛТ (Java, C#, Python с Numba) могут работать на разных платформах без перекомпиляции,

так как окончательная оптимизация происходит уже на устройстве пользователя.

4. Снижение нагрузки на память: в отличие от АОТ-компиляции, где весь код заранее переводится в машинные инструкции (что может занимать много места), ЛТ компилирует только нужные в данный момент части программы.

ЛТ-компиляция — это мощный инструмент для ускорения программ без потери гибкости. Она особенно полезна в задачах, где критична производительность: научных расчетах, обработке мультимедиа, играх и веб-приложениях. В дипломной работе ЛТ позволил эффективно обрабатывать аудиопоток в реальном времени, что было бы невозможно при использовании стандартного интерпретируемого Python.

#### **1.4 История создания и развития аудиопроцессоров**

История аудиопроцессоров началась с первых попыток управления звуковыми сигналами в начале XX века, когда инженеры искали способы улучшения качества записей и передачи звука. Первые устройства обработки звука были чисто аналоговыми и основывались на пассивных и активных электронных компонентах — резисторах, конденсаторах, трансформаторах и лампах. Одним из первых аудиопроцессоров можно считать компрессор, разработанный в 1930-х годах для радиовещания, чтобы предотвратить перегрузку передатчиков. Эти ранние устройства использовали лампы и имели ограниченную функциональность, но заложили основы динамической обработки звука.

В 1950-х годах появились первые специализированные аналоговые процессоры, такие как эквалайзеры и ревербераторы. Например, EMT 140 — пластинчатый ревербератор, созданный в 1957 году, стал стандартом в студиях звукозаписи благодаря своему характерному теплому звучанию. В тот же период разрабатывались транзисторные компрессоры, такие как UREI 1176 (1967), который использовал полевые транзисторы (FET) для быстрого и агрессивного сжатия. Эти устройства были аналоговыми, но уже облада-

ли регулируемыми параметрами (атака, релиз, порог), что делало их универсальными инструментами в студии.

1970-е годы стали временем расцвета аналоговых процессоров. Появились параметрические эквалайзеры, позволяющие точно настраивать частоту, добротность и усиление (например, API 550). В этот же период были созданы VCA-компрессоры (Voltage Controlled Amplifier), такие как dbx 160, которые предлагали более точное управление динамикой. Также развивались аналоговые задержки (например, Roland RE-201 Space Echo), использующие магнитную ленту для создания эффектов эха и повторов.

Переломным моментом стало появление цифровых аудиопроцессоров в конце 1970-х – начале 1980-х. Первые цифровые ревербераторы, такие как Lexicon 224 (1978), использовали алгоритмы на основе линий задержки с обратной связью (алгоритм Шрёдера) и позволяли имитировать акустику разных помещений. В 1980-х цифровая обработка звука стала массовой благодаря развитию микропроцессоров. Появились мультиэффект-процессоры (например, Eventide H3000), которые объединяли реверберацию, задержку, хорус и другие эффекты в одном устройстве.

1990-е годы ознаменовались переходом к программной обработке звука (DSP). С появлением мощных компьютеров и плагинов (таких как Waves, TC Electronic) аудиопроцессоры стали виртуальными. Это позволило использовать сверточную реверберацию (Convolution Reverb), которая воспроизводит импульсные характеристики реальных помещений с высокой точностью. Также развивались алгоритмические ревербераторы, такие как Altiverb и ValhallaDSP, которые сочетали физическое моделирование с гибкостью цифровых методов.

В 2000-х и 2010-х годах аудиопроцессоры стали еще более сложными благодаря искусственному интеллекту и машинному обучению. Например, iZotope RX использует спектральный анализ для восстановления поврежденных записей, а Neural DSP применяет нейросети для эмуляции гитарных усилителей и эффектов. Современные процессоры, такие как Universal Audio UAD и Plugin Alliance, сочетают аппаратное ускорение с продвинуты-

ми алгоритмами, обеспечивая минимальные задержки и студийное качество в реальном времени.

Сегодня аудиопроцессоры продолжают развиваться в сторону иммерсивного звука (Dolby Atmos, Ambisonics) и облачных технологий, позволяющих обрабатывать звук удаленно. История аудиопроцессоров – это эволюция от простых аналоговых схем к сложным цифровым системам, которые могут точно моделировать физические процессы и создавать принципиально новые звуковые эффекты.

## **1.5 Классификация и технические особенности аудиопроцессоров**

Классификация аудиопроцессоров по сфере применения охватывает несколько основных категорий устройств. Студийные процессоры предназначены для профессиональной звукозаписи и сведения, они отличаются высокой точностью обработки с разрядностью 24-32 бита и частотой дискретизации до 192 кГц, минимальным уровнем шумов и искажений, расширенным набором параметров включая многополосную обработку и side-chain, а также поддержкой профессиональных интерфейсов таких как Dante и MADI. Концертные и лайв-процессоры оптимизированы для работы в реальном времени и характеризуются сверхнизкой задержкой менее 2 мс, упрощенным управлением с предустановками, повышенной надежностью конструкции и специализированными функциями вроде подавления обратной связи и автоматического микширования. Потребительские решения ориентированы на массовый рынок и предлагают компактные размеры, портативность, упрощенные алгоритмы обработки и интеграцию с мобильными устройствами через Bluetooth и USB при доступной цене. Встраиваемые системы представляют собой специализированные решения для бытовой техники с минимальным энергопотреблением, аппаратной оптимизацией под конкретные задачи и автоматическими режимами работы, применяемые в телевизорах, автомобильных аудиосистемах и умных колонках.

С технической точки зрения цифровые процессоры реализуются на различных платформах. DSP (цифровые сигнальные процессоры) исполь-

зуют специализированные чипы вроде Analog Devices SHARC или Texas Instruments C6000, оптимизированные для потоковой обработки с параллельным выполнением операций. FPGA (программируемые логические матрицы) обеспечивают сверхнизкую задержку на уровне тактов процессора и гибкость в реализации нестандартных алгоритмов. Ключевыми характеристиками цифровых процессоров являются производительность в GMAC/s, разрядность обработки 32 или 64 бита, поддержка плавающей точки и энергоэффективность.

Методы обработки сигналов в аудиопроцессорах включают линейные и нелинейные преобразования. Линейные методы охватывают частотную коррекцию с использованием БИХ и КИХ-фильтров, линейную свертку для reverberации и пространственной обработки, а также корреляционный анализ. Нелинейные методы включают динамическую обработку типа компрессии и лимитирования, тональные преобразования вроде дисторшна и сатурации, а также амплитудную модуляцию. Адаптивные алгоритмы позволяют автоматически подстраивать параметры для шумоподавления и устранения обратной связи, используя статистический анализ сигнала и системы с обратной связью. Современные подходы включают машинное обучение с нейросетевыми моделями для восстановления аудио, разделения источников и интеллектуального сведения, а также GAN-архитектуры для синтеза эффектов. Биоинспирированные методы моделируют слуховую систему человека с использованием коклеарных фильтров и психоакустической оптимизации.

Каждый метод обработки имеет свои преимущества и ограничения. Линейные методы обеспечивают предсказуемость и стабильность, но имеют ограниченный диапазон задач. Нелинейные методы предоставляют широкие творческие возможности, но могут вызывать артефакты обработки. Адаптивные алгоритмы автоматизируют рутинные операции, но требуют времени на адаптацию. Машинное обучение открывает качественно новые возможности, но крайне требовательно к вычислительным ресурсам. Выбор конкретного метода и платформы зависит от требований к качеству обработки, работе в реальном времени и экономической целесообразности, что в совокупно-

сти определяет современное состояние и перспективы развития технологий аудиообработки.

## **1.6 Технические проблемы и перспективы развития аудиопроцессоров**

При разработке и эксплуатации аудиопроцессоров специалисты сталкиваются с рядом технических сложностей и ограничений. Одной из ключевых проблем являются фазовые искажения, возникающие при обработке сигнала различными фильтрами и эффектами, что может приводить к ухудшению стереокартинки и неестественному звучанию. Не менее важной проблемой выступают артефакты обработки - нежелательные звуковые искажения, проявляющиеся в виде цифровых щелчков, металлического призыва или неестественного окрашивания тембра. Эти артефакты особенно заметны при агрессивной обработке или каскадном включении нескольких эффектов.

Вычислительная сложность современных алгоритмов обработки звука создает серьезные требования к аппаратным ресурсам. Сложные эффекты вроде сверточной реверберации или нейросетевой обработки требуют значительной процессорной мощности, что ограничивает их применение в реальном времени. Проблема задержки (latency) особенно критична для лайв-обработки и мониторинга, где даже небольшие задержки в 10-20 мс могут нарушить восприятие музыкантами своего исполнения.

Вопросы совместимости форматов остаются актуальными в условиях многообразия аудиостандартов. Проблемы возникают при взаимодействии оборудования разных производителей, использовании устаревших протоколов или при попытках интеграции профессиональных и потребительских решений. Особенно остро это проявляется при работе с многоканальными форматами и метаданными.

Перспективные направления развития аудиопроцессоров включают несколько многообещающих технологий. Квантовые методы обработки теоретически могут решить проблему вычислительной сложности для определенных классов алгоритмов, хотя практические реализации пока находят-

ся в стадии исследований. Иммерсивный звук (3D-аудио) становится новым стандартом для кинопроизводства и игровой индустрии, требуя разработки специализированных процессоров для работы с объектно-ориентированным звуком в форматах Dolby Atmos и Ambisonics.

Адаптивные системы на основе искусственного интеллекта позволяют автоматически подстраивать параметры обработки под конкретный материал и акустические условия. Интеграция с VR/AR открывает новые возможности для создания полностью интерактивных звуковых ландшафтов, где обработка происходит в реальном времени с учетом действий пользователя. Экологичные решения направлены на снижение энергопотребления аудиооборудования без потери качества обработки.

Аспекты безопасности и надежности включают несколько важных направлений. Защита от перегрузок предотвращает повреждение оборудования при работе с мощными сигналами. Системы диагностики позволяют оперативно выявлять неисправности и отклонения в работе процессоров. Резервирование критически важных каналов обеспечивает бесперебойную работу в профессиональных приложениях. Защита от электромагнитных помех остается актуальной проблемой, особенно для аналоговых трактов и высокочувствительных микрофонных входов.

Современные разработки в области аудиопроцессоров направлены на преодоление существующих ограничений через внедрение новых алгоритмов и аппаратных архитектур. Особое внимание уделяется созданию интеллектуальных систем, способных адаптироваться к изменяющимся условиям работы, сохраняя при этом высокое качество звучания и надежность. Параллельно ведется работа над упрощением интерфейсов и снижением энергопотребления, что делает профессиональные технологии обработки звука доступными для более широкого круга пользователей.

## **2 Техническое задание**

### **2.1 Основания для разработки**

Основанием для разработки является задание на выпускную квалификационную работу бакалавра "Программное обеспечение для анализа и улучшения аудиозаписей".

### **2.2 Цель и назначение разработки**

Создание удобного и функционального инструмента для обработки аудиофайлов с возможностью применения различных звуковых эффектов в реальном времени, визуализации аудиоданных и сохранения результатов обработки. Назначение программы - предоставить пользователям простой в использовании, но мощный инструмент для базовой звуковой обработки, включающий эквалайзер, компрессор и ревербератор, с интуитивно понятным графическим интерфейсом и визуальной обратной связью. Код реализует многопоточную обработку аудио для минимизации задержек, поддерживает загрузку различных аудиоформатов, обеспечивает плавную визуализацию формы волны и частотного спектра, а также предлагает оптимизированные алгоритмы обработки звука. Приложение предназначено для музыкантов, звукорежиссеров и людей, которым требуется быстрый доступ к основным инструментам звуковой обработки без необходимости использования сложных профессиональных DAW.

### **2.3 Требования к программно-информационной системе**

#### **2.3.1 Требования к данным программно-информационной системы**

Входные данные:

- аудиофайлы форматов WAV, MP3, OGG, FLAC;
- параметры эффектов (эквалайзер, компрессор, реверберация), задаваемые пользователем.

Обрабатываемые данные:

- аудиосигнал в виде массива numpy.ndarray (форма (N, 1) для моно, (N, 2) для стерео);
- частота дискретизации (стандартно 44100 Гц, но поддерживаются другие значения);
- временные и спектральные данные для визуализации (форма волны, частотная характеристика).

Выходные данные:

- обработанный аудиофайл (экспорт в WAV, MP3);
- графики формы сигнала и АЧХ в реальном времени.

### **2.3.2 Функциональные требования к программно-информационной системе**

В разрабатываемой программе пользователь может:

- загружать аудиофайлы разных форматов (wav, mp3, ogg, flac);
- экспортировать результат, с применением выбранных им эффектов, в двух форматах: wav и mp3;
- воспроизводить, останавливать аудио, изменить позицию воспроизведения, узнать текущее время проигрывания;
- применять эффекты в режиме реального времени, включать и отключать их;
- применить шумоподавление и настроить его влияние на аудио;
- применить эквалайзер, настроить частоты;
- применить компрессор, настроить порог, коэффициент сжатия, время атаки и восстановления сигнала, усиление в выбранных диапазонах частот;
- применить реверберацию, настроить уровень эффекта, размер комнаты, затухание и чистый звук;
- видеть форму сигнала в режиме реального времени, посмотреть АЧХ аудио.

### 2.3.2.1 Шумоподавление

Реализовать модуль шумоподавления для аудиопроцессора, обеспечивающий эффективное снижение фонового шума в аудиозаписях с сохранением качества полезного сигнала. Модуль должен поддерживать несколько методов шумоподавления и предоставлять пользователю гибкие настройки степени обработки.

Методы шумоподавления:

- спектральное шумоподавление: использовать алгоритм спектрального вычитания шума на основе анализа спектра, автоматическое или ручное определение профиля шума;
- медианный фильтр: применять медианный фильтр для удаления импульсных шумов и выбросов;
- гауссово размытие: использовать гауссов фильтр для сглаживания сигнала и подавления высокочастотного шума.

Управление параметрами:

- пользователь должен иметь возможность выбрать метод шумоподавления из списка;
- регулировка степени шумоподавления (интенсивность обработки) в диапазоне от 0 (без обработки) до 100 (максимальное подавление);
- возможность включения/отключения шумоподавления.

Обработка сигнала:

- поддержка моно и стерео аудио;
- обработка должна выполняться в реальном времени или онлайн с минимальной задержкой;
- сохранение временной структуры и качественных характеристик полезного сигнала;
- предотвращение появления артефактов (искажений, «водянистости» звука).

### 2.3.2.2 Эквалайзер

Реализовать цифровой эквалайзер как часть аудиопроцессора для интерактивной обработки музыкальных и речевых аудиофайлов. Эквалайзер должен обеспечивать гибкую настройку амплитудно-частотной характеристики сигнала в реальном времени с возможностью визуализации.

Функциональные требования:

- эквалайзер должен состоять из 8 полос с фиксированными центральными частотами: 50 Гц, 150 Гц, 250 Гц, 500 Гц, 1 кГц, 3 кГц, 7 кГц, 15 кГц;
- для каждой полосы пользователь должен иметь возможность регулировать усиление/ослабление в диапазоне от -12 дБ до +12 дБ с шагом не менее 0.1 дБ;
- усиление/ослабление должно применяться к соответствующему частотному диапазону без существенных фазовых искажений и без возникновения паразитных резонансов;
- для каждой полосы должны быть реализованы цифровые фильтры второго порядка Баттервортса:

  - для самой низкой полосы (50 Гц) — lowpass фильтр;
  - для самой высокой полосы (15 кГц) — highpass фильтр;
  - для остальных полос — bandpass фильтры с шириной полосы  $\pm 20$  процентов от центральной частоты;
  - все фильтры должны рассчитываться динамически в зависимости от текущей частоты дискретизации аудиосигнала;
  - фильтры должны быть реализованы с помощью функций butter и lfilter из библиотеки scipy.signal;
  - для ускорения работы коэффициенты фильтров должны кэшироваться и пересчитываться только при изменении параметров;
  - итоговый сигнал должен формироваться как сумма результатов обработки всех полос;
  - эквалайзер должен поддерживать обработку как моно, так и стерео сигналов;

- необходимо реализовать функцию визуализации амплитудно-частотной характеристики (АЧХ) эквалайзера в реальном времени;
- все параметры эквалайзера должны быть доступны для программного и пользовательского управления (через API и/или GUI).

Эквалайзер должен быть реализован с приоритетом качества звука, стабильности работы и удобства пользовательского управления параметрами. Вся логика фильтрации и суммирования полос должна быть максимально прозрачна для дальнейшей доработки и интеграции с другими эффектами аудиопроцессора.

### 2.3.2.3 Компрессор

Реализовать цифровой многополосный компрессор для обработки аудиосигнала в составе аудиопроцессора. Компрессор должен обеспечивать независимую динамическую обработку трёх частотных диапазонов (low, mid, high) с индивидуальными настройками и возможностью визуализации изменений сигнала.

Компрессор должен обеспечивать обработку аудиосигнала в трёх частотных диапазонах с возможностью регулировки и настройки:

- low Band: 20–200 Гц;
- mid Band: 200–3000 Гц;
- high Band: 3000–20000 Гц.

Для каждого диапазона должны быть реализованы следующие параметры, доступные для настройки:

- threshold (порог срабатывания): от -60 дБ до 0 дБ с шагом не менее 0.1 дБ;
- ratio (степень сжатия): от 1:1 до 20:1.;
- attack (время атаки): от 0.1 до 500 мс;
- release (время восстановления): от 1 до 1000 мс;
- makeup gain (дополнительное усиление): от -12 до +12 дБ;
- soft knee (мягкость перехода): от 0 до 24 дБ;
- bypass (включение/выключение полосы).

Компрессор должен корректно работать с моно и стерео сигналами.

Для разделения сигнала на полосы должны использоваться цифровые фильтры (Butterworth или аналогичные) с динамическим расчётом коэффициентов в зависимости от частоты дискретизации.

Основные параметры компрессии:

1. Threshold (порог) измеряется в dB - определяет уровень входного сигнала в децибелах, при превышении которого компрессор начинает снижать громкость. Чем ниже порог, тем больше сигнал подвергается сжатию.

2. Ratio (коэффициент компрессии) – задает отношение входного превышения порога к выходному уровню сигнала. Например, при ratio 4:1, если входной сигнал превышает порог на 4 dB, то выходной уровень будет превышать порог только на 1 dB. Диапазон обычно от 1:1 (без сжатия) до 20:1 и выше. Чем выше ratio, тем сильнее сжатие.

3. Attack (атака) – время в миллисекундах, за которое компрессор начинает снижать уровень сигнала после превышения порога. Быстрая атака позволяет быстро подавлять пики, медленная — сохраняет естественную атаку звука, например, ударных.

4. Release (восстановление) – время в миллисекундах, за которое компрессор возвращается к нормальному усилению после того, как уровень сигнала опускается ниже порога. Более медленное восстановление создаёт плавное звучание, быстрое — более динамичное.

5. Knee (характеристика перехода) – определяет плавность начала сжатия вокруг порога. При значении 0 dB компрессия начинается резко (hard knee), при увеличении до 24 dB — плавно (soft knee), что делает переход менее заметным и звучание более естественным.

6. Make-up Gain (компенсационное усиление) – усиление выходного сигнала после сжатия для компенсации потерь громкости. Измеряется в децибелах. Позволяет вернуть общий уровень громкости на желаемый уровень.

Для оптимизации производительности и работы в реальном времени использованы:

- JIT-компиляция критических участков (Numba);

- векторизованные вычисления (NumPy);
- параллельная обработка полос;
- кэширование коэффициентов фильтров.

Формулы вычисления сигнала:

1. Threshold (порог срабатывания). Порог задаётся в децибелах (dB) и служит уровнем, при превышении которого начинается компрессия. Если амплитуда сигнала в линейной шкале —  $A$ , то уровень в децибелах:  $L = 20 \log_{10}(A)$ . Порог  $T$  задан в децибелах. Компрессор начинает сжимать сигнал, когда:  $L > T$ .

2. Ratio (степень компрессии). Определяет, насколько сильно уменьшается превышение уровня сигнала над порогом. Если входной уровень выше порога на  $\Delta L = L - T$ , то выходной уровень превышения будет:  $\Delta L_{out} = T + \Delta L/R$ , где  $R$  - коэффициент компрессии (ratio).

3. Attack и Release (время атаки и восстановления).  $a/r = e^{**(-1/t*fs)}$ , где  $a/r$  - атака/восстановление,  $t$  - время атаки или релиза в секундах,  $fs$  - частота дискретизации. Коэффициент усиления  $g[n]$  обновляется так:  $g[n] = a * g[n-1] + (1 - a) * gt[n]$ , если  $gt[n] < g[n-1]$ ,  $g[n] = r * g[n-1] + (1 - r) * gt[n]$ , если  $gt[n] \geq g[n-1]$ , где  $gt[n]$  - целевой коэффициент усиления вычисляемый по уровню сигнала и ratio.

4. Knee Width (ширина колена). Для входного уровня  $L$ , порога  $T$  и ширины колена  $W$ , вычисляем коэффициент усиления  $G$  в децибелах так:

- если  $L < T - W/2$ , то компрессия не применяется:  $G = 0$ ;
- если  $L > T + W/2$ , применяется полное сжатие:  $G = T + (L - T)/R - L$ ;
- если  $T - W/2 \leq L \leq T + W/2$ , плавный переход с помощью квадратичной функции:  $\Delta = L - (T - W/2)$ ,  $G = ((1/R - 1) \Delta^{**2})/2W$ .

5. Makeup Gain (усиление после компрессии). Если  $G$  — makeup gain в децибелах, то коэффициент усиления в линейной шкале:  $G = 10^{**}(G/20)$ . Итоговый выходной сигнал:  $y[n] = x[n] * g[n] * G$ , где  $y[n]$  — выходной сигнал на сэмпле  $n$ ,  $x[n]$  — входной сигнал на сэмпле  $n$ ,  $g[n]$  — коэффициент усиления компрессора (динамическое уменьшение или увеличение уровня) на сэмпле  $n$ .

Компрессор должен быть реализован с приоритетом качества звука, стабильности работы и удобства пользовательского управления параметрами. Вся логика фильтрации, разделения на полосы и динамической обработки должна быть максимально прозрачна для дальнейшей доработки и интеграции с другими эффектами аудиопроцессора.

#### 2.3.2.4 Реверберация

Реализовать алгоритм реверберации Moorer для аудиопроцессора, имитирующий акустику помещений с настраиваемыми параметрами. Алгоритм должен сочетать ранние отражения (early reflections) и поздний реверберационный хвост (late reverb), обеспечивая естественное пространственное звучание.

Основные параметры:

1. Размер помещения (Room Size): диапазон: 0.1 (малая комната) – 2.0 (большой зал). влияет на плотность и задержку ранних отражений/хвоста.
2. Соотношение Wet/Dry: Диапазон: 0 (только сухой сигнал) – 100 (только реверберация).
3. Демпфирование (Damping): Диапазон: 0 (полное отражение ВЧ) – 100 (сильное подавление ВЧ). Регулирует затухание высоких частот в хвосте.
4. Pre-Delay: диапазон: 0–200 мс. Задержка перед началом реверберации для имитации расстояния до стен.

Техническая реализация:

- использование кольцевого буфера (delay line) для обработки задержек;
- оптимизация вычислений через JIT-компиляцию (Numba) для работы в реальном времени;
- поддержка моно и стерео сигналов;
- кэширование коэффициентов фильтров при изменении параметров.

### 2.3.2.5 Визуализация

Реализовать модуль визуализации аудиосигнала, обеспечивающий отображение формы волны и амплитудно-частотной характеристики (АЧХ) в реальном времени. Модуль предназначен для наглядного представления динамики и спектрального состава звука, облегчая анализ и настройку аудио-эффектов.

Визуализация формы волны:

- отображать временную амплитуду аудиосигнала (осциллограмму) с возможностью масштабирования по времени и амплитуде;
- поддержка моно и стерео сигналов с разделением каналов;
- обновление визуализации в реальном времени с частотой не менее 30 кадров в секунду;
- обеспечить плавное и без мерцаний отображение;
- возможность отображения как полного сигнала, так и его текущего фрагмента (окна).

Визуализация амплитудно-частотной характеристики (АЧХ):

- расчёт спектра аудиосигнала с использованием быстрого преобразования Фурье (FFT);
- отображение амплитуды частотного спектра в логарифмическом или линейном масштабе по оси частот (от 20 Гц до 20 кГц);
- обновление спектра в реальном времени с возможностью регулировки оконного размера FFT для баланса между частотным и времененным разрешением;
- поддержка отображения спектра для каждого канала отдельно (в случае стерео);
- визуализация текущих настроек эквалайзера и эффектов (например, наложение АЧХ эквалайзера).

### 2.3.2.6 Управление

Реализовать модуль управления аудиопроцессором, обеспечивающий интерактивное управление воспроизведением, настройкой параметров аудиоэффектов и навигацией по аудиофайлам через графический интерфейс и программный API.

Управление воспроизведением:

- кнопки Play и Stop для запуска и остановки воспроизведения аудиофайла;
- ползунок (слайдер) для перемотки и точного позиционирования в аудио;
- отображение текущей позиции воспроизведения и общей длительности аудиофайла;
- поддержка управления воспроизведением в реальном времени без задержек.

Управление параметрами эффектов:

- регулировка параметров эквалайзера (8 полос): усиление/ослабление каждой полосы в диапазоне  $\pm 12$  дБ;
- управление параметрами многополосного компрессора: threshold, ratio, attack, release, makeup gain, soft knee для каждой полосы (low, mid, high);
- настройка параметров реверберации: wet/dry, room size, damping;
- выбор и регулировка параметров шумоподавления (метод, интенсивность);
- все регуляторы должны быть реализованы через удобные элементы управления (ползунки, кнопки, вращающиеся регуляторы).

Состояния и обратная связь:

- отображение текущих значений параметров в интерфейсе;
- обеспечение мгновенного отклика интерфейса на изменение параметров;
- индикация состояния воспроизведения (воспроизводится/паузируется/стоп);

- защита от некорректных значений параметров (валидация ввода).

### 2.3.3 Моделирование вариантов использования

На рисунке 2.1 изображена диаграмма прецедентов

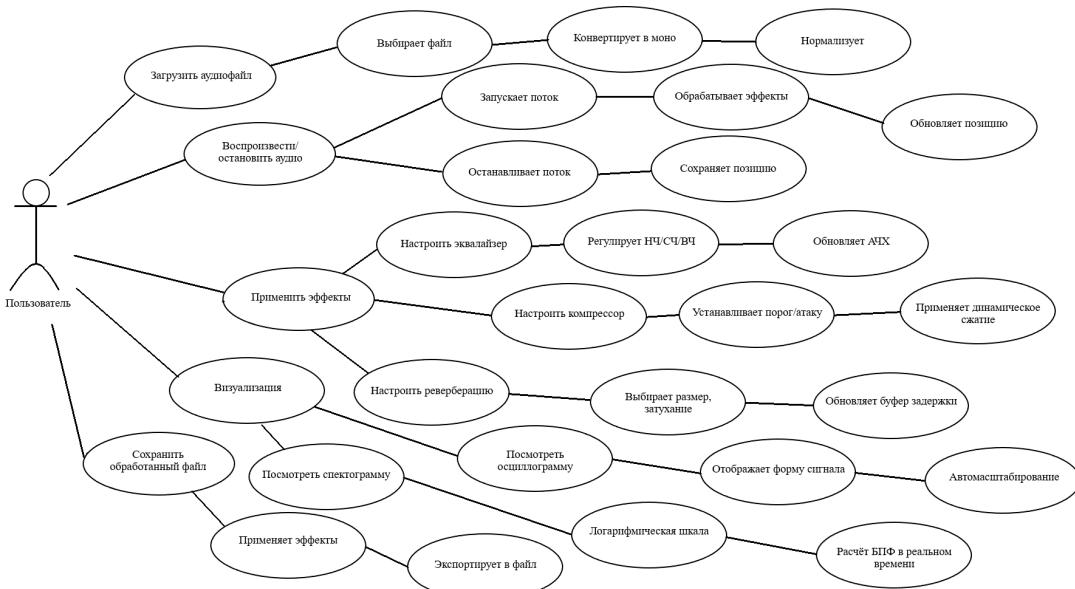


Рисунок 2.1 – Диаграмма прецедентов

#### 2.3.3.1 Вариант использования «Загрузка аудиофайла»

Заинтересованные лица и их требования: пользователь хочет загрузить аудиофайл для последующей обработки.

Требования: поддержка форматов (WAV, MP3, OGG, FLAC), автоматическая конвертация в моно, нормализация амплитуды.

Предусловие: приложение запущено, файл существует и доступен для чтения.

Постусловие: аудиоданные загружены в память, форма волны отображена на графике, кнопки "Воспроизвести" и "Сохранить" активированы.

Основной успешный сценарий:

1. Пользователь нажимает кнопку "Загрузить".
2. Открывается диалоговое окно выбора файла.
3. Пользователь выбирает файл и подтверждает выбор.

### **2.3.3.2 Вариант использования «Воспроизведение аудио»**

Заинтересованные лица и их требования: пользователь хочет прослушать загруженный файл с возможностью применения эффектов в реальном времени.

Предусловие: аудиофайл успешно загружен, аудиоустройство вывода доступно.

Постусловие: аудиопоток запущен, графики обновляются в реальном времени.

Основной успешный сценарий:

1. Пользователь нажимает кнопку «Воспроизвести».
2. Система инициализирует аудиопоток.
3. Callback-функция начинает передавать данные в устройство.
4. Позиция воспроизведения обновляется каждые 100 мс.
5. Сигнал и АЧХ отображаются в реальном времени.

### **2.3.3.3 Вариант использования «Остановка воспроизведения»**

Заинтересованные лица и их требования: пользователь хочет немедленно прекратить воспроизведение аудио.

Предусловие: идёт воспроизведение аудиофайла.

Постусловие: аудиопоток полностью остановлен, интерфейс обновлен.

Основной успешный сценарий:

1. Пользователь нажимает кнопку "Стоп".
2. Система останавливает воспроизведение аудио.
3. Текущая позиция фиксируется для возможности дальнейшего продолжения.

### **2.3.3.4 Вариант использования «Изменения позиции воспроизведения»**

Заинтересованные лица и их требования: пользователь хочет переместить аудио на произвольную позицию.

Предусловие: аудиофайл загружен.

Постусловие: текущая позиция воспроизведения изменена, при активном воспроизведении звук продолжается с новой позиции, интерфейс синхронизирован.

Основной успешный сценарий:

1. Пользователь перемещает ползунок позиции мышью.
2. Система пересчитывает позицию.
3. При воспроизведении поток начинает чтение с новой позиции.

### **2.3.3.5 Вариант использования «Применения шумоподавления»**

Заинтересованные лица и их требования: пользователь хочет уменьшить фоновый шум в аудиозаписи для улучшения качества звучания.

Предусловие: файл загружен, кнопка «Шумоподавление» нажата.

Постусловие: выбранный метод шумоподавления и степень обработки применены к аудиопотоку, визуализация обновлена.

Основной успешный сценарий:

1. Пользователь активирует кнопку «Шумоподавление».
2. Выбирает метод шумоподавления из выдвижного списка (спектральное, медианный фильтр, гауссово размытие).
3. Регулирует интенсивность шумоподавления с помощью knob.
4. Система применяет выбранный алгоритм с заданной интенсивностью к аудиосигналу.
5. Обновляется визуализация формы волны и АЧХ.

### **2.3.3.6 Вариант использования «Применения эквалайзера»**

Заинтересованные лица и их требования: пользователь хочет настроить частотный баланс (НЧ/СЧ/ВЧ).

Предусловие: файл загружен, вкладка "Эквалайзер" открыта.

Постусловие: параметры эквалайзера применены к аудиопотоку, график АЧХ обновлен.

Основной успешный сценарий:

1. Пользователь включает чекбокс «Эквалайзер».
2. Регулирует ползунки НЧ/СЧ/ВЧ.
3. Система пересчитывает коэффициент БИХ-фильтров.
4. Аудиозапись изменяется в соответствии с настройками пользователя.
5. АЧХ отображается на графике.

### **2.3.3.7 Вариант использования «Применения компрессора»**

Заинтересованные лица и их требования: пользователь хочет изменить динамический диапазон аудиосигнала для более равномерного звучания.

Предусловие: аудиофайл успешно загружен, вкладка "Компрессор" открыта.

Постусловие: параметры компрессора применены к аудиопотоку, изменения слышны при воспроизведении.

Основной успешный сценарий:

1. Пользователь активирует чекбокс «Компрессор».
2. Устанавливает диапазон обработки частот для каждого из трёх фильтров.
3. Устанавливает порог, коэффициент компрессии, время атаки и восстановления, усиление выходного сигнала.
4. Аудио обрабатывается в режиме реального времени.

### **2.3.3.8 Вариант использования «Применения реверберации»**

Заинтересованные лица и их требования: пользователь хочет добавить эффект реверберации к аудиосигналу.

Предусловие: аудиофайл успешно загружен, вкладка "Реверберация" открыта.

Постусловие: эффект реверберации применен к аудиопотоку, буфер реверберации инициализирован с новыми параметрами.

Основной успешный сценарий:

1. Пользователь активирует чекбокс "Реверберация".

2. Пользователь регулирует параметры: уровень эффекта, исходный звук, размер комнаты, затухание.
3. Аудио обрабатывается в режиме реального времени.

### **2.3.3.9 Вариант использования «Сохранение обработанного файла»**

Заинтересованные лица и их требования: пользователь хочет экспортировать результат с примененными эффектами.

Предусловие: файл загружен, эффекты настроены.

Постусловие: файл сохранен на диск в выбранном формате.

Основной успешный сценарий:

1. Пользователь нажимает "Сохранить".
2. Открывается диалог выбора формата (WAV/MP3) и пути.
3. Пользователь выбирает формат, путь и название файла.
4. Система применяет выбранные эффекты, экспортирует файл.
5. Файл успешно сохранён на диске.

### **2.3.3.10 Вариант использования «Визуализация данных»**

Заинтересованные лица и их требования: пользователь хочет видеть графическое представление аудиосигнала.

Предусловие: аудиофайл загружен и идет воспроизведение.

Постусловие: графики формы волны и АЧХ отображаются и обновляются в режиме реального времени.

Основной успешный сценарий:

1. Пользователь нажимает «Воспроизвести».
2. Система отображает графики формы волны и АЧХ.

## **2.4 Требования пользователя к интерфейсу приложения**

Приложение должно иметь следующие основные функции:

1. Основные элементы управления - кнопки управления воспроизведением: "Загрузить" "Воспроизвести" "Стоп" "Сохранить". Ползунок позиции

воспроизведения, точное позиционирование, индикатор текущей позиции в формате MM:SS / MM:SS.

2. Визуализация аудио - два синхронизированных графика: верхний график формы волны и нижний график амплитудно-частотной характеристики. Подписи осей с единицами измерения (амплитуда, частота).

3. Кнопка "Шумоподавление" с двумя положениями: активно/неактивно, выплывающий список с выбором метода, регулировка интенсивности шумоподавления.

4. Панель эффектов (вкладки):

(a) Эквалайзер - Три ползунка с подписями: низкие частоты (150Hz):  $\pm 24$  дБ, средние частоты (1kHz):  $\pm 24$  дБ, высокие частоты (5kHz):  $\pm 24$  дБ. Чекбокс активации эффекта.

(b) Компрессор - ползунки параметров: порог: -60..0 дБ, соотношение: 1..20, атака: 1..500 мс, восстановление: 10..2000 мс, усиление: 0..24 дБ. Панель отображения диапазонов изменения и применения параметров, а также ползунки настройки и регулировки диапазонов. Чекбокс активации эффекта.

(c) Реверберация: ползунки параметров: уровень эффекта: 0..1, сухой сигнал: 0..1, размер помещения: 0.1..2.0, затухание: 0.1..0.9. Чекбокс активации эффекта.

5. Обратная связь и состояние: индикатор загрузки при обработке файлов. Всплывающие уведомления об ошибках: при неудачной загрузке файла, при проблемах с аудиоустройством, при ошибках сохранения. Изменение состояния кнопок: "Воспроизвести" активна только при загруженном файле, "Стоп" активна только во время воспроизведения, "Сохранить" активна только после загрузки файла.

6. Производительность: задержка отклика интерфейса не более 100 мс, плавная анимация графиков без рывков, минимальное потребление ресурсов в фоновом режиме.

## 2.4.1 Нефункциональные требования к программно-информационной системе

### 2.4.1.1 Требования к надёжности

В процессе работы аудио-процессора могут возникнуть следующие аварийные ситуации:

- потеря доступа к аудиоустройству в связи с его отключением, изменением системных настроек звука или конфликтом с другим приложением;
- попытка загрузки повреждённого или неподдерживаемого аудиофайла с несовместимым форматом, битрейтом или частотой дискретизации;
- неожиданное прекращение работы из-за нехватки системных ресурсов при обработке объёмных файлов или одновременном применении нескольких ресурсоёмких эффектов;
- ошибки в работе эффектов обработки звука, приводящие к искажению аудиосигнала или сбоям в воспроизведении.

Приложение должно автоматически определять доступные аудиоустройства и переключаться между ними при потере соединения с текущим устройством. В случае невозможности восстановления подключения система должна сохранять возможность обработки звука без функции воспроизведения с уведомлением пользователя о возникшей проблеме.

Для работы с аудиофайлами приложение должно проверять их целостность и соответствие поддерживаемым форматам перед загрузкой, а также автоматически выполнять необходимые преобразования (нормализацию, конвертацию в моно, приведение к стандартной частоте дискретизации). При обнаружении критических ошибок в файле пользователь должен получить понятное сообщение о проблеме с указанием конкретной причины.

Для предотвращения аварийного завершения из-за нехватки ресурсов система должна контролировать использование оперативной памяти и процессорного времени, ограничивать размеры буферов обработки и корректно завершать работу при приближении к предельным значениям. Все парамет-

ры эффектов и текущее состояние обработки должны сохраняться даже при аварийном завершении работы.

При возникновении ошибок в работе эффектов обработки звука система должна автоматически сбрасывать параметры эффектов к безопасным значениям, сохраняя при этом возможность дальнейшей работы. Все критические ошибки должны фиксироваться в системном логе с указанием времени возникновения, типа ошибки и состояния системы на момент сбоя.

#### **2.4.1.2 Требования к безопасности**

Требования к приложению:

1. Перед загрузкой файла, система должна проверять формат и корректность заголовков. При обнаружении повреждённых данных программа будет выводить ошибку без попытки обработки.
2. Для безопасности управления памятью и ресурсами необходима изоляция обработки аудио, то есть каждый эффект будет применяться в отдельном потоке с контролем потребления ресурсов и будет ограничен буфере. Так же потребуется очистка временных данных, чтобы после завершения обработки или при ошибке все промежуточные буферы будут очищаться.
3. Для защиты от вирусов применится проверка структуры файлов, анализ сигнатуры файла перед декодированием, ограничение максимальной длительности обрабатываемого аудио.
4. Пользовательские настройки будут проверяться перед применением, а недопустимые значения автоматически скорректируются до ближайших безопасных.
5. При завершении работы, приложение будет останавливать все потоки, освобождать аудиоустройства.

#### **2.4.1.3 Требования к программному обеспечению**

Для реализации программы будет использоваться язык программирования Python и библиотеки: NumPy, SciPy, PyDub, SoundDevice, Matplotlib, Numba.

Для работы приложения требуется Windows 10/11 (64-bit)

#### **2.4.1.4 Требования к аппаратному обеспечению**

Минимальная конфигурация:

Центральный процессор с количеством ядер от 2 и выше с тактовой частотой от 2.0 ГГц. Поддержка инструкций SSE4.2. Оперативная память - 4 ГБ. 100 МБ свободного места для установки. Поддержка ASIO/WASAPI для профессионального использования.

#### **2.4.2 Требования к оформлению документации**

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

### **3 Технический проект**

#### **3.1 Общая характеристика решения задачи**

Задача дипломного проекта заключается в разработке программного аудиопроцессора — универсального программного комплекса для обработки аудиосигналов в реальном времени на персональном компьютере. Процессор реализует цепочку современных аудиоэффектов с возможностью гибкой настройки, а также поддерживает многополосную обработку сигнала, что позволяет применять различные параметры эффектов к разным частотным диапазонам.

Разработка аудиопроцессора, способного выполнять обработку аудиосигнала с применением эквалайзации, многополосной компрессии, реверберации и шумоподавления.

Входные данные:

- аудиосигнал в виде цифрового потока;
- настройки эффектов, задаваемые пользователем (параметры эквалайзера, компрессора, реверберации и шумоподавления).

Выходные данные:

- обработанный аудиосигнал, выводимый на устройство воспроизведения или сохраняемый в файл;
- возможность мониторинга и визуализации параметров обработки в виде отображения формы волны или АЧХ.

#### **3.2 Описание используемых технологий и языков программирования**

Для разработки аудиопроцессора был выбран язык программирования Python, что обусловлено рядом ключевых факторов:

1. Простота и читаемость кода. Синтаксис Python близок к псевдокоду, что облегчает понимание и сопровождение проекта, а также позволяет быстро экспериментировать с алгоритмами обработки звука.

2. Широкий набор библиотек для цифровой обработки сигналов. Python обладает мощными библиотеками (NumPy, SciPy, pydub и др.), которые предоставляют готовые инструменты для работы с аудиоданными, фильтрами, преобразованиями и другими DSP-операциями. Это значительно ускоряет разработку и повышает надежность кода.

3. Возможность оптимизации производительности. Для повышения скорости вычислений в критичных местах применяется JIT-компиляция с помощью Numba, что позволяет добиться близкой к нативной производительности без перехода на более низкоуровневые языки.

4. Поддержка многопоточности и асинхронной обработки. В проекте используется ThreadPoolExecutor и другие средства стандартной библиотеки для параллельной обработки аудиоданных, что улучшает отзывчивость и снижает задержки при работе в реальном времени.

5. Гибкость и расширяемость архитектуры. Модульный подход с использованием классов и четким разделением функционала (например, отдельные классы для компрессоров, эквалайзеров и ревербераторов) позволяет легко добавлять новые эффекты и модифицировать существующие.

6. Активное сообщество и наличие обучающих материалов. Благодаря большому количеству статей, учебников и примеров по цифровой обработке сигналов на Python, разработка и отладка проекта становится более эффективной.

Таким образом, выбор Python и сопутствующих технологий проектирования обусловлен их оптимальным сочетанием простоты, функциональности и производительности, что отвечает требованиям дипломного проекта по созданию программного аудиопроцессора с возможностью обработки в реальном времени и расширяемой архитектурой.

В коде используются следующие библиотеки Python:

- NumPy - обеспечивает высокопроизводительные вычисления с многомерными массивами, что критично для обработки аудиоданных, представленных в виде временных рядов;

- SciPy - используется для реализации цифровых фильтров (эквалайзер), БПФ (анализ спектра) и других математических операций;
- PyDub - библиотека для работы с аудиофайлами, предоставляющая: простые методы загрузки и сохранения в форматах WAV, MP3, OGG, FLAC, базовые операции, такие как обрезка, наложение, изменение громкости, конвертация частоты дискретизации, интеграцию с FFmpeg для поддержки дополнительных кодеков;
- Soundfile используется для чтения и записи аудиофайлов различных форматов, обеспечивая удобный доступ к аудиоданным на диске, позволяет загружать исходные аудиозаписи и сохранять результаты обработки без потери качества;
- SoundDevice - обеспечивает низкоуровневый доступ к аудиоустройствам через PortAudio. Ключевые функции: воспроизведение и запись в реальном времени с минимальной задержкой, поддержка ASIO, WASAPI, Core Audio для профессиональных аудиоинтерфейсов, гибкая настройка параметров потока: частота дискретизации, размер буфера, количество каналов;
- Matplotlib - используется для визуализации аудиоданных: построение осциллограмм (форма сигнала во временной области), отображение АЧХ (амплитудно-частотных характеристик) с логарифмической шкалой, интерактивное обновление графиков в реальном времени;
- Threading и concurrent.futures — стандартные библиотеки Python для организации многопоточной и параллельной обработки;
- Tkinter - стандартная библиотека Python для создания графического интерфейса. В проекте применяется для: построения основного окна с вкладками (эквалайзер, компрессор, реверберация), реализации интерактивных элементов: ползунки, кнопки, метки, интеграции графиков Matplotlib через FigureCanvasTkAgg;
- Numba - JIT-компилятор для оптимизации вычислительно сложных участков кода: ускорение алгоритмов компрессии и реверберации в 5–10 раз, поддержка многопоточности.

### 3.2.1 Кроссплатформенная библиотека FFmpeg

FFmpeg — это мощная кроссплатформенная библиотека для обработки мультимедиа, используемая в проекте для работы с аудиофайлами различных форматов. Взаимодействие с FFmpeg осуществляется через обёртку PyDub, что значительно упрощает операции чтения, записи и конвертации аудио.

Основные функции FFmpeg в проекте:

1. Поддержка множества аудиоформатов. Позволяет загружать и сохранять файлы в форматах: без сжатия: WAV, AIFF, с потерями: MP3, AAC, OGG, без потерь: FLAC, ALAC. Обеспечивает автоматическое определение кодека при загрузке.
2. Конвертация аудио: изменение частоты дискретизации, преобразование между форматами, конвертация числа каналов.
3. Нормализация и обработка. Автоматическая регулировка громкости.

FFmpeg был выбран по рядам преимуществ:

- универсальность: поддержка 100+ кодеков и контейнеров;
- стабильность: отлаженные алгоритмы декодирования/кодирования;
- производительность: оптимизированные нативные библиотеки;
- гибкость: Возможность тонкой настройки параметров через командные опции.

## 3.3 Эффекты

### 3.3.1 Устройство шумоподавления

Система шумоподавления реализует 4 метода с общим конвейером обработки:

1. Спектральное шумоподавление. На основе библиотеки noisereduce, анализирует FFT и подавляет шум в частотной области.
2. Фильтр Винера. Адаптивная фильтрация во временной области.
3. Медианный фильтр. Подавление импульсных шумов.
4. Гауссовский фильтр. Сглаживание высокочастотного шума.

Ключевые особенности:

- автодетекция шума: система автоматически запоминает шумовой профиль из первых 2048 сэмплов;
- гибкая настройка: регулировка силы подавления (0–100) и выбор метода под тип шума;
- потокобезопасность: все операции работают с копиями данных и защищены блокировками;
- оптимизация под реальное время: фиксированный размер FFT-окна (512 точек) для минимальной задержки.

### 3.3.2 Устройство эквалайзера

Эквалайзер — это устройство или программный алгоритм, предназначенный для корректировки амплитудно-частотной характеристики (АЧХ) звукового сигнала. Он позволяет усиливать или ослаблять определённые частотные диапазоны, изменяя тембр звука. Эквалайзер работает, разделяя входной аудиосигнал на несколько частотных полос (диапазонов), каждая из которых обрабатывается отдельно. После обработки всех полос сигналы складываются, и на выходе получается звук с изменённой частотной характеристикой.

В программе реализован цифровой параметрический эквалайзер, у которого есть три полосы частот, а так же коррекция их по громкости в децибелах. Такие эквалайзеры часто используются в профессиональной звукозаписи и сведении аудио.

Цифровой IIR-фильтр (Биквадратный, на основе Баттервортта)

IIR-фильтр (Infinite Impulse Response — бесконечная импульсная характеристика) — это тип цифрового фильтра, который использует обратную связь, благодаря чему может иметь очень крутые склоны АЧХ при малом порядке.

Биквадратный фильтр (Biquad) — это частный случай IIR-фильтра 2-го порядка, который реализуется с помощью разностного уравнения и часто используется в эквалайзерах из-за своей эффективности.

Фильтр Баттерворта — один из самых популярных типов фильтров, обеспечивающий максимально гладкую АЧХ в полосе пропускания без пульсаций.

Этот эквалайзер обеспечивает:

- низкую задержку (важно для реального времени);
- минимальные фазовые искажения;
- точную обработку частот;
- RLock для защиты состояний фильтров.

### 3.3.3 Устройство многополосного компрессора

Многополосный компрессор — устройство динамической обработки аудиосигнала, предназначенное для управления динамическим диапазоном звука с разделением сигнала на несколько частотных полос. Основная идея многополосного компрессора заключается в том, что входящий аудиосигнал сначала разделяется на несколько частотных диапазонов с помощью цифровых фильтров, после чего каждая полоса обрабатывается отдельным компрессором с индивидуальными параметрами. Такой подход позволяет более точно и эффективно контролировать динамику звука в каждом частотном диапазоне, сохраняя при этом естественность звучания и минимизируя искажения.

В реализованном компрессоре разделение сигнала осуществляется с использованием цифровых фильтров, которые выделяют низкочастотную, среднечастотную и высокочастотную полосы. Для каждой полосы создаётся отдельный компрессорный блок, реализованный в виде программного модуля, который обрабатывает сигнал независимо от других полос. Каждый из этих блоков имеет собственные настройки ключевых параметров компрессии:

- Threshold (порог);
- Ratio (коэффициент компрессии);
- Attack (атака);
- Release (восстановление);

- Knee (характеристика перехода);
- Make-up Gain (компенсационное усиление).

Реализация ЛТ-функций для динамического сжатия аудиосигнала.

Пошаговый алгоритм:

- вычисление огибающей;
- инициализация gain-массива;
- обработка каждого семпла;
- преобразование dB в линейное значение:;
- экспоненциальное сглаживание;
- применение gain к сигналу.

Особенности оптимизации:

- векторизованные операции NumPy;
- предварительный расчет коэффициентов;
- пул потоков для фоновой обработки;
- отдельные состояния для каждой полосы;
- отсутствие ветвлений в критическом цикле.

Порог срабатывания определяет уровень громкости, при превышении которого начинается сжатие сигнала. Коэффициент сжатия задаёт степень уменьшения динамического диапазона, то есть насколько сильно будет снижаться громкость сигналов, превышающих порог. Время атаки регулирует скорость срабатывания компрессора после превышения порога, а время релиза — скорость возврата усиления к исходному уровню после снижения входного сигнала ниже порога, характеристика перехода определяет плавный переход от отсутствия сжатия к активному сжатию сигнала, компенсационное усиление позволяет регулировать сигнал после обработки, чтобы вернуть ему исходную громкость. Эти параметры позволяют гибко настраивать реакцию компрессора на изменения громкости в каждой полосе, обеспечивая плавное и естественное звучание.

В основе алгоритма компрессии лежит вычисление коэффициента усиления (gain reduction), который применяется к аудиосигналу полосы. Этот коэффициент рассчитывается на основе текущего уровня сигнала и параметров

компрессии с учётом сглаживания изменения усиления для предотвращения резких артефактов в звуке. Для повышения производительности вычисления оптимизированы с помощью JIT-компиляции (Numba), а обработка полос может выполняться параллельно с использованием многопоточности, что обеспечивает минимальную задержку и возможность работы в реальном времени.

Таким образом, многополосный компрессор, реализованный в проекте, представляет собой совокупность нескольких параллельных компрессоров, каждый из которых отвечает за свой частотный диапазон и имеет независимые настройки. Это обеспечивает высокую гибкость и качество динамической обработки, позволяя адаптировать параметры под особенности конкретного аудиоматериала и задачи. Использование цифровых фильтров для разделения сигнала и оптимизированных алгоритмов компрессии обеспечивает эффективную работу устройства в реальном времени, что соответствует современным требованиям к аудиопроцессорам.

### 3.3.4 Устройство реверберации

FDN (Feedback Delay Network) — это один из самых мощных алгоритмов цифровой реверберации, используемый для создания реалистичных пространственных эффектов. Он основан на множестве линий задержки с обратной связью, образующих сложную сеть, имитирующую отражения в помещении.

Принцип работы:

- входной сигнал разделяется на несколько параллельных линий задержки, каждая из которых задерживает сигнал на разное время;
- после задержки сигнал проходит через фильтр демпфирования (имитация потери высоких частот);
- затем он умножается на коэффициенты матрицы обратной связи, определяющей, какая часть сигнала возвращается в каждую линию;
- обновлённый сигнал снова поступает на вход линий задержки;
- результирующий реверберационный сигнал формируется как сумма выходов всех линий.

Применение ЛТ-компиляции для создания алгоритма реверберации на основе задержки.

Пошаговый алгоритм:

- инициализация буферов;
- цикл обработки семплов;
- смешивание сигналов;
- обновление буфера задержки;
- циклическое перемещение по буферу.

Ключевые особенности FDN:

- реалистичность — лучше имитирует поздние отражения, чем алгоритмы на основе простых задержек;
- гибкость — можно настраивать время реверберации, демпфирование и пространственность;
- стабильность — ортогональная матрица гарантирует отсутствие бесконечного нарастания.

Для обеих функций используются идентичные параметры декоратора:

`@jit(nopython=True, nogil=True, cache=True)`, где:

- `nopython=True` - строгий режим (обязательная компиляция);
- `nogil=True` - освобождение GIL для многопоточности;
- `cache=True` - кэширование скомпилированного кода.

### 3.4 Архитектура программно-информационной системы

На рисунке 3.1 представлена многослойная архитектура системы с чётким разделение ответственности между компонентами.

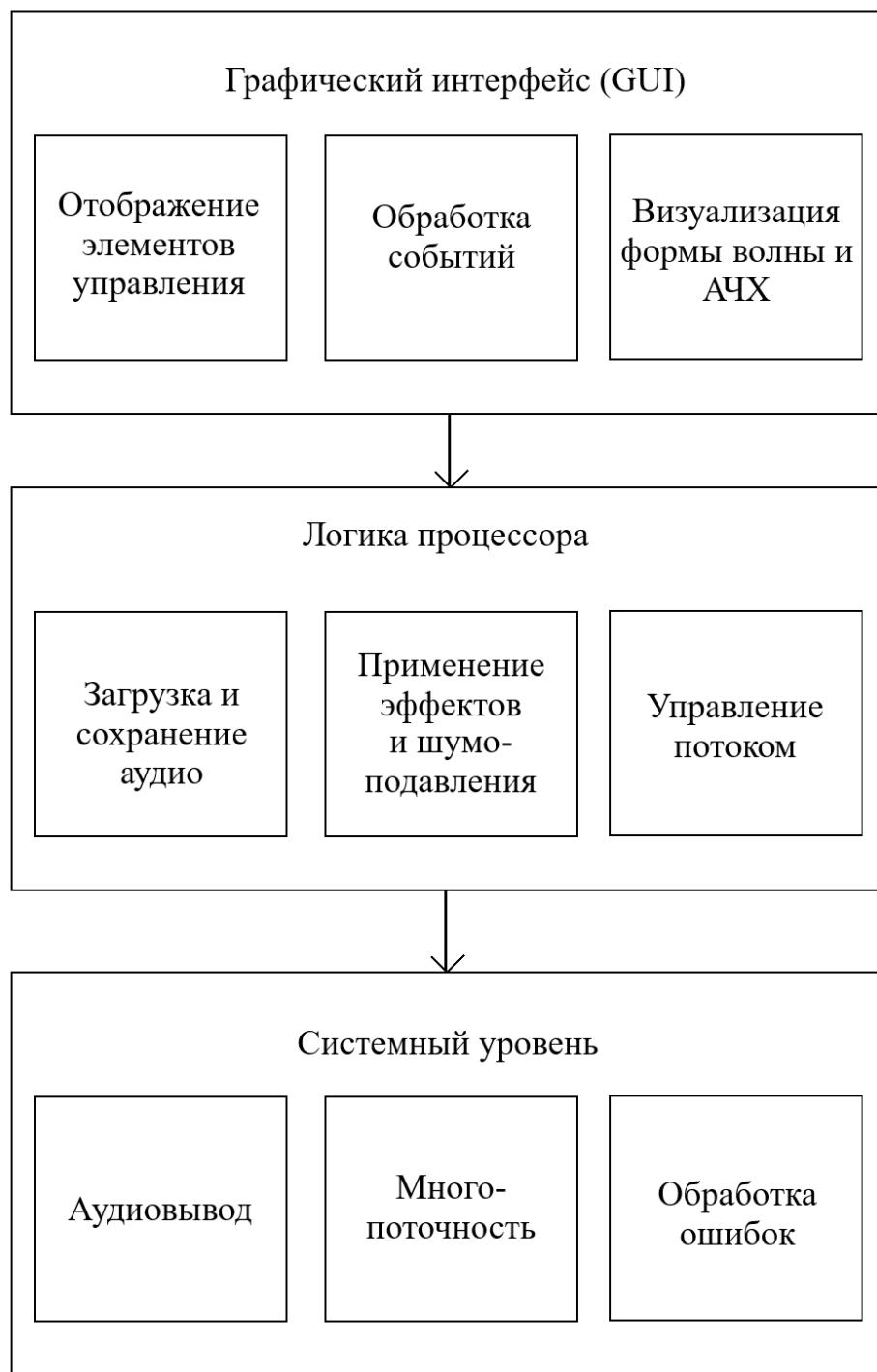


Рисунок 3.1 – Архитектура программно-информационной системы

1. Графический интерфейс отображает элементы управления (кнопки, ползунки, вкладки), визуализирует аудиоданные в реальном времени (Форма волны, АЧХ), обрабатывает пользовательские события (нажатия кнопок, изменение параметров).

2. Логика процессора отвечает за загрузку/сохранение аудио, чтение файлов через PyDub/FFmpeg, конвертирует в формат float32 для обработки. Применяет эффекты: шумоподавление, эквалайзер (БИХ-фильтры для 8 полос регулировки частот), многополосный компрессор (динамическое сжатие с параметрами), реверберация. Буферизирует данные и синхронизирует позицию воспроизведения.

3. На уровне системы происходит ввод и вывод аудио, создаётся отдельный поток для обработки аудио и очереди для передачи данных между потоками. Обрабатываются ошибки, перехватываются исключения и логируются в файл.

- графический интерфейс включает главное окно, состоящее из кнопок управление воспроизведением и загрузки/сохранения, вкладок эффектов (эквалайзер, компрессор и ревербератор), окна отображение визуализации формы волны и АЧХ;
- логика процессора читает и обрабатывает аудиофайлы, применяется выбранные эффекты по очереди, управляет всей работой программы;
- системное устройство воспроизводит звук, работает с файлами, открывает и сохраняет их, управляет несколькими задачами одновременно;
- вспомогательные модули включают частотный анализ, автоматически регулирует громкость и записывает ошибки и события.

Все части программы взаимодействуют между собой по чётким правилам. Когда пользователь меняет настройки, интерфейс передаёт их в модуль обработки, который применяет эффекты и отправляет результат на воспроизведение и отображение. Для плавной работы использовались очереди задач для работы в многопоточном режиме, оптимизирует сложные вычисления для быстрой работы.

### 3.5 Архитектура приложения

Приложение построено по многослойной модульной архитектуре с разделением на логические компоненты, взаимодействующие через четко определенные интерфейсы. В основе лежит ядро обработки аудиосигналов, окруженное графическим интерфейсом, системными сервисами и вспомогательными модулями. Графический интерфейс реализован на Tkinter и включает:

- главное окно (MainWindow) с вкладками для управления эффектами, кнопками воспроизведения/паузы и областью визуализации;
- панель эквалайзера (EQPanel) с восьмью ползунками, отображающая АЧХ через Matplotlib;
- панель компрессора (CompressorPanel) для настройки порога, сжатия, атаки, восстановления, характеристики перехода, усиления;
- панель реверберации (ReverbPanel) для настройки эффекта, сухой сигнал, размер комнаты, затухание;
- визуализатор спектра (SpectrumAnalyzer), отображающий осциллограмму (форма сигнала) и частотный спектр (БПФ) в реальном времени;
- прогресс-бар с управлением позицией воспроизведения и отображением длительности трека.

Логический процессор — центральный модуль, отвечающий за:

- загрузку и конвертацию аудио через PyDub (с использованием FFmpeg как бэкенда), включая поддержку WAV, MP3, FLAC;
- эквалайзер на БИХ-фильтрах с частотами 50 Гц, 150 Гц, 250 Гц, 500 Гц, 1 кГц, 3 кГц, 7 кГц, 15 кГц;
- компрессор с разделением на три диапазона частот, в каждом из которых есть регулировки: threshold, ratio, knee, attack, release, gain;
- реверберация на основе алгоритма Schroeder (комбинация линий задержки и FIR-фильтров);
- буферизацию данных для плавного воспроизведения и нормализацию уровня сигнала.

Системный слой обеспечивает интеграцию с ОС и оборудованием:

- аудиопоток (AudioStream) на базе SoundDevice, обрабатывающий ввод/вывод в реальном времени с настройкой latency и sample rate (44.1–192 кГц);
- менеджер потоков (ThreadManager) для параллельной обработки (отдельные потоки для: GUI, аудиообработки, визуализации);
- файловый менеджер (FileManager) с кэшированием загруженных треков и поддержкой метаданных (ID3-теги).

Архитектура обеспечивает масштабируемость (добавление новых эффектов через модули), производительность (ЛТ-компиляция, многопоточность) и отказоустойчивость (изоляция сбоев в отдельных потоках).

### 3.5.1 Многопоточная обработка

В коде используется несколько потоков для различных задач

Основные потоки класса AudioProcessor:

1. Поток обработки аудио (processing\_loop). Основной цикл обработки аудиоданных. Выполняет: обрабатывает аудиоданные из очереди `input_queue`, применяет эффекты (эквалайзер, компрессор, реверберацию), отправляет обработанные данные в очередь `output_queue`, управляет частотой обновления для минимизации нагрузки на CPU.

2. Потоки пула исполнителей (ThreadPoolExecutor). Выполнение задач в фоновом режиме. Выполняет: обработку аудио в фоне (`process_in_background`), другие тяжелые вычисления без блокировки основного потока.

3. Поток обновления позиции воспроизведения. Следит за текущей позицией воспроизведения. Обновляет ползунок и метку времени. Проверяет завершение воспроизведения.

Основные потоки класса AudioApp:

1. Поток инициализации приложения (`_background_init`). Инициализирует AudioProcessor. Загружает данные для графиков. Переключается на основной интерфейс после завершения.

2. Поток загрузки аудио (`_load_audio_in_thread`). Загрузка и конвертация аудиофайлов. Чтение файла. Конвертация в моно и нормализация. Передача данных в основной поток.

3. Поток воспроизведения аудио (через `sounddevice.OutputStream`). Непрерывная передача аудиоданных на звуковую карту. Вызывает `_audio_callback` для получения новых данных. Обрабатывает ошибки устройства. Останавливается при завершении воспроизведения.

4. Поток сохранения аудио (`_process_and_save_audio`). Сохранение обработанного аудио в файл. Применяет все эффекты к аудио. Конвертирует в выбранный формат. Сохраняет на диск.

5. Поток обновления интерфейса (`update_gui`). Периодическое обновление графиков и элементов управления. Обновляет форму волны и АЧХ (15 FPS). Обновляет позицию воспроизведения (10 FPS). Проверяет состояние воспроизведения.

Для потокобезопасности используются:

- `threading.RLock` для доступа к аудиоданным;
- `queue.Queue` для межпоточного обмена данными;
- `threading.Event` для управления состоянием;
- `after` в `Tkinter` для обновления GUI из основного потока.

### 3.6 Проект данных программно-информационной системы

Система оперирует тремя основными категориями данных: входными, промежуточными и выходными. Входные данные включают аудиофайлы в форматах WAV, MP3, FLAC и OGG с поддержкой различных характеристик (битность 16-24 бит, частота дискретизации 44.1-192 кГц), а также параметры эффектов, настраиваемые пользователем через графический интерфейс. Для эквалайзера это уровни усиления по восьми полосам (50 Гц, 150 Гц, 250 Гц, 500 Гц, 1 кГц, 3 кГц, 7 кГц, 15 кГц с диапазоном  $\pm 24$  дБ), для каждого диапазона компрессора — порог срабатывания от -60 до 0 дБ, коэффициент компрессии от 1:1 до 10:1, характеристика перехода от 0 до 100, время атаки от 0.1 до 100 мс, время восстановления от 10 до 1000 мс, усиление от

-20 до +20 дБ, для реверберации — соотношение обработанного и исходного сигнала, виртуальный размер помещения, затухание, две регулировки среза верхних и нижних частот, панорамирование.

Промежуточные данные представлены в виде нормализованных аудиобуферов в формате 32-битных чисел с плавающей запятой, организованных как моно- или стереоканальные массивы. Система использует кольцевые буфера для обработки в реальном времени и промежуточные спектральные данные, полученные через быстрое преобразование Фурье с применением оконной функции. Для хранения состояния эффектов используются специализированные структуры: коэффициенты БИХ-фильтров эквалайзера, параметры огибающей компрессора и линии задержки реверберации.

Выходные данные включают обработанные аудиофайлы в выбранных пользователем форматах (WAV с PCM-кодированием или MP3 с переменным битрейтом), сохраняющие исходные параметры частоты дискретизации или конвертируемые к стандартным значениям. Визуализационные данные содержат три основных компонента: осциллограмму последних обработанных сэмплов, частотный спектр с разрешением 8192 точек и амплитудно-частотную характеристику активных фильтров, отображаемую в логарифмическом масштабе от 20 Гц до 20 кГц.

Метаданные системы включают пользовательские пресеты эффектов, содержащие полный набор параметров обработки, историю операций для возможности отмены действий, а также технические лог-файлы с информацией о производительности и ошибках. Все данные передаются между компонентами системы через единый формат аудиоблоков, сопровождаемых метаинформацией о текущих настройках обработки, что обеспечивает согласованность работы многопоточной архитектуры.

### **3.6.1 Описание сущностей графического интерфейса**

Графический интерфейс Audio Processor представляет собой комплекс взаимосвязанных визуальных компонентов, объединенных в единое интуитивное пространство.

Интерфейс поддерживает несколько режимов отображения - компактный для мониторов с малым разрешением, расширенный с дополнительными инструментами анализа для профессиональной работы, и режим презентации с увеличенными элементами управления. Все визуальные компоненты реализованы с учетом эргономики - важные элементы выделены акцентным цветом, соблюдены принципы визуальной иерархии, обеспечена последовательная реакция на пользовательские действия. Особенностью интерфейса является синхронизация всех элементов - изменения параметров эффектов мгновенно отражаются на графиках, а действия пользователя сопровождаются тактильной обратной связью в виде тонких анимационных эффектов.

### **3.6.2 Описание сущностей логики процессора**

Основу обработки составляет низкоуровневый модуль, работающий с потоком аудиосэмплов в формате 32-битных чисел с плавающей точкой, обеспечивающий базовые операции нормализации и передискретизации. Система эффектов построена вокруг трех ключевых процессоров: эквалайзер реализует трехполосную фильтрацию через каскад БИХ-фильтров с настраиваемыми частотами среза и коэффициентами усиления, компрессор отвечает за компрессию сигнала с алгоритмами расчета огибающей и адаптивными параметрами атаки/восстановления, а ревербератор генерирует реверберацию через комбинацию линий задержки с регулируемыми параметрами затухания.

Поток данных управляется специализированным менеджером маршрутизации, который обеспечивает передачу аудиоблоков между модулями с минимальной задержкой, используя кольцевые буферы и механизм синхронизации для многопоточной работы. Состояние системы отслеживает активные эффекты, параметры обработки и текущий режим работы (реальное время/оффлайн обработка), предоставляя единый интерфейс для управления конвейером обработки.

### **3.6.3 Описание сущностей системного уровня**

Системная архитектура приложения построена на нескольких ключевых компонентах, обеспечивающих интеграцию с операционной средой и аппаратными ресурсами. Центральным элементом выступает абстрактный слой взаимодействия с аудиодрайверами, реализующий поддержку ASIO, WASAPI и Core Audio через единый кроссплатформенный API. Для управления устройствами ввода-вывода используется DeviceManager, который автоматически обнаруживает доступные аудиоинтерфейсы, анализирует их характеристики и предоставляет унифицированный интерфейс для работы с ними.

Файловая подсистема основана на компоненте, объединяющем возможности стандартных Python-библиотек для работы с файлами и мощь FFmpeg для обработки мультимедиа. Этот модуль включает кэширующий механизм для ускорения повторного доступа к аудиофайлам и систему контроля целостности данных.

Многопоточная архитектура координируется центральным диспетчером задач, который создает и управляет тремя основными типами потоков: высокоприоритетным аудиопотоком реального времени, фоновыми рабочими потоками для обработки эффектов и служебными потоками для визуализации и логирования.

## **3.7 Проектирования пользовательского интерфейса**

Центральное место занимает динамическая визуализация аудиопотока - двойной дисплей с синхронизированными осциллограммой и АЧХ, выполненный в темной цветовой гамме с акцентными элементами салатового цвета для выделения ключевых параметров сигнала. Основная рабочая область: верхняя треть экрана отведена под графики, центральная часть содержит компактные панели эффектов с интуитивными регуляторами, а нижний сектор занимает расширенная панель транспорта с профессиональными элементами управления.

Навигационная система реализована через адаптивную панель вкладок с контекстно-зависимыми элементами управления - при выборе конкретного эффекта (эквалайзер, компрессор, реверберация) нижняя панель автоматически наполняется соответствующими регуляторами, сохраняя при этом быстрый доступ к основным функциям. Все элементы управления спроектированы с учетом тактильного взаимодействия - ползунки имеют выраженные рифленые ручки с магнитными точками для часто используемых значений, кнопки обладают трехступенчатой визуальной обратной связью (покой, наведение, нажатие), а переключатели сопровождаются мягкими анимационными переходами.

## 4 Рабочий проект

### 4.1 Классы, используемые при разработке приложения

#### 4.1.1 Класс AudioProcessor

Класс для обработки аудиосигналов. Реализует эффекты (эквалайзер, компрессор, реверберацию) и предоставляет методы для их настройки и применения к аудиоданным. Поддерживает многопоточную обработку и потоковое воспроизведение.

Основные методы:

1. `__init__(self)`: инициализация параметров и буферов. Реализация: создает RLock-блокировки для потокобезопасности (`self.lock`, `self.position_lock`), инициализирует эффекты через `_init_effects()`, включая параметры эквалайзера (`eq_params`), компрессора (`compressor_params`), реверберации (`reverb_params`), настраивает пул потоков (`ThreadPoolExecutor`) для фоновых задач.
2. `load_audio(self, filename: str)`: загрузка аудиофайла через `pydub`. Реализация: конвертирует файл в моно/стерео, нормализует амплитуду, сохраняет данные как NumPy-массив в `self.audio_data`.
3. `process_in_background()`: асинхронная обработка аудио в отдельном потоке. Применение эффектов к большим аудиоблокам без зависания GUI. Обработка в `ThreadPoolExecutor` (пул из 2 потоков).
4. `processing_loop()`: основной цикл обработки аудио в фоне. Работает в отдельном потоке (`threading.Thread`). Использует очереди (`input_queue`, `output_queue`) для потокобезопасного обмена данными. Ограничивает нагрузку на CPU (через `time.sleep` при высокой загрузке).
5. `apply_all_effects(self, data: np.ndarray)`: применяет цепочку эффектов к аудиоблоку. Порядок обработки: шумоподавление (`apply_noise_reduction`), эквалайзер (`apply_eq`), компрессор (`apply_compressor`), реверберация (`apply_reverb`). Оптимизация: использует

numba.jit для apply\_compressor\_numba и apply\_reverb\_numba, буферизация состояний фильтров.

6. apply\_noise\_reduction(self, data: np.ndarray): применяет алгоритм шумоподавления к аудиоданным. Поддерживает несколько методов шумоподавления: spectral - спектральное шумоподавление (использует noisereduce), wiener - фильтр Винера, median - медианный фильтр, gaussian - гауссовский фильтр.

7. apply\_eq(self, data: np.ndarray). Для каждой полосы эквалайзера (50Hz, 150Hz, ... 15kHz): рассчитывает БИХ-фильтр через scipy.signal.butter, Применяет фильтр с учетом усиления через lfilter.

8. apply\_multiband\_compressor(self, data: np.ndarray). Логика работы: разделяет сигнал на 3 полосы (НЧ/СЧ/ВЧ) через \_apply\_bandpass и для каждой полосы: вычисляет огибающую и применяет пороговое сжатие с учётом атаки/спада. Возвращает сумму обработанных полос.

9. apply\_reverb(self, data: np.ndarray). Логика работы: смешивает входной сигнал с задержанной версией, регулирует ослабление задержанного сигнала (damping) и фильтруется (ФНЧ/ФВЧ), результат смешивается с исходным сигналом по параметрам wet и dry.

#### 4.1.2 Класс AudioApp

Главный класс GUI приложения. Содержит интерфейс для управления аудиоэффектами, визуализации сигналов и взаимодействия с AudioProcessor. Реализует загрузку/сохранение файлов и управление воспроизведением.

Основные методы:

1. \_\_init\_\_(self): создает главное окно Tkinter с вкладками (ttk.Notebook), настраивает стили (ttk.Style) для основной темы приложения, запускает фоновую инициализацию процессора через \_start\_background\_init.

2. \_create\_main\_interface(self): кнопки загрузки, управления воспроизведением, сохранением, применением шумоподавления, ползунок позиции, два графика через FigureCanvasTkAgg, вкладки с регуляторами для эквалайзера, компрессора, реверберации.

3. `update_gui(self)`: основной метод обновления графического интерфейса. Выполняет: обновление позиции воспроизведения (10 FPS), обновление графиков сигналов (15 FPS), проверку состояния воспроизведения, обработку очереди сообщений для визуализации.

4. `_audio_callback(self, outdata, frames, time_info, status)`: получает аудиоблок из `processor.apply_all_effects()`, отправляет данные в `outdata` для звуковой карты, помещает сэмплы в `plot_queue` для визуализации.

#### 4.1.3 Класс Knob

Кастомный виджет регулятора в виде крутящейся ручки, реализованный на `tk.Canvas`. Позволяет интерактивно изменять числовые параметры с помощью вращения мыши и обеспечивает визуальную обратную связь.

Основные методы:

1. `_on_drag(self, event)`: обрабатывает перемещение мыши, вращая регулятор.
2. `_update_pointer(self, angle)`: перерисовывает стрелку на `Canvas`.

#### 4.1.4 Класс App

Главный класс-обёртка приложения, отвечающий за инициализацию среды, обработку ошибок и запуск основного GUI (`AudioApp`). Реализует проверку зависимостей, глобальный обработчик исключений и настройку приоритетов процессов для стабильной работы аудио.

Основной метод `main()`: настраивает обработчик исключений (`sys.excepthook`), проверяет зависимости (`check_dependencies()`), создает экземпляр `AudioApp` и запускает главный цикл.

### 4.2 Описание элементов интерфейса пользователя

На рисунке 4.1 интерфейс программы при запуске.

1. Окно отображения графика формы волны.
2. Окно отображения графика АЧХ.
3. Кнопка загрузки аудиофайла (активно).

4. Кнопка воспроизведения аудио (активно).
5. Кнопка остановки аудио (неактивна).
6. Кнопка сохранения аудио (неактивно).
7. Кнопка шумоподавления (неактивно).
8. Knob регулировки уровня шумоподавления.
9. Отображение процентов воздействия шумоподавления на аудио.
10. Слайдер отслеживания и управление временем воспроизведения.
11. Время аудио нынешнее/полное.
12. Активная вкладка открытого окна эквалайзера.
13. Неактивная вкладка закрытого окна компрессора.
14. Неактивная вкладка закрытого окна реверберации.
15. Чекбокс включения эквалайзера.
16. Слайдер регулировки уровня частоты 50 Гц с подписанным снизу параметром настройки.
17. Слайдер регулировки уровня частоты 150 Гц с подписанным снизу параметром настройки.
18. Слайдер регулировки уровня частоты 250 Гц с подписанным снизу параметром настройки.
19. Слайдер регулировки уровня частоты 500 Гц с подписанным снизу параметром настройки.
20. Слайдер регулировки уровня частоты 1 кГц с подписанным снизу параметром настройки.
21. Слайдер регулировки уровня частоты 3 кГц с подписанным снизу параметром настройки.
22. Слайдер регулировки уровня частоты 7 кГц с подписанным снизу параметром настройки.
23. Слайдер регулировки уровня частоты 15 кГц с подписанным снизу параметром настройки.

На рисунке 4.2 окно загрузки аудиофайла.

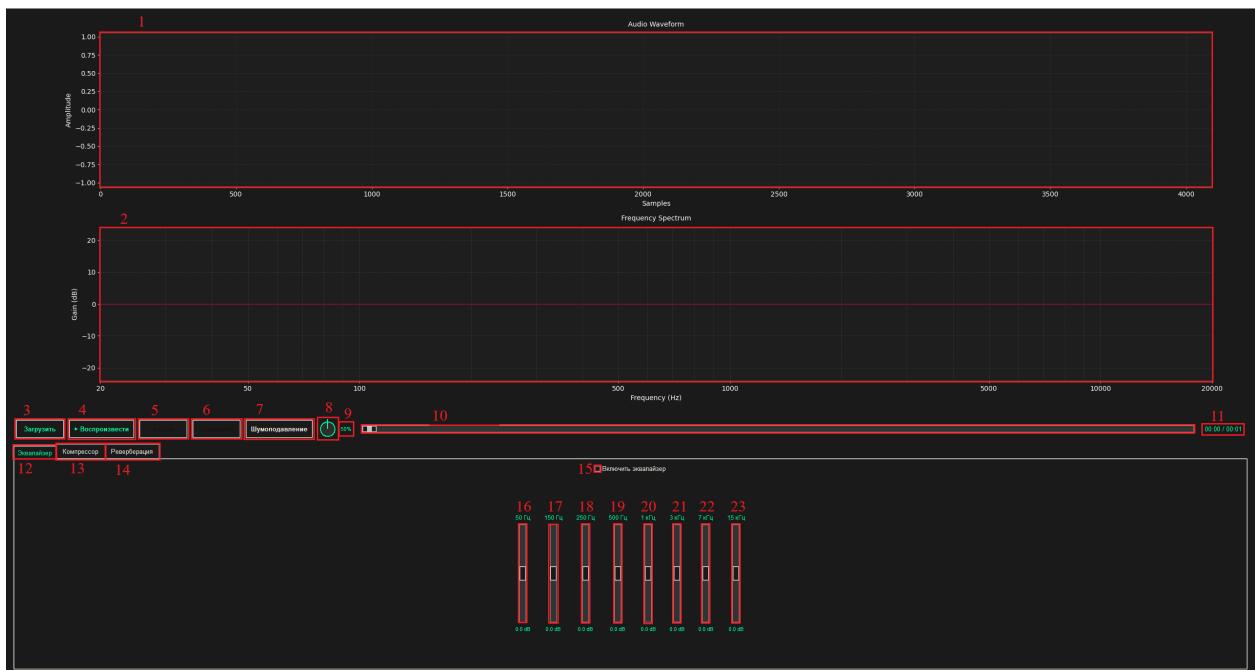


Рисунок 4.1 – Интерфейс приложения при запуске.

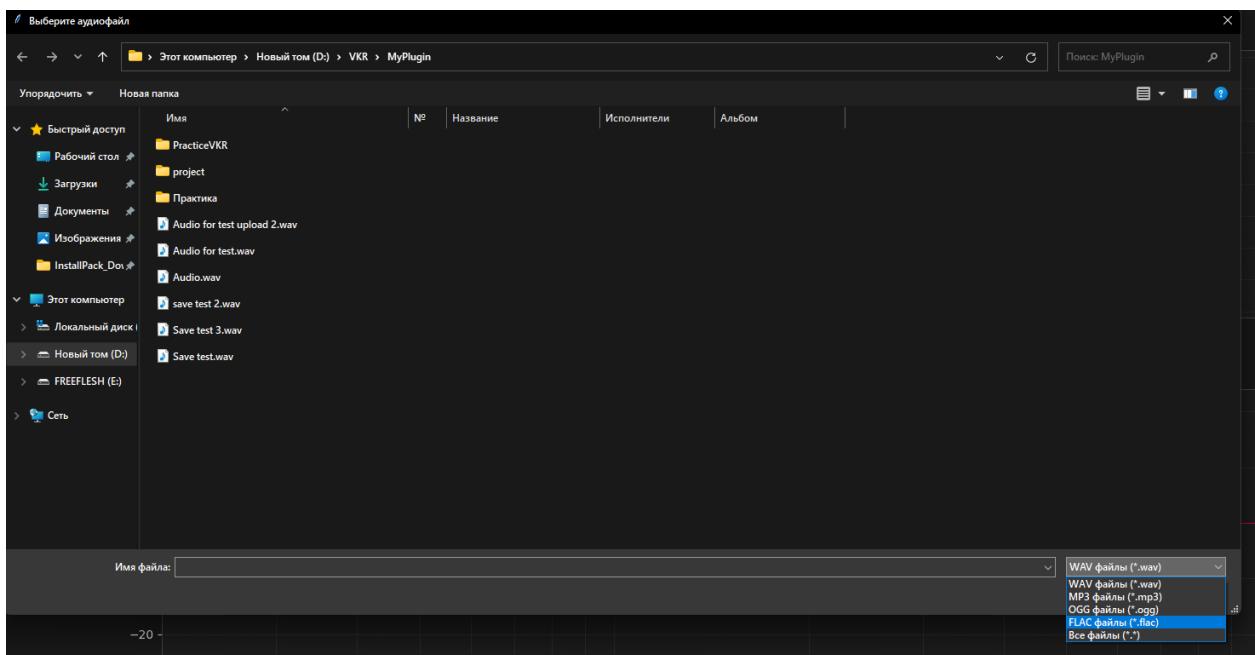


Рисунок 4.2 – Окно загрузки аудиофайла.

На рисунке 4.3 интерфейс вкладки компрессора.

1. Кнопка воспроизведения аудио (неактивно).
2. Кнопка остановки аудио (активно).
3. Кнопка сохранения аудио (активно).
4. Активная кнопка шумоподавления.
5. Окно регулировки диапазонов обработки частот.

6. Ползунок регулировки диапазона низких частот.
7. Ползунок регулировки диапазона высоких частот.
8. Подпись диапазонов.
9. Неактивный чекбокс bypass.
10. Knob регулировки Threshold с подписью значения настройки (аналагично для всех диапазонов).
11. Knob регулировки Ratio с подписью значения настройки (аналагично для всех диапазонов).
12. Knob регулировки Knee с подписью значения настройки (аналагично для всех диапазонов).
13. Knob регулировки Attack с подписью значения настройки (аналагично для всех диапазонов).
14. Knob регулировки Release с подписью значения настройки (аналагично для всех диапазонов).
15. Knob регулировки Gain с подписью значения настройки (аналагично для всех диапазонов).
16. Чекбокс активного включения компрессора.
17. Чекбокс активного bypass.



Рисунок 4.3 – Интерфейс вкладки компрессора.

На рисунке 4.4 интерфейс вкладки реверберации.

1. Knob регулировки Wet с подписью значения настройки.
2. Knob регулировки Dry с подписью значения настройки.
3. Knob регулировки Size с подписью значения настройки.
4. Knob регулировки Damping с подписью значения настройки.
5. Knob регулировки High Cut с подписью значения настройки.

6. Knob регулировки Low Cut с подписью значения настройки.
7. Knob регулировки Pan с подписью значения настройки.
8. Чекбокс активации реверберации (неактивно).

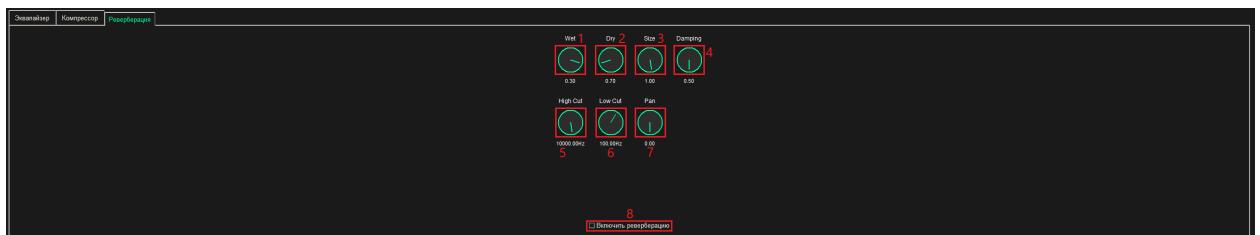


Рисунок 4.4 – Интерфейс вкладки реверберации.

На рисунке 4.5 окно сохранения аудиофайла.

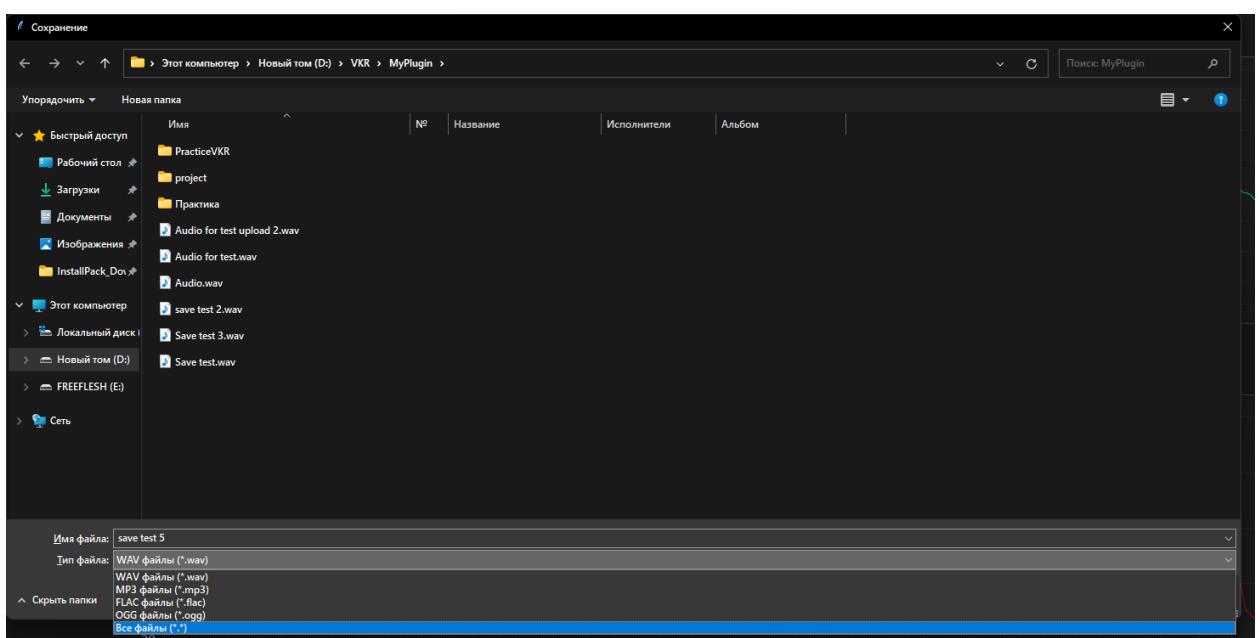


Рисунок 4.5 – Окно сохранения аудиофайла.

### 4.3 Тестирование программно-информационной системы

Тестирование аудиопроцессора — комплексный процесс, включающий проверку корректности работы всех модулей, качество обработки звука, устойчивость к ошибкам и соответствие требованиям. Для обеспечения высокого качества и надёжности программно-информационной системы необходимо провести как позитивное, так и негативное тестирование с использованием различных методов и сценариев.

## 1) Запуск приложения

Описание: Пользователь открывает приложение и оно запускается без ошибок и отображает рабочий интерфейс.

На рисунке 4.6 загрузка приложения.

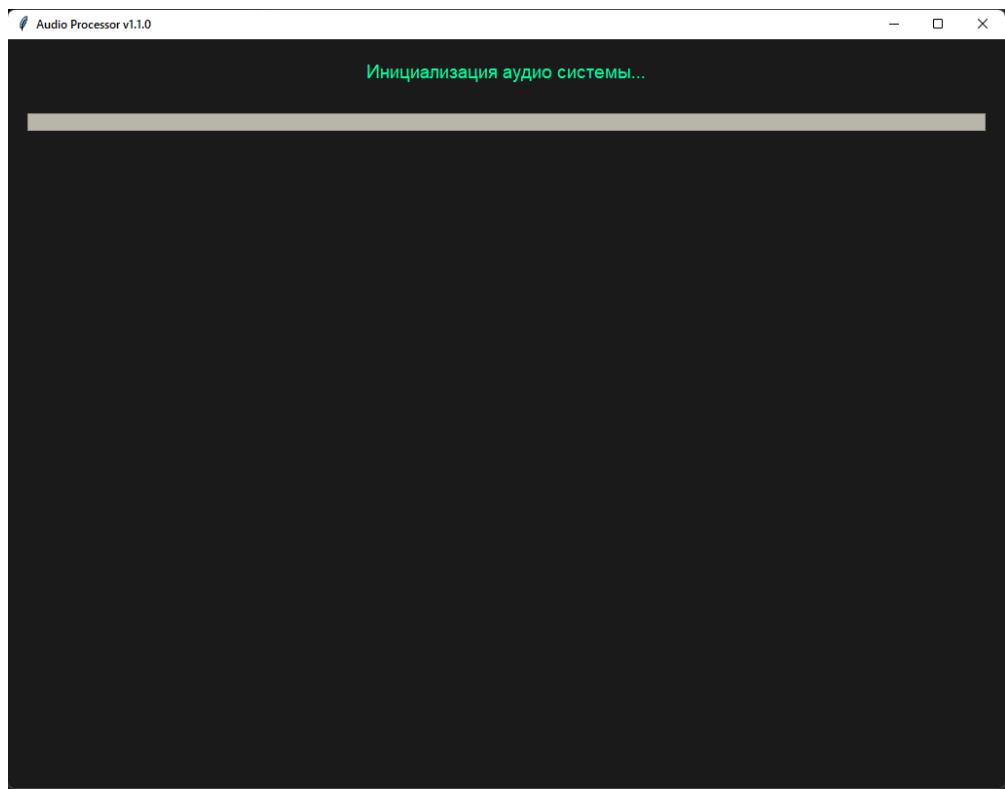


Рисунок 4.6 – Окно загрузки приложения.

На рисунке 4.7 интерфейс программы.

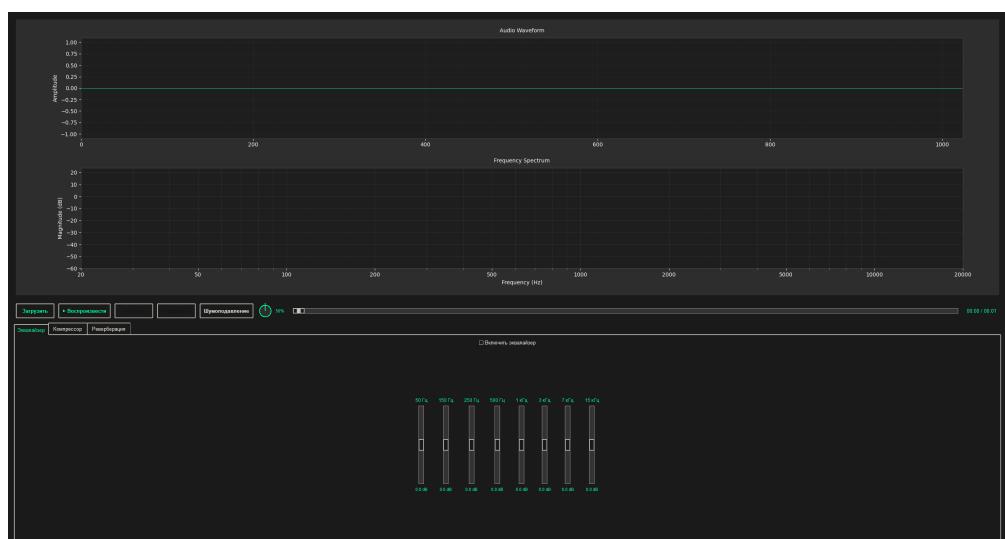


Рисунок 4.7 – Интерфейс программы.

## 2) Загрузка аудиофайла

Описание: Корректная загрузка аудиофайлов различных форматов (WAV, MP3, FLAC).

На рисунке 4.8 окно загрузки аудиофайла.

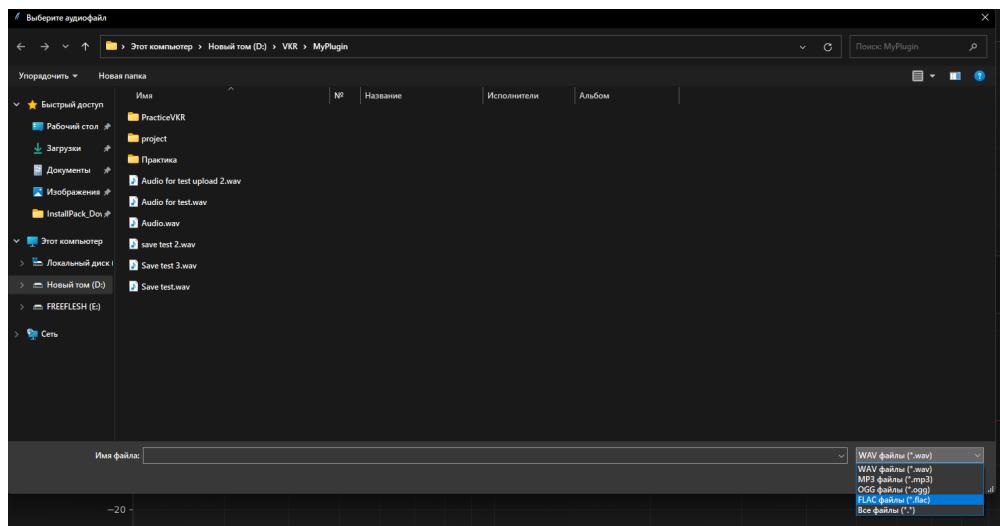


Рисунок 4.8 – Окно загрузки аудиофайла.

На рисунке 4.9 интерфейс программы при успешной загрузки аудиофайла.

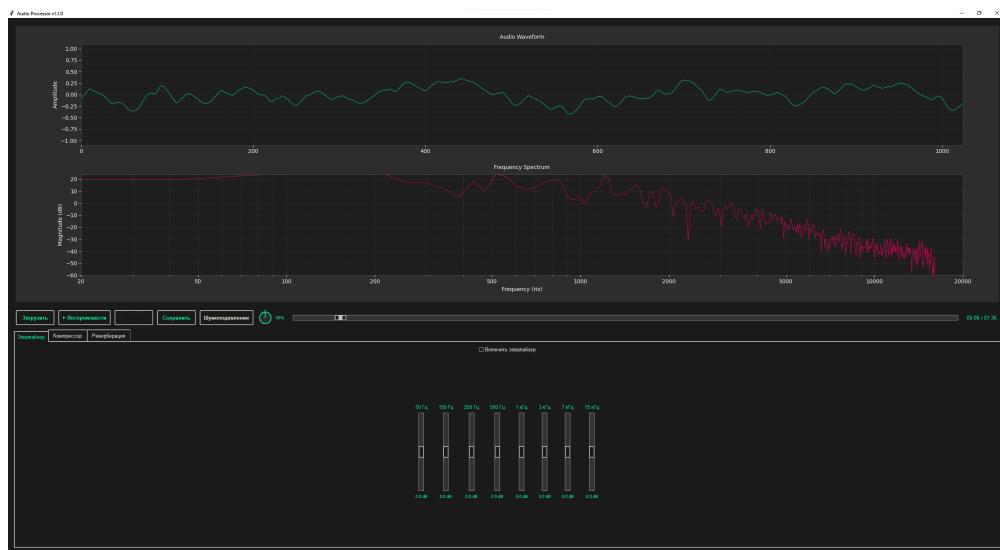


Рисунок 4.9 – Интерфейс программы при успешной загрузки аудиофайла.

### 3) Корректное воспроизведение и остановка аудио

Описание: При воспроизведении и остановки проигрывания аудио, соответственно, воспроизводится и останавливается. Так же перемещается ползунок метки воспроизведения в соответствии времени воспроизведения и при перемещении пользователем, аудио воспроизводится с момента, на который переместил пользователь.

На рисунке 4.10 перемещение ползунка при воспроизведении.



Рисунок 4.10 – Интерфейс программы при перемещении ползунка управления воспроизведением.

На рисунке 4.11 остановка воспроизведения аудио.

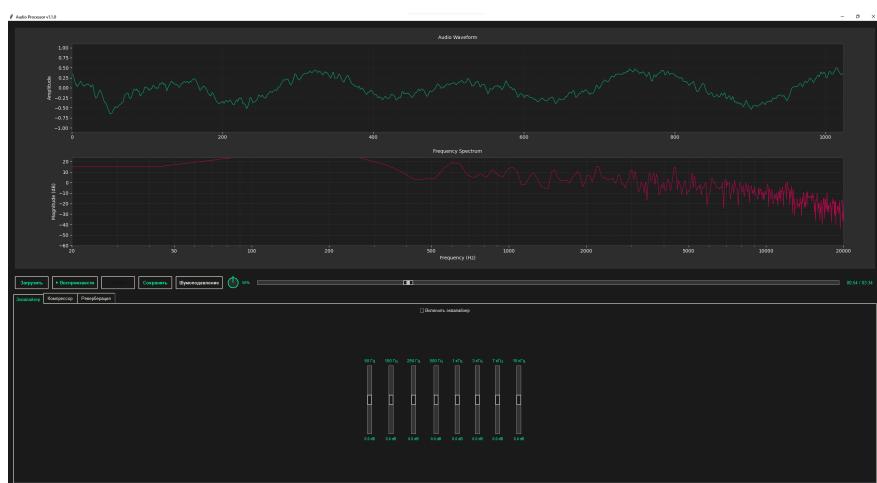


Рисунок 4.11 – Интерфейс программы при остановки воспроизведения аудио.

## 4) Обработка аудиосигнала в реальном времени

Описание: При воспроизведении аудиофайла графики формы волны и АЧХ отображают данные аудио в реальном времени.

На рисунке 4.12 отображение графиков формы волны и АЧХ в реальном времени.



Рисунок 4.12 – Отображение графиков.

На рисунке 4.13 отображение графиков формы волны и АЧХ при тишине в аудиофайле.

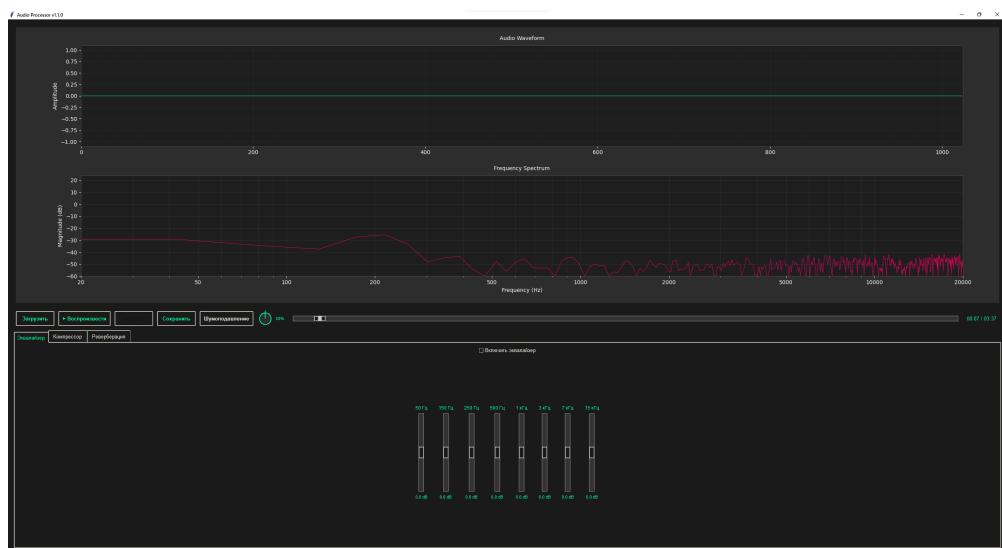


Рисунок 4.13 – Отображение графиков при тишине в аудиофайле.

## 5) Изменение параметров эквалайзера

Описание: При изменении параметров эквалайзера, регулировки ползунков, все изменения применяются сразу, без задержек и прерываний. Так же изменяются графики формы волны и АЧХ.

На рисунке 4.14 изменение параметров эквалайзера.



Рисунок 4.14 – Изменение параметров эквалайзера.

На рисунке 4.15 отображение графиков при выключенном эквалайзере.

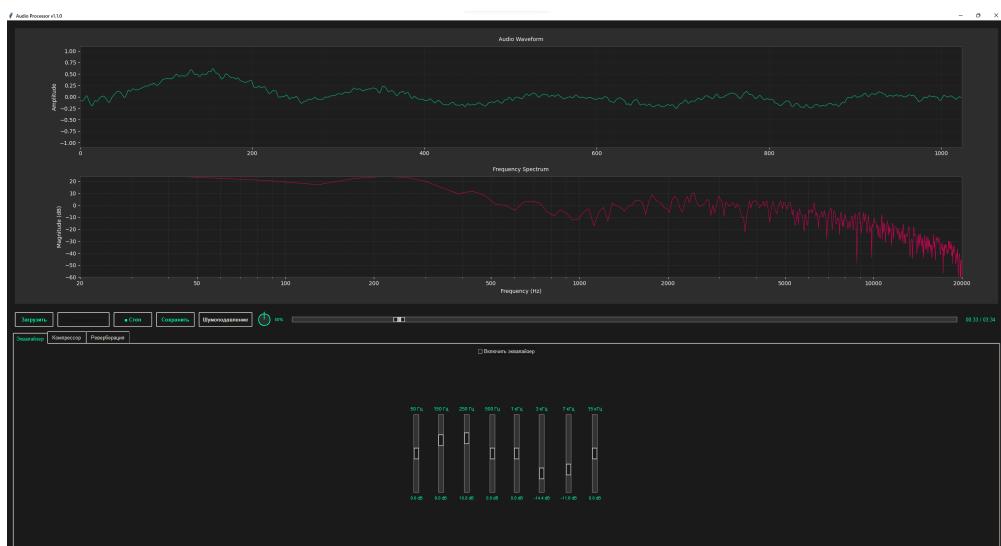


Рисунок 4.15 – Отображение графиков при выключенном эквалайзере.

## 6) Изменение параметров компрессора

Описание: При изменении параметров компрессора, регулировки knob, все изменения применяются сразу, без задержек и прерываний. При включении/выключении функции bypass она работает корректно, то есть пропускает исходный сигнал выбранной зоны. Ползунки выборания зон работают исправно и частотные диапазоны выделяются корректно. Так же изменяются графики формы волны и АЧХ.

На рисунке 4.16 изменение параметров компрессора.

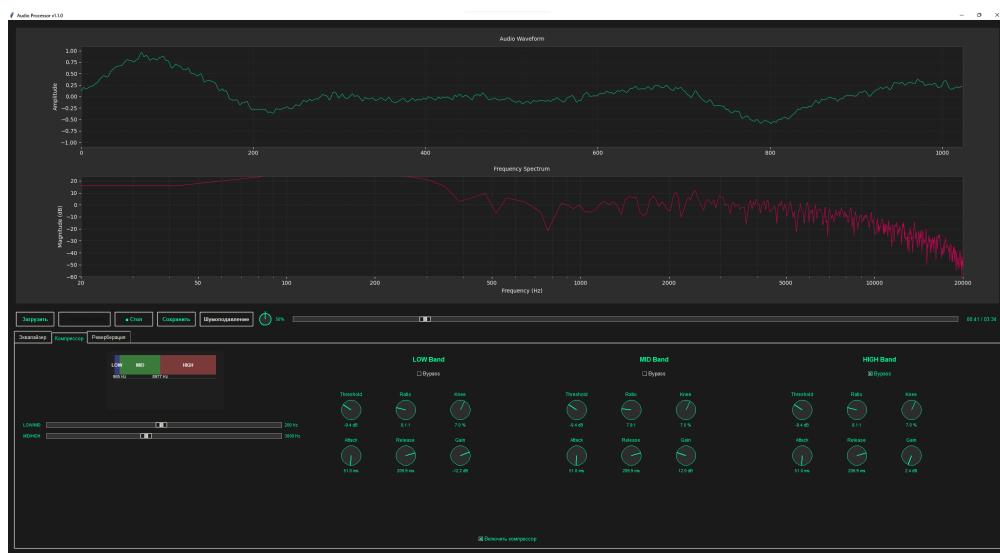


Рисунок 4.16 – Изменение параметров компрессора.

На рисунке 4.17 отображение графиков при выключенном компрессоре.

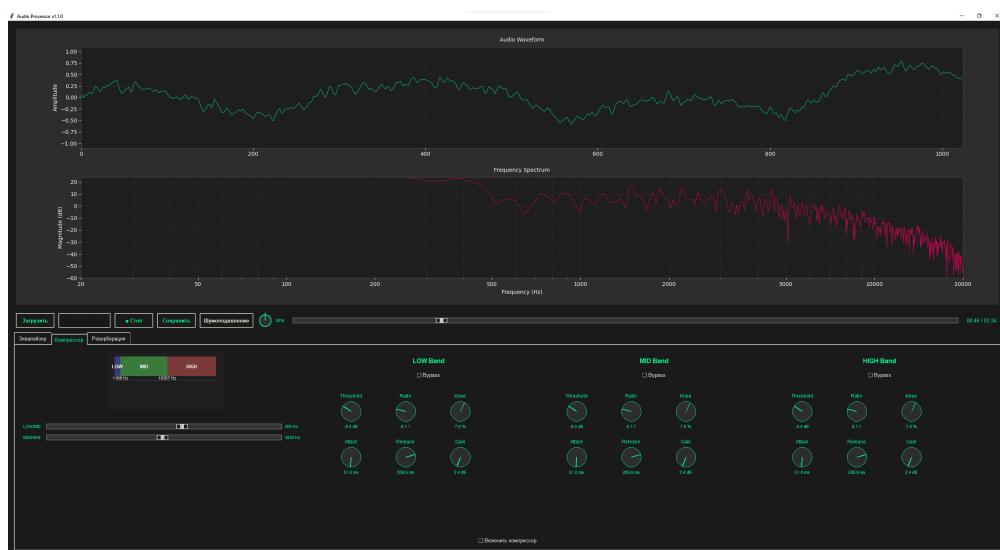


Рисунок 4.17 – Отображение графиков при выключенном компрессоре.

## 7) Изменение параметров реверберации

Описание: При изменении параметров реверберации, регулировки knob, все изменения применяются сразу, без задержек и прерываний. Так же изменяются графики формы волны и АЧХ.

На рисунке 4.18 изменение параметров компрессора.

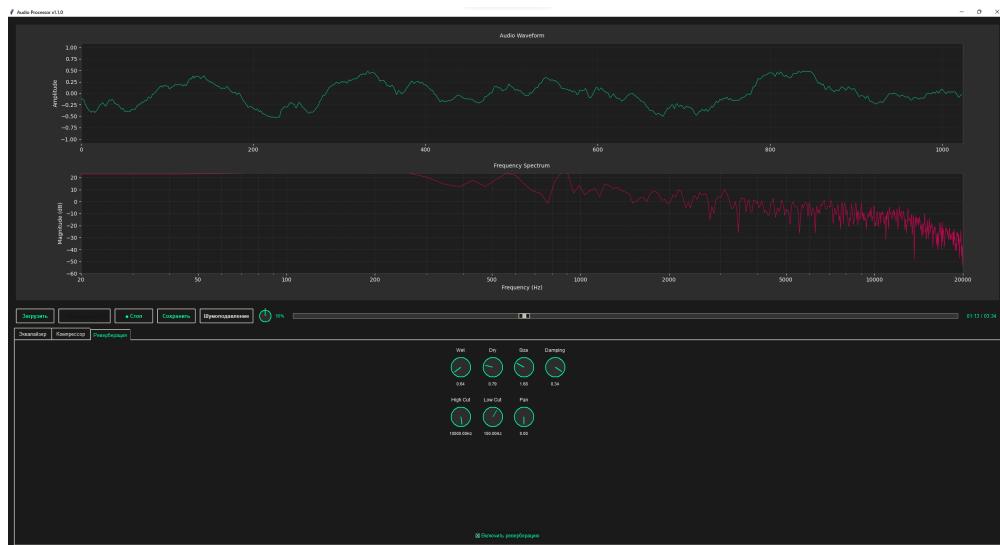


Рисунок 4.18 – Изменение параметров реверберации.

На рисунке 4.19 отображение графиков при выключенной реверберации.

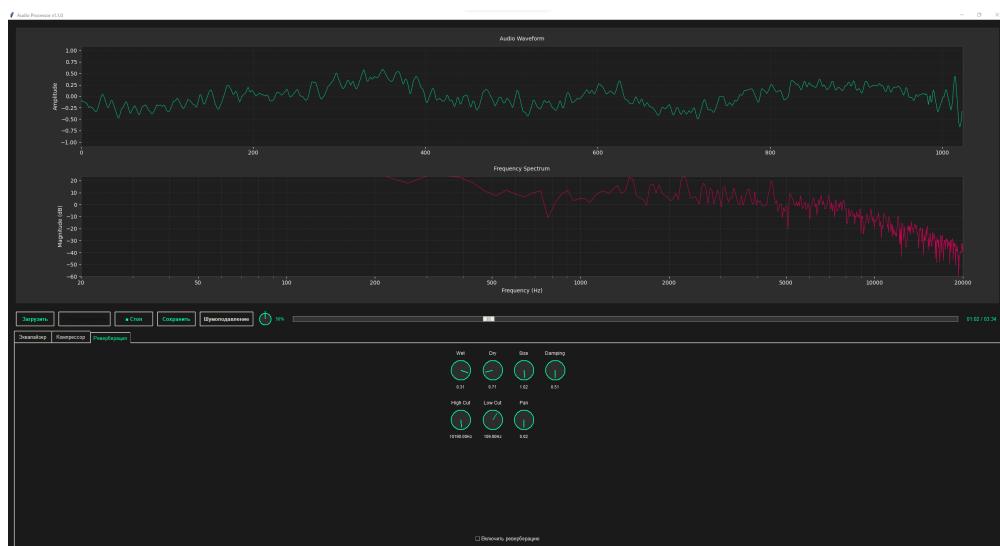


Рисунок 4.19 – Отображение графиков при выключенной реверберации.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной дипломной работы был разработан программный аудиопроцессор на языке Python, способный выполнять многополосную обработку аудиосигнала в реальном времени с применением современных эффектов, таких как компрессия, эквалайзация, реверберация и шумоподавление. В процессе работы был создан модуль архитектуры, произведена оптимизация производительности и реализована возможность гибкой настройки параметров обработки.

Задачи, поставленные в начале разработки были решены следующим образом:

- провёл анализ предметной области и существующих решений;
- спроектировал архитектуру программно-информационной системы, обеспечивающую модульность, расширяемость и эффективное взаимодействие всех компонентов;
- реализовал модуль обработки и воспроизведения аудиосигнала с поддержкой работы в реальном времени, реализовал основные эффекты обработки звука;
- создал удобный и понятный интерфейс приложения с отображением графиков формы волны и АЧХ в реальном времени и в соответствии с обработанными данными;
- обеспечена оптимизацию вычислений и минимизацию задержек.

Особое внимание уделялось оптимизации производительности с использованием технологий ЛТ-компиляции (Numba) и многопоточности, что позволило обеспечить обработку аудиосигнала в реальном времени с минимальной задержкой. Разработанный аудиопроцессор продемонстрировал высокое качество звуковой обработки, гибкость настройки параметров и устойчивость к ошибкам.

Результаты работы могут быть использованы как основа для дальнейшего развития программных аудиопроцессоров, расширения функционала и интеграции с другими аудиосистемами. Практическая значимость проекта

заключается в создании универсального инструмента для динамической обработки звука, который может применяться в профессиональной звукозаписи, радиовещании, а также в бытовых аудиоприложениях.

Таким образом, поставленные цели и задачи дипломной работы полностью достигнуты, а разработанная система соответствует современным требованиям к качеству и эффективности цифровой обработки аудиосигналов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Витязев В.В., Волченков В.А. Цифровая обработка сигналов – Москва: Лань, 2023 – 448 с., ISBN 978-5-8114-5804-9 – Текст: непосредственный.
2. Шиндор О.В., Чикрин Д.Е., Кокунин П.А. Цифровая обработка сигналов – Москва: Лань, 2025 – 400 с., ISBN 978-5-8114-5810-0 – Текст: непосредственный
3. Григорьев Д.Л. Цифровая обработка аудиосигналов / Д.Л. Григорьев – Москва: ДМК Пресс, 2018 – 352 с., ISBN 978-5-97060-670-6 – Текст: непосредственный.
4. Брайан Джонс, Тиан Чжан. Python. Карманный справочник / Brian Jones, Tiaan Chan – Санкт-Петербург: Питер, 2021 – 320 с., ISBN 978-5-4461-1640-7 – Текст: непосредственный.
5. Брайан Кинг. Python и обработка аудиосигналов / King B. – Москва: ДМК Пресс, 2020 – 288 с., ISBN 978-5-97060-960-8 – Текст: непосредственный.
6. Кузнецов Ю.А. Теория и практика цифровой обработки сигналов – Москва: Горячая линия – Телеком, 2015 – 368 с., ISBN 978-5-9912-0575-6 – Текст: непосредственный.
7. Блинов М.И. Программирование цифровых фильтров – Санкт-Петербург: Питер, 2018 – 224 с., ISBN 978-5-4461-1278-2 – Текст: непосредственный.
8. Громов В.А. Программирование аудиоприложений на Python – Санкт-Петербург: Питер, 2022 – 256 с., ISBN 978-5-4461-1642-1 – Текст: непосредственный.
9. Капустин С.А. Теория и практика цифровой фильтрации – Москва: Физматлит, 2017 – 296 с., ISBN 978-5-9221-1767-2 – Текст: непосредственный.

10. Кулагин В.П. Программирование на Python для инженеров – Москва: ДМК Пресс, 2020 – 320 с., ISBN 978-5-97060-960-8 – Текст: непосредственный
11. Тихонов А.Н. Аудиообработка в мультимедийных приложениях - М.: Горячая линия-Телеком, 2021 - 408 с., ISBN 978-5-9912-0791-0 - Текст: непосредственный.
12. Васнецов П.А. Программирование аудиоэффектов на Python - Екатеринбург: УрФУ, 2022 - 276 с., ISBN 978-5-321-02497-3 - Текст: непосредственный.
13. Семенов А.Ю. Реализация аудиоэффектов на языке Python - Москва: ДМК Пресс, 2022 - 352 с., ISBN 978-5-93700-107-8 - Текст: непосредственный.
14. Федоров Р.А. Эквалайзеры и фильтры в аудиоприложениях - Новосибирск: НГТУ, 2019 - 212 с., ISBN 978-5-7782-3784-9 - Текст: непосредственный.
15. Волков Л.А. Реверберация и пространственная обработка звука - Санкт-Петербург.: Лань, 2017 - 320 с., ISBN 978-5-8114-2452-3 - Текст: непосредственный.
16. Николаев А.Г., Соколов Е.В. Алгоритмы компрессии аудиосигналов - Москва: Радио и связь, 2018 - 256 с., ISBN 978-5-256-01987-6 - Текст: непосредственный.
17. Гришин В.О. Цифровые аудиоэффекты: теория и реализация - Москва: Горячая линия-Телеком, 2019 - 364 с., ISBN 978-5-9912-0729-3 - Текст: непосредственный
18. Иванов П.К. Современные методы цифровой обработки звука - Москва: Техносфера, 2020 - 416 с., ISBN 978-5-94836-567-1 - Текст: непосредственный.
19. Смирнов Д.А., Козлов С.В. Алгоритмы шумоподавления в аудиосистемах - Москва: ДМК Пресс, 2019 - 288 с., ISBN 978-5-97060-689-4 - Текст: непосредственный.

20. Соколов Д.В. Многопоточная обработка аудиосигналов - Москва: Радио и связь, 2021 - 320 с., ISBN 978-5-256-02245-6 - Текст: непосредственный.

## ПРИЛОЖЕНИЕ А

### Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.8.

# Сведения о ВКРБ

# Минобрнауки России

## Юго-Западный государственный университет

## Кафедра программной инженерии

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО ПРОГРАММЕ БАКАЛАВРИАТА

८

## «Разработка программного обеспечения для анализа и улучшения аудиозаписей»

Руководитель ВКРБ  
к.т.н, доцент  
Ефремова Ирина Николаевна

Автор ВКРБ  
студент группы ПО-116  
Матвеев Игорь Олегович

|   |                |                             |  |  |
|---|----------------|-----------------------------|--|--|
|   |                | ВКРБ 210616.09.03.04.25.018 |  |  |
|   |                | Сведения о ВКРБ             |  |  |
|   |                | Аль. №еся №е                |  |  |
| Фамилия И. О.                               | Полное имя     |                             |  |  |
| Алько Роберт                                | Медведев И. О. |                             |  |  |
| Родственник                                 | Сергеева Н. Н. |                             |  |  |
| Наследник                                   | Чернов А. А.   |                             |  |  |
| Бытовое обстоятельство на рабочем блюдохаре |                |                             |  |  |
| ДБЧУ ПО-16                                  |                |                             |  |  |

### Рисунок А.1 – Сведения о ВКРБ

## Цель и задачи разработки

Целью реализованной квалификационной работы является разработка программного обеспечения для анализа и улучшения аудиозаписей. Необходимо, чтобы приложение реализовывало стабильную загрузку и обработку аудиофайлов, отображало графики формы волны и амплитудно-частотной характеристики в реальном времени, сохраняло обработанные данные в различных форматах. Система включает в себя основные функции для качественной регулировки параметров аудио, что позволяет быстро и точно добиться желаемого результата.

Для реализации поставленной цели были сформулированы следующие задачи:

- 1) Провести анализ предметной области и существующих решений.
  - 2) Спроектировать архитектуру программной системы, обеспечивающую модульность, расширяемость и эффективное взаимодействие всех компонентов.
  - 3) Реализовать модуль обработки и воспроизведения аудиосигнала с поддержкой работы в реальном времени, реализовать основные эффекты обработки звука.
  - 4) Создание удобного и понятного интерфейса приложение с отображение графиков формы волны и АЧХ в реальном времени и в соответствии с обработанными данными.
  - 5) Обеспечить оптимизацию вычислений и минимизацию задержек.

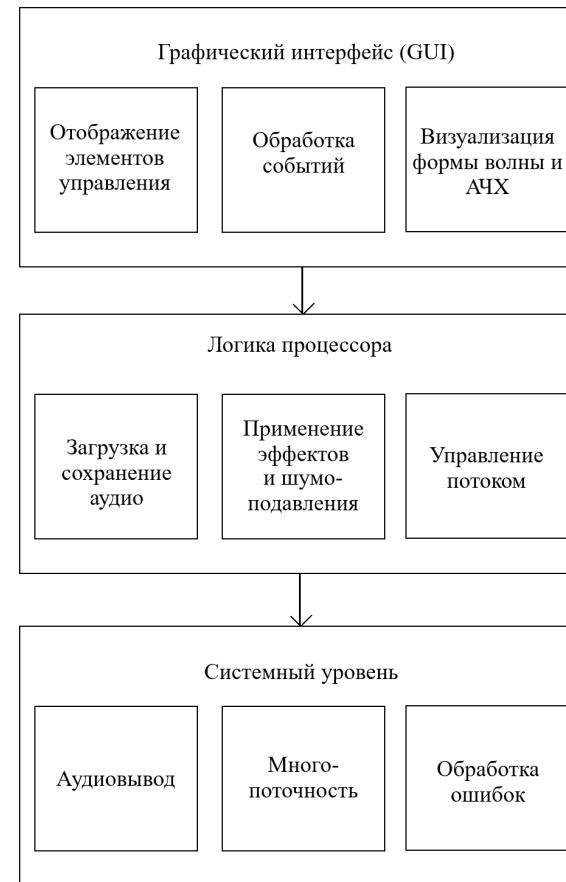
|                          |  |                              |       |          |
|--------------------------|--|------------------------------|-------|----------|
|                          |  | ВКРБ 2106160.09.03.04.25.018 |       |          |
|                          |  | Лин.                         | Номер | Название |
| Цели и задачи разработки |  |                              |       |          |
|                          |  | Анал. 2 / Анал. 1            |       |          |
| Выпуклая конформационная |  | ВЭГУ №-116                   |       |          |
| рабочая база             |  |                              |       |          |

## Рисунок А.2 – Цели и задачи разработки



Рисунок А.3 – Диаграмма прецедентов

## Архитектура приложения



#### Рисунок А.4 – Архитектура приложения

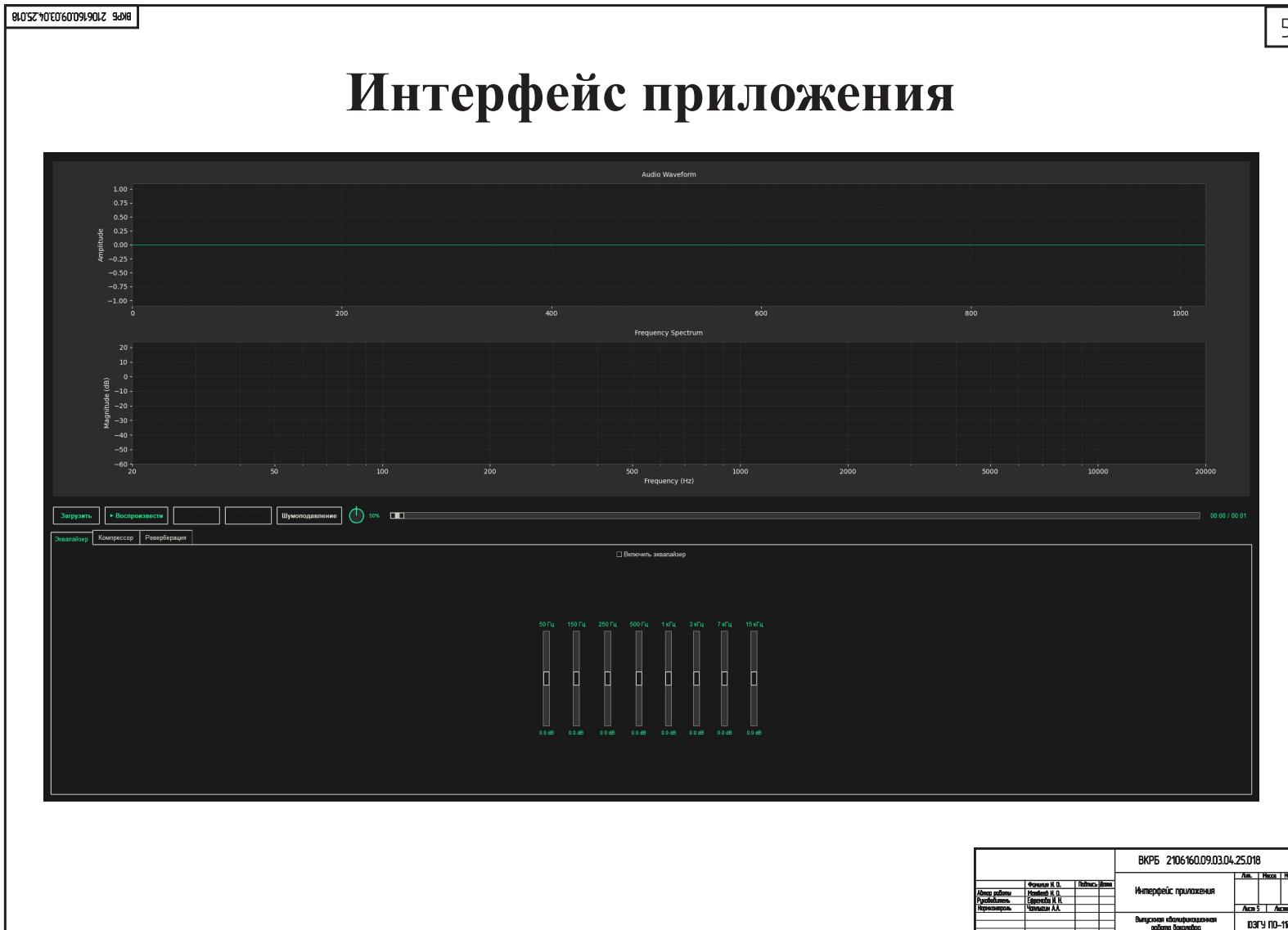


Рисунок А.5 – Интерфейс программы

# Интерфейс вкладки компрессора и графиков

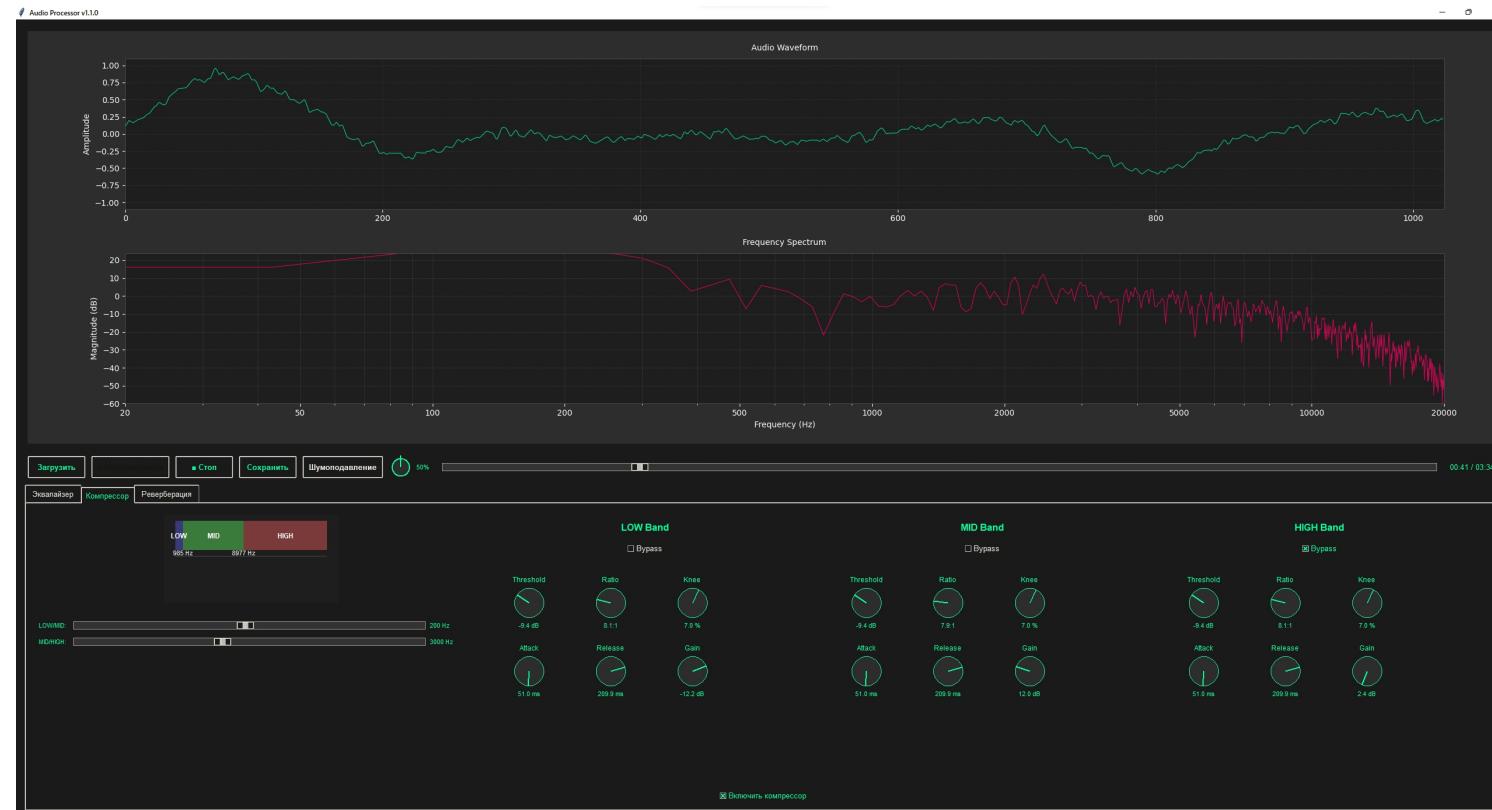


Рисунок А.6 – Интерфейс вкладки компрессора и графиков

|               |                |   |      |
|---------------|----------------|---|------|
|               |                | ВКРБ 210616.09.03.04.25.018                 |      |
|               |                | Интерфейс єквалайзера компресора і графіків |      |
|               |                | Вимірювання функціональності роботи блоків  |      |
| Адреса роботи | Фролова Н. О.  | Підпись/Ім'я                                | Анн. |
|               | Неструєв І. О. |   | І.   |
| Ідентифікатор | Іванова Н.     | Підпись/Ім'я                                | І.   |
|               | Черкаський А.  |   | І.   |
|               |                | ІзГУ ПД-116                                 |      |

# Интерфейс вкладки реверберации и графиков

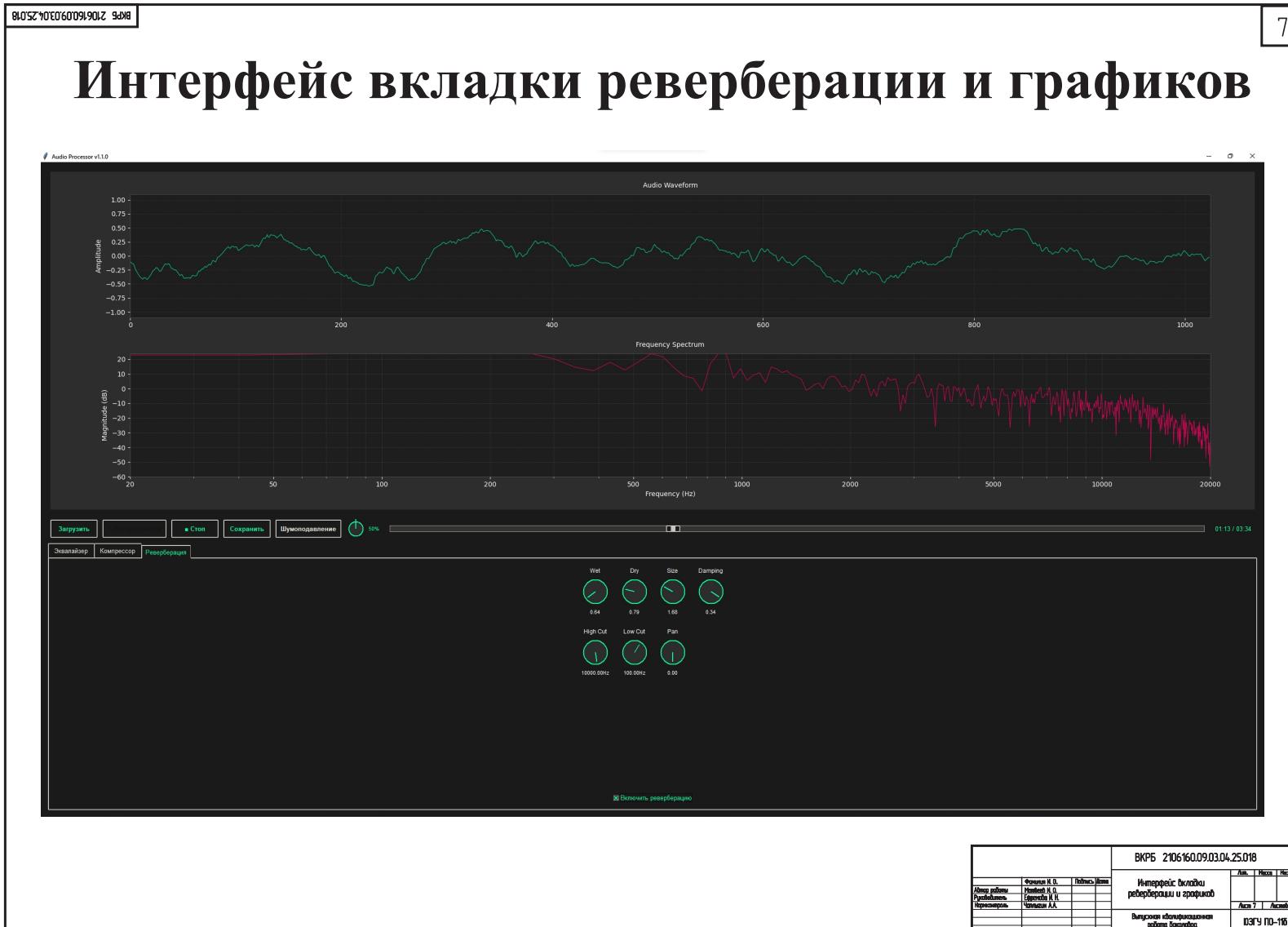


Рисунок А.7 – Интерфейс вкладки реверберации и графиков

## Заключение

В ходе выполнения выпускной квалификационной работы было разработано программное обеспечение для анализа и улучшений аудиозаписей. Реализована стабильная загрузка и обработка аудиофайлов, отображение графиков формы волны и амплитудно-частотной характеристики в реальном времени, сохранение обработанных данных в различных форматах. Созданы основные функции для качественной регулировки параметров аудио.

Указанные в техническом задании требования были успешно реализованы и задачи, поставленные в начале разработки были решены следующим образом:

- провёл анализ предметной области и существующих решений;
  - спроектирована архитектура программно-информационной системы, обеспечивающая модульность, расширяемость и эффективность взаимодействия всех компонентов;
    - реализован модуль обработки и воспроизведения аудиосигнала с поддержкой работы в реальном времени, реализованы основные эффекты обработки звука;
    - создан удобный и понятный интерфейс приложения с отображением графиков формы волны и АЧХ;
    - обеспечена оптимизация вычислений и минимизация задержек.

|               |                       |  |
|---------------|-----------------------|--|
|               |                       | ВКРБ 2106160.09.03.04.25.016                   |
| Фамилия И. О. | Полность Имя          | Линия И. О.                                    |
| Мельник И. О. | Мельник Илья Олегович | Линия И. О.                                    |
| Родственники  | Борисов И. Н.         | Линия И. О.                                    |
| Наследники    | Чечеткин А.А.         | Линия И. О.                                    |
|               |                       | Заключение                                     |
|               |                       | Выпускное квалификационное<br>работа бакалавра |
|               |                       | ИЗГУ П-15                                      |

## Рисунок А.8 – Заключение

## ПРИЛОЖЕНИЕ Б

### Фрагменты исходного кода программы

#### main.tex

```
1 \input{setup.tex}
2
3 % Режим шаблона (должен быть включен один из трех)
4 \BKPtrue
5 %\Практикtrue
6 %\Курсоваяtrue
7
8 \newcommand{\Дисциплина}{<<Проектирование и архитектура программных систем>>}
9   % для курсовой
10 \newcommand{\КодСпециальности}{09.03.04} % Курсовая
11 \newcommand{\Специальность}{Программная инженерия} % Курсовая
12 \newcommand{\Тема}{Программное обеспечение для анализа и улучшения
13   аудиозаписей} % ВКР Курсовая
14 \newcommand{\ТемаВтораяСтрока}={}
15 \newcommand{\ГдеПроводитсяПрактика}{Юго-Западном государственном университете
16   } % для практики
17 \newcommand{\РуководительПрактПредпр}{Куркина А. В.} % для практики
18 \newcommand{\ДолжнРуководительПрактПредпр}{директор} % для практики
19 \newcommand{\РуководительПрактУнивер}{Чаплыгин А. А.} % для практики
20 \newcommand{\ДолжнРуководительПрактУнивер}{к.т.н. доцент} % для практики
21 \newcommand{\Автор}{И. О. Матвеев}
22 \newcommand{\АвторРод}{Матвеев И. О.}
23 \newcommand{\АвторПолностьюРод}{Матвеева Игоря Олеговича} % для практики
24 \newcommand{\Шифр}{21-06-0160}
25 \newcommand{\Курс}{4} % для практики
26 \newcommand{\Группа}{ПО-116}
27 \newcommand{\Руководитель}{И. Н. Ефремова} % для ВКР и курсовой
28 \newcommand{\Нормоконтроль}{А. А. Чаплыгин} % для ВКР
29 \newcommand{\ЗавКаф}{А. В. Малышев} % для ВКР
30 \newcommand{\ДатаПриказа}{«04» апреля 2025~г.} % для ВКР
31 \newcommand{\НомерПриказа}{1696-с} % для ВКР
32 \newcommand{\СрокПредоставления}{«9» июня 2025~г.} % для ВКР, курсового
33 \begin{document}
34 \maketitle
35 \ifПрактика{}\else{
36   \input{ЛистЗадания}
37   \input{Реферат}\fi
38 \tableofcontents
39 \input{Обозначения}
40 \ifПрактика{}\else{\input{Введение}}\fi
41 \input{Анализ}
42 \input{TexЗадание}
43 \input{TexПроект}
44 \ifПрактика{}\else{
45   \input{РабочийПроект}
46   \input{Заключение}\fi
47 \input{СписокИсточников}
```

```
47 \ifБКР{\input{Плакаты}}\fi
48 \ifПрактика{} \else{\input{Код}}\fi
49 \end{document}
```

## TexПроект.tex

```
1 \section{Технический проект}
2
3 \subsection{Общая характеристика решения задачи}
4
5 Задача дипломного проекта заключается в разработке программного
  аудиопроцессора — универсального программного комплекса для обработки
  аудиосигналов в реальном времени на персональном компьютере. Процессор
  реализует цепочку современных аудиоэффектов с возможностью гибкой
  настройки, а также поддерживает многополосную обработку сигнала, что
  позволяет применять различные параметры эффектов к разным частотным
  диапазонам.
6
7 Разработка аудиопроцессора, способного выполнять обработку аудиосигнала с
  применением эквалайзации, многополосной компрессии, реверберации и
  шумоподавления.
8
9 Входные данные:
10 \begin{itemize}
11   \item аудиосигнал в виде цифрового потока;
12   \item настройки эффектов, задаваемые пользователем (параметры эквалайзера,
        компрессора, реверберации и шумоподавления).
13 \end{itemize}
14
15 Выходные данные:
16 \begin{itemize}
17   \item обработанный аудиосигнал, выводимый на устройство воспроизведения или
        сохраняемый в файл;
18   \item возможность мониторинга и визуализации параметров обработки в виде
        отображения формы волны или АЧХ.
19 \end{itemize}
20
21 \subsection{Описание используемых технологий и языков программирования}
22
23 Для разработки аудиопроцессора был выбран язык программирования Python, что
  обусловлено рядом ключевых факторов:
24 \begin{enumerate}
25   \item Простота и читаемость кода. Синтаксис Python близок к псевдокоду, что
        облегчает понимание и сопровождение проекта, а также позволяет быстро
        экспериментировать с алгоритмами обработки звука.
26   \item Широкий набор библиотек для цифровой обработки сигналов. Python
        обладает мощными библиотеками (NumPy, SciPy, pydub и др.), которые
        предоставляют готовые инструменты для работы с аудиоданными, фильтрами,
        преобразованиями и другими DSP-операциями. Это значительно ускоряет
        разработку и повышает надежность кода.
27   \item Возможность оптимизации производительности. Для повышения скорости
        вычислений в критичных местах применяется JIT-компиляция с помощью Numba
        , что позволяет добиться близкой к нативной производительности без
        перехода на более низкоуровневые языки.
```

```

28 \item Поддержка многопоточности и асинхронной обработки. В проекте
используется ThreadPoolExecutor и другие средства стандартной библиотеки
для параллельной обработки аудиоданных, что улучшает отзывчивость и
снижает задержки при работе в реальном времени.
29 \item Гибкость и расширяемость архитектуры. Модульный подход с
использованием классов и четким разделением функционала (например,
отдельные классы для компрессоров, эквалайзеров и ревербераторов)
позволяет легко добавлять новые эффекты и модифицировать существующие.
30 \item Активное сообщество и наличие обучающих материалов. Благодаря
большому количеству статей, учебников и примеров по цифровой обработке
сигналов на Python, разработка и отладка проекта становится более
эффективной.
31 \end{enumerate}
32
33 Таким образом, выбор Python и сопутствующих технологий проектирования
обусловлен их оптимальным сочетанием простоты, функциональности и
производительности, что отвечает требованиям дипломного проекта по
созданию программного аудиопроцессора с возможностью обработки в реальном
времени и расширяемой архитектурой.
34
35 В коде используются следующие библиотеки Python:
36 \begin{itemize}
37 \item NumPy - обеспечивает высокопроизводительные вычисления с многомерными
массивами, что критично для обработки аудиоданных, представленных в
виде временных рядов;
38 \item SciPy - используется для реализации цифровых фильтров (эквалайзеров),
БПФ (анализ спектра) и других математических операций;
39 \item PyDub - библиотека для работы с аудиофайлами, предоставляющая:
простые методы загрузки и сохранения в форматах WAV, MP3, OGG, FLAC,
базовые операции, такие как обрезка, наложение, изменение громкости,
конвертация частоты дискретизации, интеграцию с FFmpeg для поддержки
дополнительных кодеков;
40 \item Soundfile используется для чтения и записи аудиофайлов различных
форматов, обеспечивая удобный доступ к аудиоданным на диске, позволяет
загружать исходные аудиозаписи и сохранять результаты обработки без
потери качества;
41 \item SoundDevice - обеспечивает низкоуровневый доступ к аудиоустройствам
через PortAudio. Ключевые функции: воспроизведение и запись в реальном
времени с минимальной задержкой, поддержка ASIO, WASAPI, Core Audio для
профессиональных аудиоинтерфейсов, гибкая настройка параметров потока:
частота дискретизации, размер буфера, количество каналов;
42 \item Matplotlib - используется для визуализации аудиоданных: построение
осциллограмм (форма сигнала во временной области), отображение АЧХ (
амплитудно-частотных характеристик) с логарифмической шкалой,
интерактивное обновление графиков в реальном времени;
43 \item Threading и concurrent.futures — стандартные библиотеки Python для
организации многопоточной и параллельной обработки;
44 \item Tkinter - стандартная библиотека Python для создания графического
интерфейса. В проекте применяется для: построения основного окна с
вкладками (эквалайзер, компрессор, реверберация), реализации
интерактивных элементов: ползунки, кнопки, метки, интеграции графиков
Matplotlib через FigureCanvasTkAgg;

```

```

45  \item Numba - JIT-компилятор для оптимизации вычислительно сложных участков
     кода: ускорение алгоритмов компрессии и реверберации в ~510 раз,
     поддержка многопоточности.
46 \end{itemize}
47
48 \subsubsection{Кроссплатформенная библиотека FFmpeg}
49
50 FFmpeg — это мощная кроссплатформенная библиотека для обработки мультимедиа,
     используемая в проекте для работы с аудиофайлами различных форматов.
     Взаимодействие с FFmpeg осуществляется через обёртку PyDub, что
     значительно упрощает операции чтения, записи и конвертации аудио.
51
52 Основные функции FFmpeg в проекте:
53 \begin{enumerate}
54     \item Поддержка множества аудиоформатов. Позволяет загружать и сохранять
             файлы в форматах: без сжатия: WAV, AIFF, с потерями: MP3, AAC, OGG, без
             потерь: FLAC, ALAC. Обеспечивает автоматическое определение кодека при
             загрузке.
55     \item Конвертация аудио: изменение частоты дискретизации, преобразование
             между форматами, конвертация числа каналов.
56     \item Нормализация и обработка. Автоматическая регулировка громкости.
57 \end{enumerate}
58
59 FFmpeg был выбран по рядам преимуществ:
60 \begin{itemize}
61     \item универсальность: поддержка 100+ кодеков и контейнеров;
62     \item стабильность: отлаженные алгоритмы декодирования/кодирования;
63     \item производительность: оптимизированные нативные библиотеки;
64     \item гибкость: Возможность тонкой настройки параметров через командные
             опции.
65 \end{itemize}
66
67 \subsection{Эффекты}
68
69 \subsubsection{Устройство шумоподавления}
70
71 Система шумоподавления реализует 4 метода с общим конвейером обработки:
72 \begin{enumerate}
73     \item Спектральное шумоподавление. На основе библиотеки noisereduce,
             анализирует FFT и подавляет шум в частотной области.
74     \item Фильтр Винера. Адаптивная фильтрация во временной области.
75     \item Медианный фильтр. Подавление импульсных шумов.
76     \item Гауссовский фильтр. Сглаживание высокочастотного шума.
77 \end{enumerate}
78
79 Ключевые особенности:
80 \begin{itemize}
81     \item автодетекция шума: система автоматически запоминает шумовой профиль
             из первых 2048 сэмплов;
82     \item гибкая настройка: регулировка силы подавления -(0100) и выбор метода
             под тип шума;
83     \item потокобезопасность: все операции работают с копиями данных и защищены
             блокировками;

```

84 \item оптимизация под реальное время: фиксированный размер FFT-окна (512 точек) для минимальной задержки.

85 \end{itemize}

86

87 \subsubsection{Устройство эквалайзера}

88

89 Эквалайзер — это устройство или программный алгоритм, предназначенный для корректировки амплитудно-частотной характеристики (АЧХ) звукового сигнала. Он позволяет усиливать или ослаблять определённые частотные диапазоны, изменяя тембр звука. Эквалайзер работает, разделяя входной аудиосигнал на несколько частотных полос (диапазонов), каждая из которых обрабатывается отдельно. После обработки всех полос сигналы складываются, и на выходе получается звук с изменённой частотной характеристикой.

90

91 В программе реализован цифровой параметрический эквалайзер, у которого есть три полосы частот, а так же коррекция их по громкости в децибелах. Такие эквалайзеры часто используются в профессиональной звукозаписи и сведении аудио.

92

93 Цифровой IIR-фильтр (Биквадратный, на основе Баттервортта)

94

95 IIR-фильтр (Infinite Impulse Response — бесконечная импульсная характеристика) — это тип цифрового фильтра, который использует обратную связь, благодаря чему может иметь очень крутые склоны АЧХ при малом порядке.

96

97 Биквадратный фильтр (Biquad) — это частный случай IIR-фильтра 2-го порядка, который реализуется с помощью разностного уравнения и часто используется в эквалайзерах из-за своей эффективности.

98

99 Фильтр Баттервортта — один из самых популярных типов фильтров, обеспечивающий максимально гладкую АЧХ в полосе пропускания без пульсаций.

100

101 Этот эквалайзер обеспечивает:

102 \begin{itemize}

103 \item низкую задержку (важно для реального времени);

104 \item минимальные фазовые искажения;

105 \item точную обработку частот;

106 \item RLock для защиты состояний фильтров.

107 \end{itemize}

108

109 \subsubsection{Устройство многополосного компрессора}

110

111 Многополосный компрессор — устройство динамической обработки аудиосигнала, предназначенное для управления динамическим диапазоном звука с разделением сигнала на несколько частотных полос. Основная идея многополосного компрессора заключается в том, что входящий аудиосигнал сначала разделяется на несколько частотных диапазонов с помощью цифровых фильтров, после чего каждая полоса обрабатывается отдельным компрессором с индивидуальными параметрами. Такой подход позволяет более точно и эффективно контролировать динамику звука в каждом частотном диапазоне, сохраняя при этом естественность звучания и минимизируя искажения.

112

113 В реализованном компрессоре разделение сигнала осуществляется с использованием цифровых фильтров, которые выделяют низкочастотную, среднечастотную и высокочастотную полосы. Для каждой полосы создаётся отдельный компрессорный блок, реализованный в виде программного модуля, который обрабатывает сигнал независимо от других полос. Каждый из этих блоков имеет собственные настройки ключевых параметров компрессии:

```
114 \begin{itemize}
115   \item Threshold (порог);
116   \item Ratio (коэффициент компрессии);
117   \item Attack (атака);
118   \item Release (восстановление);
119   \item Knee (характеристика перехода);
120   \item Make-up Gain (компенсационное усиление).
121 \end{itemize}
```

122

123 Реализация JIT-функций для динамического сжатия аудиосигнала.

124

125 Пошаговый алгоритм:

```
126 \begin{itemize}
127   \item вычисление огибающей;
128   \item инициализация gain-массива;
129   \item обработка каждого сэмпла;
130   \item преобразование dB в линейное значение:;
131   \item экспоненциальное сглаживание;
132   \item применение gain к сигналу.
133 \end{itemize}
```

134

135 Особенности оптимизации:

```
136 \begin{itemize}
137   \item векторизованные операции NumPy;
138   \item предварительный расчет коэффициентов;
139   \item пул потоков для фоновой обработки;
140   \item отдельные состояния для каждой полосы;
141   \item отсутствие ветвлений в критическом цикле.
142 \end{itemize}
```

143

144 Порог срабатывания определяет уровень громкости, при превышении которого начинается сжатие сигнала. Коэффициент сжатия задаёт степень уменьшения динамического диапазона, то есть насколько сильно будет снижаться громкость сигналов, превышающих порог. Время атаки регулирует скорость срабатывания компрессора после превышения порога, а время релиза — скорость возврата усиления к исходному уровню после снижения входного сигнала ниже порога, характеристика перехода определяет плавный переход от отсутствия сжатия к активному сжатию сигнала, компенсационное усиление позволяет регулировать сигнал после обработки, чтобы вернуть ему исходную громкость. Эти параметры позволяют гибко настраивать реакцию компрессора на изменения громкости в каждой полосе, обеспечивая плавное и естественное звучание.

145

146 В основе алгоритма компрессии лежит вычисление коэффициента усиления (gain reduction), который применяется к аудиосигналу полосы. Этот коэффициент рассчитывается на основе текущего уровня сигнала и параметров компрессии с учётом сглаживания изменения усиления для предотвращения резких артефактов в звуке. Для повышения производительности вычисления

оптимизированы с помощью JIT-компиляции (Numba), а обработка полос может выполняться параллельно с использованием многопоточности, что обеспечивает минимальную задержку и возможность работы в реальном времени.

147  
148 Таким образом, многополосный компрессор, реализованный в проекте, представляет собой совокупность нескольких параллельных компрессоров, каждый из которых отвечает за свой частотный диапазон и имеет независимые настройки. Это обеспечивает высокую гибкость и качество динамической обработки, позволяя адаптировать параметры под особенности конкретного аудиоматериала и задачи. Использование цифровых фильтров для разделения сигнала и оптимизированных алгоритмов компрессии обеспечивает эффективную работу устройства в реальном времени, что соответствует современным требованиям к аудиопроцессорам.

149  
150 \subsubsection{Устройство реверберации}  
151  
152 FDN (Feedback Delay Network) — это один из самых мощных алгоритмов цифровой реверберации, используемый для создания реалистичных пространственных эффектов. Он основан на множестве линий задержки с обратной связью, образующих сложную сеть, имитирующую отражения в помещении.

153  
154 Принцип работы:  
155 \begin{itemize}  
156 \item входной сигнал разделяется на несколько параллельных линий задержки, каждая из которой задерживает сигнал на разное время;  
157 \item после задержки сигнал проходит через фильтр демпфирования (имитация потери высоких частот);  
158 \item затем он умножается на коэффициенты матрицы обратной связи, определяющей, какая часть сигнала возвращается в каждую линию;  
159 \item обновлённый сигнал снова поступает на вход линий задержки;  
160 \item результирующий реверберационный сигнал формируется как сумма выходов всех линий.  
161 \end{itemize}

162  
163 Применение JIT-компиляции для создания алгоритма реверберации на основе задержки.

164  
165 Пошаговый алгоритм:  
166 \begin{itemize}  
167 \item инициализация буферов;  
168 \item цикл обработки семплов;  
169 \item смешивание сигналов;  
170 \item обновление буфера задержки;  
171 \item циклическое перемещение по буферу.  
172 \end{itemize}

173  
174 Ключевые особенности FDN:  
175 \begin{itemize}  
176 \item реалистичность — лучше имитирует поздние отражения, чем алгоритмы на основе простых задержек;  
177 \item гибкость — можно настраивать время реверберации, демпфирование и пространственность;  
178 \item стабильность — ортогональная матрица гарантирует отсутствие бесконечного нарастания.

```

179 \end{itemize}
180
181 Для обеих функций используются идентичные параметры декоратора: @jit(nopython
182 =True, nogil=True, cache=True), где:
183 \begin{itemize}
184     \item nopython=True - строгий режим (обязательная компиляция);
185     \item nogil=True - освобождение GIL для многопоточности;
186     \item cache=True - кэширование скомпилированного кода.
187 \end{itemize}
188 \subsection{Архитектура программно-информационной системы}
189
190 На рисунке \ref{DiagramArch:image} представлена многослойная архитектура
191 системы с чётким разделение ответственности между компонентами.
192 \begin{figure}[p] % Разместить на отдельной странице
193     \centering
194     \includegraphics[width=0.8\linewidth]{DiagramArch}
195     \caption{Архитектура программно-информационной системы}
196     \label{DiagramArch:image}
197 \end{figure}
198 \clearpage
199
200 \begin{enumerate}
201     \item Графический интерфейс отображает элементы управления (кнопки,
202         ползунки, вкладки), визуализирует аудиоданные в реальном времени (Форма
203         волны, АЧХ), обрабатывает пользовательские события (нажатия кнопок,
204         изменение параметров).
205     \item Логика процессора отвечает за загрузку/сохранение аудио, чтение
206         файлов через PyDub/FFmpeg, конвертирует в формат float32 для обработки.
207         Применяет эффекты: шумоподавление, эквалайзер (БИХ-фильтры для 8 полос
208         регулировки частот), многополосный компрессор (динамическое сжатие с
209         параметрами), реверберация. Буферизирует данные и синхронизирует позицию
210         воспроизведения.
211     \item На уровне системы происходит ввод и вывод аудио, создаётся отдельный
212         поток для обработки аудио и очереди для передачи данных между потоками.
213         Обрабатываются ошибки, перехватываются исключения и логируются в файл.
214 \end{enumerate}
215
216 \begin{itemize}
217     \item графический интерфейс включает главное окно, состоящее из кнопок
218         управление воспроизведением и загрузки/сохранения, вкладок эффектов (
219             эквалайзер, компрессор и ревербератор), окна отображение визуализации
220             формы волны и АЧХ;
221     \item логика процессора читает и обрабатывает аудиофайлы, применяется
222         выбранные эффекты по очереди, управляет всей работой программы;
223     \item системное устройство воспроизводит звук, работает с файлами,
224         открывает и сохраняет их, управляет несколькими задачами одновременно;
225     \item вспомогательные модули включают частотный анализ, автоматически
226         регулирует громкость и записывает ошибки и события.
227 \end{itemize}
228
229 Все части программы взаимодействуют между собой по чётким правилам. Когда
230 пользователь меняет настройки, интерфейс передаёт их в модуль обработки,

```

который применяет эффекты и отправляет результат на воспроизведение и отображение. Для плавной работы использовались очереди задач для работы в многопоточном режиме, оптимизирует сложные вычисления для быстрой работы.

```
214
215 \subsection{Архитектура приложения}
216
217 Приложение построено по многослойной модульной архитектуре с разделением на
218 логические компоненты, взаимодействующие через четко определенные
219 интерфейсы. В основе лежит ядро обработки аудиосигналов, окруженное
220 графическим интерфейсом, системными сервисами и вспомогательными модулями.
221 Графический интерфейс реализован на Tkinter и включает:
222 \begin{itemize}
223     \item главное окно (MainWindow) с вкладками для управления эффектами,
224         кнопками воспроизведения/паузы и областью визуализации;
225     \item панель эквалайзера (EQPanel) с восьмью ползунками, отображающая АЧХ
226         через Matplotlib;
227     \item панель компрессора (CompressorPanel) для настройки порога, сжатия,
228         атаки, восстановления, характеристики перехода, усиления;
229     \item панель реверберации (ReverbPanel) для настройки эффект, сухой сигнал,
230         размер комнаты, затухание;
231     \item визуализатор спектра (SpectrumAnalyzer), отображающий осциллограмму (
232         форма сигнала) и частотный спектр (БПФ) в реальном времени;
233     \item прогресс-бар с управлением позицией воспроизведения и отображением
234         длительности трека.
235 \end{itemize}
236
237 Логический процессор — центральный модуль, отвечающий за:
238 \begin{itemize}
239     \item загрузку и конвертацию аудио через PyDub (с использованием FFmpeg как
240         бэкенда), включая поддержку WAV, MP3, FLAC;
241     \item эквалайзер на БИХ-фильтрах с частотами 50 Гц, 150 Гц, 250 Гц, 500 Гц,
242         1 кГц, 3 кГц, 7 кГц, 15 кГц;
243     \item компрессор с разделением на три диапазона частот, в каждом из которых
244         есть регулировки: threshold, ratio, knee, attack, rrelease, gain;
245     \item реверберация на основе алгоритма Schroeder (комбинация линий задержки
246         и FIR-фильтров);
247     \item буферизацию данных для плавного воспроизведения и нормализацию уровня
248         сигнала.
249 \end{itemize}
250
251 Системный слой обеспечивает интеграцию с ОС и оборудованием:
252 \begin{itemize}
253     \item аудиопоток (AudioStream) на базе SoundDevice, обрабатывающий ввод/
254         вывод в реальном времени с настройкой latency и sample rate -(44.1192
255         кГц);
256     \item менеджер потоков (ThreadManager) для параллельной обработки (
257         отдельные потоки для: GUI, аудиообработки, визуализации);
258     \item файловый менеджер (FileManager) с кэшированием загруженных треков и
259         поддержкой метаданных (ID3-теги).
260 \end{itemize}
261
262 Архитектура обеспечивает масштабируемость (добавление новых эффектов через
263     модули), производительность (JIT-компиляция, многопоточность) и
264     отказоустойчивость (изоляция сбоев в отдельных потоках).
```

```

245
246 \subsubsection{Многопоточная обработка}
247
248 В коде используется несколько потоков для различных задач
249
250 Основные потоки класса AudioProcessor:
251 \begin{enumerate}
252     \item Поток обработки аудио (processing\_loop). Основной цикл обработки
253         аудиоданных. Выполняет: обрабатывает аудиоданные из очереди input\_queue
254         , применяет эффекты (эквалайзер, компрессор, реверберацию), отправляет
255         обработанные данные в очередь output\_queue, управляет частотой
256         обновления для минимизации нагрузки на CPU.
257     \item Поток пула исполнителей (ThreadPoolExecutor). Выполнение задач в
258         фоновом режиме. Выполняет: обработку аудио в фоне (process\_in\_background), другие тяжелые вычисления без блокировки основного потока.
259     \item Поток обновления позиции воспроизведения. Следит за текущей позицией
260         воспроизведения. Обновляет ползунок и метку времени. Проверяет
261         завершение воспроизведения.
262 \end{enumerate}
263
264 Основные потоки класса AudioApp:
265 \begin{enumerate}
266     \item Поток инициализации приложения (\_background\_init). Инициализирует
267         AudioProcessor. Загружает данные для графиков. Переключается на основной
268         интерфейс после завершения.
269     \item Поток загрузки аудио (\_load\_audio\_in\_thread). Загрузка и
270         конвертация аудиофайлов. Чтение файла. Конвертация в моно и нормализация
271         . Передача данных в основной поток.
272     \item Поток воспроизведения аудио (через sounddevice.OutputStream).
273         Непрерывная передача аудиоданных на звуковую карту. Вызывает \_audio\_callback
274         для получения новых данных. Обрабатывает ошибки устройства.
275         Останавливается при завершении воспроизведения.
276     \item Поток сохранения аудио (\_process\_and\_save\_audio). Сохранение
277         обработанного аудио в файл. Применяет все эффекты к аудио. Конвертирует
278         в выбранный формат. Сохраняет на диск.
279     \item Поток обновления интерфейса (update\_gui). Периодическое обновление
280         графиков и элементов управления. Обновляет форму волны и АЧХ (15 FPS).
281         Обновляет позицию воспроизведения (10 FPS). Проверяет состояние
282         воспроизведения.
283 \end{enumerate}
284
285 Для потокобезопасности используются:
286 \begin{itemize}
287     \item threading.RLock для доступа к аудиоданным;
288     \item queue.Queue для межпоточного обмена данными;
289     \item threading.Event для управления состоянием;
290     \item after в Tkinter для обновления GUI из основного потока.
291 \end{itemize}
292
293 \subsection{Проект данных программно-информационной системы}
294
295 Система оперирует тремя основными категориями данных: входными,
296         промежуточными и выходными. Входные данные включают аудиофайлы в форматах
297         WAV, MP3, FLAC и OGG с поддержкой различных характеристик (битность 16-24

```

бит, частота дискретизации 44.1-192 кГц), а также параметры эффектов, настраиваемые пользователем через графический интерфейс. Для эквалайзера это уровни усиления по восьми полосам (50 Гц, 150 Гц, 250 Гц, 500 Гц, 1 кГц, 3 кГц, 7 кГц, 15 кГц с диапазоном  $\pm 24$  дБ), для каждого диапазона компрессора — порог срабатывания от -60 до 0 дБ, коэффициент компрессии от 1:1 до 10:1, характеристика перехода от 0 до 100, время атаки от 0.1 до 100 мс, время восстановления от 10 до 1000 мс, усиление от -20 до +20 дБ, для реверберации — соотношение обработанного и исходного сигнала, виртуальный размер помещения, затухание, две регулировки среза верхних и нижних частот, панорамирование.

277

278 Промежуточные данные представлены в виде нормализованных аудиобуферов в формате 32-битных чисел с плавающей запятой, организованных как моно- или стереоканальные массивы. Система использует кольцевые буфера для обработки в реальном времени и промежуточные спектральные данные, полученные через быстрое преобразование Фурье с применением оконной функции. Для хранения состояния эффектов используются специализированные структуры: коэффициенты БИХ-фильтров эквалайзера, параметры огибающей компрессора и линии задержки реверберации.

279

280 Выходные данные включают обработанные аудиофайлы в выбранных пользователем форматах (WAV с PCM-кодированием или MP3 с переменным битрейтом), сохраняющие исходные параметры частоты дискретизации или конвертируемые к стандартным значениям. Визуализационные данные содержат три основных компонента: осциллограмму последних обработанных сэмплов, частотный спектр с разрешением 8192 точек и амплитудно-частотную характеристику активных фильтров, отображаемую в логарифмическом масштабе от 20 Гц до 20 кГц.

281

282 Метаданные системы включают пользовательские пресеты эффектов, содержащие полный набор параметров обработки, историю операций для возможности отмены действий, а также технические лог-файлы с информацией о производительности и ошибках. Все данные передаются между компонентами системы через единый формат аудиоблоков, сопровождаемых метаинформацией о текущих настройках обработки, что обеспечивает согласованность работы многопоточной архитектуры.

283

284 \subsubsection{Описание сущностей графического интерфейса}

285

286 Графический интерфейс Audio Processor представляет собой комплекс взаимосвязанных визуальных компонентов, объединенных в единое интуитивное пространство.

287

288 Интерфейс поддерживает несколько режимов отображения - компактный для мониторов с малым разрешением, расширенный с дополнительными инструментами анализа для профессиональной работы, и режим презентации с увеличенными элементами управления. Все визуальные компоненты реализованы с учетом эргономики - важные элементы выделены акцентным цветом, соблюdenы принципы визуальной иерархии, обеспечена последовательная реакция на пользовательские действия. Особенностью интерфейса является синхронизация всех элементов - изменения параметров эффектов мгновенно отражаются на графиках, а действия пользователя сопровождаются тактильной обратной связью в виде тонких анимационных эффектов.

289

290 \subsubsection{Описание сущностей логики процессора}

291

292 Основу обработки составляет низкоуровневый модуль, работающий с потоком аудиосэмплов в формате 32-битных чисел с плавающей точкой, обеспечивающий базовые операции нормализации и передискретизации. Система эффектов построена вокруг трех ключевых процессоров: эквалайзер реализует трехполосную фильтрацию через каскад БИХ-фильтров с настраиваемыми частотами среза и коэффициентами усиления, компрессор отвечает за компрессию сигнала с алгоритмами расчета огибающей и адаптивными параметрами атаки/восстановления, а ревербератор генерирует реверберацию через комбинацию линий задержки с регулируемыми параметрами затухания.

293

294 Поток данных управляется специализированным менеджером маршрутизации, который обеспечивает передачу аудиоблоков между модулями с минимальной задержкой, используя кольцевые буферы и механизм синхронизации для многопоточной работы. Состояние системы отслеживает активные эффекты, параметры обработки и текущий режим работы (реальное время/оффлайн обработка), предоставляя единый интерфейс для управления конвейером обработки.

295

296 \subsubsection{Описание сущностей системного уровня}

297

298 Системная архитектура приложения построена на нескольких ключевых компонентах, обеспечивающих интеграцию с операционной средой и аппаратными ресурсами. Центральным элементом выступает абстрактный слой взаимодействия с аудиодрайверами, реализующий поддержку ASIO, WASAPI и Core Audio через единый кроссплатформенный API. Для управления устройствами ввода-вывода используется DeviceManager, который автоматически обнаруживает доступные аудиоинтерфейсы, анализирует их характеристики и предоставляет унифицированный интерфейс для работы с ними.

299

300 Файловая подсистема основана на компоненте, объединяющем возможности стандартных Python-библиотек для работы с файлами и мощь FFmpeg для обработки мультимедиа. Этот модуль включает кэширующий механизм для ускорения повторного доступа к аудиофайлам и систему контроля целостности данных.

301

302 Многопоточная архитектура координируется центральным диспетчером задач, который создает и управляет тремя основными типами потоков: высокоприоритетным аудиопотоком реального времени, фоновыми рабочими потоками для обработки эффектов и служебными потоками для визуализации и логирования.

303

304 \subsection{Проектирования пользовательского интерфейса}

305

306 Центральное место занимает динамическая визуализация аудиопотока - двойной дисплей с синхронизированными осциллограммой и АЧХ, выполненный в темной цветовой гамме с акцентными элементами салатового цвета для выделения ключевых параметров сигнала. Основная рабочая область: верхняя треть экрана отведена под графики, центральная часть содержит компактные панели эффектов с интуитивными регуляторами, а нижний сектор занимает расширенная панель транспорта с профессиональными элементами управления.

307

308 Навигационная система реализована через адаптивную панель вкладок с контекстно-зависимыми элементами управления - при выборе конкретного эффекта (эквалайзер, компрессор, реверберация) нижняя панель автоматически

наполняется соответствующими регуляторами, сохраняя при этом быстрый доступ к основным функциям. Все элементы управления спроектированы с учетом тактильного взаимодействия - ползунки имеют выраженные рифленые ручки с магнитными точками для часто используемых значений, кнопки обладают трехступенчатой визуальной обратной связью (покой, наведение, нажатие), а переключатели сопровождаются мягкими анимационными переходами.

Автор ВКР

И. О. Матвеев

(подпись, дата)

Руководитель ВКР

И. Н. Ефремова

(подпись, дата)

Нормоконтроль

А. А. Чаплыгин

(подпись, дата)

**Место для диска**